Lecture Notes in Artificial Intelligence 1967
Subseries of Lecture Notes in Computer Science
Edited by J. G. Carbonell and J. Siekmann

Lecture Notes in Computer Science
Edited by G. Goos, J. Hartmanis and J. van Leeuwen
Setsuo Arikawa  Shinichi Morishita (Eds.)

Discovery Science

Third International Conference, DS 2000
Kyoto, Japan, December 4-6, 2000
Proceedings

Springer
Preface

This volume contains 3 invited papers, 15 regular papers, and 22 poster papers that were selected for presentation at the Third International Conference on Discovery Science (DS 2000), which was held 4-6 December 2000 in Kyoto. The Program Committee selected the contributed papers from 48 submissions.

Three distinguished researchers accepted our invitation to present talks: Jeffrey D. Ullman (Stanford University), Joseph Y. Halpern (Cornell University), and Masami Hagiya (University of Tokyo).

The Program Committee would like to thank all those who submitted papers for consideration and the invited speakers. I would like to thank the Program Committee members, the Local Arrangements Committee members, and the Steering Committee members for their splendid and hard work. Finally, special thanks go to the PC Assistant Shoko Suzuki for her assistance in the development of web pages and the preparation of these proceedings.

September 2000
Shinichi Morishita
Organization

Discovery Science 2000 is organized as part of the activities of the Discovery Science Project sponsored by Grant-in-Aid for Scientific Research in the Priority Area from the Ministry of Education, Science, Sports and Culture (MESSC) of Japan, in cooperation with the Japanese Society for Artificial Intelligence, and with SIG of Data Mining, Japan Society for Software Science and Technology.

Conference Chair

Setsuo Arikawa Kyushu University

Program Committee

Shinichi Morishita (Chair) University of Tokyo
Hiroki Arimura Kyushu University
Roberto Bayardo IBM Almaden Research Center
Charles Elkan University of California, San Diego
Peter A. Flach University of Bristol
Koichi Furukawa Keio University
Randy Goebel University of Alberta
Yukiyo Kameyama Kyoto University
Naoki Katoh Kyoto University
Ross D. King University of Wales
Nada Lavrac Jozef Stefan Institute
Katharina Morik University of Dortmund
Hiroshi Motoda Osaka University
Raymond T. Ng University of British Columbia
Koichi Nijima Kyushu University
Claude Sammut University of New South Wales
Taisuke Sato Tokyo Institute of Technology
Etsuya Shibayama Tokyo Institute of Technology
Kyuseok Shim Bell Laboratories
Ayumi Shinohara Kyushu University
Carl H. Smith University of Maryland
Osamu Watanabe Tokyo Institute of Technology
Limsoon Wong Kent Ridge Digital Labs
**Local Arrangements Committee**

Masahiko Sato (Chair)  
Yukiyoshi Kameyama  
Hiroyuki Kawano  
Sadao Kurohashi  
Ayumi Shinohara  
Kyoto University  

**Steering Committee**

Setsuo Arikawa (Chair)  
Yasumasa Kanada  
Akira Maruoka  
Satoru Miyano  
Masahiko Sato  
Taisuke Sato  
Kyushu University  
University of Tokyo  
Tohoku University  
University of Tokyo  
Kyoto University  
Tokyo Institute of Technology
# Table of Contents

## Invited Papers

A Survey of Association-Rule Mining .............................. 1  
*Jeffrey D. Ullman*

Degrees of Belief, Random Worlds, and Maximum Entropy .......... 15  
*Joseph Y. Halpern*

Discovery and Deduction ........................................... 17  
*Masami Hagiya and Koichi Takahashi*

## Regular Papers

Integrating Information Visualization and Retrieval for Discovering Internet Sources ..................................................... 38  
*Hayato Ohwada and Fumio Mizoguchi*

A Unifying Approach to HTML Wrapper Representation and Learning ... 50  
*Gunter Grieser, Klaus P. Jantke, Steffen Lange, and Bernd Thomas*

Discovery of Web Communities Based on the Co-occurrence of References 65  
*Tsuyoshi Murata*

Clustering and Visualization of Large Protein Sequence Databases by Means of an Extension of the Self-Organizing Map .................... 76  
*Panu Somervuo and Teuvo Kohonen*

A Simple Greedy Algorithm for Finding Functional Relations: Efficient Implementation and Average Case Analysis .......................... 86  
*Tatsuya Akutsu, Satoru Miyano, and Satoru Kuhara*

Graph-Based Induction for General Graph Structured Data and Its Application to Chemical Compound Data .............................. 99  
*Takashi Matsuda, Tadashi Horiuchi, Hiroshi Motoda, and Takashi Washio*

Discovering Characteristic Expressions from Literary Works: A New Text Analysis Method beyond N-Gram Statistics and KWIC ........ 112  
*Masayuki Takeda, Tetsuya Matsumoto, Tomoko Fukuda, and Ichirō Nanri*

Classifying Scenarios Using Belief Decision Trees ........................ 127  
*Zied Elouedi and Khaled Mellouli*
<table>
<thead>
<tr>
<th>Title</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Practical Algorithm to Find the Best Subsequence Patterns</td>
<td>141</td>
</tr>
<tr>
<td>Masahiro Hirao, Hiromasa Hoshino, Ayumi Shinohara, Masayuki Takeda,</td>
<td></td>
</tr>
<tr>
<td>and Setsuo Arikawa</td>
<td></td>
</tr>
<tr>
<td>On-Line Estimation of Hidden Markov Model Parameters</td>
<td>155</td>
</tr>
<tr>
<td>Jun Mizuno, Tatsuya Watanabe, Kazuya Ueki, Kazuyuki Amano, Eiji</td>
<td></td>
</tr>
<tr>
<td>Takimoto, and Akira Maruoka</td>
<td></td>
</tr>
<tr>
<td>Computationally Efficient Heuristics for If-Then Rule Extraction from</td>
<td>170</td>
</tr>
<tr>
<td>Freed-Forward Neural Networks</td>
<td></td>
</tr>
<tr>
<td>Hyeoncheol Kim</td>
<td></td>
</tr>
<tr>
<td>Language Learning with a Neighbor System</td>
<td>183</td>
</tr>
<tr>
<td>Yasuhiro Makouchi and Masako Sato</td>
<td></td>
</tr>
<tr>
<td>Application of Multivariate Maxwellian Mixture Model to Plasma</td>
<td>197</td>
</tr>
<tr>
<td>Velocity Distribution Function</td>
<td></td>
</tr>
<tr>
<td>Genta Ueno, Nagatomo Nakamura, Tomoyuki Higuchi, Takashi Tsuchiya,</td>
<td></td>
</tr>
<tr>
<td>Shinobu Machida, and Tohru Araki</td>
<td></td>
</tr>
<tr>
<td>Knowledge Discovery from fMRI Brain Images by Logical Regression</td>
<td>212</td>
</tr>
<tr>
<td>Analysis</td>
<td></td>
</tr>
<tr>
<td>Hiroshi Tsukimoto, Mitsuru Kakimoto, Chie Morita, Yoshiuki Kikuchi,</td>
<td></td>
</tr>
<tr>
<td>Eiko Hatakeyama, and Yoshifumi Miyazaki</td>
<td></td>
</tr>
<tr>
<td>Human Discovery Processes Based on Searching Experiments in Virtual</td>
<td>225</td>
</tr>
<tr>
<td>Psychological Research Environment</td>
<td></td>
</tr>
<tr>
<td>Kazuhisa Miwa</td>
<td></td>
</tr>
<tr>
<td><strong>Poster Papers</strong></td>
<td></td>
</tr>
<tr>
<td>Prediction of Binding Affinities for Protein-ligand Complexes with</td>
<td>240</td>
</tr>
<tr>
<td>Neural Network Models</td>
<td></td>
</tr>
<tr>
<td>Vladimir V. Krepets and Natalya V. Belkina</td>
<td></td>
</tr>
<tr>
<td>Automatic and Accurate Determination of the Onset Time of the Quasi-</td>
<td>242</td>
</tr>
<tr>
<td>periodic Oscillation</td>
<td></td>
</tr>
<tr>
<td>Tomoyuki Higuchi</td>
<td></td>
</tr>
<tr>
<td>The Role of Choice in Discovery</td>
<td>247</td>
</tr>
<tr>
<td>Judith Devaney</td>
<td></td>
</tr>
<tr>
<td>Search for New Methods for Assignment of Complex Molecular Spectra</td>
<td>252</td>
</tr>
<tr>
<td>Takehiko Tanaka</td>
<td></td>
</tr>
<tr>
<td>Computational Analysis for Discovery on the Plasma Waves Observed by</td>
<td>255</td>
</tr>
<tr>
<td>Scientific Satellites</td>
<td></td>
</tr>
<tr>
<td>Yoshiya Kasahara, Ryotaro Niitsu, Yousuke Akimoto, and Toru Sato</td>
<td></td>
</tr>
</tbody>
</table>
Direction Finding of the Waves in Plasma Using Energy Function ........ 258
   Yoshitaka Goto, Yoshiya Kasahara, and Toru Sato

Coping the Challenge of Mutagens Discovery with GUHA+/- for Windows ........................................... 261
   Premysl Zak and Jaroslava Halova

Discovering Interpretable Rules that Explain Customers’ Brand Choice Behavior ......................................................... 263
   Yukinobu Hamuro, Naoki Katoh, and Katsutoshi Yada

Mining for 4ft Association Rules ........................................... 268
   Jan Rauch and Milan Šimůnek

Rule Discovery Technique Using Genetic Programming Combined with Apriori Algorithm ............................................. 273
   Ayahiko Niimi and Eiichiro Tazaki

Discovery of M-of-N Concepts for Classification ........................... 278
   Nobuhiro Yugami

Issues in Organizing a Successful Knowledge Discovery Contest ........ 282
   Einoshin Suzuki

Knowledge Integration of Rule Mining and Schema Discovering ........ 285
   Kohei Maruyama and Kuniaki Uehara

Discovery of Correlation from Multi-stream of Human Motion ........... 290
   Mitsuomi Shimada and Kuniaki Uehara

An Appropriate Abstraction for Constructing a Compact Decision Tree ... 295
   Yoshimitsu Kudoh and Makoto Haraguchi

Extraction Positive and Negative Keywords for Web Communities ...... 299
   Daisuke Ikeda and Sachio Hirokawa

Nonequilibrium Thermodynamics from Time Series Data Analysis ........ 304
   Hiroshi H. Hasegawa, Takashi Washio, Yukari Ishimiya, and Takeshi Saito

Automatic Determination Algorithm for the Optimum Number of States in NL-HMnet ...................................................... 306
   Motoyuki Suzuki and Shozo Makino

Comparative Study of Automatic Acquisition Methods of Image Process-
ing Procedures .......................................................... 311
   Toshihiro Hamada, Akinobu Shimizu, Toyoji Saito, Jun-ichi Hasegawa, and Jun-ichiro Toriwaki
<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extraction of Authors’ Characteristics from Japanese Modern Sentences via N-gram Distribution</td>
<td>315</td>
</tr>
<tr>
<td>Tsukasa Matsuura and Yasumasa Kanada</td>
<td></td>
</tr>
<tr>
<td>Combination Retrieval for Creating Knowledge from Sparse Document Collection</td>
<td>320</td>
</tr>
<tr>
<td>Naohiro Matsumura and Yukio Ohsawa</td>
<td></td>
</tr>
<tr>
<td>Discovery of Nominally Conditioned Polynomials Using Neural Networks, Vector Quantizers and Decision Trees</td>
<td>325</td>
</tr>
<tr>
<td>Kazumi Saito and Ryohei Nakano</td>
<td></td>
</tr>
<tr>
<td><strong>Author Index</strong></td>
<td>331</td>
</tr>
</tbody>
</table>
A Survey of Association-Rule Mining

Jeffrey D. Ullman

Stanford University, Stanford CA 94305 USA

Abstract. The standard model for association-rule mining involves a set of "items" and a set of "baskets." The baskets contain items that some customer has purchased at the same time. The problem is to find pairs, or perhaps larger sets, of items that frequently appear together in baskets. We mention the principal approaches to efficient, large-scale discovery of the frequent itemsets, including the a-priori algorithm, improvements using hashing, and one- and two-pass probabilistic algorithms for finding frequent itemsets. We then turn to techniques for finding highly correlated, but infrequent, pairs of items. These notes were written for CS345 at Stanford University and are reprinted by permission of the author. http://www-db.stanford.edu/~ullman/mining/mining.html gives you access to the entire set of notes, including additional citations and on-line links.

1 Association Rules and Frequent Itemsets

The market-basket problem assumes we have some large number of items, e.g., "bread" or "milk." Customers fill their market baskets with some subset of the items, and we get to know what items people buy together, even if we don’t know who they are. Marketers use this information to position items, and control the way a typical customer traverses the store.

In addition to the marketing application, the same sort of question has the following uses:

1. Baskets = documents; items = words. Words appearing frequently together in documents may represent phrases or linked concepts. One possible application is intelligence gathering.

2. Baskets = sentences, items = documents. Two documents with many of the same sentences could represent plagiarism or mirror sites on the Web.

1.1 Goals for Market-Basket Mining

1. Association rules are statements of the form \{X_1, X_2, \ldots, X_n\} \rightarrow Y, meaning that if we find all of \(X_1, X_2, \ldots, X_n\) in the market basket, then we have a good chance of finding \(Y\). The probability of finding \(Y\) given \(X_1, \ldots, X_n\) is called the confidence of the rule. We normally would accept only rules that had confidence above a certain threshold. We may also ask that the confidence be significantly higher than it would be if items were placed at random.
into baskets. For example, we might find a rule like \{milk, butter\} \Rightarrow bread simply because a lot of people buy bread. However, the beer/diapers story\(^1\) asserts that the rule \{diapers\} \Rightarrow beer holds with confidence significantly greater than the fraction of baskets that contain beer.

2. Causality. Ideally, we would like to know that in an association rule the presence of \(X_1, \ldots, X_n\) actually "causes" \(Y\) to be bought. However, "causality" is an elusive concept. Nevertheless, for market-basket data, the following test suggests what causality means. If we lower the price of diapers and raise the price of beer, we can lure diaper buyers, who are more likely to pick up beer while in the store, thus covering our losses on the diapers. That strategy works because "diapers causes beer." However, working it the other way round, running a sale on beer and raising the price of diapers, will not result in beer buyers buying diapers in any great numbers, and we lose money.

3. Frequent itemsets. In many (but not all) situations, we only care about association rules or causalities involving sets of items that appear frequently in baskets. For example, we cannot run a good marketing strategy involving items that almost no one buys anyway. Thus, much data mining starts with the assumption that we only care about sets of items with high support; i.e., they appear together in many baskets. We then find association rules or causalities only involving a high-support set of items (i.e., \(X_1, \ldots, X_n, Y\)) must appear in at least a certain percent of the baskets, called the support threshold.

1.2 Framework for Frequent Itemset Mining

We use the term frequent itemset for "a set \(S\) that appears in at least fraction \(s\) of the baskets," where \(s\) is some chosen constant, typically 0.01 or 1%.

We assume data is too large to fit in main memory. Either it is stored in a relational database, say as a relation \(\text{Basket}(BID, item)\) or as a flat file of records of the form \((BID, item1, item2, \ldots, itemn)\). When evaluating the running time of algorithms we:

- Count the number of passes through the data. Since the principal cost is often the time it takes to read data from disk, the number of times we need to read each datum is often the best measure of running time of the algorithm.

- There is a key principle, called monotonicity or the a-priori trick, that helps us find frequent itemsets:

If a set of items \(S\) is frequent (i.e., appears in at least fraction \(s\) of the baskets), then every subset of \(S\) is also frequent.

\(^1\)The famous, and possibly apocryphal discovery that people who buy diapers are unusually likely to buy beer.
Put in the contrapositive: a set $S$ cannot be frequent unless all its subsets are.

To find frequent itemsets, we can:

1. Proceed levelwise, finding first the frequent items (sets of size 1), then the frequent pairs, the frequent triples, etc. In our discussion, we concentrate on finding frequent pairs because:
   (a) Often, pairs are enough.
   (b) In many data sets, the hardest part is finding the pairs; proceeding to higher levels takes less time than finding frequent pairs.
   Levelwise algorithms use one pass per level.

2. Find all maximal frequent itemsets (i.e., sets $S$ of any size, such that no proper superset of $S$ is frequent) in one pass or a few passes.

### 1.3 The A-Priori Algorithm

The following is taken from [1], [2]. The algorithm called $a$-priori proceeds levelwise.

1. Given support threshold $s$, in the first pass we find the items that appear in at least fraction $s$ of the baskets. This set is called $L_1$, the frequent items. Presumably there is enough main memory to count occurrences of each item, since a typical store sells no more than 100,000 different items.

2. Pairs of items in $L_1$ become the candidate pairs $C_2$ for the second pass. We hope that the size of $C_2$ is not so large that there is not room in main memory for an integer count per candidate pair. The pairs in $C_2$ whose count reaches $s$ are the frequent pairs, $L_2$.

3. The candidate triples, $C_3$ are those sets $\{A, B, C\}$ such that all of $\{A, B\}$, $\{A, C\}$, and $\{B, C\}$ are in $L_2$. On the third pass, count the occurrences of triples in $C_3$; those with a count of at least $s$ are the frequent triples, $L_3$.

4. Proceed as far as you like (or the sets become empty). $L_i$ is the frequent sets of size $i$; $C_{i+1}$ is the set of sets of size $i + 1$ such that each subset of size $i$ is in $L_i$.

### 1.4 Why A-Priori Helps

Consider the following SQL on a $Baskets(BID, item)$ relation with $10^8$ tuples involving $10^7$ baskets of 10 items each; assume 100,000 different items (typical of Wal-Mart, e.g.).
SELECT b1.item, b2.item, COUNT(*)
FROM Baskets b1, Baskets b2
WHERE b1.BID = b2.BID AND b1.item < b2.item
GROUP BY b1.item, b2.item
HAVING COUNT(*) >= s;

Note: s is the support threshold, and the second term of the WHERE clause is to prevent pairs of items that are really one item, and to prevent pairs from appearing twice.

In the join Baskets × Baskets, each basket contributes \( \binom{10}{2} \approx 45 \) pairs, so the join has \( 4.5 \times 10^8 \) tuples.

A-priori “pushes the HAVING down the expression tree,” causing us first to replace Baskets by the result of

```
SELECT *
FROM Baskets
GROUP by item
HAVING COUNT(*) >= s;
```

If \( s = 0.01 \), then at most 1000 items’ groups can pass the HAVING condition. Reason: there are \( 10^8 \) item occurrences, and an item needs \( 0.01 \times 10^7 = 10^6 \) of those to appear in \( 1\% \) of the baskets.

Although 99% of the items are thrown away by a-priori, we should not assume the resulting Baskets relation has only \( 10^6 \) tuples. In fact, all the tuples may be for the high-support items. However, in real situations, the shrinkage in Baskets is substantial, and the size of the join shrinks in proportion to the square of the shrinkage in Baskets.

1.5 Improvements to A-Priori

Two types:

1. Cut down the size of the candidate sets \( C_i \) for \( i \geq 2 \). This option is important, even for finding frequent pairs, since the number of candidates must be sufficiently small that a count for each can fit in main memory.

2. Merge the attempts to find \( L_1, L_2, L_3, \ldots \) into one or two passes, rather than a pass per level.

1.6 PCY Algorithm

Park, Chen, and Yu [5] proposed using a hash table to determine on the first pass (while \( L_1 \) is being determined) that many pairs are not possibly frequent. PCY takes advantage of the fact that main memory is usually much bigger than the number of items. During the two passes to find \( L_2 \), the main memory is laid out as in Fig. 1.

Assume that data is stored as a flat file, with records consisting of a basket ID and a list of its items.
1. Pass 1:
   (a) Count occurrences of all items.

   (b) For each bucket, consisting of items \{i_1, \ldots, i_k\}, hash each pair to a bucket of the hash table, and increment the count of the bucket by 1.

   (c) At the end of the pass, determine \(L_1\), the items with counts at least \(s\).

   (d) Also at the end, determine those buckets with counts at least \(s\).

   Key point: A pair \((i, j)\) cannot be frequent unless it hashes to a frequent bucket, so pairs that hash to other buckets need not be candidates in \(C_2\).

   Replace the hash table by a bitmap, with one bit per bucket: 1 if the bucket was frequent, 0 if not.

2. Pass 2:
   (a) Main memory holds a list of all the frequent items, i.e. \(L_1\).

   (b) Main memory also holds the bitmap summarizing the results of the hashing from pass 1.

   Key point: The buckets must use 16 or 32 bits for a count, but these are compressed to 1 bit. Thus, even if the hash table occupied almost the entire main memory on pass 1, its bitmap occupies no more than \(1/16\) of main memory on pass 2.

   (c) Finally, main memory also holds a table with all the candidate pairs and their counts. A pair \((i, j)\) can be a candidate in \(C_2\) only if all of the following are true:
i. $i$ is in $L_1$.

ii. $j$ is in $L_1$.

iii. $(i, j)$ hashes to a frequent bucket.

It is the last condition that distinguishes PCY from straight a-priori and reduces the requirements for memory in pass 2.

During pass 2, we consider each basket, and each pair of its items, making the test outlined above. If a pair meets all three conditions, add to its count in memory, or create an entry for it if one does not yet exist.

When does PCY beat a-priori? When there are too many pairs of items from $L_1$ to fit a table of candidate pairs and their counts in main memory, yet the number of frequent buckets in the PCY algorithm is sufficiently small that it reduces the size of $C_2$ below what can fit in memory (even with 1/16 of it given over to the bitmap).

When will most of the buckets be infrequent in PCY? When there are a few frequent pairs, but most pairs are so infrequent that even when the counts of all the pairs that hash to a given bucket are added, they still are unlikely to sum to $s$ or more.

1.7 The “Iceberg” Extensions to PCY

The following is taken from [4].

1. **Multiple hash tables**: share memory between two or more hash tables on pass 1, as in Fig. 2. On pass 2, a bitmap is stored for each hash table; note that the space needed for all these bitmaps is exactly the same as what is needed for the one bitmap in PCY, since the total number of buckets represented is the same. In order to be a candidate in $C_2$, a pair must:

   (a) Consist of items from $L_1$, and

   (b) Hash to a frequent bucket in every hash table.

2. **Iterated hash tables (Multistage)**: Instead of checking candidates in pass 2, we run another hash table (different hash function!) in pass 2, but we only hash those pairs that meet the test of PCY; i.e., they are both from $L_1$ and hashed to a frequent bucket on pass 1. On the third pass, we keep bitmaps from both hash tables, and treat a pair as a candidate in $C_2$ only if:

   (a) Both items are in $L_1$.

   (b) The pair hashed to a frequent bucket on pass 1.

   (c) The pair also was hashed to a frequent bucket on pass 2.
A Survey of Association-Rule Mining

Count items  Frequent items

<table>
<thead>
<tr>
<th>Hash table 1</th>
<th>Frequent items</th>
<th>Bitmaps</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hash table 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

Pass 1  Pass 2

Fig. 2. Multiple hash tables memory utilization

Count items  Frequent items  Frequent items

<table>
<thead>
<tr>
<th>Hash table</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bitmap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Another hash table</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bitmap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bitmap</td>
</tr>
</tbody>
</table>

Counts for candidate pairs

Pass 1  Pass 2  Pass 3

Fig. 3. Multistage hash tables memory utilization

Figure 3 suggests the use of memory. This scheme could be extended to more passes, but there is a limit, because eventually the memory becomes full of bitmaps, and we can’t count any candidates.

When does multiple hash tables help? When most buckets on the first pass of PCY have counts way below the threshold s. Then, we can double the counts in buckets and still have most buckets below threshold.

When does multistage help? When the number of frequent buckets on the first pass is high (e.g., 50%), but not all buckets. Then, a second hashing
with some of the pairs ignored may reduce the number of frequent buckets significantly.

1.8 All Frequent Itemsets in Two Passes

The methods above are best when you only want frequent pairs, a common case. If we want all maximal frequent itemsets, including large sets, too many passes may be needed. There are several approaches to getting all frequent itemsets in two passes or less. They each rely on randomness of data in some way.

1. *Simple approach*: Take a main-memory-sized sample of the data. Run a levelwise algorithm in main memory (so you don’t have to pay for disk I/O), and hope that the sample will give you the truly frequent sets.

   Note that you must scale the threshold $s$ back; e.g., if your sample is 1% of the data, use $s/100$ as your support threshold.

   You can make a complete pass through the data to verify that the frequent itemsets of the sample are truly frequent, but you will miss a set that is frequent in the whole data but not in the sample.

   To minimize false negatives, you can lower the threshold a bit in the sample, thus finding more candidates for the full pass through the data. Risk: you will have too many candidates to fit in main memory.

2. *The SON approach* [6]: Read subsets of the data into main memory, and apply the “simple approach” to discover candidate sets. Every basket is part of one such main-memory subset. On the second pass, a set is a candidate if it was identified as a candidate in any one or more of the subsets.

   Key point: A set cannot be frequent in the entire data unless it is frequent in at least one subset.

3. *Toivonen’s Algorithm* [7]:

   (a) Take a sample that fits in main memory. Run the simple approach on this data, but with a threshold lowered so that we are unlikely to miss any truly frequent itemsets (e.g., if sample is 1% of the data, use $s/125$ as the support threshold).

   (b) Add to the candidates of the sample the negative border: those sets of items $S$ such that $S$ is not identified as frequent in the sample, but every immediate subset of $S$ is. For example, if $ABCD$ is not frequent in the sample, but all of $ABC$, $ABD$, $ACD$, and $BCD$ are frequent in the sample, then $ABCD$ is in the negative border.

   (c) Make a pass over the data, counting all the candidate itemsets and the negative border. If no member of the negative border is frequent in the full data, then the frequent itemsets are exactly those candidates that are above threshold.
Unfortunately, if there is a member of the negative border that turns out to be frequent, then we don’t know whether some of its supersets are also frequent, so the whole process needs to be repeated (or we accept what we have and don’t worry about a few false negatives).

2 Low-Support, High-Correlation Mining

The following material is taken from [3]. We continue to assume a “market-basket” model for data, and we visualize the data as a boolean matrix, where rows = baskets and columns = items. Key assumptions:

1. Matrix is very sparse; almost all 0’s.

2. The number of columns (items) is sufficiently small that we can store something per column in main memory, but sufficiently large that we cannot store something per pair of items in main memory (the same assumption we’ve made in all association-rule work so far).

3. The number of rows is so large that we cannot store the entire matrix in memory, even if we take advantage of sparseness and compress (again, same assumption as always).

4. We are not interested in high-support pairs or sets of columns; rather we want highly correlated pairs of columns.

2.1 Applications

While marketing applications generally care only about high support (it doesn’t pay to try to market things that nobody buys anyway), there are several applications that meet the model above, especially the point about pairs of columns or items with low support but high correlation being interesting:

1. Rows and columns are Web pages; \((r, c) = 1\) means that the page of row \(r\) links to the page of column \(c\). Similar columns may be pages about the same topic.

2. Same as (1), but the page of column \(c\) links to the page of row \(r\). Now, similar columns may represent mirror pages.

3. Rows = sentences of Web pages or documents; columns = words appearing in those sentences. Similar columns are words that appear almost always together, e.g., “phrases.”

4. Rows = sentences; columns = Web pages or documents containing those sentences. Similar columns may indicate mirror pages or plagiarisms.
2.2 Similarity

Think of a column as the set of rows in which the column has a 1. Then the similarity of two columns $C_1$ and $C_2$ is $Sim(C_1, C_2) = |C_1 \cap C_2|/|C_1 \cup C_2|$. 

\[
\begin{align*}
0 & 1 \\
1 & 0 \\
\end{align*}
\]

Example 1. $1 1 = 2/5 = 40\%$ similar

\[
\begin{align*}
0 & 0 \\
1 & 1 \\
0 & 1 \\
\end{align*}
\]

2.3 Signatures

Key idea: map ("hash") each column $C$ to a small amount of data [the signature, $Sig(C)$] such that:

1. $Sig(C)$ is small enough that a signature for each column can be fit in main memory.

2. Columns $C_1$ and $C_2$ are highly similar if and only if $Sig(C_1)$ and $Sig(C_2)$ are highly similar. (But note that we need to define "similarity" for signatures.)

An idea that doesn’t work: Pick 100 rows at random, and make that string of 100 bits be the signature for each column. The reason is that the matrix is assumed sparse, so many columns will have an all-0 signature even if they are quite dissimilar.

Useful convention: given two columns $C_1$ and $C_2$, we’ll refer to rows as being of four types $a$, $b$, $c$, $d$ depending on their bits in these columns, as follows:

<table>
<thead>
<tr>
<th>Type</th>
<th>$C_1$</th>
<th>$C_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$b$</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$c$</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$d$</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

We’ll also use $a$ as "the number of rows of type $a$," and so on.

Note, $Sim(C_1, C_2) = a/(a + b + c)$.

But since most rows are of type $d$, a selection of, say, 100 random rows will be all of type $d$, so the similarity of the columns in these 100 rows is not even defined.
2.4 Min Hashing

Imagine the rows permuted randomly in order. “Hash” each column \( C \) to \( h(C) \), the number of the first row in which column \( C \) has a 1.

The probability that \( h(C_1) = h(C_2) \) is \( a/(a + b + c) \), since the hash values agree if the first row with a 1 in either column is of type \( a \), and they disagree if the first such row is of type \( b \) or \( c \). Note this probability is the same as \( Sim(C_1, C_2) \).

If we repeat the experiment, with a new permutation of rows a large number of times, say 100, we get a signature consisting of 100 row numbers for each column. The “similarity” of these lists (fraction of positions in which they agree) will be very close to the similarity of the columns.

Important trick: we don’t actually permute the rows, which would take many passes over the entire data. Rather, we read the rows in whatever order they are given, and hash each row using (say) 100 different hash functions. For each column we maintain the lowest hash value of a row in which that column has a 1, independently for each of the 100 hash functions. After considering all rows, we shall have for each column the first rows in which the column has 1, if the rows had been permuted in the orders given by each of the 100 hash functions.

2.5 Locality-Sensitive Hashing

Problem: we’ve got signatures for all the columns in main memory, and similar signatures mean similar columns, with high probability, but there still may be so many columns that doing anything that is quadratic in the number of columns, even in main memory, is prohibitive. Locality-sensitive hashing (LSH) is a technique to be used in main memory for approximating the set of similar column-pairs with a lot less than quadratic work.

The goal: in time proportional to the number of columns, eliminate as possible similar pairs the vast majority of the column pairs.

1. Think of the signatures as columns of integers.
2. Partition the rows of the signatures into bands, say \( l \) bands of \( r \) rows each.
3. Hash the columns in each band into buckets. A pair of columns is a candidate-pair if they hash to the same bucket in any band.
4. After identifying candidates, verify each candidate-pair \( (C_i, C_j) \) by examining \( Sig(C_i) \) and \( Sig(C_j) \) for similarity.

Example 2. To see the effect of LSH, consider data with 100,000 columns, and signatures consisting of 100 integers each. The signatures take 40Mb of memory, not too much by today’s standards. Suppose we want pairs that are 80% similar. We’ll look at the signatures, rather than the columns, so we are really identifying columns whose signatures are 80% similar — not quite the same thing.
If two columns are 80% similar, then the probability that they are identical in any one band of 5 integers is \((0.8)^5 = 0.328\). The probability that they are not similar in any of the 20 bands is \((1 - 0.328)^{20} = 0.00035\). Thus, all but about 1/3000 of the pairs with 80%-similar signatures will be identified as candidates.

Now, suppose two columns are only 40% similar. Then the probability that they are identical in one band is \((0.4)^5 = 0.01\), and the probability that they are similar in at least one of the 20 bands is no more than 0.2. Thus, we can skip at least 4/5 of the pairs that will turn out not to be candidates, if 40% is the typical similarity of columns.

In fact, most pairs of columns will be a lot less than 40% similar, so we really eliminate a huge fraction of the dissimilar columns.

### 2.6 k-Min Hashing

Min hashing requires that we hash each row number \(k\) times, if we want a signature of \(k\) integers. With \(k\)-min hashing, in \(k\)-min hashing we instead hash each row once, and for each column, we take the \(k\) lowest-numbered rows in which that column has a 1 as the signature.

To see why the similarity of these signatures is almost the same as the similarity of the columns from which they are derived, examine Fig. 4. This figure represents the signatures \(\text{Sig}_1\) and \(\text{Sig}_2\) for columns \(C_1\) and \(C_2\), respectively, as if the rows were permuted in the order of their hash values, and rows of type \(d\) (neither column has 1) are omitted. Thus, we see only rows of types \(a, b,\) and \(c,\) and we indicate that a row is in the signature by a 1.

\[
\begin{pmatrix}
1 & 1 \\
1 & 0 \\
1 & 1 \\
0 & 1 \\
. & . \\
. & . \\
. & . \\
1 & 1 \\
1 & 0 \\
1 & 1
\end{pmatrix}
\]

\[100\text{ 1's}\]

Fig. 4. Example of the signatures of two columns using \(k\)-min hashing.
Let us assume $c \geq k$, so the typical situation (assuming $k = 100$) is as shown in Fig. 4: the top 100 rows in the first column includes some rows that are not among the top 100 rows for the second column. Then an estimate of the similarity of $\text{Sig}_1$ and $\text{Sig}_2$ can be computed as follows:

$$|\text{Sig}_1 \cap \text{Sig}_2| = \frac{100a}{a+c}$$

because on average, the fraction of the 100 top rows of $C_2$ that are also rows of $C_1$ is $a/(a+c)$. Also:

$$|\text{Sig}_1 \cup \text{Sig}_2| = 100 + \frac{100c}{a+c}$$

The argument is that all 100 rows of $\text{Sig}_1$ are in the union. In addition, those rows of $\text{Sig}_2$ that are not rows of $\text{Sig}_1$ are in the union, and the latter set of rows is on average $100c/(a+c)$ rows. Thus, the similarity of $\text{Sig}_1$ and $\text{Sig}_2$ is:

$$\frac{|\text{Sig}_1 \cap \text{Sig}_2|}{|\text{Sig}_1 \cup \text{Sig}_2|} = \frac{\frac{100a}{a+c}}{100 + \frac{100c}{a+c}} = \frac{a}{a+2c}$$

Note that if $c$ is close to $b$, then the similarity of the signatures is close to the similarity of the columns, which is $a/(a+b+c)$. In fact, if the columns are very similar, then $b$ and $c$ are both small compared to $a$, and the similarities of the signatures and columns must be close.

### 2.7 Amplification of 1’s (Hamming LSH)

If columns are not sparse, but have about 50% 1’s, then we don’t need mini-hashing; a random collection of rows serves as a signature. *Hamming LSH* constructs a series of matrices, each with half as many rows as the previous, by OR-ing together two consecutive rows from the previous, as in Fig. 5.

There are no more than $\log n$ matrices if $n$ is the number of rows. The total number of rows in all matrices is $2n$, and they can all be computed with one pass through the original matrix, storing the large ones on disk.

In each matrix, produce as *candidate pairs* those columns that:

1. Have a medium density of 1’s, say between 20% and 80%, and
2. Are likely to be similar, based on an LSH test.

Note that the density range 20–80% guarantees that any two columns that are at least 50% similar will be considered together in at least one matrix, unless by bad luck their relative densities change due to the OR operation combining two 1’s into one.

A second pass through the original data confirms which of the candidates are really similar.
Fig. 5. Construction of a series of exponentially smaller, denser matrices

This method exploits an idea that can be useful elsewhere: similar columns have similar numbers of 1's, so there is no point ever comparing columns whose numbers of 1's are very different.

References

Degrees of belief, random worlds, and maximum entropy

Joseph Y. Halpern
Dept. of Computer Science
Cornell University
Ithaca, NY 14853.

Abstract. Consider a doctor with a knowledge base KB consisting of first-order information (such as “All patients with hepatitis have jaundice”), statistical information (such as “80% have hepatitis”), and default information (such as “patients with pneumonia typically have fever”). The doctor may want to make decisions regarding a particular patient, using the KB in some principled way. To do this, it is often useful for the doctor to assign a numerical “degree of belief” to measure the strength of her belief in a given statement A. I focus on one principled method for doing so. The method, called the random worlds method, is a natural one: For any given domain size N, we can look at the proportion of models satisfying A among models of size N satisfying KB. If we don’t know the domain size N, but know that it is large, we can approximate the degree of belief in A given KB by taking the limit of this fraction as N goes to infinity.

In many cases that arise in practice, the answers we get using this method can be shown to match heuristic assumptions made in many standard AI systems. I also show that when the language is restricted to unary predicates (for example, symptoms and diseases, but not relations such as “Taller than”), the answer provided by the random worlds method can often be computed using maximum entropy. On the other hand, if the language includes binary predicates, all connections to maximum entropy seem to disappear. Moreover, almost all the questions one might want to ask can be shown to be highly undecidable.

I conclude with some general discussion of the problem of finding reasonable methods to do inductive reasoning of the sort considered here, and the relevance of these ideas to data mining and knowledge discovery.

The talk covers joint work with Faaheem Bacchus, Adam Grove and Daphne Koller [1,2].

References


© Springer-Verlag Berlin Heidelberg 2000


Discovery and Deduction

Masami Hagiya and Keichi Takahashi

University of Tokyo and Electrotechnical Laboratories
hagiya@is.s.u-tokyo.ac.jp
http://nicosia.is.s.u-tokyo.ac.jp/members/hagiya.html

Abstract. Deduction is usually considered to be the opposite of induction. However, deduction and induction can be related in many ways. In this paper, two endeavors that try to relate discovery science and verification technology are described. The first is discovery by deduction, where attempts to find algorithms are made using verifiers. Case studies of finding algorithms for concurrent garbage collection and for mutual exclusion without semaphores are described. Superoptimization can also be classified as work in this field. Recent work on finding authentication protocols using a protocol verifier is also briefly surveyed.

The second endeavor is discovery for deduction. This concerns the long-standing problem of finding induction formulae or loop invariants. The problem is regarded as one of learning from positive data, and the notion of safe generalization, which is commonly recognized in learning from positive data, is introduced into iterative computation of loop invariants. The similarity between the widening operator in abstract interpretation and Gold's notion of identification in the limit is also discussed.

1 Introduction

Deduction is usually considered the opposite of induction. For example, inductive inference of a function means to guess the definition (program) of the function from examples of inputs and outputs. On the other hand, deductive inference means to derive a theorem by applying inference rules to axioms. However, deduction and induction can be related in many ways. In this paper, we describe two endeavors that try to relate discovery science and verification technology.

An older endeavor, deductive approach to program synthesis [20], aimed at synthesizing programs under a deductive framework. This technique is also known as constructive programming [24, 13, 16]. In this approach, the specification of a program that is being synthesized is given as a formula using formal logic. The formula takes the form, \( \forall x. \exists y. P(x, y) \), where \( x \) denotes an input and \( y \) an output. The predicate, \( P(x, y) \), is called the input-output relation, and specifies the condition that the output \( y \) should satisfy with respect to the input \( x \). The formula itself is called the existence theorem.

In the deductive approach, the existence theorem is proved constructively. It is not necessary for the underlying logic to be constructive, but if the theorem is proved under constructive logic, the proof of \( \forall x. \exists y. P(x, y) \) is guaranteed to be
constructive. From a constructive proof of $\forall x.\exists y. P(x, y)$, it is possible to extract a function, $f$, that satisfies $\forall x. P(x, f(x))$. Since the function is represented as a term in formal logic, it can be regarded as a program if the symbols in the term are interpreted appropriately. A program is thus obtained from its specification.

This approach, although it elegantly explicates the relationship between programs and proofs, is practically infeasible, unless a very powerful theorem prover for the underlying logic is available. Since such a prover does not exist in general, human assistance is required to prove the existence theorem. If human assistance is required at all, then it is not a machine but a human who finds a program because it becomes almost obvious for a human to conceive the underlying program while he or she writes the proof. This approach, therefore, cannot be considered as a method for machine discovery of programs, but as a method for human discovery. Proof animation, recently advocated Hayashi, is more explicit in this direction [17].

In this paper, we describe another approach to synthesizing programs or algorithms under a deductive framework. Model checking is a methodology for automatically verifying a program by exploring its state space [6]. It has been established as a method for automatically verifying hardware, protocols, and software (not concrete programs, but algorithms or designs at abstract levels). Automatic verifiers employed in model checking are called model checkers. They explore the state space constructed from a given program and its specification expressed using temporal logic.

In order to synthesize a program using a model checker, a space of programs must first be defined to which the target program is expected to belong. There are many ways to define such a space. It can be defined by a set of parameters. A program belonging to the space is then defined as a tuple of the values of the parameters. Programs can also be written in a programming language. In this case, the space consists of programs expressed in the language. Since such a space is infinite in general, it is usually necessary to impose restrictions on the size of programs.

An automatic verifer, i.e., model checker, is then invoked on each program in the program space to try to verify the specification that the program should satisfy. The verifier determines, within a finite time, whether the specification is satisfied or not. Therefore, it is possible to search for a program that satisfies the specification by simply searching through the program space.

In this paper, this approach is used to find algorithms for concurrent garbage collection and for mutual exclusion without semaphores. These case studies are described in detail in the next section.

Superoptimization, which is a method to automatically synthesize the code generation table of a compiler, can also be classified as an attempt in this research direction [21, 12, 10]. It searches for a sequence of machine code that corresponds to an operation in an intermediate language.

In this paper, Perrig and Song’s recent work on finding authentication protocols using a protocol verifier is also briefly surveyed [23]. It is an attempt to synthesize an authentication protocol using a protocol verifier, called Athena [25].
This work is important for two reasons. One is that authentication protocols are very short in general. Since each protocol consists of only a few message exchanges, the protocol space to explore is of a feasible size. The other is that the correctness of an authentication protocol is very subtle and requires machine verification. Since it is not easy for a human to check the correctness of a protocol, it is also difficult to find a correct one.

The second part of this paper describes the relationship between inductive theorem proving and learning from positive data. Inductive theorem proving is to automatically prove a theorem by mathematical induction. In the field of automated deduction, this is the long standing problem of finding induction formulas or loop invariants. In order to automate mathematical induction, it is necessary to automatically synthesize appropriate induction formulae employed in mathematical induction. In particular, the correctness of a loop program requires a formula, called the loop invariant, which always holds whenever the loop body is executed. The correctness is proved by mathematical induction using the loop invariant as the induction formula.

If we execute a loop program with concrete input, we obtain a sequence of concrete states that the program takes at the beginning of the loop body. If we can obtain a general representation of the concrete states, then this representation can be used as the loop invariant. This problem can be considered as that of learning from positive data, because we only have concrete states that are positive examples of the general representation.

In learning from positive data, it is crucial to avoid over-generalization. Many generalization procedures have been proposed that avoid over-generalization and run in polynomial time. In this paper, the theory of learning from positive data is applied, and the notion of safe generalization is introduced into the iterative computation of a loop invariant. This means that iterative computation always avoids over-generalization, and if it ever converges, it yields the exact loop invariant.

If the domain is infinite, however, iterative computation does not always converge in a finite number of steps. In the field of abstract interpretation, the technique called widening is used to guarantee finite convergence. Therefore, the widening operator in abstract interpretation and Gold’s notion of identification in the limit show close similarity.

2 Discovery by Deduction — Searching for Algorithms by Automatic Verifiers

Model checking is a method used to exhaustively search through a state space to verify that error or starvation states are not reachable from an initial state [6].

A brief explanation of model checking follows.

Any computer system, whether it is hardware or software, can be modeled as a state transition system, which consists of a set of states and the transition relation between these states. A state in this context means a state that the entire computer system takes at a certain point of time. If a system is comprised of
multiple processes, a state of the entire system is a tuple of states, each taken by
an individual process. The transition relation is a binary relation between states
that determines, for each state, which states can follow it. A state transition
system, therefore, comprises a directed graph whose nodes are states and whose
edges represent the transition relation.

There are two kinds of properties that should be verified with respect to a
state transition system: safety and liveness properties. To verify a safety property
is to repeat state transitions from an initial state and ensure that error states
are not reachable. This is reduced to a reachability problem in a directed graph.
To verify a liveness property is to check whether starvation of a certain kind may
occur under some fairness conditions. If a system is comprised of two processes,
fairness roughly means that both processes are executed equally often, i.e., they
are scheduled with an equal chance. There are many kinds of fairness conditions,
though, and care must be taken as to what kind of fairness should be assumed
on a particular system. Starvation means that a process is blocked infinitely, to
wait for some resources to be released, under the assumption that fairness holds.
If starvation never occurs, the system is said to satisfy liveness.

The detection of starvation is more difficult than that of error states, because
starvation is not a property of a single state. A search must be made for an
infinite execution path that satisfies both starvation and fairness. If the state
space is finite, this problem is reduced to that of finding a loop that satisfies
certain properties in a directed graph.

In this paper, we are interested in using verifiers to find algorithms. We define
a space of algorithms, and search through it to find an algorithm that satisfies
safety and/or liveness.

In the rest of this section, attempts made to find algorithms in the following
fields are described:

- concurrent garbage collection,
- mutual exclusion without semaphores,
- superoptimization,
- authentication protocols.

2.1 Concurrent Garbage Collection

The first case study using model checking is finding algorithms for concurrent
garbage collection. Such an algorithm is comprised of two processes: a collector
that collects garbage cells, and a mutator that manipulates cell pointers and does
some computations. Algorithms such as on-the-fly [9] and snapshot [27, 28] are
well known for concurrent garbage collection. Although these two algorithms are
based on completely different ideas, they can be modeled in a uniform framework.
For example, both employ four cell colors: white, black, gray, and free. The color
“free” means that a cell is free, i.e., allocatable.

In our framework, the mutator refers to some registers, each holding a pointer
to a cell. The collector also refers to the registers when it marks the cells in use.
Since the collector begins marking with the pointers held in the registers, a register is also called a \textit{root}.

In this study, we assume that each cell has only one field to simplify the framework. (Fortunately, this restriction turned out to be irrelevant for the algorithms discovered.) Figure 1 shows registers and heaps.

![Fig. 1. Registers and cells.](image)

The collector takes the following four steps.

- **shade:** The collector makes all cells that are directly reachable from a register gray. This step is executed with the mutator stopped.
- **mark:** The collector selects a gray cell and makes it black. If it refers to another cell which is white, then the white cell is made gray. If there is no gray cell, the collector goes to the next step. This step is executed concurrently with the mutator.
- **append:** The collector selects a white cell and makes it free. If there is no white cell, the collector goes to the next step. This step is executed concurrently with the mutator.
- **unmark:** The collector selects a black or gray cell and makes it white. If there is no black or gray cell, the collector goes back to the first step, shade. This step is executed concurrently with the mutator.

The mutators used in the on-the-fly and snapshot algorithms have different operations. In this case study, thirteen operations are defined that cover both algorithms. In the following description, R[i] denotes the contents of the i-th register, and F[i] denotes the contents of the field of the i-th cell. Indices of cells begin with 1, and the index 0 denotes nil (the null pointer). The procedure, \texttt{white\rightarrow gray(i)}, makes the white i-th cell gray, and i is not 0.

(0) Allocate a new cell in the shade step.
(1) Allocate a new cell in the mark step.
(2) Allocate a new cell in the append step.
(3) Allocate a new cell in the unmark step.
(4) Make a newly allocated cell gray.
(5) Make a newly allocated cell black.
(6) \texttt{white\_to\_gray}(F[R[i]]); F[R[i]] := 0;
(7) F[R[i]] := 0;
(8) R[i] := F[R[j]]; \texttt{white\_to\_gray}(R[i]);
(9) R[i] := F[R[j]];
(10) F[R[i]] := R[j]; \texttt{white\_to\_gray}(F[R[i]]);
(11) F[R[i]] := R[j];
(12) \texttt{white\_to\_gray}(F[R[i]]); F[R[i]] := R[j];

Allocation of a cell is accomplished by a combination of one of (0), (1), (2) or
(3), and one of (4) or (5). Each of (0), (1), (2) and (3) corresponds to a collector
step.

An error state occurs when a cell that is reachable from a register (root)
becomes free. If error states are not reachable from an initial state, an algorithm
for concurrent garbage collection is said to be safe. Although liveness is also
required for concurrent garbage collection, we search for algorithms only according
to the safety property.

In this case study, we searched for algorithms with respect to a finite model.
The model consists of only three cells and three registers. A state of the model
can therefore be represented by 20 bits. The only initial state is that in which
all the registers hold the 0 value, all the cells are free, and the collector is in the
shade step.

We defined the algorithm space according to whether or not each of thirteen
operations is allowed. The number of algorithms in the space is therefore \(2^{13}\).
We applied finite model checking to each algorithm and computed the maximal
algorithms that satisfy the safety property, i.e., do not reach an error state. Algorithms
are maximal if they allow as many operations as possible. The verifier,
\textit{i.e.}, model checker, was implemented in \texttt{C}.

The following maximal algorithms were identified:

- 000011111111 no allocation
- 111100111111 no allocation
- 01111111110111 a variant of on-the-fly
- 111110110111 on-the-fly
- 011111011001 snapshot
- 1111101011001 a variant of snapshot
- 111111110100 a variant of on-the-fly
- 1111111111000 no field update

Each algorithm is represented by thirteen bits corresponding to the thirteen
operations, where 0 indicates that the corresponding operation is inhibited, and
1 indicates that it is allowed. Among these algorithms, the first two do not
permit allocation of new cells. The last one does not permit the fields to be
updated. Therefore, only the following five algorithms are meaningful for con-
current garbage collection:

- on-the-fly and its two variants,
2.2 Mutual Exclusion without Semaphores

This case study looks for variants of Dekker’s algorithm for mutual exclusion among processes. In the previous study, the algorithm space was defined by a fixed set of operations. In other words, the space was comprised of tuples of thirteen boolean values. In this case study, we define the space of programs consisting of pseudo-instructions.

Dekker’s algorithm realizes mutual execution among processes without semaphores. Figure 2 shows an instance of the algorithm for two processes. It can be generalized to an arbitrary number of processes, but we consider only two processes in this paper. In the figure, \( m_1 \) denotes the number of the process that is executing the code (1 or 2), and \( m_2 \) denotes the number of the other process (2 or 1). The entry part realizes mutual execution before entering the critical section, and the finishing part is executed after the critical section. The idle part represents a process-dependent task.

The safety property of Dekker’s algorithm is:

Two processes do not simultaneously enter the critical section.

Liveness is:

There does not exist an execution path (loop) that begins and ends with the same state, at which one process is in its entry part, and satisfies the following conditions.

- The process stays in the entry part on the execution path, i.e., it does not enter the critical section.
- Both processes execute at least one instruction on the execution path.

In this case study, liveness was also checked during the search, using the nested depth-first search. This is a technique developed for SPIN, one of the more popular model checkers [18]. The verifier, as well as the search problem discussed later, was implemented in Java.

We represent Dekker’s algorithm using pseudo-code consisting of pseudo-instructions. The pseudo-code may refer to three variables, each of which holds a boolean value.
for (; ;) {
    // beginning of the entry part
    flags[me] = true;
    while (flags[you] == true) {
        if (turn != me) {
            flags[me] = false;
            while (turn != me) {} // Critical section
            flags[me] = true;
        }
    } // end of the entry part

    // the critical section

    // beginning of the finishing part
    turn = you;
    flags[me] = false;
    // end of the finishing part

    // the idle part
}

Fig. 2. Dekker’s algorithm.

- FLAG1: This variable corresponds to flags[1] in Figure 2.
- FLAG2: This variable corresponds to flags[2] in Figure 2.
- TURN: This variable corresponds to turn in Figure 2. TURN=true means turn=1, and TURN=false means turn=2.

The set of pseudo-instructions are:

- SET variable
- CLEAR variable
- IF variable {instructions}
- IF_NOT variable {instructions}
- WHILE variable {instructions}
- WHILE_NOT variable {instructions}

The entry part of process 1 is represented by the following pseudo-code.

```c
SET FLAG1
WHILE FLAG2 {
    IF_NOT TURN {
        CLEAR FLAG1
        WHILE_NOT TURN {
            SET FLAG1
        }
    }
}
```
In this case study, we searched for variants of the entry part of Dekker’s algorithm that satisfy both safety and liveness. The entry parts of the two processes were assumed to be symmetric in the following sense. If one refers to FLAG1 (or FLAG2), then the other refers to FLAG2 (or FLAG1). If one contains an instruction that refers to TURN, then the other contains the corresponding symmetric instruction, where IF and IF NOT, WHILE and WHILE NOT, and SET and CLEAR are symmetric to each other.

In the first experiment, we searched for a 5-instruction pseudo-code consisting of the following instructions. Note that the original entry part consists of 6 instructions.

- WHILE FLAG2 ...
- IF FLAG2 ...
- WHILE NOT TURN ...
- IF NOT TURN ...
- SET FLAG1
- CLEAR FLAG1

We discovered the code shown in Figure 3(a). This code, however, is equivalent to the original code of Dekker’s algorithm, in the sense that they have the same meaning as sequential programs. We also found more than ten variants similar to the above, all equivalent to the original code as sequential programs.

In the next experiment, we imposed the following restriction.

If both processes are in their entry or finishing parts, they run with the same speed, i.e., they both execute one instruction in a single state transition of the entire system. If they read or write to the same variable simultaneously, it is nondeterministic which process wins (runs first). If both processes are not in their entry or finishing parts, one process is chosen nondeterministically and allowed to execute one instruction.

Under this restriction, we discovered the code in Figure 3(b), which consists of 4 instructions. This is the only 4-instruction code that we discovered. We also searched for a 3-instruction code, but failed.

```
SET FLAG1
WHILE FLAG2 {
  WHILE NOT TURN {
    CLEAR FLAG1
  }
  SET FLAG1
}
```

(b)

**Fig. 3.** Generated pseudo-code.
The correctness of the discovered code is not obvious. We believe that finding algorithms by verifiers is effective in situations where complex conditions are imposed that are difficult for humans to grasp.

2.3 Superoptimization

Superoptimization [21, 12, 10] is the most practical and successful work in this field. It is a technique used to synthesize the code generation table of a compiler. Given the specification of a code sequence, corresponding to some operation in an intermediate language, it searches for sequences of machine code that satisfy that specification.

Massalin initiated this technique, later made popular by Granlund and Kenner, who used it to synthesize a code generation table for the GCC compiler. The code in Figure 4(a) was found by Massalin's superoptimizer for Motorola’s 68020. This is the code for the following function (written in C):

\[
\text{int signum(int x) \{} \\
\quad \text{if (x>0) return 1;} \\
\quad \text{else if (x<0) return -1;} \\
\quad \text{else return 0;} \\
\text{\}}
\]

Recently, Mizukami repeated Massalin's work for Sparc, and obtained the code in Figure 4(b) [22].

\[
\begin{array}{ll}
\text{(x in d0)} & \text{(%x0)} \\
\text{add.1 d0,d0} & \text{addcc %x0, %x0, %x0} \\
\text{subx.1 d1,d1} & \text{subxcc %x0, %x0, %x0} \\
\text{negx.1 d0} & \text{addcc %x0, %x0, %x0} \\
\text{addr.1 d1,d1} & \text{(signum(x) in %x0)} \\
\text{(signum(x) in d1)} & \\
\end{array}
\]

(a) Motorola 68000 (b) Sparc

Fig. 4. Code generated by superoptimizers.

Superoptimization does not employ a real verifier; it only checks the correctness of the discovered code using some random numbers. This is sufficient in practice, since a human has the final decision as to whether to incorporate the code sequences obtained into the code generation table or not. In this final step, the correctness of the generated code can be manually checked.

2.4 Authentication Protocols

Finding algorithms using verifiers is effective in a field where the algorithms are very short, but still require substantial effort to verify their correctness.
Security protocols, such as those used for authentication, are good examples of such algorithms.

Perrig and Song [23] recently used Song’s protocol verifier, Athena [25], to try to discover symmetric-key and asymmetric-key mutual authentication protocols that satisfy the agreement property. They also defined a metric function that measures the cost or overhead of protocols. This function was intended to be applied to correct protocols to select the most efficient. The function was also used to restrict the set of generated protocols.

After setting UNIT_ELEMENT_COST=1 (cost to send a nonce or a principal name), NEW_NONCE_COST=1 (cost to generate a new nonce), and ASYM_ENCRYPTION_COST=3 (cost to encrypt a message with an asymmetric key), they succeeded in discovering the following protocol among those whose cost is less than or equal to 14:

**Protocol:**

\[
A \rightarrow B: \{N_A, A\}_{K_A} \\
B \rightarrow A: \{N_A, N_B, B\}_{K_A} \\
A \rightarrow B: N_B
\]

The protocol coincides with the Needham-Schroeder protocol, except for the last message. In the Needham-Schroeder protocol, the last message is \(\{N_B\}_{K_A}\). This difference occurred because they did not include secrecy in the specification.

The symmetric mutual authentication protocols they found, shown in Figure 5, are more interesting. All these protocols are simpler than any known previously. The first protocol was found using a metric function whose UNIT_ELEMENT_COST is high, while the last two were found using a metric function whose ASYM_ENCRYPTION_COST is high.

\[
A \rightarrow B: \{N_A, A\}_{K_A} \quad A \rightarrow B: N_A, A \\
B \rightarrow A: \{N_A, N_B, A\}_{K_A} \quad B \rightarrow A: \{N_A, N_B, B\}_{K_A} \\
A \rightarrow B: N_B \quad A \rightarrow B: N_B \quad A \rightarrow B: N_B
\]

**Fig. 5.** Protocols generated by the automatic protocol generator.

Perrig and Song demonstrate the applicability of their method based on these results. Their success is due mainly to their efforts to reduce the number of protocols by imposing reasonable restrictions, and to prune protocols by cheap tests before applying the protocol verifier.

## 3 Discovery for Deduction — Reachability Computation and Learning from Positive Data

The problem of synthesizing a loop invariant should have close relationship with that of learning from positive data [2, 5]. In this section, the problem of characterizing reachable states of a state transition system is discussed. This is a
problem of finding a finite representation of the set of states that are reachable from an initial state.

Assume a state transition system. Let \( S \) be the set of all states of the system, and \( \rightarrow \) be the transition relation between states. The function \( f \) can be defined that maps a subset of \( S \) to a subset of \( S \) as follows:

\[
f(S) = \{ s' \in S \mid s \in S, s \rightarrow s' \}
\]

This map is called the state transformer.

Assume a single initial state, \( s_0 \). Let \( S_{\omega} \) be the set of states that are reachable from \( s_0 \),

\[
S_{\omega} = \bigcup_{i \geq 0} f^i(\{s_0\}) \subseteq S
\]

The goal is to obtain a finite representation of \( S_{\omega} \) by inductive inference, i.e., to infer \( S_{\omega} \) from examples in \( S_{\omega} \). In the following, \( S_{\omega} \) is termed the target language.

### 3.1 Safe Generalization

Let \( S \) be a set of data and \( L_0 \) be a language family containing \( \emptyset \). In our case, \( S \) is the set of all states of the state transition system under consideration. A language is a subset of \( S \), and a language family is a set of languages. We assume that the target language, \( S_{\omega} \subseteq S \), belongs to \( L_0 \).

An example of a language family is \((\mathcal{T}\mathcal{P}\mathcal{L})_k\), which is a union of at most \( k \) tree patterns [3]. In this case, \( S \) is the set \( \mathcal{T}_\Sigma \) of closed terms composed of function and constant symbols taken from \( \Sigma \). A tree pattern is a term composed of functions, symbols, and variables. A tree pattern represents a set of closed terms that are instances of the tree pattern. If \( p_1, \ldots, p_n \) are tree patterns, where \( n \leq k \), then \( p_1, \ldots, p_n \) represents

\[
L(p_1) \cup \cdots \cup L(p_n) \subseteq S,
\]

where \( L(p_k) \) denotes the set of closed instances of \( p_k \).

Let \( L \) be a language family containing \( L_0 \), i.e., \( L_0 \subseteq L \). For example, we can take \((\mathcal{T}\mathcal{P}\mathcal{L})_k \) as \( L \) in the previous example, where \( L_0 = (\mathcal{T}\mathcal{P}\mathcal{L})_k \).

Let \( S' \) be a subset of \( S_{\omega} \), i.e., \( S' \subseteq S_{\omega} \), and \( L \) be a language belonging to \( L \), i.e., \( L \in L \). We say that \( L \) is a safe generalization of \( S' \), if the following conditions are satisfied:

- \( S' \subseteq L \).
- For any \( L_0 \in L_0 \), if \( S' \subseteq L_0 \), then \( S' \subseteq L \subseteq L_0 \).

Since \( S_{\omega} \subseteq L_0 \), \( L \subseteq S_{\omega} \) is guaranteed by the second condition.

### 3.2 Reachability Computation by Safe Generalization

Let \( S_0 = \{ s_0 \} \). Recall that \( S_{\omega} \) was defined as follows:

\[
S_{\omega} = \bigcup_{i \geq 0} f^i(S_0) \subseteq S
\]
Let $L_0 = \emptyset \in L$.

Assume that we have $S_i$ and $L_i$ such that $S_1 \subseteq S_\omega$, $L_i \subseteq S_\omega$, and $L_i \in L$. Let $S' = S_i \cup L_i$ and let $L_{i+1}$ be a safe generalization of $S'$. We then define $S_{i+1}$ as follows.

$$S_{i+1} = f(L_{i+1})$$

Since $L_{i+1}$ is a safe generalization of $S'$ and $S' \subseteq S_\omega$, $L_{i+1} \subseteq S_\omega$ holds. Therefore, $S_{i+1} \subseteq S_\omega$ also holds. It can be shown that $S_\omega = \bigcup_{i \geq 0} L_i$ and $L_0 \subseteq L_1 \subseteq L_2 \subseteq \cdots$.

The final question to ask is whether $S_\omega$ is identifiable in the limit [11], i.e., does $L_i \equiv S_\omega$ hold after a finite number of steps. This question can be rephrased as whether there exists $i$ such that $S_i = f(L_i) \subseteq L_i$. (We later discuss this problem in relation to the widening operator in abstract interpretation.)

Reachability computation using safe generalization explained so far is summarized in Figure 6.

**Algorithm** Reachability:

$s := \{s_0\};$

$L := \emptyset$;

repeat

$L :=$ a safe generalization of $S \cup L$;

$s := f(L)$;

until $S \subseteq L$;

output $L$;

Fig. 6. Reachability computation using safe generalization.

Arimura et al. defined a polynomial time algorithm, $k$-MMG, that computes a $k$-mmg ($k$-minimal multiple generalization) of a given set $S$ of closed trees [3]. Their algorithm returns a minimal language of $(\mathcal{T} \mathcal{P} \mathcal{E}_\Sigma)^k$ containing $S$, i.e., a set of at most $k$ tree patterns, whose union covers $S$, and which is minimal among such patterns. They incorporated this algorithm into Angluin’s inference machine, as shown in Figure 7 [1]. In the figure, $\min(S)$ denotes a $k$-mmg of $S$.

There are several problems with this algorithm. First, it is only defined for closed trees. This problem can be easily avoided, though, if variables in $S$ are regarded as constants. Second, the algorithm only returns one of the $k$-mmg’s, whereas, in general, there are more than one $k$-mmg. For example, if $S = \{f(a, b), f(a', b), f(a, b')\}$, then both $\{f(a, y), f(a', b), f(x, b), f(a, b')\}$ and $\{f(x, b), f(a, b')\}$ are $2$-mmg’s. This problem is crucial, because $L_2$ is not guaranteed to be a subset of the target language. Third, the algorithm always looks through $S$. The algorithm is not local and is not incremental. Therefore, $k$-mmg’s cannot be employed for safe generalization.

Theoretically, what we require is strongly monotone learning from text (positive data) as described by Lange and Zeugmann [19]. Moreover, for practical applications, we believe that updates of hypotheses should be local and incremental.
procedure $M$
begin
input: an infinite sequence $w_1, w_2, \ldots$ of strings;
output: an infinite sequence $g_1, g_2, \ldots$ of guesses;
set $g_0$ to be the null index, i.e., $L_{g_0} = \emptyset$, set $S = \emptyset$ and set $i = 0$;
repeat
read the next example $w_i$ and add it to $S$;
if $w_i \notin L_{g_{i-1}}$ then let $g_i$ be $\min(S)$ else let $g_i$ be $g_{i-1}$;
output $g_i$ and let $i$ be $i + 1$;
forever: /* main loop */
end.

Fig. 7. Angluin’s inference machine $M$.

For our purpose, the framework of learning using queries is more suitable than that of simple inductive inference, though we cannot ask queries in general. Arimura et al. formulated the procedure in Figure 8, which learns unions of at most $k$ tree patterns using subset queries [4]. This framework is considered to be an instance of the more general framework of learning from entailment [5]. In particular, the generic algorithm for learning from entailment proposed by Arimura and Yamamoto shows a close correspondence with our algorithm in Figure 6. This is because their algorithm only generates hypotheses that are subsets of the target language.

In their algorithm in Figure 9, $H_*$ denotes the target concept, and $EQ(H)$ is called an entailment equivalence query, which asks whether hypothesis $H$ is equivalent to $H_*$. The other kind of query, $SQ(C)$, called a subsumption membership query, asks whether clause $C$ is subsumed by $H_*$. Generalization of clauses $C$ and $D$ is denoted by $Gen(C, D)$. (Since their algorithm is formulated in inductive logic programming, the target concept is assumed to be a Horn sentence.)

Note that their algorithm only generates hypotheses that are subsets of the target, and therefore counterexamples returned by $EQ$ are always positive examples. The check, $S \subseteq L$, in our algorithm in Figure 6 therefore corresponds to $EQ$.

As for $EQ$, by employing safe generalization, our algorithm uses only generalizations that pass $SQ$. In other words, safe generalization makes $SQ$ unnecessary. In the next section, we give an example of safe generalization that is local and incremental.

3.3 Example of Safe Generalization — Depth-Bounded Tree Patterns

An example of safe generalization is presented in this section. Let $L_0$ be the family of unions of patterns whose maximum depth is at most $D$. Since $L_0$ is of a finite cardinality, it is theoretically inferable from positive data [2]. Let $[p]$ be the maximum depth of tree pattern $p$. It is formally defined as follows:

$$[x] = 0 \quad (x : \text{variable})$$
**Procedure: LEARN**

**Given:** the equivalence and the subset oracles for the target set $H_x \in TP^k$.

**Output:** a set $H$ of at most $k$ tree patterns equivalent to $H_x$.

begin

$H := \emptyset$;

until $EQUIV(H)$ returns “yes” do

begin

let $w$ be a counterexample returned by the equivalence query;

if there is some $h \in H$ such that $SUBSET(h \cup w)$ returns “yes” then

 generalize $H$ by replacing $h$ with $h \cup w$

else if $|H| < k$ then

 generalize $H$ by adding $w$ into $H$

else

 return “failed”

endif

end /* main loop */

return $H$;

end

**Fig. 8.** A learning algorithm for $TP^k$ using equivalence and restricted subset queries.

**Algorithm** EntLearn:

$H := \emptyset$;

while $EQ(H)$ returns “no” do /* $H \neq H$ */

Let $E$ be a counterexample returned by the query;

$D := Missing(E, H)$;

foreach $C \in H$ do make query $SQ(\text{Gen}(C, D))$;

if the query returned “yes” for some $C \in H$ then

$H := (H - \{C\}) \cup \{ \text{Gen}(C, D) \}$

else

$H := H \cup \{E\}$;

endif

end /* while */

return $H$;

**Fig. 9.** A generic algorithm for learning $H_x$ from entailment.
\[ c = 1 \quad (c: \text{constant}) \]
\[ f(p_1, \ldots, p_n) = \max([p_1], \ldots, [p_n]) + 1 \]

Let us write \( L(p) \) for the set of instances of tree pattern \( p \). \( L(p) \) contains not only closed terms, but also tree patterns that are instances of \( p \). Then \( L_0 \) can be formally defined as follows.

\[
L_0 = \{ L(p_1) \cup \cdots \cup L(p_n) \mid [p_i] \leq D \}
\]

Note that \( L(p) \) henceforth denotes the set of all the instances of \( p \), including tree patterns.

Let \( L \) be the family of unions of patterns whose minimum depth is greater than \( D \).

\[
L = \{ L(p_1) \cup \cdots \cup L(p_n) \mid [p_i] > D \}
\]

The minimum depth, \( [p] \), of tree pattern \( p \) is defined as follows:

\[
[x] = 0 \quad (x: \text{variable})
\]
\[
[c] = \infty \quad (c: \text{variable})
\]
\[
[f(p_1, \ldots, p_n)] = \min([p_1], \ldots, [p_n]) + 1
\]

Let \( \alpha, \beta, \ldots \) denote sequences of positive integers. For tree pattern \( p \) and \( \alpha \), \( p|_{\alpha} \) is defined as follows:

\[
p|_\epsilon = p \quad (\epsilon: \text{empty sequence})
\]
\[
f(p_1, \ldots, p_n)|_{\alpha} = p_i|_{\alpha}
\]

We say that \( \alpha \) is the position of \( p|_{\alpha} \) in \( p \). The length of \( \alpha \), denoted by \(|\alpha|\), is called the depth of \( p|_{\alpha} \) in \( p \).

Consider the following conditions:

- \( q_1, \ldots, q_N \) are tree patterns.
- \([q_i] \leq D \) for \( 1 \leq i \leq N \).
- \( p_1, p_2 \in L(q_1) \cup \cdots \cup L(q_N) \).
- \([p_1], [p_2] > D \).
- \( p = p[x] = \text{lgg}(p_1, p_2) \), where \( \text{lgg}(p_1, p_2) \) denotes the least general generalization of \( p_1 \) and \( p_2 \), and \( x = x_1, \ldots, x_n \) is a sequence of variables that are introduced in \( \text{lgg}(p_1, p_2) \).
- \( p_1 = p[t] \), where \( t = t_1, \ldots, t_n \) is a sequence of tree patterns \((p[t]) \) denotes the result of substituting \( t \) for \( x \) in \( p[x] \).
- \([p] > D \).
- There do not exist two subpatterns \( r[x] \) and \( s[x] \) of \( p \) that satisfy the following conditions.
  - \( p|_{\alpha} = r[x] \) and \( p|_{\beta} = s[x] \).
  - \(|\alpha| \leq D \) and \(|\beta| \leq D \).
  - \( r[x] \neq s[x] \).
  - \( r[t] = s[t] \).
Theorem 1. If the above conditions are all satisfied, then \( p \in L(q_i) \) for some \( q_i \).

(proof) Assume \( p_i \in L(q_i) \), \( q_i = q_i[y] \) and \( p_i = q_i[u] \), where \( y = y_1, \ldots, y_m \) is a sequence of variables, and \( u = u_1, \ldots, u_m \) is a sequence of tree patterns. Since \( |p| > D \), those subpatterns of \( p_i \) that are replaced with variables in \( p \) are all at depth greater than \( D \). Therefore, since \( |q_i| \leq D \), those patterns are all subpatterns of some \( u_j \).

If \( u_j \) occurs only once in \( q_i \), we can define \( v_j[x] = p_i \alpha \), where \( u_j = q_i |\alpha \). We then have \( u_j = v_j \).

\[
q_i = \cdots \cdot y_j \cdots y_j \cdots \\
p_i = \cdots \cdot u_j \cdots u_j \cdots
\]

Let \( r[x] \) and \( s[x] \) be the corresponding subpatterns in \( p \).

\[
p = \cdots \cdot r[x] \cdots s[x] \cdots \\
p_i = \cdots \cdot r[t] \cdots s[t] \cdots
\]

By the last condition of the theorem, \( r[x] \) and \( s[x] \) must be equal. We can then define \( v_j[x] = r[x] = s[x] \). We also have \( u_j = v_j[t] \).

Finally, we have

\[
p = q_i[v_1[x], \ldots, v_m[x]],
\]

and \( p \in L(q_i) \). (end of proof)

We now assume that the target language, \( S_\alpha \), belongs to \( L_\alpha \). We also assume

\[
S' = L(p_1) \cup \cdots \cup L(p_n).
\]

If \( |p_i| \leq D \), we instantiate those variables that occur in \( p_i \) at depth less than or equal to \( D \) and obtain \( p_i, \ldots, p_m \) such that \( |p_i| > D \) and

\[
L(p_i) = L(p_i_1) \cup \cdots \cup L(p_i_m).
\]

(We assume \( \Sigma \) is finite.) We then replace \( p_i \) with \( p_i_1, \ldots, p_i_m \). Therefore, we can always assume \( |p_i| > D \) for all \( i \), so \( S' \in L \). In this context, we have the following theorem for safe generalization.

Theorem 2. We pick \( p_1 \) and \( p_2 \) from \( S' \) and compute \( p = \text{agg}(p_1, p_2) \). If \( |p| > D \), then we check the last condition of Theorem 1. If the condition holds, we replace \( p_1 \) and \( p_2 \) in \( S' \) with \( p \). The resulting set of tree patterns is a safe generalization of \( S' \).

The final question to ask is whether the sequence of languages converges. Let \( q_i = q_i[y] \) and \( q_i |\alpha = y_j \), where \( \alpha \) is the most shallow position of \( y_j \) in \( q_i \). Take \( \beta \) such that \( \alpha \beta = D + 1 \). If \( p_1 \) and \( p_2 \) satisfy the following conditions, then we can apply safe generalization to \( p_1 \) and \( p_2 \).

- \( p_1 = q_i[u] \) and \( p_2 = q_i[u'] \).
\[ |p_1| > D \text{ and } |p_2| > D.\]

- \( u_k = u'_k \) unless \( k = j \).
- \( u_{j|\alpha} \) and \( u'_{j|\alpha} \) begin with the same symbol unless \( \gamma = \beta \delta \) form some \( \delta \).
- \( u_{j|\beta} \) and \( u'_{j|\beta} \) begin with different function (or constant) symbols.
- \( u_{j|\beta} \) does not occur at any position of \( p_1 \) except at \( \alpha' \beta \), where \( q_{j|\alpha'} = y_j \).

By these conditions, a variable is introduced at position \( \alpha \beta \) of \( p_1 \) by safe generalization.

During the iterative computation of \( L_i \), such \( p_1 \) and \( p_2 \) should appear, since any term in \( S_i \) (or its generalization) is eventually generated. This means that, for any such \( y_j \) and \( \beta \), it is always possible to introduce a variable at \( \beta \) in \( u_j \).

For the last condition \( u_{j|\beta} \) does not occur \( \cdots \), \( \Sigma \) must have a sufficient number of function or constant symbols. In some cases, we should further instantiate \( p_1 \) and \( p_2 \) to guarantee this condition.

### 3.4 Widening and Identification in the Limit

**Abstract interpretation** is a methodology for analyzing programs using an abstract domain whose elements correspond to a set of program states. An abstract domain forms a lattice, and it is possible to obtain the general representation of a loop invariant using simple iterative computation.

If the abstract domain is finite, such iterative computation always terminates. More formally, if \( P_0 \leq P_1 \leq P_2 \leq \cdots \) is a non-decreasing infinite sequence of elements of the lattice, then there exists \( n \) such that for all \( i \geq n \), \( P_{i+1} = P_i \).

However, if the abstract domain is infinite, a non-decreasing sequence does not always terminate. In abstract interpretation, widening operators are employed to guarantee termination. A binary operator, \( \nabla \), is termed a **widening operator** [7] if it satisfies the following conditions:

- \( P \leq P \nabla Q \) and \( Q \leq P \nabla Q \).
- For any non-decreasing infinite sequence \( P_0 \leq P_1 \leq P_2 \leq \cdots \), the sequence, \( Q_0 \leq Q_1 \leq Q_2 \leq \cdots \), defined below finitely converges.
  - \( Q_0 = P_0 \).
  - \( Q_{i+1} = Q_i \nabla P_{i+1} \).

The last condition means that there exists \( n \) such that for all \( i \geq n \), \( Q_{i+1} = Q_i \).

This condition corresponds to the identification in the limit as described by Gold [11].

However, widening operators do not preserve the limit in general. The limit of \( Q_0 \leq Q_1 \leq Q_2 \leq \cdots \) may be greater than that of \( P_0 \leq P_1 \leq P_2 \leq \cdots \).

An example of a widening operator follows. Let the abstract domain be a set of closed intervals \([l, u]\) on the real axis. It forms a lattice with respect to set inclusion. A widening operator for this domain is

\[
[l, u] \nabla [l', u'] = \begin{cases} 
-\infty & \text{if } l' < l \text{ then } -\infty \text{ else } l, \\
\infty & \text{if } u' > u \text{ then } +\infty \text{ else } u 
\end{cases}
\]

Another example of a widening operator is that of polyhedra analysis [8, 15]. The abstract domain is the set of polyhedra in \( \mathbb{R}^n \). It forms a lattice with respect
to set inclusion. (The previous example is a special case where \( n = 1. \)) Let \( P \) and \( Q \) be polyhedra. They can be expressed as sets of linear inequalities. Then \( P \vee Q \) can be defined as the set of linear inequalities of \( P \) that are satisfied by \( Q \). The following is a refinement of this operator [14, 15]:

All the inequalities of \( P \) satisfied by \( Q \) are kept in the result, together with all the inequalities of \( Q \) that are mutually redundant with an inequality of \( P \), i.e., saturated by the same vertices and rays of \( P \).

Widening operators are employed to guarantee termination or accelerate convergence. However, they may cause information to be lost in the sense that convergence does not reach the exact limit. Therefore, it is important to combine safe generalization and widening. If \( Q \vee P_{n+1} \) is a safe generalization of \( Q \vee P_{n+1} \), then the sequence \( Q_0 \leq Q_1 \leq Q_2 \leq \cdots \) reaches the exact limit.

4 Concluding Remarks

In the first endeavor discussed in this paper, discovery by deduction, the issue of how to decrease the size of the search space used to find programs is the most crucial issue. For example, Perrig and Song pruned protocols before applying the protocol verifier. More research should be performed to obtain general principles for efficient search for programs.

The notion of safe generalization was introduced in the second endeavor. Since many techniques for learning from positive data have been proposed, application of these techniques to reachability computation is fruitful. In particular, it should provide many leads to designing widening operators.

Acknowledgements

We deeply thank Akihiro Yamamoto, Hiroki Arimura and Takashi Yokomori for valuable comments on Section 3 of this paper. The work reported in this paper was supported, in part, by the Ministry of Education, Science, Sports and Culture, under the grants 11480062 and 10143105.

References


Integrating Information Visualization and Retrieval for Discovering Internet Sources

Hayato Ohwada and Fumio Mizoguchi

Information Media Center,
Science University of Tokyo
Noda, Chiba, 278-8510, Japan
{ohwada,mizo}@ia.noda.sut.ac.jp

Abstract. One of the important technologies in knowledge discovery is to access the desired information from the large amount of data stored on the WWW. At present, such information can be accessed by a browser itself or by using a keyword search function. However, browsing is a time consuming task where a user must access individual pages one by one. Furthermore, it is hard for users to provide reasonable keywords to discover their desired pages in general. This paper outlines an approach of integrating information visualization and retrieval to improve effectiveness WWW information access. In this approach, the link structure of WWW is displayed in a 3-D hyperbolic tree in which the height of a node within the tree indicates a user’s “interestingness”. Here, interestingness is calculated by a fitting function between a page and user-supplied keywords, and this measure can be used to filter irrelevant pages, reducing the size of the link structure. Such functions are incorporated within our browser, allowing us to discover desired pages from a large web site incrementally. Relatively large web sites were selected to show the performance of the proposed method with improved accuracy and efficiency in WWW information access.

1 Introduction

It is natural to regard WWW as a repository for knowledge discovery on the Internet. At present, such information can be accessed by a browser itself or by using a keyword search function. However, it is difficult to discover desired web pages using these functions due to the following reasons.

– Lack of global view of WWW information
  There are no functions for viewing the overall structure of WWW information. Thus, a user must access individual pages one by one.

– Lack of query navigation function
  Although keyword search or database-like queries realize efficient information access, it is hard for users to provide reasonable queries to discover their desired pages in general.

© Springer-Verlag Berlin Heidelberg 2000
Some solutions to the first problem have recently been provided. The hyperbolic tree, developed by Lamping [Lamping 95], is known among the human-interface community as a reasonable tool for providing a global view of WWW information. It presents a GUI for hierarchical information structure including WWW and allows displaying the whole information structure within a hyperbolic plane. It also supports focus of attention by mouse operation; thus a user can see the structure at any angle.

As for the second problem, an automatic or semi-automatic structuring method for WWW information has been reported in the database community. Ashish, et al. [Ashish 97] and Atzeni et al. [Atzeni 97] developed “wrappers” to structure multiple web sites and to construct a WWW information database. Such approaches are useful for effective information retrieval by integrating various Internet sources and realizing SQL-like queries.

Although the above successful results were obtained by distinct communities, we hope these can be integrated to give birth to supporting tools for knowledge discovery from WWW information. A hyperbolic tree approach is a comprehensive visual tool, but it is difficult to access a large amount of information using mouse operation only. In contrast, a database approach realizes efficient information access, but still ignores interactive retrieval processes in knowledge discovery. This paper conjectures that a combination use of information visualization and retrieval functions can navigate users to their desirable pages with improved accuracy and efficiency in WWW information access.

The rest of the paper is devoted to describe how visualization and retrieval functions are reconstructed based on the previous results and are integrated into a WWW information access system (WIAS). The information retrieval part of WIAS calculates the fitness of a web page to a user-supplied query. The visual part of WIAS draws a hyperbolic tree in which each node corresponds to a web page and the height of a node is the fitness of the page to the query. A web page with low fitness is filtered; thus the structure of interesting pages only are displayed as a hyperbolic tree. These functions realize interactive information retrieval, allowing us to access WWW information effectively by cooperating with existing browsers.

2 Interactive Information Retrieval

2.1 Information structure and operations

The target information structure is defined as a directed graph consisting of a set of texts (usually web pages) and their links (specified by anchor tags in HTML). Each text is composed of combinations of tags and strings. A tag is either a prespecified one like HTML or a user-defined one like XML. In general, a tag is specified as \(<\text{tag}\>\text{str}</\text{tag}\>\) where \(\text{str}\) is a string. For text \(t\), we assume that \(\text{tag}, \text{str} \in t\) if \(t\) contains \(\text{tag}\>\text{str}</\text{tag}\>\).

A string may contain tags that are nested. Let \(n\) be the depth of a tag structure. This case is denoted as \(\text{tag}_1, \ldots, \text{tag}_n, \text{str} \in t\). For example, we have a text
<chapter><section>s1</section></chapter>.

Text t has the following information:

chapter."s1" ∈ t
category.section."s1" ∈ t

We put a limited length of hypertext searchability and avoid loop of links between hypertexts. In this setting, the above-mentioned information structure constructs a tree whose node is a hypertext.

We have some operations on a tree. Let \( D \) be a display in which every node of tree (tree) appears. Node \((t)\), which appears at the center of \( D \), is called the focus, and we denote this fact as \( \text{focus}(\text{tree}, t) \in D \). A user can change focus from \( t \) to a distinct node \( t' \). After changing, the updated display \( D' \) has the information, \( \text{focus}(\text{tree}, t') \in D' \) and \( \text{focus}(\text{tree}, t) \notin D' \).

An action to move node \((t)\) at position \((x)\) can be denoted by \( T_{t \rightarrow x} \). For two displays, \((D)\) and \((D')\), this action is specified by the mapping \( D \xrightarrow{T_{t \rightarrow x}} D' \). After taking an action \((T_{t \rightarrow o})\) to move a node to the center \((o)\), \( \text{focus}(\text{tree}, t) \in D' \) holds in the updated display \((D')\). Suppose that node \((t)\) has information denoted by \( \text{tag} \). \( s \in t \) and \( s \) appears in display \((D)\). Such a tag in \( t \) is called attribute and includes the title and the updated time of \( t \). Node \( t \) also has some attributes such as the fitness of \( t \) to a user-entered query, access count and history. Such attributes are presented in the display.

### 2.2 Features of interactive IR

Interactive Information Retrieval (IIR) integrates structure visualization and retrieval of WWW information, providing interactive and incremental access to Internet sources. This framework has the following features:

1. Visualizing the whole structure of hypertexts
   In order to look over a hypertext structure, the structure is displayed as a hyperbolic tree where nodes of upper-level texts are put around the center and their size is relatively large. In contrast, nodes of lower-level texts are distributed at the corners of the display. A user can see a focal node and nodes around it easily, and can also take in the larger hierarchical structure at a glance.

2. Smooth change of focus
   Focus change is used to access nodes of lower-level texts. From the viewpoint of human cognition, this change is done smoothly, so that there is little gap between text transitions. When the focal node is specified by a mouse click, several displays \((D_1)\) are created and appear as a sequence \( D \rightarrow D_1 \rightarrow D_2 \rightarrow ... \rightarrow D' \) to show the continuous process of focus change. Mouse dragging allows us to arbitrarily change viewpoints, and this provides a direct manipulation interface to access nodes easily.
3. Attribute-value query on the semi-structured texts
   This extracts information like attributes and keywords from a flat text and realizes an attribute-value query. By parsing the text, such information can be expressed as follows:

   \[ t, a_1, a_2, \ldots, a_n, s \in t \]

   where \( t \) is a text, \( s \) is a string and \( a_i \) is a nested tag. The expression is then matched with a user-entered query \( \langle a_1', \ldots, a_n', s' \rangle \). The matching is approximated based on a similarity operator \( \sim \), and is substituted in the following conditions:

   \[ a_1 \sim a_1' \land \ldots \land a_n \sim a_n' \land s \sim s' \]

   This means that we introduce a similarity measure between two strings or tags and process a query by computing the fitness of texts to be retrieved based on this measure. Retrieved texts are ranked in this way.

4. Visualizing attribute values for each node
   Attribute values for each node have useful information to navigate users to their desired texts. These values are also displayed within a hyperbolic tree. Attributes such as the title of the text and the fitness to a query tell users which nodes should be accessed. By changing the focus based on these attribute values, users may easily access their desired texts.

5. Filtering texts based on attribute values
   While every node is displayed in the previous feature, this feature eliminates meaningless nodes by using a threshold to determine nodes to be displayed. Consider node \( t \) such that every lower-level node \( t' \) of \( t \) does not have a reasonable attribute value (which is determined by the threshold). In this case, we remove \( t \) from the hyperbolic tree, and reduce the tree size. This allows users to focus on interesting nodes.

The above features are interconnected within our IIR framework. In particular, features (3), (4) and (5) are repeatedly executed. Through such a process, a user progressively explores the desired texts. This is a distinctive feature of IIR.

IIR accelerates this interactive aspect by cooperating with a browser because feature (2) above is inter-related with the browser. After a node is mouse clicked, focus is changed, and the corresponding text is displayed in the browser. In the same way, when a user browses a text, this text becomes the focus within a hyperbolic tree display. This type of interaction helps users to understand which portion of hypertext structure is focused on.

3 WWW Information Access System

This section describes how the IIR framework is realized within WIAS. WIAS takes the source sent from a WWW server and access log of the server as input information. This information is processed by tag and string parsers, and is used in the modules such as structure visualization and query manipulation. We also
developed a browser that interacts with the IIR modules. All modules except for the string parser are implemented in Java language, enhancing the portability of WIAS. The visual part of WIAS extends a 2-D hyperbolic tree to a 3-D type because node information is represented as the height of the node. A typical hyperbolic tree algorithm lays out nodes as a circle, but our algorithm lays them out as an ellipse. Thus, front nodes may not hide rear nodes.

![Structure of hypertexts](image)

**Fig.1.** Output of WIAS

The output of WIAS is shown in Figure 1. A window is decomposed into a display for the structure of hypertexts, and areas for entering queries and commands. Each text is displayed as a node whose height indicates node information such as user-access count and fitness to a query. The height is a good indicator for efficiently accessing interesting texts. A method of calculating the fitness is shown in the next section.

A hyperbolic tree can be changed arbitrarily by mouse operation. Focus is changed by clicking the mouse on a node. Mouse dragging can be accepted at any position, making it easy for users to change the viewpoint of the tree.
Focus is changed by a distance-preserving transformation on a hyperbolic plane. The transformation $T_{i\rightarrow x}$ mentioned in Section 2 is distance-preserving. Given a mouse click, the position of every node on the hyperbolic plane is calculated using $T_{i\rightarrow o}$. As for mouse dragging, if a node $(t)$ is at the mouse-pressed position, the node moves at the center of the hyperbolic plane and then moves at the mouse-released position. This transformation is a composite of $T_{i\rightarrow o}$ and $T_{i\rightarrow x}$.

### 3.1 Computing the fitness of a text

A query for text retrieval is represented by the conjunction:

$$B_1 \land \ldots \land B_k$$

where $B_i (i = 1, k)$ is a literal. A literal is of a specialized form:

$$tag_1.tag_2.\ldots.tag_n.str$$

where $tag_i$ and $str$ are strings. For each text $t$, a conjunction $B_1 \land \ldots \land B_k$ is true if $\forall_i B_i \in t$ is true. The result of the query is a set of texts whenever the conjunction is true.

We then consider an approximated query based on partial matching between strings usually used in the information retrieval community. The proposed method is based on TFIDF in which a similarity between documents (or strings) is defined [Salton 91]. In the following, we briefly introduce TFIDF.

Let $T$ be a vocabulary list of atomic terms that appear in a set of strings $s_1, \ldots, s_m$ being retrieved as texts. A string $s_p (1 \leq p \leq m)$ is associated with a vector $v_p \in \mathbb{R}^{|T|}$ whose element takes a real value. For a term $a \in T$, an element of a vector $v_p$ is represented by $v_{p,a}$. The value of $v_{p,a}$ indicates the importance of term $a$ with respect to the string associated with $v_{p,a}$. If $a$ does not occur in $s_p$, $v_{p,a}$ is 0. Otherwise, the importance is computed in the following:

$$v_{p,a} = \log(TF_{v_{p,a}} + 1) \cdot \log\left(\frac{m}{c_a}\right)$$

where $TF_{v_{p,a}}$ is the number of times that term $a$ occurs in the string $s_p$, and $c_a$ is the total number of strings that contain the term $a$.

The similarity between strings $s_p$ and $s_q$ can be computed using this vector representation of strings. Let $v_p$ and $v_q$ be vector representations of the strings $s_p$ and $s_q$, respectively. The similarity is defined as follows:

$$\text{sim}(s_p, s_q) = \frac{\sum_{a \in T} v_{p,a} v_{q,a}}{\|v_p\| \cdot \|v_q\|}$$

---

1 WIAS uses the Japanese morphological analysis system, Chasen [Matsumoto 99], to obtain atomic terms that are noun in a dictionary.

2 Although $v_p$ is very long in a matrix notation, this is quite sparse. Thus, efficient implementation is possible.
We use the above TFIDF framework to compute the fitness of a text to a user-entered query. Consider an atomic query $B = tag_1, \ldots, tag_n, str$. If there is an attribute-value pair $C_i = tag'_1, \ldots, tag'_m, str' \in t$ in text $t$, the similarity between the pair and the query is computed in the following:

$$sim(C_i, B) = \prod_{j=1}^{n} sim(tag_j, tag'_j) \times sim(str, str')$$

We then define the fitness $FIT(t, B)$ of the text $t$ to the query $B$ as follows:

$$FIT(t, B) = \max_l sim(C_l, B)$$

This means that the most similar attribute-value pair is selected to compute the fitness.

Finally, the fitness of text to conjunctive query $B_1 \land \ldots \land B_k$ is defined in the following:

$$FIT(t, B_1, \ldots, B_k) = \prod_{i=1}^{k} FIT(t, B_i)$$

Since the similarity measure in the TFIDF framework is between zero and one, the fitness is also between zero and one. In our HIR framework, fitness is displayed as the height of a node within a hyperbolic tree.

### 3.2 Reducing the tree

Filtering uninteresting nodes is the most important feature of WIAS. Given a query, the fitness to the query for each node is computed and displayed as the height of the node. The filtering function then removes nodes that have lower fitness, and restructures a reduced hyperbolic tree. This is very useful for large web sites because users can focus on interesting texts only.

Figure 2 shows a filtering process in WIAS. The left figure is a hyperbolic tree of our university research division web site (Science University of Tokyo, Information Media Center, http://www.imc.sut.ac.jp/) consisting of 226 texts. Since the height of each node indicates the fitness of the associated text to a user-supplied query, higher nodes are interesting for the user. There are nine nodes among a large number of nodes in the figure.

The right figure shows a reduced hyperbolic tree. Even uninteresting nodes that are reachable to interesting nodes still exits in the hyperbolic tree. The reduced tree constructs a web site for the user and allows the user to see the manageable-sized web site. Although existing search engines list up pages that are interesting for users, it is impossible to see the relationships between the searched pages. In contrast, a hyperbolic tree representation allows users to capture the web page structure and makes it easy to find interesting portal sites. Moreover, some queries can be put incrementally, and more interesting pages can be explored. A combinatorial use of information retrieval and visualization supports processes in knowledge discovery.
4 Experiments

We evaluate WIAS in the following points:

Accuracy  User-desirable texts should be accessed in an exhaustive manner. It is preferable that there is no missing texts.

Efficiency  Cost to reach desirable texts should be low. Uninteresting texts should not be accessed.

These properties tend to be mutually exclusive, and thus well-balanced information access is needed. Based on the above points, two experiments were conducted to show the effectiveness of hyperbolic tree visualization and filtering.

4.1 Effectiveness of visualization

A web site we used is our university research division site mentioned in Section 3.2. We selected as subjects ten bachelors belonging to our university who did not access the site before experiments. We divided the subjects into two groups; one uses a web browser with hyperbolic tree visualization and the other use the browser only. We gave each subject to 10 problems in which a hint is put as an abstract of a text to be found, and recorded passage time to access all texts and operation history. A problem was given, for example, “Search a text where you can see the picture of XXX professor in YYY workshop presentation”. Since sufficient keywords were given, the subjects could find target texts correctly. Note that the possible operations the subjects used were mouse click for hyperlink selection, page back and forward, home position and bookmark registration.

Figure 3 shows the number of texts the subjects found. The graph indicates that use of hyperbolic tree visualization does not contribute efficient information access at an early stage, but achieves efficiency when 15 minutes passed. This means that global view function of hyperbolic tree visualization can assist users to find their desirable texts efficiently when a target web site becomes larger.
Fig. 3. Browsing performance with and without use of hyperbolic tree visualization

Fig. 4. Title reference in browsing

Checking subjects’ histories about mouse operations is needed to clarify what is going on in browsing texts. WIAS allows user to specify a node by mouse move and to see the associated text title. Figure 4 shows how much times subjects see text titles in browsing about 50 texts. The total times was over 100 titles to see all the texts. At the beginning, the times increased dramatically, but were slightly changed after ten texts were found. This indicates that users saw the overall structure of the web site at the beginning and then found texts to be accessed. As a conclusion, the structure visualization based on the hyperbolic tree provides an efficient and exhaustive browsing function.
4.2 Effectiveness of filtering

The second experiment shows the advantage of hypertext filtering by combining visualization and query-based retrieval. The website we selected is our university site (http://www.sut.ac.jp) consisting of 3000 texts. This site supports search engine InfoSeek that is used to compare the IIR framework with mixed use of browsing and search engine. Since there are quite a few texts in the site, it is difficult to display the whole structure of the texts in a hyperbolic tree style. This means that a user must pose queries and filter a number of texts incrementally. We also have ten subjects that are divided into two groups. Group 1 subjects use a browser and InfoSeek, and the others use query-based retrieval and filtering of WIAS functions. The problem given to the subjects is to find five research laboratory documents related to concepts such as “environment” and “automobile.” The subjects posed some keywords and repeatedly filtered the hypertexts.

![Fig. 5. Browsing process without use of WIAS](image1)

![Fig. 6. Browsing process with use of WIAS](image2)

Figures 5 and 6 show the experiment results. The average times to find texts that correspond to the portal sites of the five research laboratories are 23 minutes without using WIAS and 12 minutes with using WIAS. In the figures, character
“S” indicates that a subject posed a query, and character “F” with an arrow indicates that a subject found a text. The two graphs allow us to see the advantage of using WIAS. Specifically, the number of queries posed is quite small for WIAS use, and the number of mouse clicks is half that for normal browsing and keyword searching. This is because WIAS supports visualization of texts including keywords in a hierarchical manner, and thus it is easy for the subjects to browse and find upper-level nodes that are portal site pages of research laboratories. This result exemplifies the advantage of integrating visualization and retrieval in the IIR framework that provides a significant insight as a supportive tool for discovering WWW information.

5 Comparison with other work

Ashish et al. [Ashish 97] and Atzeni et al. [Atzeni 97] have attempted to structure WWW information to support SQL-like queries. Their approach is to construct a database from multiple web sites using “wrapper” programs that deal with semi-structured information. Our framework does not compete with theirs but may exploit it in generating attribute-value pairs from WWW information. The difference is that a query in WIAS is relatively simple because WIAS just focuses on retrieving desired texts. Thus, we do not necessarily deal with a general relational model. This increases the efficiency of information retrieval by adopting special data structures.

We suggest query manipulation based on TFIDF that was developed by the information retrieval community. Such an approximate approach has not been taken in database research literature. Recently, Cohen introduced a “soft” join operation like relational database framework [Cohen 99]. This extends the join using TFIDF in order to provide query manipulation suitable for WWW information. Data and the query are expressed by first-order logic, and information is retrieved by variable binding. This means that a soft join has greater expressive power than our IIR framework. However, visualization is not considered or such a first-order representation is not exploited within WWW structure visualization. Our method focuses on text retrieval only, and text fitness to a query can be displayed within structured visualization.

Visualization methods for WWW information are recently proposed within a discovery science community [Hirokawa 98] [Sawai 98] [Shibayama 98]. However, integration with information retrieval is not realized. In our approach, a web browser was specially designed and implemented to communicate with visualization and retrieval, showing the performance of this integrated approach.

Munzner designed an information space in which multiple hyperbolic trees are configured three dimensionally [Munzner 98]. However, there are a number of nodes on the information space map, and it is hard to focus on the appropriate portion of a web site. An alternative visualization scheme was developed as a “cone” tree by Robertson [Robertson 91]. In a cone tree, nodes are distributed in a three-dimensional space, and thus front nodes may hide rear nodes. Moreover, focus change is not as easy as in a hyperbolic tree representation.
6 Conclusions

In this paper, we proposed an integration of structure visualization and retrieval of WWW information. The resulting system, WIAS, consists of hyperbolic tree visualization and attribute-value pair query manipulation, and provides a filtering function to reduce the size of the WWW information structure. Experiments were conducted to show the advantage of the interactive information retrieval framework in WIAS. The obtained statistics demonstrate the effectiveness of WIAS in accessing WWW information. Since WIAS can be interconnected with our browser, the proposed framework will help users browse a large number of texts on the Internet.

References


A Unifying Approach to HTML Wrapper Representation and Learning

Gunter Grieser¹, Klaus P. Jantke², Steffen Lange³, and Bernd Thomas⁴

¹ Technische Universität Darmstadt, FB Informatik
  Alexanderstraße 10, 64283 Darmstadt, Germany
  e-mail: grieser@informatik.tu-darmstadt.de

² Deutsches Forschungszentrum für Künstliche Intelligenz
  Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
  e-mail: jantke@dfki.de

³ Universität Leipzig, Institut für Informatik
  Augustusplatz 10-11, 04109 Leipzig, Germany
  e-mail: slange@informatik.uni-leipzig.de

⁴ Universität Koblenz, FB Informatik
  Rammsweg 1, 56070 Koblenz, Germany
  e-mail: bthomas@uni-koblenz.de

Abstract. The number, the size, and the dynamics of Internet information sources bears abundant evidence of the need for automation in information extraction. This calls for representation formalisms that match the World Wide Web reality and for learning approaches and learnability results that apply to these formalisms. The concept of elementary formal systems is appropriately generalized to allow for the representation of wrapper classes which are relevant to the description of Internet sources in HTML format. Related learning results prove that those wrappers are automatically learnable from examples. This is setting the stage for information extraction from the Internet by exploitation of inductive learning techniques.

1 Motivation

Today's online access to millions or even billions of documents in the World Wide Web is a great challenge to research areas related to knowledge discovery and information extraction (IE). The general task of IE is to locate specific pieces of text in a natural language document.

The authors' approach draws advantage from the fact that all documents prepared for the Internet in HTML, in XML or in any other possibly forthcoming syntax have to be interpreted by browsers sitting anywhere in the World Wide Web. For this purpose, the documents do need to contain syntactic expressions which are controlling its interpretation including its visual appearance and its interactive behaviour. In HTML, these are the text formatting and annotating strings (tags), and in \LaTeX, for instance, there are numerous commands. The
document’s contents is embedded into those syntactic expressions which are usually hidden from the user.

The user is typically not interested in the varying syntactic tricks invoked for information presentation, but in the information itself. Accordingly, the present approach assumes that the user deals exclusively with the desired contents, whereas a system for IE should deal with the syntax.

In a characteristic scenario of system-supported IE, the user is taking a source document and is highlighting representative pieces of information (s)he is interested in. It is assumed that the user’s view at a certain document, which might evolve gradually or which might even change over time, can be represented as a certain relation of text strings from the underlying source. Thus, the user’s input are just a few sample instances from the relation (s)he is seeing when looking at the given source document.

It is left to the system to understand how the target information is wrapped into syntactic expressions. This is a first learning task posed to the system.

Next, the system has to generalize the wrapper concept hypothesized for coming up with an extraction procedure. Applied to the given source document, this extraction mechanism generates a certain hypothetical relation which is returned to the user. The step of generalization mentioned is a second learning task to be solved by the system.

In response to providing a few samples illustrating the user’s view at the source document, (s)he receives a list of extracted tuples. The user may compare the system’s guess to the results aimed at and, in dependence on the comparison’s outcome, complain about the result — if necessary — by indicating erroneously extracted tuples or by supplementing those tuples missing in the system’s output.

When the user is returning this information to the system, a new cycle of learning and information extraction is initiated. Several further interactions may follow, and the extraction mechanism generated may be applied to further source documents. Several cycles of interaction will improve the IE results.

From a theoretical perspective, the system is performing a two-level learning process based on particular positive and negative examples provided by the user.

2 Introduction

Let us explain this approach with the help of the bibliography of this paper. Imagine, the list of referenced papers is presented in a semi-structured form, namely HTML, as follows:

\[html\]
\[
\text{body}\]
\[
\text{h1 References}\]
\[
\text{ol}\]
\[
\text{li}\]
\[
\text{D. Angluin, ‘Inductive inference of formal languages from positive data’}, \]
\[
\text{Information and Control}, \text{45/4}, 117-135, (1980)\]
\[
\text{b}
\]
\[
\text{li}\]
\[
\[
\text{b}
\]
\[
\text{li}\]
\[
\text{S. Arikawa, ...}
\]

\text{Figure 1.}
In order to extract information, we can assume special text segments to be delimiters (henceforth called anchors) marking the beginning and the end of the relevant information to be extracted.

For instance, to extract the authors of a publication, for example, the text segments \((ii)\) resp. \(.'\) are interpreted as the left and the right anchor for the authors' name(s). The year of publication (more exactly, its last two digits) can then be identified by the left anchor \((19)\) and the right anchor \(./(ii)\). Using these anchors, the extraction of all authors' names and the relevant years of publication from the bibliography represented as HTML document is straightforward. One first tries to find the left and right anchor for the authors' name(s) and then, to the right, the nearest left and right anchor for the year of publication. This is repeated as long as those pairs of anchors can be found.

Several approaches [8, 14, 15] in the IE community use this basic idea to define extraction procedures \((\text{wrappers or templates})\) based on their own description language. Further, investigations showed that wrappers can be classified according to their expressiveness based on several structural constraints. This leads to the general question whether or not such regularly structured descriptions can automatically be constructed and furthermore learned. A rule based way to define a wrapper for the previously discussed example is presented in Figure 2.

Here, by capital letters we denote variables, terminal symbols are typeset in italics. The first rule can be interpreted as follows: An author \(A\) and the year of publication \(Y\) can be extracted from a document \(D\) in case that \((i)\) \(D\) matches the pattern \(X_1 L_1 R \alpha X_2 L_2 Y R \beta X_3\) and \((ii)\) the instantiations of the variables meet certain constraints. For example, the constraint \(R_1\) states that the variable \(R_1\) can only be replaced by the string \(./\) or the string \(./(i)\). Further constraints like \(\text{nc-}r_1(A)\) explicitly state which text segments are not suited to be substituted for the variable \(A\) (cf. rules 2–5 and 15–16). In this particular case, text segments that contain the substrings \(./\) or \(./(i)\) are not allowed. If a document \(D\) matches the pattern \(X_1 L_1 R \alpha X_2 L_2 Y R \beta X_3\) and if all specified constraints are fulfilled, then the instantiations of the variables \(A\) and \(Y\) yield the information required.

\[
\text{extract}(A, Y, X_1 L_1 R \alpha X_2 L_2 Y R \beta X_3) \leftarrow \text{L}_1(A_1), \text{nc-}r_1(A), r_1(R_1), \\
\text{nc-}l_2(X_1), \\
\text{L}_2(A_2), \text{nc-}r_2(Y), r_2(R_2). \\
\text{nc-}r_1(X) \leftarrow \text{not-}r_1(X). \\
\text{c-}r_1(X) \leftarrow r_1(X). \\
\text{c-}r_1(Y) \leftarrow c-\gamma_1(X). \\
\text{c-}r_1(Y) \leftarrow c-\gamma_1(Y). \\
\text{nc-}l_2(X) \leftarrow \text{not-}l_2(X). \\
\text{c-}l_2(X) \leftarrow l_2(X). \\
\text{c-}l_2(Y) \leftarrow c-\gamma_2(X). \\
\text{c-}l_2(Y) \leftarrow c-\gamma_2(Y). \\
\text{nc-}r_2(X) \leftarrow \text{not-}r_2(X). \\
\text{c-}r_2(X) \leftarrow r_2(X). \\
\text{c-}r_2(Y) \leftarrow c-\gamma_2(X). \\
\text{c-}r_2(Y) \leftarrow c-\gamma_2(Y). \\
\text{L}_2(/(i).) \\
\text{r}_2(/(i).) \\
\text{r}_2(/(i).) \\
\text{r}_2(/(i).) \\
\text{r}_2(/(i).) \\
\text{r}_2(/(i).)
\]

**Figure 2.**
The aim of this paper is twofold. We propose a uniform way to represent wrappers, namely advanced elementary formal systems (AEFSs, for short). This concept is generalizing SMULLYAN's [12] elementary formal systems (EFSs) which have thoroughly been studied in different respects. We investigate the expressive power of AEFSs and show that it is sufficient for wrapper representation though its semantics is still reasonable. To lay a cornerstone of our application-oriented work, we investigate which learnability results achieved for EFSs lift to AEFSs. Additionally, we prototypically show how AEFSs can be used to describe a certain class of HTML wrappers, so-called island wrappers (cf. [14,15]). We prove the learnability of island wrappers from only positive examples under certain natural assumptions.

3 Advanced Elementary Formal Systems

In this section, we introduce a quite general formalism to describe wrappers, namely advanced elementary formal systems (AEFSs, for short). In addition, we study the expressive power of AEFSs and deal with the question of whether or not AEFSs can be learned from examples.

AEFSs generalize SMULLYAN's [12] elementary formal systems (EFSs, for short) which he introduced to develop his theory of recursive functions. In the last years, the learnability of EFSs has intensively been studied within several formal frameworks (cf. [4,16,3,5,11,17,10]).

3.1 Elementary Formal Systems

Next, we provide notions and notations that allow for a formalization of EFSs.

Assume three mutually disjoint sets — a finite set $\Sigma$ of characters, a finite set $\Pi$ of predicates, and an enumerable set $X$ of variables. We call every element in $(\Sigma \cup X)^+$ a pattern and every string in $\Sigma^+$ a ground pattern. For a pattern $\pi$, we let $v(\pi)$ be the set of variables in $\pi$.

Let $p \in \Pi$ be a predicate of arity $n$ and let $\pi_1, \ldots, \pi_n$ be patterns. Let $A = p(\pi_1, \ldots, \pi_n)$. Then, $A$ is said to be an atomic formula (an atom, for short). $A$ is ground, if all the patterns $\pi_i$ are ground. Moreover, $v(A)$ denotes the set of variables in $A$.

Let $A$ and $B_1, \ldots, B_n$ be atoms. Then, $r = A \leftarrow B_1, \ldots, B_n$ is a rule, $A$ is the head of $r$, and all the $B_i$ form the body of $r$. Then, $r$ is a ground rule, if all atoms in $r$ are ground. Additionally, if $n = 0$, then $r$ is called a fact and sometimes we write $A$ instead of $A \leftarrow$.

Let $\sigma$ be a non-erasing substitution, i.e., a mapping from $X$ to $\Sigma^+$. For any pattern $\pi$, $\pi\sigma$ is the pattern which one obtains when applying $\sigma$ to $\pi$. Let $C = p(\pi_1, \ldots, \pi_n)$ be an atom and let $r = A \leftarrow B_1, \ldots, B_n$ be a rule. Then, we set $C\sigma = p(\pi_1\sigma, \ldots, \pi_n\sigma)$ and $r\sigma = A\sigma \leftarrow B_1\sigma, \ldots, B_n\sigma$. If $r \sigma$ is ground, then it is said to be a ground instance of $r$.

**Definition 1** ([5]) Let $\Sigma$, $\Pi$, and $X$ be fixed, and let $\Gamma$ be a finite set of rules. Then, $S = (\Sigma, \Pi, \Gamma)$ is said to be an EFS.
EFSs can be considered as particular logic programs without negation. There are two major differences: (i) patterns play the role of terms and (ii) unification has to be realized modulo the equational theory
\[ E = \{ o(x, o(y, z)) = o(o(x, y), z) \}, \]
where \( o \) is interpreted as concatenation of patterns.

As for logic programs (cf., e.g., [9]), the semantics of an EFS \( S \) can be defined via the following operator \( T_S \). In the corresponding definition, we use the following notations. For any EFS \( S = (\Sigma, \Pi, \Gamma) \), \( B(S) \) denotes the set of all well-formed ground atoms over \( \Sigma \) and \( \Pi \), and \( G(S) \) denotes the set of all ground instances of rules in \( \Gamma \).

**Definition 2** Let \( S \) be an EFS and let \( I \subseteq B(S) \). Then, we let \( T_S(I) = I \cup \{ A \mid A \leftarrow B_1, \ldots, B_n \in G(S) \text{ for some } B_1 \in I, \ldots, B_n \in I \} \).

Note that, by definition, the operator \( T_S \) is idempotent and monotonic.

As usual, we let \( T^{n+1}_S(I) = T_S(T^n_S(I)) \), where \( T^0_S(I) = I \), by convention.

**Definition 3** Let \( S \) be an EFS. Then, we let \( \text{Sem}(S) = \bigcup_{n \in \mathbb{N}} T^n_S(\emptyset) \).

In general, \( \text{Sem}(S) \) is semi-decidable, but not decidable. However, as we will see below, \( \text{Sem}(S) \) turns out to be decidable in case that \( S \) meets several natural syntactical constraints (cf. Theorem 5).

Finally, by \( E \) we denote the collection of all EFSs.

When using EFSs to describe wrappers, a problem that usually occurs is that one has to put constraints on the patterns that form admissible substitutions for particular variables (cf., e.g., the wrapper in Figure 2). One approach to cope with this problem is to explicitly describe the admissible patterns. In some cases, it is more convenient to explicitly describe the exceptions and to postulate that every pattern that does not serve as exception is admissible. In contrast to EFSs, AEFSs provide enough flexibility to realize the latter approach, as well.

### 3.2 Beyond Elementary Formal Systems

Informally speaking, an AEFS is an EFS that may additionally contain rules of the form \( A \leftarrow \neg B_1 \), where \( A \) and \( B_1 \) are atoms and \( \neg \) stands for negation as finite failure. The underlying meaning is as follows. If, for instance, \( A = p(x_1, \ldots, x_n) \) and \( B_1 = q(x_1, \ldots, x_n) \), then the predicate \( p \) succeeds if the predicate \( q \) fails.

However, taking the conceptual difficulties into consideration that occur when defining the semantics of logic programs with negation as finite failure (cf., e.g., [9]), AEFSs are constrained to meet several additional syntactic requirements (cf. Definition 4). The requirements posed guarantee that, similarly to stratified logic programs (cf., e.g., [9]), the semantics of AEFSs can easily be described. Moreover, as a side-effect, the requirements posed guarantee that AEFSs inherit some of the convenient properties of EFSs (cf., e.g., Theorem 5).

Before formally defining how AEFSs look like, we need some more notations. Let \( \Gamma \) be a set of rules (including rules of the form \( A \leftarrow \neg B_1 \)). Then, \( hp(\Gamma) \) denotes the set of predicates that appear in the head of any rule in \( \Gamma \).
Definition 4 AEFSs and their semantics are defined according to the following rules. Let $S'$ = $(\Sigma, \Pi', \Gamma')$ be an EFS. Moreover, let $S_1 = (\Sigma, \Pi_1, \Gamma_1)$ and $S_2 = (\Sigma, \Pi_2, \Gamma_2)$ be AEFSs.

1. $S = S'$ is an AEFS. Moreover, we let $\text{Sem}(S) = \text{Sem}(S')$.
2. If $\Pi_1 \cap \Pi_2 = \emptyset$, then $S = (\Sigma, \Pi_1 \cup \Pi_2, \Gamma_1 \cup \Gamma_2)$ is an AEFS. Moreover, we let $\text{Sem}(S) = \text{Sem}(S_1) \cup \text{Sem}(S_2)$.
3. Let $p \notin \Pi_1$ and $q \in \Pi_1$ be predicates of arity $n$. Then, $S = (\Sigma, \Pi_1 \cup \{p\}, \Gamma_1 \cup \{p(x_1, \ldots, x_n) \leftarrow \neg q(x_1, \ldots, x_n)\})$ is an AEFS. Moreover, we let $\text{Sem}(S) = \text{Sem}(S_1) \cup \{p(s_1, \ldots, s_n) \mid s_i \in \Sigma^+, \ldots, s_n \in \Sigma^+, q(s_1, \ldots, s_n) \notin \text{Sem}(S_1)\}$.
4. Let $hp_1(\Gamma') \cap \Pi_1 = \emptyset$. Then, $S = (\Sigma, \Pi'_1 \cup \Pi_1, \Gamma'_1 \cup \Gamma_1)$ is an AEFS. Moreover, we let $\text{Sem}(S) = \bigcup_{p \in \Pi} T^p_1(\text{Sem}(S_1))$.

Finally, by $\text{AE}$ we denote the collection of all AEFSs.

Having a closer look at Figure 2, one realizes that AEFSs can be used to describe interesting wrappers.

3.3 Expressiveness

In the following, we show how AEFSs can be used to describe formal languages and relate the resulting language classes to the language classes of the classical CHOMSKY hierarchy (cf. [7]). Although we are mainly interested in using AEFSs for describing wrappers, we strongly believe that the established relations are quite helpful to better understand the expressive power of AEFSs.

Definition 5 Let $S = (\Sigma, \Pi, \Gamma)$ be an AEFS and let $p \in \Pi$ be a unary predicate. We let $L(S, p) = \{s \mid p(s) \in \text{Sem}(S)\}$.

Intuitively speaking, $L(S, p)$ is the language which the AEFS $S$ defines via the unary predicate $p$.

Definition 6 Let $M \subseteq \text{AE}$. Then, the set $\mathcal{L}(M)$ of all languages that are definable with AEFSs in $M$ contains every language $L$ for which there are an AEFS $S = (\Sigma, \Pi, \Gamma)$ in $M$ and some unary predicate $p \in \Pi$ such that $L = L(S, p)$.

For example, $\mathcal{L}(\text{AE})$ is the class of all languages that are definable by AEFSs.

Our first result puts the expressive power of AEFSs into the right perspective. Let $\mathcal{L}_{re}$. be the class of all recursively enumerable languages.

Theorem 1 $\mathcal{L}_{re} \subseteq \mathcal{L}(\text{AE})$.

Moreover, the following closedness properties can be shown.

Theorem 2 $\mathcal{L}(\text{AE})$ is closed under the operations union, intersection, complement, and concatenation.

To elaborate a more accurate picture, similarly to [3], we next introduce several constraints on the structure of the rules an EFS resp. AEFS may contain.

Let $r$ be a rule of form $A \leftarrow not B_i$ or $A \leftarrow B_i, \ldots, B_n$. Then, $r$ is said to be variable-bounded iff, for all $i \leq n$, $v(B_i) \subseteq v(A)$. Moreover, $r$ is said to be
length-bounded if, for all substitutions \( \sigma \), \(|A\sigma| \geq |B_1\sigma| + \cdots + |B_n\sigma|\). Clearly, if \( r \) is length-bounded, then \( r \) is also variable-bounded. Note that, in general, the opposite does not hold. Finally, \( r \) is said to be simple if \( A \) is of form \( p(\pi) \) and \( \pi \) is a pattern in which every variable occurs at most once.

**Definition 7** Let \( S = (\Sigma, \Pi, \Gamma) \) be an AEFS. Then, \( S \) is said to be variable-bounded if all \( r \in \Gamma \) are variable-bounded. Moreover, \( S \) is said to be length-bounded if all \( r \in \Gamma \) are length-bounded. Next, \( S \) is said to be regular if \( \Pi \) contains exclusively unary predicates and all \( r \in \Gamma \) are length-bounded as well as simple. Finally, by \( vb\text{-AE} (vb\text{-E}) \), \( lb\text{-AE} (lb\text{-E}), \) and \( reg\text{-AE} (reg\text{-E}) \) we denote the collection of all variable-bounded, length-bounded, and regular AEFSs (EFSs).

Now, similarly to Theorem 2, the following result can be established.

**Theorem 3** \( L(vb\text{-AE}), L(lb\text{-AE}), \) and \( L(reg\text{-AE}) \) are closed under the operations union, intersection, complement, and concatenation.

The next theorem summarizes the announced relations to some important language classes of the Chomsky hierarchy (cf. [7]). Here, \( L_{cs} \) and \( L_{cf} \) denote the class of all context sensitive and context free languages, respectively.

**Theorem 4**
1. \( L_{r,e} \subseteq L(vb\text{-AE}) \).
2. \( L_{cs} = L(lb\text{-AE}) \).
3. \( L_{cf} \subseteq L(reg\text{-AE}) \subseteq L_{cs} \).

Assertion (2) of the latter theorem allows for the following insight:

**Theorem 5** If \( S \in lb\text{-AE} \), then \( \text{Sem}(S) \) is decidable.

Note that, for length-bounded EFSs, the equivalent of Theorem 5 has already been shown in [5]. Moreover, for EFSs, Theorem 4 rewrites as follows:

**Theorem 6** ([5])
1. \( L_{r,e} = L(vb\text{-E}) \).
2. \( L_{cs} = L(lb\text{-E}) \).
3. \( L_{cf} = L(reg\text{-E}) \).

### 3.4 Learnability

At the end of this section, we present some basic results concerning the question of whether or not AEFSs can be learned from examples. To be more precise, we study the learnability of several language classes that are definable by AEFSs within the learning model introduced by Gold [6]. As we shall see, the results obtained provide a firm basis to derive answers to the question to what extent, if ever, HTML wrappers can automatically be synthesized from examples.
Let us briefly review the necessary basic concepts. We refer the reader to [2] and [18] which contain all missing details concerning Gold's [6] model.

There are several ways to present information about formal languages to be learned. The basic approaches are defined via the concept text and informant, respectively. A text is just any sequence of words exhausting the target language. An informant is any sequence of words labelled alternatively by 1 or 0 such that all the words labelled by 1 form a text whereas the remaining words labelled by 0 constitute a text of the complement of the target language.

Now, an algorithmic learner (henceforth, called IIM) receives as its inputs larger and larger initial segments of a text \( t \) [an informant \( i \)] for a target language \( L \) and generates as its outputs hypotheses. In our setting, an IIM is supposed to generate AEFSs resp. EFSs as hypotheses. An IIM learns a target language \( L \) from text \( t \) [informant \( i \)], if the sequence of its outputs stabilizes on an AEFS which correctly describes \( L \). Now, an IIM is said to learn \( L \) from text [informant], if it learns \( L \) from every text [every informant] for it. Furthermore, some language class \( C \) is said to be learnable from text [informant], if there is an IIM which learns every language \( L \in C \) from text [informant].

Next, we summarize the established learnability and non-learnability results.

**Theorem 7**

1. \( L(\text{vb-} \mathcal{AE}) \) is not learnable from informant.
2. \( L(\text{vb-} \mathcal{AE}) \) is learnable from informant.

**Proof.** By Theorem 4, \( L_{\text{cf}} \subseteq L(\text{vb-} \mathcal{AE}) \). Since there is no learning algorithm which is capable to learn the class \( L_{\text{cf}} \) of all recursive languages from informant (cf. [6]), we obtain (1) because of \( L_{\text{cf}} \subseteq L_{\text{cf}} \). Furthermore, since \( L(\text{vb-} \mathcal{AE}) = L_{\text{cf}} \), we know that \( L(\text{vb-} \mathcal{AE}) \) constitutes an effectively enumerable class of recursive languages. Hence, we get (2), since every effectively enumerable class of recursive languages is learnable from informant (cf. [6]). \( \square \)

Based on weaker information, if exclusively positive examples are available, only relatively small language classes turn out to be learnable at all. Next, for all \( k \geq 1 \), we let \( \text{vb-} \mathcal{AE}^k \) (\( \text{vb-} \mathcal{E}^k \)) denote the collection of all AEFSs (EFSs) which consists of at most \( k \) rules.

**Theorem 8**

1. For all \( k \geq 2 \), \( L(\text{vb-} \mathcal{AE}^k) \) is not learnable from text.
2. \( L(\text{vb-} \mathcal{AE}^1) \) is learnable from text.

**Proof.** Since, by definition, \( L(\text{vb-} \mathcal{AE}^1) = L(\text{vb-} \mathcal{E}^1) \), (2) is a special case of Assertion (2) in Theorem 9. It remains to verify (1).

Let \( \Sigma = \{a\} \), let \( L_0 = \{a\}^+ \) and, for all \( j \geq 1 \), let \( L_j = L_0 \setminus \{a^j\} \). It is folklore that there is no learning algorithm that is able to learn all languages in \( C = \{L_j \mid j \in \mathbb{N}\} \) from positive data (cf., e.g., [18], for the relevant details). It suffices to show that \( C \subseteq L(\text{vb-} \mathcal{AE}^2) \). This can be seen as follows. In case of \( j = 0 \), let \( S_0 = (\Sigma, \{p\}, \{p(x) \leftarrow \}) \). Clearly, \( L(S_0, p) = L_0 \). Next, let \( j \geq 1 \). Then, \( S_j = (\Sigma, \{p, q\}, \{q(a^j) \leftarrow, p(x) \leftarrow \neg q(x)\}) \). Clearly, \( L(S_j, p) = L_j \). \( \square \)
For EFSs, the equivalent of Theorem 7 holds, as well. In contrast, Theorem 8 rewrites as follows.

**Theorem 9 ([11])**

1. $\mathcal{L}(\mathfrak{b}-\mathcal{E})$ is not learnable from text.
2. For all $k \geq 1$, $\mathcal{L}(\mathfrak{b}-\mathcal{E}^k)$ is learnable from text.

## 4 Wrappers for HTML Documents

Next, we demonstrate that AEFSs provide an appropriate framework to describe wrappers of practical relevance. Henceforth, the corresponding wrappers are called island wrappers. As a main result, we show under which assumptions learning techniques can be invoked to automatically generate island wrappers from positive examples.

Semi-structured documents carry different information and, moreover, the relevance of the information naturally depends on the users’ perspective. As in most IE approaches, we assume that the content of a semi-structured document $D$ (more formally, its semantics) is a set of tuples which $D$ contains. For example, the tuples $(D, \text{Angluin}, 80), (D, \text{Angluin and C.H. Smith}, 83), \ldots$ form relevant information in the list of references of the present paper (cf. Figure 1). Now, the aim of the IE task is to provide a wrapper which allows one to extract all tuples of this kind from this document as well as from all similar documents. In contrast to rather traditional approaches to IE in which it is the users’ task to construct the relevant wrappers, we are interested in algorithms that automatically synthesize appropriate wrappers from examples.

### 4.1 Semantics of HTML Documents

Let $D \in \Sigma^+$ be a document. The information which $D$ contains is a finite set of tuples of strings $(s_1, \ldots, s_n)$ where, as a rule, each of these strings must occur in $D$. Together with a tuple $(s_1, \ldots, s_n)$, it is important to know to which subword in $D$ a string $s_i$ is referring to. For instance, consider the tuple $(s_1, s_2) = (\text{B. Thomas}, 99)$ in the list of references. Then, it might be intended that $(s_1, s_2)$ has its origin either completely in reference [14] or completely in [15]. It is rather unlikely that $s_1$ belongs to [14] and that $s_2$ belongs to [15].

More formally speaking, the semantics of a document $D$ is more than a set of tuples that describe the information contained. The semantics of $D$ depends on an interpretation $I$ which relates the strings in the tuples to subwords in $D$. More formally speaking, a function $S : \Sigma^+ \rightarrow \wp((\Sigma^+)^n)$ is a semantics iff there is an interpretation $I$ such that for all $D \in \Sigma^+$ and for all $(s_1, \ldots, s_n) \in S(D)$ the Conditions (1) and (2) are fulfilled, where

1. $I((s_1, \ldots, s_n), D)$ describes at which positions the $s_i$ begin in $D$.
2. $s_{i+1}$ begins in $D$ after $s_i$ ends in $D$. 
Intuitively speaking, if examples are provided that illustrate a certain semantics $S$, a learning algorithm is supposed to learn a wrapper that implements $S$, i.e., the wrapper must allow for the extraction of all tuples in $S(D)$ for any document $D$.

4.2 Island Wrappers

Island wrappers generalize the structure of the wrapper that has been designed to extract certain information from the HTML representation of the list of references (cf. Figure 2).

Island wrappers are AEFSs that consist of several basic length-bounded AEFSs describing anchor languages and of a couple of fixed top level rules that determine the interplay between these basic AEFSs (cf. Figure 3). To be more precise, consider an island wrapper that is designed to extract $n$-tuples from a given HTML document. For every extraction variable $V_i$, there have to be basic AEFSs $S_{l_i}$ and $S_{r_i}$ that define the left and right anchor languages $L_{l_i}$ and $L_{r_i}$ via the unary predicates $p_{l_i}$ and $p_{r_i}$, i.e., $L(S_{l_i}, p_{l_i}) = L_{l_i}$ and $L(S_{r_i}, p_{r_i}) = L_{r_i}$.

As a rule, it is assumed that the set of predicate symbols used in the basic AEFSs $S_{l_i}$ and $S_{r_i}$ have to be mutually disjoint. Now, intuitively, a string $s_i$ can be substituted for the extraction variable $V_i$ only in case that the actual document $D$ contains a substring $u_i \circ s_i \circ u_i$ that meets $u_i \in L_{l_i}$ and $u_i \in L_{r_i}$.

Thus, the anchor languages put constraints on the surroundings in which the relevant strings $s_i$ are embedded in $D$. Moreover, as argued at the beginning of the last subsection, further minimality constraints are necessary. The strings substituted for the extraction variables must be as short as possible, while the distance between them has to be as small as possible. The top level rules needed are depicted in Figure 3. Let $p_{l_1}, p_{l_2}, \ldots, p_{l_n}, p_{l_{n+1}}, p_{r_1}, p_{r_2}, \ldots, p_{r_n}$ be predicates not occurring in all the basic AEFSs $S_{l_i}$ and $S_{r_i}$.

\begin{verbatim}
\begin{align*}
\forall (V_1, V_2, \ldots, V_n, L_1, L_2, R_1, R_2, \ldots, L_n, R_n) \Rightarrow & \quad p_{l_1}(L_1), \quad p_{l_2}(V_1), \quad p_{r_1}(R_1), \\
& \quad p_{l_2}(L_2), \quad p_{r_2}(V_2), \quad p_{r_1}(R_2), \\
& \quad \ldots, \\
& \quad p_{l_n}(L_n), \quad p_{r_n}(V_n), \quad p_{r_n}(R_n).
\end{align*}
\end{verbatim}

\begin{verbatim}
\begin{align*}
& \quad \text{nc-}p_{l_1}(X) \leftarrow \text{not } c-p_{l_1}(X), \\
& \quad \text{nc-}p_{l_2}(X) \leftarrow p_{l_1}(X), \\
& \quad \text{nc-}p_{r_1}(X) \leftarrow p_{r_2}(X), \\
& \quad \text{nc-}p_{r_2}(X) \leftarrow c-p_{r_1}(X), \\
& \quad \text{nc-}p_{r_3}(X) \leftarrow c-p_{r_2}(X), \\
& \quad \ldots
\end{align*}
\end{verbatim}

Figure 3.

Formally speaking, an island wrapper is an AEFS $S_{iu} = (\Sigma, \Pi, \Gamma)$, where $\Pi$ is the collection of all predicates that occur either in rules in Figure 3 or in rules belonging to the basic AEFSs $S_{l_i}$ and $S_{r_i}$ and $\Gamma$ contains all rules in Figure 3 and all rules in the basic AEFSs $S_{l_i}$ and $S_{r_i}$. As a matter of fact, note that every $S_{iu}$ is length-bounded. Hence, by Theorem 5, $Sem(S_{iu})$ is decidable. The latter makes island wrappers particularly tailored for IE.
In order to use an island wrapper $S_{iw}$ to extract information from a document $D$, all the $n$-tuples $(s_1, \ldots, s_n)$ that meet $w(s_1, \ldots, s_n, D) \in \text{Sem}(S_{iw})$ have to be computed. Since all the $s_i$ are substrings of $D$, this can be done effectively. An island wrapper $S_{iw}$ defines a particular view at the document $D \in \Sigma^+$, namely $\text{view}(S_{iw}, D) = \{(s_1, \ldots, s_n) \mid w(s_1, \ldots, s_n, D) \in \text{Sem}(S_{iw})\}$. Furthermore, the island wrapper $S_{iw}$ implements a particular semantics $S$ iff $\text{view}(S_{iw}, D) = S(D)$.

Having the non-learnability results from Subsection 2.4 in mind it is unrealistic to assume that the class of all island wrappers is learnable from positive examples. For a better understanding of the principal power and limitations of the learning approach under consideration, we provide some finer look at island wrappers by putting some natural constraints on the admissible anchor languages. By $\mathcal{AE}_{iw}$ we denote the collection of all length-bounded island wrappers. For all $k \in \mathbb{N}$, $\mathcal{AE}_{iw}^k$ is the set of all island wrappers in $\mathcal{AE}_{iw}$ that are built upon anchor languages $L$ with $\text{card}(L) \leq k$.

### 4.3 Learning Island Wrappers from Marked Text

Now, we are ready to study the question under which assumptions island wrappers can be learned from positive examples. As defined above, island wrappers differ in their anchor languages only. Hence, the overall learning task reduces to the problem to find AEFSs which describe the relevant anchor languages.

However, a potential user does not provide elements of the anchor languages to the system. Instead, the user marks interesting information in the HTML documents under inspection. To illustrate this, consider again the semi-structured representation of the list of references. In Figure 4, the information the user is interested in is underlined.

```
<html><body>
<h1>References</h1>
<ol>
<li>D. Angluin, "Inductive inference of formal languages from positive data", (1980).\cite{1}
<li>S. Arikawa, ...
</li></ol>
</body></html>
```

Figure 4.

Clearly, the marked document provides only implicit information concerning the anchor languages to be learned. For instance, it can easily be deduced that the left anchor language $L_1$, contains a string that forms a suffix of the text segment $\langle$/html$/\langle$/body$/\langle$/h1$/\langle$/ol$/\langle$/li$/\langle$/a$/\rangle$/\rangle$/\rangle$/\rangle$/\rangle$ and a (possibly) different string that is a suffix of the text segment $\langle$/li$/\langle$/li$/\rangle$/\rangle$. Moreover, the right anchor language $L_2$, must contain at least one string that is a prefix of the text segment $\langle$/li$/\langle$/li$/\rangle$/\rangle$.

To be precise, the learner does not receive a text for any of the relevant anchor languages as input. Therefore, the results from Subsection 2.4 do only translate indirectly – after an appropriate adaptation – to our setting of learning wrappers from marked texts.
Let $S$ be a semantics under some fixed interpretation $I$. A marked text $t$ for $S$ is any sequence of pairs $(D, P)$ fulfilling the following conditions:

1. Each $D$ is a document from $\Sigma^+$.
2. $P$ is a finite set of pairs $(s, p)$, where $s = (s_1, \ldots, s_n)$ is an $n$-tuple in $S(D)$ and $p = (p_1, \ldots, p_n)$ describes where the $s_i$ begin in $D$, i.e., $p = I(s, D)$.
3. The sequence is exhaustive, i.e., for every document $D \in \Sigma^+$, every $n$-tuple in $S(D)$ eventually appears in $t$.

Now, a wrapper learner (henceforth called WIM) receives as input larger and larger initial segments of a marked text for some target semantics and generates as outputs AEFSs describing wrappers. A WIM is said to learn a semantics if the sequence of its outputs stabilizes on a wrapper which implements the target semantics. Finally, a WIM learns a class $C$ of wrappers if it learns every semantics that is implementable by a wrapper in $C$.

Our first result points out the general limitations of wrapper induction.

**Theorem 10** The class $\mathcal{AE}_{iw}$ is not learnable from marked text.

*Proof.* Consider the following quite simple collection of island wrappers for extracting 2-tuples. Let $\Sigma = \{a, b\}$, let $\Gamma$ be the set of rules depicted in Figure 3 for the case of $n = 2$, and let $\Pi$ be the set of all predicates used in $\Gamma$. Now, we set $\Gamma_1 = \{p_1(a)\} \cup \{p_3(a)\}$, and $\Gamma_4 = \{p_2(\ell)\}$. Additionally, for every $j \in \mathbb{N}$, we set $\Gamma_2 = \{p_1(a)\} \cup \{p_3(aX)\}$ as well as $\Gamma_3 = \{p_1(\ell)\} \cup \{p_3(aX)\}$. By definition, all island wrappers $S_j$ do only differ in the left anchor language of the first extraction variable $V_1$.

Finally, we claim that the collection $\{S_j \mid j \in \mathbb{N}\}$ is not learnable from marked text. This can be shown by applying arguments similarly to those used in [6] for proving that superfinite language classes are not learnable from ordinary text. We omit the details.

In contrast, if there is a uniform bound on the cardinality of the relevant anchor languages, learning becomes possible.

**Theorem 11** For all $k \geq 1$, the class $\mathcal{AE}_{iw}^k$ is learnable from marked text.

*Proof.* Let $S_{iw} \in \mathcal{AE}_{iw}^k$ be an island wrapper to extract $n$-tuples and let $t = (D_1, P_1), (D_2, P_2), \ldots$ be a marked text for $S_{iw}$ under some fixed interpretation. We claim that the following WIM $M$ learns $S_{iw}$ when successively fed $t$.

When learning an island wrapper from marked text, one may proceed as follows: In a first step, decompose the overall learning problem into several problems of learning anchor languages from ordinary text. In a second step, solve the derived individual learning problems independently and in parallel. In a concluding step, the solutions of the individual problems are combined to formulate a solution for the overall learning problem.

There are three different types of individual learning problems to attack. One problem consists in learning the left anchor language of the first extraction.
variable \( V_i \) (type A), another one in learning the right anchor language of the last variable \( V_n \) (type C). Moreover, there are \( n-1 \) learning problems of type B that consists in simultaneously learning the right anchor language of a variable \( V_i \) and the left anchor languages of a variable \( V_{i+1} \) (\( 1 \leq i \leq n-1 \)).

**WIM M**: On input \((D_1, P_1), \ldots, (D_m, P_m)\) do the following.

Set \( t^0, \ldots, t^n \) to be the empty sequences.

For each \( j = 1, \ldots, m \), do the following:

For each pair \((\{s_1, \ldots, s_n\}, \{p_1, \ldots, p_n\})\) in \( P_j \), compute (identified by the given starting positions \( p_1, \ldots, p_n \) of \( s_1, \ldots, s_n \)) the uniquely determined substrings \( w_0, w_1, \ldots, w_n \) of \( D_j \) such that \( D_j = w_0 s_1 w_1 s_2 \ldots w_{n-1} s_n w_n \).
Append \( w_0 \) to \( t^0 \), \( w_1 \) to \( t^1 \), \ldots, and \( w_n \) to \( t^n \).

Let \( \Gamma \) be the rules depicted in Figure 3 and \( \Pi \) be the set of all predicates used to formulate the rules in \( \Gamma \). Do in parallel:

- On input \( t^0 \), run \( M_A \) — an IHM for problems of type A. Fix \( \Gamma^I = M_A(t^0) \).
  \( T_i \) is built by replacing, everywhere in \( \Gamma^I \), the predicate \( p \) by \( p_i \).
- For \( i = 1, \ldots, n-1 \), compute in parallel:
  On input \( t^i \), run \( M_B \) — an IHM for problems of type B. Fix \( \Gamma^I = M_B(t^i) \).
  \( T_i \) is built by replacing, everywhere in \( \Gamma^I \), the predicates \( p \) and \( q \) by \( p_i \) and \( p_{i+1} \), respectively.
- On input \( t^n \), run \( M_C \) — an IHM for problems of type C. Fix \( \Gamma^I = M_C(t^n) \).
  \( T_i \) is built by replacing, everywhere in \( \Gamma^I \), the predicate \( p \) by \( p_\star \).

Output the EFS \( (\Sigma, \Pi, \Gamma^I) \) with \( \Gamma^I = \Gamma \cup \Gamma_0 \cup \Gamma_1 \cup \ldots \cup \Gamma_n \).

The IHM \( M_A \) for learning problems of type A is defined as follows.

**IHM M_A**: On input \( S = u_0, \ldots, u_k \) do the following:

Set \( T^I = \emptyset \). Determine the set \( E \) of all non-empty suffixes of strings in \( S \).
For all strings \( e \in E \) check whether or not, for all \( a \in \Sigma \), \( u = a \circ e \) for some \( u \in S \). Let \( T \) be the set of all strings \( e \) passing this test. Goto (a1).

(a1) If \( T = \emptyset \), output \( \Gamma^I \). Otherwise, goto (a2).

(a2) Determine a shortest string \( e \in T \). Set \( \Gamma^I = \Gamma^I \cup \{p(e)\} \) and \( T = T \setminus T_i \), where \( T_i \) contains all strings in \( T \) with the suffix \( e \). Goto (a1).

The IHM \( M_C \) can be obtained from \( M_A \) by replacing everywhere the term suffix by prefix. It is not hard to see that \( M_A \) and \( M_C \) behave as required.

It remains to define an IHM for learning problems of type B. In the definition of \( M_B \), we let \( \Pi^U = \{nc-q(X)\circ e\circ c-q(X), c-q(X)\circ e\circ c-q(X), c-q(X)\circ e\circ c-q(Y), e\circ c-q(X), c-q(X)\circ e\circ c-q(Y), r(XY Z)\circ e\circ p(X), nc-q(Y), q(Z)\} \) and \( \Pi^H = \{p, q, nc-q, c-q, r\} \).

**IHM M_B**: On input \( S = u_0, \ldots, u_k \) do the following:

Let \( B \) and \( E \) be the set of all non-empty prefixes and suffixes of strings in \( S \).
Moreover, let \( B^I = \{p(b)\circ e\mid b \in B \} \) and \( E^I = \{q(e)\circ e\mid e \in E \} \).

Let \( H \) be the collection of all sets \( h \subseteq B^I \cup E^I \) such that \( h \) contains at most \( k \) rules from \( B^I \) and at most \( k \) rules from \( E^I \).

Search for an \( h \in H \) such that, for every \( u \in S \), \( r(u) \in Sem(S) \) holds, where \( S \) is the EFS \( (\Sigma, \Pi^H, \Gamma^H \cup h) \). If such an \( h \) is found, let \( \Gamma^I \) be the lexicographically first of them. Otherwise, set \( \Gamma^I = \emptyset \). Output \( \Gamma^I \).
The verification of $M_B$'s correctness is a little more involved. Due to the space constraints, the details have to be omitted.

5 Conclusions

The conventional concept of so-called Elementary Formal Systems has been generalized to Advanced Elementary Formal Systems (AEFS) which have been proven sufficiently expressive for representing certain wrappers of practical relevance. So-called island wrappers are appropriate for representing classes of HTML documents under a particular perspective. Island wrappers are characterized as length-bounded AEFSs.

The user is directed to http://LEXIKON.dfki.de, where illustrative examples can be found which demonstrate the usefulness of the present approach.

Learnability of formal languages is known to be hard. Even island wrappers are not automatically learnable from examples only. This is throwing some light at the difficulties of invoking learning techniques for information extraction from the Internet.

The authors introduced additional constraints on the families of anchor languages which are induced by island wrappers and investigated the impact of these constraints on learnability. Anchor languages meeting such a constraint turn out to be learnable from positive examples drawn from given semi-structured documents. Thus, their corresponding island wrappers are learnable.

The results about representability and learnability above are setting the stage for some specific approach towards automated information extraction from the Internet. The basic scenario is as follows.

Given a sample Internet document, a user might have a particular view at the document and at the information contained therein which is relevant to him (her). By marking text passages from this document, the user is specifying this view in some detail. Marked text passages result in so-called marked text, which is a formal concept within the underlying theoretic setting. If sufficiently expressive examples have been marked, the intended view can be learned automatically. This is done by inductive inference of island wrappers which are particular AEFSs. Any learned island wrapper does not only allow to extract information from the documents it has been synthesized upon, but it also applies to a potentially unlimited number of further documents not seen so far.

The results of the present paper justify scenarios of this type and prove that information extraction through inductive learning does really work. Several problems are left to future investigations. Among these, there is the question for particularly efficient algorithms and the question for generalization to hierarchically structured document sources.

Finally, note that the representation formalism developed is well suited to describe other wrapper classes of practical relevance. For instance, the wrapper classes from [8] can easily be described and their learnability can be studied within this formal framework.
References

Discovery of Web Communities Based on the Co-occurrence of References

Tsuyoshi Murata
Department of Computer Science, Gunma University
1-5-1 Tenjin-cho, Kiryu, Gunma, 370-8515 Japan
murata@cs.gunma-u.ac.jp

Abstract. This paper proposes a method of discovering Web communities. A complete bipartite graph $K_{i,j}$ of Web pages can be regarded as a community sharing a common interest. Discovery of such community is expected to assist users’ information retrieval from the Web. The method proposed in this paper is based on the assumption that hyperlinks to related Web pages often co-occur. Relations of Web pages are detected by the co-occurrence of hyperlinks on the pages which are acquired from a search engine by backlink search. In order to find a new member of a Web community, all the hyperlinks contained in the acquired pages are extracted. Then a page which is pointed by the most frequent hyperlinks is regarded as a new member of the community. We have built a system which discovers complete bipartite graphs based on the method. Only from a few URLs of initial community members, the system succeeds in discovering several genres of Web communities without analyzing the contents of Web pages.

1 Introduction

According to an announcement of Inktomi and NEC Research Institute, the number of Web pages in the world surpasses one billion documents as of January 2000[8]. In order to find useful pages from such vast Web network, many users rely on search engines such as AltaVista or Yahoo. However, they often find difficulty because of ubiquitous synonymy (different words having the same meaning) and polysemy (the same word having multiple meanings). A system which discovers related Web pages is expected to assist users’ information retrieval from the Web.

In general, Web pages contain various forms of information such as sentences, images, and sounds. Understanding all of such contents and classifying the pages appropriately are not easy tasks even for humans. A large number of studies have been made for finding relations of Web pages based on linguistic information. Broeder defines document similarity as the ratio of common subsequence of words[3], Chang employs TFIDF as the criterion for document similarity[6]. Although these methods are widely applicable to ordinary documents, utilizing hyperlinks, which are the information peculiar to Web pages, is expected to help greatly for accurate classification of Web pages.
This paper proposes a method of discovering communities of Web pages based on the co-occurrence of references. A system for discovering Web communities is developed based on the method. The input to the system is a few URLs of Web pages about specific topics such as movies or sports. The output of the system is the communities of Web pages sharing common interests with the input pages.

In this paper, a community is defined as a set of Web pages whose hyperlinks form a complete bipartite graph $K_{ij}$, as mentioned in Kumar’s paper of Web Trawling[13]. In $K_{ij}$ graph, each of $i$ pages contains hyperlinks to all of the $j$ pages. We will call the former $i$ pages as fans and the latter $j$ pages as centers in this paper. The procedure for community discovery is as follows: first, our system acquires fans which have hyperlinks to all the input centers by backlink search on a search engine. From the HTML files of the acquired fans, all the hyperlinks are extracted. Then a page which is pointed by the most frequent hyperlink is added as a new member of centers. By repeating these two steps, a Web community is searched without analyzing the contents of Web pages. Since the method utilizes backlink information, implicit relation between the pages which have no direct hyperlinks to each other can be detected. The system succeeds in discovering several genres of communities only from a few input URLs.

2 Related Work

Hyperlinks often give authority to the contents they point to. Several attempts have been made on link analysis[10] and Web visualization[9][14] since hyperlinks are expected to give important clues for finding relations among Web pages. However, this assumption is not always true because of the following reasons. First, hyperlinks between two pages in the same Web site very often serve a purely navigational function and typically do not represent conferral of authority[12]. Second, related Web sites frequently do not reference one another because they are rivals, they are on opposite sides of thorny social issues, or they simply do not aware of each other’s presence[13]. In order to find relations among Web pages, it is not enough to investigate hyperlinks from the pages; hyperlinks to the pages, which we call backlinks, are often more important. Although it is not possible to search all the backlinks that point to an arbitrary Web page, many of them can be obtained by backlink search on a search engine.


In order to measure the degree of relation between two URLs, this system
performs a search on AltaVista by using the URLs as keywords. The value of the number of pages searched from both URLs divided by the number of pages searched from either URL, which is called Jaccard coefficient, is calculated for evaluating the relation between the two URLs. This is because hyperlinks of related Web pages often co-occur in a Web page. The length of edges connecting two URLs is defined as the reciprocal of Jaccard coefficient of the two URLs so that related Web pages are located close to each other on a graph. This method of measuring the degree of relation based on the number of co-referring pages is similar to the technique of REFERRAL [11] which visualizes researchers’ social network.

For the discovery of relation or rank of Web pages based on the structure of hyperlinks, several researches have been made such as Kleinberg’s Clever project [7], Kumar’s Web Trawling [13], and Page’s PageRank algorithm [16]. HITS algorithm [12], which is one of the central ideas of Clever project, employs authority and hub as the criteria for measuring the usefulness of Web pages. For any particular topic, there tend to be a set of authoritative pages focused on the topic, and a set of hub pages which contain links to useful, relevant pages to the topic. The algorithm associates an authority weight and a hub weight with each Web page. If a page is pointed to by many good hubs, its authority weight should be increased. Authority weight of a page is updated to be the sum of hub weights over all pages that link to the page. In the same manner, hub weight of a page is updated to be the sum of authority weights over all pages that are linked by the page. HITS algorithm iterates the calculation of both weights and outputs a list of the pages with the largest hub weights and the largest authority weights. However, HITS algorithm needs to assemble a different root set for each target topic, and then to prioritize pages in the context of that particular topic.

Kumar’s Web Trawling is the method for discovering communities from the graph structure of Web pages. For example, Web pages of aircraft enthusiasts have hyperlinks to all the homepages of major commercial aircraft manufacturers
such as Boeing and Airbus. These pages and hyperlinks compose a complete bipartite graph \( K_{i,j} \) since each of \( i \) pages contains hyperlinks directed to all of the \( j \) pages. Kumar regards such a graph in the Web network as a cyber-community sharing a common interest. By the search of graph from the snapshot data of Web network, more than a hundred thousand communities are discovered.

Page’s PageRank algorithm calculates the rank of each Web page by propagating the probability distribution that users visit. For each page, its probability is calculated as the sum of the probabilities of the pages that link to the page. The propagation of probabilities is iterated until they converge. The algorithm efficiently computes ranks for 518 million hyperlinks, and it is employed in Google search engine (http://www.google.com/).

These three approaches make use of hyperlinks as clues for detecting relation or rank of Web pages. It is true that these three approaches are effective, but they require a large-scale database of HTML files. Since the number of Web pages in the world is increasing rapidly, building such a database that covers most of the Web network and renewing it are not simple tasks.

Our discovery system described in this paper acquires data from other Web servers during the processes of discovery in order to use new abundant data. In addition to that, the system acquires backlink information from a search engine. The output of the system is a list of URLs which share common interests with the input URLs. It often happens that a user is already familiar with some Web pages of specific topic and needs to find more pages about the topic. If a discovery system outputs a set of related pages which contains the user’s familiar pages, the result is easily accepted by the user. Discovery of a set of pages which are related to given pages is important for achieving Web recommendations that are convincing for a user.

3 A Method of Discovering Web Communities

The goal of our method is to discover a Web community sharing a common interest. Initial members of a community is a few URLs that a user has provided. The overall discovery procedure consists of the following three steps. Each of these steps is explained in the following subsections.

1. Search of fans using a search engine
2. Addition of a new URL to centers
3. Sort of centers in the order of frequency

As mentioned before, a complete bipartite graph \( K_{i,j} \), which is the target graph of our discovery method, is composed of a set of \( i \) pages and a set of \( j \) pages; each of the \( i \) pages has hyperlinks pointing to all of the \( j \) pages. In the following explanation, fans refer to the set of \( i \) pages and centers refer to the set of \( j \) pages.
3.1 Search of fans using a search engine

In our method, input URLs are accepted as initial centers, and fans which co-refer all of the centers are searched. As shown in Figure 2, fans are searched from the centers by backlink search on a search engine. In general, popular centers have too many backlinks. In such cases, a fixed number of high-ranking URLs are selected as fans. Most of the search engines rank pages according to the relevance to input keywords. However, there is not much public information about the specific ranking algorithms used by current search engines[1]. In our method, high-ranking URLs are selected just as a matter of convenience. Since WWW is changing rapidly every day, acquisition of new data is indispensable for the discovery of current (not outdated) communities. By the backlink search on a search engine, relatively new data can be acquired through the internet.

![Diagram of fans and centers](image)

Fig. 2. Search of fans using a search engine

3.2 Addition of a new URL to centers

The next step is to add a new URL to centers based on the hyperlinks of acquired fans. The fans’ HTML files are acquired through the internet, and all the hyperlinks contained in the files are extracted. The hyperlinks are sorted in the order of frequency. Since hyperlinks to related Web pages often co-occur, the top-ranking hyperlink is expected to point to a page whose contents are closely
related to the centers. As shown in Figure 3, the URL of the page is added as a new member of centers.

The above two steps are repeatedly applied in order to acquire many centers. In general, the number of fans decreases according as the number of centers increases since there are more Web pages that contain fewer hyperlinks. The above two steps are repeatedly applied until there are few fans which refer all the members of centers.

![Diagram](image)

**Fig. 3.** Addition of a new URL to centers

3.3 Sort of centers in the order of frequency

Based on the above two steps, a community of related Web pages is generated. However, since our method is based solely on the co-occurrence of hyperlinks, genres of newly-added centers might be different from those of input URLs. For example, a community generated from the URLs of personal computers may contain the URLs of video games. It is not easy to detect such change of genres and to find true boundary of a community. If such boundary cannot be found, it is desirable for a user to rank the URLs of communities in the order of relevance to input URLs. In order to achieve such ranking, communities are generated many times: our method generates communities for every pair of input URLs. For example, if five URLs are provided, ten \( (=C_2) \) communities are generated from the pairs of 1st & 2nd URLs, 1st & 3rd URLs, ..., and 4th & 5th URLs.
Then the centers of all the generated communities are sorted in the order of frequency. The sorted result is expected to reflect the rank of relevance to input URLs because highly-ranked URLs co-occur many times with the input URLs.

4 Experiments

Based on the method described above, a system for discovering Web communities is developed using Java. It is desirable that input URLs are popular ones that many others refer to. As the input to the system, URLs of 100hot.com site (http://www.100hot.com) are used in our experiments. 100hot.com is a collection of a hundred famous Web pages for 74 genres, such as art, movies, travel, and so on. The site is administered by Go2Net, and its ranking is based on the Web surfing patterns of more than 100,000 surfers worldwide. Some genres in the site are duplicated (such as “Entertainment/Book Sites” and “Shopping/Book Sites”), and some are sponsored by companies concerned (for example, “DVD Best Sellers” is sponsored by Amazon.com). These genres are excluded from our experiments. In order to discover communities for the remaining 33 genres, top five URLs of each genre are provided to the system as inputs. Our system generates a community for every pair of the input URLs and outputs the centers of all the communities sorted in the order of frequency.

In order to evaluate the quality of the system’s output URLs, the ranking of 100hot.com are regarded as the collection of “correct answers”. In another words, if a URL is listed in the 100hot.com ranking of corresponding genre, it is regarded as a “correct answer”, otherwise it is regarded as an “incorrect answer”. Since there are many relevant URLs which are not listed in 100hot.com site, this evaluation criterion is rather too severe for the system. However, we dare to employ this criterion since it clarifies the power of our system. As a search engine for backlink search, AltaVista (http://www.altavista.com) is used. The results of the experiments are shown in Table 1. The first column of the table shows genres. The second and third column of the table (total, correct) show the total number of acquired centers for corresponding genre, and the number of “correct answers” among them respectively. From fourth to seventh column (1Q, 2Q, 3Q, 4Q) show the number of “correct answers” in each quarter of ordered list of output URLs. For example, “1Q” shows the number of “correct answers” which are located in the first quarter of the list of output URLs.

Table 1 shows that the system performs very well for many genres. The system discovers these related pages only from five input URLs. As a detailed example, the result of genre “Kids” is shown below. The following five URLs are given to the system as inputs:

- www.pbs.org
- www.headbone.com
- www.bolt.com
- www.yahooligans.com
- www.discovery.com
Table 1. Results of the experiments

<table>
<thead>
<tr>
<th>genre</th>
<th>total</th>
<th>correct</th>
<th>1Q</th>
<th>2Q</th>
<th>3Q</th>
<th>4Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>Art</td>
<td>117</td>
<td>10</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Books</td>
<td>172</td>
<td>6</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Events</td>
<td>184</td>
<td>28</td>
<td>6</td>
<td>11</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>Music</td>
<td>130</td>
<td>42</td>
<td>21</td>
<td>10</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>Jobs</td>
<td>73</td>
<td>15</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Loans</td>
<td>142</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>College</td>
<td>196</td>
<td>47</td>
<td>20</td>
<td>13</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Kids</td>
<td>172</td>
<td>30</td>
<td>3</td>
<td>9</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Gambling</td>
<td>53</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Movies</td>
<td>89</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Games</td>
<td>137</td>
<td>17</td>
<td>8</td>
<td>10</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Family</td>
<td>93</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Food</td>
<td>94</td>
<td>7</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Gardening</td>
<td>148</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Pets</td>
<td>146</td>
<td>9</td>
<td>1</td>
<td>0</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Cars</td>
<td>92</td>
<td>13</td>
<td>16</td>
<td>6</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Chat</td>
<td>96</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Dating</td>
<td>62</td>
<td>10</td>
<td>6</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Spirits</td>
<td>171</td>
<td>4</td>
<td>13</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Travel</td>
<td>124</td>
<td>12</td>
<td>10</td>
<td>16</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Magazines</td>
<td>74</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Newspapers</td>
<td>167</td>
<td>23</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Auction</td>
<td>158</td>
<td>9</td>
<td>6</td>
<td>3</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Flowers</td>
<td>141</td>
<td>8</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Shopping</td>
<td>162</td>
<td>6</td>
<td>6</td>
<td>9</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Health</td>
<td>130</td>
<td>10</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Sports</td>
<td>95</td>
<td>14</td>
<td>7</td>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Developer</td>
<td>123</td>
<td>9</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Hardware</td>
<td>164</td>
<td>9</td>
<td>9</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Mac OS</td>
<td>143</td>
<td>2</td>
<td>7</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unix</td>
<td>95</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Windows</td>
<td>130</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

average: 124.8 19.8 4.4 8.3 7.9

The top 10 of output URLs are shown below:

1. www.cyberkids.com
2. www.ctw.org
3. www.exploratorium.edu
4. www.si.edu
5. www.bonus.com
6. www.kids-space.org
7. www.discovery.com
8. www.yourulesschool.com
9. www.planetzoom.com
10. www.kidscom.com

All of these URLs except www.planetzoom.com (9th) are listed in Kids genre of 100hot.com. If you watch the contents of each URL, you will agree that www.planetzoom.com is also a site for Kids, although it is not listed in 100hot.com. These results show that the system has abilities of discovering many URLs which are related to input URLs.

5 Discussion

5.1 The quality of discovered communities

As shown in Table 1, the number of “correct answers” varies considerably with the genres of input URLs. This means that there are great difference in the quality of discovered communities. For example, communities for “Finance”, “College”, “Kids”, “Cars”, and “Newspapers” are of better quality than those of “Art”, “Family”, and “Chat”. There are various factors for such difference in quality: the number of Web pages belonging to a genre, the number of hyperlinks contained in the pages of a genre, and the contents of a search engine used for backlink search.

The method of discovery described in this paper is effective for the Web communities whose pages are densely connected by many hyperlinks. If the number of centers is rather limited and many fans refer to most of them, our system performs well. The difference of experimental results with genres are caused by users’ browsing pattern and link pattern for each genre, and it does not indicates the limitation of our method.

5.2 Sort of centers in the order of frequency

As mentioned in section 3.3, our system generates communities many times from every pair of input URLs, and sorts acquired centers in the order of frequency. Since it often happens that co-occurring hyperlinks point to pages of completely different genres, acquired URLs should be sorted in order to minimize bad influences of accidental co-occurrences of hyperlinks. Table 1 shows that more “correct answers” are found in higher rank of the output list such as 1Q and 2Q. This means that the URLs which are closely related to the genre of input URLs are located in higher rank of the output list. Although our method is very simple, it succeeds in ranking URLs in the order of relevance to a certain extent.

The list of URLs that our system discovers often contains some portal sites, such as news sites and search engines. In order to avoid such sites to be mixed in discovered communities, anchor descriptions on hyperlinks are expected to be useful. In addition to the co-occurrence of references proposed in this paper, additional information about the contents of referring sites will contribute to the discovery of purer communities.
5.3 Data acquisition from WWW servers through the internet

The system described in this paper acquires needed data from WWW servers and AltaVista in the process of discovery. Although the system accesses to several Web servers, its accesses do not concentrate on one server. The number of accesses to each server is at most the number of centers of a Web community. With the accesses through the internet, new abundant data are available for our discovery system.

Most of the time for community discovery is spent on the acquisition of HTML files through the internet. In our system, only the data acquired within a certain time limit are used for discovery. This time limit is inevitable in order to achieve discovery within practical time even when some WWW servers are down or network condition is really bad.

5.4 Comparison with related works

In order to evaluate the quality of system’s output, previous researches such as HITS or Web Trawling perform experiments using human subjects, or just show the results in their papers. In this paper, the URLs listed in 100hot.com site are regarded as “correct answers”, and they are used for evaluation. Although this criterion is rather severe for the evaluation of related URLs, our system succeeds in discovering 19.8 “correct” URLs on average only from five input URLs. This result is hard to compare with other related researches, but it is surprising that our simple method succeeds in the discovery of so many related URLs.

Our system is quite different from other Web discovery system in that needed data is acquired from WWW servers in the process of discovery. Systems of HITS and Web Trawling require considerable amount of Web pages to be collected and provided in advance. Their performance depend heavily on the amount and the quality of the input data. On the experiments of Kumar’s Web Trawling, used snapshot data are over a year and a half old. Therefore, some of the discovered communities have already disappeared in the current Web network. Since many Web pages are generated and disappeared every day, large-scale Web snapshot data are hard to be obtained and maintained. Our system acquires data in the process of discovery and discovers communities which actually exist in the current Web network only from a few input URLs.

6 Conclusion

This paper describes a method of discovering communities of related Web pages based on the co-occurrence of references. Our community discovery system searches a complete bipartite graph from the data acquired through the internet. The results show that our system has abilities of discovering several genres of related Web pages only from a few input URLs. There are a number of areas for further work:
**Detecting dynamic changes of communities**

Backlink information acquired from a search engine may change as time elapses. By comparing the communities which are generated at regular time intervals will clarify dynamic changes of communities.

**Giving weights to hyperlinks based on their location**

Hyperlinks to related Web pages are often placed close to each other on a Web page. By giving weight to such hyperlinks, more accurate relation of Web pages will be discovered.

**References**

Clustering and Visualization of Large Protein Sequence Databases by Means of an Extension of the Self-Organizing Map

Panu Somervuo and Teuvo Kohonen
Helsinki University of Technology
Neural Networks Research Centre
P.O.Box 5400, FIN-02015 HUT, Finland
panu.somervuo@hut.fi, teuvo.kohonen@hut.fi

Abstract. New, more effective software tools are needed for the analysis and organization of the continually growing biological databases. An extension of the Self-Organizing Map (SOM) is used in this work for the clustering of all the 77,977 protein sequences of the SWISS-PROT database, release 37. In this method, unlike in some previous ones, the data sequences are not converted into histogram vectors in order to perform the clustering. Instead, a collection of true representative model sequences that approximate the contents of the database in a compact way is found automatically, based on the concept of the generalized median of symbol strings, after the user has defined any proper similarity measure for the sequences such as Smith-Waterman, BLAST, or FASTA. The FASTA method is used in this work. The benefits of the SOM and also those of its extension are fast computation, approximate representation of the large database by means of a much smaller, fixed number of model sequences, and an easy interpretation of the clustering by means of visualization. The complete sequence database is mapped onto a two-dimensional graphic SOM display, and clusters of similar sequences are then found and made visible by indicating the degree of similarity of the adjacent model sequences by shades of gray.

1 Introduction

The amount of DNA sequences, protein sequences, and molecule structures studied and reported, e.g., in the Internet is already overwhelming. One should develop better tools for the analysis of the existing databases. Thereby, however, it will also become possible to make new discoveries, without the need to carry out the real biological and chemical experiments.

Among the new challenges one may mention finding the hidden relations between the data items, revealing structures from large databases, and representing the results to the human in a comprehensible way. The classification and clustering of the sequences may reveal new unknown connections between them. The visualization of large data sets in a compact way may give insights into the data and lead to the development of new ideas and theories.

© Springer-Verlag Berlin Heidelberg 2000
Although the data mining applications in general require very specialized and tailored solutions, it is interesting to note that some general principles and methods can already define a framework for these tasks. One such method is the Self-Organizing Map (SOM) [11, 13, 14, 16]. It is a clustering and visualization tool, which has been applied to a diversity of problems. This paper points out the potential of the SOM in the clustering and organization of large sequence databases.

The SOM has already been applied to the clustering of protein sequences. In [6], the sequences were converted into 400-dimensional dipeptide histogram vectors. In [7], similar amino acids were grouped together before computing the histogram vectors. In [9], the sequences were converted into vectors by fractal encoding. Before that the sequences were aligned. In [2], each position of the sequence was represented as a 20-dimensional vector; each vector component corresponded to one amino acid. The whole sequence was then converted into an $L$-by-$20$-dimensional vector, where $L$ is the length of the global alignment of all sequences. As a conclusion, in all these works the data has been encoded by vectors before feeding to the SOM.

A new method suggested by Kohonen [15], however, allows the organization of non-vectorial data items, too. The clustering and organization of the sequence database can therefore be based on any user-defined algorithm, e.g. Smith-Waterman [20], BLAST [1], or FASTA [19]. In the present work, the FASTA method was used for computing the sequence similarities. The SOM was then applied to clustering all the 77,977 protein sequences of the SWISS-PROT database, release 37 [3].

2 The Self-Organizing Map for both vectorial and nonvectorial data

In its original form the Self-Organizing Map is a nonlinear projection method that maps a high-dimensional metric vector space, or actually only the manifold in which the vectorial samples are located, onto a two-dimensional regular grid in an orderly fashion [11, 14]. The SOM differs from the traditional projection methods such as multidimensional scaling, MDS [17] in that unlike in the latter, each original sample is not represented separately, but a much smaller set of model vectors, each of the latter associated with one of the grid nodes, is made to approximate the set of original samples. The SOM thus carries out a kind of vector quantization, VQ [8], in which, however, the model vectors (called codebook vectors in VQ) may be imagined to constitute the nodes of a flexible, smooth network that is fitted to the manifold of the samples.

The SOM principle is not restricted to metric vector spaces, however. It has been pointed out by one of the authors [15] that any set of items, for which a similarity or distance measure between its elements can be defined, can be mapped on the SOM grid in an orderly fashion. This is made possible by the following principle, which combines the concept of the generalized median of a set [12] with the batch computation principle of the SOM [14].
Fig. 1. Illustration of the SOM algorithm for nonvectorial data. Each of the input items $x(1), x(2), \ldots$ is copied into the sublist under that model that has the smallest distance from the respective input item. After that, the generalized median $m_i$ in each neighborhood set $N_i$ is determined, and the old value, say $m_i$, is replaced by $m_i$. This cycle is repeated from the beginning as many times as the models are not changed any longer.

Let us concentrate on the special SOM that is able to map nonvectorial items. Consider Fig. 1 in which a regular grid is shown, with some general model $m_1, \ldots, m_p$ associated with each grid node. Assume that a sublist that contains a subset of input items $x(i)$ can be associated with each model. Each of the input items $x(1), x(2), \ldots$ is compared with all the models and listed under that one that has the smallest distance from the respective input item. The $x(1), x(2), \ldots$ will thus be distributed under the closest models.

Define for each model, say $m_i$, a neighborhood set $N_i$ (the set of models located within a certain radius from the node $i$ in the grid). Consider the union of all the sublists within $N_i$ (shown by the set line in Fig. 1) and try to find the “middlemost” input sample in $N_i$. This sample is called the generalized median of $N_i$, and it is defined to be identical with the input sample that has the smallest sum of distances from all the other samples of $N_i$.

In forming the sum of distances, the contents of the sublists within $N_i$ can be weighted so that the weight is a function of the distance of the nodes of the grid from, say, node $i$. This corresponds to the neighborhood function used with the traditional SOM [14].

Comment 1. If the input samples had been real scalars and the distance measure were the absolute value of their difference, it is easy to show that the “generalized median” coincides with the arithmetic median.

Comment 2. If the input samples were real vectors, and the distance measure were Euclidean, and if the item with the smallest sum of the squares of
distances from the other items were sought, the “generalized median” would coincide with the arithmetic mean of the union of the lists. In this case the “median” is not restricted to the input samples, but belongs to the same domain.

For each \( N_i \) in Fig. 1, \( i = c, d, \ldots, p \) the generalized median is now determined, and the old models \( m_c \ldots m_p \) are replaced by the respective generalized medians.

After this replacement, the original models have now been changed, and if the same input samples are compared with them, they are now redistributed in a different way in the lists. Eventually, however, in a finite number of iterations of this type the process will converge, after which the models approximate the input samples in an orderly fashion.

It is not yet mathematically proven that the above process converges, at least into a unique equilibrium. In practice, convergence means that the lists will not be changed any longer in further iterations. Furthermore, there may exist alternative states into which the map may converge. A proof of a similar “batch map” process with vectorial items has been presented [4], but any conclusions for nonvectorial items can only be drawn from the experimental results, for which no problems have so far existed.

**Comment 3.** Like in the traditional SOM for vectorial items [14], the radius of the neighborhood set \( N_i \) in the beginning of the process may be selected as fairly large and put to shrink monotonically in further iterations. The speed of shrinking should be determined experimentally so that the global ordering is achieved.

### 3 Clustering of 77,977 protein sequences

The SWISS-PROT database, release 37 (12/98) [3] consists of 77,977 protein sequences. The sequences contain altogether 28,268,293 amino acid residues. Organization of a database of this size, and representation of the result in a compact form is a challenging task. Our purpose was to use the definition of distances between the protein sequences, as made in the FASTA method [19] for the computation of the SOM as described in the previous section. A 30-by-20 SOM size was chosen.

The convergence of the nonvectorial SOM algorithm is safer and faster, if the initial models are already two-dimensionally ordered, roughly at least, although not yet optimized. In a couple of earlier works [6, 7], protein sequences were ordered according to the similarity of their dipeptide histograms. We found this method useful for the definition of a rough initial order to the SOM. Then, however, extra auxiliary *model vectors* have to be introduced and associated with the nodes. The initial ordering of the vectorial models in this auxiliary SOM proceeded in the traditional way. Each map node was provided with a 400-dimensional model vector, each component of which was initialized with a random value between zero and unity, whereas the vectors were normalized to unit length. Training was made by the 400-dimensional dipeptide histograms.

...
using 30 batch cycles. A Gaussian neighborhood kernel, the standard deviation of which decreased linearly from 30 to 1 during training, was used.

Next the nodes were labeled by those protein sequences that represented the medians in the sublists under the respective nodes (cf. Fig. 1). When this labeling was ready, the vectorial parts of the models could be abandoned, and the ordering could be continued by the method described in Sec. 2.

After this initializing phase, the true protein sequences were used as inputs as described in Sec. 2 and the winner nodes were determined by the FASTA method. The source code for the FASTA computation was extracted from the FASTA program package, version 3.0 [18]. The parameter ktpu was set to 2, the amino acid substitution scores were taken from the BLOSUM50 matrix, and the final optimized score for the sequence similarity was computed by dynamic programming.

The SOM was trained for twenty batch cycles, using the neighborhood radius of one. (Since the SOM was already ordered, there was no need to use a shrinking kernel any longer.) Since the sequence similarities instead of their distances were finally computed, for the “median” we had to take that sequence in the union of the neighboring sublists that had the largest sum of similarity values with respect to all the other sequences in the neighboring lists. The Gaussian neighborhood function was applied for the weighting of the similarities.

It would have presented a very high computing load to the algorithm if all the 77,977 protein sequences had been used as inputs at each batch computation cycle. The computing load could be reduced to less than ten percent, without essentially deteriorating the (statistical) accuracy of the batch computation, by randomly picking up 6,000 sample sequences from the 77,977 ones for each batch cycle. After 20 such sampled training cycles, one final training cycle was carried out using all the available sequences as the inputs.

The resulting SOM is shown in Fig. 2. The map nodes have been labeled according to the identifiers of the final prototypes that resulted in the “median map” method.

For comparison, another labeling was carried out by listing all data sequences under the best-matching nodes and then performing the majority voting for each list according to the PROSITE classes, release 15 [10] of the sequences. This result is shown in Fig. 3. Since the PROSITE database did not give any class for 37,743 sequences of the SWISS-PROT database, the PROSITE label of the node does not necessarily characterize all sequences of the node.

The clusters can be characterized by means of the known protein families. Those classes whose members are strongly similar are mapped to small areas on the map, while other classes may be spread more widely. Actins and rubisco-large are examples of the classes which form sharp areas on the map. Globin is a large family which is composed of subfamilies. The globin sequences are mostly mapped on the top-left corner of the SOM. Hemoglobin beta chains are represented on the corner, hemoglobin alpha chains are in the cluster below catalases, and myoglobin is located below hemoglobin alpha chains. One sharp cluster on the top of the map consists of efactor-gtp sequences. Between globins
Fig. 2. A 30-by-20-unit hexagonal SOM grid. The SOM was constructed using all the 77,977 protein sequences of the SWISS-PROT release 37. Each node contains a prototype sequence and a list of data sequences. The labels on the map nodes are the SWISS-PROT identifiers [3] of the prototype sequences. The upper label in each map node is the mnemonic of the protein name and the lower label is the mnemonic of the species name. The similarities of the neighboring prototype sequences on the map are indicated by shades of gray. The light shades indicate a high degree of similarity, and the dark shades a low degree of similarity, respectively. Light areas on the map reveal large clusters of similar sequences.
and e-factor-gtp there is a cluster of the hsp70 family. Tubulins are mapped to two closely located areas, one of which is characterized by alpha subunits and another by beta subunits, respectively.

Since there are altogether 1,352 classes in the PROSITE database, not all of them can be discussed in detail. But a general idea of the capability of the SOM can be gained by investigating the projections of the most frequent classes. Therefore the PROSITE classes were sorted according to their frequency in the SWISS-PROT database. The 32 most frequent classes were then projected on the SOM by finding the best-matching unit of each sequence belonging to the given class. The resulting class distributions are shown in Fig. 4.

In the visualization of the class distributions, some PROSITE classes were combined. For example, the actins class consists of 249 sequences of the family actins, act-like, 232 sequences of actins_2, and 227 sequences of actins_3. Trypsin_ser and trypsin_his were combined to the single trypsin class. Thiol_protease_lys, thiol_protease_his, and thiol_protease_ser were combined to the single thiol_protease class. Cytochrome_b class in the figure consists of both cytochrome_b_2o and cytochrome_b_heme. The distribution of protein_kinase_atp (1040 sequences) is not shown, because it was identical with the distribution of the protein_kinase_4com (1093 sequences).

Analyzing the cluster contents according to known protein families can give information about the specificity of the prototype sequences, like in the organization of the database performed on the basis of the sequence similarities. The
classification of the sequences according to the PROSITE classes, however, may also include structural information about the protein molecules. At any rate, many PROSITE classes were mapped to small and sharp areas on the SOM display.

Once the SOM has been trained, it is very fast to compute the projection of any new sequence. This requires only as many sequence comparisons as there are prototype sequences on the map. In the current work, the SOM contained 600 prototype sequences. Thus the work needed for classifying the new sequence into a prototype class is considerably lighter than comparison with all the 77,977 sequences of the whole database.

4 Discussion

This paper is based on the combination of two new possibilities: accessibility to masses of biological data in the Internet, and recent development of a clustering and visualization method that can cope with the masses of raw nonvectorial data in an unsupervised way.

The currently existing search engines for biological databases may give thousands of matches as a result of a short DNA sequence as a query sequence. The SOM can serve as a global visualization display, onto which also the results obtained by other means can be mapped. The sequence similarities can then be investigated based on the projections of the sequences on the map. The results for one query sequence can be all mutually similar or they can form distinct clusters, which can be reached by visual browsing.

The special Self-Organizing Map for symbolic items has been applied in this work for the first time to a major problem, self-organization of the 77,977 protein sequences of the SWISS-PROT database. Contrasted with earlier works, this extension of the SOM allows the use of any similarity measure for sequences. The resulting clustering and ordering of the data reflects the properties of the chosen similarity measure. The present result, where the similarities are computed by the FASTA method, is a two-dimensional map where similar proteins are mapped to the same node or neighboring nodes, and the structures of the clusters are thereby visualized, too. The geometrically organized picture makes it possible to illustrate the relationships of a large amount of sequences at a glance.

Since the SOM provides an ordered display of the representative prototype items of the data set, it may be used, e.g., for designing oligonucleotide or cDNA arrays (see [5] for a collection of reviews on microarray analysis). If the arrays were ordered using the SOM, similar oligonucleotides would be located close to each other in the array thus helping the visual interpretation of the data.

A great advantage of the SOM is that the basic form of the algorithm is very simple and straightforward to implement. It is therefore easy to apply the SOM to various tasks. The SOM can be used as a data mining and visualization tool for any data set, for which a similarity or distance measure between its elements can be defined.
Fig. 4. Projections of the 32 most frequent PROSITE classes of the SWISS-PROT database on the SOM. Each subfigure represents the distribution of one class. The prototype sequences of the map nodes are the same as in Fig. 2. The shades of gray indicate the number of the protein sequences belonging to the given class in each map node. The maximum value (darkest shade of gray) is scaled to unity in each subfigure. The total number of the sequences in each class is shown in the parentheses after the PROSITE name.
References


A Simple Greedy Algorithm for Finding Functional Relations: Efficient Implementation and Average Case Analysis

Tatsuya Akutsu¹, Satoru Miyano¹, and Satoru Kuhara²

¹ Human Genome Center, Institute of Medical Science, University of Tokyo, 4-6-1 Shirokanedai, Minato-ku, Tokyo 108-8639, Japan
{sakutsu.miyano}@ims.u-tokyo.ac.jp
² Graduate School of Genetic Resources Technology, Kyushu University, Hakozaki, Higashi-ku, Fukuoka 812-8581, Japan
kuhara@grt.kyushu-u.ac.jp

Abstract. Inferring functional relations from relational databases is important for discovery of scientific knowledge because many experimental data in science are represented in the form of tables and many rules are represented in the form of functions. A simple greedy algorithm has been known as an approximation algorithm for this problem. In this algorithm, the original problem is reduced to the set cover problem and a well-known greedy algorithm for the set cover is applied. This paper shows an efficient implementation of this algorithm that is specialized for inference of functional relations. If one functional relation for one output variable is required, each iteration step of the greedy algorithm can be executed in linear time. If functional relations for multiple output variables are required, it uses fast matrix multiplication in order to obtain non-trivial time complexity bound. In the former case, the algorithm is very simple and thus practical. This paper also shows that the algorithm can find an exact solution for simple functions if input data for each function are generated uniformly at random and the size of the domain is bounded by a constant. Results of preliminary computational experiments on the algorithm are described too.

1 Introduction

Many scientific rules are represented in the form of functions. For example, an output value $y_j$ may be a function of several input variables $x_{i_1}, \ldots, x_{i_u}$ (i.e., $y_j = f_j(x_{i_1}, \ldots, x_{i_u})$). For another example, a simple differential equation of the form $\frac{dy}{dx} = f_j(x_{i_1}, \ldots, x_{i_u})$ can also be considered as a function if we can know the values of $\frac{dy}{dx}$ (e.g., using $\frac{dy}{dx}$ in place of $\frac{dy}{dx}$). Moreover, many experimental data in sciences are represented in the form of tables. Therefore, inferring functional relations from tables is important for scientific discovery. Since a relational database consists of tables, this problem is almost equivalent to inference of functional relations from relational databases.
Inference of functional relations (or almost equivalently, inference of functional dependencies) from relational databases is rather a classical problem in the field of KDD (knowledge discovery in databases) [2,9-11]. Since $y_j = f_j(x_{i_1}, \ldots, x_{i_d})$ holds for all $x_{i_1}, \ldots, x_{i_d}$ if $y_j = f_j(x_1, \ldots, x_i)$ holds, it is usually required to find the minimum set or the minimal sets of input variables. Mannila and Räihä proposed a heuristic algorithm for finding functional dependencies [10]. Inference of functional dependencies with small noises was also studied and PAC-type analysis was made [2,9]. Unfortunately, Mannila and Räihä proved that finding a functional dependency with the minimum number of input attributes (i.e., $d$ is minimum) is NP-hard [11]. Therefore, development of heuristic algorithms and/or approximation algorithms is important. As mentioned before, Mannila and Räihä proposed a heuristic algorithm [10]. Akutsu and Bao proposed a simple greedy algorithm in which the original problem was reduced to the set cover problem [3]. Although an upper bound on the approximation ratio (on $d$) is given, the time complexity is not low if it is implemented as it is. Even for finding a functional relation for one output variable (i.e., one $y_j$), it takes $O(m^2ng)$ time ($O(m^2n)$ time using an efficient implementation for the set cover problem [7]), where $n$ denotes the number of attributes, $m$ denotes the number of tuples and $g$ denotes the number of main iterations in the greedy algorithm. This time complexity is too high for applying the greedy algorithm to large databases.

This paper gives a simple implementation of the greedy algorithm, which runs in $O(mng)$ time in the case of finding a functional relation for one output variable. Each iteration can be done in linear time since the size of input data (i.e., input table) is $O(mn)$. This complexity is reasonable because $g$ is usually small (e.g., < 10). This algorithm has some similarity with decision tree construction algorithms, where the similarity and difference are to be discussed in the final section. By the way, in some applications, it is required to infer functional relations for multiple output variables simultaneously. For example, in inference of genetic networks [4], functional relations should be inferred for all genes (i.e., for $n$ genes). In such a case, $O(m^2ng)$ time is still required using the efficient implementation mentioned above. Therefore, we developed an improved algorithm for a special case in which $g$ can be regarded as a constant and the size of the domain is bounded by a constant. This algorithm is based on a fast matrix multiplication algorithm [6] as in [4], and the time complexity is $O(m^{\omega-2}n^2 + mn^{2.376})$, where $\omega$ is the exponent of matrix multiplication (currently, $\omega < 2.376$ [6]). Although this algorithm is not practical, it is faster than the $O(m^2ng)$ time algorithm when $m$ is large.

This paper also gives an average case analysis of the greedy algorithm for simple functions (such as AND of literals, OR of literals), under the condition that input data are generated uniformly at random and the size of the domain is bounded by a constant. In this case, the greedy algorithm finds an exact solution with high probability, where the probability is taken over all possible input data. This gives another theoretical guarantee to the algorithm. Recall that it is already known that the greedy algorithm outputs a solution with a guaranteed
approximation ratio even in the worst case [3]. Therefore, the greedy algorithm works very well for the average case inputs, whereas the greedy algorithm does not work so badly even in the worst case.

In order to ensure the effectiveness of the algorithm, we made preliminary computational experiments. Since we were particularly interested in the application to inference of genetic networks, we made computational experiments on Boolean networks, where the Boolean network is a mathematical model of a genetic network [13]. The results of computational experiments suggest that the greedy algorithm is very useful.

Before describing details, we briefly discuss about the difference between association rules [1] and functional relations, since inference of association rules is well studied. In order to explain the difference, we consider very simple rules on the binary domain \(\{0, 1\}\). The following is a typical example of an association rule: \((x_i = 1) \land (x_j = 0) \land (x_k = 1) \rightarrow y_j = 1\), which means that if \(x_i = 1\), \(x_j = 0\) and \(x_k = 1\) hold then \(y_j\) should be 1, where we are considering a simplified version of an association rule and thus we do not consider support and confidence. In this case, \(y_j\) can take any value if either one of \(x_i \neq 1\), \(x_j \neq 0\) or \(x_k \neq 1\) holds. The following is a typical example of a functional relation: \(y_j = x_i \land \neg x_j \land x_k\), which means that \(y_j = 1\) if and only if \(x_i = 1\), \(x_j = 0\) and \(x_k = 1\) hold. Therefore, \(y_j\) should be 0 if either one of \(x_i \neq 1\), \(x_j \neq 0\) or \(x_k \neq 1\) holds. Although association rules are convenient for representing various kinds of knowledges, functional relations seem to be more appropriate for representing concrete rules (such as differential equations).

2 Preliminaries

In this paper, we assume a fixed and finite domain \(\mathcal{D}\) for all attributes, where extension to cases in which different domains are assigned to different attributes is straight-forward and thus omitted. Extension to the domain of real numbers will be discussed in Section 3.

For simplicity, we consider two sets of attributes: the set of input attributes and the set of output attributes, where these two sets are not necessarily disjoint. Usual functional dependencies can be treated by letting both sets be identical to the original set of attributes. Input attributes and output attributes are also called input variables and output variables, respectively. Let \(x_1, \ldots, x_n\) be input variables. Let \(y_1, \ldots, y_l\) be output variables. Let \(\langle x_{i_1}(k), \ldots, x_{i_d}(k), y_{j_1}(k), \ldots, y_{j_l}(k) \rangle\) be the \(k\)-th tuple in the table, where \(x_{i}(k) \in \mathcal{D}\), \(y_{j}(k) \in \mathcal{D}\) for all \(i, j, k\). Then, we define the problem of inference of functional relations in the following way.

**INPUT:** \(\langle x_{i_1}(k), \ldots, x_{i_d}(k), y_{j_1}(k), \ldots, y_{j_l}(k) \rangle\) \((k = 1, \ldots, m)\), where \(x_{i}(k) \in \mathcal{D}\) and \(y_{j}(k) \in \mathcal{D}\) for all \(i, j, k\).

**OUTPUT:** for each \(y_j\), a set \(X_j = \{x_{i_1}, \ldots, x_{i_d}\}\) with the minimum cardinality (i.e., minimum \(d\)) for which there exists a function \(f_j(x_{i_1}, \ldots, x_{i_d})\) such that \((\forall k)\langle y_{j}(k) = f_j(x_{i_1}(k), \ldots, x_{i_d}(k)) \rangle\) holds.
It should be noted that the problem is defined as a minimization problem since the number of sets of input variables satisfying the condition can be exponential. If multiple sets with the same cardinality satisfy the condition, any set can be output. It should also be noted that we do not require the explicit representation of the function \( f_j \) because the number of possible \( f_j \)'s with \( d \)-input variables on domain \( D \) is \( |D|^d \) and thus exponential space \( \Omega(\log |D|) \) is required in order to represent \( f_j \). Thus, we do not mind representation of \( f_j \). Instead, we only mind sets of input variables.

Since each \( y_j \) can be treated independently, we assume \( l = 1 \) unless otherwise stated.

3 Simple Greedy Algorithm

It is known that inference of functional dependencies is NP-hard [11]. Therefore, a simple greedy algorithm has been proposed [3]. We denote this algorithm by GREEDY. In GREEDY, the original problem is reduced to the set cover problem and a well-known greedy algorithm for the set cover [8] is applied. The following is a pseudo-code for GREEDY.

\[
S \leftarrow \{(k_1, k_2) | k_1 < k_2 \text{ and } y_1(k_1) \neq y_1(k_2)\} \\
X_1 \leftarrow \{\} \\
X \leftarrow \{x_1, \ldots, x_n\} \\
\text{while } S \neq \{\} \text{ do} \\
\quad \text{for all } x_i \in X \text{ do} \\
\quad \quad S_i \leftarrow \{(k_1, k_2) \in S | x_i(k_1) \neq x_i(k_2)\} \\
\quad \quad \text{let } x_i \text{ be the variable with the maximum } |S_i| \\
\quad \quad S \leftarrow S - S_i \\
\quad X \leftarrow X \setminus \{x_i\} \\
\quad X_1 \leftarrow X_1 \cup \{x_i\} \\
\text{output } X_1
\]

Using the result on the approximation ratio of the greedy algorithm for the set cover [8], the following result was obtained.

Theorem 1. [8] Suppose that \( f_1(x_{i_1}(k), \ldots, x_{i_d}(k)) = y_1(k) \) holds for all \( k \). Then GREEDY outputs a set of variables \( \{x_{i_1}, \ldots, x_{i_d}\} \) such that \( g \leq (2 \ln m + 1)l \) and there exists a function \( f'_1 \) satisfying \( f'_1(x_{i_1}(k), \ldots, x_{i_d}(k)) = y_1(k) \) for all \( k \).

Note that \( O(\log m) \) lower bound on the approximation ratio was also proven in [3]. Thus, the approximation ratio of GREEDY is optimal except a constant factor.

GREEDY may be modified for the domain of real numbers by replacing \( y_1(k_1) \neq y_1(k_2) \) and \( x_i(k_1) \neq x_i(k_2) \) with \( |y_1(k_1) - y_1(k_2)| > \delta \) and \( |x_i(k_1) - x_i(k_2)| > \delta' \) respectively, using appropriate \( \delta \) and \( \delta' \).
4 Efficient Implementation

GREEDY takes $O(m^2ng)$ time (for $l = 1$) if it is executed as it is, where $g$ is the number of the iterations of the while loop (i.e., the size of $X_1$). GREEDY takes $O(m^2ng)$ time even if an efficient implementation [7] for the set cover problem is used. For large $n$, $m$, it would take too long time. Therefore, we should consider more efficient implementation. In this section, we will describe an improved implementation of GREEDY (we call it GREEDY1) which works in $O(mng)$ time. Since the size of input data is $O(mn)$, this means that each iteration can be done in linear time.

We assume without loss of generality that $\mathcal{D} = \{0, 1, 2, \ldots, D - 1\}$, where $D$ is a constant (recall that we assumed a finite domain). We partition the set of tuples into blocks $B_1, \ldots, B_H$, where each tuple $(x_1(k), \ldots, y_1(k))$ is denoted by $k$. The method of partition will be described later. For each block $B_k$, for each input variable $x_i$, and for each $(p, q) \in \mathcal{D} \times \mathcal{D}$, we define $c_{k,i}(p, q)$ by $c_{k,i}(p, q) = \left| \{ k \in B_k \mid x_i(k) = p \text{ and } y_1(k) = q \} \right|$. Then, we define $c_{k,i}$ by

$$c_{k,i} = \sum_{q \neq q'} \left( \sum_{p \neq p'} c_{k,i}(p, q) \cdot c_{k,i}(p', q') \right).$$

We say that $(k, k')$ is covered by $x_i$ if both $y_1(k) \neq y_1(k')$ and $x_i(k) \neq x_i(k')$ hold. Then, $c_{k,i}$ denotes the number of pairs $(k, k')$ from $B_k$ that are covered by $x_i$ (i.e., $S_i = \sum_{k} c_{k,i}$).

Then, the following pseudo-code describes the improved algorithm:

```
B_1 \leftarrow \{1, \ldots, m\}
H \leftarrow 1
X_1 \leftarrow \{\}
X \leftarrow \{x_1, \ldots, x_n\}
while \left| \{y_1(k) \mid k \in B_k\} \right| > 1 \text{ for some } h \leq H \text{ do }
  for all \( x_i \in X \) do
    c_i \leftarrow \sum_{k=1}^H c_{k,i}
    Let \( x_i \) be the variable with the maximum \( c_i \)
  for all \( B_h \) and \( p \) \((p = 0, \ldots, D - 1)\) do
    \( B_{h,p} \leftarrow \{ k \in B_h \mid x_i(k) = p \} \)
    Remove \( B_{h,p} \) such that \( B_{h,p} = \{\} \)
    Replace \( B_1, \ldots, B_H \) by the remaining \( B_{h,p} \)'s
  \( X \leftarrow X - \{x_i\} \)
  \( X_1 \leftarrow X_1 \cup \{x_i\} \)
output \( X_1 \)
```

Note that condition \( \left| \{y_1(k) \mid k \in B_k\} \right| > 1 \) means that there exists a pair of tuples from $B_k$ which should be covered by using additional variables not in $X_1$. From the definitions of $B_k$ and $c_{k,i}$, it is easy to see that GREEDY1 is equivalent to GREEDY.
A Simple Greedy Algorithm for Finding Functional Relations

91

Fig. 1. Example of execution of GREEDY1. In the first iteration, $x_3$ is selected because $c_1 = 3$, $c_2 = 3$ and $c_3 = 4$. In the second iteration, either $x_1$ or $x_2$ is selected because $c_1 = 1$ and $c_2 = 1$.

Next, we consider the time complexity. Since $B_k$'s are the partition of \{1, 2, \ldots, m\}, $B_{h,k}$'s can be computed in $O(m)$ time (per while loop). So, the most time consuming part is computation of $c_{h,i}(p,q)$'s. In order to compute $c_{h,i}(p,q)$'s, we use the following procedure:

\[
\begin{align*}
\text{for all } p,q \in D & \times D \text{ do} \\
& c_{h,i}(p,q) \leftarrow 0 \\
\text{for } h = 1 \text{ to } H \text{ do} \\
& \text{ for all } k \in B_h \text{ do} \\
& \quad p \leftarrow x_{i}(k) \\
& \quad q \leftarrow y_{i}(k) \\
& \quad c_{k,i}(p,q) \leftarrow c_{k,i}(p,q) + 1
\end{align*}
\]

Since $D$ is assumed to be a constant and $\sum_h |B_h| \leq m$, this procedure works in $O(m)$ time. Therefore, the total time complexity is $O(mng)$. We can use the same space to store $c_{k,i}(p,q)$'s for different $x_i$'s. Thus, the space complexity is linear (i.e., $O(mn)$).

**Theorem 2.** Suppose that $f_1(x_{i_1}(k), \ldots, x_{i_g}(k)) = y_{i}(k)$ holds for all $k$. Then GREEDY1 outputs a set of variables \{$x_{i_1}, \ldots, x_{i_g}$\} in $O(mng)$ time such that $g \leq (2\ln m + 1)d$ and there exists a function $f'_1$ satisfying $f'_1(x_{i_1}(k), \ldots, x_{i_g}(k)) = y_{i}(k)$ for all $k$.

Even if $D$ is not a constant, the total time complexity is $O(mng\log n)$ by using an appropriate data structure for maintaining $c_{k,i}(p,q)$.

Until now, we assumed $l = 1$. But, in some cases, we should find functional relations for many $y_i$'s. For example, in inference of genetic networks [4], we should find functional relations for all genes (i.e., $l = n$), where the number of genes is at least several thousands even for microorganisms such as Yeast. It would take $O(mng^2y)$ time if we apply GREEDY1 to all $y_i$ independently. Therefore, it is worthy to consider an efficient implementation for the case of $l = n$. 
For small $g$ and small $D$ (for example, $g, D \leq 3$), we can develop an improved algorithm using a fast matrix multiplication as in [4]. Here, we show a brief sketch of the algorithm. Recall that the most time consuming part of GREEDY1 is computation of $c_{h,i}(p,q)$’s. Recall that $c_{h,i}(p,q)$ is defined as follows.

$$c_{h,i}(p,q) = |\{ k \in B_h | x_i(k) = p \text{ and } y_j(k) = q \}|.$$ 

Since $c_{h,i}(p,q)$ must be computed for each $y_j$, we use $c_{h,i,j}(p,q)$ to denote the value of $c_{h,i}(p,q)$ for $y_j$.

Here, we define an $m$-dimensional vector $x^p_i$ by

$$(x^p_i)_k = \begin{cases} 1, & \text{if } x_i(k) = p, \\ 0, & \text{otherwise}, \end{cases}$$

where $(x^p_i)_k$ denotes the $k$-th element of a vector $x^p_i$. We define $y^p_j$ in a similar way. Then, $c_{h,i,j}(p,q)$ is equal to the inner product $x^p_i \cdot y^p_j$. Therefore, for each fixed $h, p, q$, we can compute $c_{h,i,j}(p,q)$'s by using a matrix multiplication as in [4]. Of course, this computation should be done for all combinations of $h, p, q$.

But, if $g$ and $D$ are constant, the number of combinations is a constant. In such a case, the total time complexity is $O(m^{2+\omega}n^2 + mn^{2+\omega-1})$, where $\omega$ is the exponent of matrix multiplication (currently, $\omega < 2.376$ [6]). It is smaller than $O(mn^2)$ if $m$ is large. We denote this algorithm by GREEDY2.

**Theorem 3.** Suppose that $f_j(x_{i_1}(k), \ldots, x_{i_g}(k)) = y_j(k)$ holds for all $k$ and for all $y_j$. Suppose also that $g$ (the number of iterations) and $D$ are bounded by a constant. Then GREEDY2 outputs a set of variables $\{x_{i_1}, \ldots, x_{i_g}\}$ for all $y_j$ in $O(m^{2+\omega}n^2 + mn^{2+\omega-1})$ time such that there exists a function $f'_j$ satisfying $f'_j(x_{i_1}(k), \ldots, x_{i_g}(k)) = y_j(k)$ for all $k$.

5 Average Case Analysis for Simple Functions

Even if we only consider the Boolean domain $\mathcal{D} = \{0, 1\}$ and we restrict functions to be either AND of literals or OR of literals, the inference problem remains NP-hard [4]. But, on the average, GREEDY (or equivalently GREEDY1) finds correct sets of input variables with high probability in that case, where the average is taken over all possible inputs and we assume that input data are generated uniformly at random. We also assume that output value $y_1$ depends only on input variables $x_{i_1}, \ldots, x_{i_g}$. In this section, we show a sketch of the proof for this case and then we discuss the extension to other functions and other domains.

5.1 Analysis of AND/OR Functions

For simplicity, we only consider the following function:

$$y_1(k) = x_{i_1}(k) \land x_{i_2}(k) \land \cdots \land x_{i_g}(k).$$
Fig. 2. Illustration for average case analysis of a Boolean function $y_1(k) = x_1(k) \land x_2(k) \land x_3(k)$. In the first iteration, $\frac{2^4 - 1}{2^4 - 2} \cdot |S| = 2^2$ pairs are covered. In the second iteration, $\frac{2^4 - 1}{2^4 - 2} \cdot |S| = 2^3$ pairs are covered.

From the symmetry, the other AND functions and OR functions can be treated in an analogous way.

Since we assume that input data are generated uniformly at random, $\text{Prob}(x_i(k) = 1) = \text{Prob}(x_i(k) = 0) = 0.5$ holds for all $x_i$ and for all $k$. Therefore, for each assignment $A$ to $x_1, \ldots, x_i$, the number $|\{k : (x_i(k), \ldots, x_i(k)) = A\}|$ is expected to be very close to $\frac{1}{2^i}m$ in most cases if $m$ is sufficiently large, where we denote an assignment by a vector of 0,1's (for example, $(0, 1, 1)$ denotes $x_1 = 0, x_2 = 1, x_3 = 1$). Among $2^d$ possible assignments to $x_1, \ldots, x_i$, only $(1, 1, \ldots, 1)$ can make $y_1(k) = 1$. Therefore, at the first line of GREEDY, $|S| \approx \frac{1}{2^i}m \times \frac{2^d - 1}{2^d - 2}m$ is expected. If $x_i \notin \{x_i, \ldots, x_i\}$, $\text{Prob}(x_i(k) \neq x_i(k')) = \frac{1}{2}$ holds for each pair $(k, k') \in S$ because $x_i$'s are assumed to be independent. Therefore, for $x_i, x_i, \ldots, x_i$, $|S_i| \approx \frac{1}{2}|S|$ is expected. On the other hand, if $x_i \notin \{x_i, \ldots, x_i\}$, $\text{Prob}(x_i(k) \neq x_i(k')) = \frac{2^d - 1}{2^d - 2}m$ holds and thus $|S_i| \approx \frac{2^d - 1}{2^d - 2}|S|$ is expected (see Fig. 2). Since $\frac{2^d - 1}{2^d - 2}|S| > \frac{1}{2}|S|$, it is expected that one of $x_1, \ldots, x_i$ is selected in the first iteration of GREEDY.

Assume that $x_i$ is selected in the first iteration. Then, at the beginning of the second iteration, $|S| \approx \frac{1}{2^i}m \times \frac{2^d - 1}{2^d - 2}m$ is expected. If $x_i \notin \{x_i, \ldots, x_i\}$, $|S_i| \approx \frac{1}{2}|S|$ is expected too. If $x_i \notin \{x_i, \ldots, x_i\}$, $|S_i| \approx \frac{1}{2^i}m \times \frac{2^d - 1}{2^d - 2}m$ is expected. Therefore, it is expected that one of $x_1, \ldots, x_i$ is selected in the second iteration of GREEDY.

In this way, it is expected that $x_1, \ldots, x_i$ are selected and output by GREEDY. Making detailed probabilistic analysis, we can obtain the following theorem.

**Theorem 4.** Suppose that $\mathcal{D} = \{0, 1\}$ and functions are restricted to be either AND of $d$ literals or OR of $d$ literals, where $d$ is a constant. Suppose that input
data are generated uniformly at random (i.e., \( \text{Prob}(x_i(k) = 1) = 0.5 \) for all \( x_i \) and all \( k \)). Then, for sufficiently large \( m (m = \Omega(n \log n)) \), GREEDY outputs the correct set of input variables \( \{x_{i_1}, \ldots, x_{i_d}\} \) for \( y_1 \) with high probability (with probability \( > 1 - \frac{1}{m^2} \) for any fixed constant \( c \)).

Note that if we choose \( d \) variables with \( d \)-highest \( |S_i| \)'s in the first iteration of GREEDY, we can output the correct set of variables \( \{x_{i_1}, \ldots, x_{i_d}\} \) with high probability. However, the chance that correct \( x_{i_d} \) is selected in the second iteration of GREEDY is higher because \( \frac{d_{i_{d-2}}^2}{d_{i_{d-1}}^2} > \frac{d_{i_{d-1}}^2}{d_{i_{d-2}}^2} \). Therefore, GREEDY (equivalently GREEDY1) should be used.

### 5.2 Towards Analysis of More General Functions

Unfortunately, GREEDY cannot output the correct set of variables for XOR functions. For example, consider the case of \( y_1(k) = x_{i_1}(k) \oplus x_{i_2}(k) \). Then, \( |S_{i_1}| \approx \frac{1}{2} |S| \) and \( |S_{i_2}| \approx \frac{1}{4} |S| \) are expected. Thus, GREEDY may fail to find \( x_{i_1} \) or \( x_{i_2} \). However, this case seems to be an exceptional case. From computational experiments on \( d \leq 4 \), we found that properties similar to that in Section 5.1 held for many Boolean functions of \( d \leq 4 \). Although we do not yet succeed, we are trying to clarify the class of functions for which GREEDY outputs correct sets of input variables in the average case.

Extension of the Boolean domain to other fixed domain \( D \) is possible. In this case, AND function may be replaced by a function of the form

\[
\text{if } x_{i_1}(k) = z_1 \land x_{i_2}(k) = z_2 \land \cdots \land x_{i_d}(k) = z_d \text{ then } y_1 = z_{d+1} \text{ else } y_1 = z_{d+2},
\]

where \( z_i \in D \). If \( |D| \) is a constant, an analysis similar to that in Section 5.1 is possible. But, if \( |D| \) is large, it is difficult to discriminate \( x_{i_1}, \ldots, x_{i_d} \) from the other \( x_{i_*} \)'s because the gap between \( |S_i| \)'s will become very small.

Although we assumed noiseless cases, real data contain noises. For example, \( y_1(k) = f_j(x_{i_1}(k), \ldots, x_{i_d}(k)) \) may not hold with some error probability \( \varepsilon \) (i.e., \( \text{Prob}(y_1(k) \neq f_j(x_{i_1}(k), \ldots, x_{i_d}(k))) = \varepsilon \)). However, it is expected that GREEDY still works for this case. Although GREEDY would output variables not in \( \{x_{i_1}, \ldots, x_{i_d}\} \) if all pairs in \( S \) should be covered, we can stop GREEDY after the \( d \)-th iteration. In this case, it is expected for sufficiently small \( \varepsilon \) and sufficiently large \( m \) that GREEDY will output the correct set of variables.

### 6 Preliminary Computational Experiments

We made preliminary computational experiments in order to ensure the effectiveness of GREEDY1. As mentioned in Section 1, we are particularly interested in the application to inference of genetic networks. Therefore, we made computational experiments using a mathematical model of a genetic network.

For modeling genetic networks, various mathematical models have been proposed. The Boolean network is a simplest model among them [13]. Although this is a conceptual model and real genetic networks are much more complex, this
model may be still useful for analysis of real experimental data because the errors of real experimental data are large and thus real values should be classified into several discrete values (e.g., 0, 1 in the Boolean network). Thus, we used Boolean networks in this computational experiment. We can treat inference of Boolean networks by letting $D = \{0, 1\}$ and $l = n$ [4]. However, we made experiments only on cases of $l = 1$ because $y_j$'s can be treated independently. For all computational experiments, we used a PC with 700MHz AMD Athlon CPU (1CPU) on which the Turbo Linux 4.2 operating system was running.

First we made computational experiment on AND functions. We examined all combinations of $n = 1000$, $m = 100, 500, 1000$ and $d = 2, 3, 4, 5$. For each case, we calculated the average CPU time required for inference and the ratio of successful executions of GREEDY1 over 100 trials, where we say that GREEDY1 is successful if it outputs the correct set $\{x_i, \ldots, x_n\}$. We modified GREEDY1 so that it stopped after the $d$-th iteration. For each case, we generated input data and an AND function in the following way. First, we choose $d$ different variables $x_i, \ldots, x_d$ randomly from $x_1, \ldots, x_n$ and then we choose an AND function $f_k(x_i, \ldots, x_k)$ randomly from $2^d$ possible AND functions (i.e., AND of literals). For all $x_i (i = 1, \ldots, n)$ and for all $k (k = 1, \ldots, m)$, we let $x_i(k) = 0$ with probability 0.5 and $x_i(k) = 1$ with probability 0.5. For all $k$, we let $y_i(k) = f_k(x_i(k), \ldots, x_i(k))$. The following table shows the result, where the percentage of the success ratio (%) and the average CPU time (sec.) are given for each case.

<table>
<thead>
<tr>
<th>$m$</th>
<th>$d = 2$</th>
<th>$d = 3$</th>
<th>$d = 4$</th>
<th>$d = 5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>100% 0.006</td>
<td>83% 0.014</td>
<td>2% 0.022</td>
<td>1% 0.030</td>
</tr>
<tr>
<td>500</td>
<td>100% 0.113</td>
<td>100% 0.165</td>
<td>99% 0.218</td>
<td>39% 0.274</td>
</tr>
<tr>
<td>1000</td>
<td>100% 0.280</td>
<td>100% 0.399</td>
<td>100% 0.540</td>
<td>95% 0.659</td>
</tr>
</tbody>
</table>

It is seen from this table that GREEDY1 outputs the correct sets of variables with high probability for AND functions, and the probability increases as $m$ increases. Note also that CPU time increases near linear to $d$. Although the CPU time for $m = 1000$ is longer than twice of that for $m = 500$, it is still near linear to $m$.

Next, we made computational experiment on general Boolean functions of $d$ input variables. In this case, we choose a Boolean function $f_k(x_i, \ldots, x_k)$ randomly from $2^d$ possible Boolean functions. The following table shows the result, where the percentage of the success ratio (%) and the average CPU time (sec.) are given for each case.

<table>
<thead>
<tr>
<th>$m$</th>
<th>$d = 2$</th>
<th>$d = 3$</th>
<th>$d = 4$</th>
<th>$d = 5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>50% 0.007</td>
<td>52% 0.017</td>
<td>32% 0.024</td>
<td>9% 0.035</td>
</tr>
<tr>
<td>500</td>
<td>52% 0.111</td>
<td>77% 0.177</td>
<td>92% 0.238</td>
<td>89% 0.304</td>
</tr>
<tr>
<td>1000</td>
<td>40% 0.275</td>
<td>76% 0.424</td>
<td>97% 0.576</td>
<td>98% 0.747</td>
</tr>
<tr>
<td>2000</td>
<td>50% 0.612</td>
<td>77% 0.945</td>
<td>92% 1.238</td>
<td>100% 1.642</td>
</tr>
</tbody>
</table>

It is seen that for sufficiently large $m$, the success ratio increases as $d$ increases. It suggests that GREEDY1 can find the correct sets of variables not
only for AND/OR functions but also for most Boolean functions if $d$ is not small. In the case of $d = 2$, the success ratio is around 50%. This is reasonable because the number of AND/OR functions with two input variables is 8, where there are 16 possible Boolean functions with two input variables.

Next, we examined noisy case for combinations of $d = 3, 5$, $\epsilon = 0.1, 0.2, 0.3$ and $n = 500, 1000, 2000$, where we used general Boolean functions. In this case, we let $y_j(k) \neq f_j(x_{i_1}(k), \ldots, x_{i_d}(k))$ with probability $\epsilon$. The following table shows the result, where only the percentage of the success ratio (%) is given for each case.

<table>
<thead>
<tr>
<th>$d = 3$</th>
<th>$d = 3$</th>
<th>$d = 3$</th>
<th>$d = 5$</th>
<th>$d = 5$</th>
<th>$d = 5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\epsilon = 0.1$</td>
<td>$\epsilon = 0.2$</td>
<td>$\epsilon = 0.3$</td>
<td>$\epsilon = 0.1$</td>
<td>$\epsilon = 0.2$</td>
<td>$\epsilon = 0.3$</td>
</tr>
<tr>
<td>$m = 500$</td>
<td>68%</td>
<td>71%</td>
<td>79%</td>
<td>74%</td>
<td>32%</td>
</tr>
<tr>
<td>$m = 1000$</td>
<td>71%</td>
<td>72%</td>
<td>69%</td>
<td>89%</td>
<td>78%</td>
</tr>
<tr>
<td>$m = 2000$</td>
<td>77%</td>
<td>73%</td>
<td>76%</td>
<td>98%</td>
<td>97%</td>
</tr>
</tbody>
</table>

It is seen that the success ratio decreases as $\epsilon$ increases. But, for cases of $\epsilon \leq 0.2$ and $m = 2000$, the success ratios are close to those in the noiseless case. Therefore, it seems that GREEDY1 works well even in the noisy environment if sufficiently large number of data are provided.

Finally, we examined the effect of $n$ to the CPU time, where we used $d = 3$, $m = 1000$ and $\epsilon = 0$. The following table shows the result, where only CPU times (sec.) are given.

<table>
<thead>
<tr>
<th>$n = 1000$</th>
<th>$n = 2000$</th>
<th>$n = 3000$</th>
<th>$n = 4000$</th>
<th>$n = 5000$</th>
<th>$n = 6000$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.431</td>
<td>0.570</td>
<td>1.306</td>
<td>1.732</td>
<td>2.200</td>
<td>2.635</td>
</tr>
</tbody>
</table>

From this, it is seen that the CPU time is linear to $n$. Note that it took less than 3 seconds even in the case of $n = 6000$ (and $m = 1000$).

The research group directed by the third author is now making biological experiments on Yeast genes using several hundreds of mutants in which some genes are modified. This corresponds to a case of $n \approx 6000$, $m \approx 1000$, and $l \approx 6000$. Therefore, it is expected that it will take 3 x 6000 seconds (since it takes about 3 seconds per $y_j$) if GREEDY1 is applied to real experimental data on Yeast genes. Now we are preparing to apply GREEDY1 to analysis of Yeast genes since this computation time is not too long. Of course, real data are much more complex than artificial data used in the above. For example, instead of Boolean values, real numbers are used in real data. However, measurement errors are very large and thus real values should be classified into several discrete values. We are now seeking for appropriate threshold values for such classification.

Note that even in the average case, GREEDY1 sometimes fails to find the correct set of variables. However, it is not a crucial disadvantage in real applications. In practice, it is not required that all the relations found by the inference algorithm are correct because correctness of the relations or hypotheses will be checked in detail by an expert and/or by other experiments.
7 Concluding Remarks

In this paper, we have shown an efficient implementation of a simple greedy algorithm for inferring functional relations. It is fast enough to be applied to large data sets. We have also shown an average case analysis of the greedy algorithm though the analysis was restricted to a very special case. However, results of computational experiments suggest that the greedy algorithm outputs correct solutions in many average cases. Therefore, further studies on the average analysis might be possible and should be done. Of course, real data may be far from average case input data because input attributes are not necessarily independent. However, it is still guaranteed that GREEDY1 outputs an approximate solution even in the worst case [3]. We are now trying to apply the greedy algorithm to analysis of real experimental data on genetic networks. Although this attempt is not yet successful, we successfully applied a similar greedy algorithm to classification of cancer cells using gene expression data [5]. This fact suggests that greedy type algorithms may be useful for analysis of real data.

Finally, we would like to mention about the similarity between GREEDY1 and the algorithms for constructing decision trees [12]. In decision tree algorithms, input data are partitioned into smaller blocks as descending the tree from the root to the leaves, and such criteria as the entropy score is used for selecting the attribute at each node. Recall that in GREEDY1, input data are partitioned into smaller blocks as the number of iterations increases, and the attribute which covers the largest number of pairs is selected in each iteration. Since the number of covered pairs can be considered as a kind of score, the most important difference lies in that different attributes can be selected at different nodes of the same height in the decision tree, whereas the same attribute must be selected at all the nodes of the same height in GREEDY1 (see Fig. 3). Therefore, decision tree algorithms could be used for finding functional relations if we would put a restriction on the decision tree that the same attribute must be selected at all the nodes of the same height. Since various techniques have been developed for decision trees, it would be interesting to apply these techniques for improving GREEDY1.
Acknowledgements

This work was partially supported by the Grant-in-Aid for Scientific Research on Priority Areas, “Genome Science”, of the Ministry of Education, Science, Sports and Culture in Japan.

References

Graph-Based Induction for General Graph Structured Data and Its Application to Chemical Compound Data

Takashi Matsuda, Tadashi Horiuchi, Hiroshi Motoda and Takashi Washio

I.S.I.R., Osaka University, 8-1 Mihogaoka, Ibaraki, Osaka 567-0047, JAPAN

Abstract. Most of the relations are represented by a graph structure, e.g., chemical bonding, Web browsing record, DNA sequence, inference pattern (program trace), to name a few. Thus, efficiently finding characteristic substructures in a graph will be a useful technique in many important KDD/ML applications. However, graph pattern matching is a hard problem. We propose a machine learning technique called Graph-Based Induction (GBI) that efficiently extracts typical patterns from graph data in an approximate manner by stepwise pair expansion (pairwise chunking). It can handle general graph structured data, i.e., directed/undirected, colored/uncolored graphs with/without (self) loop and with colored/uncolored links. We show that its time complexity is almost linear with the size of graph. We, further, show that GBI can effectively be applied to the extraction of typical patterns from chemical compound data from which to generate classification rules, and that GBI also works as a feature construction component for other machine learning tools.

1 Introduction

Data having graph structure are abound in many practical fields such as molecular structures of chemical compounds, information flow patterns in the internet, DNA sequences and its 3D structures, and inference patterns (program traces of reasoning process). Thus, knowledge discovery from structured data is one of the major research topics in recent data mining and machine learning study. The approach proposed by Agrawal and Srikant to mine sequential patterns was one of the initiating works in this field [Agrawal95]. Since then several approaches have been proposed from different angles for sequential or structural data. Mannila et al. proposed an approach to mine frequent episodes from sequences [Mannila97]. Shinatani and Kitsuregawa devised a fast mining algorithm for sequential data using parallel processing [Shinatani98]. Srikant et al. used taxonomy hierarchy as background knowledge to mine association rules [Srikant97]. In this paper we focus on mining typical patterns in a graph structure data. By “typical” we mean frequently appearing subgraphs in the whole graph data.

Conventional empirical inductive learning methods use an attribute-value table as a data representation language and represent the relation between...
attribute values and classes by use of decision tree [Quinlan86] or induction rules [Michalski90,Clark89]. Association rules [Agrawal94] widely used in data mining fall in this type of data representation. However, the attribute-value table is not suitable for representing a more general and structural data. Inductive logic programming (ILP) [Muggleton89] which uses the first-order predicate logic can represent general relationship in data. Further, ILP has a merit that it can encode domain knowledge in the same representation language and the acquired knowledge can be added and utilized as the background knowledge. However, it still has time complexity problem. We have explored a different approach called GBI (Graph Based Induction) by directly encoding the target relations in form of general graph. Its expressiveness stands between the attribute-value table and the first-order predicate logic. GBI is similar to SUBDUE [Cook94] which also tries to extract substructure in a graph. What differs most is that GBI can find multiple patterns whereas SUBDUE can find only one substructure that minimizes the description length of the total graph using a computationally constrained beam search with a capability of inexact match. GBI is much faster and can handle much larger graphs.

Finding typical patterns from the whole graph involves graph matching as a subproblem which is known to be very hard [Fortin96]. The approach taken by GBI is quite simple and is heuristic-based. It is based on the notion of pairwise chunking and no backtracking is made (thus approximate). Its time complexity is almost linear with the size of the input graph. It can handle directed/undirected, colored/uncolored graphs with/without (self) loop and with colored/uncolored links. Some applications may require extracting patterns from a single huge graph of millions of nodes whereas some others from millions of small graphs of several tens of nodes. GBI works for both situations. Some applications may require only approximate solutions whereas some others exact solutions. As is evident, GBI gives only approximate solutions.

In the following sections, we describe the method of Graph-Based Induction (section 2), discuss the time complexity of GBI from both theoretical and experimental points of view (section 3), show how GBI is applied to chemical compound analyses (section 4), and conclude the paper by summarizing the main contribution.

2 Graph-Based Induction

2.1 Work in the Past

The idea of pairwise chunking dates back several years [Yoshida99]. It was originally proposed in the study of concept learning from graph structure data [Yoshida95,Yoshida94] and the method was called GBI (Graph-Based Induction). The central intuition behind is that a pattern that appears frequently enough is worth paying attention to and may represent an important concept (which is implicitly embedded in the input graph). In other words, the repeated patterns in the input graph represent typical characteristics of the given object environment. The original GBI was so formulated to minimize the graph size
by replacing each found pattern with one node that it repeatedly contracted the graph. The idea of pairwise chunking is shown in Figure 1. The graph size definition reflected the sizes of extracted patterns as well as the size of contracted graph. This prevented the algorithm from continually contracting, which meant the graph never became a single node. To reduce the complexity of search, the ordering of links is constrained to be identical if the two subgraphs are to match (meaning that the notion of isomorphism is relaxed), and an opportunistic beam search similar to genetic algorithm was used to arrive at approximate solutions. Because the primitive operation at each step in the search is to find a good set of linked pair nodes to chunk (pairwise chunking), we later adopted an indirect index rather than a direct estimate of the graph size to find the promising pairs. The new method was shown effective by applying it to build an on-line user adaptive interface [Motoda98]. However, in all of these, the type of graph allowed is restricted to basically a tree (i.e., a node can only have one outgoing link).

![Diagram](image)

**Fig. 1.** The idea of graph contraction by pairwise chunking

### 2.2 Representation of Graph by Table

In order to apply GBI to general graph structured data by removing the limitation on the type of graph and the constraint on the matching, we are currently representing a graph structured data using a set of tables that record the link information between nodes. For brevity we only explain the case of a directed graph with unlabeled links but the similar representation applies to a more general graph. A directed graph such as shown in Figure 2 can be represented using Table 1. For example, the first line in this table shows that the node No.1 has
node name “a” and also has nodes No.7 and No.10 as its child nodes. Thereby, the restriction of the link ordering is no more imposed.

Fig. 2. An example of directed graph

Table 1. An example of table representation translated from a directed graph

<table>
<thead>
<tr>
<th>Node No.</th>
<th>Node Name</th>
<th>Child Node No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
<td>7, 10</td>
</tr>
<tr>
<td>2</td>
<td>b</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>d</td>
<td>8, 11</td>
</tr>
<tr>
<td>4</td>
<td>b</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>a</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>b</td>
<td>9</td>
</tr>
<tr>
<td>7</td>
<td>d</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>b</td>
<td>11, 12</td>
</tr>
<tr>
<td>9</td>
<td>b</td>
<td>12</td>
</tr>
<tr>
<td>10</td>
<td>a</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>c</td>
<td></td>
</tr>
</tbody>
</table>

2.3 Basic Algorithm

The stepwise pair expansion (pairwise chunking) is performed by repeating the following three steps. The extracted chunks represent some characteristic properties of the input data.

S1. If there are patterns identical to the chunked pattern in the input graph, rewrite each of them to a single node of the same new label.

S2. Extract all linked pairs in the input graph.

S3. Select the most typical pair among the extracted pairs and register it as the pattern to chunk.

Each time we perform the pairwise chunking, we keep track of link information between nodes in order to be able to restore the original graph(s) or represent the extracted patterns in terms of the original nodes. This is realized by keeping two kinds of node information: “child node information” (which node in the pattern the link goes to) and “parent node information” (which node in the pattern the link comes from). These two kinds of information are also represented by tables (not shown here). Chunking operation can be handled by manipulating these three tables. The basic algorithm is shown in Figure 3. Currently we use a simple “frequency” of pairs as the evaluation function to use for stepwise pair expansion. Note that self-loop distinction flag is necessary to distinguish whether the parent and the child are the same node or not when
their labels are the same in counting the number of pairs. Further note that the algorithm can naturally handle a graph with loop substructures inside.

In order to apply the algorithm to undirected graphs, undirected graphs are converted to directed graphs by imposing a certain fixed order to node labels. For example, by ordering node labels as \( a \rightarrow b \rightarrow c \), the graph on the left in Figure 4 is uniquely converted to the directed graph on the right.

### Fig. 4. An example of conversion from undirected graph to directed graph

### 3 Performance Evaluation

Let \( N, L, P, C \) respectively denote the total number of nodes in the graph, the average number of links going out of one node, the number of different kinds of pairs in the graph, the number of different kinds of chunked patterns derived from the graph data. The time complexity to read the input data represented by
the table form as shown in the previous section is $O(NL)$, because the program must read all the link information in the graph. The time complexity to count the number of pairs for each kind is $O(NL)$, because the program must search all the links in the graph and the total number of links in the graph is $NL$. The time complexity to select the pair to be chunked is $O(P)$, because the program must find the most frequent pair by scanning all the pair information. The time complexity to perform the pairwise chunking is $O(NL)$, because the program must search all the links in the graph. The time complexity to update the pair information is $O(P)$, because the program must search all kinds of pairs in the graph. The program repeats the above process until the total number of chunked patterns becomes $C$. Therefore, the total time complexity is $O(NL + NL + C(P + NL + P)) = O(CP + NL)$.

We have confirmed this experimentally. Figure 6 shows the computation time (by a machine with Pentium II 400MHz CPU and 256MByte Memory) for artificially generated graphs with random structure: average number of outgoing links 3 and 5, and size of the graph 100 to 10,000 (See Figure 5 for an example graph). Figure 7 shows the computation time for graphs with random structure of fixed size (200 nodes): the outgoing link existence probability is changed from 10% to 100% and the number of node labels is changed from one to three. In all cases chunking was terminated when the maximum number of the pairs becomes less than 4% of total nodes in the graph.

As predicted, it is found that the computation time increases almost linearly with the size of the graph (number of nodes and links in the graph). It takes more time for the graphs with fewer node labels when the graph size is equal because the same patterns appear more often in the graphs with fewer node labels and there are more chances of pairwise chunking. It takes more time for the graphs with more links going out from each node when the graph size is equal because there are more candidate pairs to be chunked.

![Figure 5: Examples of graph data for experimental evaluation](image-url)
4 Extracting Patterns from Chemical Compound Data

4.1 Application to Carcinogenicity Data

The carcinogenesis prediction is one of the crucial problems in the chemical control of our environments and in the industrial development of new chemical compounds. However, the experiments on living bodies and environments to evaluate the carcinogenesis are quite expensive and very time consuming, and thus it is sometimes prohibitive to rely solely on experiments from both economical and efficiency point of view. It will be extremely useful if some of these properties can be shown predictive by the structure of the chemical substances before being actually synthesized.

We explored the possibility of predicting chemical carcinogenicity using our method. The task is to find structures typical to carcinogen of organic chlorides comprising C, H and Cl. The data were taken from the National Toxicology Program Database. We used the same small dataset that was used in [Matsumoto99] in which typical attributes representing substructure of the substances were symbolically extracted and used as inputs to a neural network by which to induce a classifier. The data consists of 41 organic chlorides out of which 31 are carcinogenic (positive examples) and 10 non-carcinogenic (negative examples). There are three kinds of links: single bonding, double bonding and aromatic bonding. Several examples of the organic chlorine compounds that have carcinogenicity are shown in Figure 8.

We treated the carbon, chlorine and benzene ring as distinctive nodes in graphs and ignored the hydrogen for our initial analyses. Further, we treated the single bond, double bond, triple bond and bond between benzene rings as links with different labels. Figure 9 shows an example of the conversion from
the organochlorine compound to the graph structured data. Each compound is associated with a \( \text{Log} P \) value (a measure of hydrophobicity). \( \text{GBI} \) can’t handle this value and thus, we ignored it. Computation time was not an issue for the graphs of sizes less than several tens nodes for \( \text{GBI} \). It ran in seconds.

\[ \text{Fig. 8.} \ \text{Examples of carcinogenetic organic chlorine compounds} \]

\[ \text{Fig. 9.} \ \text{Conversion to graph structured data} \]

Figure 10 shows patterns extracted from the positive cases and those extracted from the negative cases. By comparing these two sets of patterns, we can derive useful rules from the patterns which appear only in either positive patterns or negative patterns. A few examples are shown in Figure 11.

\[ \text{Fig. 10.} \ \text{Example patterns extracted from positive and negative examples} \]

### 4.2 Application to Mutagenicity Data

Some chemical compounds are known to cause frequent mutations which are structural alterations in DNA. Since there are so many chemical compounds, it is impossible to obtain mutagenicity data for every compound from biological experiments. Accurate evaluation of mutagenic activity from the chemical structure (structure-activity relationships) is really desirable. Furthermore, the mechanism of mutation is extremely complex and known only in part. Some evidence supports the existence of multiple mechanistic pathways for different
classes of chemical compounds. If this leads to a hypothesis for the key step in the mechanisms of mutation, it will be very important in mutagenesis research.

The data were taken from [Debnath90]. This data contains 230 aromatic or heteroaromatic nitro compounds. Mutagenesis activity was discretized into four categories: Inactive: activity = −99, Low: −99 < activity < 0.0, Medium: 0.0 ≤ activity < 3.0, High: 3.0 ≤ activity. By this categorization, we can classify the above compounds into 22 Inactive cases, 68 Low cases, 105 Medium cases and 35 High cases. The percentages of the classes of high, medium, low and inactive are 15.2%, 45.7%, 29.5% and 9.6% respectively. Each compound is associated with two other features: LogP and LUMO (a property of electric structure). We ignored these two values as before. We treated the single bond, double bond, triple bond and aromatic bond as different link labels, the carbon, chlorine and nitrogen as distinctive nodes but ignored the hydrogen as before. Furthermore, artificial links are added from each node to the other nodes where the number of links between the two is 2 to 6. This is to emulate variable nodes and links with “s” (wildcard).

In this analysis, we used GBI as a feature construction tool as well as using it as a pattern extractor from which to generate classification rules. These patterns and the LogP and LUMO are used as a set of features for the decision tree learner C4.5 [Quinlan93]. Each found pattern is evaluated and ranked by the following measure (Eq. 1), which is the maximum relative class ratio of the four classes. Here, i, l, m, h stands for the number of compounds for each class which has this pattern as a subgraph, and I, L, M, H stands for the original number of compounds for each class.

\[
M_{rel} = \max \left\{ \frac{i}{I}, \frac{l}{L}, \frac{m}{M}, \frac{h}{H} \right\}
\] (1)

Patterns of the top \( n \) (\( n = 10, 20, 30, 40, 50 \) with no artificial links and \( n = 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 \) with artificial links) in the ranking are used as pattern features for C4.5.

Figures 12, 13, 14 and 15 shows how the frequency count of the best pair to chunk, the value of the evaluation measure of the chunked pattern, its size
in terms of the number of nodes change as the chunking proceeds. As each of the linked pair nodes has a different chunking history, the latter two are not monotonic with chunking steps. Table 4.2 shows the prediction error of C4.5 for various parameter settings. “All” means that all chosen chunked patterns and both LogP and LUMO are used as the attributes. “Without cross validation (w/o CV)” means that the error is measured for all the data from which the tree is induced (training data). From this it is shown that the most decisive parameter is LogP (30.4%). But by choosing a good set of patterns as attributes, its effect is about the same as LogP (24.8%). Using all reduces the error to 13.9% for w/o CV. In all cases, the error of 10 fold cross validation is very large and the problem is felt very hard. The predictive error of 10 fold cross validation using all the chunked patterns by the method described in [Matsuda00] results in 52.6%. In this method, each pattern is converted to a rule using the same measure (Eq. 1) to assign a class. Rules are applied in increasing order of their support, and if the support is the same, rules of more complex pattern are applied first. The result is consistent with Table 4.2. One possible reason for this poor predictive capability is that the patterns are more descriptive than discriminative. This is not because the method is not working well, but because no classifier can result in a better predictive error. This means that the attributes we have prepared are not sufficient enough and we need some other features (e.g., 3-D structural information) to improve classification power. Some examples of the extracted rules are shown in Figure 18. We could not find the wildcard expression for a meaningful rule. This is because we neglected hydrogen.

Fig. 12. Max. no. of pairs and class predictive evaluation measure vs. chunk steps without artificial links

Fig. 13. Chunk size (no. of nodes) vs. chunk steps without artificial links

Fig. 14. Max. no. of pairs and class predictive evaluation measure vs. chunk steps with artificial links

Fig. 15. Chunk size (no. of nodes) vs. chunk steps with artificial links
In this paper, we showed how we can expand the capability of the Graph-Based Induction algorithm to handle more general graphs, i.e., directed graphs with 1) multiple inputs/outputs nodes and 2) loop structure (including a self-loop). The time complexity of the implemented program was evaluated from both theoretical and experimental points of view and it was shown that the algorithm runs almost linearly to the graph size. The algorithm was applied to two kinds of chemical compound data (carcinogenicity data and mutagenicity data) in order to extract useful patterns. It was also shown that GBI can be effectively used as a feature construction component of other machine learning method, which enables combined use of other non-structural features and structural features.
Ongoing and future work includes 1) deciding the rational evaluation method for derived rules in chemical compound data, 2) investigating the sensitivity of chunk ordering, 3) using statistical index (e.g., Gini Index [Breiman84]) or the description length (e.g., [Cook94]) in stead of the simple “frequency” as the evaluation function for stepwise expansion and 4) introducing a new index which corresponds to the notion of “similarity” of human concept.

Acknowledgement

The authors are very grateful to Prof. Takashi Okada in Kansei University for his extensive support to provide chemical compound data and the associated expertise in the chemistry domain.

References


Graph Based Induction for General Graph Structured Data


Discovering Characteristic Expressions from Literary Works: a New Text Analysis Method beyond \(N\)-Gram Statistics and KWIC

Masayuki Takeda\(^1\), Tetsuya Matsumoto\(^3\), Tomoko Fukuda\(^2\), and Ichirō Nanri\(^3\)

\(^1\) Department of Informatics, Kyushu University 33, Fukuoka 812-8581, Japan
\(^2\) Fukuoka Jo Gakuin College, Ogōri 838-0141, Japan
\(^3\) Junshin Women’s Junior College, Fukuoka 815-0036, Japan
{takeda, tetsuya}@i.kyushu-u.ac.jp
{tomoko-f@muc, nanri-i@msj}.biglobe.ne.jp

Abstract. We attempt to extract characteristic expressions from literary works. That is, our problem is, given literary works by a particular writer as positive examples and works by another writer as negative examples, to find expressions that appear frequently in the positive examples but do not so in the negative examples. It is considered as a special case of the optimal pattern discovery from textual data, in which only the substring patterns are considered. One reasonable approach is to create a list of substrings arranged in the descending order of their goodness, and to examine a first part of the list by a human expert. Since there is no word boundary in Japanese texts, a substring is often a fragment of a word or a phrase. How to assist the human expert is a key to success in discovery. In this paper, we propose (1) to restrict to the prime substrings in order to remove redundancy from the list, and (2) a way of browsing the neighbor of a focused string as well as its context. Using this method, we report successful results against two pairs of anthologies of classical Japanese poems. We expect that the extracted expressions will possibly lead to discovering overlooked aspects of individual poets.

1 Introduction

Analysis of expressions is one of the most fundamental methods in literary studies. In classical Japanese poetry Waka, there are strict rules in the choice and combination of poetic words. For instance, the word “Uguisu” (Japanese bush warbler) should be used linked with the word “Ume” (plum-blossom). It is significant in Waka studies, therefore, to consider how poets learned such rules and developed their own expressions. We would like to investigate how they learned certain expressions from their predecessors.

From this point of view, we have established a method that automatically extracts similar poems in expression from Waka database \[12, 11\]. Using this method, we discovered affinities of some unheeded poems with some earlier ones, and this raised an interesting issue for Waka studies and we could give a convincing conclusion to it.
We have proved that the poem (As if it were in darkness, / My parental heart / Is blind and lost in / The ways of caring about my child) by Fujiwara-no-Kanesuke, one of the renowned thirty-six poets, was in fact based on a model poem found in Kokinshū. The same poem had been interpreted just to show “frank utterance of parents’ care for their child.” Our study revealed the poet’s techniques in composition half hidden by the heart-warming feature of the poem by extracting the same structure between the two poems.

We have compared Tametadashū, the mysterious anthology unidentified in Japanese literary history, with a number of private anthologies edited after the middle of the Kamakura period (the thirteenth-century) using the same method, and found that there are about 10 pairs of similar poems between Tametadashū and Sokonshū, an anthology by Shōtetsu. The result suggests that the mysterious anthology was edited by a poet in the early Muromachi period (the fifteenth-century). There have been surmised dispute about the editing date since one scholar suggested the middle of Kamakura period as a probable one. We have had a strong evidence about this problem.

While continuing to find affinities among Waka poems, it is also necessary to give some additional conditions to the method when we compare poets in parent-child or teacher-student relationships. It is easily inferred that poet A (the master poet) greatly influences poet B (the disciple poet), and that the poems by poet B may have a lot of allusions to those by poet A. In fact, many scholars have already noticed such apparent literary relationships. In such cases, it is much more important to clarify the differences than to enumerate affinities. For example, when poet B hardly adopts the expressions frequently used by poet A, it will give us a chance to study their relationship in a different way.

In this paper, we will compare two poets’ private anthologies and try to make clear the differences and features in the similar expressions on the basis of their frequencies. This is in fact derived from the same methodology we have been practicing so far.

Shimozono et al. [10] formulated the problem of finding good pattern that distinguishes two sets Pos and Neg of strings (called positive and negative examples, respectively) as an instance of optimized pattern discovery, originally proposed by Fukuda et al. [6]. The problem is:

**Given.** Two finite sets Pos and Neg of non-empty strings.

**Find.** A pattern $\pi$ that minimizes a statistical measure function $G(\pi; \text{Pos, Neg})$.

For instance, the classification error, the information entropy, the Gini index, and the $\chi^2$ index are used as a statistical measure function $G$. Note that these measures are functions of 4-tuple $(p_1, n_1, p_0, n_0)$, where $p_1$ and $n_1$ are the numbers of strings in Pos and Neg that contain $\pi$, respectively, and $p_0 = |\text{Pos}| - p_1$ and $n_0 = |\text{Neg}| - n_1$. Namely, the goodness of a pattern depends only on its frequencies in Pos and Neg. Shimozono et al. [10] showed an efficient algorithm for proximity word-association patterns. Although only the classification error measure is dealt with in [10], the algorithm works for many other measures [1].
Our problem is regarded as a special case of this problem. That is, we deal with only the patterns of the form \( *w^* \), where \( * \) is a wildcard that matches any string and \( w \) is a non-empty string. We call such patterns the substring patterns. We have a trivial \( O(m) \) time and space algorithm since there exist essentially \( O(m) \) candidates for the best substring \( w \), where \( m \) is the total length of the strings in \( S = Pos \cup Neg \), and therefore difficulty lies mainly on how to find an appropriate goodness measure.

However, it seems that no matter what measure we use, most of ‘good’ patterns are not so good in practice; they are obvious and worthless in many cases. For this reason, discovery requires an effort by domain experts to examine an upper part of list of patterns arranged in the decreasing order of the goodness. This corresponds to the step of interpreting mined patterns, a ‘postprocessing’ of data mining in knowledge discovery process [5]. We believe that how to support this step the domain experts is a key to success. In this paper we tackle this problem with the weapon of stringology.

Let \( \text{Sub}(S) \) be the set of substrings of strings in \( S \). All the strings in \( \text{Sub}(S) \) could be candidates for characteristic expressions. The candidate strings, however, are not independent each other in the sense that some strings subsume other ones. Moreover, we are frequently faced with the case that two strings in the superstring–substring relation have the same frequency, and therefore have the same value of goodness. (Recall that the goodness measures considered in this paper depend only on the frequencies in positive and negative examples.) For instance, in the first eight imperial anthologies (from Kokinsū to ShinKokinshū), every occurrence of the string “SHI-NO-NO-YA” (consisting of 4 syllables) is a substring of “YO-SHI-NO-NO-YA-MA” (6 syllables; Mt. Yoshino). In such case we want to remove shorter strings.

On the other hand, researchers are interested in not just frequency of a word but in its actual use. That is, they would access the context of each word appearance. In addition, a candidate string is often a fragment of a word or a phrase and seems to be meaningless. In order to find the word or the phrase that contains this fragment as a substring, the researchers need to check what string immediately precedes (follows) this fragment, for every occurrence of it. Thus, it is necessary for the researchers to see the possible superstrings of a focused candidate string.

In order to introduce a structure into \( \text{Sub}(S) \), we use the equivalence relation, first defined by Blumer et al. [2], which has the following properties:

– Each equivalence class has a unique longest string which contains any other member as a substring, and we regard it as the representative of the class.
– All the strings in an equivalence class have the same frequency in \( S \) (and therefore have the same value of goodness.)
– The number of equivalence classes is linear with respect to the total length of the strings in \( S \).

A string in \( \text{Sub}(S) \) is said to be prime if it is the representative of some equivalence class under this equivalence relation. The basic idea is to consider only the prime substrings as candidates for characteristic expressions. Any non-prime
substring with a high goodness value can be found as a part of the prime substring equivalent to it, even though we remove all the non-prime substrings from the list.

It should be stated that the data structure called the suffix tree [3] exploits a similar and more popular equivalence relation, which also satisfies the above three conditions. This equivalence relation is finer than the one by Blumer et al. [2], and therefore includes much more equivalence classes. From the viewpoint of computational complexity, this is not a significant difference because the two equivalence relations both satisfy the third condition. However, the difference is crucial for researchers who must check the candidate strings. In fact the number of equivalence classes was reduced to approximately 1/4 by using the one by Blumer et al. in our experiment using Waka poems, as will be shown in Section 5.

We would create a candidate list only of prime substrings. To support the domain experts who inspect the list, we develop a kind of browser to see:

- The non-prime substrings equivalent to each prime substring.
- The superstring–substring relationships between the prime strings.

The symmetric compact DAWG [2] for $S$ is useful for this purpose. It is a directed acyclic graph such that the vertices are the prime substrings, and the labeled edges represent: What happens when appending a possible letter to the left or right end of a prime substring? For instance, in the first eight imperial anthologies, appending a single letter “YA” to the right end of the string “YO-SHI-NO-NO” yields “YO-SHI-NO-NO-YA-MA.” In the case of another letter “HA,” it yields the string “MI-YO-SHI-NO-NO-HA-NA.” (Notice that not only “HA-NA” is appended to the right, but also “MI” is appended to the left.) On the other hand, the result of appending “MI” to the left of the same string is simply “MI-YO-SHI-NO-NO.”

We would consider to draw interactively a subgraph of the symmetric compact DAWG whose nodes are limited to the ones reachable from/to a focused node (namely, they are substrings or superstrings of the focused string). This subgraph, however, is rather complicated for a reasonable size $S$ in the real world. For this reason, we shall instead draw:

- The subgraph consisting of the prime substrings that are substrings of the focused one. (It is small enough because the focused string is rather short in practice.)
- The right and left context trees which represent the same information as the subgraph consisting of the prime substrings that are superstrings of the focused one.

The right context tree of a prime substring $x$ is essentially the same as the subtree of the node $x$ in the suffix tree [3] for $S$, but augmented by adding to every node $xy$ a label of $\gamma[x]y$ such that $\gamma xy$ is the prime substring that is equivalent to $xy$. The left context tree is defined in a similar way.

Most of the existing text analysis tools have the KWIC (Key Word In Context) display [8]. The left and the right context trees of a focused prime substring enable researchers to grasp the context of the string more quickly compared with
KWIC, especially when the number of appearances of the string being checked is relatively large.

Using the suggested method, we extracted the strings that differentiate two anthologies, scholarly scrutinized them, and succeeded in finding the characteristic expressions. We compared Sankashū by Saigyō with Shūgysōku by Jien, and Shūgusō by Fujiwara-no-Teika with Tameišu by his son, Fujiwara-no-Tameie separately, and procured such expressions highlighting differences between each two anthologies.

Saigyō and Jien were both priests, but their lives were so contrastive in status and social circumstances. Being a son of the famous regent, Jien could not make a break with the government. However, it is well known that Jien sought much help from Saigyō in his last days not only about poetic composition but also about the way of life. For instance, he confessed his plan of becoming a hermit.

On the other hand, Tameie was rigorously trained by his own father, Teika, for he was in the direct descent of the Mikohidari dynasty famous for producing poets. The number of Teika’s poems Tameie referred to is not small.

In each pair of anthologies, there necessarily exist similar poems. However, as we have demonstrated so far, we could collect certain differences in expressions. This, we expect, will possibly lead to discovering overlooked aspects of individual poets.

It may be relevant to mention that this work is a multidisciplinary study between the literature and the computer science. In fact, the second author from the last is a Waka researcher and the last author is a linguist in Japanese language.

2 Substring Statistics for Japanese Literary Studies

If we want to use a word statistics, we need to perform a task of word segmentation, because there is no word boundary in texts written in Japanese, like other non-Western languages. This is a difficult and time-consuming task. For example, it is reported in [9] that it took eight years to build a part-of-speech tagged corpus of “the Tale of Genji.” In the case of Waka poetry, building such corpora is more difficult because the frequently used technique, called “Kake-kotoba,” which exploits the ambiguities of a word or part of word. We can say that a unique word segmentation is essentially impossible in such a situation.

Recently some researchers noticed that using substring statistics instead of word statistics is in fact useful in expression analysis, although a substring could be a fragment of a word or a phrase. For example, Kondo [7] proposed an expression analysis method based on n-gram statistics, and reported some differences between the expressions used by poets and poetess in Kokinshū, the most famous imperial anthology. The n-gram statistics is essentially the same as

1 A sort of homonymic punning where the double meaning of a word or part of word is exploited. In English it is usually called the pivot words because it is used as a pivot between two series of sounds with overlapping syntactical and semantic patterns.
the substring statistics since \( n \) cannot be fixed. This work opened the door for the application of substring statistics to Japanese literary works.

However, Kondo restricted herself to the substrings which (1) are of length from 3 to 7, (2) occur more than once, and (3) are used only by male (not used by female), to ease the burden. In this paper, we will show that such a restriction can be removed by exploiting combinatorial properties on strings.

3 Prime Substrings

In this section, we give a formal definition of the prime substrings, and present some of their properties.

3.1 Preliminary

Let \( \Sigma \) be a finite alphabet. An element of \( \Sigma^* \) is called a string. Strings \( x, y, \) and \( z \) are said to be a prefix, substring, and suffix of the string \( u = xyz \), respectively. A string \( u \) is said to be a superstring of a string \( y \) if \( y \) is a substring of \( u \). The length of a string \( u \) is denoted by \(|u|\). The empty string is denoted by \( \epsilon \), that is, \(|\epsilon| = 0\). Let \( \Sigma^+ = \Sigma^* - \{\epsilon\} \). The \( i \)th symbol of a string \( u \) is denoted by \( u[i] \) for \( 1 \leq i \leq |u| \), and the substring of a string \( u \) that begins at position \( i \) and ends at position \( j \) is denoted by \( u[i:j] \) for \( 1 \leq i \leq j \leq |u| \). For convenience, let \( u[i:j] = \epsilon \) for \( j < i \). Let \( \text{Sub}(w) \) denote the set of substrings of \( w \). Let \( \text{Sub}(S) = \bigcup_{w \in S} \text{Sub}(w) \) for a set \( S \) of strings. For a set \( S \) of strings, denote by \( \|S\| \) the total length of the strings in \( S \), and denote by \(|S|\) the cardinality of \( S \).

3.2 Definition of Prime Substrings

**Definition 1.** Let \( S \) be a non-empty finite subset of \( \Sigma^+ \). For any \( x \in \text{Sub}(S) \), let

\[
\text{Beginpos}_S(x) = \{ \langle w, j \rangle \mid w \in S, 0 \leq j \leq |w|, x = w[j+1:j+|x|] \}.
\]

\[
\text{Endpos}_S(x) = \{ \langle w, j \rangle \mid w \in S, 0 \leq j \leq |w|, x = w[j-|x|+1:j] \}.
\]

For any \( x \notin \text{Sub}(S) \), let \( \text{Beginpos}_S(x) = \text{Endpos}_S(x) = \emptyset \).

For example, if \( S = \{ \text{babbe, ababb} \} \), then \( \text{Beginpos}_S(a) = \text{Beginpos}_S(ab) = \{ \langle \text{babbe, 1} \rangle, \langle \text{ababb, 0} \rangle, \langle \text{ababb, 2} \rangle \} \), \( \text{Beginpos}_S(c) = \{ \langle \text{babbe, 4} \rangle \} \), and \( \text{Endpos}_S(ab) = \text{Endpos}_S(abbe) = \{ \langle \text{babbe, 4} \rangle, \langle \text{ababb, 5} \rangle \} \).

From here on, we omit the set \( S \), and write simply as \( \text{Beginpos} \) and \( \text{Endpos} \).

**Definition 2.** Let \( x \) and \( y \) be any strings in \( \Sigma^* \). We write as \( x \equiv_L y \) if \( \text{Beginpos}(x) = \text{Beginpos}(y) \), and write as \( x \equiv_R y \) if \( \text{Endpos}(x) = \text{Endpos}(y) \). The equivalence class of a string \( x \in \Sigma^* \) with respect to \( \equiv_L \) (resp. \( \equiv_R \)) is denoted by \([x]_{\equiv_L} \) (resp. \([x]_{\equiv_R} \)).
If \( S = \{ babbc, ababb \} \), then \( [\varepsilon]_L = [\varepsilon]_R = \{ \varepsilon \} \), \( [a]_L = \{ a, ab \} \), \( [bb]_L = \{ bb, abb, babbc \} \), and \( [c]_R = \{ c, bc, bbc, abbc, babbc \} \).

Note that all strings that are not in \( \text{Sub}(S) \) form one equivalence class under \( \equiv_L \). This equivalence class called the degenerate class. All other classes are called nondegenerate.

It follows from the definition of \( \equiv_L \) that, if \( x \) and \( y \) are strings in the same nondegenerate class under \( \equiv_L \), then either \( x \) is a suffix of \( y \), or vice versa. Therefore, each nondegenerate equivalence class in \( \equiv_L \) has a unique longest member.

Similar discussion holds for \( \equiv_R \).

**Definition 3.** For any string \( x \) in \( \text{Sub}(S) \), let \( \overleftarrow{x} \) and \( \overrightarrow{x} \) denote the unique longest members of \( [x]_L \) and \( [x]_R \), respectively.

For any string \( x \) in \( \text{Sub}(S) \), there uniquely exist strings \( \alpha \) and \( \beta \) such that \( \overleftarrow{x} = \alpha x \) and \( \overrightarrow{x} = x\beta \). In the running example, we have \( \overleftarrow{\varepsilon} = \varepsilon \), \( \overleftarrow{a} = ab \), \( \overleftarrow{bb} = babb \), \( \overleftarrow{bab} = babbc \), and \( \overleftarrow{c} = babbc \).

Figure 1 shows the suffix tree [3] for \( S = \{ babbc, ababb \} \). Note that the nodes of the suffix tree are the strings with \( x = \overleftarrow{x} \). On the other hand, Fig. 2 shows the directed acyclic word graph (DAWG for short) [3] for \( S \). Note that the nodes are the nondegenerate equivalence classes in \( \equiv_R \). The DAWG is the smallest automaton that recognizes the set of suffixes of the strings of \( S \) if we designate some nodes as final states appropriately.

**Definition 4.** For any string \( x \) in \( \text{Sub}(S) \), let \( \overleftarrow{x}^* \) be the string \( \alpha x \beta \) such that \( \alpha \) and \( \beta \) are the strings satisfying \( \overleftarrow{x} = \alpha x \) and \( \overrightarrow{x} = x\beta \).

In the running example, \( \overleftarrow{\varepsilon}^* = \varepsilon \), \( \overleftarrow{a}^* = ab \), \( \overleftarrow{bb}^* = babb \), and \( \overleftarrow{bab}^* = babbc \).

**Definition 5.** Strings \( x \) and \( y \) are said to be equivalent on \( S \) if and only if

1. \( x \notin \text{Sub}(S) \) and \( y \notin \text{Sub}(S) \), or
2. \( x, y \in \text{Sub}(S) \) and \( \overleftarrow{x} = \overleftarrow{y} \).
This equivalence relation is denoted by \( x \equiv y \). The equivalence class of \( x \) under \( \equiv \) is denoted by \([x]_\equiv\).

Notice that, for any string \( x \) in \( \text{Sub}(S) \), the string \( \overrightarrow{x} \) is the longest member of \([x]_\equiv\). Intuitively, \( \overrightarrow{x} = \alpha x \beta \) means that:

- Every time \( x \) occurs in \( S \), it is preceded by \( \alpha \) and followed by \( \beta \).
- Strings \( \alpha \) and \( \beta \) are as long as possible.

Now, we are ready to define the prime substrings.

**Definition 6.** A string \( x \) in \( \text{Sub}(S) \) is said to be prime if \( \overrightarrow{x} = x \).

**Lemma 1 (Blumer et al. (1987)).** The equivalence relation \( \equiv \) is the transitive closure of the relation \( \equiv^R \cup \equiv_L \).

It follows from the above lemma that \( \overrightarrow{x} = (\overrightarrow{x}) = (\overleftarrow{x}) \) for any string \( x \) in \( \text{Sub}(S) \).

### 3.3 Properties of Prime Substrings

Recall that the number of all substrings of \( S \) is \( O(\|S\|^2) \). This can be reduced to \( O(\|S\|) \) by considering the substrings \( x \) such that \( \overrightarrow{x} = x \). In fact the suffix tree is a data structure that exploits this property. Similarly, the DAWG achieves its \( O(\|S\|) \) space complexity by identifying every substring \( x \) with the substring \( \overrightarrow{x} \).

Since the number of prime substrings of \( S \) is also \( O(\|S\|) \), it seems that there is no advantage in considering only the prime substrings. In practical application, however, it has a big advantage because the users do not have to examine non-prime substrings. The next lemma gives more tight bounds.

**Lemma 2 (Blumer et al. (1987)).** Assume \( \|S\| > 1 \). The number of the nondegenerate equivalence classes in \( \equiv_L \ (\equiv_R) \) is at most \( 2\|S\| - 1 \). The number of the nondegenerate equivalence classes in \( \equiv \) is at most \( \|S\| + |S| \).

Thus, the number of prime substrings of \( S \) is at most \( \|S\| + |S| \). In practice, the number of prime substrings is usually smaller than both the number of substrings \( x \) with \( \overrightarrow{x} = x \) and the number of substrings \( x \) with \( \overleftarrow{x} = x \), which are upper-bounded by \( 2\|S\| - 1 \).

Let \( \text{Prime}(S) \) be the set of prime substrings of \( S \), i.e. \( \text{Prime}(S) = \{\overrightarrow{x} \mid x \in \text{Sub}(S)\} \).
Definition 7. The symmetric compact DAWG for $S$ is the triple $(V, E_L, E_R)$ where $V = \text{Prime}(S)$ is the set of vertices, and $E_L, E_R \subseteq V \times V \times \Sigma^*$ are two kinds of labeled edges defined by:

\[
E_L = \{(x, \gamma \sigma x \delta, \gamma \sigma) \mid x \in V, \sigma \in \Sigma, \delta \in \Sigma^*, \gamma \sigma x \delta = \overline{\sigma x} \}
\]

\[
E_R = \{(x, \delta x \sigma \gamma, \sigma \gamma) \mid x \in V, \sigma \in \Sigma, \delta, \gamma \in \Sigma^*, \delta x \sigma \gamma = \overline{x \sigma} \}.
\]

The compact DAWG for $S$ is the graph $(V, E_R)$.

Figure 3 show the compact DAWG for the running example. Compared with the suffix tree of Fig. 1 and with the DAWG of Fig. 2, the nodes represent only the prime substrings, and therefore it is expected that the number of nodes are smaller than those of the suffix tree and the DAWG.

Figure 4 shows the symmetric compact DAWG for the running example.

Fig. 3. Compact DAWG for $S = \{babbc, ababb\}$. The edge labeled by $\sigma \gamma$ from the node $x$ to the node $y$ corresponds to the fact that $x \sigma$ is equivalent to $y$ under $\equiv$, where $x, y \in \text{Prime}(S), \sigma \in \Sigma, \gamma \in \Sigma^*$. For instance, the arrow labeled “$abb$” from the node “$b$” to the node “$babbc$” means that $ba$ is equivalent to $babbc$ under $\equiv$.

Fig. 4. Symmetric compact DAWG for $S = \{babbc, ababb\}$. The solid and the broken arrows represent the edges in $E_R$ and $E_L$, respectively.

Lemma 3 (Blumer et al. (1987)). Both the compact DAWG and the symmetric compact DAWG for $S$ can be constructed in $O(|S|)$ time and space.
4 Browser for Finding Characteristic Substrings

We would find ‘good’ patterns from text strings in the following manner.

1. Choose an appropriate measure of the goodness of the patterns which depends only on their frequencies.
2. Compute the goodness of all possible patterns, and create a list of patterns arranged in the decreasing order.
3. Evaluate an upper part of the list by a domain expert.

However, most of patterns in the upper part of the list are not so good in practice. They are obvious and/or worthless in many cases. For this reason, it is most important to develop an effective way of supporting the domain expert in the third step.

The patterns we are dealing with are restricted to the substring patterns, which are of the form $*w*$, where $*$ is the wildcard that matches any string in $\Sigma^*$ and $w$ is a non-empty string in $\Sigma^+$. Since we assume that the goodness of a substring pattern depends only on its frequencies, we can restrict $w$ to a prime substring. Thus, we exclude the non-prime substrings from the list of candidates for characteristic expressions. There is no risk of overlooking any good non-prime substring $x$ because it must appear as a substring in the prime substring $\mathcal{P}$ with the same value of goodness.

To support the expert we develop a browser to see the following information.

- Among the substrings of the focused string, which are equivalent to it?
- Among the prime substrings, which are superstrings (substrings) of the focused string?

The symmetric compact DAWG is useful for this purpose. That is, the strings equivalent to a focused prime substring $x$ can be obtained from the incoming edges of the node $x$. If $(x', x, \sigma \gamma) \in E_R$, then any string $z$ such that $x' \sigma \geq z \geq x$ is equivalent to $x$, where $u \geq v$ means that $u$ is a substring of $v$. Similarly, if $(x', x, \sigma \gamma) \in E_L$, any string $z$ such that $\sigma x' \geq z \geq x$ are equivalent to $x$. The string $x' \sigma (\sigma x')$ appears exactly once within the string $x$, and so it is easy to grasp the strings $z$ satisfying the inequality.

To give the domain expert an illustration of the superstring–substring relation on the prime substrings, we would consider to draw interactively the subgraph of the symmetric compact DAWG in which the nodes are restricted to the ones reachable from/to a focused node. However, the subgraph is still large and complicated for a reasonable size set $S$ of strings in the real world. Instead, we shall draw

- The subgraph consisting of the prime substrings that are substrings of the focused one. (It is small enough because the focused string is rather short in practice.)
- The right and left context trees which represent the same information as the subgraph consisting of the prime substrings that are superstrings of the focused one. The former represents the information in $E_R$, and the latter in $E_L$. 

The right context tree of a prime substring \( x \) is essentially the same as the subtree of the node \( x \) in the suffix tree [3] for \( S \), but augmented by adding to every node \( xy \), \( \gamma xy = \rightarrow (y \in \Sigma^*) \), a label of “\( \gamma[x]y \)" such that \( \gamma xy = \leftarrow y \). (The left context tree is defined in a similar way.) There may be two nodes \( xy_1 \) and \( xy_2 \) such that \( y_1 \neq y_2 \) but \( \leftarrow y_1 = \leftarrow y_2 \). Therefore more than one node may have the same label if ignoring the square brackets ([,]). Figure 5 shows the left and the right context trees of \( x = b \) for \( S = \{babbc, ababb\} \).

5 Experimental Results

We carried out experiments for Waka poems and prose texts.

5.1 Goodness Measures

The statistical measures like the classification error, the information entropy, and the Gini index are abstracted as follows [4]. Let \( p_1 \) and \( n_1 \) denote the numbers of strings in \( Pos \) and \( Neg \) that contain \( \pi \), respectively, and let \( p_0 = |Pos| - p_1 \) and \( n_0 = |Neg| - n_1 \). Let

\[
 f(p_1, n_1, p_0, n_0) = \frac{p_1 + n_1}{N} \cdot \psi \left( \frac{p_1}{p_1 + n_1} \right) + \frac{p_0 + n_0}{N} \cdot \psi \left( \frac{p_0}{p_0 + n_0} \right),
\]

where \( N = p_1 + n_1 + p_0 + n_0 \) and \( \psi \) is a non-negative function with the following properties:

- \( \psi(1/2) \geq \psi(r) \) for any \( r \in [0, 1] \).
- \( \psi(0) = \psi(1) = 0 \).
- \( \psi(r) \) increases in \( r \) on \( [0, 1/2] \) and decreases in \( r \) on \( [1/2, 1] \).

The information entropy measure is the function \( f \) such that

\[
 \psi(r) = -r \log r - (1 - r) \log(1 - r).
\]

The classification error and the Gini index measures are also obtained by letting \( \psi(r) = \min(r, 1 - r) \) and \( \psi(r) = 2r(1 - r) \), respectively.

In our experiment, we used the information entropy measure. We also tested the classification error and the Gini index, but no substantial differences were observed.
5.2 Text Strings We Used

Our experiments were performed against the following classical Japanese literary works.

(A) Two private anthologies Sankashū by the priest Saigyō (1118–1190), and Shūgyokushū by the priest Jien (1155–1225). It is well-known that Saigyō was a great influence on Jien. In fact, Jien composed many poems using similar expressions preferred by Saigyō.

(B) Two private anthologies Shūigusō by Fujiwara no Teika (1162–1241), and Tameieshū by Fujiwara no Tameie (1198–1275). Teika is a poet and a literary theorist, who ranks among the greatest of Waka poets. The poems of his son Tameie were influenced by the poems and the theory of Teika.

(C) The Tale of Genji, written by Murasaki Shikibu (Lady Murasaki), which is considered the first novel ever written. It consists of 54 chapters. Some scholars have convinced that the Tale of Genji is not all by the same writer. Especially, it is often claimed that the author of the main chapters and the author(s) of the Uji chapters (the last 10 chapters) are not the same person. The aim is to compare the last 10 chapters with the other 44 chapters.

Table 1 shows the numbers of nondegenerate equivalence classes for the text strings of (A), (B), and (C), under the three equivalence relations \( \equiv_L \), \( \equiv_R \), and \( \equiv \). The result imply that the limitation to the prime substrings drastically reduce the number of the substrings to be examined by human experts. Thus, the notion of the prime substrings makes the substring statistics based text analysis be realistic.

Table 1. Comparison of the numbers of nondegenerate equivalence classes against the three equivalence relations, \( \equiv_L \), \( \equiv_R \), and \( \equiv \).

| Anthologies                   | \( |S| \) | \( |S| \) | \( |Sub(S)| \) | \# nondegen. equiv. classes |
|-------------------------------|-------|-------|-------------|---------------------------|
| Pos                           |       |       |             |                           |
| Sankashū (1,552 poems)        | 7,355 | 229,728 | 2,817,436   | 259,576                   |
| Shūgyokushū (5,803 poems)     |       |       |             |                           |
| Pos                           |       |       |             |                           |
| Shūigusō (2,985 poems)        | 5,086 | 158,290 | 1,989,446   | 183,358                   |
| Tameieshū (2,101 poems)       |       |       |             |                           |
| Pos                           |       |       |             |                           |
| Tale of Genji (Chap. 1–44)    | 54    | 859,796 | 1,493,709,707 | 1,182,601
| (Chap. 45–54)                 |       |       |             | 1,181,439 251,343 |

5.3 Characteristic Expressions Extracted

It should be noted that a Waka poem consists of five lines, and we can restrict to the substrings of lines of Waka poems. In order to exclude the substrings which stretch over two or more lines, we let \( S \) be the set of lines of Waka poems in the anthologies. Then, we have 50,345 and 37,477 prime substrings to be examined.
for (A) Sankashū and Shūgyokushū, and for (B) Shūigushō and Tameieshū, respectively.

On the other hand, there are no punctuations in the texts of “the Tale of Genji” we used. We have 54 text strings each of which corresponds to a chapter of the book. For this reason, we define \( p_1 (n_1) \) to be the number of occurrences of pattern in \( Pos (Neg) \), not to be the number of strings that contain it.

We created lists of prime substrings arranged using the information entropy measure. Table 2 shows the best 40 prime substrings for (A) Sankashū and Shūgyokushū, and (B) Shūigusō and Tameieshū. (We omitted the list for “the Tale of Genji.”) From the upper part of the list, we can notice the following, by using a prototype of our browser.

– While Shūgyokushū has Buddhist terms like “Nori-no-michi” (the road of dharma), “Nori-no-hana” (the flower of dharma), and “Makoto-no-michi” (the road of truth) in 20, 16, and 18 poems, respectively, there is no such expression in Sankashū. It should be mentioned that the first two terms were obtained by browsing the right context tree of the string “no-ri-no,” which ranks the 15th, and the last one was obtained from the left context tree of the string “no-mi-chi,” which ranks the 24th. See Fig. 6.

– There are 21 examples of “···wo Ikanisemu” (most of them are “Mi wo Ikanisemu”) in Shūgyokushū and there is none in Sankashū. This is interesting and seems to suggest their differences in their beliefs in Buddhism and in their ways of lives. The expression “wo Ikanisemu” was obtained from the left context tree of the string “i-ka-ni-se,” which ranks the 31st.

– In Tameieshū, there are many expressions using “Oi-no-nezame” (wakeful night for the aged) and “Oi-no-Namida” (tears of the aged), but there is not such in Shūigusō at all. Though Tameie was conscious of old age in his poems, he did not live longer than his father had done. (Teika died at 80 and Tameie at 78.) Surveying Shinpen-Kokkataikan, a collection of 1,162 anthologies of Waka poems (about 450,000 poems in total), we find that the expressions “Oi-no-nezame” and “Oi-no-Namida” most frequently appear in Tameieshū. These expressions, therefore, definitely characterize Tameie’s poetry. In his last days, he was involved in the family feud about his successor. It was resulted in dividing the dynasty into three as Nijō, Kyōgoku, and Reizei in the next generation. This is quite a contrast to the case of Teika, who could decide Tameie as his only one successor.

– In the narrative or the characters’ speech in the Uji chapters of the Tale of Genji, we observed some characteristic expressions like “Ikanimo-ikanimo,” which ranks the 90th, and “Shikiwazakana,” which ranks the 108th. But most of the strings collected from the book are proper nouns like characters’ names, titles and place names, and they largely depend on the story settings and the characters. So we have to say that this is far from discovery. We could have removed such strings under conditions either of \( p_1 = 0 \) or of \( n_1 = 0 \), but there was risk of excluding some other important strings too. We need to consider how to adapt a filtering method to prose works.
Table 2. Best 40 prime substrings from two pairs of private anthologies. The hyphens '-' are inserted between syllables, each of which was written as one Kana character although romanized here.

<table>
<thead>
<tr>
<th>G</th>
<th>P1</th>
<th>N1</th>
<th>Substring</th>
<th>G</th>
<th>P1</th>
<th>N1</th>
<th>Substring</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5115</td>
<td>35</td>
<td>391  KU-NO</td>
<td>1</td>
<td>66665</td>
<td>14</td>
<td>103 O-i</td>
</tr>
<tr>
<td>2</td>
<td>5115</td>
<td>35</td>
<td>351  HA-RI-NO</td>
<td>2</td>
<td>66656</td>
<td>2</td>
<td>636 I-I-NO</td>
</tr>
<tr>
<td>3</td>
<td>5118</td>
<td>16</td>
<td>271  MI-YO</td>
<td>3</td>
<td>5171</td>
<td>6</td>
<td>54 WO-KU-RA</td>
</tr>
<tr>
<td>4</td>
<td>5120</td>
<td>919</td>
<td>2822 TE</td>
<td>4</td>
<td>6723</td>
<td>11</td>
<td>60 WO-KU</td>
</tr>
<tr>
<td>5</td>
<td>5122</td>
<td>29</td>
<td>344  NO-SO</td>
<td>5</td>
<td>5731</td>
<td>246</td>
<td>74 SO-RA</td>
</tr>
<tr>
<td>6</td>
<td>5124</td>
<td>41</td>
<td>29  KO-KO-CHI</td>
<td>6</td>
<td>6735</td>
<td>4</td>
<td>38 NI-KI-RU</td>
</tr>
<tr>
<td>7</td>
<td>5124</td>
<td>44</td>
<td>33  KO-KO-CHI</td>
<td>7</td>
<td>6736</td>
<td>109</td>
<td>168 KO-RA</td>
</tr>
<tr>
<td>8</td>
<td>5126</td>
<td>21</td>
<td>273  NO-SO-RA</td>
<td>8</td>
<td>6738</td>
<td>3</td>
<td>34 KU-RA-YA-MA</td>
</tr>
<tr>
<td>9</td>
<td>5127</td>
<td>60</td>
<td>495  SO-RA</td>
<td>9</td>
<td>6739</td>
<td>54</td>
<td>106 N-NO</td>
</tr>
<tr>
<td>10</td>
<td>5129</td>
<td>9</td>
<td>163 MI-YO-RI</td>
<td>10</td>
<td>6739</td>
<td>3</td>
<td>33 WO-KU-RA-YA-MA</td>
</tr>
<tr>
<td>11</td>
<td>5131</td>
<td>3</td>
<td>129 SU-MI-YO</td>
<td>11</td>
<td>6740</td>
<td>0</td>
<td>23 O-I-NO-NE</td>
</tr>
<tr>
<td>12</td>
<td>5131</td>
<td>99</td>
<td>167 MA-SHI</td>
<td>12</td>
<td>6741</td>
<td>69</td>
<td>7 NA-KA-ME</td>
</tr>
<tr>
<td>13</td>
<td>5132</td>
<td>7</td>
<td>150 MI-YO-RI-NO</td>
<td>13</td>
<td>6746</td>
<td>6</td>
<td>35 KU-RA-YA</td>
</tr>
<tr>
<td>14</td>
<td>5132</td>
<td>3</td>
<td>114 SU-MI-YO-RI-NO</td>
<td>14</td>
<td>6747</td>
<td>0</td>
<td>19 O-I-NO-NE-SA-ME</td>
</tr>
<tr>
<td>15</td>
<td>5133</td>
<td>7</td>
<td>147 NO-SI-RI</td>
<td>15</td>
<td>6747</td>
<td>88</td>
<td>16 SO-TE-NO</td>
</tr>
<tr>
<td>16</td>
<td>5134</td>
<td>4872281</td>
<td>307 YO</td>
<td>16</td>
<td>6748</td>
<td>292</td>
<td>115 I-RO</td>
</tr>
<tr>
<td>19</td>
<td>5134</td>
<td>53</td>
<td>418  YU-FU</td>
<td>17</td>
<td>6748</td>
<td>57</td>
<td>99 NI-KE</td>
</tr>
<tr>
<td>20</td>
<td>5134</td>
<td>8</td>
<td>150 TSU-KA-SE</td>
<td>18</td>
<td>6752</td>
<td>53</td>
<td>6 IK-E</td>
</tr>
<tr>
<td>21</td>
<td>5135</td>
<td>0</td>
<td>67  NO-KO-RI</td>
<td>19</td>
<td>6752</td>
<td>9</td>
<td>36 RA-YA-MA</td>
</tr>
<tr>
<td>22</td>
<td>5135</td>
<td>117</td>
<td>722  HA-RU</td>
<td>20</td>
<td>6753</td>
<td>2</td>
<td>22 RA-SO-RA-YA-MA</td>
</tr>
<tr>
<td>23</td>
<td>5136</td>
<td>13545327</td>
<td>NO</td>
<td>21</td>
<td>6753</td>
<td>77</td>
<td>114 NA-MI-TA</td>
</tr>
<tr>
<td>24</td>
<td>5136</td>
<td>3</td>
<td>101  SE-MI-YO-RI-NO</td>
<td>22</td>
<td>6754</td>
<td>53</td>
<td>7 KU-YO</td>
</tr>
<tr>
<td>25</td>
<td>5136</td>
<td>2</td>
<td>90   YO-HA-NO</td>
<td>23</td>
<td>6754</td>
<td>2</td>
<td>21 WO-KU-RA-NO</td>
</tr>
<tr>
<td>26</td>
<td>5137</td>
<td>0</td>
<td>62   NO-MI-CHI</td>
<td>24</td>
<td>6754</td>
<td>141</td>
<td>173 KA-MI</td>
</tr>
<tr>
<td>27</td>
<td>5137</td>
<td>14</td>
<td>3    KA-NA-SHI-KA</td>
<td>25</td>
<td>6755</td>
<td>234</td>
<td>92 SO-TE</td>
</tr>
<tr>
<td>28</td>
<td>5137</td>
<td>1</td>
<td>75   KO-RU</td>
<td>26</td>
<td>6755</td>
<td>39</td>
<td>39 SA-KU-YO</td>
</tr>
<tr>
<td>29</td>
<td>5138</td>
<td>9</td>
<td>0    TSU-TSU-MA</td>
<td>27</td>
<td>6755</td>
<td>136</td>
<td>166 NA-RI</td>
</tr>
<tr>
<td>30</td>
<td>5138</td>
<td>55</td>
<td>79   NI-TE</td>
<td>28</td>
<td>6756</td>
<td>42</td>
<td>74 KE-KE</td>
</tr>
<tr>
<td>31</td>
<td>5138</td>
<td>6</td>
<td>119  MA-TSU-KA-SE</td>
<td>29</td>
<td>6756</td>
<td>54</td>
<td>8 HO-HI</td>
</tr>
<tr>
<td>32</td>
<td>5138</td>
<td>1</td>
<td>73   SU-MI-NO</td>
<td>30</td>
<td>6756</td>
<td>2</td>
<td>20 NO-NE-SA-ME</td>
</tr>
<tr>
<td>33</td>
<td>5138</td>
<td>4</td>
<td>102  KA-NA-SHI-SE</td>
<td>31</td>
<td>6756</td>
<td>91</td>
<td>121 NA-KI</td>
</tr>
<tr>
<td>34</td>
<td>5139</td>
<td>37</td>
<td>306  YA-MA-NO</td>
<td>32</td>
<td>6757</td>
<td>47</td>
<td>6 NI-HO-HI</td>
</tr>
<tr>
<td>35</td>
<td>5139</td>
<td>152</td>
<td>848  KA-SE</td>
<td>33</td>
<td>6757</td>
<td>0</td>
<td>13 WO-KU-RA-YA-NO-MA-NO</td>
</tr>
<tr>
<td>36</td>
<td>5139</td>
<td>2</td>
<td>81   HO-TO-KE</td>
<td>34</td>
<td>6758</td>
<td>4</td>
<td>23 RA-NO-YA</td>
</tr>
<tr>
<td>37</td>
<td>5139</td>
<td>129</td>
<td>744  A-KI</td>
<td>35</td>
<td>6758</td>
<td>208</td>
<td>226 NA-SHI</td>
</tr>
<tr>
<td>38</td>
<td>5139</td>
<td>330</td>
<td>912  TSU-KI</td>
<td>36</td>
<td>6758</td>
<td>7</td>
<td>286 NE-SA-ME</td>
</tr>
<tr>
<td>39</td>
<td>5139</td>
<td>23</td>
<td>223  NA-HO</td>
<td>37</td>
<td>6758</td>
<td>1</td>
<td>16 NA-MI-TA-NA</td>
</tr>
<tr>
<td>40</td>
<td>5139</td>
<td>21</td>
<td>211  NO-RI</td>
<td>38</td>
<td>6758</td>
<td>1</td>
<td>16 WO-KU-RA-NO-YA-MA</td>
</tr>
<tr>
<td>41</td>
<td>5139</td>
<td>19</td>
<td>11   MI-KÉ</td>
<td>39</td>
<td>6758</td>
<td>133</td>
<td>44 TE-NO</td>
</tr>
<tr>
<td>42</td>
<td>5139</td>
<td>74</td>
<td>131  TE-NI</td>
<td>40</td>
<td>6759</td>
<td>1350</td>
<td>1089 SA</td>
</tr>
</tbody>
</table>

6 Concluding Remarks

We have reported successful results for Waka poems, but not for prose texts. We considered all prime substrings as candidates for characteristic expressions. However it seems that some filtering process is needed for prose texts. In [13], we successfully discovered from Waka poems characteristic patterns, called Fushi, which are regular patterns whose constant parts are restricted to sequences of auxiliary verbs and postpositional particles. To find a good filtering for prose texts will be our future work.
Fig. 6. Left and right context trees of “NO-MI-CHI.” In both trees the top string is “NO-MI-CHI.” In the left context tree (the left one), the 9th and 12th strings from the top are “NO-RI-NO-MI-CHI” and “MA-KO-TO-NO-MI-CHI,” respectively.

References

Classifying Scenarios using Belief Decision Trees

Zied Elouedi¹ and Khaled Mellouli¹

Institut Supérieur de Gestion de Tunis, 41 Avenue de la liberté, Cité Bouchnouche,  
2000 Le Bardo, Tunis, Tunisie  
zied.elouedi@isg.rnu.tn, khaled.mellouli@ihec.rnu.tn

Abstract. In this paper, we propose a method based on the belief decision tree approach, to classify scenarios in an uncertain context. Our method uses both the decision tree technique and the belief function theory as understood in the transferable belief model in order to find the classes of the scenarios (of a given problem) that may happen in the future.

Two major phases will be ensured: the construction of the belief decision tree representing the scenarios belonging to the training set and which may present some uncertainty in their class membership, this uncertainty is presented by belief functions. Then, the classification of new scenarios characterized generally by uncertain hypotheses’ configurations.

1 Introduction

A scenario is defined as a set of elementary hypotheses aiming at analyzing future events and anticipating what may happen. The task of classifying scenarios is one of the major preoccupations facing decision makers since its capability to assign similar scenarios to the same class, this will help finding the best strategic planning regarding a given problem.

Due to the uncertainty that may occur either in the configurations of scenario’s hypotheses or even in the classes of the training scenarios, the classification task becomes more and more difficult. Ignoring this uncertainty or mistreating it may lead to erroneous results.

In this paper, we propose a method based on belief decision trees in order to ensure the classification of scenarios in an uncertain context.

The belief decision tree is a classification technique that we have developed [4], [5], it is an extension of the standard decision tree approach, based on the belief function theory in order to cope with the uncertainty related to the parameters of any classification problem.

The belief decision tree approach offers a suitable framework to deal with the classification of scenarios in an uncertain environment. The use of the belief function theory as understood in the transferable belief model, allows a better representation of uncertainty characterizing the scenarios, especially the uncertainty expressed by experts.

This paper is composed as follows: We start by presenting the notion of scenarios and their objectives, then we give an overview of what we call a belief
decision tree where we introduce at first the basics of the belief function theory and those related to the decision tree technique, next we describe the construction and the classification procedures related to the belief decision tree approach. Finally, we detail our method for classifying scenarios using belief decision trees. An example illustrating our proposed method will be presented.

2 Scenarios

A scenario describes a future situation. It allows to elicit and to anticipate about what may happen. Indeed, it is composed of a set of hypotheses related to the components of the given problem (field). Each hypothesis could take one or several configurations. Hence, a scenario is considered as a combination of these configurations.

These hypotheses and their configurations are defined through interviews and different questions given to experts and actors implicated in the given problem. To ensure this objective, several methods are proposed in the literature, the most used is the Delphi technique [6].

Two types of scenarios are defined:

- The exploratory scenarios based on past and present trends in order to elicit future. They are equivalent to the classical forecasting.
- The anticipatory scenarios built on the basis of different visions of future desired or redoubted. In fact, we have to fix the future objectives and try to find how to ensure them.

The scenario method plays an important role especially in decision problems given its capability to deal with various fields and consequently help decision makers to find the appropriate strategic planning.

The scenario method is mainly based on experts’ opinions [3], [6], [7], [9]. It includes different steps. The major ones dealing with scenarios are their assessment and their classification.

The latter step related to the classification of scenarios allows to group scenarios sharing similar characteristics in the same class. By taking into account the class of the scenario, more reliable decisions may be taken.

3 Belief Decision Trees

In this section, we briefly review the basics of the belief function theory as understood in the Transferable Belief Model (TBM) [15], [16], [17] and those of the decision tree technique [10], [11], [12]. Then, we describe our belief decision tree approach based on both the belief function theory and the decision tree method.
3.1 Belief Function Theory

Definitions. Let $\Theta$ be a set of mutually exclusive and exhaustive events referred to as the frame of discernment. A basic belief assignment (bba) (called initially basic probability assignment [13]) over $\Theta$ is a function $m$ defined as follows:

$$m: 2^\Theta \rightarrow [0, 1] \text{ such that:}$$

$$\sum_{A \subseteq \Theta} m(A) = 1$$

For each subset $A$ belonging to the frame $\Theta$, $m(A)$ measures the part of belief that is committed exactly to $A$ and which can not be apportioned to any strict subset of $A$.

We call a subset $A \subseteq \Theta$ such that $m(A) > 0$, a focal element. A basic belief assignment having only $\Theta$ as a focal element is called a vacuous belief function having the characteristic to represent the total ignorance [13].

The belief function $bel$, corresponding to a basic belief assignment $m$, represents the total belief committed to each subset $A$ of the frame of discernment $\Theta$: $bel(A) = \sum_{B \subset A} m(B)$ and $bel(\emptyset) = 0$.

Note that assessments of the bba are explained in [15] and [18].

Combination. Consider two distinct pieces of evidence on the same frame $\Theta$ represented respectively by two bba’s $m_1$ and $m_2$. Two kinds of combination at least may be defined [15]:

- The Conjunctive Rule: providing a new bba that represents the combined impact of the two pieces of evidence. So we get:

$$m_1 \land m_2(A) = \sum_{B,C \subseteq \Theta, B \cap C = A} m_1(B).m_2(C)$$

The conjunctive rule can be seen as an unnormalized Dempster’s rule of combination. The Dempster’s rule is defined as [13]:

$$(m_1 \oplus m_2)(A) = K \sum_{B,C \subseteq \Theta, B \cap C = A} m_1(B).m_2(C)$$

where

$$K^{-1} = 1 - \sum_{B,C \subseteq \Theta, B \cap C = \emptyset} m_1(B).m_2(C)$$

and

$$(m_1 \oplus m_2)(\emptyset) = 0$$

$K$ is called the normalization factor.

- The Disjunctive Rule: inducing a bba that expresses the case when we only know that at least one of the two pieces of evidence actually holds but we do not which one. So we get:

$$m_1 \lor m_2(A) = \sum_{B,C \subseteq \Theta, B \cup C = A} m_1(B).m_2(C)$$

Note that since the conjunctive and the disjunctive rules are both commutative and associative, so combining several pieces of evidence induced from distinct information sources (either conjunctively or disjunctively) may be easily ensured by applying repeatedly the chosen rule.
Vacuous Extension of Belief Functions. Let \( X \) and \( Y \) be two sets of variables such that \( Y \subseteq X \). Let \( m^Y \) be a bba defined on the domain of \( Y \), \( \Theta_Y \) which is the cross product of the different variables of \( Y \). The extension of \( m^Y \) to \( \Theta_X \), denoted \( m^{Y_X} \), means that the information in \( m^Y \) is extended to a larger frame \( X \) [8]:

\[
m^{Y_X} (A \times \Theta_{X-Y}) = m^Y (A) \text{ for } A \subseteq \Theta_Y
\]

\[
m^{Y_X} (B) = 0 \text{ if } B \text{ is not in the form } A \times \Theta_{X-Y}
\]

Decision Process. The Transferable Belief Model (TBM) developed by Smets presents a solution to make decisions within the belief function framework. In fact, the TBM is based on two levels:

- a credal level where beliefs are entertained and quantified by using belief functions.
- a pignistic level where beliefs are used to make decisions and where they are represented by probability functions called the pignistic probabilities.

When a decision is needed, we use the pignistic transformation which builds a pignistic probability function \( \text{Bet} P \) from the initial bba \( m \) (handled in the credal level) as follows [16]:

\[
\text{Bet} P(\theta) = \sum_{A \subseteq \Theta, \theta \in A} \frac{m(A)}{m(A) + (1 - m(A))}, \text{ for all } \theta \in \Theta
\]

Note that \( m(\emptyset) \) is interpreted as the part of belief given to the fact that none of the hypotheses in \( \Theta \) is true or as the amount of conflict between the pieces of evidence. In the case of a normalized context the value of \( m(\emptyset) \) is equal to zero.

3.2 Decision Trees

Several learning methods have been developed for ensuring classification. Among these techniques, the decision tree may be one of the most commonly used.

Decision trees are characterized by their capability to break down a complex decision problem into several simpler decisions. They represent a sequential procedure for deciding the membership class of a given instance. Their major advantage resides on providing a powerful formalism for expressing classification knowledge [10] and providing comprehensible classifiers.

The decision tree technique is composed of two major procedures:

1. The first for building the tree: Based on a given training set, a decision tree can be built. It consists in finding for each decision node the "appropriate" test attribute by using an attribute selection measure and also to define the class labeling each leaf. As a result, we get a decision tree where decision nodes represent attributes, branches correspond to the possible attribute values and leaves including sets of instances belonging to the same class.
2. The second for the classification: Once the tree is constructed and in order to classify a new instance, we start by the root of the decision tree, then we test the attribute specified by this node. According to the result of the test, we move down the tree branch relative to the attribute value of the given instance. This process will be repeated until a leaf is encountered. This leaf is labeled by a class.

3.3 Belief Decision Tree

Due to the uncertainty that may occur in the parameters of any classification problem, we have developed what we call a belief decision tree [4], [5]. Such approach presents a classification method in an uncertain context based on both the decision tree technique and the belief function theory as explained in the TBM.

The use of the belief function theory in decision trees provides a suitable framework thanks to its ability to treat subjective, personal judgments on the different parameters (attributes, classes) of any classification problem. It permits to represent beliefs not only on elementary hypotheses but also for a collection (disjunction) of hypotheses.

Besides, this theory allows experts to express partial beliefs in a much more flexible way than probability functions do. It also permits to handle partial or even total ignorance concerning classification parameters.

Furthermore, it offers appropriate tools to combine several pieces of evidence like the conjunctive and the disjunctive rules. Decision making is ensured by applying the pigistic transformation.

In addition to these advantages, it is easy applied in reasoning based systems like expert systems, decision support systems...

Construction Procedure. As described for a standard decision tree, the construction of a belief decision tree is mainly based on a training set of instances. Since we deal with uncertainty, the structure of this set will change from the traditional one.

In fact, we assume that the uncertainty will occur only in the classes of training instances. Such uncertainty is generally due to lack of information.

Since we use the belief function theory, for each training object's class, we define a basic belief assignment showing beliefs given by experts on the different classes to which this object may belong.

Due to the uncertainty in the training set, the leaf is not attached to a unique class but it would be labeled by a bba. This bba represents the beliefs on the possible classes associated with the path from the root to this leaf.

Once the structure of the training set is defined, we present our algorithm of constructing a belief decision tree which is an extension of the ID3 algorithm [10] within the belief function framework.

Let T be the training set, A be the set of attributes and \( m^\theta \{ I_x \} \) be the bba defined on \( \theta \) the set of the n possible classes representing beliefs given by the experts on the actual class of the training instance \( I_x \).
Our algorithm is described as follows:

1. Generate the root node containing all the training instances in T.
2. If the treated node satisfies one of the following conditions (known as a stopping criterion):
   - It contains only one object.
   - There is no attribute to test.
   - The information gain (defined below) of the remaining attributes are less or equal to zero.

   Then, the node is declared as a leaf where its bba $m_L$ is equal to the average bba of the objects belonging to this leaf:

   \[
   m_L = \frac{\sum_{L \in L} m^\Theta \{L\}}{|L|}
   \]  

   where $|L|$ represents the number of instances belonging to the leaf L.

3. If the node does not satisfy the stopping criterion, for each attribute $A_i \in A$, compute the information gain \( \text{Gain}(T, A_i) \), then the attribute presenting the maximum information gain will be selected as a test attribute and will be considered as the root of the current tree.

4. According to the selected attribute values, apply the partitioning strategy allowing to divide the training set T into training subsets $T_1, T_2, \ldots$. Each subset involves the objects having one value of the selected attribute.

5. Repeat the same process for each training subset while verifying the stopping criterion for the remaining attributes.

6. Stop when all the nodes of the latter level of the tree are leaves.

   The information gain \( \text{Gain}(T, A_i) \) of the attribute $A_i$ has to take into account the uncertainty lying on the classes. It is defined as follows:

   \[
   \text{Gain}(T, A_i) = \text{Inf}_o(T) - \text{Inf}_{o_{A_i}}(T)
   \]  

   where

   \[
   \text{Inf}_o(T) = -\sum_{i=1}^{n} \text{Bet} P^\Theta \{T\}(C_i) \log_2 \text{Bet} P^\Theta \{T\}(C_i)
   \]  

   and

   \[
   \text{Inf}_{o_{A_i}}(T) = -\sum_{v \in \text{values}(A_i)} \frac{|T_v^{A_i}|}{|T|} \text{Inf}_o(T_v^{A_i})
   \]  

   where $\text{Bet} P^\Theta \{T\}$ is the average pignistic probability function taken over the training set T and defined by:

   \[
   \text{Bet} P^\Theta \{T\}(C) = \frac{\sum_{L \in T} \text{Bet} P^\Theta \{L\}(C)}{|T|}
   \]  

   and $T_v^{A_i}$ is the training subset when the value of the attribute $A_i$ is equal to $v$. 


We have to mention that the attribute selection measure used in our algorithm represents the extension of the information gain of Quinlan to an uncertain context by using the basics of the belief function theory. Our attribute selection measure allows to handle uncertainty, represented by basic belief assignment functions, in classes of training instances. As explained handling beliefs instead of probabilities in the training instances is more flexible and presents a more generalized context for dealing with uncertainty.

**Classification Procedure.** The classification procedure is ensured by taking into account the induced belief decision tree.

As we deal with an uncertain environment, the new instances to classify may be characterized by missing or uncertain attribute values.

In fact, this uncertainty can be represented by a basic belief assignment on the set of all the possible values of each attribute. As known the use of bba's includes also particular cases (instances with some certain attribute values, the ones with disjunctive values in some of their attributes and the ones with missing values in some of their attributes).

Let $m^{A_i}$ be the bba representing the part of beliefs committed exactly to the different values relative to the attribute $A_i$ of the new instance to classify. This bba is defined on the frame of discernment $\Theta_{A_i}$ including all the possible values of the attribute $A_i$.

Let $\Theta_A$ be the global frame of discernment relative to all the attributes. It is equal to the cross product of the different $\Theta_{A_i}$. We denote by $\Theta_A = \times_{i=1,k} \Theta_{A_i}$.

Since a given instance is described by a set of combination of values where each one is relative to an attribute, we have to find the bba expressing beliefs on the different attributes’ values of the new instance to classify. In other words, we have to look for the joint bba $m^{\Theta_A}$ representing beliefs on all the instance’s attributes by applying the conjunctive rule of combination:

$$m^{\Theta_A} = \wedge_{i=1,k} m^{A_i}$$

$m^{A_i}$ is the extension of the bba $m^{A_i}$ to the frame $\Theta_A$.

Then, for each focal element $x$ relative to $m^{\Theta_A}$, we have to compute the belief functions $bel^\theta[x]$ defined on the set of the possible classes $\Theta$ given $x$. This belief function is equal to the result of the disjunctive combination rule between belief function’s leaves corresponding to this focal element $x$.

Once computed, these belief functions are averaged using $m^{\Theta_A}$ such that:

$$bel^\theta[m^{\Theta_A}](\theta) = \frac{1}{\sum_{x \in \Theta_A} m^{\Theta_A}(x)}.bel^\theta[x](\theta) \text{ for } \theta \in \Theta$$

Hence, $bel^\theta[m^{\Theta_A}]$ represents total beliefs of this new instance to belong to the different classes related to the problem. To know the part of belief exactly committed to the different classes of $\Theta$, we may easily induce $m^\theta[m^{\Theta_A}]$.

Finally, we apply the pignistic transformation to this bba in order to get the probability of this instance to belong to each singular class.
4 Classifying Scenarios using Belief Decision Trees

In this section, we present the different steps leading to the classification of scenarios using a belief decision tree. Each step will be illustrated by an example explaining its real unfolding.

4.1 Scenarios vs Training Set

A scenario is seen as a combination of hypotheses’ configurations where each one is relative to a hypothesis $H_i$.

To represent scenarios within a belief decision tree, we consider the hypotheses as being the attributes, whereas the configurations corresponding to each hypothesis are assimilated to the attribute values.

Hence, we get a training set of scenarios characterized by certain configurations’ hypotheses, however there may be some uncertainty in their classes defined for each one by a bba on the set of classes. These bba’s are given by experts.

Example 1 Let’s consider a simple example presenting scenarios regarding the agriculture field. For simplicity sake, we define only three elementary hypotheses composing these scenarios:

- $H_1$: the rainfall which can be high or weak.
- $H_2$: the temperature which can be hot, mild, cold.
- $H_3$: the wind with values strong or weak.

A possible scenario is for example having a high rainfall, with hot temperature and weak wind. In fact, there are twelve possible scenarios.

There are three classes to which the scenarios, related to this problem, may belong:

- $C_1$: regrouping the favorable scenarios for the agriculture field.
- $C_2$: regrouping the neutral scenarios for the agriculture field.
- $C_3$: regrouping the disastrous scenarios for the agriculture field.

We have an expert’s beliefs about the classes of scenarios that have occurred over the last six years, we get the following results:

<table>
<thead>
<tr>
<th>$S_i$</th>
<th>Rainfall</th>
<th>Temperature</th>
<th>Wind</th>
<th>bba’s</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>High</td>
<td>Mild</td>
<td>Weak</td>
<td>$m^w{S_1}$</td>
</tr>
<tr>
<td>$S_2$</td>
<td>High</td>
<td>Cold</td>
<td>Strong</td>
<td>$m^e{S_2}$</td>
</tr>
<tr>
<td>$S_3$</td>
<td>High</td>
<td>Hot</td>
<td>Weak</td>
<td>$m^w{S_3}$</td>
</tr>
<tr>
<td>$S_4$</td>
<td>Weak</td>
<td>Hot</td>
<td>Weak</td>
<td>$m^w{S_4}$</td>
</tr>
<tr>
<td>$S_5$</td>
<td>High</td>
<td>Mild</td>
<td>Weak</td>
<td>$m^w{S_5}$</td>
</tr>
<tr>
<td>$S_6$</td>
<td>Weak</td>
<td>Hot</td>
<td>Strong</td>
<td>$m^e{S_6}$</td>
</tr>
</tbody>
</table>
where
\[ m^\Theta(S_1)(C_1) = 0.8; \quad m^\Theta(S_1)(\Theta) = 0.2; \]
\[ m^\Theta(S_2)(C_1 \cup C_2) = 1; \]
\[ m^\Theta(S_3)(C_2) = 0.2; \quad m^\Theta(S_3)(C_3) = 0.4; \quad m^\Theta(S_3)(C_1 \cup C_2) = 0.2; \quad m^\Theta(S_3)(\Theta) = 0.2; \]
\[ m^\Theta(S_4)(C_3) = 0.6; \quad m^\Theta(S_4)(C_2 \cup C_3) = 0.2; \quad m^\Theta(S_4)(\Theta) = 0.2; \]
\[ m^\Theta(S_5)(C_1) = 0.9; \quad m^\Theta(S_5)(\Theta) = 0.1; \]
\[ m^\Theta(S_6)(C_3) = 1; \]

As we noted there is some uncertainty concerning the classes of these training scenarios.

For example, 0.8 is the part of belief committed to the scenario \( S_1 \) to belong to the class of scenarios favorable for the agriculture, where 0.2 is the part of belief committed to the whole frame \( \Theta \).

However, for the scenario \( S_6 \), the expert is sure that \( S_6 \) is disastrous, whereas for the scenario \( S_2 \), he is certain that it is either favorable or neutral for the agriculture and not disastrous.

This training set will allow us to build the corresponding belief decision tree representing a learning taking into account these six training scenarios.

### 4.2 Construction Procedure using Scenarios

In order to construct a belief decision tree based on scenarios, we have to adopt the parameters used in the construction algorithm of a belief decision tree to the case handling scenarios instead of ordinary objects.

Therefore, the idea is to build a tree taking into account the scenarios belonging to the training set characterized by uncertain classes. In fact, the belief decision tree relative to the training scenarios will be built by employing a recursive divide and conquer strategy (as described in subsection 3.3). Its steps can be summarized as follows:

- By using the information gain measure (extended to the uncertain context), a hypothesis (the one having the highest gain) will be chosen in order to partition the training set of scenarios. Therefore, the chosen hypothesis is selected as the root node of the current tree.
- Based on a partitioning strategy, the current training set will be divided into training subsets by taking into account the configurations of the selected hypothesis.
- When the stopping criterion is satisfied, the training subset will be declared as a leaf.

Once the tree is built, this allows to classify new scenarios.

**Example 2** Let’s continue with the example 1. In order to find the root of the belief decision tree relative to the training set \( T \), we have to compute the information gain of each hypothesis.

We start by computing \( \text{Info}(T) \):
\[ \text{Info}(T) = -\sum_{i=1}^{n} \text{BetP}^\theta(T)(C_i) \log_2 \text{BetP}^\theta(T)(C_i) \]

To compute the average pignistic probability \( \text{BetP}(T) \), we have at first to calculate the different \( \text{BetP}(S_i) \) where \( i \in \{1, 2, ..., 6\} \) (see table 2):

<table>
<thead>
<tr>
<th>( S_i )</th>
<th>( \text{BetP}^\theta(S_i)(C_1) )</th>
<th>( \text{BetP}^\theta(S_i)(C_2) )</th>
<th>( \text{BetP}^\theta(S_i)(C_3) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S_1 )</td>
<td>0.86</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>( S_2 )</td>
<td>0.5</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>( S_3 )</td>
<td>0.17</td>
<td>0.37</td>
<td>0.46</td>
</tr>
<tr>
<td>( S_4 )</td>
<td>0.07</td>
<td>0.17</td>
<td>0.76</td>
</tr>
<tr>
<td>( S_5 )</td>
<td>0.94</td>
<td>0.03</td>
<td>0.03</td>
</tr>
<tr>
<td>( S_6 )</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\( \text{BetP}^\theta(T) \) the average pignistic probability taking over the whole training set \( T \), is defined by applying the equation 5:

\[
\begin{align*}
\text{BetP}^\theta(T)(C_1) &= \frac{1}{6} (0.86 + 0.5 + 0.17 + 0.07 + 0.94) - 0.42; \\
\text{BetP}^\theta(T)(C_2) &= 0.19; \\
\text{BetP}^\theta(T)(C_3) &= 0.39; \\
\text{Hence Info}(T) &= 1.511;
\end{align*}
\]

Then, we have to compute the information gain relative to each hypothesis related to the agriculture filed, we get:

\[
\begin{align*}
\text{Gain}(T, \text{rainfall}) &= 1.511 - 1.056 - 0.455; \\
\text{Gain}(T, \text{temperature}) &= 1.511 - 0.885 - 0.626; \\
\text{Gain}(T, \text{wind}) &= 1.511 - 1.464 - 0.047;
\end{align*}
\]

The rainfall hypothesis presents the highest information gain. Thus, it will be chosen as the root relative to the training set \( T \).

So, we get the following decision tree (see Fig 1):

![Belief decision tree](image)

**Fig.1.** Belief decision tree (first step).

Now, we have to apply the same process on the two training subsets \( T_{\text{rainfall}} \) including the scenarios characterized by high rainfall and \( T_{\text{weak}} \) including those characterized by weak rainfall.
This process will be halt when the stopping criterion is fulfilled for all the training subsets.

For the problem related to the agriculture field, the final belief decision tree is as follows (see Fig 2):

![Belief Decision Tree](image)

Fig. 2. Belief decision tree representing training scenarios.

where $m^{O}\{S_i\}$ is the average bba of the subset including the scenarios $S_1$ and $S_2$. So, we get: $m^{O}\{S_i\}(C_1) = 0.85; m^{O}\{S_i\}(\Theta) = 0.15$;

**Classification of New Scenarios.** This phase is very important since it allows to classify scenarios characterized by uncertain hypothesis configurations. This classification will be ensured by taking into account the constructed belief decision tree.

In fact, training scenarios (and their classes) presented by the means of a belief decision tree constitutes a convenient framework to classify new scenarios. Our classification method allows to handle not only uncertain hypotheses’ configurations (described by basic belief assignments) but includes also hypotheses with certain or disjunctive or even unknown configurations.

The classification of a new scenario that may happen, provides a good capability to decision makers to fix the appropriate strategic planning according to beliefs assigned to the classes to which this scenario may belong.

**Example 3** Let’s continue with the example 2, we would like to classify the scenario $S$ lying on the agriculture field that may occur in the year 2002.

Let’s define by: $H = \{rainfall, temperature, wind\}$

\[
\Theta^{\text{rainfall}} = \{\text{high, weak}\}
\]

\[
\Theta^{\text{temperature}} = \{\text{hot, mild, cold}\}
\]

\[
\Theta^{\text{wind}} = \{\text{strong, weak}\}
\]

And $\Theta_H = \Theta^{\text{rainfall}} \times \Theta^{\text{temperature}} \times \Theta^{\text{wind}}$

The expert is not sure about the “future” configurations of some of the three elementary hypotheses related to the scenario $S$ (to classify). He presents his opinions as follows:

\[
m^{\text{rainfall}}(\{\text{high}\}) = 0.6; m^{\text{rainfall}}(\Theta^{\text{rainfall}}) = 0.4;
\]

\[
m^{\text{temperature}}(\{\text{mild}\}) = 1;
\]
\[ m_{\text{wind}}(\theta_{\text{wind}}) = 1; \]

In other words, the scenario \( S \) to classify is characterized by some uncertainty regarding the rainfall hypothesis, a mild temperature and a total ignorance concerning the wind hypothesis (presented by a vacuous bba).

So what will be the class of this "uncertain" scenario \( S \)?

We start by extending the different hypotheses’ bba’s to \( \Theta_H \), we get:

\[
\begin{align*}
    m_{\text{rain}}^{\text{all} \mid H}(\{\text{high}\} \times \Theta_{\text{temperature}} \times \Theta_{\text{wind}}) &= 0.6; \\
    m_{\text{temperature}}^{\text{all} \mid H}(\Theta_{\text{rain}} \times \Theta_{\text{temperature}} \times \Theta_{\text{wind}}) &= 0.4; \\
    m_{\text{wind}}^{\text{all} \mid H}(\Theta_{\text{rain}} \times \Theta_{\text{temperature}} \times \Theta_{\text{wind}}) &= 1; \\
    m_{\text{wind}}^{\text{all} \mid H}(\Theta_{\text{rain}} \times \Theta_{\text{temperature}} \times \Theta_{\text{wind}}) &= 1;
\end{align*}
\]

In order to get the beliefs committed to the possible scenarios that may happen in the year 2002, we have to combine these extended bba’s by using the conjunctive rule of combination. We get:

\[
\begin{align*}
    m^{\Theta_H} &= m_{\text{rain}}^{\text{all} \mid \Theta_H} \land m_{\text{temperature}}^{\text{all} \mid \Theta_H} \land m_{\text{wind}}^{\text{all} \mid \Theta_H} \text{ such that:} \\
    m^{\Theta_H} ((\text{high, mild, strong}), (\text{high, mild, weak})) &= 0.6; \\
    m^{\Theta_H} ((\text{high, mild, strong}), (\text{weak, mild, strong}), \text{weak, mild, weak}) &= 0.4;
\end{align*}
\]

We note that there are two focal elements with basic belief masses equal respectively to 0.6 and 0.4.

Then, we have to find beliefs on classes (defined on \( \Theta \)) given the configurations of the hypotheses characterizing the new scenario \( S \) to classify. These beliefs have to take into account the two focal elements. According to the belief decision tree induced in the example 2 (see Fig 2), we get:

\[
\begin{align*}
    bel^\theta([\text{high, mild, strong}), (\text{high, mild, weak})) &= bel^\theta \{ S_{15} \}; \\
    bel^\theta([\text{high, mild, strong}), (\text{high, mild, weak}), \text{weak, mild, strong}), \text{weak, mild, weak}) &= bel^\theta \{ S_{15} \} \lor bel^\theta \{ S_4 \} \lor bel^\theta \{ S_6 \}
\end{align*}
\]

Let \( bel_1 = bel^\theta \{ S_{15} \} \) and \( bel_2 = bel^\theta \{ S_{15} \} \lor bel^\theta \{ S_4 \} \lor bel^\theta \{ S_6 \} \)

The values (see table 3) of \( bel_1 \) are induced from the bba \( m^\theta \{ S_{15} \} \), whereas those of \( bel_2 \) are computed from the combination of \( bel^\theta \{ S_{15} \} \), \( bel^\theta \{ S_4 \} \) and \( bel^\theta \{ S_6 \} \) using the disjunctive rule.

<table>
<thead>
<tr>
<th>( \tau )</th>
<th>( \tau_1 )</th>
<th>( \tau_2 )</th>
<th>( \tau_3 )</th>
<th>( \tau_{12} )</th>
<th>( \tau_{13} )</th>
<th>( \tau_{23} )</th>
<th>( \tau_{123} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( bel_1 )</td>
<td>0</td>
<td>0.85</td>
<td>0</td>
<td>0.85</td>
<td>0.85</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>( bel_2 )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.51</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Hence, these two belief functions will be averaged (using the values of the bba \( m^{\Theta} \)), we get:

\[
\begin{align*}
\text{Bet}^{\Theta}[m^{\Theta n}](\emptyset) & = 0; \\
\text{Bet}^{\Theta}[m^{\Theta n}](C_1) & = 0.6 \times 0.85 + 0.4 \times 0 - 0.51; \\
\text{Bet}^{\Theta}[m^{\Theta n}](C_2) & = 0; \\
\text{Bet}^{\Theta}[m^{\Theta n}](C_3) & = 0; \\
\text{Bet}^{\Theta}[m^{\Theta n}](C_1 \cup C_2) & = 0.6 \times 0.85 - 0.51; \\
\text{Bet}^{\Theta}[m^{\Theta n}](C_1 \cup C_3) & = 0.6 \times 0.85 + 0.4 \times 0.51 - 0.71; \\
\text{Bet}^{\Theta}[m^{\Theta n}](C_2 \cup C_3) & = 0 \\
\text{Bet}^{\Theta}[m^{\Theta n}](\Theta) & = 1
\end{align*}
\]

Applying the pignistic transformation \(^1\) gives us the probability on each singular class, the pignistic probability will be defined as follows:

\[
\text{Bet}^P(C_1) = 0.71; \quad \text{Bet}^P(C_2) = 0.09; \quad \text{Bet}^P(C_3) = 0.2;
\]

Hence, the probability that the scenario S belongs respectively to the classes C_1, C_2 and C_3 are respectively 0.71, 0.09 and 0.2. So, it seems that the scenario S (that may happen in the year 2002) has more chances (0.71) to be favorable for the agriculture field.

5 Conclusion

In this paper, we have presented a method for classifying scenarios using belief decision trees. Our method has the advantage to handle the uncertainty that may characterize either the classes of the training scenarios ensuring the construction of the belief decision tree or the configurations of the hypotheses making up the scenario to classify.

The result of classification of scenarios provides a significant help to decision makers to conceive their strategic policy.

Evaluation of this belief decision tree approach and comparisons with classical classification techniques are now our major research interest.

References


\(^{1}\) From the different values of \( \text{Bet}^P[m^{\Theta n}] \), we can easily deduce the corresponding basic belief masses and then apply the pignistic transformation
A Practical Algorithm to Find the Best Subsequence Patterns

Masahiro Hirao, Hiromasa Hoshino, Ayumi Shinohara, Masayuki Takeda, and Setsuo Arikawa

Department of Informatics, Kyushu University 33, Fukuoka 812-8581, JAPAN
{hirao, hoshino, ayumi, takeda, arikawa}@i.kyushu-u.ac.jp

Abstract. Given two sets of strings, consider the problem to find a subsequence that is common to one set but never appears in the other set. The problem is known to be NP-complete. We generalize the problem to an optimization problem, and give a practical algorithm to solve it exactly. Our algorithm uses pruning heuristic and subsequence automata, and can find the best subsequence. We show some experiments, that convinced us the approach is quite promising.

1 Introduction

String is one of the most fundamental structure to express and reserve information. In these days, a lot of string data are available. String processing has vast application area, such as Genome Informatics and Internet related works. It is quite important to discover useful rules from large text data or sequential data [1, 6, 9, 22]. Finding a good rule to separate two given sets, often referred as positive examples and negative examples, is a critical task in Discovery Science as well as Machine Learning.

Shimozono et al. [20] developed a machine discovery system BONSAI that produces a decision tree over regular patterns with alphabet indexing, from given positive set and negative set of strings. The core part of the system is to generate a decision tree which classifies positive examples and negative examples as correctly as possible. For that purpose, we have to find a pattern that maximizes the goodness according to the entropy information gain measure, recursively at each node of trees. In the current implementation, a pattern associated with each node is restricted to a substring pattern, due to the limit of computation time. One of our motivations of this study is to extend the BONSAI system to allow subsequence patterns as well as substring patterns at nodes, and accelerate the computation time.

However, there is a large gap between the complexity of finding the best substring pattern and subsequence pattern. Theoretically, the former problem can be solved in linear time, while the latter is NP-hard.

In this paper, we give a practical solution to find the best subsequence pattern which separates a given set of strings from the other set of strings. We propose a practical implementation of exact search algorithm that practically
avoids exhaustive search. Since the problem is NP-hard, essentially we are forced to examine exponentially many candidate patterns in the worst case. Basically, for each pattern \( w \), we have to count the number of strings that contain \( w \) as a subsequence in each of two sets. We call the task of counting the numbers as answering subsequence query. The computational cost to find the best subsequence pattern mainly comes from the total amount of time to answer these subsequence queries, since it is relatively heavy task if the sets are large, and many queries will be needed. In order to reduce the time, we have to either (1) asking queries as few as possible, or (2) speeding up to answer queries. We attack the problem from both these two directions.

At first, we reduce the search space by appropriately pruning redundant branches that are guaranteed not to contain the best pattern. We use a heuristics inspired by Morishita and Sese [18], combined with some properties on the subsequence languages.

Next, we accelerate answering for subsequence queries. Since the sets of strings are fixed in finding the best subsequence pattern, it is reasonable to preprocess the sets so that answering subsequence query for any pattern will be fast. We take an approach based on a deterministic finite automaton that accepts all subsequences of a string. Actually, we use subsequence automata for sets of strings, developed in [11]. Subsequence automaton can answer quickly for subsequence query, at the cost of preprocessing time and space requirement to construct it.

Since these two approaches are different in their aims, we expect that a balanced integration of these two would result in the most efficient way to find the best subsequence patterns. In order to verify the performance of our algorithm, we are performing some experiments on these two approaches. We report some results of the experiments, that convinced us it is quite promising.

2 Preliminaries

Let \( \Sigma \) be a finite alphabet, and let \( \Sigma^* \) be the set of all strings over \( \Sigma \). For a string \( w \), we denote by \(|w|\) the length of \( w \), and for a set \( S \), we denote by \(|S|\) the cardinality of \( S \). We say that a string \( v \) is a prefix (substring, suffix, resp.) of \( w \) if \( w = v y \ (w = x v y, w = x v, \) resp.) for some strings \( x, y \in \Sigma^* \). We say that a string \( v \) is a subsequence of a string \( w \) if \( v \) can be obtained by removing zero or more characters from \( w \), and say that \( w \) is a supersequence of \( v \). We denote by \( v \preceq w \) that \( v \) is a substring of \( w \), and by \( v \preceq_{\text{seq}} w \) that \( v \) is a subsequence of \( w \). For a string \( v \), we define the substring language \( L^{\text{str}}(v) \) and subsequence language \( L^{\text{seq}}(v) \) as follows:

\[
L^{\text{str}}(v) = \{ w \in \Sigma^* \mid v \preceq w \}, \text{ and } \\
L^{\text{seq}}(v) = \{ w \in \Sigma^* \mid v \preceq_{\text{seq}} w \}, \text{ respectively.}
\]

The following lemma is obvious from the definitions.

**Lemma 1.** For any strings \( v, w \in \Sigma^* \),
1. if $v$ is a prefix of $w$, then $v \preceq_v w$,
2. if $v$ is a suffix of $w$, then $v \preceq_v w$,
3. if $v \preceq_v w$ then $v \preceq_{eq} w$,
4. $v \preceq_v w$ if and only if $L^v(w) \supseteq L^w(v)$,
5. $v \preceq_{eq} w$ if and only if $L^v(w) \supseteq L^w(v)$.

3 Formulation of the Problem

Let $good$ be a function from $\Sigma^* \times 2^\Sigma^* \times 2^\Sigma^*$ to the set of real numbers. We formulate the problem to be solved as follows.

Definition 1 (Finding the best pattern according to $good$).

Input Two sets $S, T \subseteq \Sigma^*$ of strings.
Output A string $w \in \Sigma^*$ that maximizes the value $good(w, S, T)$.

Intuitively, the value $good(w, S, T)$ expresses the goodness to distinguish $S$ from $T$ using the rule specified by a string $w$. The definition of $good$ varies for each application. For examples, the $\chi^2$ values, entropy information gain, and gini index are frequently used (See [18]). Essentially, these statistical measures are defined by the numbers of strings that satisfy the rule specified by $w$. In this paper, we only consider the rules defined as substring languages and subsequence languages. We call these problems as finding the best substring pattern, and finding the best subsequence pattern, respectively. Let $L$ be either $L^v$ or $L^{eq}$.

Then any of the above examples of the measures can be described in the following form.

$$good(w, S, T) = f(x_w, y_w, |S|, |T|),$$

where

$$x_w = |S \cap L(w)|,$$
$$y_w = |T \cap L(w)|.$$

For example, the entropy information gain, which is introduced by Quinlan [19] and also used in BONSAI system [20], can be defined in terms of the function $f$ as follows:

$$f(x, y, x_{max}, y_{max}) = -\frac{x + y}{x_{max} + y_{max}} I(x, y) - \frac{x_{max} - x + y_{max} - y}{x_{max} + y_{max}} I(x_{max} - x, y_{max} - y),$$

where $I(s, t) = \begin{cases} 0 & \text{if } s = 0 \text{ or } t = 0, \\ -\frac{s}{\log 2} - \frac{t}{\log 2} & \text{otherwise}. \end{cases}$

When the sets $S$ and $T$ are fixed, the values $x_{max} = |S|$ and $y_{max} = |T|$ become constants. Thus, we abbreviate the function $f(x, y, x_{max}, y_{max})$ to $f(x, y)$ in the sequel.

Since the function $good(w, S, T)$ expresses the goodness of a string $w$ to distinguish two sets, it is natural to assume that the function $f$ satisfies the conicality, defined as follows.
Definition 2. We say that a function $f(x, y)$ is conic if

- for any $0 \leq y \leq y_{\text{max}}$, there exists an $x_1$ such that
  $\bullet$ $f(x, y) \geq f(x', y)$ for any $0 \leq x < x' \leq x_1$, and
  $\bullet$ $f(x, y) \leq f(x', y)$ for any $x_1 \leq x < x' \leq x_{\text{max}}$.

- for any $0 \leq x \leq x_{\text{max}}$, there exists a $y_1$ such that
  $\bullet$ $f(x, y) \geq f(x, y')$ for any $0 \leq y < y' \leq y_1$, and
  $\bullet$ $f(x, y) \leq f(x, y')$ for any $y_1 \leq y < y' \leq y_{\text{max}}$.

Actually, all of the above statistical measures are conic. We remark that any convex function is conic.

Lemma 2. Let $f(x, y)$ be a conic function defined over $[0, x_{\text{max}}] \times [0, y_{\text{max}}]$. For any $0 \leq x < x' \leq x_{\text{max}}$ and $0 \leq y < y' \leq y_{\text{max}}$, we have

$$f(x, y) \leq \max\{f(x', y'), f(x', 0), f(0, y'), f(0, 0)\}, \text{ and}$$

$$f(x', y') \leq \max\{f(x, y), f(x, y_{\text{max}}), f(x_{\text{max}}, y), f(x_{\text{max}}, y_{\text{max}})\}.$$

Proof. We show the first inequality only. The second can be proved in the same way. Since $f$ is conic, we have $f(x, y) \leq \max\{f(x', y'), f(x', 0), f(0, y'), f(0, 0)\}$. Moreover, we have $f(x, 0) \leq \max\{f(0, 0), f(x', 0)\}$ and $f(x, y') \leq \max\{f(0, y'), f(x', y')\}$. Thus the inequality holds. \qed

In the rest of the paper, we assume that any function $f$ associated with the objective function $\text{good}$ is conic, and can be evaluated in constant time.

Now we consider the complexity of finding the best substring pattern and subsequence pattern, respectively. It is not hard to show that finding the best substring pattern can be solved in polynomial time, since there are only $O(N^2)$ substrings from given sets of strings, where $N$ is the total length of the strings, so that we can check all candidates in a trivial way. Moreover, we can solve it in linear time, by using generalized suffix trees [12].

Theorem 1. We can find the best substring pattern in linear time.

On the other hand, it is not easy to find the best subsequence pattern. First we introduce a very closely related problem.

Definition 3 (Consistency problem for subsequence patterns).
Input: Two sets $S, T \subseteq \Sigma^*$ of strings.
Question: Is there a string $w$ that is a subsequence for each string $s \in S$, but not a subsequence for any string $t \in T$?

The problem can be interpreted as a special case of the finding the best subsequence pattern. The next theorem shows the problem is intractable.

Theorem 2 ([13, 16, 17]). The consistency problem for subsequence patterns is NP-complete.

Therefore, we are essentially forced to enumerate and evaluate exponentially many subsequence patterns in the worst case, in order to find the best subsequence pattern. In the next section, we show a practical solution based on pruning search trees. Our pruning strategy utilizes the property of subsequence languages and the conicality of the function.
4 Pruning Heuristics

In this section, we introduce two pruning heuristics, inspired by Morishita and Sese [18], to construct a practical algorithm to find the best subsequence pattern.

For a conic function \( f(x, y) \), we define

\[
F(x, y) = \max \{ f(x, y), f(x, 0), f(0, y), f(0, 0) \}, \quad \text{and} \\
G(x, y) = \max \{ f(x, y), f(x, y_{\max}), f(x_{\max}, y), f(x_{\max}, y_{\max}) \}.
\]

**Theorem 3.** For any strings \( v, w \in \Sigma^* \) with \( v \preceq_{\text{seq}} w \),

\[
\begin{align*}
f(x_w, y_w) &\leq F(x_v, y_v), \\
f(x_v, y_v) &\leq G(x_w, y_w).
\end{align*}
\]

**Proof.** By Lemma 1 (5), \( v \preceq_{\text{seq}} w \) implies that \( L_{\text{seq}}(v) \supseteq L_{\text{seq}}(w) \). Thus \( x_v = |S \cap L_{\text{seq}}(v)| \geq |S \cap L_{\text{seq}}(w)| = x_w \). In the same way, we can show \( y_v \geq y_w \). By Lemma 2, we have \( f(x_w, y_w) \leq F(x_v, y_v) \). The second inequality can be verified similarly. \( \square \)

In Fig. 1, we show our algorithm to find the best subsequence pattern from given two sets of strings, according to the function \( f \). Optionally, we can specify the maximum length of subsequences. We use the following data structures in the algorithm.

**StringSet** Maintain a set \( S \) of strings.
- **void append(string w)** : append a string \( w \) into the set \( S \).
- **int numOfSubseq(string seq)** : return the cardinality of the set \( \{ w \in S \mid \text{seq} \preceq_{\text{seq}} w \} \).
- **int numOfSuperseq(string seq)** : return the cardinality of the set \( \{ w \in S \mid w \preceq_{\text{seq}} \text{seq} \} \).

**PriorityQueue** Maintain strings with their priorities.
- **bool empty()** : return **true** if the queue is empty.
- **void push(string w, double priority)** : push a string \( w \) into the queue with priority \( \text{priority} \).
- **(string, double) pop()** : pop and return a pair \((\text{string}, \text{priority})\), where \( \text{priority} \) is the highest in the queue.

The next theorem guarantees the completeness of the algorithm.

**Theorem 4.** Let \( S \) and \( T \) be sets of strings, and \( \ell \) be a positive integer. The algorithm \( \text{FindMaxSubsequence}(S, T, \ell) \) will return a string \( w \) that maximizes the value \( \text{good}(w, S, T) \) among the strings of length at most \( \ell \).

**Proof.** First of all, we consider the behavior of the algorithm whose lines marked by ‘\*’ are commented out. That is, we first assume that the lines 10, 13 and 20–23 are skipped. In this case, we show that the algorithm performs the exhaustive
string FindMaxSubsequence(StringSet S, T, int maxLength = ∞)
string prefix, seq, maxSeq;
double upperBound = ∞, maxVal = −∞, val;
int x, y;
StringSet Forbidden = ∅;
PriorityQueue queue; /* Best First Search*/
qu.provide("", ∞);
while not queue.empty() do
  (prefix, upperBound) = queue.pop();
  if upperBound < maxVal then break;
  foreach c ∈ Σ do
    seq = prefix + c; /* string concatenation */
    if Forbidden.numOfSuperseq(seq) == 0 then
      x = S.numOfSubseq(seq);
      y = T.numOfSubseq(seq);
      val = f(x, y);
      if val > maxVal then
        maxVal = val;
        maxSeq = seq;
      upperBound = max{f(x, y), f(x, 0), f(0, y), f(0, 0)};
      if upperBound < maxVal then
        Forbidden.append(seq);
      else
        if |seq| < maxLength then
          queue.push(seq, upperBound);
  return maxSeq;

Fig. 1. Algorithm FindMaxSubsequence. In our pseudocode, indentation indicates block structure, and the break statement is to jump out of the closest enclosing loop.

search in a breadth first manner. Since the value of upperBound is unchanged, PriorityQueue is actually equivalent to a simple queue. The lines 14–16 evaluate the value good(seq, S, T) of a string seq, and if it exceeds the current maximum value maxVal, we update maxVal and maxSeq in lines 17–19. Thus the algorithm will examine all strings of length at most ℓ, in increasing order of the length, and it can find the maximum.

We now consider the lines 20, 21, and 23. Let v be the string currently represented by the variable seq. At lines 14 and 15, x_v and y_v are computed. At line 20, upperBound = F(x_v, y_v) is estimated and if upperBound is less than the current maximum value maxVal, the algorithm skips pushing v into the queue. It means that any string w of which v is a prefix will not evaluated. We can show that such a string w can never be the best subsequence as follows. Since v is a prefix of w, we know v is a subsequence of w, by Lemma 1 (1) and (3). By
Theorem 3 (1), the value \( f(x_w, y_w) \leq F(x_v, y_v) \), and since \( F(x_v, y_v) < \maxVal \), the string \( w \) can never be the maximum.

Assume the condition \( upperBound < \maxVal \) holds at line 10. It implies that any string \( v \) in the queue can never be the best subsequence, since the queue is a priority queue so that \( F(x_v, y_v) \leq upperBound \), which means \( f(x_v, y_v) \leq F(x_v, y_v) \) by Theorem 3 (1). Therefore \( f(x_v, y_v) < \maxVal \) for any string \( v \) in the queue, and we can jump out of the loop immediately.

Finally, we take account of lines 13 and 22. Initially, the set \( Forbidden \) of strings is empty. At line 22, a string \( v \) is appended to \( Forbidden \) only if \( upperBound = F(x_v, y_v) < \maxVal \). At line 13, if the condition

\[
\text{Forbidden.numOfSuperseq}(seq) == 0
\]

does not hold, \( seq \) will not be evaluated. Moreover, any string of which \( seq \) is a prefix will not be evaluated either, since we do not push \( seq \) in the queue at line 25 in this case. Nevertheless, we can show that these cuts never affect the final output as follows. Assume that \( \text{Forbidden.numOfSuperseq}(seq) \neq 0 \) for a string \( seq \). It implies that there exists a string \( u \in \text{Forbidden} \) such that \( seq \) is a subsequence of \( u \). In another word, \( u \) is a subsequence of \( seq \). Since \( u \) is in \( \text{Forbidden} \), we know that \( F(x_u, y_u) < \maxVal \) at some moment. By Theorem 3 (2), the value \( f(x_u, y_u) \) can never exceeds \( \maxVal \). Thus the output of the algorithm is not changed by these cuts.

By the above theorem, we can safely prune the branches. We now consider the cost of performing these heuristics. The cost of the first heuristics at lines 20, 21, and 23 is negligible, since evaluating the \( upperBound \) at line 20 is negligible compared to evaluate \( x \) and \( y \) at lines 14 and 15. On the other hand, the second heuristics at lines 13 and 22 may be expensive, since the evaluation of \( \text{Forbidden.numOfSuperseq}(seq) \) may not be so easy when the set \( \text{Forbidden} \) becomes large.

Anyway, one of the most time-consuming part of the algorithm is the lines 14 and 15. Here, for a string \( seq \), we have to count the number of strings in the sets \( S \) and \( T \) that are subsequences of \( seq \). We remark that the set \( S \) and \( T \) are fixed within the algorithm \( \text{FindMaxSubsequence} \). Thus we have a possibility to speed up counting, at the cost of some appropriate preprocessing. We will discuss it in the next section.

5 Using Subsequence Automata

In this section, we pay our attention to the following problem.

**Definition 4 (Counting the matched strings).**

**Input** A finite set \( S \subseteq \Sigma^* \) of strings.

**Query** A string \( seq \in \Sigma^* \).

**Answer** The cardinality of the set \( S \cap L^\omega(seq) \).

Of course, the answer to the query should be very fast, since many queries will arise. Thus, we should preprocess the input in order to answer the query
quickly. On the other hand, the preprocessing time is also a critical factor in our application. In this paper, we utilize automata that accept subsequences of strings. Baeza-Yates [5] introduced the directed acyclic subsequence graph (DASG) of a string \( t \) as the smallest deterministic partial finite automaton that recognizes all possible subsequences of \( t \). By using DASG of \( t \), we can determine whether a string \( s \) is a subsequence of a string \( t \) in \( O(|s|) \) time. He showed a right-to-left algorithm for building the DASG for a single string. On the other hand, Troníček and Melichar [21] showed a left-to-right algorithm for building the DASG for a single string.

We now turn our attention to the case of a set \( S \) of strings. A straightforward approach is to build DASGs for each string in \( S \). Given a query string \( seq \), we traverse all DASGs simultaneously, and return the total number of DASGs that accept \( seq \). It clearly runs in \( O(k|seq|) \) time, where \( k \) is the number of strings in \( S \). When the running time is more critical, we can build a product of \( k \) DASGs so that the running time becomes \( O(|seq|) \) time, at the cost of preprocessing time and space requirement. This is the DASG for a set of strings.

Baeza-Yates also presented a right-to-left algorithm for building the DASG for a set of strings [5]. Moreover, Troníček and Melichar [21], and Crochemore and Troníček [7] showed left-to-right algorithms for building the DASG for a set of strings.

In [11], we considered a subsequence automaton as a deterministic complete finite automaton that recognizes all possible subsequences of a set of strings, that is essentially the same as DASG. We showed an online construction of subsequence automaton for a set of strings. Our algorithm runs in \( O(|\Sigma|(m + k) + N) \) time using \( O(|\Sigma|m) \) space, where \( |\Sigma| \) is the size of alphabet, \( N \) is the total length of strings, and \( m \) is the number of states of the resulting subsequence automaton. This is the fastest algorithm to construct a subsequence automaton for a set of strings, to the best of our knowledge. We can extend the automaton so that it answers the above Counting the matched strings problem in a natural way (See Fig. 2).

Although the construction time is linear to the size \( m \) of automaton to be built, unfortunately \( m = O(n^k) \) in general, where we assume that the set \( S \) consists of \( k \) strings of length \( n \). (The lower bound of \( m \) is only known for the case \( k = 2 \), as \( m = \Omega(n^2) \) [7].) Thus, when the construction time is also a critical factor, as in our application, it may not be a good idea to construct subsequence automaton for the set \( S \) itself. Here, for a specified parameter \( mode > 0 \), we partition the set \( S \) into \( d = k/\text{mode} \) subsets \( S_1, S_2, \ldots, S_d \) of at most \( \text{mode} \) strings, and construct \( d \) subsequence automata for each \( S_i \). When asking a query \( seq \), we have only to traverse all automata simultaneously, and return the sum of the answers. In this way, we can balance the preprocessing time with the total time to answer (possibly many) queries. In the next section, we experimentally evaluate the optimal value of the parameter \( mode \) in some situation.
Fig. 2. Subsequence automaton for $S = \{abab, abb, bb\}$, where $\Sigma = \{a, b\}$. Each number on a state denotes the number of matched strings. For example, by traverse the states according to a string $ab$, we reach the state whose number is 2. It corresponds to the cardinality $|L_{ab}(ab) \cap S| = 2$, since $ab \preceq_{\text{seq}} abab$, $ab \preceq_{\text{seq}} abb$, and $ab \npreceq_{\text{seq}} bb$.

6 Implementation and Experiments

In this section, we report some results on our experiments. We are implementing our algorithm in Fig. 1 using C++ language with Standard Template Library (STL). For the PriorityQueue, we use the standard priority queue in STL. Concerning with the StringSet, we have implemented the function $\text{numOfSubseq}(\text{seq})$ in the following two ways depending on the value of $\text{mode}$. In case of $\text{mode} = 0$, we do not use subsequence automata. For each string $w$ in the set, we check whether $\text{seq}$ is a subsequence of $w$ or not in a trivial way, and return the number of matched strings. Thus we do not need to preprocess the set. For the cases $\text{mode} \geq 1$, we construct $k/\text{mode}$ subsequence automata in the preprocess, where $k$ is the number of strings in the set. On the other hand, the function $\text{numOfSuperseq}(\text{seq})$ is implemented in a trivial way without using any special data structure.

We examined the following two data as input.

Transmembrane Amino acid sequences taken from the PIR database, that are converted into strings over binary alphabet $\Sigma = \{0, 1\}$, according to the alphabet indexing discovered by BONSAI [20]. The average length of the strings is about 30. $S_1$ consists of 70 transmembrane domains, and $T_1$ consists of 100 non-transmembrane domains.

DNA DNA sequences of yeast genome over $\Sigma = \{A, T, G, C\}$. The lengths of the strings are all 30. We selected two sets $S_2$ and $T_2$ based on the functional categories. $|S_2| = 31$ and $|T_2| = 35$.

We note that $\langle S_1, T_1 \rangle$ is an easy instance, while $\langle S_2, T_2 \rangle$ is a hard instance, in the sense that the best score for $\langle S_1, T_1 \rangle$ is high, while that for $\langle S_2, T_2 \rangle$ is low. As we will report, the facts affect the practical behaviors of our algorithm.

In order to verify the effect of the first heuristics and the second heuristics, we compared the searching time to find the best subsequence pattern of our algorithm.
pruning1 We use the first heuristics only, by commented out the lines 13 and 22.

pruning2 We use both the first and second heuristics.

exhaustive We do not use any heuristics, by commented out the lines 10, 13 and 20–23.

Our experiments were carried out both on a workstation AlphaServer DS20 with an Alpha 21264 processor at 500MHz running Tru64 UNIX operating system (WS), and on a personal computer with Pentium III processor at 733MHz running Linux (PC).

First we verified the effect of the first heuristics and the second heuristics. Fig. 3 shows the numbers of strings actually evaluated and the running time at PC, when maxLength varies and mode was fixed to 0. The both graphs (a)
Table 1. Preprocessing time and search time (seconds) at PC. The data is Transmembrane.

<table>
<thead>
<tr>
<th>mode</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>preprocessing</td>
<td>0.023</td>
<td>0.054</td>
<td>0.120</td>
<td>0.273</td>
<td>0.470</td>
<td>0.796</td>
<td>1.378</td>
<td>2.108</td>
<td>3.083</td>
<td>4.543</td>
</tr>
<tr>
<td></td>
<td>exhaustive</td>
<td>1.502</td>
<td>1.560</td>
<td>0.906</td>
<td>0.710</td>
<td>0.599</td>
<td>0.535</td>
<td>0.494</td>
<td>0.460</td>
<td>0.425</td>
<td>0.379</td>
</tr>
<tr>
<td></td>
<td>pruning1</td>
<td>0.067</td>
<td>0.077</td>
<td>0.046</td>
<td>0.037</td>
<td>0.031</td>
<td>0.025</td>
<td>0.023</td>
<td>0.022</td>
<td>0.020</td>
<td>0.019</td>
</tr>
<tr>
<td></td>
<td>pruning2</td>
<td>0.060</td>
<td>0.069</td>
<td>0.047</td>
<td>0.040</td>
<td>0.035</td>
<td>0.033</td>
<td>0.031</td>
<td>0.030</td>
<td>0.029</td>
<td>0.028</td>
</tr>
</tbody>
</table>

and (c) show that the pruning2 gives the most effective pruning with respect to the number of evaluated strings, as we expected. For example, pruning2 reduces the search space approximately half compared to pruning1, when maxLength is 14 in (c). However, the running time behaves differently as we expected. The graph (b) shows that the running time reflects the number of evaluated strings, while the graph (c) shows that pruning2 was much slower than pruning1. This is because the overhead of maintaining the set Forbidden and the response time of the query to Forbidden, since we implemented it in a trivial way. By comparing (a) and (b) with (c) and (d) respectively, we see that the instance $\langle S_1, T_1 \rangle$ of Transmembrane is easy to solve compared to $\langle S_2, T_2 \rangle$ of DNA, because some short subsequences with high score were found in an early stage so that the search space is reduced drastically.

We now verify the effect of introducing subsequence automata. Table 1 shows the preprocess time, and search time for each search method, where mode is changed from 0 to 10. We can see that the preprocessing time increases with the mode, as we expected, since the total size of the automata increases. On the other hand, the search time decreases monotonically with the mode for any search method except the case mode = 0, since each subsequence query will be answered quickly by using subsequence automata. The search time in the case mode = 1 is slightly slower than that in the case mode = 0. It implies that traversing an automaton is not so faster than naive matching of subsequence when answering subsequence queries. We suppose that the phenomena arise mainly from the effect of CPU caches.

In order to see the most preferable value of mode at which the total running time is minimized, refer to Fig. 4 (a), (b), and (c) that illustrates Table 1. The total running time, that is the sum of preprocessing and search time, is minimized at mode = 3 for exhaustive search (a). On the other hand, unfortunately, for both pruning1 in (b) and pruning2 in (c), the total running time is minimized at mode = 0. It means that in this case, subsequence automata could not reduce the running time. Especially, at the workstation (d), search without using subsequence automata (mode = 0) is much faster than any other mode. We guess that it is also caused by the CPU caches.

By these results, we verified that the pruning heuristics and subsequence automata reduce the time to find the best subsequence pattern, independently.
Fig. 4. Total running time of (a) exhaustive search and (b)(c)(d) pruning search. The experiments (a), (b) and (c) are performed at PC, while (d) at WS.

7 Concluding Remarks

We have discussed how to find a subsequence that maximally distinguishes given two sets of strings, according to a specified objective function. The only requirement to the objective function is the conicality, that is weaker than the convexity, and almost of all natural measures to distinguish two sets will satisfy the property.

In this paper, we focused on finding the best subsequence pattern. However, we can easily extend our algorithm to enumerate all strings whose values of the objective function exceed the given threshold, since essentially we examine all strings, with effective pruning heuristics. Enumeration may be more preferable in the context of text data mining [6, 9, 22].
In our current implementation, the function \texttt{numOfSuperseq} is realized in a trivial way, that slows down the pruning in some situation. If we can construct a supersequence automata efficiently, the second heuristic will be more effective.

We remark that the function \texttt{G} in Theorem 3 (2) is not actually used in our algorithm, since our algorithm starts from the empty string and tries to extend it. Another approach is also possible, that starts from a given string and tries to shrink it. In this case, the function \texttt{G} will be applicable.

In [8, 15] an episode matching is considered, where the total length of the matched strings is bounded by a given parameter. It will be very interesting to extend our approach to find the best episode to distinguish two sets of strings. Moreover, it is also challenging to apply our approach to find the best pattern in the sense of pattern languages introduced by Angluin [2], where the related consistency problems are shown to be very hard [13, 14, 17]. Arimura et al. showed an another approach to find the best proximity pattern [3, 4, 10]. It may be interesting to combine these approaches into one.

We plan to install our algorithm into the core of the decision tree generator in the BONSAI system [20].

Acknowledgements

The authors would like to thank Prof. Albert Apostolico and Prof. Hiroki Arimura for fruitful discussion.

References


On-line Estimation of Hidden Markov Model Parameters

Jun Mizuno, Tatsuya Watanabe, Kazuya Ueki, Kazuyuki Amano, Eiji Takimoto and Akira Maruoka
{mizuno, naebe, ueki, ama, t2, maruoka}@maruoka.ecei.tohoku.ac.jp

Graduate School of Information Sciences, Tohoku University
Sendai, 980-8579, Japan.

Abstract. In modeling various signals such as the speech signal by using the Hidden Markov Model (HMM), it is often required to adapt not only to the inherent nonstationarity of the signal, but to changes of sources (speakers) who yield the signal. The well known Baum-Welch algorithm tries to adjust HMM so as to optimize the fit between the model and the signal observed. In this paper we develop an algorithm, which we call the on-line Baum-Welch algorithm, by incorporating the learning rate into the off-line Baum-Welch algorithm. The algorithm performs in a series of trials. In each trial the algorithm somehow produces an HMM \( M_t \), then receives a symbol sequence \( \omega_t \), incurring loss \( -\ln \Pr(\omega_t|M_t) \) which is the negative log-likelihood of the HMM \( M_t \) evaluated at \( \omega_t \). The performance of the algorithm is measured by the additional total loss, which is called the regret, of the algorithm over the total loss of a standard algorithm, where the standard algorithm is taken to be a criterion for measuring the relative loss. We take the off-line Baum-Welch algorithm as such a standard algorithm. To evaluate the performance of an algorithm, we take the Gradient Descent algorithm. Our experiments show that the on-line Baum-Welch algorithm performs well as compared to the Gradient Descent algorithm. We carry out the experiments not only for artificial data, but for some reasonably realistic data which is made by transforming acoustic waveforms to symbol sequences through the vector quantization method. The results show that the on-line Baum-Welch algorithm adapts the change of speakers very well.

1 Introduction

The Hidden Markov Model (HMM, for short), which can be viewed as a stochastic signal model to produce sequences of symbols, has been extensively used to model the sequences of observations in various research fields such as speech recognition and computational biology. In application it is often required to adjust the model parameters of HMM to optimize the fit between the model and the signal sequences, where the fitness is measured by the likelihood that the resultant HMM assigns on the sequences. More generally this is formulated as the approximate maximum likelihood model (MLM) problem for HMMs: Find an HMM with a given topology that assigns likelihood on the input sequences

© Springer-Verlag Berlin Heidelberg 2000
not much smaller than the maximum likelihood assigned by an optimal HMM. Abe and Warmuth [1] showed that although the approximate MLM problem is equivalent to the PAC learning problem for HMMs, they are NP-hard if the alphabet size is not bounded by a constant\(^1\). Practically, the EM (expectation-maximization) technique known as Baum-Welch algorithm [7] is widely used to seek for an HMM with sufficiently large likelihood. The Baum-Welch algorithm is a method of reestimating the parameters of the HMM so as to increase the likelihood of a set of symbol sequences being observed. Given \(M_1\) as an initial HMM and a set of symbol sequences \(w^T = (w_1, \ldots, w_T)\), the Baum-Welch algorithm produces a new HMM \(M\) such that the likelihood for \(w^T\) never decreases as the model changes, i.e., \(\Pr(w^T|M_1) \leq \Pr(w^T|M)\). Here each \(w_t = (w_{t,1}, \ldots, w_{t,h_t})\) is a sequence of symbols of certain length \(h_t\), and we assume that \(\Pr(w^T|M) = \prod_{t=1}^T \Pr(w_t|M)\), that is, the likelihood for each sequence is independently evaluated. Taking the HMM obtained as the revised initial HMM and running the Baum-Welch algorithm for it, we can perform the Baum-Welch algorithm repeatedly until we get an HMM \(M\) such that the likelihood \(\Pr(w^T|M)\) converges to a maximal value. It should be noticed that since \(\Pr(w^T|M)\) might be trapped in a local maximum, the HMM \(M\) obtained in this way is not guaranteed to maximize the likelihood \(\Pr(w^T|M)\). In what follows, we call an individual symbol sequence \(w_t\) an instance.

In this paper we consider the approximate MLM problem in the on-line framework. Namely, the algorithm is presented an instance (a symbol sequence), one at a time, and at each time an instance is given, the algorithm must produce an HMM in an attempt that the likelihood for the next instance is large. Our work is motivated by applications such as the speaker adaptation, where instances are generated by a signal source (speaker) that may change with time. So, our objective is not to get a single HMM \(M\) as the final hypothesis that assigns high likelihood on the whole instances (i.e., to make \(\Pr(w^T|M)\) large), but to get an HMM \(M_t\) for each time \(t\) that predicts the next instance \(w_{t+1}\) with high probability (i.e., to make \(\Pr(w_t|M_t)\) large).

More precisely, the protocol proceeds in trials as follows. In each trial \(t\), the algorithm somehow produces an HMM \(M_t\) and then receives an instance \(w_t\). In this trial the algorithm incurs a loss defined as

\[
L(w_t, M_t) = -\ln \Pr(w_t|M_t),
\]

which is the negative log-likelihood that \(M_t\) assigns on \(w_t\). The total loss of the algorithm for the whole sequence \(w^T\) of the instances is given by \(\sum_{t=1}^T L(w_t, M_t)\).

First consider the following regret or relative loss

\[
\sum_{t=1}^T L(w_t, M_t) - \inf_{M^*} \sum_{t=1}^T L(w_t, M^*).
\]

\(^1\) More exactly, the results are proved for the class of probabilistic automata (PAs) which are closely related to HMMs. In particular, the hardness result for PAs implies that the approximate MLM problem for HMMs with some topology is NP-hard.
This is the additional total on-line loss of the algorithm over the total loss of the best HMM \( M^* \). Our first goal of the algorithm is to minimize the regret no matter what the instance sequence \( w^T \) is given. As the regret gets smaller, the algorithm is more likely to have performed as well as the best off-line (batch) algorithm. (Later we will give another notion of regret called the adaptive-regret that captures the adaptiveness of algorithms more effectively.) Note that the best HMM \( M^* \) is the maximum likelihood model for \( w^T \) because \( \sum_{t=1}^T L(w_t, M^*) = -\ln \prod_{t=1}^T \Pr(w_t | M^*) = -\ln \Pr(w^T | M^*) \).

There are a number of similar related works on the on-line estimation of HMM parameters [6, 4]. The protocol used there is slightly different from ours in that symbols (not sequences) are presented on-line, resulting in a long single sequence. Namely, in each trial \( t \) after producing an HMM \( M_t \), the algorithm observes a symbol \( w_t \) rather than a symbol sequence. Note that our protocol can be seen as a special case of theirs, because \( w^T \) can be regarded as a single sequence if the “reset” symbol is assumed to be inserted between instances to form the single sequence. In other words, our protocol gives the algorithm the additional information on the times when the model should be reset. Obviously, for applying to speech recognition, our protocol is more appropriate because the breaths in speech play the role of the reset symbol. Moreover, in the usual setting, a target HMM \( M_0 \) is assumed to exist, which generates the sequence \( w = (w_1, \ldots, w_T) \), and the performance of the algorithm is measured in terms of the speed of convergence, i.e., how fast the hypotheses \( M_t \)'s converge to the target \( M_0 \). This contrasts with our framework in which we do not need to assume any target HMM and the performance of the algorithm is measured in terms of the regret.

There is a large body of work on proving regret bounds that has its root in the Minimum Description Length community [10, 14, 11, 12, 15, 16]. So far the results developed in this area can apply only to the case where the model class of probability density functions is simple, say, to the exponential families\(^2\). In most cases the regrets are of the form of \( O(\ln T) \). For a similar related problem for linear regression, a simple algorithm was developed and its regret was shown to be \( O(\ln T) \) [13, 2]. Cover and Odentlich considered a more complex problem of universal portfolio and showed that again \( O(\ln T) \) regret can be achieved [9]. So we conjecture that for the on-line MLM problem of HMMs there exists an algorithm that achieves \( O(\ln T) \) regret. However, as suggested in [1], such an algorithm should be computationally infeasible (if it exists) because even in the off-line setting, obtaining a model that approximates to the best model \( M^* \) is NP-hard.

So we have to give up developing an efficient on-line algorithm that is competitive with the best off-line algorithm. Instead we let the on-line algorithm compete with the Baum-Welch algorithm used in off-line fashion. Namely, we

\(^2\) The exponential families include many fundamental classes of distributions such as Bernoulli, Binomial, Poisson, Gaussian, Gamma and so on.
consider the following regret

$$
\sum_{t=1}^{T} L(w_t, M_t) - \sum_{t=1}^{T} L(w_t, M_{h}),
$$

where $M_{h}$ is the HMM obtained by running the Baum-Welch algorithm on $w^T$ until $Pr(w^T|M_{h})$ converges. We call $M_{h}$ the limit model, starting from the initial model $M_{i}$. Now we restate our first goal as follows: Minimize the regret of (1).

In this paper we give a simple heuristic on-line algorithm based on the Baum-Welch algorithm. More specifically, in each trial $t$, we first get the HMM $M'_{i}$ by feeding $M_{i}$ and $w_{t}$ to the Baum-Welch algorithm. Then we merge two models $M'_{i}$ and $M_{i}$ with some mixture rate $0 < \eta_{t} < 1$ to obtain a new model $M_{i+1}$.

Note that the mixture rate $\eta_{t}$ may depend on $t$. Actually, it is observed from our experiments that the choice of rating scheme influences the performance of the algorithm very much. Intuitively, a smaller $\eta_{t}$ makes the algorithm more conservative, while a larger $\eta_{t}$ makes the algorithm more adaptive to recent instances. We call this algorithm the on-line Baum-Welch algorithm.

To see the performance of this algorithm, we did some experiments with artificially created data and compare the results to another on-line algorithm based on Gradient Descent (GD) method [3]. The results show that the on-line Baum-Welch algorithm with an appropriate rating scheme is slightly better than the GD algorithm. However it turns out that the rating scheme that gives small regret strongly depends on the data we deal with.

One of the redeeming features of on-line algorithms is that they naturally adapt to the change of environment. We examined the adaptability of the two algorithms using instance sequences formed by concatenating several sequences that are generated by different sources, which are artificially designed HMMs. Surprisingly, the regret was negative, which means that the on-line algorithms perform well as compared to the off-line Baum-Welch algorithm. The reason why this happens is that the on-line algorithms tend to follow the change of sources, whereas the limit model $M_{h}$ reflects the source’s structure which is in common with a variety of sequences in $w^T$. To measure the adaptiveness of on-line algorithms, we introduce the notion of adaptive-regret. Suppose that all $T$ trials are partitioned into $p$ intervals $T_{1}, \ldots, T_{p}$ with $\{1, \ldots, T\} = T_{1} \cup \cdots \cup T_{p}$ such that the instances $W_{i} = \{w_{t} \mid t \in T_{i}\}$ in the $i$th interval are assumed to be generated by the $i$th source. Let $M_{h_i}$ be the limit model obtained by applying the off-line Baum-Welch algorithm on the instances $W_{i}$ in the $i$th interval. Then the adaptive-regret is defined as

$$
\sum_{t=1}^{T} L(w_t, M_t) - \sum_{i=1}^{p} \sum_{t \in T_{i}} L(w_t, M_{h_i}) = \sum_{i=1}^{p} \left( \sum_{t \in T_{i}} L(w_t, M_t) - \sum_{t \in T_{i}} L(w_t, M_{h_i}) \right).
$$

Namely, to get small adaptive-regret, in every $i$th interval the algorithm must perform as well as $M_{h_i}$ without knowing the change of intervals. Our second goal is to minimize the adaptive-regret. We test the on-line algorithms on symbol sequences extracted from real speech data pronounced by several speakers. Our algorithms seem to adapt the change of speakers very well.
2 The Baum-Welch algorithm

The integers \( \{1, \ldots, N\} \) denote the states of an HMM. In particular state 1 is assumed to denote the initial state. The integers \( \{1, \ldots, K\} \) denotes observation symbols as well. An HMM \( M \) is specified by a pair \((A, B)\) of matrices. The first matrix \( A \) is of dimension \([N, N]\) and its \((i, j)\) component \( a_{i,j} \) denotes the transition probability of moving from state \( i \) to state \( j \). The second matrix \( B \) is of dimension \([N, K]\) and its \((j, k)\) component \( b_{j,k} \) denotes the probability of symbol \( k \) being output provided current state \( j \). Clearly
\[
\sum_{j=1}^{N} a_{i,j} = 1 \quad \text{and} \quad \sum_{k=1}^{K} b_{j,k} = 1
\]
must hold for every \( 1 \leq i \leq N \). An HMM \( M = (A, B) \) generates a sequence of symbols. In what follows we assume that an HMM makes state transitions a certain times denoted \( l \). The HMM \( M \) performs \( l \) state transitions as follows: Starting in the initial state \( 1 \), it iterates the following for \( m = 1, \ldots, l \). Suppose that the current state is \( s_{m-1} = i \) (Note that \( s_0 = 1 \)). Then it chooses state \( j \) with probability \( a_{i,j} \) and moves to \( s_m = j \). After arriving at state \( j \), it chooses symbol \( k \) with probability \( b_{j,k} \) and outputs the symbol \( w_m = k \). Let the sequence obtained be denoted by \( \mathbf{w} = (w_1, \ldots, w_l) \in \{1, \ldots, K\}^l \). The probability that the HMM \( M \) generates the sequence \( \mathbf{w} \) on a particular sequence of states \( s = s_1, \ldots, s_l \) is given as
\[
\Pr(\mathbf{w}, s | M) = \prod_{m=1}^{l} a_{s_{m-1}, s_m} b_{s_m, w_m}.
\]
Hence the probability (likelihood) that \( M \) generates \( \mathbf{w} \) is given as
\[
\Pr(\mathbf{w} | M) = \sum_{s \in \{1, \ldots, N\}^l} \Pr(\mathbf{w}, s | M).
\]
We call a sequence \( \mathbf{w} \) an instance.

We are now ready to give the Baum-Welch algorithm. When given an initial HMM \( M_1 \) together with a sequence of instances \( \mathbf{w}^T = (\mathbf{w}_1, \ldots, \mathbf{w}_T) \), the Baum-Welch algorithm tries to find the HMM \( M \) that maximizes the concave function \( \Pr(\mathbf{w}^T | M_1) \log \Pr(\mathbf{w}^T | M) \) rather than \( \Pr(\mathbf{w}^T | M) \). This is a standard EM (Expectation-Maximization) technique (See, e.g., [8]) that guarantees that \( \Pr(\mathbf{w}^T | M_1) \leq \Pr(\mathbf{w}^T | M) \) holds. Solving this maximization problem we can derive the algorithm to update the parameters. Let \( \#(i \rightarrow j) \) and \( \#(k \mid j) \) be the random variables that denote the number of transitions from state \( i \) to state \( j \) and the number of symbol \( k \) output at state \( j \), respectively, in a sequence of state transitions we consider. We compute the expected number of these variables given that the sequence \( \mathbf{w}^T \) is observed. Here the expectation is taken with respect to the probability space specified by \( M_1 \). More precisely, for all \( 1 \leq i, j \leq N \) and \( 1 \leq k \leq K \), we compute

\[
m_{i,j} = \frac{1}{T} \sum_{t=1}^{T} E(\#(i \rightarrow j) \mid \mathbf{w}_t, M_1)
\]
and

\[
m_{j,k} = \frac{1}{T} \sum_{t=1}^{T} E(\#(k \mid j) \mid \mathbf{w}_t, M_1),
\]
where $E(\cdot)$ denotes the expected number. We also compute the expected number of state $i$ visited given that $w^T$ is observed, i.e.,

$$n_i = \sum_{j=1}^{N} n_{i,j}.$$  

These expectations can be efficiently computed by applying the well-known forward-backward algorithm to $M_1 = (A_1, B_1)$ initially given. Then we update the parameters according to

$$a_{i,j} = \frac{n_{i,j}}{n_i},$$  

$$b_{j,k} = \frac{m_{j,k}}{n_j},$$

which give the new model $M = (A, B)$. We call $M$ obtained in this way the on-pass model from $M_1$. Moreover, repeating the above procedure with $M_1 = M$ until $Pr(w^T | M)$ converges, we get the limit model $M_B$. Generally, the choice of the initial model $M_1$ affects the performance of the limit model $M_B$. Especially, the topology (the number of states and the transitions of zero probability) of HMMs is inherited from the initial model through the Baum-Welch updates.

### 3 On-line Baum-Welch algorithm

In the on-line setting, a natural way to reestimate the parameters of the HMM would be to do exactly what the off-line Baum-Welch algorithm would do after seeing the last instance. This is called the incremental on-line update rule [2]. Namely, after seeing the $t$th instance $w_t$, the incremental off-line algorithm computes

$$n_{i,j}^t = \frac{1}{t} \sum_{q=1}^{t} E(\#(i \rightarrow j) \mid w_q, M_1),$$  

$$m_{j,k}^t = \frac{1}{t} \sum_{q=1}^{t} E(\#(k \mid j) \mid w_q, M_1),$$

$$n_i^t = \sum_{j=1}^{N} n_{i,j}^t$$

and then update the parameters according to $a_{i,j} = n_{i,j}^t/n_i^t$ and $b_{j,k} = m_{j,k}^t/n_j^t$. Note that we can recursively compute these expectations very efficiently as follows:

$$n_{i,j}^{t+1} = \left(1 - \frac{1}{t}\right) n_{i,j}^t + \frac{1}{t} E(\#(i \rightarrow j) \mid w_t, M_1),$$  

$$m_{j,k}^{t+1} = \left(1 - \frac{1}{t}\right) m_{j,k}^t + \frac{1}{t} E(\#(k \mid j) \mid w_t, M_1).$$  

(2)  

(3)
This suggests that we should maintain the expectation parameters $n_{i,j}^t$ and $m_{j,k}^t$ instead of directly maintaining the model parameters $a_{i,j}$ and $b_{j,k}$. The above update rules for the expectation parameters can be viewed as a convex combinations of the old parameters and the expectations for the last instance $w_t$, where the expectations are taken w.r.t. the initial model $M_1$. Clearly, the final model specified by $n_{i,j}^T$ and $m_{j,k}^T$ after seeing all instances $w^T$ is the same as the one-pass model from $M_1$. Since in most cases the performance of the one-pass model is much worse than that of the limit model $M_0$, we cannot expect that the regret of the incremental off-line algorithm is reasonably small. It should be noticed that throughout all trials the initial model $M_1$ continues to be used for taking the expectations in (2) and (3), which would make the algorithm too conservative.

In an attempt to adjust the model to the sequence more quickly, we use the most recent model $M_t$ instead of $M_1$ for taking the expectations. Accordingly we use some mixture rate $\eta_t$ instead of $1/t$ in general. That is, we propose the following update rules:

$$n_{i,j}^t = (1 - \eta_t)n_{i,j}^{t-1} + \eta_t E(\#(i \rightarrow j) \mid w_t, M_t),$$

$$m_{j,k}^t = (1 - \eta_t)m_{j,k}^{t-1} + \eta_t E(\#(k \mid j) \mid w_t, M_t).$$

The model parameters $(A, B)$ are implicitly represented through the expectation parameters in terms of the followings: $a_{i,j} = n_{i,j} / \sum_{s=1}^N n_{i,s}$ and $b_{j,k} = m_{j,k} / \sum_{s=1}^N m_{j,s}$. We call this algorithm the On-line Baum-Welch algorithm. The way of specifying the mixture rate $\eta_t$ is called the rating scheme. For instance, the rating scheme for the incremental off-line algorithm is $\eta_t = 1/t$.

4 The GD algorithm

The GD (Gradient Descent) algorithm is a general scheme for finding a maximal (minimal) point of a given function $f$. For the sake of completeness we briefly show how the GD algorithm works. When given an initial point $x$, it slightly changes the point to $x'$ according to the rule $x' = x + \eta \nabla f(x)$ for some learning rate $\eta$, where $\nabla f(x)$ denotes the gradient of $f$ evaluated at $x$. Unfortunately, since the parameter space is bounded, i.e., $a_{i,j}, b_{j,k} \in [0,1]$, the update might move the points out of the parameter space. To make the quantities stay in the parameter space, Baldi and Chauvin [3] introduced a transformation that maps the original parameter space $(A, B)$ to an unbounded parameter space $(U, V)$. More precisely, the algorithm maintains the parameters $u_{i,j}$ and $v_{j,k}$ that represent the original parameters in terms of

$$a_{i,j} = e^{u_{i,j}} / \sum_{j} e^{u_{i,j}};$$

$$b_{j,k} = e^{v_{j,k}} / \sum_{k} e^{v_{j,k}}.$$
It is clear that while the new parameter space is the unbounded one, the corresponding parameters given by the above equations stay within the parameter space. Applying the GD algorithm in the \((U, V)\) space, we have the following update rule for \((U, V)\). Suppose that the current parameter is \(M_t = (U, V)\) and a new instance \(w_t\) is given. Then the GD algorithm update the parameters according to

\[
    u_{i,j}^{t+1} = u_{i,j} + \eta \left( E(\#(i \rightarrow j) \mid w_t, M_t) - a_{i,j} \sum_{s=1}^{N} E(\#(i \rightarrow s) \mid w_t, M_t) \right),
\]

\[
    v_{j,k}^{t+1} = v_{j,k} + \eta \left( E(\#(k \mid j) \mid w_t, M_t) - b_{j,k} \sum_{s=1}^{N} E(\#(i \rightarrow s) \mid w_t, M_t) \right).
\]

Finally it produces the new HMM \(M_{t+1} = (U', V')\).

5 Experiments

We tested the performances of the on-line Baum-Welch and the GD algorithms by some experiments.

5.1 Regret

To see how the choice of the rating scheme \(\eta_t\) influences the performance of the on-line Baum-Welch algorithm, we examined the regret of the algorithm with various rating schema. The instance sequences we used are generated by some artificially designed HMMs with \(N = 3\) and \(K = 2\), and all instances \(w_t = (w_{t,1}, \ldots, w_{t,t})\) are of the common length \(t = 10\). In this section we take two particular instance sequences, denoted \(w^T_1\) and \(w^T_2\), respectively, which are generated by different sources. In our experiments, we only consider the polynomial rating scheme of the form of \(\eta_t = c(1/t)^d\) for some non-negative constants \(c\) and \(d\). Recall that the regret of the On-line Baum-Welch algorithm (OBW, for short) is denoted by

\[
    R(OBW, w^T) = \sum_{t=1}^{T} L(w_t, M_t) - \sum_{t=1}^{T} L(w_t, M_B),
\]

where \(M_t\) is the HMM that the OBW algorithm produces before seeing the instance \(w_t\) and \(M_B\) is the limit model for \(w^T\), starting from the same initial model as the one used by the OBW algorithm. Although the choice of the initial model also affects the regret, we simply chose it randomly in our experiments. Note that the regret on the sequence \(w^T\) can be viewed as a function of \(T\). We show in Fig. 1 how fast the regret grows with \(T\) for various polynomial rating schema \(\eta_t = (1/t)^d\) (We fixed \(c = 1\)). In particular, the regrets shown in Fig. 1-(a) and Fig. 1-(b) are on the instance sequences \(w^T_1\) and \(w^T_2\), respectively. From these experiments we can see that even the asymptotic behavior of the regret
Fig. 1. The regrets of the OBW algorithm with rating schema $\eta_t = (1/t)^d$ for various $d$. Fig. (a) represents the regret on $w_t^1$ and Fig. (b) on $w_t^2$. A reasonable choice of rating scheme for $w_t^1$ would be $\eta_t = \sqrt{1/t}$ while $\eta_t = 1/t$ for $w_t^2$. 
Fig. 2. The regrets of the OBW algorithm with some of rating schema $\eta_t = c(1/t)^d$ compared with those of the GD algorithm with the optimally tuned learning rates. $\eta = 0.3$ for $\mathbf{w}_t^1$ and $\eta = 0.025$ for $\mathbf{w}_t^2$. Fig. (a) and Fig. (b) show the results for $\mathbf{w}_t^1$ and $\mathbf{w}_t^2$, respectively.
depends both on the input sequence and on the rating scheme. Moreover, the best choice of \( d \) for the polynomial rating scheme depends on the input sequence.

To compare the performance of the OBW algorithm with the GD algorithm, we applied the GD algorithm on the same sequences \( w_I^t \) and \( w_F^T \). We show in Fig. 2 the regrets \( R(\text{GD}, w^T) \) of the GD algorithm as well as the regrets \( R(\text{OBW}, w^T) \) of the OBW algorithm with some rating schema \( \eta_t = c(1/t)^d \).

Again Fig. 2-(a) and Fig. 2-(b) are the regrets on \( w_I^t \) and \( w_F^T \), respectively. For each sequence the learning rate \( \eta \) for the GD algorithm is optimized: We chose \( \eta = 0.3 \) for \( w_I^t \) and \( \eta = 0.025 \) for \( w_F^T \). Namely, the optimal learning rate for the GD algorithm also strongly depends on the input sequence. In Fig. 2 we include the regrets of the OBW algorithm with constant rating schema \( \eta_t \) = c. These constants \( c = 1/4 \) for \( w_I^t \) and \( c = 1/60 \) for \( w_F^T \) are chosen to minimize the regrets. For both sequences, we can see that the OBW algorithm with appropriate choices of the rating scheme performed better than the GD algorithm with the optimal learning rate.

5.2 Adaptability

We did an experiment to see how the on-line algorithms adapt to changes of the signal source. To see this we used three different HMMs with \( N = K = 5 \). For each of them we get a sequence of 500 instances, hence obtaining three sequences. Again the length of all instances is set to \( l = 10 \). By concatenating these three sequences, we finally get a single sequence \( w^T \) of length \( T = 1500 \). First we observe the following regret-per-trial of algorithm \( A \in \{ \text{OBW, GD} \} \):

\[
R_t(A, w^T) = L(w_t, M_t) - L(w_t, M_B)
\]

for each trial \( t \). Here the model \( M_B \) is the limit model for the whole sequence \( w^T \). Thus the regret-per-trials sum up to the regret, i.e., \( R(A, w^T) = \sum_{t=1}^{T} R_t(A, w^T) \).

In Fig. 3 we give the regrets-per-trial \( R_t(\text{OBW}, w^T) \) and \( R_t(\text{GD}, w^T) \). We tuned the rates for both algorithms appropriately and set \( \eta_t = 0.03 \) for the OBW algorithm and \( \eta = 0.25 \) for the GD algorithm. It is interesting to note that a diminishing rating scheme \( \eta_t \) with \( \lim_{t \to \infty} \eta_t = 0 \) for the OBW algorithm no longer works because in the situation where the source is changing, the algorithm have to "forget" the past data to follow the recent change. Intuitively, a constant rate makes the algorithm forget the past data exponentially fast.

Surprisingly, we can see that the regrets \( R(\text{OBW}, w^T) \) and \( R(\text{GD}, w^T) \) are negative. That is, the online algorithms beat the off-line Baum-Welch algorithm. Moreover, although the online algorithms perform badly at the moment that the source is changed (\( t = 500 \) and \( t = 1000 \)), they rapidly adapt the new source.

Next we compare the performance of the on-line algorithms with switching HMMs, rather than with the single model \( M_B \). More precisely, for each interval \( \mathcal{T}_i = \{500(i - 1) + 1, \ldots, 500i\} (i \in \{1, 2, 3\}) \), let \( M^i_B \) be the limit model for the instances \( \{w_t | t \in \mathcal{T}_i\} \) in the \( i \)th interval. In other words, \( M^i_B \) models the source of the \( i \)th interval. Now we consider the following adaptive regret-per-trial of algorithm \( A \):

\[
AR_t(A, w^T) = L(w_t, M_t) - L(w_t, M^i_B)
\]
**Fig. 3.** The regret-per-trial for a sequence generated by changing sources. The source is changed at every 500 trials. The rating scheme and the learning rate for the OBW and the GD algorithms are tuned appropriately: $\eta_r = 0.03$ for OBW and $\eta = 0.25$ for GD.

**Fig. 4.** The adaptive regret-per-trial. The setting of the experiment is same as in Fig. 3.
where \( t \in \mathcal{T}_i \), and define the \textit{adaptive regret} as

\[
AR(A, w^T) = \sum_{t=1}^{T} AR_t(A, w^T).
\]

Fig. 4 shows the adaptive regret-per-trial of the OBW and the GD algorithms. From this we can see that the on-line algorithms adapt the HMM to the source of each interval without knowing when the source is changed.

### 5.3 Experiments on speech data

We did the same experiments on natural speech data. The data consists of the acoustic waveforms pronounced by 10 speakers. There are 503 sentences per speaker. The acoustic waveforms was transformed to symbol sequences by the vector quantization (VQ) method with 1024 symbols, and then segmented into subsequences that correspond to phonemes. We then extract from them subsequences that correspond to vowels. Namely, for each speaker \( i \) and each vowel \( p \in \{/a/, /o/, /u/, /i/, /e/\} \), we made a set of sequences \( W_{i,p} \). A member of \( W_{i,p} \) is a symbol sequence (an instance) of vowel \( p \) pronounced by speaker \( i \). We then concatenated the 10 sets of all speakers for a particular vowel \( p \) to make an instance sequence \( w^T \). That is, \( w^T \) is a sequence of instances in \( W_p = \bigcup_{i=1}^{10} W_{i,p} \). For example, for \( p = /i/ \), \( W_{i,p} \) contains 1818 instances of \( p \) for each speaker \( i \). So \( w^T \) is a sequence of length \( T = 18180 \) and the speaker changes at every 1818 trials in \( w^T \). The length of instances ranges from 3 to 15. For this sequence \( w^T \), we tested the performance of the OBW and the GD algorithms. Here we restrict the topology of HMM to the left-right model with three states (See Fig. 5), which has been successfully utilized in speech recognition to model phonemes. In Fig. 6 we show that the adaptive regret-per-trial for the instance sequence \( w^T \) from \( W_{i,p} \). The algorithms seem to adapt the change of speakers very well. The regret \( R(A, w^T) \) and the adaptive regret \( AR(A, w^T) \) for \( A \in \{\text{OBW, GD}\} \) is shown below. The OBW algorithm seems to perform slightly better than the GD algorithm.

<table>
<thead>
<tr>
<th>( A )</th>
<th>( R(A, w^T) )</th>
<th>( AR(A, w^T) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBW</td>
<td>-117630</td>
<td>65397</td>
</tr>
<tr>
<td>GD</td>
<td>-108631</td>
<td>74396</td>
</tr>
</tbody>
</table>
Fig. 6. The adaptive regret-per-trial for $W_{ij}$. The speaker changes at every 1818 trials. The rating scheme for the OBW algorithm is $\eta_t = 0.003$ and the learning rate for the GD algorithm is $\eta = 0.25$.

6 Concluding remarks

We proposed an on-line variation of the Baum-Welch algorithm (the OBW algorithm) and evaluated the performance of the algorithm in terms of the regret, which measures the relative performance compared to the off-line Baum-Welch algorithm. As far as we know, such a competitive analysis is the first attempt for parameter estimation of HMMs. Throughout the experiments, the OBW algorithm seems to perform marginally better than the GD algorithm. In particular, the algorithm could be applied to speaker adaptation.

However, it is not clear how the rating scheme $\eta_t$ should be chosen. Intuitively, while the instances come from a single source, the rate should be somehow decreasing with time. But it should be quickly recovered as the source changes. So we have to develop a rating scheme that depends on the past sequence.

Another method to make the algorithm adaptive would be to make use of the idea behind the variable share algorithm, which is developed for on-line linear regression where the best regressor changes with time [5]. At the end of each trial the algorithm shares a fraction of parameters to others according to the likelihood for the last instance, so that if the likelihood becomes small (which is interpreted as the source changed), the parameters of too small value quickly get recovered.

Theoretically it is interesting to investigate the worst case regret of the algorithms. We would like to have $o(T)$ regret for any input sequence. But Fig. 1
suggested that the regret of both the OBW and the GD algorithms grow linearly
in $T$. The performance could be improved by using a more sophisticated rating
scheme and/or the sharing method.

Acknowledgment

The authors are grateful to Syozo Makino and Yuichi Ohkawa for useful dis-

cussions and preparing the speech data, and to anonymous referees for valuable

comments.

References

1. N. Abe and M. K. Warmuth. On the computational complexity of approximating
with the exponential family of distributions. In 15th Conference on Uncertainty
4. D. Collins and T. Rydén. A new maximum likelihood gradient algorithm for on-
line Hidden Markov Model identification. In ICASSP 98, volume 4, 2261–2264,
1998.
parameters based on the Kullback-Leibler information measure. IEEE Transaction
7. S. E. Levinson, L. R. Rabiner, and M. M. Sondhi. An introduction to the appli-
cation of the theory of probabilistic functions of a markov process to automatic
8. G. McLachlan and T. Krishnan. The EM Algorithm and Extensions. A Wiley-
10. J. Rissanen. Fisher information and stochastic complexity. IEEE Transactions on
11. J. Takeuchi and A. Barron. Asymptotically minimax regret for exponential
12. J. Takeuchi and A. Barron. Asymptotically minimax regret by bayes mixtures. In
Department of Computer Science, Royal Holloway, University of London, 1997.
14. Q. Xie and A. Barron. Asymptotic minimax regret for data compression, gambling,
15. K. Yamanishi. A decision-theoretic extension of stochastic complexity and its
In Proc. 10th ALT, volume 1720 of Lecture Notes in Artificial Intelligence, 26–38,
1999.
Computationally Efficient Heuristics for If-Then Rule Extraction from Feed-Forward Neural Networks

Hyemecheol Kim

Department of Computer Science Education, College of Education, Korea University, Anam-Dong 5-Ga, Seoulo, 136-701, Korea
hkim@comedu.korea.ac.kr

Abstract. In this paper, we address computational complexity issues of decompositional approaches to if-then rule extraction from feed-forward neural networks. We also introduce a computationally efficient technique based on ordered-attributes. It reduces search space significantly and finds valid and general rules for single nodes in the networks. Empirical results are shown.

1 Introduction

How to extract if-then rules from a trained feed-forward neural network has been investigated by researchers [1, 2, 3, 4, 5, 7, 8]. However, the rule extraction processes are computationally expensive since the rule search space is increased exponentially with the number of input attributes (i.e., input domain). Several techniques have been introduced to reduce the searching complexity of decompositional approaches such as KT heuristics and complexity control parameters [3, 4, 5] described in Sect. 2. Some other techniques include the soft-weight-sharing used in MofN algorithm [9]. It clusters the weights into equivalence classes. This approach, however, requires a special training technique and is not sufficiently accurate description of the network.

This paper summarizes the existing heuristics used to reduce the rule search space and proposes a computationally efficient technique, the ordered-attribute search. The proposed algorithm extracts valid and general rules from single nodes in $O(n \log n)$ for most cases. Empirical results are also provided.

2 Complexity Issues in Rule Extraction

2.1 If-Then Rules

A rule has the form \emph{“if the premise, then the conclusion.”} The premise is composed of a number of positive attributes (e.g., $x_i$) and negative attributes (e.g., $\neg x_i$), and so is the conclusion (e.g., $c_i$ and $\neg c_i$). A rule is called a confirming rule if the conclusion is $c_i$, or a disconfirming rule if the conclusion is $\neg c_i$. In
the basic form of a rule, the rule’s premise is limited to a conjunction of attributes and the rule’s conclusion is limited to a single attribute (i.e., a single class). However, the presence of multiple rules with same conclusion represents disjunction. A rule with a conjunction of conclusions is represented by multiple rules with same premise but different conclusions.

Quality of a rule is evaluated with a few criteria. First of all, a rule should be valid. The validity for a rule is defined as follows. Whenever the rule’s premise holds, so does its conclusion in the presence of any combination of the values of attributes not referenced by the rule [3]. Other criteria include accuracy (specificity or goodness of fit) and generality (simplicity). Accuracy is about how often the rule is classified correctly. Generality is about how often the left-hand side of a rule occurs. It is also related to coverage of the rule. As the rule gets simpler (i.e., shorter), it covers more instances in the input domain.

2.2 Decompositional Approach

Decompositional approaches to rule extraction from a trained neural network (i.e., a feed-forward multi-layered neural network) involves the following phases:

1. Intermediate rules are extracted at the level of individual units within the network.
2. The intermediate rules from each unit are aggregated to form the composite rule base for the neural network. It rewrites rules to eliminate the symbols which refer to hidden units but are not predefined in the domain. In the process, redundancies, subsumptions, and inconsistencies are removed.

At each non-input unit of a network, \( n \) incoming connection weights and a threshold are given. Rule extraction at the unit finds a set of incoming attribute combinations that are valid and maximally-general. Fig. 1 gives an example: A combination \( (x_1, x_2) \) is a valid rule because its weight-sum (i.e., \( 3 + 4 = 7 \)) is always greater than the threshold (i.e., \(-1\)) regardless of the values of other incoming units. Another example is (not \( x_3 \)) which is also valid.

Fig. 1. An individual node with five incoming connection weights and a threshold. Input values for input nodes are binary: 1 or 0.
2.3 Computational Complexity

Decompositional algorithms extract intermediate rules from each non-input nodes and then rewrite them to form composite rule base for the network. In this paper, we focus on the computational complexity in the rule extraction from a single node with n incoming binary attributes. Given the n attributes, there are $2^n$ (i.e., $\sum_{l=0}^{n} C_l^n$) possible combinations of the attributes and $3^n$ (i.e., $\sum_{l=0}^{n} 2^l C_l^n$) possible rules in the rule space.

For a combination with $l$ attributes (i.e., $0 \leq l \leq n$), each attribute could be encoded positive or negative. Therefore, there are $2^l$ possible rule representations for the combination. For each combination of size $l$, we discard $2^l - 1$ representations for the combination from rule search space instantly by the following two steps:

1. For each attribute, the attribute is encoded positive if its corresponding weight is positive; it is encoded negative otherwise. For example, attributes $(x_1, x_3, x_5)$ in Fig. 1 is represented as “$x_1$ and not $x_3$ and $x_5$” for its candidate rule. The lowest total-input for a candidate rule $R$ is defined as follows:

$$\text{total-input}_{\text{lowest}}(R) = \sum_{i=1}^{n} w_i x_i \begin{cases} w_i x_i = |w_i| & \text{if } x_i \in P \\ x_i = 1 & \text{if } w_i < 0 \text{ and } x_i \notin P \\ x_i = 0 & \text{if } w_i \geq 0 \text{ and } x_i \notin P \end{cases}$$

where the $P$ is a set of attributes in the premise of $R$.

2. Validity is tested for the candidate rule. By the definition of validity, the candidate rule is valid if its lowest total-input is greater than its threshold.

Validity testing costs $O(1)$ for each combination and thus this method reduces rule search space from $3^n$ to $2^n$.

It is NP-hard to obtaining all possible combinations of rules and a feasible alternative is often to extract key rules that cover most of the domain knowledge. In this paper, we want to search $b$ best rules, that is, the most simple but valid rule combinations among all $2^n$ possible combinations of incoming weight connections.

2.4 Tree-Based Search

Tree-based search uses a tree structure where each node is one of $2^n$ combinations. At depth $l$ in the search tree, length of combinations is $l$. The tree-based search begins with a root node (i.e., 0 or the shortest length combination) to a leaf node (i.e., $n$ or the longest length combination), checking its validity. If it finds a valid rule combination, it stops generating children nodes and takes it as a rule with the following property: “if a combination of a node is a valid rule, combinations of its descendant nodes are eliminated from the search tree space.” which reduces search space. For $b = 1$, the worst case $O(2^n)$ occurs when the rule
is a leaf node combination of length $n$. The worst case for any $b > 1$ occurs at $C_{l+1}^n < b \leq C_l^n$ where $\lceil \frac{n}{b} \rceil \leq l < n$. Its worst-case search cost for any $b$ is:

$$O\left(\sum_{l=0}^{\lceil \frac{n}{b} \rceil} C_l^n\right) \leq O(\text{Tree-based search}) \leq O\left(\sum_{l=0}^{n} C_l^n\right) = O(2^n) \quad (2)$$

**KT Heuristics.** Fu’s KT algorithm [2, 3, 4] is an improved tree search algorithm which reduces search space and memory space heuristically. The KT algorithm reduces the search space by dividing the $n$ incoming attributes into $p$ positive and $q$ negative attributes (st. $p + q = n$). The basic idea of the separation derives from the fact that the complexity of an exponential space is greater than the sum of complexities of its sub-spaces. That is, $x^l \geq (a \cdot x)^l + (b \cdot x)^l$ where $l > 1, a < 1, b < 1$ and $a + b = 1$. For each combination in positive attribute sub-space, its validity is tested. If the positive attribute combination is not valid, negative attribute sub-space is searched to form a valid combination composed of both positive and negative attributes. Thus the search space size for positive attribute sub-space is $\sum_{l=0}^{p} C_l^p = 2^p$. Even though this approach reduces the complexity in most cases, the worst case complexity remains same if $p = n$ (i.e., no negative attributes). If every positive attribute combination needs negative attribute search, the search space is also $2^p \cdot 2^q = 2^n$.

**Complexity Control Parameters.** In addition to the heuristics, some parameters are introduced to control the complexity of the rule search procedure [5]. Some of the parameters include:

- **Representative attributes:** When there are too many incoming connections to concept nodes, only highly weighted connections are considered. This makes sense because input summation seems to be determined by a few highly weighted (highly relevant) connections, not by all the connections. Thus, we can set a threshold on the weight or pick a certain percentage of attributes with high pertinence. The attributes selected are called representative attributes.

- **Certainty factor threshold ($\theta_{CF}$):** The certainty factor (CF) of a rule is defined as the sigmoid transformation of the value obtained by subtracting the threshold from the weight sum of all the attributes in the rule’s premise. If the CF is in [0.5, 1.0], the rule is a confirming rule. If the CF is in [0.0, 0.5], the rule is a disconfirming rule and convert it into [-1.0, -0.5]. We may discard rules with the CF lower than a predefined threshold.

- **Rule simplification and merging similar rules:** The generated rules are refined by simplifying them using symbolic logic.

The parameters control the trade-off of quality of rules and searching complexity. However, finding optimal values for the parameters is mostly accomplished trial and error and might add another computational complexity to the process.
3 Ordered-Attribute Search

Gallant [6] describes a procedure to find a single rule to explain the conclusion reached by the neural network given a case. His method involves the ordering of attributes based on absolute weight values. However, he argues that the procedure works only if the network was very small; the number of implicitly encoded in-then rules can grow exponentially with the number of incoming units. In this section, we introduce a different search algorithm called the ordered-attribute search (OAS) algorithm. The OAS algorithm involves the following three procedures:

1. Attribute Contribution Scoring: For each attribute, its contribution to a candidate rule is calculated.
2. Attribute Sorting: Attributes are sorted in descending order according to their contribution scores.
3. Rule Searching: With the attributes sorted by their contribution scores, valid and maximally general rules are to be searched.

Consider a rule search tree. A root node (i.e., length 0) represents a default rule candidate $R_{\text{default}}$. Adding attributes to the $R_{\text{default}}$ increases (or decreases) its lowest total-input by the amount of weight values of the attributes. For binary attribute cases, the contribution score $C(x_i)$ of an attribute $x_i$ is defined as follows: $C(x_i) = |w_i|$. Therefore, a positive (or negative) attribute $x_i$ added increases the lowest total-input of $R_{\text{default}}$ by $C(x_i)$. The $\text{esum}(R)$ is defined as a sum of contribution scores of attributes in the premise of the candidate rule $R$. Then lowest total-input for the $R$ is defined as follows:

$$\text{total-input}_{\text{lowest}}(R) = \text{esum}(R) + \text{total-input}_{\text{lowest}}(R_{\text{default}})$$

Note that total-input$_{\text{lowest}}(R)$ is proportional to $\text{esum}(R)$ since the the lowest total-input of $R_{\text{default}}$ is constant.

After the contribution scores are calculated, attributes are sorted in descending order by their contribution scores as follows: $a_1, a_2, a_3, \ldots, a_n$. Then, we use the generic algorithm that lists the combinations of length $l$ in the following order:

$$(a_1, a_2, \ldots, a_{l-1}, a_l), \ldots (a_1, a_2, \ldots, a_{l-1}, a_n), \ldots (a_1, a_3, \ldots, a_{l+1}), \ldots (a_{n-l+1}, \ldots, a_n)$$

At each depth $l$ (i.e., $1 \leq l \leq n$) in an OAS search tree, the number of combinations is $\binom{n}{l}$ and they are listed from the left to the right as defined at listing (4).

**Corollary 1.** In the list of combinations of length $l$, the first combination (i.e., the left-most one) holds the highest $\text{esum}$. The last one in the list holds the lowest $\text{esum}$.

Due to Corollary 1, it is straightforward to find the valid combination of shortest length. Starting from the depth 1 at an OAS search tree, only the first node (i.e.,
combination) at each depth is tested for validation. If it is not valid, the other
nodes at the depth are not valid either (by Corollary 1). Worst case complexity
for this process is \( O(n) \). Let’s say that the first valid combination is found at
depth \( d \) after \( d \) tests. Then the following lemma is true.

**Lemma 2.** All the combination nodes from depth 0 to depth \( (d-1) \) are not valid.

**Proposition 3.** By the definition of \( csum \) and combination listing in listing (4),
the following inequalities are true:

1. \( csum(a_{s_1}, a_{s_2}, \ldots, a_{s_n}) > csum(a_{s_1}, \ldots, a_{s_{n-1}}, a_{s_n}) \)
2. \( csum(a_{s_1}, \ldots, a_{s_{n-1}}, a_{s_n} > csum(a_{s_1}, \ldots, a_{s_{n-1}}, a_{s_n}, a_{s_{n+1}} \geq \max(s_{j+1}), \ldots, a_{s_{n+1}} \geq \max(s_{j+1}, i_{j+1})) \)
3. \( csum(a_{s_1}, a_{s_2}, \ldots, a_{s_n}) > csum(a_{s_1}, a_{s_2}, a_{s_3} \geq \max(s_{j+1}), \ldots, a_{s_n} \geq \max(s_{j+1}, i_{j+1})) \).

The \( V \) set of a combination \( R \) is defined as the set of all combinations with
smaller \( csum \), as defined by the three inequalities in Proposition 3. Then the
the following is true.

**Lemma 4.** Given a combination at depth \( d \), if the combination is not valid,
combinations of its \( V \) set are not valid either.

The three inequalities in Proposition 3 reduce the search space significantly.
At depth 1, validity is tested from the first combination in the list; valid one
is stored in a rule candidate set and then next one in the list is tested; if it
is not valid, its \( V \) set is eliminated from the search space instantly. For example,
let the smallest size \( l \) be 3 and the number of attributes \( n \) be 10. Then
the number of possible combinations is \( C_{10}^{3} = 120 \). Consider a combination
\( R = (a_1, a_2, a_3) \). By the inequality 1 in Proposition 3, lowest total-input of \( R \)
without the terms of \( (a_4, a_5, a_6) \ldots (a_{10}, a_{10}) \). Thus, if the \( R \) is not valid,
\( \sum_{i=0}^{l-1} 1 = 4 \) combinations are eliminated from the search space. By the
inequality 2, \( csum \) of \( R \) is greater than the ones of \( (a_1, a_2, a_3, a_{l+1}) \). It
eliminates \( \sum_{j=3}^{n-1} \sum_{k=0}^{n-k-1} 1 = 20 \) combinations. By the inequality 3,
\( csum \) of \( R \) is greater than the ones of \( (a_1, a_2 \geq \max(s_{j+1}), a_{l+1}) \). It
eliminates \( \sum_{j=3}^{n-1} \sum_{k=0}^{n-k-1} 1 = 80 \) combinations. The \( csum(R) \)
is greater than \( csum \) of 104 other combinations. So is the lowest total-input of
the \( R \). This heuristic reduces the search space significantly. Fig. 2 illustrates
the number of combinations that can be eliminated when current combination (i.e.,
combination id) in the list is not valid.

Another heuristic in the OAS is from the property of tree structure: If a
combination at depth \( l \) is valid, its child combinations at depth \( l + 1 \) are also
valid. Therefore the combinations subsumed to a valid rule (i.e., parent node)
can be discarded from the search space. From the OAS heuristics, the following
types of combinations are eliminated from the search tree:
1. invalid combinations (Lemma 2 and 4) and
2. valid combinations which are subsumed to a valid combination.

All other combinations at the tree are subject to a validity test for rule candidacy.

In the problem of rule extraction from a neuron unit which involves \( n \) incoming binary attributes, we want to find the \( b \) rules which are valid and maximally general (i.e., the rules having shorter length than others). Even though there are \( 2^n \) possible rules, only \( C_{n/2}^n \) unique rules are possible in maximum because longer length rules are subsumed to a shorter length rule. Hence the following is true:

\[
1 \leq b \leq C_{\frac{n}{2}}^n.
\]

Contribution scoring procedure costs \( O(n) \). Sorting phase costs \( O(n \log n) \) with well-known sorting algorithms such as heap sort. Once attributes are sorted, searching the best rule costs \( O(l) \) where \( 1 \leq l \leq n \). The \( l \) is the smallest length of valid rules to be searched. When \( b \) is greater than 1, the search starts from length \( l \) combinations, eliminating \( \sum_{k=0}^{l-1} C_k^n \) combinations from search space. Even when searching \( C_l^n \) combinations, it reduces a large part of search space by Proposition 3. Thus complexity of the OAS is:

\[
O(n \log n) \leq O(GAS) \leq O(n \log n) + O(C_{\frac{n}{2}}^n).
\] (5)

In summary, worst-case complexities of three search algorithms for any \( b \) are

\[
\begin{align*}
O(\text{Exhaustive search}) &= O(2^n) \\
O(2^{n-1}) &\leq O(\text{Tree-based search}) \leq O(2^n) \\
O(n \log n) &\leq O(\text{Ordered-attribute search}) < O(2^{n-1}).
\end{align*}
\] (6)
4 Empirical Results

The OAS algorithm is applied to public domains: XOR, Iris and Hypothyroid. For XOR problem, we show intermediate rules extracted from each non-input units and final rules rewritten with the intermediate ones. An example of rewriting procedure is also provided. For Iris and Hypothyroid, we show the performance of individual rules and rulesets generated by the OAS. Performance of an individual rule is evaluated by coverage and accuracy defined by: coverage = (pos + neg)/total and rule accuracy = pos/(pos + neg) where pos is the number of positive instances, neg is the number of negative instances, and total is the number of instances in the test set. Note that (pos + neg) is the number of instances covered by the rule, and thus total = pos + neg + notcovered where the notcovered is the number of instances not covered by the rule. Performance of a ruleset is evaluated by the following: coverage = (pos + neg)/total, confidence = pos/(pos + neg) and classification accuracy = pos/total. The accuracy of a ruleset is defined differently than that of a single rule. It is the performance of a ruleset classifier over the complete domain.

4.1 XOR

A fully-connected network is configured with 2 binary input attributes (i.e., x0 and x1), 4 hidden units (i.e., h0, h1, h2 and h3) and 1 output unit (i.e., y). Intermediate rules extracted from the non-input nodes and final rules are listed in Table 1. Rewriting procedure aggregates intermediate rules by eliminating the hidden unit symbols which are not predefined in the domain. Rewriting starts with the output layer and rewrites rules of one layer every time in terms of rules of the next layer closer to the input of the network. For example, we rewrite the rule R12 from output unit y “if not h0 and not h2 then y = 0.965”. Since the “not h0” and “not h2” are not defined in the domain, they are replaced by R1 and R6, respectively. After simple logical refinement, final rule FR12 is obtained. Note that R14 is rewritten into two final rules FR14-1 and FR14-2 while R15 does not form any valid final rules. The OAS generates a complete set of final rules for the XOR problem.

4.2 Iris Domain

The Fisher’s iris domain involves 4 continuous-valued features and 3 concepts. Each continuous-valued feature is discretized to three interval attributes, resulting in a total 12 binary input attributes. A three-layered neural network (12-4-3) is configured. Two experiments are performed: (1) comprehensive capability is evaluated by looking at how well training data instances are translated into a small set of concise rules, and (2) prediction capability is evaluated by looking at how well the extracted ruleset classifies unseen data instances, compared with the network.

A network is trained with a data set of 150 instances (50 instances for each concept) and accuracy is 98.67%, with only two instances mis-classified. The OAS
<table>
<thead>
<tr>
<th>Intermediate rules</th>
<th>Rule</th>
<th>Condition</th>
<th>Confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>from h0</td>
<td>R1.</td>
<td>x0 and x1</td>
<td>h0 = 0.827</td>
</tr>
<tr>
<td></td>
<td>R2.</td>
<td>not x0</td>
<td>h0 = 0.785</td>
</tr>
<tr>
<td></td>
<td>R3.</td>
<td>not x1</td>
<td>h0 = 0.780</td>
</tr>
<tr>
<td>from h1</td>
<td>R4.</td>
<td>x0 and x1</td>
<td>h1 = 0.787</td>
</tr>
<tr>
<td></td>
<td>R5.</td>
<td>not x0 and not x1</td>
<td>h1 = 0.778</td>
</tr>
<tr>
<td>from h2</td>
<td>R6.</td>
<td>x0 and x1</td>
<td>h2 = 0.847</td>
</tr>
<tr>
<td></td>
<td>R7.</td>
<td>not x0</td>
<td>h2 = 0.819</td>
</tr>
<tr>
<td></td>
<td>R8.</td>
<td>not x1</td>
<td>h2 = 0.805</td>
</tr>
<tr>
<td>from h3</td>
<td>R9.</td>
<td>x0</td>
<td>h3 = 0.980</td>
</tr>
<tr>
<td></td>
<td>R10.</td>
<td>x1</td>
<td>h3 = 0.980</td>
</tr>
<tr>
<td></td>
<td>R11.</td>
<td>not x0 and not x1</td>
<td>h3 = 0.937</td>
</tr>
<tr>
<td>from y</td>
<td>R12.</td>
<td>not h0 and not h2</td>
<td>y = 0.966</td>
</tr>
<tr>
<td></td>
<td>R13.</td>
<td>h3</td>
<td>y = 0.974</td>
</tr>
<tr>
<td></td>
<td>R14.</td>
<td>h0 and h2 and not h3</td>
<td>y = 0.992</td>
</tr>
<tr>
<td></td>
<td>R15.</td>
<td>h1 and h2 and not h3</td>
<td>y = 0.885</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rewritten final rules</th>
<th>Rule</th>
<th>Condition</th>
<th>Confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FR12.</td>
<td>x0 and x1</td>
<td>y = 0.966</td>
</tr>
<tr>
<td></td>
<td>FR13.</td>
<td>not x0 and not x1</td>
<td>y = 0.974</td>
</tr>
<tr>
<td></td>
<td>FR14-1.</td>
<td>not x0 and x1</td>
<td>y = 0.992</td>
</tr>
<tr>
<td></td>
<td>FR14-2.</td>
<td>x0 and not x1</td>
<td>y = 0.992</td>
</tr>
</tbody>
</table>

Table 1. XOR: The intermediate rules and final rules.

(Ordered-Attribute Search) algorithm is applied to the network and nine rules are obtained Table 2. Performance of the ruleset and each individual rules are evaluated over the 150 instances and are listed in Table 3.

For prediction accuracy evaluation, the set of 150 instances is divided into 2 sets with 75 instances each. A network1 is trained with a training set (i.e., dataset1) and evaluated with a test set (i.e., dataset2). A ruleset1 is generated from the network1 and evaluated with the test set. A network2 and a ruleset2 are generated with dataset2 and evaluated with dataset1. Prediction accuracy of the networks and extracted rulesets are illustrated in Table 4.

### 4.3 Hypothyroid Disease Domain

The hypothyroid disease data set involves 2 concepts (hypothyroid and negative) and 26 variables: 7 continuous-valued and 19 binary attributes. The data set contains 3163 instances: 151 hypothyroid and 3012 negative instances. Some instances contain several missing values.

Since the data set involves continuous-valued variables, missing values and unbalance between the two concept instances, preprocessing is accomplished before experiments are performed. At first, the instances that include missing values in attribute TSH, TT4 and FTI are filtered out, leaving 2694 instances (150 hypothyroid, 2544 negative). To collect the same number of instances for each
**Table 2.** Iris: Nine individual rules.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Size</th>
<th>Coverage (%)</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>2</td>
<td>30.0</td>
<td>100</td>
</tr>
<tr>
<td>R2</td>
<td>2</td>
<td>50.0</td>
<td>100</td>
</tr>
<tr>
<td>R3</td>
<td>3</td>
<td>28.0</td>
<td>100</td>
</tr>
<tr>
<td>R4</td>
<td>3</td>
<td>2.0</td>
<td>100</td>
</tr>
<tr>
<td>R5</td>
<td>2</td>
<td>45.5</td>
<td>98.9</td>
</tr>
<tr>
<td>R6</td>
<td>3</td>
<td>10.0</td>
<td>100</td>
</tr>
<tr>
<td>R7</td>
<td>2</td>
<td>39.5</td>
<td>98.7</td>
</tr>
<tr>
<td>R8</td>
<td>2</td>
<td>12.5</td>
<td>96</td>
</tr>
<tr>
<td>R9</td>
<td>2</td>
<td>16.5</td>
<td>96.97</td>
</tr>
</tbody>
</table>

**Table 3.** Iris: Performances of individual rules and Comprehension accuracy of a ruleset.

<table>
<thead>
<tr>
<th>Ruleset</th>
<th>Size</th>
<th>Coverage (%)</th>
<th>Confidence (%)</th>
<th>Classification (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>9</td>
<td>98.67</td>
<td>98.65</td>
<td>97.33</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Prediction (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>network1</td>
</tr>
<tr>
<td>network2</td>
</tr>
</tbody>
</table>

**Table 4.** Iris: Prediction accuracy of networks and rulesets.
concepts, 150 negative instances are selected randomly from the 2544 negatives. Thus, a data set of 300 instances (150 hypothyroids and 150 negatives) is obtained for experiments. The data set is divided into two different sets, each containing 150 instances (75 hypothyroids, 75 negatives). One set is used for training and the other set for testing. Among the 25 variables, 24 variables are used for this experiment because variable TGB includes too many missing values (91.8% missing). Thus two variables related to TGB (TGB and TGB measured) are excluded in this experimental data set. Six continuous-valued attributes are discretized to several binary attributes according their interval distributions, resulting in total 52 binary attributes.

A 3-layered neural network is configured (52-5-1) and trained with a training set. Rules are extracted from the trained network using the OAS algorithm and evaluated over training and testing set. Performances of neural networks and extracted rule sets are listed in Table 5. Neural network 1 is the one trained on Set1, and network 2 is trained on Set2. Ruleset1 is a set of rules extracted from neural network 1 and ruleset2 is from network2. Individual rules in ruleset2 are listed in Table 6 and their performance (coverage and accuracy) are in Table 7.

<table>
<thead>
<tr>
<th>Classification (%)</th>
<th>set1</th>
<th>set2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network 1 (52-5-1)</td>
<td>98.7</td>
<td>96.0</td>
</tr>
<tr>
<td>Network 2 (52-5-1)</td>
<td>97.3</td>
<td>98.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Coverage (%)</th>
<th>Confidence (%)</th>
<th>Classification (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>set1</td>
<td>set2</td>
<td>set1</td>
</tr>
<tr>
<td>Ruleset1 (7 rules)</td>
<td>75.33</td>
<td>64.7</td>
</tr>
<tr>
<td>Ruleset2 (8 rules)</td>
<td>88</td>
<td>87.3</td>
</tr>
</tbody>
</table>

Table 5. Hypothyroid: Performance of two rule sets extracted from two neural networks

5 Conclusions

We addressed computational complexity issues in extracting if-then rules from a trained feed-forward neural networks and summarized related heuristics. In this paper, we introduced a computationally efficient technique based on ordered-attributes. This technique extracts valid and general rules from single nodes in \(O(n \log n)\) most cases and less than \(O(2^n - 1)\) worst case. It is applied to well-known public domain data sets and the experimental results are provided. The extracted rules are evaluated with coverage, accuracy and confidence and shown to be efficient and comprehensive.
Table 6. Hypothyroid: Eight individual rules in ruleset 2.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Size</th>
<th>Coverage (%)</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>3</td>
<td>29.3</td>
<td>100</td>
</tr>
<tr>
<td>R2</td>
<td>3</td>
<td>34.7</td>
<td>100</td>
</tr>
<tr>
<td>R3</td>
<td>3</td>
<td>46.7</td>
<td>100</td>
</tr>
<tr>
<td>R4</td>
<td>3</td>
<td>26.7</td>
<td>100</td>
</tr>
<tr>
<td>R5</td>
<td>4</td>
<td>15.3</td>
<td>100</td>
</tr>
<tr>
<td>R6</td>
<td>4</td>
<td>39.3</td>
<td>100</td>
</tr>
<tr>
<td>R7</td>
<td>3</td>
<td>19.3</td>
<td>100</td>
</tr>
<tr>
<td>R8</td>
<td>5</td>
<td>30.7</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 7. Hypothyroid: Performances of individual rules in ruleset 2. Evaluated on test set (set1).

In this work, we focused on domain complexity of discrete input attributes. However, complexity in continuous-valued domain is much more difficult and how to handle it is very important research issue. Another issue that needs to be investigated is the intermediate rule aggregation complexity in the decompositional approaches. The complexity is also increased with the number of intermediate rules extracted from single nodes.

References

5. Fu, LiMin, Kim, Hyeoncheol.: Abstraction and Representation of Hidden Knowledge in an Adapted Neural Network. unpublished, CISE, University of Florida (1994)
Language Learning with a Neighbor System∗

Yasuhiito Mukouchi and Masako Sato

Department of Mathematics and Information Sciences
College of Integrated Arts and Sciences
Osaka Prefecture University, Sakai, Osaka 599-8531, Japan
e-mail: {mukouchi, sato}@mi.cias.osakai-u.ac.jp

Abstract. We consider inductive language learning from positive examples, some of which may be incorrect. In the present paper, the error or incorrectness we consider is the one described uniformly in terms of a distance over strings. Firstly, we introduce a notion of a recursively generable distance over strings, and define a $k$-neighbor closure of a language $L$ as the collection of strings each of which is at most $k$ distant from some string in $L$. Then we define a $k$-neighbor system as the collection of original languages and their $j$-neighbor closures with $j \leq k$, and adopt it as a hypothesis space. In ordinary learning paradigm, a target language, whose examples are fed to an inference machine, is assumed to belong to a hypothesis space without any guarantee. In this paper, we allow an inference machine to infer a neighbor closure instead of the original language as an admissible approximation. We formalize such kind of inference, and give some sufficient conditions for a hypothesis space.

1 Introduction

In many real-world applications of machine discovery or machine learning from examples, we have to deal with incorrect examples. In the present paper, we consider language learning from observed incorrect examples together with correct examples, i.e., from imperfect examples. Some correct examples may not be presented to the learner. It is natural to consider that each observed incorrect example has some connection with a certain correct example on a target language to be learned. The incorrect examples we consider here are the ones described uniformly in terms of a distance over strings. Assume that the correct example is a string $v$ and the observed example is a string $w$. In case we are considering the so-called Hamming distance and two strings $v$ and $w$ have the same length but differ just one symbol, then we estimate the incorrectness as their distance of one. In case we are considering the edit distance and $w$ can be obtained from $v$ by deleting just one symbol and inserting one symbol in another place, then we estimate the incorrectness as their distance of two. Firstly, we introduce a notion of a recursively generable distance over strings, and define a $k$-neighbor closure of a language $L$ as the collection of strings each of which is at most $k$ distant


© Springer-Verlag Berlin Heidelberg 2000
from some string in \( L \). Then we define a \( k \)-neighbor system as the collection of original languages and their \( j \)-neighbor closures with \( j \leq k \), and adopt it as a hypothesis space.

There are various approaches to language learning from incorrect examples (cf. e.g. Jain [5], Stephan [15], and Case and Jain [3]). Stephan [15] has formulated a model of noisy data, in which a correct example crops up infinitely often, and an incorrect example only finitely often. There is no connection between incorrect examples considered there and correct examples.

In 1967, Gold [4] introduced a notion of identification in the limit. An inference machine \( M \) is said to identify a language \( L \) in the limit, if the sequence of guesses from \( M \), which is successively fed a sequence of examples of \( L \), converges to a correct expression of \( L \), that is, all guesses from \( M \) become a unique expression after a certain finite time and that the expression is a correct one. In this criterion, a target language, whose examples are fed to an inference machine, is assumed to belong to a hypothesis space which is given in advance. However, this assumption is not appropriate, if we want an inference machine to infer or to discover an unknown rule which explains examples or data obtained from scientific experiments. In their previous paper, Mukouchi and Arikawa [10] discussed both refutability and inferability of the hypothesis space concerned from examples. If a target language is a member of the hypothesis space, then an inference machine should identify the target language in the limit, otherwise it should refute the hypothesis space itself in a finite time. They showed that there are some rich hypothesis spaces that are refutable and inferable from complete examples (i.e., positive and negative examples or an informant), but refutable and inferable classes from only positive examples are very small. In relation to refutable inference, Lange and Watson [9] and Mukouchi [12] also proposed inference criteria relaxing the requirements of inference machines, and Jain [6] also deals with the problem for recursively enumerable languages. On the other hand, Mukouchi [11] took a minimal language as an admissible approximate language within the hypothesis space, and forced an inference machine to converge to an expression of a minimal language of the target language which may not belong to the hypothesis space. Kobayashi and Yokomori [7] also proposed inference criterion requiring an inference machine to infer an admissible approximate language within the hypothesis space concerned.

As mentioned above, the obtained examples may have errors, and thus the observed language consisting of the observed examples may not belong to the hypothesis space, even when the original target language belongs to the hypothesis space. Therefore we have to take account of languages not belonging to the hypothesis space concerned. In this paper, we also take a minimal language as an admissible approximate language within the hypothesis space for the observed language. By doing this, we guarantee that if the observed examples have no errors and the target language is in the hypothesis space, then an inference machine converges to a correct expression of the target language. Furthermore, by taking a \( k \)-neighbor system as a hypothesis space, we can expect an inference machine to infer an original target language, even when the observed examples have some
errors. Roughly speaking, an inference machine $M$ k-neighbor-minimally infers a class $\mathcal{C}$ from positive examples, if for every observed language $L$, $M$ converges an expression of $L$'s minimal language within a k-neighbor system which is least distant from some language in $\mathcal{C}$.

We formalize k-neighbor-minimal inferability, and give some sufficient conditions for a hypothesis space. Finally, as its application, we show that the class of pattern languages is k-neighbor-minimally inferable from positive examples.

2 Preliminaries

2.1 A Language and a Distance

Let $\Sigma$ be a fixed finite alphabet. Each element of $\Sigma$ is called a constant symbol. Let $\Sigma^+$ be the set of all nonnull constant strings over $\Sigma$ and let $\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$, where $\varepsilon$ is the null string. A subset $L$ of $\Sigma^*$ is called a language. For a string $w \in \Sigma^*$, the length of $w$ is denoted by $|w|$.

A language $L \subseteq \Sigma^*$ is said to be recursive, if there is a computable function $f : \Sigma^* \to \{0, 1\}$ such that $f(w) = 1$ iff $w \in L$ for $w \in \Sigma^*$.

We consider a distance between two strings defined as follows:

**Definition 1.** Let $N = \{0, 1, 2, \cdots\}$ be the set of all natural numbers. A function $d : \Sigma^* \times \Sigma^* \to N \cup \{\infty\}$ is called a distance over strings, if it satisfies the following three conditions:

(i) For any $v, w \in \Sigma^*$, $d(v, w) = 0$ iff $v = w$.
(ii) For any $v, w \in \Sigma^*$, $d(v, w) = d(w, v)$.
(iii) For any $u, v, w \in \Sigma^*$, $d(u, v) + d(v, w) \geq d(u, w)$.

A distance $d$ is said to be recursive, if there is an effective procedure that computes $d(v, w)$ for any $v, w \in \Sigma^*$ with $d(v, w) \neq \infty$.

Then we define a k-neighbor closure of a language as follows:

**Definition 2.** Let $d : \Sigma^* \times \Sigma^* \to N \cup \{\infty\}$ be a distance over strings and let $k \in N$.

The k-neighbor closure $\overline{w}^{(d,k)}$ of a string $w \in \Sigma^*$ w.r.t. $d$ is the set of all strings each of which is at most $k$ distant from $w$, that is, we put $\overline{w}^{(d,k)} = \{v \in \Sigma^* \mid d(v, w) \leq k\}$.

The k-neighbor closure $\overline{L}^{(d,k)}$ of a language $L \subseteq \Sigma^*$ w.r.t. $d$ is the set of all strings each of which is at most $k$ distant from some string in $L$, that is, we put $\overline{L}^{(d,k)} = \bigcup_{w \in L} \overline{w}^{(d,k)} = \{v \in \Sigma^* \mid \exists w \in L \text { s.t. } d(v, w) \leq k\}$.

By Definition 1, we see that $\{w\} = \overline{w}^{(d,0)} \subseteq \overline{w}^{(d,1)} \subseteq \overline{w}^{(d,2)} \subseteq \cdots$ and $L = \overline{L}^{(d,0)} \subseteq \overline{L}^{(d,1)} \subseteq \overline{L}^{(d,2)} \subseteq \cdots$.

The following lemma is obvious:

**Lemma 1.** Let $d$ be a distance and let $k \in N$.

For a language $L \subseteq \Sigma^*$ and for a string $w \in \Sigma^*$, $w \in \overline{L}^{(d,k)}$ if and only if $\overline{w}^{(d,k)} \cap L \neq \emptyset$.185
For a set $S$, we denote by $|S|$ the cardinality of $S$.

**Definition 3.** A distance $d$ is said to have finite thickness, if for any $w \in \Sigma^*$, $\mathfrak{m}^{(d,k)}$ is finite.

A distance $d$ is said to be recursively generable, if $d$ has finite thickness and there exists an effective procedure that on inputs $k \in N$ and $w \in \Sigma^*$ enumerates all elements in $\mathfrak{m}^{(d,k)}$ and then stops.

We note that a notion of the recursively generable finite-set-valued function was introduced by Lange and Zeugmann [8].

**Example 1.** (a) We consider a distance known as the Hamming distance. For a string $w$ and for an integer $i$ with $1 \leq i \leq |w|$, by $w[i]$, let us denote the $i$-th symbol appearing in $w$. For two strings $v,w \in \Sigma^*$, let

$$d(v,w) = \begin{cases} 2 | \{ i \mid 1 \leq i \leq |v|, v[i] \neq w[i] \} |, & \text{if } |v| = |w|, \\ \infty, & \text{if } |v| \neq |w|. \end{cases}$$

Clearly, this distance $d$ is recursively generable.

(b) Next, we consider a distance known as the edit distance. Roughly speaking, the edit distance $d$ over two strings $v,w \in \Sigma^*$ is the least number of editing steps needed to convert $v$ to $w$. Each editing step consists of a rewriting step of the form $a \rightarrow \varepsilon$ (a deletion), $\varepsilon \rightarrow b$ (an insertion), or $a \rightarrow b$ (a change), where $a,b \in \Sigma$.

Clearly, this distance $d$ is recursively generable.

Let $d$ be a recursively generable distance and let $k \in N$. Then, for any $v,w \in \Sigma^*$, by checking $v \in \mathfrak{m}^{(d,k)}$, whether $d(v,w) \leq k$ or not is recursively decidable. Therefore $d$ turns to be a recursive distance. Let $L \subseteq \Sigma^*$ be a recursive language. Then, for any $w \in \Sigma^*$, by checking $\mathfrak{m}^{(d,k)} \cap L \neq \emptyset$, whether $w \in \mathfrak{L}^{(d,k)}$ or not is recursively decidable. Therefore $\mathfrak{L}^{(d,k)}$ is also a recursive language.

In the present paper, we exclusively deal with a recursively generable distance, and simply refer it as a distance without any notice.

### 2.2 Inferability from Examples

**Definition 4 (Angluin [2]).** A class $\mathcal{L} = \{ L_i \}_{i \in N}$ of languages is said to be an indexed family of recursive languages, if there is a computable function $f : N \times \Sigma^* \rightarrow \{0,1\}$ such that $f(i,w) = 1$ iff $w \in L_i$.

In what follows, we assume that a class of languages is an indexed family of recursive languages, and identify a class with a hypothesis space.

A positive presentation, or a text, of a nonempty language $L \subseteq \Sigma^*$ is an infinite sequence $w_1, w_2, \cdots \in \Sigma^*$ such that $\{ w_1, w_2, \cdots \} = L$. In what follows, $\sigma$ or $\delta$ denotes a positive presentation, $\sigma[n]$ denotes the $\sigma$'s initial segment of length $n \in N$, and $\sigma[n]^+$ denotes the set of all elements appearing in $\sigma[n]$.

An inductive inference machine (IIM, for short) is an effective procedure, or a certain type of Turing machine, which requests inputs from time to time.
and produces positive integers from time to time. The outputs produced by the machine are called guesses. For an IIM $M$ and for a finite sequence $\sigma[n] = w_1, w_2, \ldots, w_n$, by $M(\sigma[n])$, we denote the last guess of $M$ which is successively presented $w_1, w_2, \ldots, w_n$ on its input requests.

Then we define the inferability of a class of languages as follows:

Definition 5 (Gold [4], Angluin [2]). An IIM $M$ is said to converge to an index $i$ for a positive presentation $\sigma$, if there is an $n \in N$ such that for any $m \geq n$, $M(\sigma[m]) = i$.

An IIM $M$ is said to infer a class $\mathcal{L}$ in the limit from positive examples, if for any $L_i \in \mathcal{L}$ and for any positive presentation $\sigma$ of $L_i$, $M$ converges to an index $j$ for $\sigma$ such that $L_j = L_i$.

A class $\mathcal{L}$ is said to be inferable in the limit from positive examples, if there is an IIM which infers $\mathcal{L}$ in the limit from positive examples.

In the above definition, the behavior of an IIM is not specified, when we feed a positive presentation of a language which is not in the class concerned. On the other hand, Mukouchi [11] proposed inference criterion requiring an inference machine to infer an admissible approximate language within the hypothesis space concerned.

Let $S$ be a subset of $\Sigma^*$ and let $\mathcal{L}$ be a class. Then a language $L \subseteq \Sigma^*$ is a minimal language of $S$ within $\mathcal{L}$, if (i) $S \subseteq L$ and (ii) for any $L_i \in \mathcal{L}$, $S \subseteq L_i$ implies $L_i \not\subseteq L$.

The set of all minimal languages in $\mathcal{L}$ of $S$ within $\mathcal{L}$ is denoted by $\text{MIN}(S, \mathcal{L})$.

Definition 6 (Mukouchi [11]). An IIM $M$ is said to minimally infer a class $\mathcal{L}$ from positive examples, if it satisfies the following condition: For any nonempty language $L \subseteq \Sigma^*$ and for any positive presentation $\sigma$ of $L$, if $\text{MIN}(L, \mathcal{L}) \neq \phi$, then $M$ converges to an index $i$ for $\sigma$ such that $L_i \in \text{MIN}(L, \mathcal{L})$.

A class $\mathcal{L}$ is said to be minimally inferable from positive examples, if there is an IIM which minimally infers $\mathcal{L}$ from positive examples.

Now, we introduce our successful learning criterion we consider in the present paper.

Definition 7. Let $d$ be a distance and let $k \in N$.

For a class $\mathcal{L} = \{L_i\}_{i \in N}$, let us put $\mathcal{L}^{(d,k)} = \{L_{i}^{(d,k)}\}_{i \in N}$.

A k-neighbor system $\mathcal{L}^{(d,\leq k)}$ of a class $\mathcal{L}$ w.r.t. $d$ is the collection of languages each of which is a $j$-neighbor closure w.r.t. $d$ of some language in $\mathcal{L}$ for some $j \leq k$, that is, we put $\mathcal{L}^{(d,\leq k)} = \bigcup_{j=0}^{k} \mathcal{L}^{(d,j)}$.

For a nonempty language $L \subseteq \Sigma^*$, a pair $(i, j) \in N \times N$ is said to be a weak $k$-neighbor-minimal answer for $L$, if $j \leq k$ and $L_{i}^{(d,j)} \in \text{MIN}(L, \mathcal{L}^{(d,\leq k)})$.

For a nonempty language $L \subseteq \Sigma^*$, a pair $(i, j) \in N \times N$ is said to be a $k$-neighbor-minimal answer for $L$, if (i) $(i, j)$ is a weak $k$-neighbor-minimal answer for $L$ and (ii) for any pair $(i', j')$ with $j' < j$, $L_{i'}^{(d,j')} \notin \text{MIN}(L, \mathcal{L}^{(d,\leq k)})$. 

Language Learning with a Neighbor System 187
An IPM $M$ is said to \textit{k-neighbor-minimally} (resp., \textit{weak k-neighbor-minimally}) infer a class $\mathcal{L}$ w.r.t. $d$ from positive examples, if it satisfies the following condition: For any nonempty language $L \subseteq \Sigma^*$ and for any positive presentation $\sigma$ of $L$, if $\text{MIN}(L, \overline{L}_i^{(d, \leq k)}) \neq \phi$, then $M$ converges to an integer $\langle i, j \rangle$ for $\sigma$ such that $(i, j)$ is a $k$-neighbor-minimal (resp., weak $k$-neighbor-minimal) answer for $L$, where $\langle \cdot, \cdot \rangle$ represents the Cantor’s pairing function.

A class $\mathcal{L}$ is said to be \textit{k-neighbor-minimally} (resp., \textit{weak k-neighbor-minimally}) inferable w.r.t. $d$ from positive examples, if there is an IPM which $k$-neighbor-minimally (resp., weak $k$-neighbor-minimally) infers $\mathcal{L}$ w.r.t. $d$ from positive examples.

We note that, by the definition, a class $\mathcal{L}$ is (weak) $0$-neighbor-minimally inferable w.r.t. $d$ from positive examples, if and only if $\mathcal{L}$ is minimally inferable from positive examples.

Assume that a class $\mathcal{L}$ is $k$-neighbor-minimally inferable w.r.t. $d$ from positive examples for some $k \in N$ and for some distance $d$. For any $L_i \in \mathcal{L}$, $L_i = \overline{L}_i^{(d, k)}$ and $\text{MIN}(L_i, \overline{L}_i^{(d, \leq k)}) \neq \{ \}$, then $L_i$ is a $k$-neighbor-minimal answer for $L_i$, thus $L_i = L_{i'}$ and $j' = 0$. Thus the class $\mathcal{L}$ is also inferable in the limit from positive examples.

Therefore (weak) $k$-neighbor-minimal inferability can be regarded as a natural extension of ordinary inferability as well as minimal inferability.

The rest of this section is devoted to summarize some known results related to this study.

Angluin [2] characterized an ordinary inferability as follows:

\textbf{Definition 8 (Angluin [2])}. Let $\mathcal{L} = \{ L_i \}_{i \in N}$ be a class. A set $S \subseteq \Sigma^*$ is said to be a \textit{finite tell-tale set} of a language $L_i \in \mathcal{L}$ within $\mathcal{L}$, if (i) $S$ is a finite subset of $L_i$ and (ii) for any $L_j \in \mathcal{L}$, $S \subseteq L_j$ implies $L_j \not\subseteq L_i$.

\textbf{Theorem 1 (Angluin [2])}. A class $\mathcal{L}$ is inferable in the limit from positive examples, if and only if there is an effective procedure which on input $i$ enumerates a finite tell-tale set $S$ of $L_i \in \mathcal{L}$ within $\mathcal{L}$.

Some sufficient but useful conditions for ordinary inferability have been presented.

\textbf{Definition 9 (Wright [16], Motoki et al. [13])}. A class $\mathcal{L}$ is said to \textit{have infinite elasticity}, if there are two infinite sequences $w_0, w_1, w_2, \cdots \in \Sigma^*$ and $L_{j_1}, L_{j_2}, \cdots \in \mathcal{L}$ such that for any $i \geq 1$,

$$\{ w_0, w_1, \cdots, w_{i-1} \} \subseteq L_{j_i} \text{ but } w_i \notin L_{j_i}.$$  

A class $\mathcal{L}$ is said to have \textit{finite elasticity}, if $\mathcal{L}$ does not have infinite elasticity.

\textbf{Theorem 2 (Wright [16])}. If a class $\mathcal{L}$ has finite elasticity, then there is an effective procedure which on input $i$ enumerates a finite tell-tale set $S$ of $L_i$ within $\mathcal{L}$, and thus $\mathcal{L}$ is inferable in the limit from positive examples.
Definition 10 (Angluin [2]). A class \( \mathcal{L} \) is said to have finite thickness, if for any nonempty finite set \( S \subseteq \Sigma^* \), \( \forall \{L_i \in \mathcal{L} \mid S \subseteq L_i\} \) is finite.

As easily seen, a class \( \mathcal{L} \) has finite thickness, if and only if for any \( w \in \Sigma^* \), \( \forall \{L_i \in \mathcal{L} \mid w \in L_i\} \) is finite.

Theorem 3 (Wright [16], Angluin [2]). If a class \( \mathcal{L} \) has finite thickness, then \( \mathcal{L} \) has finite elasticity, and thus \( \mathcal{L} \) is inferable in the limit from positive examples.

Definition 11 (Sato [14]). A class \( \mathcal{L} \) is said to satisfy MEF-condition, if for any nonempty finite set \( S \subseteq \Sigma^* \) and for any \( L_i \in \mathcal{L} \) with \( S \subseteq L_i \), there is an \( L_j \in \text{MIN}(S, \mathcal{L}) \) such that \( L_j \subseteq L_i \).

A class \( \mathcal{L} \) is said to satisfy MFF-condition, if for any nonempty finite set \( S \subseteq \Sigma^* \), \( \forall \text{MIN}(S, \mathcal{L}) \) is finite.

A class \( \mathcal{L} \) is said to have M-finite thickness, if \( \mathcal{L} \) satisfies both MEF-condition and MFF-condition.

We note that, as easily seen, if a class has finite thickness, then the class has M-finite thickness.

Theorem 4 (Sato [14]). If a class \( \mathcal{L} \) has M-finite thickness and every language in \( \mathcal{L} \) has its finite tell-tale set within \( \mathcal{L} \), then \( \mathcal{L} \) is inferable in the limit from positive examples.


Theorem 5 (Mukouchi [11]). If a class \( \mathcal{L} \) has M-finite thickness and finite elasticity, then \( \mathcal{L} \) is minimally inferable from positive examples.

3 Neighbor-Minimal Inferability

3.1 Some Properties

Theorem 6. Let \( d \) be a distance and let \( k \in \mathbb{N} \).

If a class \( \mathcal{L} \) has finite thickness (resp., finite elasticity or M-finite thickness), then the class \( \mathcal{L}^{(d,k)} \) also has finite thickness (resp., finite elasticity or M-finite thickness).

Proof. (1) Let \( \mathcal{L} \) be a class with finite thickness. Then let \( w \in \Sigma^* \), and let us put \( S = w^{(d,k)} \) and \( \mathcal{F} = \{L \in \mathcal{L} \mid S \cap L \neq \phi\} \). Since we have assumed that \( d \) has finite thickness, we see that both \( \forall S \) and \( \forall \mathcal{F} \) are finite.

Let \( L' \in \mathcal{L}^{(d,k)} \) be a language with \( w \in L' \). Then there is an \( L \in \mathcal{L} \) such that \( L' = L^{(d,k)} \). Since \( w \in L' \), it follows that there is a \( u \in L \) such that \( d(w, u) \leq k \), i.e., \( w \in S \). Therefore \( S \cap L \neq \phi \) and thus \( L \in \mathcal{F} \).

This means that \( \forall \{L' \in \mathcal{L}^{(d,k)} \mid w \in L'\} \subseteq \{L^{(d,k)} \mid L \in \mathcal{F}\} \), and thus \( \forall \{L' \in \mathcal{L}^{(d,k)} \mid w \in L'\} \) is not greater than \( \forall \mathcal{F} \), which is finite.
Therefore $\overline{L}^{(d,k)}$ has finite thickness.

(II) The proof can be done by contraposition. Suppose that the class $\overline{L}^{(d,k)}$ has infinite elasticity, that is, suppose that there are two infinite sequences $w_0, w_1, \ldots \in \Sigma^*$ and $L'_{j_1}, L'_{j_2}, \ldots \in \overline{L}^{(d,k)}$ such that $\{w_0, w_1, \ldots, w_{i-1}\} \subseteq L'_{j_i}$ but $w_i \notin L'_{j_i}$ for any $i \geq 1$.

For each $i \geq 1$, let $L_{j_i} \in \mathcal{L}$ be a language such that $L'_{j_i} = \overline{L}_{j_i}^{(d,k)}$.

Put $S_0 = \overline{w_0}^{(d,k)}$, $I_0 = N$, and $n_0 = 0$. For each $i \in I_0 \setminus \{n_0\}$, since $w_0 \in L'_{j_i}$, it follows by Lemma 1 that $S_0 \cap L_{j_i} \neq \phi$. However, $S_0$ is a finite set, we see that there is at least one $u_0 \in S_0$ such that $u_0$ belongs to infinitely many $L_{j_i}$'s with $i \in I_0 \setminus \{n_0\}$.

We define $u_m$'s and $n_m$'s ($m \geq 1$) recursively as follows: Put $I_m = \{i \in I_{m-1} \setminus \{n_{m-1}\} \mid \{u_0, u_1, \ldots, u_{i-1}\} \subseteq L_{j_i}\}$ and $n_m = \min I_m$. We note that, by the construction, $I_m$ is infinite. Put $S_m = \overline{w_m}^{(d,k)}$. For each $i \in I_m \setminus \{n_m\}$, since $w_m \in L'_{j_i}$, it follows by Lemma 1 that $S_m \cap L_{j_i} \neq \phi$. Since $S_m$ is a finite set, we see that there is at least one $u_m \in S_m$ such that $u_m$ belongs to infinitely many $L_{j_i}$'s with $i \in I_m \setminus \{n_m\}$.

By the construction, for each $i \geq 1$, $\{u_0, u_1, \ldots, u_{i-1}\} \subseteq L_{j_{n_i}}$. Furthermore, for each $i \geq 1$, since $w_i \notin L'_{j_{n_i}}$, we see by Lemma 1 that $S_{n_i} \cap L_{j_{n_i}} = \phi$, and thus $u_i \notin L_{j_{n_i}}$. Therefore $\mathcal{L}$ has infinite elasticity.

(III) Let $\mathcal{L}$ be a class with $M$-finite thickness. Let $S \subseteq \Sigma^*$ be a nonempty finite set. Then let us put $S_0 = \overline{S}^{(d,k)}$ and $S = \{S' \subseteq S_0 \mid S \subseteq \overline{S'}^{(d,k)}\}$. It is easy to see that for any $L \in \mathcal{L}$, if there is an $S' \in S$ such that $S' \subseteq L$, then $S \subseteq \overline{L}^{(d,k)}$.

Claim A: Let $L' \in \overline{L}^{(d,k)}$ with $S \subseteq L'$. Then for any $L \in \mathcal{L}$ with $L' = \overline{L}^{(d,k)}$, there exists an $S' \in S$ such that $S' \subseteq L$.

Proof of the Claim A. Let $L \in \mathcal{L}$ be a language such that $L' = \overline{L}^{(d,k)}$, and put $S' = L \cap S_0$. Then $S' \subseteq L$ holds.

We show that $S' \in S$. Clearly, $S' \subseteq S_0$ holds. Let $w \in S$. Since $S \subseteq \overline{S'}^{(d,k)}$, it follows that $w \in \overline{L}^{(d,k)}$, and thus there is a $u \in L$ such that $d(u, w) \leq k$. By $w \in S$, we see that $u \in S_0$. Therefore $u \in L \cap S_0$, i.e., $u \in S'$, and thus $w \in \overline{S'}^{(d,k)}$. This means that $S \subseteq \overline{S'}^{(d,k)}$. Therefore $S' \in S$.

Let us put $\mathcal{M} = \bigcup_{S' \in S} \text{MIN}(S', \mathcal{L})$. Since $\mathcal{L}$ satisfies MFF-condition, we see that $\mathcal{L}$ is finite.

Claim B: For any $L' \in \text{MIN}(S, \overline{L}^{(d,k)})$, there exists an $L \in \mathcal{M}$ such that $L' = \overline{L}^{(d,k)}$.

Proof of the Claim B. Let $L' \in \text{MIN}(S, \overline{L}^{(d,k)})$ and let $L_1 \in \mathcal{L}$ with $L' = \overline{L}^{(d,k)}$. Then, by Claim A, there is an $S' \in S$ such that $S' \subseteq L_1$.

Since $S' \subseteq L_1$ and $\mathcal{L}$ satisfies MEF-condition, we see that there is an $L \in \text{MIN}(S', \mathcal{L})$ such that $L \subseteq L_1$. By $S' \in S$, it follows that $L \in \mathcal{M}$ and $S \subseteq \overline{L}^{(d,k)}$.
\( L_1^{(d,k)} = L' \). Since \( L' \in \text{MIN}(S, L^{(d,k)}) \), it turns out that \( L_1^{(d,k)} = L_1^{(d,k)} \). That is, \( L' = L_1^{(d,k)} \) for \( L \in \mathcal{M} \).

This claim means that \( \text{MIN}(S, \mathcal{L}^{(d,k)}) \subseteq \{ L^{(d,k)} \mid L \in \mathcal{M} \} \), and thus \( z \in \text{MIN}(S, \mathcal{L}^{(d,k)}) \) is not greater than \( z \mathcal{M} \), which is finite. Thus \( \mathcal{L}^{(d,k)} \) satisfies MFF-condition.

**Claim C:** The class \( \mathcal{L}^{(d,k)} \) satisfies MEF-condition, that is, for any \( L' \in \mathcal{L}^{(d,k)} \), if \( S \subseteq L' \), then there exists an \( L'' \in \text{MIN}(S, \mathcal{L}^{(d,k)}) \) such that \( L'' \subseteq L' \).

**Proof of the Claim C.** Let \( L' \in \mathcal{L}^{(d,k)} \) with \( S \subseteq L' \). Let us consider the class \( \mathcal{N} = \{ L_1^{(d,k)} \subseteq L' \mid L \in \mathcal{M} \} \). As noted above, \( z \mathcal{M} \) is finite, and so is \( z \mathcal{N} \).

Firstly, we show that \( \mathcal{N} \) is nonempty. Let \( L_1 \in \mathcal{L} \) be a language such that \( L' = L_1^{(d,k)} \). Since \( S \subseteq L' = L_1^{(d,k)} \), it follows by Claim A that there is an \( S_1 \subseteq S \) such that \( S_1 \subseteq L_1 \). Then there is an \( L_2 \in \text{MIN}(S_1, \mathcal{L}) \) such that \( S_1 \subseteq L_2 \subseteq L_1 \), because \( \mathcal{L} \) satisfies MEF-condition. Thus \( L_2 \in \mathcal{M} \) and \( S \subseteq L_2^{(d,k)} \subseteq L_1^{(d,k)} = L' \) holds. Hence \( L_2^{(d,k)} \in \mathcal{N} \), and thus \( \mathcal{N} \) is nonempty.

Since \( \mathcal{N} \) is a nonempty finite class of languages, there is a minimal language \( L'' \in \mathcal{N} \) within \( \mathcal{N} \). We show that \( L'' \in \text{MIN}(S, \mathcal{L}^{(d,k)}) \). Since \( L'' \in \mathcal{N} \), we see that \( S \subseteq L'' \). Suppose that there is an \( L''' \in \mathcal{L}^{(d,k)} \) such that \( S \subseteq L''' \subseteq L'' \). Let \( L_3 \in \mathcal{L} \) be a language such that \( L''' = L_3^{(d,k)} \). Since \( S \subseteq L''' \), we see by Claim A that there is an \( S' \subseteq S \) such that \( S' \subseteq L_3 \). Then there is an \( L_4 \in \text{MIN}(S', \mathcal{L}) \) such that \( S' \subseteq L_4 \subseteq L_3 \), because \( \mathcal{L} \) satisfies MEF-condition. Thus \( L_4 \in \mathcal{M} \) and \( S \subseteq L_4^{(d,k)} \subseteq L_3^{(d,k)} = L''' \subseteq L'' \subseteq L' \) holds. This means \( L_4^{(d,k)} \in \mathcal{N} \), which contradicts that \( L'' \) is a minimal language within \( \mathcal{N} \). Therefore there is no \( L''' \in \mathcal{L}^{(d,k)} \) such that \( S \subseteq L''' \subseteq L'' \), and thus \( L'' \in \text{MIN}(S, \mathcal{L}^{(d,k)}) \), which concludes the proof.

Therefore the class \( \mathcal{L}^{(d,k)} \) has M-finite thickness.

**Lemma 2.** If two classes \( \mathcal{L} \) and \( \mathcal{L}' \) have finite thickness (resp., finite elasticity or M-finite thickness), then the class \( \mathcal{L} \cup \mathcal{L}' \) also has finite thickness (resp., finite elasticity or M-finite thickness).

By Theorem 6 and Lemma 2, we have the following theorem:

**Theorem 7.** Let \( d \) be a distance and let \( k \in N \).

If a class \( \mathcal{L} \) has finite thickness (resp., finite elasticity or M-finite thickness), then the class \( \mathcal{L}^{(d,k)} \) also has finite thickness (resp., finite elasticity or M-finite thickness).

For a class \( \mathcal{L} = \{ L_i \}_{i \in N} \) and for \( n \geq 1 \), let us put

\[
\mathcal{L}^{(d,k)} = \{ L_{i_1} \cup \cdots \cup L_{i_n} \mid i_1, \ldots, i_n \in N \text{ and } L_{i_1}, \ldots, L_{i_n} \in \mathcal{L} \}.
\]

By assuming a computable bijective coding from \( N^n \) to \( N \), the new class above becomes an indexed family of recursive languages.
Theorem 8 (Wright [16]). Let \( n \in N \).
If a class \( \mathcal{L} \) has finite elasticity, then \( \mathcal{L}^{\leq n} \) also has finite elasticity.

Theorem 9 (Sato [14]). Let \( n \in N \).
If a class \( \mathcal{L} \) has \( M \)-finite thickness, then \( \mathcal{L}^{\leq n} \) also has \( M \)-finite thickness.

By Theorems 8, 9 and 6, and Lemma 2, we have the following theorem:

Theorem 10. Let \( d \) be a distance, let \( k \in N \) and let \( n \in N \).
If a class \( \mathcal{L} \) has finite elasticity (resp., \( M \)-finite thickness), then \( \mathcal{L}^{(d, \leq k)} \) also has finite elasticity (resp., \( M \)-finite thickness).

3.2 \( k \)-Neighbor-Minimal Inferability

The following theorem is a direct consequence of Theorems 5, 7 and 10:

Theorem 11. Let \( d \) be a distance, let \( k \in N \) and let \( n \in N \).
If a class \( \mathcal{L} \) has \( M \)-finite thickness and finite elasticity, then \( \mathcal{L} \) and \( \mathcal{L}^{\leq n} \) are weak \( k \)-neighbor-minimally inferable w.r.t. \( d \) from positive examples.

For a language \( L \subseteq \Sigma^* \) and for \( n \in N \), we put \( L^{\leq n} = \{ w \in L \mid |w| \leq n \} \).
We note that for two recursive languages \( L, L' \subseteq \Sigma^* \) and for \( n \in N \), whether \( L^{\leq n} \subseteq L'^{\leq n} \) or not is recursively decidable.

Lemma 3. Let \( L, L' \subseteq \Sigma^* \) be two languages.
(a) If \( L \subsetneq L' \), then there is an integer \( m \in N \) such that for any \( n \geq m \), \( L^{\leq n} \subsetneq L'^{\leq n} \).
(b) If \( L \not\subsetneq L' \), then there is an integer \( m \in N \) such that for any \( n \geq m \), \( L^{\leq n} \not\subsetneq L'^{\leq n} \).

Lemma 4. Let \( \mathcal{L} \) be a class with finite thickness.
For a nonempty language \( L \subseteq \Sigma^* \), there is an integer \( m \in N \) such that for any \( n \geq m \) and for any \( L' \in \mathcal{L} \) with \( L \not\subsetneq L' \), \( L^{\leq n} \not\subsetneq L'^{\leq n} \).

Proof. Let \( L \subseteq \Sigma^* \) be a nonempty language and let us put \( \mathcal{L}' = \{ L' \in \mathcal{L} \mid L \not\subsetneq L' \} \).
Since \( \mathcal{L} \) has finite thickness, we see that \( \mathcal{L}' \) is finite. Let us put \( \mathcal{L}' = \{ L'_1, \ldots, L'_t \} \).

For each \( i \) with \( 1 \leq i \leq t \), since \( L \not\subsetneq L'_i \), we see by Lemma 3 that there is an \( m_i \in N \) such that for any \( n \geq m_i \), \( L^{\leq n} \not\subsetneq L'^{\leq n}_i \). Put \( m = \max\{ m_1, \ldots, m_t \} \).
Then the lemma holds.

Theorem 12. Let \( d \) be a distance and let \( k \in N \).
If a class \( \mathcal{L} \) has finite thickness, then \( \mathcal{L} \) is \( k \)-neighbor-minimally inferable w.r.t. \( d \) from positive examples.
Procedure HLM $M$;
begin
let $T_0 := \phi$ and let $n := 0$;
repeat
let $n := n + 1$;
read the next example $w$;
let $T_n := T_{n-1} \cup \{w\}$;
let $I_n := \{(i, m) \mid 0 \leq i \leq n, 0 \leq m \leq k, T_n \subseteq \mathcal{L}_i^{(d,m)}\}$;
let $M_n := \{(i, m) \in I_n \mid \forall (i', m') \in I_n, \mathcal{L}_{i'}^{(d,m')} \not\subseteq \mathcal{L}_i^{(d,m)}\}$;
if $M_n \neq \phi$ then
let $m_n := \min\{m \mid (i, m) \in M_n\}$ and let $i_n := \min\{i \mid (i, m) \in M_n\}$;
else
let $m_n := 0$ and let $i_n := n$;
output $(i_n, m_n)$;
forever;
end.

Fig. 1. An HLM which $k$-neighbor-minimally infers a class w.r.t. $d$ from positive examples.

Proof. Let $\mathcal{L} = \{L_i\}_{i \in N}$ be a class with finite thickness. Then, by Theorem 7, we see that $\mathcal{L}^{(d,\leq k)}$ has finite thickness. Let us consider the procedure in Figure 1.

Assume that we feed a positive presentation $\sigma$ of a nonempty language $L_{\text{target}} \subseteq \Sigma^*$ to the procedure.

Let $I_\infty = \{(i, m) \mid i \in N, 0 \leq m \leq k, L_{\text{target}} \subseteq \mathcal{L}_i^{(d,m)}\}$ and let $M_\infty = \{(i, m) \in I_\infty \mid \forall (i', m') \in I_\infty, \mathcal{L}_{i'}^{(d,m')} \not\subseteq \mathcal{L}_i^{(d,m)}\}$. We note that $\text{MIN}(L_{\text{target}}, \mathcal{L}_i^{(d\leq k)}) = \{(L_i^{(d,m)}) \mid (i, m) \in M_\infty\}$.

Since $\mathcal{L}^{(d,\leq k)}$ has finite thickness, we see that there is a finite subset $T_{\text{core}}$ of $L_{\text{target}}$ such that $\text{MIN}(T_{\text{core}}, \mathcal{L}_i^{(d,\leq k)}) = \text{MIN}(L_{\text{target}}, \mathcal{L}_i^{(d,\leq k)})$. Since $\sigma$ is a positive presentation and $T_{\text{core}}$ is a finite subset of $L_{\text{target}}$, it follows that there is an $n_{\text{core}} \in N$ such that for any $n \geq n_{\text{core}}$, $T_{\text{core}} \subseteq T_n$.

Furthermore, since $\mathcal{L}_i^{(d,\leq k)}$ has finite thickness, $\text{MIN}(L_{\text{target}}, \mathcal{L}_i^{(d,\leq k)})$ consists of finitely many languages, and let us put $\text{MIN}(L_{\text{target}}, \mathcal{L}_i^{(d,\leq k)}) = \{L_1', \ldots, L_t'\}$. For each $i$ with $1 \leq i \leq t$, $L_i' \in \text{MIN}(L_{\text{target}}, \mathcal{L}_i^{(d,\leq k)})$, and thus for any $j$ with $1 \leq j \leq t$, $L_i' \not\subseteq L_j'$. Therefore, by Lemma 3, we can take $n_{i,j}$'s ($1 \leq i, j \leq t$) such that for any $n \geq n_{i,j}$, $L_i^{(d\leq k)} \not\subseteq L_j^{(d\leq k)}$. Let us put $n_A = \max\{n_{i,j} \mid 1 \leq i, j \leq t\}$.

Claim A: Let $L \in \text{MIN}(L_{\text{target}}, \mathcal{L}_i^{(d,\leq k)})$. For any $L' \in \mathcal{L}_i^{(d,\leq k)}$, if $T_{\text{core}} \subseteq L'$, then for any $n \geq n_A$, $L_i^{(d\leq n)} \not\subseteq L_j^{(d\leq n)}$.

Proof of the Claim A. Let $L' \in \mathcal{L}_i^{(d,\leq k)}$ with $T_{\text{core}} \subseteq L'$.
(I) In case of $L' \notin \text{MIN}(L_{\text{目标}}, \mathcal{L}^{(d,\leq k)})$. The claim is clear from the definition of $n_{A}$.

(II) In case of $L' \notin \text{MIN}(L_{\text{目标}}, \mathcal{L}^{(d,\leq k)})$. Since $\mathcal{L}^{(d,\leq k)}$ has finite thickness, there is an $L'' \in \text{MIN}(L_{\text{目标}}, \mathcal{L}^{(d,\leq k)})$ such that $L'' \subseteq L'$. Thus, for any $n \in N$, $L'' \subseteq L'$. By the definition of $n_{A}$, for any $n \geq n_{A}$, $L'' \not\subseteq L$. Therefore, for any $n \geq n_{A}$, $L'' \not\subseteq L$.

By (I) and (II), the claim holds.

**Claim B:** For any $(i, m) \in M_{\infty}$, there exists an $n_{i,m}^{B} \in N$ such that for any $n \geq n_{i,m}^{B}$, $(i, m) \in M_{n}$.

**Proof of the Claim B.** Let $(i, m) \in M_{\infty}$. Since $\mathcal{L}_{i}^{(d,m)} \in \text{MIN}(L_{\text{目标}}, \mathcal{L}^{(d,\leq k)})$, it follows that for any $n \in N$, $T_{n} \subseteq L_{\text{目标}} \subset \mathcal{L}_{i}^{(d,m)}$. Therefore, for any $n \geq i$, $(i, m) \in I_{n}$.

For any $(i', m') \in I_{n}$ with $n \geq n_{\text{core}}$, $T_{\text{core}} \subseteq \mathcal{L}_{i'}^{(d,m')}$, and thus by Claim A we see that for any $n' \geq n_{A}$, $(L_{i'}^{(d,m')}) \not\subseteq n' \not\subseteq L_{i}^{(d,m)}$. Therefore, by putting $n_{i,m}^{B} = \max\{i, n_{\text{core}}, n_{A}\}$, the claim holds.

**Claim C:** Let $M = \{(i, m) \mid i \in N, 0 \leq m \leq k\}$. There exists an $n_{C} \in N$ such that for any $n \geq n_{C}$ and for any $(i, m) \in M \setminus M_{\infty}$, $(i, m) \notin M_{n}$.

**Proof of the Claim C.** For each $n \in N$, we put $L_{n} = \{L_{i}^{(d,m)} \mid (i, m) \in M_{n}\}$. Let $(i, m) \in M \setminus M_{\infty}$.

(I) In case of $L_{\text{目标}} \not\subseteq L_{i}^{(d,m)}$. By the definition of $T_{\text{core}}$, we see that $T_{\text{core}} \subseteq L_{i}^{(d,m)}$. Therefore, for any $n \geq n_{\text{core}}$, $T_{n} \subseteq L_{i}^{(d,m)}$ and thus $(i, m) \notin M_{n}$.

(II) In case of $L_{\text{目标}} \subseteq L_{i}^{(d,m)}$. By Claim B, we see that there is an $n_{\text{min}} \in N$ such that for any $n \geq n_{\text{min}}$, $L_{1}^{(d)} \subseteq L_{n}$. For each $j$ with $1 \leq j \leq t$, by Lemma 4, there is an $n_{j} \in N$ such that for any $L' \in \mathcal{L}$ with $L' \subseteq L''$, $L'' \subseteq L' \subseteq L''$. Let us put $n_{\text{min}} = \max\{n_{\text{min}}, n_{1}, \ldots, n_{t}\}$.

Since $(i, m) \notin M_{\infty}$ and that $\mathcal{L}^{(d,\leq k)}$ has finite thickness, there is an $L_{j}^{(d,m)} \in \text{MIN}(L_{\text{目标}}, \mathcal{L}^{(d,\leq k)})$ such that $L_{j}^{(d,m)} \not\subseteq L_{i}^{(d,m)}$. Thus, for any $n \geq n_{\text{inl}}$, there is an $L' \in L_{n}$ such that $L' \subseteq L' \subseteq L' \subseteq L'$. Let $n_{C} = \max\{n_{\text{core}}, n_{\text{inl}}\}$. Then, by (I) and (II), the claim holds.

Let $m_{\infty} = \min\{m \mid (i, m) \in M_{\infty}\}$ and let $i_{\infty} = \min\{i \mid (i, m_{\infty}) \in M_{\infty}\}$. By Claims B and C, we see that the procedure in Figure 1 converges to $(i_{\infty}, m_{\infty})$ for $\sigma$.

4 **Pattern Languages and Their Union**

In the present section, we consider the class $\mathcal{P\Sigma}$ of pattern languages introduced by Angluin [1].

Let $X$ be a set of variable symbols. A pattern is a nonnull string of constant symbols in $\Sigma$ and variable symbols in $X$. The pattern language $L(p)$ generated
by a pattern \( p \) is the set of all constant strings obtained by substituting nonnull constant strings for the variable symbols in \( p \).

For example, let \( \Sigma = \{a, b, c\} \) and let \( X = \{x, y, \cdots\} \). Then \( p = axbycy \) is a pattern, and the set \( \{aabac, abbbc, acebc, aaabaac, aabbabc, \cdots\} \) is the pattern language of \( p \).

Since two patterns that are identical except for renaming of variable symbols generate the same pattern language, we do not distinguish one from the other. The set of all patterns is recursively enumerable and whether \( w \in L(p) \) or not is recursively decidable for a constant string \( w \) and for a pattern \( p \). Therefore we can consider the class \( \mathcal{P} \mathcal{A} \mathcal{T} \) of pattern languages as an indexed family of recursive languages, where the pattern itself is considered as an index.

For a constant string \( w \in \Sigma^* \) and for a pattern \( p \), if \( w \in L(p) \), then \( p \) is not longer than \( w \). Since the number of patterns shorter than a fixed length is finite, the class \( \mathcal{P} \mathcal{A} \mathcal{T} \) has finite thickness (cf. Angluin [1]). Therefore, by Theorems 3 and 12, we have the following theorems:

\textbf{Theorem 13 (Angluin [1])}. The class \( \mathcal{P} \mathcal{A} \mathcal{T} \) is inferable in the limit from positive examples.

\textbf{Theorem 14.} Let \( d \) be a distance and let \( k \in N \).

The class \( \mathcal{P} \mathcal{A} \mathcal{T} \) is \( k \)-neighbor-minimally inferable w.r.t. \( d \) from positive examples.

Furthermore, Wright [16] showed that the class \( \mathcal{P} \mathcal{A} \mathcal{T}^{[\leq n]} \) has finite elasticity. Therefore, by Theorems 2 and 11, we have the following theorems:

\textbf{Theorem 15 (Wright [16])}. Let \( n \in N \).

The class \( \mathcal{P} \mathcal{A} \mathcal{T}^{[\leq n]} \) is inferable in the limit from positive examples.

\textbf{Theorem 16.} Let \( d \) be a distance, \( k \in N \) and let \( n \in N \).

The class \( \mathcal{P} \mathcal{A} \mathcal{T}^{[\leq n]} \) is weak \( k \)-neighbor-minimally inferable w.r.t. \( d \) from positive examples.

\section{5 Concluding Remarks}

We have introduced a notion of a recursively generateable distance and defined a \( k \)-neighbor closure of a language by taking incorrectness into consideration. Then we have formalized \( k \)-neighbor-minimal inferability and gave some sufficient conditions.

We have shown that the class with \( M \)-finite thickness and finite elasticity is weak \( k \)-neighbor-minimally inferable from positive examples (Theorem 11). It is known that there are many rich classes that have \( M \)-finite thickness and finite elasticity (cf. Sato [14] and Mukouchi [11]). On the other hand, we could show that the class with finite thickness is \( k \)-neighbor-minimally inferable from positive examples (Theorem 12). This difficulty comes from necessity of eliminating, in a finite time, the possibility that infinitely many non-minimal languages may
behave as if they are minimal languages (cf. Claim C in the proof of Theorem 12).

As a future work, we can consider refutable inference from complete examples, some of which are incorrect. As another future investigation, it is worth to develop an efficient learning algorithm for the class of e.g. regular pattern languages in our framework.

References

Application of Multivariate Maxwellian Mixture Model to Plasma Velocity Distribution Function

Genta Ueno¹, Nagatomo Nakamura², Tomoyuki Higuchi³, Takashi Tsuchiya³, Shinobu Machida¹, and Tohru Araki¹

¹ Department of Geophysics, Graduate School of Science, Kyoto University, Kyoto 606-8502, Japan
genta@kugi.kyoto-u.ac.jp
² Sapporo Gakuin University, Hokkaido 069-8555, Japan
³ The Institute of Statistical Mathematics, Tokyo 106-8569, Japan

Abstract. Recent space plasma observations provide us three-dimensional velocity distributions which are found to have multiple peaks. We propose a method for analyzing such a velocity distribution via a multivariate Maxwellian mixture model whose each component represents each of the multiple peaks. The parameters of the model are determined through the EM algorithm. For an auto judgment of preferable number of components of the mixture model, we introduce a method of examining the number of extremum of the resulting mixture model. We show applications of our method to observations in the plasma sheet boundary layer and in the central plasma sheet in the terrestrial magnetosphere.

1 Introduction

From direct satellite observations of space plasma, we have obtained macroscopic physical quantities by calculating velocity moments of particle velocity distribution functions (e.g. number density, bulk velocity and temperature). This fluid description assumes that the plasma is in a state of local thermal equilibrium. Under that assumption, a distribution function of particle velocity is given as a normal distribution which is called Maxwellian distribution in a field of the plasma physics. The (multivariate) Maxwellian distribution is given by

\[
g (v|V, T) = \left( \frac{m}{2\pi} \right)^{3/2} \frac{1}{\sqrt{|T|}} \exp \left[ -\frac{m}{2} (v - V)^T T^{-1} (v - V) \right],
\]

where \( m \) is the mass of the particle, \( V \) is the bulk velocity vector, \( T \) is the temperature matrix, and superscript \( T \) denotes transpose.

Observational techniques progressed notably today, and made it possible to measure detailed shape of distribution function in the three-dimensional velocity space. These observations revealed that there are many cases where space plasmas is not in thermal equilibrium. Their velocity distributions are not a single Maxwellian but consist of multiple peaks. This is because since the space plasma such as the solar wind is basically collisionless with large mean-free-path (about

© Springer-Verlag Berlin Heidelberg 2000
1 Astronomical Unit: the distance between the Sun and the Earth). Therefore, we have to be careful that they may give the same velocity moments even if the shapes of distribution functions are different. For instance, when the plasma with two beam components whose velocity vectors are sunward and anti-sunward, respectively, and the numbers of particles of each component are the same, the bulk velocity becomes zero from calculation. On the other hand, when a stagnant plasma is observed, the bulk velocity also becomes zero. When we deal with the two-beam distribution, we should separate it into two beams and calculate the velocity moments for each beam. It has been difficult, however, to evaluate the shape of distribution function, especially when more than one components partially overlap each other. Furthermore, it produces a serious problem when we treat many multi-component cases statistically.

In this paper, we construct a method of representing three-dimensional distribution function by a multivariate Maxwellian mixture model in which the parameter values are obtained by the EM (Expectation-Maximization) algorithm [2][4]. With this method, we can express the shape of the function and find a possible way to conduct a statistical analysis. The organization of this paper is the following. In Sect. 2, we describe the data of plasma velocity distribution functions. A fitting method with the multivariate Maxwellian mixture model is presented in Sect. 3, followed by the consideration to determine the preferable number of components in the mixture model in Sect. 4. Two applications are demonstrated in Sect. 5. In Sect. 6, We discuss a problem of model selection.

2 Data

2.1 Instrumentation

We used the data obtained by the Low Energy Particle Energy-per-charge Analyzer (LEP-EA) onboard the Geotail spacecraft. LEP-EA consists of two nested sets of quadrupole electrostatic analyzers to measure three-dimensional velocity distributions of electrons (with EA-e) and ions (with EA-i) simultaneously and separately. In the present observation, EA-i covers the energy range from 32 eV/q to 39 keV/q divided into 32 bins, in which 24 bins are equally spaced on a logarithmic scale in energies higher than 630 eV/q and have width of ±9.4 % of the center energy, while the lower-energy 8 bins are spaced linearly with width of ±40 eV/q (±20 eV/q for the lowest energy bin). The full energy range is swept in a time which is 1/16 of the spacecraft spin period (about 3 seconds). The field of view is fan-shaped with ~ 10° × 145°, in which the longer dimension is perpendicular to the spin plane and divided into seven directions centered at elevation angles of 0°, ±22.5°, ±45° and ±67.5° with each width of 6–10° (wider for higher elevation angles). Thus, count rate data of dimension 32 (energy bin) × 7 (elevation angle) × 16 (azimuthal sector) are generated in one spin period. These classes in the velocity space are shown in Fig. 1.

The complete three-dimensional velocity distributions are obtained in a period of four spins (about 12 seconds) owing to the telemetry constraints; the
count data are accumulated during the four-spin period. A more detailed description of LEP instrumentation is given in [3].

Assume that an electrostatic analyzer detected the particle count $C(v_{pqr})$ [#] in a sampling time $\tau$ [s], where $v_{pqr}$ [m/s] is the particle velocity. Subscriptions $p$, $q$ and $r$ take integers and denote the position in the three dimensional velocity space. LEP-EA, for instance, has $p = 1, 2, \cdots, 32; q = 1, 2, \cdots, 7; r = 1, 2, \cdots, 16$, where we choose $p$, $q$ and $r$ as indicators of energy bin, elevation angle and azimuthal sector, respectively. Thus we obtain the total number of the particle count $N$ [#]:

$$N = \sum_{p,q,r} C(v_{pqr}).$$

### 2.2 Density Function

Under the assumption that the incident differential particle flux is uniform within the energy and angular responses of the analyzer, the velocity distribution function $f_0(v_{pqr})$ [m$^3$/s$^2$] is given by

$$f_0(v_{pqr}) = 2 \times 10^4 \frac{1}{\tau \varepsilon G} \frac{C(v_{pqr})}{(v_{pqr}^Tv_{pqr})^2},$$

where $m$ [kg] and $q$ [C] are the mass and the charge of the particle, $\varepsilon$ is the detection efficiency, $G$ [cm$^2$ sr eV/eV] is the geometrical-factor and superscript $T$ denotes transpose. LEP-EA has $\varepsilon$ and $G$ as functions of elevation angle: $\varepsilon = \varepsilon_0, G = G_0$. Integrating $f_0(v_{pqr})$ over the velocity space, we obtain the number density $n$ [#/m$^3$]:

$$n = \sum_{p,q,r} f_0(v_{pqr}) \, dv_{pqr}$$

In this paper, we treat the probability function:

$$f(v_{pqr}) = \frac{f_0(v_{pqr}) \, dv_{pqr}}{n},$$
so that
\[ \sum_{p,q,r} f(v_{pqr}) = 1. \]  
(6)

Since observed distribution is a function of discrete variables, it is necessary to consider the mixture of probability function, but we approximate it by the mixture of Maxwellian that is a probability density function.

3 Maxwellian Mixture Model

We will fit the probability function (5) by the mixture model composed of the sum of \( s \) multivariate Maxwellian distributions:
\[ f(v_{pqr}) \simeq \sum_{i=1}^{s} n_i \tilde{g}_i(v_{pqr}|V_i, T_i), \]  
(7)
where \( n_i \) is the mixing proportion of Maxwellians (\( \sum_{i=1}^{s} n_i = 1, 0 < n_i < 1 \)). Each Maxwellian \( \tilde{g}_i \) is written as
\[ \tilde{g}_i(v_{pqr}|V_i, T_i) \]  
(8)
\[ = \left( \frac{m}{2\pi} \right)^{3/2} \frac{1}{\sqrt{|T_i|}} \exp \left[ -\frac{m}{2} (v_{pqr} - V_i)^T T_i^{-1} (v_{pqr} - V_i) \right], \]
where \( V_i \) is the bulk velocity vector and \( T_i \) is the temperature matrix of \( i \)-th Maxwellian.

On the condition above, we consider the log-likelihood:
\[ l(\theta) = N \sum_{p,q,r} f(v_{pqr}) \log \sum_{i=1}^{s} n_i \tilde{g}_i(v_{pqr}|V_i, T_i) \]  
(9)
in order to obtain the maximum likelihood estimator of each parameter, where
\[ \theta = (n_1, n_2, \cdots, n_s, V_1, V_2, \cdots, V_s, T_1, T_2, \cdots, T_s) \]
(10)
means the all unknown parameters.

Partially differentiate (9) with respect to \( V_i, T_i^{-1} \) and put them equal to zero, maximum likelihood estimators (denoted by \( \hat{\cdot} \)) of the mixing proportion, the bulk velocity vector and the temperature matrix of each Maxwellian are given by:
\[ \hat{n}_i = \sum_{p,q,r} f(v_{pqr}) \hat{P}_i(v_{pqr}), \]  
(11)
\[ \hat{V}_i = \frac{1}{\hat{n}_i} \sum_{p,q,r} f(v_{pqr}) \hat{P}_i(v_{pqr}) v_{pqr}, \]  
(12)
\[ \hat{T}_i = \frac{1}{\hat{n}_i} \sum_{p,q,r} f(v_{pqr}) \hat{P}_i(v_{pqr}) \frac{1}{2} m (v_{pqr} - \hat{V}_i) (v_{pqr} - \hat{V}_i)^T, \]  
(13)
where

\[
\hat{P}_t(v_{pvr}) = \frac{\hat{n}_i g_i(v_{pvr}| V_i, T_i)}{\sum_{j=1}^s \hat{n}_j g_j(v_{pvr}| V_j, T_j)}
\]  

(14)

is an estimated posterior probability.

On the basis of these equations, we will estimate the unknown parameters by the EM algorithm \[2\][4]. In the following procedure, \( t \) denotes an iteration counter of the EM algorithm. Suppose that superscript \((t)\) denotes the current values of the parameters after \( t \) cycles of the algorithm for \( t = 0, 1, 2, \cdots \).

**Setting Initial Value:** \( t = 0 \). Set the initial values of the parameters of each Maxwellian and posterior probability. At first, we classify the data in \( s \) groups \((G_i; i = 1, 2, \cdots, s)\) using the k-means algorithm, and set the initial value of the posterior probability:

\[
P_i^{(0)}(v_{pvr}) = \begin{cases} 1 & (v_{pvr} \in G_i) \\ 0 & (v_{pvr} \notin G_i) \end{cases}
\]  

(15)

where \( i = 1, 2, \cdots, s \). From the initial value of the posterior probability, we calculate \( \hat{n}_i^{(0)}, \ V_i^{(0)}, \ T_i^{(0)} \) by (11), (12) and (13).

**Parameter Estimation by the EM Algorithm:** \( t \geq 1 \).

**E-step (Expectation step).** At E-step of the \( t \)-th iteration \( (t \geq 1) \), we compute the posterior probability by Eq. (14). At the same time, we estimate the mixing proportion \( n_i^{(t)} \) by Eq. (11).

**M-step (Maximization step).** At M-step of the \( t \)-th iteration, we choose the values of the bulk velocity vector and temperature matrix as maximum likelihood estimators by Eqs. (12) and (13).

**Judgment of Convergence.** We finish the iteration if the following convergence condition is satisfied:

\[
\left| t^{(t)} \left( \hat{\theta}^{(t)} \right) - t^{(t-1)} \left( \hat{\theta}^{(t-1)} \right) \right| < \epsilon \quad \text{and} \quad \left\| \hat{\theta}^{(t)} - \hat{\theta}^{(t-1)} \right\| < \delta,
\]  

(16)

where \( \epsilon \) and \( \delta \) are sufficiently small positive number. If the above condition is not satisfied, return to the E-step with replacing \( t \) by \( t + 1 \).
4 Preferable Number of Components

When we fit a distribution function by the Maxwellian mixture model, we should examine that this fitting is reasonable. Fig. 2 shows two examples of applying a single Maxwellian and a two-Maxwellian mixture model for different types of observed distribution functions. In Fig. 2(a), the upper plot shows the observed bimodal distribution, and the lower two plots show the fitting result of a single Maxwellian at the left-hand and a two-Maxwellian mixture at the right-hand, respectively. We find that the two-Maxwellian fitting well presents the bimodal observation in this example. Fig. 2(b) shows an example of unimodal observation and the fitting results; we should not approve the two-Maxwellian fitting of the observation.

![Fig. 2. Examples of applying of the single Maxwellian and the two-Maxwellian mixture model fitting for different type of observations: (a) bimodal distribution and (b) unimodal distribution](image)

Here, we introduce a method of judging which of models, i.e., a single component model or a two-component mixture model is preferable for each observation. We adopt the following principle. If the resulting two-component mixture model has two peaks, the observation would also have two peaks. Hence, we conclude that the two-component mixture model is reasonable to use. On the other hand, if the resulting two-component mixture model has only one peak, the observation would be a unimodal distribution: we should use a usual single Maxwellian fitting.

4.1 Diagnostics of Fitting

To judge whether the fitting result is reasonable, therefore, we count the number of peaks of the resulting fitting model. Furthermore, to count the number of peaks, we count the number of extremum of the model. Let us consider when we
fit some data \( f(v) \) by a two-Maxwellian mixture model and the fitting result is computed as

\[
f(v) \simeq n_1 g_1(v|V_1, T_1) + n_2 g_2(v|V_2, T_2).
\]  

(17)

To count the number of peaks, we need to count the number of \( v \) that satisfy

\[
\frac{d}{dv} \left( n_1 g_1(v|V_1, T_1) + n_2 g_2(v|V_2, T_2) \right) = 0.
\]  

(18)

It is difficult, however, to treat the three-dimensional variable \( v \). In the following of this section, we derive one set of simultaneous equations of one-dimensional variables, whose number of solutions are equivalent to the number of \( v \) that satisfy Eq. (18).

Without loss of generality, we can shift the origin of the velocity space such that the bulk velocity of the component \( i \) \( (V_1) \) vanishes:

\[
f(v') \simeq n_1 g_1(v'|0, T_1) + n_2 g_2(v'|V', T_2),
\]  

(19)

where we put \( v' = v - V_1 \), \( V' = V_2 - V_1 \). Since we are interested in the topological form of the function, we can change the scale and rotate the main axes such that the temperature matrix becomes isotropic. That is, since \( T_1 \) is a symmetric matrix, we can define a new matrix \( L \) that satisfies \( LL^T = T_1^{-1} \). As an expression of \( L \), we choose

\[
L = \begin{pmatrix} \sqrt{\lambda_1} x_1 \\ \sqrt{\lambda_2} x_2 \\ \sqrt{\lambda_3} x_3 \end{pmatrix}
\]  

(20)

where \( \lambda_1, \lambda_2, \lambda_3 \) are the eigenvalues of \( T_1^{-1} \), and \( x_1, x_2, x_3 \) are the corresponding eigenvectors whose absolute values are unity. Transforming the coordinate with \( L^T \), that is, putting new vectors \( u \) and \( U \) that satisfy \( u = L^T v' \) and \( U = L^T V' \), then we can express the fitting function (19) as

\[
f(u) \simeq \frac{1}{\sqrt{|T_1^t|}} \left[ n_1 g_1(u|0, I) + n_2 g_2(u|I, L^T T_2 L) \right],
\]  

(21)

where \( I \) is the unit matrix.

Since \( T_2 \) is a symmetric matrix, \( (L^T T_2 L)^{-1} \) is also symmetric, so that we can carry out the orthogonal transformation such that \( (L^T T_2 L)^{-1} \) becomes a diagonal matrix \( M \). That is, when we put \( \mu_1, \mu_2, \mu_3 \) as eigenvalues of \( (L^T T_2 L)^{-1} \) and \( y_1, y_2, y_3 \) as corresponding eigenvectors, and rotate the main axes such that \( u = (y_1, y_2, y_3) w \), then we obtain

\[
f(w) \simeq \frac{1}{\sqrt{|T_1|}} \left[ n_1 g_1(w|0, I) + n_2 g_2(w|W, M^{-1}) \right],
\]  

(22)
where

\[
M = \begin{pmatrix}
\mu_1 & O \\
\mu_2 & \mu_3 \\
O & \mu_3 \\
\end{pmatrix}
\]  

and \( U = (y_1, y_2, y_3) W \).

To obtain the (transformed) velocity \( w \) that gives an extremum, we differentiate Eq. (22) with respect to \( w \). We then obtain the extremum condition for \( w \):

\[
w = \left[ n_{1} g_{1} (w|0, I) + n_{2} g_{2} (w|W, M^{-1}) M \right]^{-1}
\cdot n_{2} g_{2} (w|W, M^{-1}) MW,
\]

that is,

\[
w_\alpha = \frac{\mu_0 W_\alpha}{n_{1} g_{1} (w|0, I) + n_{2} g_{2} (w|W, M^{-1}) + \mu_0}
\]

where \( \alpha = 1, 2, 3 \).

To examine the Eq. (25), we introduce the parameter \( \xi \) defined by

\[
\xi = \frac{n_{1} g_{1} (w|0, I)}{n_{2} g_{2} (w|W, M^{-1})},
\]

so that

\[
w_\alpha = \frac{\mu_0 W_\alpha}{\xi + \mu_0}.
\]

Substituting Eq. (27) into Eq. (26), we obtain an equation with respect to \( \xi \):

\[
\xi = \frac{n_{1} g_{1} (w(\xi)|0, I)}{n_{2} g_{2} (w(\xi)|W, M^{-1})},
\]

A solution of Eq. (28) is given by the node of the line

\[
\eta(\xi) = \xi
\]

and the curve

\[
\eta(\xi) = \frac{n_{1} g_{1} (w(\xi)|0, I)}{n_{2} g_{2} (w(\xi)|W, M^{-1})}
\]

in the \( \xi-\eta \) plane. In this way, formerly three dimensional problem has been reduced to one dimensional and it becomes easier to examine the number of extremum.
4.2 Simulation Study

We determine which of fitting results is preferable by checking the resulting two-mixture fitting. The left-hand panel of Fig. 3(a) shows the fitting result represented by the sum of the two components obtained for the bimodal observation shown in Fig. 2(a). This result has three extrema (two maxima and one saddle point between them) as denoted by three dots. We plot each relation of the simultaneous equations (29) and (30) in the $\xi$-$\eta$ plane in the right-hand panel of the Fig. 3(a). In a practical data processing, we first find three intersections of these two graphs which is equivalent to the number of the extremum of the resulting two-component mixture model, then judge that the two-mixture model is preferable to a single component model.

Similarly, the left-hand panel of Fig. 3(b) shows the resulting two-mixture fitting for unimodal observation shown in Fig. 2(b), and the right-hand panel shows the relation between $\xi$ and $\eta$ based on the simultaneous equations (29) and (30). Different from case (a), the number of the solution of the simultaneous equations is unity and the number of extremum of two mixture model is also unity. We adopt, therefore, the single Maxwellian model which represents usual velocity moments.

Fig. 3. The number of the extremum of the fitted function of two-Maxwellian mixture distribution for different type of observations: (a) bimodal distribution and (b) unimodal distribution. In both (a) and (b), the left-hand plot shows the two-mixture distribution whose extremum is denoted by the dots, and the right-hand plot shows the line (Eq. (29)) and the curve (Eq. (30)) whose intersections are also denoted by the dots.
5 Application

Now we apply the fitting method explained above to the observation of ion distribution function. We first fit by the mixture distribution with fixed components of the number of two, then examine the result, and judge whether the fitting by two components is reasonable or single component is better to fit the observation. In the following part of this section, we will apply our method to two different kinds of observation: one is an example that the fitting by two components is preferable and the other is that one component fitting is better.

(a) 1554:58 - 1555:10 UT on January 14, 1994

(b) 1558:58 - 1559:10 UT on January 14, 1994

Fig. 4. (a) Observation of ion velocity distribution function in the $v_x$-$v_y$ plane in the time interval of 1554:58-1555:10 UT on January 14, 1994 (left), fitting function by the single Maxwellian ($s = 1$, center) and fitting function by the two Maxwellian mixture model ($s = 2$, right). (b) In the time interval of 1558:58-1559:10 UT on the same day

First, let us apply to the ion velocity distribution in the time interval 1554:58-1555:10 on January 14, 1994. The left-hand panel of Fig. 4 (a) shows the observation obtained in the plasma sheet boundary layer. We show the distribution functions sliced by the $v_x$-$v_y$ plane whose value is black-to-white-coded according to the bar at the right of the panel. When we see the $v_x$-$v_y$ slice, we find hot component and cold component whose bulk velocities are about $(v_x, v_y) = (1000 \text{ km/s}, 0 \text{ km/s})$ and $(v_x, v_y) = (-200 \text{ km/s}, -500 \text{ km/s})$, respectively.
When we fit that data with a single Maxwellian, we obtain the estimated parameters shown in the second row of Table 1 (a). With these parameters, we can produce the distribution function shown in the central panel of Fig. 4 (a). This corresponds to what we deal with by the usual velocity moment, and it is hot and has a shifted bulk velocity compared with the observation.

Table 1. Estimated parameters for single Maxwellian and for two-Maxwellian mixture model: (a) in the time interval 1554:58–1555:10 on January 14, 1994, and (b) in the time interval 1558:58–1559:10 on the same day

(a) 1554:58–1555:10 UT on January 14, 1994

<table>
<thead>
<tr>
<th>n (cc)</th>
<th>V_x [km/s]</th>
<th>V_y</th>
<th>V_x</th>
<th>T_x [eV]</th>
<th>T_y</th>
<th>T_90</th>
<th>T_99</th>
<th>T_999</th>
<th>T_9999</th>
<th>T_99999</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.036</td>
<td>522</td>
<td>189</td>
<td>44</td>
<td>5660</td>
<td>518</td>
<td>-86</td>
<td>1901</td>
<td>-29</td>
<td>1074</td>
</tr>
<tr>
<td>1</td>
<td>0.018</td>
<td>1753</td>
<td>66</td>
<td>20</td>
<td>2914</td>
<td>-136</td>
<td>79</td>
<td>3004</td>
<td>78</td>
<td>2100</td>
</tr>
<tr>
<td>2</td>
<td>0.019</td>
<td>-84</td>
<td>-394</td>
<td>60</td>
<td>54</td>
<td>-81</td>
<td>14</td>
<td>603</td>
<td>-80</td>
<td>3100</td>
</tr>
</tbody>
</table>

(b) 1558:58–1559:10 UT on January 14, 1994

<table>
<thead>
<tr>
<th>n (cc)</th>
<th>V_x [km/s]</th>
<th>V_y</th>
<th>V_x</th>
<th>T_x [eV]</th>
<th>T_y</th>
<th>T_90</th>
<th>T_99</th>
<th>T_999</th>
<th>T_9999</th>
<th>T_99999</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.085</td>
<td>139</td>
<td>-312</td>
<td>-87</td>
<td>3609</td>
<td>13</td>
<td>63</td>
<td>3407</td>
<td>-42</td>
<td>3721</td>
</tr>
<tr>
<td>1</td>
<td>0.024</td>
<td>84</td>
<td>-99</td>
<td>-22</td>
<td>6117</td>
<td>116</td>
<td>-178</td>
<td>5183</td>
<td>556</td>
<td>4690</td>
</tr>
<tr>
<td>2</td>
<td>0.041</td>
<td>171</td>
<td>-437</td>
<td>-126</td>
<td>2498</td>
<td>66</td>
<td>39</td>
<td>2011</td>
<td>-532</td>
<td>3150</td>
</tr>
</tbody>
</table>

This problem, however, is easily solved by applying a two-component mixture model. Similarly, we give the estimated parameter in the third and fourth rows of Table 1 (a), and display the calculated distribution function in the right-hand panel of Fig. 4 (a). We found the hot and cold components seen in the observed distribution were reproduced adequately.

For this example, we found that the fitting by two-component is more preferable than by single component by counting the number of the solutions of simultaneous equations (29) and (30). Fig. 5 (a) shows the relation of ξ and η. The two graphs have three intersections, that is, the two-component mixture model has two maxima and one minimum between them. Thus, the two-component fitting is justified, which agrees with our inspection of the observed distribution.

The other example is a distribution function observed in the time interval of 1558:58–1559:10 on the same day. These data were obtained in the central plasma sheet. As can be seen in the left-hand panel of Fig. 4 (b), it is appropriate to think that this consists of single hot component whose bulk velocity is located near the origin of the velocity space. Hence, when we fit the data, we should adopt not two-Maxwellian mixture but single Maxwellian model.

In the central panel of Fig. 4 (b), we show the calculated distribution function with the single component model. Used parameters are shown in the second row of Table 1(b). In this case, this single-component fitting appears to be sufficient. Further, we display the result when used the two-component mixture model. The right-hand panel shows the calculated distribution function with the estimated
parameters shown in the third and fourth rows of Table 1 (b). For this example, we think that this two-component fitting is an over-fitting.

In this case, when we examine the number of solutions of the simultaneous equations for $\xi$ and $\eta$, we found that they have only one solution as shown in Fig. 5 (b). Since the resulting two-component mixture model has no minimal in this example, we will adopt the usual velocity moments obtained by the single component fitting.

6 Discussion

In the course of choosing the preferable number of components, we first fit the data with a two-component mixture model, then examine whether there is a saddle point on the segment between the bulk velocities of the model.

When we select the preferable number of components, that is, when we compare the goodness between the models, it has been known that AIC (Akaike Information Criterion, [1]) is useful. If there are several candidate models, we can find the best one by finding the model with smallest value of AIC defined by

$$AIC = -2\max l(\theta) + 2 \dim \theta,$$

In the present analysis, however, AIC has not given an expected criterion as we explain in the following.

When we apply the mixture model whose component number is 1 to 6 ($s = 1, 2, \ldots, 6$) to the same data in the previous section, the corresponding $\dim \theta$, $\max l(\theta)$, AIC, and BIC (explained below) are summarized in Table 2. The baselines of AIC and BIC are taken to be those of the single Maxwellian fittings. In case (a), AIC decreases with increasing the number of components and has a minimum at the number of components 5. This number, however, is thought to be too large to fit the observation. On the other hand, in case (b), AIC has no minimum in the range of the number 1 to 6, so we have to prepare seven or more
Table 2. Comparison of six Maxwellian mixture models. Asterisks denote the minimum of AIC and BIC.

(a) 1554:58–1555:10 UT on January 14, 1994: \( N = 3156 \)

<table>
<thead>
<tr>
<th>Comp.</th>
<th>( \dim \theta )</th>
<th>( \max \ell (\theta) )</th>
<th>( \text{AIC} - \text{AIC}(1) )</th>
<th>( \text{BIC} - \text{BIC}(1) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9</td>
<td>-72217.82</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>2</td>
<td>19</td>
<td>-68560.85</td>
<td>-7293.94</td>
<td>-7273.65</td>
</tr>
<tr>
<td>3</td>
<td>29</td>
<td>-68286.86</td>
<td>-7821.91</td>
<td>-7781.33</td>
</tr>
<tr>
<td>4</td>
<td>39</td>
<td>-67586.02</td>
<td>-9203.61</td>
<td>-9142.75</td>
</tr>
<tr>
<td>5</td>
<td>49</td>
<td>-67176.28</td>
<td>* -10003.08</td>
<td>* -9921.94</td>
</tr>
<tr>
<td>6</td>
<td>59</td>
<td>-67565.62</td>
<td>-9204.41</td>
<td>-9102.98</td>
</tr>
</tbody>
</table>

(b) 1558:58–1559:10 UT on January 14, 1994: \( N = 4860 \)

<table>
<thead>
<tr>
<th>Comp.</th>
<th>( \dim \theta )</th>
<th>( \max \ell (\theta) )</th>
<th>( \text{AIC} - \text{AIC}(1) )</th>
<th>( \text{BIC} - \text{BIC}(1) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9</td>
<td>-113789.51</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>2</td>
<td>19</td>
<td>-113025.13</td>
<td>-348.77</td>
<td>-326.32</td>
</tr>
<tr>
<td>3</td>
<td>29</td>
<td>-113302.66</td>
<td>-933.70</td>
<td>-888.82</td>
</tr>
<tr>
<td>4</td>
<td>39</td>
<td>-113232.97</td>
<td>-1033.09</td>
<td>-985.76</td>
</tr>
<tr>
<td>5</td>
<td>49</td>
<td>-113195.14</td>
<td>-1108.77</td>
<td>-1018.99</td>
</tr>
<tr>
<td>6</td>
<td>59</td>
<td>-112407.97</td>
<td>* -2663.08</td>
<td>* -2550.86</td>
</tr>
</tbody>
</table>

components for the appropriate fitting. This number of component also seems to be too large.

These results do not agree with our intuition. We think that it is due to the following three reasons. First, adopting the Maxwellian distribution as a component distribution of mixture model is not appropriate, which is notable in the high-energy range. Since the observation has a heavy tail in the high-energy range, it is necessary to have many components for fitting such a tail accurately. One of the heavy tail distributions is \( \kappa \) distribution defined by

\[
g_{\kappa} (v_{pr} | V, T_i, \kappa_i) = \left( \frac{m}{2\pi} \right)^{3/2} \frac{\Gamma(\kappa_i)}{\Gamma(\kappa_i - 3/2)} \frac{1}{\sqrt{|T_i|}} \left[ 1 + \frac{m}{2} (v_{pr} - V_i)^T T_i^{-1} (v_{pr} - V_i) \right]^{-\kappa_i/2} (32)
\]

This converges to Maxwellian distribution in the limit of \( \kappa_i \to \infty \). When we select \( \kappa \) distribution as a component distribution (i.e., \( \kappa \) mixture model), the algorithm in Sect. 3 can work by including the \( \kappa_i \) renewing step. By carrying out a calculation, however, we found that the estimated \( \kappa_i \) value is the order of \( 10^4 \), which means that the distribution is practically the Maxwellian, that is, we failed to lift up the tail of \( \kappa \) distribution. Since \( \kappa \) distribution includes Maxwellian, we could treat the data in more comprehensive way if the fitting by \( \kappa \) mixture model was successfully done. We need some improvement in our algorithm.

The next reason for decreasing AIC for increasing the number of component is that the observation has many data number (\( N = 3156 \) and 4860 for the examples (a) and (b), respectively). Log-likelihood is multiplied by data number.
N as defined in Eq. (9), and has the order of $10^4$–$10^5$ in the two examples. On the other hand, the dimension of free parameters \( \theta \) is the order of $10^0$–$10^1$. AIC is determined practically by log-likelihood and is not affected by the dimension of free parameters as a penalty term.

We then evaluated by BIC instead of AIC. BIC is an information criterion such that posterior probability is maximized and defined as\cite{5}

\[
\text{BIC} = -2 \ell(\theta) + \frac{1}{2} \cdot \dim \theta \cdot \log N. \tag{33}
\]

With BIC, however, we obtained the same result as with AIC (see Table 2).

Finally, the decreasing AIC also depends on higher one count level near the origin of the velocity space, which is seen as a ‘hole’ in the distribution function near the origin. This is a property of electrostatic analyzers, and it means poor resolution of density of velocity distribution near the origin. For instance, let us assume that an ambient plasma distribution \( f_{\text{am}} \) is a simple Maxwellian which has a single peak near the origin of the velocity space. Then an electrostatic analyzer observes this plasma, that is, detects as count \( C^{\text{obs}} \). When the count equivalent to \( f_{\text{am}} \) (calculated by Eq. (3) is less than one, \( C^{\text{obs}} \) becomes zero since \( C^{\text{obs}} \) takes an integer. This omitting occurs especially near the origin of the velocity space, which produce the observed distribution \( f_{\text{am}}^{\text{obs}} \) having no peak near the origin but forming a hole like a caldera. Therefore, multi components that AIC requires is to present the edge of the caldera. We should reconsider the treatment the density near the origin.

In this paper, we only examined the existence of a saddle point between the two bulk velocities in the present case study in which the observed distribution has two peaks whose mutual distance is long enough compared with each temperature matrix. But this procedure is not valid if two bulk velocities is close to each other since the saddle point does not appear between the two velocities. Such data are seen in the electron distribution functions (photo-electron and ambient-electron components) and also in the solar wind plasma (core and halo components). For such data, we need other criterion instead of the saddle point searching method. We address this as a future study.

Acknowledgment. We would like to thank Prof. T. Mukai for providing us with Geotail/LEP data.

References

Knowledge Discovery from fMRI Brain Images by Logical Regression Analysis

Hiroshi Tsukimoto¹, Mitsuru Kakimoto¹, Chie Morita¹, Yoshiaki Kikuchi²,
Eiko Hatakeyama³, Yoshifumi Miyazaki³

¹ Corporate Research & Development Center, Toshiba Corporation, 1, Komukai
Toshiba-cho, Saiwai-ku, Kawasaki 212-8582 Japan
hiroshi.tskimoto, mitsuru.kakimoto, chie.morita@toshiba.co.jp
² Tokyo Metropolitan University of Health Sciences, Higashi-ogu 7-2-10, Arakawa-ku,
Tokyo 110-8551 Japan
ykiyuki@metro-hs.ac.jp
³ Kansei Fukushima Research Center, Tohoku Fukushima University, Kunimigaoka,
Aoba-ku, Sendai 980-3201 Japan
eiko@tfu-mail.tfue.ac.jp
⁴ Forestry and Forest Products Research Institute, Ministry of Agriculture, Forestry
and Fisheries, P.O.Box 16, Tsukuba, Ibaraki 305-8687 Japan
miyukk@fpri.afric.go.jp

Abstract. This paper presents knowledge discovery from fMRI brain images. The algorithm for the discovery is the Logical Regression Analysis, which consists of two steps. The first step is regression analysis. The second step is rule extraction from the regression formula obtained by the regression analysis. In this paper, we use nonparametric regression analysis as a regression analysis, since there are not sufficient data in knowledge discovery from fMRI brain images. The algorithm has been applied to two experimental tasks, finger tapping and calculation. Experimental results show that the algorithm has rediscovered well-known facts and discovered new facts.

1 Introduction

Analysis of brain functions using functional magnetic resonance imaging (fMRI), positron emission tomography (PET), magnetoencephalography (MEG) and so on is called non-invasive analysis of brain functions[4]. As a result of the ongoing development of non-invasive analysis of brain function, detailed functional brain images can be obtained, from which the relations between brain areas and brain functions can be understood, for example, the relation between a subarea and another subarea in the motor area and a finger movement.

Several brain areas are responsible for a brain function. Some of them are connected in series, and others are connected in parallel. Brain areas connected in series are described by “AND” and brain areas connected in parallel are described by “OR”. Therefore, the relations between brain areas and brain functions are described by rules.
Researchers are trying to heuristically discover the rules from functional brain images. Several statistical methods, for example, principal component analysis, have been developed. However, the statistical methods can only present some principal areas for a brain function. They cannot discover rules. This paper presents an algorithm for the discovery of rules from fMRI brain images.

fMRI brain images can be dealt with by supervised inductive learning. However, the conventional inductive learning algorithms[5] do not work well for fMRI brain images, because there are strong correlations between attributes(pixels) and a small number of samples.

There are two solutions for the above two problems. The first one is the modification of the conventional inductive learning algorithms. The other one is nonparametric regression. The modification of the conventional inductive learning algorithms would require a lot of effort. On the other hand, nonparametric regression has been developed for the above two problems. We use nonparametric regression for the knowledge discovery from fMRI brain images. The outputs of nonparametric regression are linear formulas, which are not rules. However, we have already developed a rule extraction algorithm from regression formulas[9],[10]. [12].

The algorithm for knowledge discovery from fMRI brain images consists of two steps. The first step is nonparametric regression. The second step is rule extraction from the linear formulas obtained by the nonparametric regression. The method is a Logical Regression Analysis(LRA), that is, a knowledge discovery algorithm consisting of regression analysis and rule extraction from the regression formulas.

We applied the algorithm to artificial data, and we confirmed that the algorithm works well for artificial data[11]. We have applied the algorithm to real f-MRI brain images. The experiments are finger tapping and calculations. This paper reports that the algorithm works well for real f-MRI data, has rediscovered well-known facts regarding finger tapping and calculations, and discovered new facts regarding calculations.

Section 2 explains the knowledge discovery from fMRI images by Logical Regression Analysis. Section 3 describes the experiments.

2 Knowledge discovery from fMRI brain images by Logical Regression Analysis

2.1 The outline

The brain is 3-dimensional. In fMRI brain images, a set of 2-dimensional images(slices) represents a brain. See Fig. 1. 5 slices are obtained in Fig. 1. Fig. 2 shows a real fMRI brain image. When an image consists of $64 \times 64(=4096)$ pixels, Fig. 2 can be represented as Fig. 3. In Fig. 3, white pixels mean activations and black pixels mean inactivations. Each pixel has the value of the activation.

The regression analysis includes the nonlinear regression analysis using neural networks.
An experiment consists of several measurements. Fig. 4 means that a subject repeats a task (for example, finger tapping) three times. "ON" in the upper part of the figure means that a subject executes the task and "OFF" means that the subject does not execute the task, which is called rest. Bars in the lower part of the figure mean measurements. The figure means 24 measurements. When 24 images (samples) have been obtained, the data of a slice can be represented as Table 1.

Y(N) in the class stand for on/off of an experimental task. From Table 1, machine learning algorithms can be applied to fMRI brain images. In the case of Table 1, the attributes are continuous and the class is discrete.

Attributes (pixels) in image data have strong correlations between adjacent pixels. Moreover it is very difficult or impossible to obtain sufficient fMRI brain
samples, and so there are few samples compared with the number of attributes (pixels). Therefore, the conventional supervised inductive learning algorithms such as C4.5[5] do not work well, which was confirmed in [11].

Nonparametric regression works well for strong correlations between attributes and a small number of samples. The rule extraction algorithm can be applied to the linear formula obtained by nonparametric regression. The algorithm for the discovery of rules from fMRI brain images consists of nonparametric regression and rule extraction.

2.2 Nonparametric regression analysis

First, for simplification, the 1-dimensional case is explained[1]. Nonparametric regression is as follows:

Let \( y \) stand for a dependent variable and \( t \) stand for an independent variable and let \( t_j (j = 1, \ldots, m) \) stand for measured values of \( t \). Then, the regression formula is as follows:

\[
y = \sum a_j t_j + \epsilon (j = 1, \ldots, m),
\]

where \( a_j \) are real numbers and \( \epsilon \) is a zero-mean random variable. When there are \( n \) measured values of \( y \),

\[
y_i = \sum a_j t_i + \epsilon_i (i = 1, \ldots, n)
\]

In conventional linear regression, error is minimized, while, in nonparametric regression, error plus continuity or smoothness is minimized.

In fMRI brain images, there are continuities among the pixels, that is, adjacent measured values of the independent variable have continuity in the influence over the dependent variables[3]. Therefore, the continuity of coefficients \( a_{ij} \) is added to the evaluation value as follows:

\[
\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{m} (a_{j+1} - a_j)^2
\]

When \( \lambda \) is fixed, the above formula is the function of \( a_j (\hat{y}_i \) is the function of \( a_i \)). Therefore, \( a_i \)'s are determined by minimizing the evaluation value, and the optimal value of \( \lambda \) is determined by cross validation.

In 2-dimensional nonparametric regression, the evaluation value for the continuity of coefficients \( a_{ij} \) is modified. In 1 dimension, there are two adjacent measured values, while, in 2 dimensions, there are four adjacent measured values. The evaluation value for the continuity of coefficients is not \((a_{ij} - a_i)^2\), but the differences of first order between a pixel and the four adjacent pixels in the image. For example, in the case of pixel 66 in Fig.3, the adjacent pixels are
pixel 2, pixel 65, pixel 67, and pixel 139, and the evaluation value is as follows:
\[(a_{66} - a_2)^2 + (a_{66} - a_{65})^2 + (a_{66} - a_{67})^2 + (a_{66} - a_{139})^2.\]

2.3 Applying nonparametric regression analysis to fMRI brain images

When nonparametric regression analysis is applied to fMRI brain images, there are a few problems. Nonparametric regression analysis should be applied only to the pixels corresponding to brains. Therefore, areas corresponding to brains are extracted from fMRI brain images. The extraction can be executed by Statistical Parametric Mapping:SPM[6], which is widely used in brain science. SPM transforms brains to the standard brain[7].

There is no continuity between the inside and the outside of a brain. Therefore, the continuity of parameters in nonparametric regression analysis is not assumed at the boundary.

2.4 Rule extraction

The rule extraction algorithm in the discrete domain

The basic algorithm is that linear formulas are approximated by Boolean functions. Let \(f_i\) be the values of a linear formula. Let \(g_i(0\text{ or } 1)\) be the values of Boolean functions. The basic method is as follows:
\[g_i = \begin{cases} 1 & f_i \geq 0.5, \\ 0 & f_i < 0.5. \end{cases}\]

This algorithm minimizes Euclidean distance. The basic algorithm is exponential in computational complexity, and therefore, a polynomial algorithm is needed. The authors have presented the polynomial algorithm. The details can be found in [9], [10].

Extension to the continuous domain

Continuous domains can be normalized to \([0,1]\) domains by some normalization method. So only \([0,1]\) domains have to be discussed. First, we have to present a system of qualitative expressions corresponding to Boolean functions, in the \([0,1]\) domain. The expression system is generated by direct proportion, reverse proportion, conjunction and disjunction. The direct proportion is \(y = x\). The inverse proportion is \(y = 1 - x\), which is a little different from the conventional one \((y = -z)\), because \(y = 1 - x\) is the natural extension of the negation in Boolean functions. The conjunction and disjunction will be also obtained by a natural extension. The functions generated by direct proportion, inverse proportion, conjunction and disjunction are called continuous Boolean functions, because they satisfy the axioms of Boolean algebra. For example, \(z = x \lor y\) means that when \(x\) increases\((decreases)\) or \(y\) decreases\((increases)\), \(z\) increases\((decreases)\). For details, refer to [8]. In the domain \([0,1]\), linear formulas are approximated by continuous Boolean functions. The algorithm is the same as in the domain \([0,1]\).

Note that the independent variables in nonparametric regression should be normalized to \([0,1]\).
2.5 Related techniques

This subsection briefly explains two popular techniques, z-score and independent component analysis, and compares them with LRA.

z-score

z-score is widely used in fMRI brain images. z-score is calculated pixel-wise as follows:

\[
z = \frac{M_t - M_c}{\sqrt{\sigma_t^2 + \sigma_c^2}},
\]

where:

\(M_t\) : Average of task images

\(M_c\) : Average of rest images

\(\sigma_t\) : Standard deviation of task images

\(\sigma_c\) : Standard deviation of rest images

Task images mean the images when a subject performs an experimental task. Rest images mean the images when a subject does not perform an experimental task.

When z-score is 0, the average of task images equals the average of rest images. When z-score is 1 or 2, the difference between the average of task images and the average of rest images is big.

The areas whose z-scores are big are related to the experimental task. However, z-score does not tell which slices are related to the experimental task and does not tell the connections among the areas such as serial connection, parallel connection.

ICA

Independent Component Analysis (ICA) \(^2\) is applied to fMRI brain images. LRA is advantageous compared with ICA respecting the following points:

1. LRA uses classes. That is, LRA uses task/rest information, while ICA does not use task/rest information.
2. LRA conserves the spatial topologies in the images, while ICA cannot conserve the spatial topologies in the images.
3. LRA works well in the case of small samples, while it is not sure if ICA works well in the case of small samples.
4. LRA does not fall into a local minimum, while ICA falls into a local minimum.
5. LRA’s outputs can represent the connections among areas, while ICA’s outputs cannot represent the connections among the areas.

3 Experiments

LRA is applied to two tasks, finger tapping and calculation. The brain part of fMRI brain images are extracted using Standard Parametric Mapping:SPM(a
software for brain images analysis[6]) and LRA is applied to each slice. Rule extraction is applied to the results of nonparametric analysis. That is, the linear functions obtained by nonparametric regression analysis are approximated to continuous Boolean functions. Therefore, the domains of the linear functions should be [0, 1], and so the values of pixels should be normalized to [0, 1]. See the table in subsection 2.1.

The extracted rules are represented by conjunctions, disjunctions and negations of areas. The conjunction of areas means the co-occurrence activation of the areas. The disjunction of areas means the parallel activation of the areas. The negation of an area means a negative correlation.

### 3.1 Finger tapping

The first experimental task is a finger-to-thumb opposition task of the right hand, which is called finger tapping for simplification. The experimental conditions are as follows:

- Magnetic field: 1.5 tesla
- Pixel: 64x64
- Subject number: 1
- Task sample number: 30
- Rest sample number: 30

Table 2 shows the errors of nonparametric regression analysis. The errors was defined in 2.2. Slice 0 is the image of the bottom of brain and slice 33 is the image of the top of brain. Due to the experimental conditions, slices 0-4 have no data, which means that the bottom part images were not taken in the experiment.

<table>
<thead>
<tr>
<th>Slice</th>
<th>Error</th>
<th>Slice</th>
<th>Error</th>
<th>Slice</th>
<th>Error</th>
<th>Slice</th>
<th>Error</th>
<th>Slice</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>#</td>
<td>8</td>
<td>0.737</td>
<td>10</td>
<td>0.020</td>
<td>24</td>
<td>0.002</td>
<td>32</td>
<td>0.008</td>
</tr>
<tr>
<td>1</td>
<td>#</td>
<td>9</td>
<td>0.773</td>
<td>17</td>
<td>0.189</td>
<td>25</td>
<td>0.018</td>
<td>33</td>
<td>0.006</td>
</tr>
<tr>
<td>2</td>
<td>#</td>
<td>10</td>
<td>0.812</td>
<td>18</td>
<td>0.022</td>
<td>26</td>
<td>0.017</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>#</td>
<td>11</td>
<td>0.013</td>
<td>19</td>
<td>0.024</td>
<td>27</td>
<td>0.111</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>#</td>
<td>12</td>
<td>0.836</td>
<td>20</td>
<td>0.017</td>
<td>28</td>
<td>0.002</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.815</td>
<td>13</td>
<td>0.809</td>
<td>21</td>
<td>0.017</td>
<td>29</td>
<td>0.003</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.752</td>
<td>14</td>
<td>0.162</td>
<td>22</td>
<td>0.016</td>
<td>30</td>
<td>0.003</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.764</td>
<td>15</td>
<td>0.002</td>
<td>23</td>
<td>0.018</td>
<td>31</td>
<td>0.006</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Slices whose errors are small are related to the experimental task. The errors of slice 15, slice 21, and slice 28 are small. Therefore, rule extraction is applied to these slices. LRA can generate rules including disjunctions. However, the rules including disjunctions are too complicated to be interpreted by human experts in brain science, because they have paid little attention to the phenomena. Therefore, the rules including disjunctions are not generated.
Fig. 5 shows the extracted rule of slice 15, Fig. 6 shows the extracted rule of slice 24, and Fig. 7 shows the extracted rule of slice 28. White means high activity, dark grey means low activity, and black means non-brain parts. White and dark grey areas are connected by conjunction. For example, let $A$ stand for the white area in Fig. 6 and $B$ stand for the dark grey area in Fig. 6. Then, Fig. 6 is interpreted as $A \land \overline{B}$, which means area $A$ is activated and area $B$ is inactivated.

Figures are taken from feet, and so the left side in the figures means the right side of the brain, and the right side in the figures means the left side of the brain. The upper side in the figures means the front of the brain, and the lower side in the figures means the rear of the brain.

![Fig. 5. finger slice 15](image1) ![Fig. 6. finger slice 24](image2) ![Fig. 7. finger slice 28](image3)

White area in the upper side of Fig. 5 is related to movement planning, although the finger tapping is not so complicated. White area in the left side is basal nuclei, which adjust the movement. White area in Fig. 6 is supplementary motor area, which is related to voluntary movement or movement planning. White area in the right side of Fig. 7 means the activity of motor area and sensory area of fingers. LRA has rediscovered the relations that humans discovered.

LRA has generated rules, but conjunctions or negations are difficult to be interpreted by human experts, because the researchers have paid no attention to the co-occurrences of areas and negative correlations.

### 3.2 Calculation

The second experimental task is calculation. In the experiment, a subject adds a number repeatedly in the brain. The experimental conditions follow:

- Magnetic field: 1.5 tesla
- Pixel number: 64x64
- Subject number: 8
- Task sample number: 34
- Rest sample number: 36
Table 3 shows the errors of nonparametric regression analysis. The errors of nonparametric regression analysis are bigger than those in the case of finger tapping, which means that the regions related to calculation are located in 3 dimension. We focus on the slices whose errors are small, that is, the slices related to calculation. 133, ..., 336 in the table are the numbers of subjects.

Table 3. Results of nonparametric regression analysis (calculation)

<table>
<thead>
<tr>
<th>slice</th>
<th>133</th>
<th>135</th>
<th>312</th>
<th>317</th>
<th>321</th>
<th>331</th>
<th>332</th>
<th>336</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.924</td>
<td>0.882</td>
<td>0.444</td>
<td>0.547</td>
<td>0.0069</td>
<td>0.876</td>
<td>0.435</td>
<td>0.306</td>
</tr>
<tr>
<td>1</td>
<td>0.418</td>
<td>0.030</td>
<td>0.546</td>
<td>0.587</td>
<td>0.298</td>
<td>0.814</td>
<td>0.028</td>
<td>0.046</td>
</tr>
<tr>
<td>2</td>
<td>0.375</td>
<td>0.538</td>
<td>0.337</td>
<td>0.435</td>
<td>0.278</td>
<td>0.723</td>
<td>0.381</td>
<td>0.738</td>
</tr>
<tr>
<td>3</td>
<td>0.016</td>
<td>0.510</td>
<td>0.585</td>
<td>0.430</td>
<td>0.282</td>
<td>0.343</td>
<td>0.402</td>
<td>0.798</td>
</tr>
<tr>
<td>4</td>
<td>0.456</td>
<td>0.437</td>
<td>0.519</td>
<td>0.446</td>
<td>0.157</td>
<td>0.636</td>
<td>0.419</td>
<td>0.058</td>
</tr>
<tr>
<td>5</td>
<td>0.120</td>
<td>0.460</td>
<td>0.473</td>
<td>0.376</td>
<td>0.265</td>
<td>0.098</td>
<td>0.385</td>
<td>0.366</td>
</tr>
<tr>
<td>6</td>
<td>0.905</td>
<td>0.434</td>
<td>0.602</td>
<td>0.138</td>
<td>0.380</td>
<td>0.475</td>
<td>0.420</td>
<td>0.541</td>
</tr>
<tr>
<td>7</td>
<td>0.001</td>
<td>0.230</td>
<td>0.430</td>
<td>0.309</td>
<td>0.119</td>
<td>0.175</td>
<td>0.482</td>
<td>0.547</td>
</tr>
<tr>
<td>8</td>
<td>0.001</td>
<td>0.388</td>
<td>0.434</td>
<td>0.222</td>
<td>0.478</td>
<td>0.246</td>
<td>0.387</td>
<td>0.704</td>
</tr>
<tr>
<td>9</td>
<td>0.908</td>
<td>0.473</td>
<td>0.362</td>
<td>0.281</td>
<td>0.300</td>
<td>0.409</td>
<td>0.139</td>
<td>0.913</td>
</tr>
<tr>
<td>10</td>
<td>0.001</td>
<td>0.008</td>
<td>0.447</td>
<td>0.357</td>
<td>0.341</td>
<td>0.358</td>
<td>0.227</td>
<td>0.908</td>
</tr>
<tr>
<td>11</td>
<td>0.001</td>
<td>0.006</td>
<td>0.385</td>
<td>0.380</td>
<td>0.167</td>
<td>0.273</td>
<td>0.115</td>
<td>0.914</td>
</tr>
<tr>
<td>12</td>
<td>0.001</td>
<td>0.736</td>
<td>0.302</td>
<td>0.312</td>
<td>0.397</td>
<td>0.021</td>
<td>0.181</td>
<td>0.900</td>
</tr>
<tr>
<td>13</td>
<td>0.828</td>
<td>0.703</td>
<td>0.525</td>
<td>0.222</td>
<td>0.455</td>
<td>0.846</td>
<td>0.204</td>
<td>0.739</td>
</tr>
<tr>
<td>14</td>
<td>0.558</td>
<td>0.822</td>
<td>0.349</td>
<td>0.523</td>
<td>0.029</td>
<td>0.229</td>
<td>0.140</td>
<td>0.474</td>
</tr>
<tr>
<td>15</td>
<td>0.525</td>
<td>0.805</td>
<td>0.208</td>
<td>0.569</td>
<td>0.107</td>
<td>0.439</td>
<td>0.338</td>
<td>0.374</td>
</tr>
<tr>
<td>16</td>
<td>0.571</td>
<td>0.778</td>
<td>0.404</td>
<td>0.509</td>
<td>0.008</td>
<td>0.354</td>
<td>0.377</td>
<td>0.493</td>
</tr>
<tr>
<td>17</td>
<td>0.009</td>
<td>0.007</td>
<td>0.159</td>
<td>0.615</td>
<td>0.238</td>
<td>0.159</td>
<td>0.561</td>
<td>0.774</td>
</tr>
<tr>
<td>18</td>
<td>0.089</td>
<td>0.060</td>
<td>0.665</td>
<td>0.010</td>
<td>0.011</td>
<td>0.033</td>
<td>0.519</td>
<td>0.711</td>
</tr>
<tr>
<td>19</td>
<td>0.642</td>
<td>0.238</td>
<td>0.573</td>
<td>0.405</td>
<td>0.185</td>
<td>0.426</td>
<td>0.470</td>
<td>0.089</td>
</tr>
<tr>
<td>20</td>
<td>0.887</td>
<td>0.514</td>
<td>0.385</td>
<td>0.376</td>
<td>0.149</td>
<td>0.177</td>
<td>0.214</td>
<td>0.430</td>
</tr>
<tr>
<td>21</td>
<td>0.282</td>
<td>0.532</td>
<td>0.256</td>
<td>0.028</td>
<td>0.018</td>
<td>0.219</td>
<td>0.303</td>
<td>0.548</td>
</tr>
<tr>
<td>22</td>
<td>0.281</td>
<td>0.415</td>
<td>0.615</td>
<td>0.167</td>
<td>0.045</td>
<td>0.213</td>
<td>0.352</td>
<td>0.528</td>
</tr>
<tr>
<td>23</td>
<td>0.521</td>
<td>0.422</td>
<td>0.229</td>
<td>0.227</td>
<td>0.048</td>
<td>0.306</td>
<td>0.050</td>
<td>0.450</td>
</tr>
<tr>
<td>24</td>
<td>0.814</td>
<td>0.270</td>
<td>0.401</td>
<td>0.439</td>
<td>0.013</td>
<td>0.212</td>
<td>0.350</td>
<td>0.570</td>
</tr>
<tr>
<td>25</td>
<td>0.336</td>
<td>0.304</td>
<td>0.411</td>
<td>0.195</td>
<td>0.469</td>
<td>0.148</td>
<td>0.414</td>
<td>0.069</td>
</tr>
<tr>
<td>26</td>
<td>0.005</td>
<td>0.008</td>
<td>0.300</td>
<td>0.180</td>
<td>0.477</td>
<td>0.107</td>
<td>0.388</td>
<td>0.541</td>
</tr>
<tr>
<td>27</td>
<td>0.535</td>
<td>0.062</td>
<td>0.324</td>
<td>0.191</td>
<td>0.308</td>
<td>0.279</td>
<td>0.455</td>
<td>0.413</td>
</tr>
<tr>
<td>28</td>
<td>0.719</td>
<td>0.010</td>
<td>0.371</td>
<td>0.271</td>
<td>0.167</td>
<td>0.436</td>
<td>0.237</td>
<td>0.649</td>
</tr>
<tr>
<td>29</td>
<td>0.942</td>
<td>0.310</td>
<td>0.400</td>
<td>0.257</td>
<td>0.169</td>
<td>0.363</td>
<td>0.020</td>
<td>0.775</td>
</tr>
<tr>
<td>30</td>
<td>0.898</td>
<td>0.360</td>
<td>0.547</td>
<td>0.283</td>
<td>0.209</td>
<td>0.467</td>
<td>0.464</td>
<td>0.157</td>
</tr>
<tr>
<td>31</td>
<td>0.746</td>
<td>0.026</td>
<td>0.250</td>
<td>0.445</td>
<td>0.187</td>
<td>0.197</td>
<td>0.084</td>
<td>0.195</td>
</tr>
</tbody>
</table>

LRA can generate rules including disjunctions. However, the rules including disjunctions are too complicated to be interpreted. Therefore, the rules including disjunctions are not generated.
Table 4 summarizes the results of LRA. Numbers in parenthesis mean slice numbers. Activation in the left angular gyrus and supramarginal gyrus was observed in 4 and 3 cases, respectively, and that in the right angular gyrus and supramarginal gyrus was observed in 3 cases and 1 case, respectively. Clinical observations show that damage to the left angular and supramarginal gyri causes acalculia which is defined as an impairment of the ability to calculate. Despite the strong association of acalculia and left posterior parietal lesions,
there are certain characteristics of acaulia that have led to the suggestion of a right-hemispheric contribution. Clinical observations also suggest that acaulia is caused by lesions not only in the left parietal region and frontal cortex but also in the right parietal region. Fig. 8 shows slice 18 of subject 331 and Fig. 9 shows slice 26 of subject 331.

Significant activation was observed in the left inferior frontal gyrus in 6 out of 8 cases. On the other hand, none was observed in the right inferior frontal gyrus. The result suggests that the left inferior frontal gyrus including Broca’s area is activated in most subjects in connection with implicit verbal processes required for the present calculation task. Furthermore, significant activation in frontal region including middle and superior frontal regions was found in 8 cases (100%) in the left hemisphere and in 3 cases in the right hemisphere. The left dorsolateral prefrontal cortex may play an important role as a working memory for calculation. Fig. 10 shows slice 17 of 135 and Fig. 11 shows slice 18 of subject 317.

In addition to these activated regions, activation in cingulate gyrus, cerebellum, central regions and occipital regions was found. The activated regions depended on individuals, suggesting different individual strategies. Occipital regions are related to spatial processing, and cingulate gyrus is related to intensive attention. Central regions and cerebellum are related to motor imagery. 5 out of 8 subjects use cingulate gyrus, which means that they are intensively attentive. Fig. 12 shows slice 17 of 133. 5 out of 8 subjects use cerebellum, which is
thought not to be related to calculation. Fig. 13 shows slice 1 of 332. The above
two results are very interesting discoveries that have never been experimentally
confirmed so far. The problem of whether these regions are specifically related
to mental calculation or not is to be investigated in further research with many
subjects.

LRA has generated rules consisting of regions by conjunction and negation.
As for conjunctions and negations, the results showed the inactivated regions
simultaneously occurred with the activated regions. In the present experiment,
inactivation in the brain region contralateral to the activated region was ob-
served, suggesting inhibitory processes through corpus callosum. LRA has the
possibility of providing new evidence in brain hemodynamics.

3.3 Future work

In the experiments, LRA is applied to slices, that is, 2-dimensional fMRI brain
images. However, complicated tasks such as calculation are related to at least
a few areas, and so the application of LRA to a set of a few slices is neces-
sary for a fruitful knowledge discovery from fMRI brain images. Moreover, 2-
dimensional nonparametric regression analysis can be regarded as knowledge
discovery after attribute selection. Therefore, it is desired that LRA be applied
to 3-dimensional fMRI brain images. However, the nonparametric regression
analysis of 3-dimensional fMRI brain images needs a huge computational time.
Therefore, the computational time should be reduced, which is included in fu-
ture work. The interpretation of conjunctions and negations are also included in
future work.

4 Conclusions

This paper has presented an LRA algorithm for the discovery of rules from fMRI
brain images. The LRA algorithm consists of nonparametric regression and rule
extraction from the linear formula obtained by the nonparametric regression.
The LRA algorithm has discovered new relations respecting brain functions.

References

1. Eubank, R.L.: Spline Smoothing and Nonparametric Regression, Marcel
7. Talairach, J. and Tournoix: Coplanar Stereotactic atlas of the human brain, New York:


Human Discovery Processes Based on Searching Experiments in Virtual Psychological Research Environment

Kazuhisa Miwa

Graduate School of Human Informatics, Nagoya University, Nagoya 464-8601, Japan
miwa@cog.human.nagoya-u.ac.jp

Abstract. For designing experiments in social and human sciences, we must often consider various complex factors that seem to decide subjects’ performance. It is sometimes difficult to make complete experimental planning in which hypotheses guiding the experiments are established prior to executing the experiments. Even if the situation stands, experts in the field systematically organize their experimental processes. We propose Searching Experimental Scheme (SES) that enables them do so. For confirming the validity of SES, we construct virtual psychological experimental environment using a cognitive simulator in which subjects try to generate hypotheses and conduct experiments as scientists do. We analyze the subjects’ behavior based on SES and discuss the relation between the characteristics of their behavior and their performance of discovering targets.

1 Introduction

We can divide the ways of acquiring empirical data in the process of discovery into two basic categories: experimentation and observation. In experimentation, data are systematically gathered based on previously formed hypotheses. In experimental psychology, the most orthodox example is Factorial Design (FD) experiments in which focused factors by a researcher are systematically manipulated by clearly established hypotheses, and the relation between the manipulated factors and the observed data is identified. On the other hand, in observation, the systematic data collection as above is not made. In usual cases, hypotheses for manipulating experimental factors cannot be formed. So ways of gathering data become Trial and Error (TE) search in which experimental data are randomly observed for forming an initial hypothesis.

Experimental design in real research environment usually reflects the characteristics of both of the two typical categories above. For example, searching some levels of certain controlled factors may be lost even though the global structure of the experimental design is FD, or the experimental design is locally FD but the global structure (i.e. the relation of each local unit) may be TE. We call these intermediate ways of experimentation “searching experiments”, which is a key concept of this study. The process of searching experiments appears when (1) an experimental space that subjects try to search is huge, so the subjects cannot

© Springer-Verlag Berlin Heidelberg 2000
search whole combinations of the experimental conditions at once, (2) a goal itself is ambiguous, that is, a research objective itself is being searched, and (3) the relation between independent and dependent factors cannot be clearly predicted because of the lack of knowledge on the research domain or the existence of complex interaction among the experimental factors.

Searching experiments are essentially important especially in social and human sciences because most of the research situations are relatively complex and satisfy the conditions above [1] [7]. Researchers use searching experiments effectively for organizing their experimental processes systematically under the complex research situations. In this study, we propose “Searching Experimental Scheme” that enables researchers perform systematic search even though well-organized experimental planning such as FD experiments cannot be adopted. Then we analyze subjects’ behavior based on the scheme. We also discuss the relation between the characteristics of searching behavior and the performance of subjects’ discovering targets. To do so, using a discovery task that satisfies the conditions in which searching experiments appear, we let subjects experience a series of experimental processes, such as setting up a research objective, forming a hypothesis, designing experiments, performing experiments, interpreting experimental results, and rearranging additional experiments.

To discuss the issues above, it is difficult to let subjects perform real psychological experiments because of its executing cost. So in this study we let them perform virtual psychological experiments using a cognitive simulator that is constructed as a computer program instead of performing real experiments. Subjects behave as an experimental psychologist in the research environment provided by the simulator [8].

2 Virtual Psychological Research Environment

2.1 Wason’s 2-4-6 Task

The simulator used in this study is a cognitive model that simulates collaborative discovery processes in which two problem solvers interactively solve a traditional discovery task, the Wason’s 2-4-6 task, while referring mutual experimental results [9]. Subjects participate in this experiment as an experimental psychologist who studies collaborative discovery processes using the Wason’s task.

The standard procedure of the 2-4-6 task is as follows. Subjects are required to find a rule of relationship among three numerals. In the most popular situation, a set of three numerals, “2, 4, 6”, is presented to subjects at the initial stage. The subjects form hypotheses about the regularity of the numerals based on the presented set. Subjects conduct experiments by producing a new set of three numerals and present them to an experimenter. This set is called an instance. An experimenter gives Yes feedback to subjects if the set produced by subjects is an instance of the target rule, or No feedback if it is not an instance of the target. Subjects carry out continuous experiments, receive feedback from each experiment, and search to find the target.
Two types of experimentation, Ptest and Ntest, are considered. Ptest is experimentation using a positive instance for a hypothesis, whereas Ntest is experimentation using a negative instance. For example, when a subject has a hypothesis that three numerals are evens, an experiment using an instance, “2, 8, 18”, corresponds to Ptest, and an experiment with “1, 2, 3” corresponds to Ntest. Note that the positive or negative test is defined based on a subject’s hypothesis, on the other hand, Yes or No feedback is on a target. We should also notice the pattern of hypothesis reconstruction based on the combination of a hypothesis testing strategy and an experimental result (Yes or No feedback from an experimenter). When Ptest is conducted and No feedback is given, the hypothesis is disconfirmed. Another case of disconfirmation is the combination of Ntest and Yes feedback. On the other hand, the combinations of Ptest - Yes feedback and Ntest - No feedback confirm the hypothesis.

2.2 Interactive Production System

We have developed an interactive production system architecture for constructing the cognitive simulator and providing the virtual psychological research environment. The architecture consists of five parts: production sets of System A; production sets of System B; a working memory of System A; a working memory of System B; and a commonly shared blackboard (see Figure 1). The two systems interact through the common blackboard. That is, each system writes elements of its working memory on the blackboard and the other system can read them from the blackboard. The model solving the Wason’s 2-4-6 task has been constructed using this architecture.

![Diagram of Interactive Production System](attachment:image.png)

**Fig. 1.** Basic structure of the interactive production system architecture

The model has the knowledge on the regularities of three numerals, which is used for hypothesis generation in the process of solving the 2-4-6 task. The knowledge is organized as the dimension-value lists. For example, “continuous events”, “three evens”, and “the first numeral is even” are example values of a dimension, “Even-Odd”. The dimensions the systems use are: Even-Odd, Order, Interval, Range of digits, Certain digit, Mathematical relationship, Multiples, Divisors, Sum, Product, Different.
The way of searching the hypothesis space is controlled by the system’s parameter that decides the hypothesis formation strategy (see Table 2).

Basically the model searches the hypothesis space randomly in order to generate a hypothesis (when a value of the parameter, [random], is set). Three hypotheses, “three continuous evens”, “interval is 2”, and “three evens”, are particular. Human subjects tend to generate these hypotheses at first when the initial instance, “2, 4, 6”, is presented. So our model also generates these hypotheses first prior to other possible hypotheses when a value of the parameter, [human], is set.

You can see the detailed specifications of this model in Miwa & Okada, 1996 [5].

2.3 An Example Behavior of the Simulator

Table 1 shows an example result of the computer simulations. The target was “Divisor of three numerals is 12”. Two systems interactively found the target. One system, System A, always used Ptest in its experiments, and the other, System B, used Ntest. The table principally consists of three columns. The left-most and right-most columns indicate hypotheses formed by System A and System B respectively. The middle column indicates experiments, that is, generated instances, Yes or No feedback, and the distinction of Ptest or Ntest conducted by each system. Each experiment was conducted alternately by two systems, and the results of the experiments were sent to both of the two systems. The left-most number in each column indicates a series of processing, from #1 through #41. The right-most number in the left-most and right-most columns indicates the number of each hypothesis being continuously confirmed. System A confirmed its hypotheses at #4, #10, #16, which were introduced by self-conducted experiments at #3, #9, #15. System B confirmed its hypotheses at #17, #29, which were introduced by other-conducted experiments at #15, #27.

What we should note is that the simulator actually simulates human discovery processes. The validity of the simulator as a cognitive model has been already verified in other our papers. So usage of this simulator provides more realistic research environment in which we can observe searching processes of subjects who behave as an experimental psychologist.

2.4 Parameters Deciding the Model’s Behavior

The parameters that decide the simulator’s behavior consist of 6 factors that are indicated in Table 2. Five parameters except a first parameter, Target, are set up for controlling each of two interacting systems.

Let us now compare the situation in which searching experiments appear (see 1.) and the virtual experimental environment provided by the simulator. First, it is impossible to search the whole experimental space because it consists of two hundred million conditions (= 35 × 52 × 42 × 52 × 52 × 52). Second, focused factors, which are decided based on research objectives, are independently selected by subjects themselves; actually achieved solutions of every subject are different.
Table 1. An example behavior of the simulator

<table>
<thead>
<tr>
<th>Hypotheses by System A</th>
<th>Experiments</th>
<th>Hypotheses by System B</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 Continuous even numbers</td>
<td>0 1 2, 4, 6 Yes</td>
<td>-</td>
</tr>
<tr>
<td>4 The product is 48.</td>
<td>3 0 6, 6, -6, 17 No Next by SysA</td>
<td>5 The sum is a multiple of 4.</td>
</tr>
<tr>
<td>8 The product is 48.</td>
<td>1 1 9 24, -1, -2 No Next by SysA</td>
<td>6 The sum is a multiple of 4.</td>
</tr>
<tr>
<td>10 First + Second = Third.</td>
<td>0 1 12 3, 7, -20 No Next by SysA</td>
<td>7 The sum is a multiple of 4.</td>
</tr>
<tr>
<td>14 First + Second = Third.</td>
<td>1 1 15 -10, 2, -8 No Next by SysA</td>
<td>8 The sum is a multiple of 4.</td>
</tr>
<tr>
<td>16 Divisor is 12.</td>
<td>0 1 18 -5, -14, -9 No Next by SysA</td>
<td>9 The second is 4.</td>
</tr>
<tr>
<td>20 Divisor is 12.</td>
<td>0 1 21 2, 4, 6 Yes Next by SysA</td>
<td>10 The second is 4.</td>
</tr>
<tr>
<td>22 Divisor is 12.</td>
<td>1 2 24 -7, 3, 12 No Next by SysA</td>
<td>11 Divisor is 12.</td>
</tr>
<tr>
<td>26 Divisor is 12.</td>
<td>3 2 27 2, 12, -12 Yes Next by SysA</td>
<td>12 Divisor is 12.</td>
</tr>
<tr>
<td>28 Divisor is 12.</td>
<td>4 3 30 8, 12, -2 No Next by SysA</td>
<td>13 Divisor is 12.</td>
</tr>
<tr>
<td>32 Divisor is 12.</td>
<td>5 3 33 2, 6, -2 Yes Next by SysA</td>
<td>14 Divisor is 12.</td>
</tr>
<tr>
<td>34 Divisor is 12.</td>
<td>6 3 36 -2, -7, -8 No Next by SysA</td>
<td>15 Divisor is 12.</td>
</tr>
<tr>
<td>38 Divisor is 12.</td>
<td>7 3 39 4, 3, -12 Yes Next by SysA</td>
<td>16 Divisor is 12.</td>
</tr>
<tr>
<td>40 Divisor is 12.</td>
<td>8 3 41 8 No Next by SysA</td>
<td>17 Divisor is 12.</td>
</tr>
</tbody>
</table>

Third, there are complex interactions especially among three factors: hypothesis testing strategies, hypothesis formation strategies, and targets. These points support that the research environment used in this study embodies the situation in which searching experiments appear.

3 Experiments

Six graduate students participated in the experiment. They attended a graduate school class given by the author. The topic of the class was experimental psychological studies on human hypothesis testing. So the experimental situation was that the subjects who had obtained basic psychological knowledge on human hypothesis testing were required to study collaborative discovery processes in the experimental environment, applying the basic knowledge to the collaborative situation. Each subject individually participated in the experiment. After instructional guidance for 20 minutes, the main experiment, in which the subjects manipulated the simulator and studied collaborative discovery processes, was carried out for 2 hours; then an interview for 15 minutes was conducted.

In the main experiment, the subjects performed experiments manipulating the simulator independently. An experimental planning sheet was used; the sheet consisted of 5 items: (1) a research objective (what do they investigate); (2) hypotheses, (3) an experimental design, (4) experimental results, and (5) interpretation of the experimental results. The subjects filled out the former three items, and then they actually conducted experiments manipulating the simulator. After the experiments they filled out the latter two items of the sheet. They repeated this procedure during the main experiment. In the interview after the experimental session, subjects' conclusions (i.e. what do they find) through the whole experiments were identified.
Table 2. Six factors of the simulator

<table>
<thead>
<tr>
<th>Factors</th>
<th>Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target [T]</td>
<td>[1] - [35]</td>
</tr>
<tr>
<td></td>
<td>Thirty-five kinds of targets that were used in the experiment. For example, Target #1 is &quot;ascending numbers&quot;; Target #35 is &quot;three different numbers&quot;.</td>
</tr>
<tr>
<td>Hypothesis testing</td>
<td>[0], [25], [50], [75], [100]</td>
</tr>
<tr>
<td>strategies [HT]</td>
<td>The probability of conducting positive tests in generating instances. [100] and [0] mean that systems always conduct positive tests and negative tests, respectively.</td>
</tr>
<tr>
<td>Hypothesis formation</td>
<td>[human], [random], [specific], [general]</td>
</tr>
<tr>
<td>strategies [HF]</td>
<td>[human] means that systems generate hypotheses as humans do. [random]: generating hypotheses randomly. [specific]: generating specific hypotheses prior to general ones. [general]: generating general hypotheses prior to specific ones.</td>
</tr>
<tr>
<td># of activated instances</td>
<td>[all], [6], [5], [4], [3]</td>
</tr>
<tr>
<td>[AI]</td>
<td>The number of instances that can be activated at once in the working memory when generating hypotheses.</td>
</tr>
<tr>
<td># of maintained</td>
<td>[all], [5], [4], [3], [2]</td>
</tr>
<tr>
<td>hypotheses [RH]</td>
<td>The number of rejected hypotheses that can be maintained in the working memory.</td>
</tr>
<tr>
<td>Condition for</td>
<td>[all], [5], [4], [3], [2]</td>
</tr>
<tr>
<td>terminating the search</td>
<td>The number of continuous confirmations when systems terminate the search. [2] means when a hypothesis is continuously confirmed two times, systems recognize the hypothesis as the solution, and terminate the search.</td>
</tr>
</tbody>
</table>

After the combination of every level of the 6 factors is decided, twenty simulations are automatically executed in the condition. Then the experimental system presents (1) the ratio of each of the two systems correctly finding a target, (2) the ratio of at least one of the two systems reaching a correct solution, and (3) the average number of generated instances for reaching a correct solution. The system also presents a model’s solution process of each simulation in addition to the final results as above; however, on the basis of the experimenter’s instruction, the subjects only focus on the final performance of the systems and try to find factors that explain the performance. The experimental system automatically records subjects’ experimental behavior. Additionally, the processes were also recorded by a video camera, and subjects’ verbal protocols were gathered. Those protocols were used as secondary data for identifying subjects’ behavior when their description on the experimental planning sheet was ambiguous.

4 Searching Experimental Scheme

4.1 Expanded Search Within/Out of Focused Factors

In this study, we describe subjects’ experimental processes based on “Searching Experimental Scheme” (SES). Figure 2 shows the experimental space consisting of the 6 factors of the simulator, that is, the combinations of every level of each
factor [2]. From Factor 2 through Factor 6, every level is decided in each of the two systems. The bold lines of Figure 2 show an example combination: Factor 1, a used target is “ascending numbers” (Target #1); Factor 2 and Factor 3, the combination of hypothesis testing and formation strategies is positive testing and specific formation strategies in one system v.s. negative testing and general formation strategies in the other system; Factor 4, whole instances in the working memory can be activated; Factor 5, every hypothesis can be maintained in the memory; Factor 6, search is terminated when a hypothesis is supported by three continuous confirmations.

![Fig. 2. experimental space of the simulator](image)

Subjects’ behavior will be described on SES shown in Figure 3. Figure 3 consists of three basic units, Unit A11, Unit A12, and Unit A21. Each unit corresponds to a set of subjects’ searching behavior. We regard a series of continuous experiments guided by single experimental design on a piece of experimental planning sheet as a set of searching behavior.

In Unit A11, a subject manipulates Factor $n$ and Factor $m$, and searches some levels of the factors indicated by the bold lines. We call these manipulated factors “focused factors”. Focused factors are indicated by dark gray boxes. Next in Unit A12, another factor, Factor $p$, indicated by a light gray box, is manipulated while fixing the levels of the focused factors already searched in Unit A11. We call this searching behavior “expanded search out of focused factors”.

Moreover, subjects do not necessarily search whole levels of focused factors within a single unit; so they often conduct additional search of the focused factors. For example, in Unit A21, a subject searches other levels of the focused factors than the levels that have been already searched in Unit A11. We call this searching behavior “expanded search within focused factors”. Although subjects cannot search all levels of focused factors at once because of their cognitive resource constraints, they try to analyze the effects of the focused factors on the total performance by conducting the expanded search within focused factors. Moreover, by conducting the expanded search out of focused factors, they try to
know how the results obtained on the focused factors are affected by fluctuation of other factors. The expanded search within/out of focused factors reflects the characteristics of well organized searching experiments.

4.2 Levels of Searching Behavior

An important point in using SES defined above is that we can identify several levels of regularity of subjects’ searching behavior. First, on the most basic level, a chunk (a unified set of subjects’ searching behavior) is represented as each of Unit A11, Unit A12, and Unit A22. We call each chunk a “Unit”. Next on the second level, a chunk is constructed by expanded search out of focused factors. We call this chunk a “Series”. On this level, the subjects’ behavior in Figure 3 is unified into two chunks: one chunk is Series A1 that consists of Unit A11 and Unit A12 and the other is Series A2. Finally on the third level, whole subjects’ behavior in Figure 3, organized by expanded search within focused factors, is regarded as one chunk. We call this chunk on the highest level a “Block”.

Now we should define each termination of a Series and a Block. A Series continues when subjects manipulate other factors than focused factors while fixing the already searched levels (or a part of the levels) of the focused factors. A Series terminates when conducting expanded search out of focused factors while shifting the search of the focused factors to new levels that have not been examined. A Block continues when subjects manipulate focused factors while fixing the already searched levels (or a part of the levels) of other factors than the focused factors. A Block terminates when both of focused factors and other factors are manipulated at once.

Figure 4 shows an example searching behavior of Subject B described based on SES.
4.3 Three Stages of Chunking

Figure 5 shows the total number of experiments of each subject and the numbers of chunks on the three levels (Unit, Series, and Block) defined in the previous section. Constructing chunks on the higher levels means higher organization of subjects’ searching behavior; so Figure 5 indicates the situation of phased organization processes of subjects’ searching behavior.

Now to model the patterns of the phased organization process, let us consider the 2 factorial (3 x 3) experimental design. Figure 6 (a) shows the case in which the experiments are performed based on FD in which all levels of two focused factors are searched at once. In this case a Unit is equal to a Block. Searching behavior is only organized through the process of constructing a Unit from individual experiments. We call this organization process “first stage of chunking”. On the other hand, when experiments are performed based on TE search, every experiment is independent from each of the former and latter experiments; so no chunking happens. In this case each single experiment constructs a Block (see Figure 6 (c)). Characteristics of subjects’ behavior of searching experiments appear in expanded search within/out of focused factors; they can be modeled,
from the viewpoint of the three levels of chunking, based on the patterns depicted in Figure 6 (b) through (d). In Figure 6 (b), after manipulating a single focused factor, subjects conduct expanded search out of the focused factor. We call this organization of searching behavior “second stage of chunking”. In Figure 6 (c), although a subject manipulates two factors at once, the whole levels of the focused factors are not searched in the first unit; so expanded search within the focused factors appears. We call this organization process of behavior “third stage of chunking”. In Figure 6 (d), both types of expanded search appear.

4.4 Compression Ratio of Chunking

We understand, through comparing Figure 5 and Figure 6, the behavior of Subject A represents the characteristics of FD experimental processes, whereas the behavior of Subject F represents TE search from the viewpoint of the three stages of chunking. The behavior of other four subjects represents the characteristics of searching experiments in which they organize their behavior on the second and third stages of chunking.

To clarify the discussions above, we define the compression ratio of chunking. The compression ratios on the first, second, and third stages of chunking are defined as the ratio of the number of Units to the total number of experiments, the ratio of the number of Series to the number of Units, and the ratio of the number of Blocks to the number of Series, respectively. As the compression ratio decreases, it means that higher compression is made. FD experimental behavior is structured only on the first stage of chunking on which higher compression is made; so the compression ratio is reduced from 1.0 whereas chunking on the second and third stages is not performed on which the compression ratios almost equal 1.0. On the other hand, in TE searching behavior the compression ratio on any stage of chunking nearly equals 1.0. The characteristics of behavior of searching experiments appear on the second and third stages of chunking on which the compression ratio is relatively reduced from 1.0.
Figure 6. Patterns of three stages of chunking

Figure 7 shows the compression ratios of each of the six subjects on the three stages of chunking. For example, let us consider an example behavior of Subject B depicted in Figure 4. The compression ratios of Subject B on the three stages of chunking were .32, .45, and .80, because the number of experiments, Units, Series, and Blocks were, as seen in Figure 4, thirty-four, eleven, five, and four (also see Figure 5). Figure 7 indicates that the compression ratio of Subject A on the first stage is the smallest, and the compression ratios on the second and third stages nearly equal 1.0; so the behavior of Subject A reflects the characteristics of TE search. In terms of other four subjects, chunking on the second or third stages, in addition to the first stage, is performed; so their behavior reflects the characteristics of searching experiments. Additionally, Figure 7 indicates that Subject D organizes his behavior on the second stage by expanded search out of focused factors because the compression ratio on this stage is smaller than that on the third stage. On the other hand Subject E organizes it on the third stage by expanded search within focused factors.
5 Searching Behavior and Performance of Discovery

5.1 Categorization of Final Solutions

Next we consider the relation between the characteristics of searching behavior described above and final solutions reached by each of the six subjects. In Table 3, searched factors related to the final solutions by the six subjects are indicated, each of which is classified from the following two viewpoints. First, the solutions are divided into two categories from the viewpoint of their generality. Solutions in one category refer to the factors that affect the system's performance while comparing several levels of the factors or mentioning to the effects of fluctuations of other factors. One example is "in terms of hypothesis formation strategies, the combination of the specific and general strategies produces the highest performance regardless of fluctuations of other factors." On the other hand, some subjects simply reported an individual level of searched factors that seem to decide the system's performance. One example is "in terms of hypothesis formation and testing strategies, when the former is the general strategy and the latter is the combination of the positive and negative testing strategies, the ratio of correct solutions reaches high." We call the former type of solutions general solutions whereas the latter specific solutions.

As the second viewpoint, the solutions in Table 3 are also classified from their validity. The correctness of each solution is decided based on both of knowledge on human discovery processes which has been obtained from cognitive psychological studies using the Wason's task [3] [4] and knowledge on regularities of our simulator's behavior identified in other our papers [5] [6]. We can divide the solutions of the six subjects into two categories from the two viewpoints mentioned above. One type of solutions is general and correct solutions whereas the other type is specific and incorrect solutions. Subject A, Subject B, and Subject C reached the former type of solutions, whereas Subject D, Subject E, and Subject F reached the latter type.
Table 3. Categorization of subjects’ solutions

<table>
<thead>
<tr>
<th>Subject</th>
<th>Factors</th>
<th>Generality</th>
<th>Validity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject A</td>
<td>AI-RH</td>
<td>General</td>
<td>Correct</td>
</tr>
<tr>
<td>Subject B</td>
<td>HF</td>
<td>General</td>
<td>Correct</td>
</tr>
<tr>
<td>Subject C</td>
<td>T</td>
<td>General</td>
<td>Correct</td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>Specific</td>
<td>Correct</td>
</tr>
<tr>
<td>Subject D</td>
<td>HT-HF</td>
<td>Specific</td>
<td>Incorrect</td>
</tr>
<tr>
<td>Subject E</td>
<td>HT-HF</td>
<td>Specific</td>
<td>Correct</td>
</tr>
<tr>
<td>Subject F</td>
<td>HT-HF</td>
<td>Specific</td>
<td>Incorrect</td>
</tr>
</tbody>
</table>

5.2 Factors Deciding Subjects’ Performance

Now we move to discussions on the relation between the characteristics of subjects’ behavior that were clarify in 4. and the solutions that each of the six subjects reached. Let us see Figure 7 again. The compression ratios of the subjects who reached general and correct solutions on the first stage of chunking are smaller than the ratios of those who reached specific and incorrect solutions. This indicates that the subjects who got correct and general solutions made higher compression on the first stage of chunking; those who got incorrect solutions could not. This insists that even though the characteristics of searching experiments appear on the second and third stages of chunking, chunking on the basic first stage is crucial for organizing their behavior.

Moreover Table 4 shows searched factors by each subject. The underlined factors indicate the factors related to the final solutions of each subject. The indexes, “o”, “x”, and “-”, indicate systematically searched factors, randomly searched factors, and factors that were not searched, respectively. The systematic search means that the subjects searched whole levels of the factors or some representative levels, such as levels that have the highest or lowest values. Table 4 shows that two subjects, Subject B and Subject C, who reached correct solutions systematically searched two kinds of focused factors at once or conducted systematic search of other factors by expanded search out of focused factors. Subject A, even though he also reached correct solutions, did not conduct the expanded search. The reason is because the focused factors by Subject A, AI and RH, do not interact with other factors. On the other hand, every subject who reached incorrect solutions simply manipulated a single factor and could not conduct expanded search out of focused factors. Moreover some of them failed in systematic search of the focused factors.
### 6 Summary and Conclusions

In this paper, defining experimental processes that reflected the characteristics of both of FD experiments and TE search as *searching experiments*, we analyzed the ways of organizing behavior of *searching experiments* using SES. We also discussed the relation between the characteristics of the behavior and the performance of discovering targets. We understood, through the analysis of subjects’ behavior on SES, that they organized their behavior on the three levels, Unit, Series, and Block. Chunking on the second and third stages, constructing Series and Blocks, reflected the characteristics of behavior of *searching experiments*.

In the latter part of this paper, we clarified that subjects who reached general and correct solutions effectively performed chunking on the first stage, which worked as the basis of organizing searching behavior, and systematically manipulated searched factors. One of our future works is to establish the ways for feedback of description of subjects’ experimental processes based on SES and to discuss its educational effects.

### References


#### Table 4. Searched factors and subjects’ performance

<table>
<thead>
<tr>
<th>Subject</th>
<th>Focused Factors</th>
<th>Other Searched Factors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject A</td>
<td>AI EH HT RH</td>
<td>-</td>
</tr>
<tr>
<td>Subject B</td>
<td>T T HF 0</td>
<td>HT x</td>
</tr>
<tr>
<td>Subject C</td>
<td>T T HT 0</td>
<td>HT 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subject</th>
<th>Focused Factors</th>
<th>Other Searched Factors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject D</td>
<td>HT</td>
<td>0</td>
</tr>
<tr>
<td>Subject E</td>
<td>ambiguous</td>
<td>AI-HF x</td>
</tr>
<tr>
<td>Subject F</td>
<td>HT</td>
<td>0</td>
</tr>
</tbody>
</table>

---

238 Kazuhisa Miwa


Prediction of Binding Affinities for Protein-Ligand Complexes with Neural Network Models

Vladimir V. Krepets, Natalya V. Belkina

Laboratory of Molecular Graphics & Drug Design, Institute of Biomedical Chemistry.
Pogodinskaya Street, 10, Moscow, 119832, Russia.
krepets@ibmh.msk.su, belkina@ibmh.msk.su

Molecular mechanisms of drug action are often based on an interaction of them with target macromolecules, such as proteins and nucleonic acids. The formation of ligand-target complexes is typical for biologically active compounds, including activators and inhibitors of various enzymes. Prediction of the dissociation constant (Kd) of protein-ligand complex is often used as a scoring function for the modeled complexes and there are many approaches in the field of prediction of such constant [1]. In the present work various parameters of protein-ligand complexes were used to predict Kd. These parameters can be quickly calculated immediately during docking procedure, which we usually used for complexes modeling. The artificial feedforward neural networks (AFNNs) were used as a mathematical approach to prediction of protein-ligand complexes Kd. In practice, neural networks are especially useful for classification and function approximation problems, which have a lot of training data. Neural networks are often used in situations where you do not have enough prior knowledge to set the activation function, as in case of the prediction of the protein-ligand complexes dissociation constant.

The Kd values for 83 various complexes of biological molecule [2] were used in the present work. All of these complexes have crystallographic coordinates of 3D structures. This set of complexes was divided randomly into two subsets. The training set includes 68 points and the test one includes 15 points. Hereinafter Kd will appear as the predicted values. The crystallographic data for all complexes passed preliminary handling according the uniform scheme by using program suite Sybyl [3]:

- rebuild hydrogen atoms in molecules;
- remove crystallographic water molecules (except the case of HIV protease, where one molecule of water was accepted as an element of a ligand);
- check and correct types of atom and bond;
- solvate the complexes;
- optimize the structure of complexes in the water environment.

Estimation of following parameters were done for all of 83 complexes:
1. The number of atoms in target and ligand part of complex.
2. The value of energy due to electrostatic interactions [3].
3. The attitude of the closed surface in a complex to the full surface which is accessible for water (sphere radius 1.4 Å) in unbound molecule. These parameters were estimated both for ligand and target parts of complex [4].
4. Analogous parameters estimated by using sphere with 0.5 Å radius [4].
5. The changes of integral parameters of hydrophilicy and lipophilicy and the changes...
of value of hydrophilic and lipophilic areas. These parameters were calculated by using the original program, based on a molecular lipophilicity potential [5]. So, we have got 15 independent variables, which were used to design statistical models for Kd predicting. The AFNN with one hidden layer and following set of activation functions (F1, F2 and F3) were applied to the models design: F1(x) = x; F2(x) = sign(x)\*ln(1+|x|); F3(x) = sign(x)\*(e^|x|-1). Three models were considered: linear model (1N) and two non-linear models - combination of activation functions F2 and F3 (2N) and combination of activation functions F1, F2 and F3 (3N). All models were constructed by using the original software [6]. Neural Network Constructor (NNC) was developed for creating non-linear models and for successfully solving statistical problems arising from different fields of knowledge, for instance, biochemistry. NNC v.3.01 is free software available on WWW. Some statistical characteristics of created models are shown in the table 1. In the table R² means the square of binary correlation coefficient, MSE - mean square error. Subscript "learn" corresponds to the tuning process, "control" corresponds to the "leave-one-out" procedure and "test" corresponds to applying of created model to the test sample.

Table 1. Statistical characteristics of the models.

<table>
<thead>
<tr>
<th>Model</th>
<th>R²learn</th>
<th>MSElearn</th>
<th>R²control</th>
<th>MSEcontrol</th>
<th>R²test</th>
<th>MSEtest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear model</td>
<td>0,634</td>
<td>1,098</td>
<td>0,571</td>
<td>1,193</td>
<td>0,686</td>
<td>0,947</td>
</tr>
<tr>
<td>2 neurons model</td>
<td>0,743</td>
<td>0,920</td>
<td>0,582</td>
<td>1,225</td>
<td>0,730</td>
<td>0,924</td>
</tr>
<tr>
<td>3 neurons model</td>
<td>0,837</td>
<td>0,733</td>
<td>0,705</td>
<td>0,995</td>
<td>0,806</td>
<td>0,979</td>
</tr>
</tbody>
</table>

As shown above, the using of simple linear model allow obtaining valid result for Kd prediction on the test set. Using 2N model gives only insignificant improvement for predictive ability. 3N model is much better if we take into account first five indexes (R²learn; MSElearn; R²control; MSEcontrol; R²test). When we pay any attention to the last quality index (MSEtest) we can notice that the last model is not so good as would be desirable. It is necessary to note that the square of correlation coefficient on testing is more than the square of correlation coefficient on "leave-one-out" procedure. It can be explained by following reasons:

- inaccuracy of Kd measurement;
- inaccuracy of RSA method;
- insufficiency of a sample;
- non correct description of atom and bond types in using force field.

There are several ways to improve the model quality: refinements of source data, robust estimating of model parameters or increase size and variety of the training set.

3. Sybyl 6.4, Tripos Inc., 1699 South Hanley Road, St Louis, Missouri, 63144.
Automatic and Accurate Determination of the Onset Time of the Quasi-periodic Oscillation

Tomoyuki Higuchi

The Institute of Statistical Mathematics,
4-6-7 Minami-Azabu, Minato-ku, Tokyo 106-8569 Japan
higuchi@ism.ac.jp

1 Introduction

The identification of the start (onset) time of the quasi-periodic oscillation (QPO), which is called the Pi 2 pulsations in magnetospheric physics, from the ground magnetic field observation is usually carried out by focusing on a wave-like component obtained by applying a linear band-pass filter [5, 6]. When the background magnetic field (i.e., time-dependent mean value structure) and/or the amplitude of high-frequency components (i.e., time-dependent variance structure) change rapidly around the initial period of Pi 2 pulsations, any linear band-pass filter, which also includes the procedure based on a simple modification of the wavelet analysis, always generates a pseudo precursor prior to a true onset time. In such a case, an accurate determination of onset time requires a nonlinear filter which enables us to separate only the wavy-like component associated with Pi 2 pulsations from the time-varying mean and/or variance structures with various discontinuities. In this study we introduce a locally fixed time series model which partitions the time series into three segments and to model each segment as the linear combination of several possible components. An optimal partition obtained by the minimum AIC procedure allows us to determine an onset time precisely even for the above-mentioned case. We illustrate this procedure by showing an application to actual data sets.

2 Treatment of Rapid Decrease in Trend

The time series \( Y_{1:N} = [y_1, \ldots, y_N] \) is a scalar observation which is the H component recorded by a magnetometer at the ground station [8]. We sometimes observe an extremely rapid decrease in the background magnetic field measured at the high latitude stations. A preparatory removal of such rapid change in the trend from the original observations enhances efficiency and accuracy in an estimation of parameters involved in describing a time series model, because an onset time determination in our approach is based on a representation of the time series by a flexible model with many unknown parameters. Prior to an analysis of an onset determination we therefore apply a detrending procedure which fits a parametrically described function \( \mu_n(\theta) \) to \( y_n \), where \( \theta \) is a parameter vector for representing \( \mu_n \) that is a function of \( n \).

© Springer-Verlag Berlin Heidelberg 2000
The detrending procedure begins by examining a sequence of the first difference of original time series and identifying intervals each of which is defined by consecutive data points with the first difference value smaller than a certain threshold, $\gamma_b$. We denote the $j$th interval by $D_j = [i_j, A, i_j, B]$ ($j = 1, \ldots, J$), where $J$ is the number of intervals with a rapid decrease. A detailed examination of $y_n$ for $n \in D_j$ founds that a rapid decrease can be approximated by the first quarter of the cycle of a cosine function. Specifically a function form for the $j$th rapid decrease, $f_{n,j}$, is given by

$$f_{n,j} = (g_{j,A} - g_{j,B}) \cos \left( \frac{2n \pi (n - i_j, A)}{4(i_{j,B} - i_{j,A})} \right) + g_{j,B} \quad \text{for } n \in D_j. \quad (1)$$

$\mu_n$ for an interval between $D_j$ and $D_{j+1}$, specified by $h_n^j$, is simply given by a linear function:

$$h_n^j = \left( \frac{g_{j+1,A} - g_{j,B}}{i_{j+1,A} - i_{j,B}} \right) (n - i_{j,B}) + g_{j,B} \quad \text{for } n \in C_j, \quad (2)$$

where $C_j = (i_{j,B}, i_{j+1,A})$. $\mu_n$ for an interval before $D_1$ is given by a constant: $h_n^1 = g_{1,A}$. Similarly, for an interval after $D_J$, i.e., $C_J = (i_{j,B}, N)$, $\mu_n$ is given by $h_n^J = g_{J,B}$.

For given set of $D_1, \ldots, D_J$, an optimal set of $(g_{j,A}, g_{j,B})$ ($j = 1, \ldots, J$) is easily obtained by applying the least squares fit. Actually a minor adjustment of a location of $D_j$ itself is carried out by minimizing the squared residuals. Eventually $\mu_n$ is represented with a parameter vector $\theta$ which consists of $4J$ variables:

$$\theta = \left[ (i_{1,A}, g_{1,A}), (i_{1,B}, g_{1,B}), \ldots, (i_{J,A}, g_{J,A}), (i_{J,B}, g_{J,B}) \right]^T. \quad (3)$$

As a result, a procedure for obtaining an optimal $\theta$, $\theta^*$, turns out to become non-linear. The detrended signal, $e_n$, is defined by $e_n = y_n - \mu_n(\theta^*) \ (n = 1, \ldots, N)$.

### 3 Data Partition

Suppose that a wave train of the Pi 2 pulsation is observed in $E_{1:N} = [e_1, \ldots, e_N]$, and denote its starting and ending points by $k_1 + 1$ and $k_2$, respectively. According to a total interval is divided into three sub-intervals:

$$E_{1:N} = [e_1^{(1)}, \ldots, e_{k_1}^{(1)}] \cup [e_{k_1+1}^{(2)}, \ldots, e_{k_2}^{(2)}] \cup [e_{k_2+1}^{(3)}, \ldots, e_N] \quad (4)$$

A presence of the Pi 2 pulsations is assumed only for an interval $I^{(2)}$. The Akaike Information Criterion (AIC) [1] for $E_{1:N}$, $\text{AIC}_N$, is given by

$$\text{AIC}(k_1, k_2) = \text{AIC}_N = \text{AIC}^{(1)} + \text{AIC}^{(2)} + \text{AIC}^{(3)}, \quad (5)$$

that is a function of $k_1$ and $k_2$, where $\text{AIC}^{(m)}$ is the AIC for the $m$th interval [7]. The onset and offset time of the Pi 2 pulsations are given by the optimal dividing points, $k_1^* \ and \ k_2^*$, respectively, which are determined by minimizing the $\text{AIC}_N$. 

---

**Automatic and Accurate Determination of the Onset Time**

243
4 Time Series Model for Each Segment

Suppose that the time series $e_n$ for the $m$-th interval is given by the following observation model

$$e_n = t^{(m)}_n + s^{(m)}_n + w^{(m)}_n, \quad w^{(m)}_n \sim N(0, \sigma^2(m)) \quad (m = 1, 2, \text{ and } 3),$$

where $t^{(m)}_n$ is a stochastic trend component and is assumed to follow a system model [4]

$$t^{(m)}_n = 2t^{(m)}_{n-1} - t^{(m)}_{n-2} + v^{(m)}_n, \quad v^{(m)}_n \sim N(0, \tau^2(m)).$$

$w^{(m)}_n$ is the observation noise. $s^{(2)}_n$ corresponds to the signal associated with the Pi 2 pulsations which is assumed to be a stochastic process with colored power spectrum. Obviously, $s^{(1)}_n = s^{(3)}_n \equiv 0$.

In this study $s^{(2)}_n$ is furthermore decomposed into the quasi-periodic oscillation (QPO) component $q_n$ and autoregressive (AR) component $r_n$: $s^{(2)}_n = q_n + r_n$. $q_n$ and $r_n$ are modeled by

$$q_n = 2 \cos(2\pi f_c) q_{n-1} - q_{n-2} + v^q_n, \quad v^q_n \sim N(0, \tau^2_q),$$

and

$$r_n = \sum_{j=1}^{J\text{AR}} a_j r_{n-j} + v^r_n, \quad v^r_n \sim N(0, \tau^2_r),$$

respectively. $f_c$ corresponds to a reciprocal of a period of the Pi 2 pulsations in unit of data points. In this study it is treated as unknown parameter and need not be given beforehand. The presence of system noise in (8) makes the cycle stochastic rather than deterministic, and thus the QPO model allows us to represent a periodic component of distinct frequency $f_c$ with stochastically time-varying amplitude and phase [3].

The AR component is introduced to represent the locally stationary component in $s_n$. Namely, whereas $q_n$ describes a signal with an eminent peak in power spectrum (i.e., line spectrum), $r_n$ accounts for a signal having a continuous spectrum. Several trials with changing $J\text{AR}$ in applications founds that a simple treatment of fixing $J\text{AR} = 4$ is sufficient in our study.

5 Parameter Estimation Procedure

The time series model presented in previous section can be formulated by a state space model (SSM) [2] as follows:

$$x^{(m)}_n = F^{(m)} x^{(m)}_{n-1} + G^{(m)} v^{(m)}_n,$$

$$e_n = D^{(m)} x^{(m)}_n + w^{(m)}_n.$$
For example, the time series model for $I^{(2)}$ can be represented by the SSM in which the corresponding vectors and matrices are

$$x_0^{(2)} = [t_n^{(2)}, t_{n-1}^{(2)}, q_0, q_{n-1}, r_n, r_{n-1}, r_{n-2}, r_{n-3}]',$$

$$D^{(2)} = [1, 0, 1, 0, 0, 0, 0].$$

$$F^{(2)} = \begin{pmatrix} 2 & -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & C & -1 & 0 & 0 & 0 & 0 \\ 0 & a_1 & a_2 & a_3 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}, \quad G^{(2)} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad v_n^{(2)} = \begin{pmatrix} v_{y_0}^{(2)} \\ v_{y_1}^{(2)} \\ v_{y_2}^{(2)} \\ v_{y_3}^{(2)} \end{pmatrix},$$

where $C = 2 \cos(2\pi f_c)$. Here the empty entries of $F^{(2)}$ and $G^{(2)}$ are all zero and $v_0^{(2)} \sim N(0, R^{(2)})$ with a diagonal variance matrix of $R^{(2)} = \text{diag}(\tau_1^{(2)}, \tau_2^{(2)}, \tau_3^{(2)})$.

An optimal estimation for $t_n^{(2)}$, $q_0$, and $r_n$ is given by the estimated $x_0^{(2)}$ that is obtained by the Kalman filter and smoother [2]. Here $r_1^{(2)}$, $r_2^{(2)}$, and $r_3^{(2)}$ are unknown parameters to be optimized. Then the time series model for $I^{(2)}$ involves nine unknown parameters:

$$\lambda^{(2)} = [\sigma^{(2)}, \tau_1^{(2)}, f_c, \tau_2^{(2)}, \tau_3^{(2)}, a_1, a_2, a_3, a_4]' .$$

(12)

The optimal $\lambda^{(2)}$, $\lambda^{(2)*}$, can be determined by minimizing the log-likelihood, $\ell(\lambda^{(2)}) = \log p(E_{k_1+1:k_1} | \lambda^{(2)})$, where $E_{k_1+1:k_1} = [\varepsilon_{k_1+1}, \ldots, \varepsilon_{k_1}]$ [4]. The AIC value for $I^{(2)}$, $\text{AIC}^{(2)}$, is also defined by

$$\text{AIC}^{(2)} = -2\ell(\lambda^{(2)*}) + 2 \times \text{dim}(\lambda^{(2)}).$$

(13)

Similarly, $\text{AIC}^{(1)}$ and $\text{AIC}^{(3)}$ in (5) are also defined.

6 Result and Summary

Fig. 1 shows one of results of the decomposition obtained by applying our procedure to data sets in each of which a typical Pi2 pulsation is observed. The data set that we examined is the H component measured at Kotel’ney (Russia) from 1996 May 26 16:10.00 17:10.00. The sampling time is a second, and thus $N = 3,600$. The two vertical lines indicate the estimated $k_1^*$ and $k_2^*$, respectively. The three lines are the estimated $r_n$, $q_0$, and original observation $y_n$, from the above, respectively. The horizontal arrow illustrates that the minimum $\text{AIC}_N$ procedure finds an optimal $k_1$. The thick line is the estimated trend component: $\mu_n + t_n^{(m)}$. For this case, three rapid decrease are identified; namely, $J = 3.$
The advantages of applying our procedure are summarized as follows. First, our model for decomposition is robust to a rapid change in trend, and then it gives us a good separation of the Pi2 wave component. Second, the onset time can be objectively determined by minimizing an information criterion, AIC. It turns out that our method is free from the ambiguity of onset time determination. Finally, our procedure is fully automatic.

Acknowledgments. We thank all members of the 210° magnetic meridian network project (P.I. Prof. Yumoto, Kyushu Univ.). The author thanks to Mr. Uozumi for his help to select the data sets.

References
The Role of Choice in Discovery

Judith Devaney

National Institute of Standards and Technology, Gaithersburg MD, 20899-8051, USA, judith.devaney@nist.gov, WWW home page: http://www.itl.nist.gov/div895/savg

1 Introduction

In discovery with real data, one is always working with approximations. Even without noise, one computes over a finite set of unequally spaced numbers, approximating one’s values with this set. With noise, the values are even more uncertain. Along with the ambiguity of the exact values of numbers, there exists Occam’s razor to prefer simpler equations. That is, if a straight line will explain the data, then that is generally thought preferable to equations of higher power. It should be noted that this preference is a choice, however, and does not always work [8]. The preference needs to be codified to be automated, but codifying simplicity is not straightforward [8]. Finally, there needs to be a quantitative way to measure the goodness of an equation after it is chosen. Function finding is numeric induction, so there can never be certainty. But assigning a value to the inductive support provides a way to compare results across tasks. Thus discovery of functional forms can be divided into three tasks: 1— choosing a search technique to find the set of best equations within the limitations of finite precision arithmetic and noise; 2— choosing from among the best equations based on some criteria that encodes preference; and 3— choosing a metric for the inductive support of the found equation.

2 Choices

I. Choice of Search Technique: Equation Signatures

One way to search for equations is with regression, but this confounds the ambiguity of the coefficients with the ambiguity of the type. However, if the equation type is known, finding the coefficients is an already solved problem. So it is sufficient to search for the type. The method to search for types is based on equation signatures [3][4]. Definition Equation Signature: an equation signature is a property of an equation type that is independent of its coefficients and that can be used to identify its type uniquely. It should be noted that searching for types using equation signatures is less error prone not only because only one thing is being searched for at a time, but because signatures are independent of each other and can therefore be productively combined. Some signatures are [3][4]:

1.0 Linear Equations The signature of a linear equation measures flatness. The general linear equation is a hyperplane which has the property that

a unit normal constructed on one side of the plane is equal to a unit normal constructed anywhere else on the same side of the plane. So the variance in the unit normal provides the signature. It is a metric that tells one how close the data is to being a linear equation in whatever dimensional space one is working in. It also makes it easier to mitigate the impact of noise by choosing points far apart for the computation of the unit normal.

2.0 Quadratic Equations The general equation of the second degree in two dimensions is a conic section which consist of the circle, the ellipse, the hyperbola, the parabola, and degenerate forms. In its most general form, the equation of the conic section is \( ax^2 + 2bxy + cy^2 + 2dx + 2ey + f = 0 \), where the more complicated versions of the equation occur when the conic section is rotated and/or translated from the origin. Two example signatures for 2-D quadratics:

1.0 \textit{j-invariant}

A pencil of lines consists of all lines that can be drawn through a given point. The anharmonic or cross ratio of a pencil of lines, whose sides pass through four fixed points of a conic, and whose vertex is any variable point of it, is constant [10]. Thus, if \( V \) is any point on a conic and \( A, B, C, D \) are any four other points on a conic section, as \( V \) is moved around the conic section and \( A, B, C, D \) stay fixed, the ratio stays the same. See Figure 1. Any five points may be chosen for the calculation. For example, they may come from different branches of a hyperbola. This ratio can be computed in terms of distances or slopes [1]. If \( V \) is a point on a conic and \( A, B, C, D \) are four other points on a conic section, and \( \alpha \) is the slope of \( VA \), \( \beta \) is the slope of \( VB \), \( \gamma \) is the slope of \( VC \), \( \delta \) is the slope of \( VD \), then the anharmonic or cross ratio \( a \) is:

\[
\alpha = \frac{(\beta-\gamma)(\delta-\beta)}{(\gamma-\beta)(\alpha-\gamma)}
\]

While the ratio is constant, it is dependent on the order in which the four fixed points are taken. Depending on the order of selection of the points, the ratio can be calculated in twenty-four ways to yield six different ratios [1]. Fortunately, there is a function of the anharmonic ratio that is independent of the order of the points. This is the \textit{j-invariant} [7] which is:

\[
\frac{(1-\alpha)(\beta-\gamma)}{(\gamma-\beta)(1-\alpha)}
\]

Once the four points have been selected, this constant and remains the same constant under any translation or rotation of the plane.

![Fig. 1. The anharmonic or cross ratio of \( V_1 \) relative to \( A, B, C, D \) is equal to the anharmonic ratio of \( V_2 \) relative to \( A, B, C, D \).](image)
rotation of the axes. If a different set of four points is selected it is also constant but takes on a different value.

2.0 Linearity of the parallel chords
As first proved by Appolonius in the third century B.C. [5], a conic section has the property that the midpoints of any set of parallel chords through it are collinear [12]. So the collinearity of the midpoints of parallel chords is a signature of the general equation of the second degree in two variables. The test for collinearity can utilize the signature of the linear equation defined above. It is unlikely that a given set of points will be lined up so that a set of parallel chords will have both points of intersection with the conic section on each chord. However, it is possible to pick a direction and create parallel lines through each of the other points in the data set. Then it is only necessary to know the other point of intersection with the conic section to enable calculation of the midpoint. Pascal’s hexagon theorem run backwards [4] is used to find the other point of intersection. Then calculation of the midpoint of each chord is straightforward. Thus the algorithm to determine if a point set is a conic section is as follows:

Pick two points from the dataset; compute their slope and calculate their midpoint.
Create parallel chords through the other points in the dataset.
Run Pascal’s hexagram construction backwards to obtain the other point of intersection of each chord with the conic section.
Compute the midpoint of each chord.
Use the linear equation signature to determine if these points are collinear.

Figure 2 is an example of this using case 105 from the function-finding dataset [11].

![Fig. 2. Midpoints of a system of parallel chords for case 105 of the function-finding dataset [11.](image)](image)

More signatures for quadratics in two and higher dimensions are given in [4]. One particularly nice attribute of equation signatures is that they make possible the search for higher order equations with lower order methods, e.g. straight lines [4]. Another nice feature of this method is that one can define measures
that mean the same thing from one problem to another. Thus it becomes possible to list the set of good equations. This is important because noise may make it impossible to determine unambiguously the best equation. This leads to the necessity of the next choice.

II. Choice of Equation: à Priori Ordering There are many ways to select an equation from a group of equations. Dorothy Wrinch and Harold Jeffreys [13] argue for an à priori Ordering of Equations. According to them, a simple law is à priori more probable than a complex law. They don’t require the ordering need to be strictly monotonic; it can have branches and loops. Their concept of an explicit choice mechanism has the advantage of showing clearly the preference of one equation over another. Ordering all the equations of physics is a dauntling task. But it is tractable to create an ordering within a given class of equations. This is the approach suggested here. The functions are divided into a number of generic types. Within each type the equations are grouped into equivalence classes. Take the simple generic type: \( g^k = ax^k + b \) The goal is to define an ordering based solely on values of \( k_1 \) and \( k_2 \) that would cover all possibilities and would be intuitively acceptable. Let the exponents be expressed by fractions \( j/l \) where \( j \) and \( l \) are non-negative integers, with \( j \) and \( l \) relatively prime. Then let the sequence be the non-decreasing series of \( j/l \) where the value of the sequence at any point is equal to the sum of the numerator and denominator i.e. \( j + l \); and within each equal value of the sequence, the \( j/l \) are ordered by increasing \( l \). The first few terms of this series are: 0/1, 1/1, 2/1, 1/2, 3/1, .... Their corresponding sequence values are: 1, 2, 3, 4, .... To find the sequence number of any rational number, reduce it until the numerator and denominator are relatively prime, then add the numerator and denominator. The result is the sequence value of the number. The simplicity value (S-value) of an equation will now be defined to be the sum of the sequence values of its exponents after the original exponents have been converted to positive whole numbers. Where composite terms exist, e.g. \( xy \), sum the values of the exponents, then compute the S-value of the result. Is this a reasonable grading? Qualitatively yes. Every equation can appear since every rational number can be expressed this way. Lower degree equations have lower values than higher degree equations. More complex equations have higher values. Now it is possible to compare equations by their S-values. S-value differences of zero or one may not mean much, but larger differences clearly do. This is an ordering for one class of equations. Others are possible. The important point is that there is some predefined ordering. For more complex classes of equations, other encodings are needed. Once there is an ordering, the best can be chosen, but support for it needs to be determined which leads to the next choice.

III. Choice of Evidential Support Method: Concinnity There needs to be a way to quantitatively measure the goodness of a given equation. This is numeric induction, so there can never be certainty. The eliminative and variative method originally proposed by Francis Bacon [9] is used here. This method uses a series of tests of great variety that are designed to
eliminate possibilities. As more and more tests of different types are passed, the result becomes harder to refute. After testing, theories can be graded ordinarily, by the number of tests they passed. Define: **Concinnity Index** This is the number of tests passed by the equation, out of the number of tests performed.

This index is not a single number, and can't be if you want to emphasize its dependence on the number of tests passed. An individual test gauges support for an equation just according to the property the particular test measures. By combining many tests, a harmonious picture of the validity of the equation emerges. This is also a way to say how far the chosen equation is from the next best. If the best can pass ten tests while the next best can only pass one, this provides a measure of the distance between the equations.

Component tests of the Concinnity Index include two types of tests: tests for randomness of residuals, and other tests based on the equation type. Due to much study of random number generators, there are many tests for randomness [6]. Tests include the serial correlation test, a binning test, a run test. Tests based on equation type deal with signatures. If the equation was not found by the signature method, signatures provide a good test. Signatures can also be combined to make a stronger test. For example, the variables in an equation can be linearized and the resulting points tested for linearity. It should be noted that all of these tests can and should also be performed for points outside the interval that the equation was found in. Additionally, point prediction outside the interval is a crucial test.

In summary, choice has been present at every stage of this process.

**References**

Search for New Methods for Assignment of Complex Molecular Spectra

Takehiko Tanaka

Kyushu University, Department of Chemistry, Faculty of Science, Hakozaki 6-10-1, Higashiku, Fukuoka 812-8581, Japan
tata.scc@mbox.nc.kyushu-u.ac.jp

Recent advance of spectroscopic instruments has allowed us to obtain a large amount of spectral data in machine readable forms. High resolution molecular spectra contain abundant information on structures and dynamics of molecules. However, extraction of such useful information necessitates a procedure of spectral assignment in which each spectral line is assigned a set of quantum numbers. This procedure has traditionally been performed by making use of regular patterns that are obviously seen in the observed spectrum. However, we often encounter complex spectra in which such regular patterns may not be readily discerned. The purpose of the present work is to search for new methods which can assist in assigning such complex molecular spectra. We wish to devise computer aided techniques for picking out regular patterns buried in a list of observed frequencies which look like randomly distributed. We hope that we may depend on great computational power of modern computers.

Previously [1,2], we have proposed a method, which we tentatively refer to as "second difference method" and suggested that this technique may be developed as a useful tool for analysis of complex molecular spectra. This method has been tested with success on the observed spectrum of a linear molecule DCCCl [1]. We have also presented a further test using an artificial data corresponding to an infrared spectrum of a linear molecule HCCBr [2]. However, we recently encountered a set of data for which the method in the original form did not work well.

The present article describes a revised algorithm which was developed as a remedy but proved to be a substantial improvement from the original one. The revised algorithm is based on the same basic assumptions as for the original one.

1) A complex spectrum is formed as a result of overlap of many spectral series, each of which has a simple structure.

2) The frequencies of spectral lines belonging to a series may be represented to a good approximation by a quadratic function of a running number.

The second assumption means that if we let \( f(k), k = 1, 2, 3, \ldots \) be frequencies of spectral lines belonging to a series, the second difference

\[
\Delta^2(k) = f(k + 2) - 2f(k + 1) + f(k)
\]

(1)

© Springer-Verlag Berlin Heidelberg 2000
would be almost constant independent of \( k \), and therefore the third difference

\[
\Delta^3(k) = f(k + 3) - 3f(k + 2) + 3f(k + 1) - f(k)
\]

would be very small for all \( k \).

We coded a FORTRAN program briefly described as follows:

1. List of observed frequencies is read in.
2. 3-membered chains are generated. A 3-membered chain is defined as an array of three frequencies \((f_1, f_2, f_3)\) chosen from the list of observed frequencies. To save computation time and memory, the upper and lower limits of the second difference \( f_3 - 2f_2 + f_1 \) as well as those of the first difference \( f_2 - f_1 \) may be given as preset values to restrict the 3-membered chains.
3. From \( n \)-membered chains, \( n+1 \)-membered chains are generated. The method for this will be discussed below in some detail.
4. ⟨3⟩ is repeated until the chain length reaches a preset value.
5. For each chain with the preset length, frequencies are least-squares fitted to a polynomial of a preset order, and standard deviation is calculated.
6. Chains and their standard deviations are listed in the order of ascending standard deviation.

In ⟨3⟩, the method to extend an \( n \)-membered chain \((f_1, \ldots, f_{n-2}, f_{n-1}, f_n)\) to an \( n+1 \)-membered chain is as follows. We calculate

\[
f_{\text{pred}} = f_{n-2} - 3f_{n-1} + 3f_n.
\]

If we find in the list of observed frequencies a frequency which falls between \( f_{\text{pred}} - \Delta f_{\text{allow}} \) and \( f_{\text{pred}} + \Delta f_{\text{allow}} \), it is added to the tail of the \( n \)-membered chain to generate an \( n+1 \)-membered chain. This method increases the chain length by one in such a way that the second differences \( f_n - 2f_{n-1} + f_{n-2} \) and \( f_{n+1} - 2f_n + f_{n-1} \) do not differ by more than \( \Delta f_{\text{allow}} \), which is a preset value. The \( \Delta f_{\text{allow}} \) value will be set to such a magnitude that in a true spectral series \( f(1), f(2), \ldots \) the absolute value of the third difference

\[
\Delta^3(k) = f(k + 3) - 3f(k + 2) + 3f(k + 1) - f(k)
\]

for any \( k \) would not exceed \( \Delta f_{\text{allow}} \).

The spectrum for which the second difference method in the original form was unsuccessful is that of trans-glyoxal. It has been observed by Professor Kato’s group in Kobe University by means of Doppler-free two-photon absorption spectroscopy at a very high resolution [3]. The profile of the spectrum is as follows. It consists of about 1700 spectral lines distributed in the region 22187–22215 cm\(^{-1}\). The lines belong to 23 major series, each containing more than 50 lines, as well as to a large number of minor series.
We applied the present method to this spectrum under the following conditions. We selected 3-membered chains for which first and second differences were between 0 and 1.2 cm$^{-1}$ and between −0.015 and 0 cm$^{-1}$, respectively. The $\Delta f_{\text{allow}}$ value was set to 0.003 cm$^{-1}$. The chain length was preset at 15. The frequencies of the resulting chains were least squares fitted to polynomials of the 4-th order. Then, 7084 chains were obtained with standard deviations of 0.000070 cm$^{-1}$ at the smallest and 0.006340 cm$^{-1}$ at the largest. We inspected 240 15-membered chains with the smallest standard deviations whether each chain was a part of a true spectral series, and found that 130 of them corresponded to true spectral series. There were 75 chains for which only one line was erroneously merged. Similarly, 28, 5, and 2 chains had 2, 3, and 4 erroneously merged lines.

Except for a few cases, it was found that the number of the erroneously merged lines was at most two. Even if there are merged two erroneous lines, it would not be too difficult to arrive at the correct assignment starting from them. Out of 23 major spectral series, 19 were detected by the chains with less than 3 erroneously lines among the 240 chains. These results indicate the usefulness of the revised algorithm.

References

Computational Analysis for Discovery on the Plasma Waves Observed by Scientific Satellites

Yoshiya Kasahara, Ryotaro Nitsu, Yousuke Akimoto, and Toru Sato

Graduate School of Informatics, Kyoto University
Yoshida, Sakyo-ku, Kyoto 606-8501, JAPAN,
E-mail: kasahara@kuee.kyoto-u.ac.jp

1 Introduction

Akebono is a Japanese scientific satellite, which was launched in 1989 for precise observations of the magnetosphere. Akebono is successfully operated for more than 11 years and the accumulated data amounts to about 1 Tbyte of digital data and about 20,000 audiotapes of analogue data. The VLF instruments on-board the Akebono satellite are designed to investigate plasma waves from a few Hz to 17.8 kHz [1, 2]. In the field of conventional geophysics, research is mainly performed by the following way: (1) a scientist discovers observation result as an evidence of a new theory, or (2) a scientist tries to explain a very unique observation result theoretically. In the analysis of physical phenomenon from data observed by one satellite, it is quite difficult to tell whether we see a temporal change, a spatial change or mixture of them. Hence it is indispensable for investigating these phenomena to examine as large amount of data as possible, although they can be clarified to some extent by event study. Our aim is to develop new computational techniques for extracting the attributes of the plasma waves from the enormous data sets of Akebono and to discover epoch-making knowledge.

2 Extraction of the Attributes of Plasma Waves

In the past eleven years, so many kinds of plasma waves were observed in the Earth’s magnetosphere by the Akebono satellite. Some of them are artificial waves propagating from the ground and the others are natural waves generated in the space plasma. As the spectrum of each wave is attributed to various generation mechanisms and propagation modes, the generation/propagation mechanisms of these waves reflect the plasma environment around the earth, which depends on altitude, latitude, and a variety of geophysical parameters such as solar activity, geomagnetic activity, season, and local time. For example, electrostatic broadband low frequency noise is one of wave phenomena frequently observed in the auroral region. Using all wave data obtained by Akebono for nine years, we could get many new findings on the phenomenon; the wave is distributed in the limited latitude region, the region is extended toward the

© Springer-Verlag Berlin Heidelberg 2000
lower latitude while the geomagnetic activity is higher, and the intensity becomes largest in winter and weakest in summer [3]. These findings show that it is quite valuable in deriving a dynamic structure of the magnetosphere to investigate these waves and clarify their characteristics. As the total amount of the data obtained by Akebono is huge, it is necessary to develop a computational technique that efficiently classifies the wave spectra. We made an attempt to classify the plasma waves in a systematic way.

Assuming that the wave intensity of each wave mode is represented by a function of multi-dimensional parameters, we firstly examine the occurrence frequencies of the wave intensity versus principal parameters. In Fig. 1(a), we show an example of contour map which is used for the classification of wave modes. In the figure, the vertical axis indicates wave intensity and the horizontal axis indicates invariant latitude of the observation point, and the occurrence frequency of the wave intensity at 5.62 kHz is represented by contour level. We find that the distribution of the occurrence frequency is then divided into some clusters, which correspond to the different kinds of the wave modes such as auroral hiss, chorus, magnetospheric hiss etc., which are typical wave phenomena in the frequency range around several kHz. We identify the wave mode for each cluster by sampling some data included in it. Using this method, we successfully extracted several kinds of wave modes.

As a next step, parameter dependence on the spatial and temporal distribution of the plasma waves extracted from more than 27,000 data sets are examined statistically. Fig. 1(b) shows the spatial distribution of chorus emission extracted by this method. In the figure, the radius of the circle indicates the radial distance from the Earth’s center and magnetic local time is taken along the circumference. We can find that chorus emission is usually observed larger than the radial distance of 4 \( r_e \), where \( r_e \) is the radius of the Earth, in the magnetic local time range from 0 to 18. The distribution becomes broader and farther in the dayside than in the nightside. This result is a quite interesting finding from a geophysical point of view.

3 Discussion

There are a variety of plasma waves, and the multi-dimensional parameter dependence of these waves reflects the geophysical rules that control the plasma environment around the Earth. Data sets continuously obtained by Akebono for more than eleven years contain a lot of information to discover these rules, whereas increase of the volume of data sets makes us impossible to apply the conventional analysis techniques. In the present paper, we attempt to extract the particular phenomena in a systematic way using the occurrence frequency of the wave intensity. We select the principal parameters arbitrarily from a scientific point of view and we found that our approach is generally agreeable. However, it is still at a preliminary stage and there will be two important items to be developed for the future work; an algorithm for clustering the phenomena by
using the information of the wave spectra, and an auto-compression algorithm of unnecessary dimensions among the many parameters.

References


Fig. 1. (a) Classification of wave modes and (b) an example of spatial distribution of extracted wave modes.
Direction Finding of the Waves in Plasma Using Energy Function

Yoshitaka GOTO, Yoshiya KASAHARA and Toru SATO
Dept. of Communications and Computer Engineering, Kyoto University
Yoshida, Sakyo-ku, Kyoto 606-8501, JAPAN
E-mail: ygoton@aso.cce.i.kyoto-u.ac.jp

1 Introduction

Investigation of space environment around the earth has become a critical issue. In addition to the artificial waves propagating from the ground, a variety of natural plasma waves generated in the magnetosphere are detected by scientific satellites. As the characteristics of these waves reflect the plasma environment around the earth, it is important to measure the propagation (wave normal) directions of these waves. Since signals observed by a satellite may be a mixture of the waves from multiple directions, we must assume unknown parameters of these waves without prior information. In the present study, we propose a new direction finding method derived from the concept of energy function.

2 WDF Method with a Gaussian Distribution Model

The wave distribution function (WDF) is derived from the concept that observed signals can be defined as a distribution of the wave energy density relative to the direction \((\theta, \phi)\). The WDF \(F(\omega, \theta, \phi)\) is related to the spectral matrix by the following equation,

\[
S_{ij} = \frac{\pi}{2} \int_0^{2\pi} \int_0^\pi a_{ij}(\omega, \theta, \phi)F(\omega, \theta, \phi)\sin \theta d\theta d\phi, \quad (i, j = 1, \ldots, 6),
\]

where \(S_{ij}\) are the elements of the spectral matrix calculated from electric and magnetic wave fields at the observation point, and \(a_{ij}(\omega, \theta, \phi)\) are the integration kernels theoretically determined by the plasma parameters. Using the known parameters \(S_{ij}\) and \(a_{ij}\), we can estimate the function \(F\) by solving the set of integral equations (1), but it is an ill-posed problem whose solution may not be unique. In the present study, we propose a Gaussian distribution model in which we assume that observed signal consists of \(m\) clusters of waves whose distribution function \(F\) is represented as \(\alpha_l \exp \left\{ -\left( \frac{d_{\Omega}(\theta, \phi)}{\delta_l} \right)^2 \right\}\), where \(\alpha_l\) is the intensity at the center of distribution \((\theta_l, \phi_l)\), \(d_l\) represents the angular extent of the distribution, and \(d_{\Omega}(\theta, \phi)\) is the angle between \((\theta, \phi)\) and \((\theta_l, \phi_l)\). Unknown parameters \((\theta_l, \phi_l, \alpha_l, d_l)\) are determined by the non-linear least-squares fitting method.

© Springer-Verlag Berlin Heidelberg 2000
In order to get an adequate solution, the number of arrival waves \( n \) and the initial values for \((\theta_l, \phi_l, \alpha_l, d_l)\) must be determined. We assume that the observed spectral matrix \( S_{\text{obs}} \) is composed by a combination of the estimated spectral matrices \( S_{\text{est}} \)

\[
S_{\text{obs}} = \sum_{l=1}^{n} x_l S_{\text{est}} \tag{2}
\]

where \( n \) is the total number of discrete points in \((\theta, \phi)\) space used for the initial value determination. In the condition that no variables \((x_1 \cdots x_n)\) are negative, it is confirmed by the simplex method that there is no solution which satisfies (2). We define the following energy function \( E \) as a dispersion of the ratios of corresponding elements of both sides in (2),

\[
E = \sigma^2 \left( \sum_{l=1}^{n} x_l S_{\text{est}} / S_{\text{obs}} \right). \tag{3}
\]

Applying the steepest descent method combined with the random search method, we can solve the dynamics system of \( E \) and obtain an optimum solution of \((x_1 \cdots x_n)\) which realizes the minimum \( E \). This solution is referred to as the approximate distribution of the WDF for the determination of the initial values of the parameter fitting. This pre-processing is useful for examining the validity of the final solution reconstructed by the Gaussian distribution model.

3 Simulation

The performance of the WDF method with the Gaussian distribution model is evaluated using the computer-generated spectral matrices \( S_{ij} \) calculated by (1) from given wave distribution functions \( F(\omega, \theta, \phi) \). In the simulation, the source wave is assumed to be whistler mode wave at 10kHz, and the plasma frequency and the cyclotron frequency of electron are 60kHz and 400kHz, respectively. Several cases are examined by varying the distribution of arrival waves.

The given and reconstructed wave distributions in the case where \( F \) is composed of two Gaussian distribution with the parameters \((\theta_1, \phi_1, \alpha_1, d_1) = (30^\circ, 60^\circ, 3, 10^\circ) \) and \((\theta_2, \phi_2, \alpha_2, d_2) = (60^\circ, 200^\circ, 1, 30^\circ) \) are shown as Fig. 1. It is found that the wave distributions are successfully reconstructed, and the fitting errors are small enough. In the case where \( F \) is assumed to be distributed along the resonance angle of whistler mode wave, the given and approximate wave distribution at the pre-processing stage are shown as Fig. 2. In this case, the wave distribution is practically reconstructed by the combination of several Gaussian distributions at the pre-processing stage, although the given distribution is non-Gaussian.

4 Results

The direction finding method using the wave distribution function with the Gaussian distribution model is proposed. It is found that the wave distribution is well
reconstructed by a combination of the Gaussian distributions. It is remarkable that the proposed method is applicable to all considered cases where the arrival wave is a point source or a combination of extended sources. The amount of the calculation time is small enough for the practical use.

This method is applied to the data observed by the Akebono satellite, such as Omega signals, whistlers, chorus emissions. The derived wave normal directions are in the acceptable range from theoretical viewpoints.

References

Coping The Challenge of Mutagenes Discovery with GUHA+/- for Windows

Premysl Zak¹ and Jaroslava Halova²

Academy of Sciences of The Czech Republic

1 Institute of Computer Science, Pod vodarenskou vezi 2, CZ 182 07 Prague 8, Czech Republic
zak@cs.cas.cz

2 Institute of Inorganic Chemistry, CZ 250 68 Rez near Prague, Czech Republic
halova@iic.cas.cz

The famous mutagenity data set of 230 compounds [1] encoded by topological descriptors [2] was processed by GUHA method.

Basic ideas of GUHA (General Unary Hypotheses Automaton) method were given in [3] already in 1966. The aim of GUHA method is to generate hypotheses on relations among properties of the objects which are in some sense interesting.

The hypothesis is generally composed of two parts: antecedent (A) and succedent (S). Antecedent and succedent are tied together by so called generalized quantifier, which describes the relation between them. Given antecedent and succedent, frequencies of four possible combinations can be computed and expressed in compressed form as so called four-fold table (ff-table):

<table>
<thead>
<tr>
<th>ff-table</th>
<th>Succedent (S)</th>
<th>Non(succedent)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antecedent (A)</td>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>Non(antecedent)</td>
<td>c</td>
<td>d</td>
</tr>
</tbody>
</table>

Here a is the number of the objects satisfying antecedent and succedent (implication is valid), b is the number of the objects satisfying antecedent but not satisfying succedent (implication is not valid), etc.

The basic generalized quantifier defined and used in GUHA is given by Fisher exact test known from mathematical statistics, and relative frequency PROB=a/(a+b).

In our previous paper [4] we used GUHA for Structure-Activity Relationships (SAR) with the same original data [1] as in this work, encoded by original fingerprint descriptors. Topological descriptors [5] for coding the same data have been used in this work. Mutagenicity data set (230 compounds) [1] was given in two tables. Both coding methods can describe compounds in the same manner, therefore there can be redundancy in the data. This redundancy is inconvenient in the search for Structure-Activity Relationships (SAR), but the used method (GUHA) enables the choice of the best of redundant variables for given dependency relation.

We presented a number of hypotheses [5] discovered by GUHA +/- [6]. The following two hypotheses are interesting from the point of view of toxicology:

- **High number of ortho-peri carbon atoms** ⇒ **high mutagenicity.**
- **High Balaban index** AND **low total negative p** ⇒ **low mutagenicity.**
  
  \( (\text{p} \text{ is number of valence electrons of atom i}) \) [2]

The first hypothesis is identical with the hypothesis in our previous paper [4], where the data was encoded by our fingerprint descriptors.

The next step should be interpretation of these hypotheses and generation of more precise hypotheses including three or more variables in antecedent, in accordance with knowledge of the variables.

Our assumption, that GUHA can be used in the search for interdependencies, seems to be right. We tried to draw dependency graphs of the best hypotheses and they confirmed the trends.

The applicability of GUHA to coding by fingerprint descriptors [4] and topological indices [5] in SAR has been demonstrated. GUHA is able to proceed the whole original data set.

According to the theory of global interpretation of multiple hypotheses testing (see [7] chapter 8) the global significance of our results was considered. From this point of view the results as a whole can be interpreted as sufficiently reliable knowledge on the universe from which the data form a random sample.

**Acknowledgement**

The authors wish to express their cordial thanks to Professor Milon Tichy from National Institute of Public Health, Prague, for his generous help, invaluable advice, thorough discussions of toxicological aspects of the work, and kind encouragement.

**References**

Discovering Interpretable Rules that Explain Customers’ Brand Choice Behavior*

Yukinobu Hamuro¹, Naoki Katoh², and Katsutoshi Yada³

¹ Faculty of Business Administration, Osaka Sangyo University, Daito, Osaka, Japan, email: hamuro@adm.osaka-sandai.ac.jp
² Department of Architecture and Architectural Systems, Kyoto University, Kyoto, Japan, email: naoki@archi.kyoto-u.ac.jp
³ Faculty of Commerce, Kansai University, Suita, Osaka, Japan, email: yada@ipcku.kansai-u.ac.jp

1 Introduction

The problem about brand choice or brand switching has been discussed for a long time in a marketing research field [1][2][3][6]. They focus on revealing a probability of brand switching and what factors are related to the brand switching. However, brand choice behavior of individual customer has been neglected in most of existing literature. In this study, we consider the problem of finding an optimal distribution strategy of discount coupon that determines to which customers and at what price coupons should be distributed, using detailed customer information.

Such a way to distribute a discount coupon based on customer information is available on today’s new technology. For example, Pharma, which is one of the biggest drugstore chains in Japan, developed advanced POS terminal that has a function to issue a customized coupon [5]. Furthermore, in today’s virtual store on the Internet it is possible to set a different price on a different customer.

In view of this, we formulate the problem as two different optimization problems by paying a special attention to the interpretability of rules. We then propose a heuristic method for solving it. Using huge sales data of the biggest drugstore chain in Japan, we have applied the algorithm to the case of detergents. Using the method we found some interesting rules which can be implemented in real business.

It should be pointed out here that the interpretability of the rules obtained from the solution is extremely important because even if the solution is good, the rules obtained will never be implemented in actual business action plan, if they are not interpretable. Rules generated by decision trees such as See5 [7] are usually difficult to interpret from practical point of view.

* Research of this paper is partly supported by the Grant-in-Aid for Scientific Research on Priority Areas (A) by the Ministry of Education, Science, Sports and Culture of Japan. We also express our gratitude to G&G Pharma Inc. for kindly providing us with their sales data.

© Springer-Verlag Berlin Heidelberg 2000
2 Problem Formulation

We formulate the problem of finding an optimal distribution strategy of discount coupons as the following two optimization problems using different criteria under the budget constraint.

(1) Maximization of Customer Gain: The first problem is to maximize the increase of the number of customers (called customer gain) who will newly begin to buy the brand with the coupon under the budget constraint. Let \( q \) denote the number of customers who will purchase the target brand by using a coupon at certain discounting price \( p_1 \), let \( q_0 \) denote the number of customers who will purchase the target brand even if the target brand is sold without discounting. Then the problem is to select customers so as to maximize \( q - q_0 \) under the budget constraint.

(2) Maximization of Effective Cost: In the second problem, we take into the marginal utility of the discount coupon. Suppose that the number of customers who will buy a brand is \( N_p \) at the discount price \( p \). Among \( N_p \) customers there may be those who are willing to buy the brand even at a price \( p' \) higher than \( p \). Let \( N_{p'} \) be the number of such customers. Thus, the cost, \( (p' - p)N_{p'} \) can be regarded as the wasted cost. Subtracting the wasted cost from the cost actually incurred represents the effective cost. The second problem is to maximize the effective cost under the budget constraint.

Suppose we have \( n \) attributes denoted by \( A_1, A_2, \ldots, A_n \). For each attribute \( A_i \), it is assumed that the domain of \( A_i \) (denoted by \( \text{dom}(A_i) \)) is appropriately discretized, i.e., \( \text{dom}(A_i) = \{a_{i1}, a_{i2}, \ldots, a_{is_i}\} \), where \( s_i = \text{dom}(A_i) \). We assume in this formulation that the discount price which is a decision variable is a discrete variable. Associated with attribute \( A_i \), let \( R_i \) be a pair of \( (S_i, d_j) \) where \( S_i \subseteq \text{dom}(A_i) \) and \( d_j \) is the discount price. The meaning of \( R_i \) is that we distribute a discount coupon at the price \( d_j \) to the customers whose attribute \( A_i \) takes the value in \( S_i \). Then the rule \( R \) is a union of \( R_i \) expressed as \( R = \bigcup_{i=1}^{n} R_i \).

Namely, the customers selected by the rule is those for which there exists at least one attribute such that attribute \( A_i \) of a customer takes the value in \( S_i \) for some \( i \). This rule is called 1D rule since we construct rules by considering attributes separately. We can easily extend it to 2D rule. Then letting \( u_1(R) \) denote as the number of \( q - q_0 \) that satisfy the rule \( R \), the first problem is formulated as

\[
P1 : \text{maximize } u_1(R) \text{ subject to } C(R) \leq B,
\]

where \( B \) is a given budget and \( C(R) \) is the total cost incurred by implementing rule \( R \). Letting \( u_2(R) \) denote the effective cost for a given rule \( R \), the second problem is formulated as

\[
P2 : \text{maximize } u_2(R) \text{ subject to } C(R) \leq B.
\]

\(^1\) We define a discounting price by two different ways, that are relative discounting price in time sequence and relative discounting price between two brands.
3 Greedy Algorithm

Although NP-completeness is not yet proven, problem P1 seems difficult. Problem P2 is NP-complete because the simpler version where there exist only two attributes and there is no redemption cost is equivalent to the one discussed in [4] which was shown to be NP-complete. Hence we propose a heuristic algorithm. Let us first consider the problem P1 for 1D rule. Let \( d_0, d_1, \ldots, d_m \) be the set of possible discount values, where \( d_0 \) means no discount. For attribute \( A_h \), and for \( a_i \in \text{dom}(A_h) \), let us focus on the customer group \( G_i \) whose \( A_h \) takes \( a_i \). Let \( g_i = |G_i| \). Let \( p_{ij} \) for \( 0 \leq j \leq m \) denote the probability that a customer in \( G_i \) purchases the brand when the discount is \( d_{ij} \). Then the expected customer gain is \( (p_{ij} - p_0)g_i \), and the expected total cost spent for group \( G_i \) is \( (d_jp_{ij} + c_d)g_i \), where the first term in the parenthesis represents expected redemption cost per customer and the second the distribution cost. Let \( r_{ij} \) denote the ratio of the expected customer gain to the expected total cost i.e., \( r_{ij} = (p_{ij} - p_0)/(d_jp_{ij} + c_d) \). If \( r_{ij} \) is large, the discount by \( d_j \) for customer group \( G_i \) is effective. Thus, we compute \( r_{ij}^* = \max_{1 \leq j \leq m} r_{ij} \). We compute such \( r_{ij}^* \) for every customer group in every attribute. Our algorithm selects the customer group that attains the highest \( r_{ij}^* \) and the rule \((G_i, d_j)\) is adopted, that is, a coupon of discount \( d_j \) will be sent to all customers of \( G_i \). In order to eliminate the possibility that a customer receives two coupons, we ignore the customers of \( G_i \) in succeeding process. We repeat this process until the whole budget is consumed. For problem P2, the algorithm is essentially the same as the one for P1 except that the objective function is replaced by the effective cost. We now formally define the effective cost. Let us consider the customer group \( G_i \) as before. Then the expected number of customers in \( G_i \) who start to purchase the brand only after the discount becomes \( d_{ik} \) is \( (p_{ik} - p_{i,k-1})g_i \). Thus, the wasted cost for such customers is \( (d_j - d_{ik})(p_{ik} - p_{i,k-1})g_i \). Summing this cost over all possible discount values, we can compute the total wasted cost as \( \sum_{k=1}^{j-1}(d_j - d_{ik})(p_{ik} - p_{i,k-1})g_i \). Then the effective cost is written as \( d_jp_{ij} - \sum_{k=0}^{j-1}(d_j - d_{ik})(p_{ik} - p_{i,k-1})g_i \). We can similarly describe the algorithm for 2D rules, but we omit it here.

4 Experimental Results and Observations

Using the greedy algorithm we showed, we analyze the real purchase data from the drugstore chain, which contains 30,358 customers’ purchasing history in two years from 1996 to 1998. We focus on the product category of laundry detergent, and we select the customers who purchased the laundry detergent for previous two years. There are three major brands in Japan referred by “Brand1”, “Brand2” and “Brand3”. All results below are from the Brand1’s point of view. The data set consists of three categories of attributes, which are 1) the target attribute that takes 1 if a customer bought the target brand (Brand1) otherwise 0, 2) discount price on that purchasing, and 3) predictive attributes such as age, brand loyalty, the number of visits, which are based on the cus-
tomer’s basic information or the purchase history. We have applied the greedy algorithm to the data set. The results are summarized as follows.

a) Derived Rules: Table 1 shows the top three rules obtained by solving $P_1$ for distribution cost of 100 yen. The first rule, for example, reads that if we issue coupons whose discount price is lower than usual by 100 yen to the customers whose purchased share of Brand2 is in a range of 50% to 58% and purchased share of a daily necessity is in a range of 29.9% to 41.4%, then purchasing probability increases from 17.8% to 87.9%. These rules are very meaningful and interpretable because we can understand what type of customers change to the target brand at what discount price.

b) 1D rule vs. 2D rule vs. Random sampling: Figure 1 shows how customer gain increases as the budget increases for the solutions of $P_1$ obtained by one-dimensional (1D) rule, two-dimensional (2D) rule, and random sampling. As we expected, 2D rules are superior to 1D rules which are superior to random sampling.

c) Training accuracy: To measure the accuracy of the rules we build, we separate the original data into training and test sets. We then apply the rules generated by using training data against test data. Figure 2 shows the result that accuracy (difference between training and test) of 1D rules are better than 2D rules because overfitting is high for 2D.

5 Conclusion

We have formulated the problem of determining to which customers and at what price the manufacturer should distribute discount coupons as two optimization problems, and we developed a heuristic method, which can be easily implemented in real business field. The experimental result shows that our method produces rules which are meaningful and interpretable for real business world.

References

Discovering Interpretable Rules that Explain Customers' Brand Choice Behavior

Table 1. Examples of the 2D rules obtained by solving \( P1 \) for distribution cost=100

<table>
<thead>
<tr>
<th>Type of Customers</th>
<th>Discount Price</th>
<th>Probability Gain</th>
<th>Customer Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>29.9% \leq \text{Share of Daily necessity} \leq 41.4% \text{ cheaper than usual by 100}</td>
<td>17.8% \text{--} 87.9%</td>
<td>208</td>
<td></td>
</tr>
<tr>
<td>Share of Brand_1=0 \text{ cheaper than Brand_2 by 200}</td>
<td>11.0% \text{--} 76.1%</td>
<td>365</td>
<td></td>
</tr>
<tr>
<td>-3198 \leq \text{Profit} \leq 23 \text{ cheaper than usual by 87}</td>
<td>24.5% \text{--} 82.4%</td>
<td>312</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 1. Effect of 1D, 2D rules and random sampling

Fig. 2. Training accuracy on 1D and 2D rules
Mining for 4ft Association Rules

Jan Rauch, Milan Šimůnek

Faculty of Informatics and Statistics, University of Economics, W. Churchill Sq. 4, 13067 Prague, Czech Republic, e-mail: rauch@vse.cz, xsimm16@vse.cz

1 Introduction

An association rule [1] is an expression of the form \( X \rightarrow Y \) where \( X \) and \( Y \) are sets of items. The intuitive meaning is that transactions (e.g. supermarket baskets) containing set \( X \) of items tend to contain set \( Y \) of items. Two measures of intensity of association rule are used, confidence \( C \) and support \( S \).

A 4ft association rule is an expression of the form \( \varphi \sim \psi \) where \( \varphi \) and \( \psi \) are derived Boolean attributes. The intuitive meaning is that \( \varphi \) and \( \psi \) are in relation given by the symbol \( \sim \). This symbol is called 4ft quantifier.

Our first goal is to introduce 4ft association rules and to argue for their usefulness, see section 2. There are various classes of 4ft association rules, e.g. equivalency rules, double implication or implication rules. There are also 4ft association rules corresponding to statistical hypotheses tests, e.g. to \( \chi^2 \). We show that association rule is a special case of 4ft association rules. We will also define conditional 4ft association rules, see section 3.

The second goal is to introduce procedure 4ft-Miner mining both for 4ft association rule and for conditional 4ft association rule. This procedure is introduced in section 4.

2 4ft Association Rules

A 4ft association rule is an expression \( \varphi \sim \psi \) where \( \varphi \) and \( \psi \) are derived Boolean attributes. Boolean attributes \( \varphi \) and \( \psi \) correspond to \{0,1\}-columns of analysed data matrix \( M \) (a database relation). They are derived from original columns \( A_1, \ldots, A_K \) of \( M \), see Tab. 1.

We suppose that \( a_{1,1} \) is the value of attribute \( A_1 \) for object \( o_1 \), \( a_{n,K} \) is the value of attribute \( A_K \) for object \( o_n \), etc. Relation \( \varphi \sim \psi \) is evaluated on the basis of 4ft table, see Tab. 2.

<table>
<thead>
<tr>
<th>object</th>
<th>( A_1 )</th>
<th>( A_2 )</th>
<th>\cdots</th>
<th>( A_K )</th>
<th>( \varphi )</th>
<th>( \psi )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( o_1 )</td>
<td>( a_{1,1} )</td>
<td>( a_{1,2} )</td>
<td>\cdots</td>
<td>( a_{1,K} )</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\cdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
</tr>
<tr>
<td>( o_n )</td>
<td>( a_{n,1} )</td>
<td>( a_{n,2} )</td>
<td>\cdots</td>
<td>( a_{n,K} )</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ \begin{array}{c|cc}
\varphi & \psi & \neg \psi \\
\hline
a & b & c \\
\hline
\end{array} \]

Tab. 1 - Data matrix \( M \) Tab. 2 - 4ft table of \( M \), \( \varphi \) and \( \psi \)

© Springer-Verlag Berlin Heidelberg 2000
Here $a$ is the number of objects (rows of $M$) satisfying both $\psi$ and $\varphi$, $b$ is the number of objects satisfying $\psi$ and not satisfying $\varphi$, etc. The quadruple $<a,b,c,d>$ is the 4ft-table of $\varphi$ and $\psi$ in $M$, we abbreviate it by $4ft(M,\varphi,\psi)$.

Various kinds of dependencies of $\varphi$ and $\psi$ can be expressed by suitable conditions concerning 4ft table. Each condition corresponds to the 4ft-quantifier. We say that $\varphi \sim \psi$ is true in data matrix $M$ (symbolically $Val(\varphi \sim \psi \ , M) = true$) if the corresponding condition concerning 4ft-table of $\varphi$ and $\psi$ in $M$ is satisfied.

Examples (see [2], [4]):

- 4ft-quantifier $\equiv_{p,a}$ of $p$-equivalence for $0 < p \leq 1$ and $s > 0$ is defined by the condition $\frac{a}{a+b+c+d} \geq p \land a \geq s$. It means that $\varphi$ and $\psi$ have the same value (either true or false) for at least $100p$ per cent of all objects of $M$ and that there are at least $s$ objects of $M$ satisfying both $\varphi$ and $\psi$.

- 4ft-quantifier $\Leftrightarrow_{p,a,s}$ of founded double implication for $0 < p \leq 1$ and $s > 0$ is defined by the condition $\frac{a}{a+b+c+d} \geq p \land a \geq s$. It means that at least $100p$ per cent of objects of $M$ satisfying $\varphi$ or $\psi$ satisfy both $\varphi$ and $\psi$ and that there are at least $s$ objects of $M$ satisfying both $\varphi$ and $\psi$.

- 4ft-quantifier $\Rightarrow_{p,a,s}$ of founded implication for $0 < p \leq 1$ and $s > 0$ is defined by the condition $\frac{a}{a+b+c+d} \geq p \land a \geq s$. It means that at least $100p$ per cent of objects of $M$ satisfying $\varphi$ satisfy also $\psi$ and that there are at least $s$ objects of $M$ satisfying both $\varphi$ and $\psi$.

- 4ft-quantifier $\Rightarrow_{p,a,s}^{\downarrow}$ of lower critical implication for $0 < p \leq 1$, $0 < \alpha < 0.5$, and $s > 0$ is defined by the condition $\sum_{i}^{a+b_i} (a+b_i)p^i(1-p)^{a+b_i-i} \leq \alpha \land a \geq s$. It corresponds to a statistical test (on the level $\alpha$) of the null hypothesis $H_0 : P(\psi|\varphi) \leq p$ against the alternative one $H_1 : P(\psi|\varphi) > p$. Here $P(\psi|\varphi)$ is the conditional probability of the validity of $\psi$ under the condition $\varphi$.

"Classical" association rule $X \rightarrow Y$ can be also understood as the 4ft association rule. Let us suppose that $A_1, \ldots, A_K$ are Boolean attributes corresponding to particular items, $a_{1,1} = 1$ means that supermarket basket $a_1$ contains item $A_1$, etc., see Tab. 1. Then, association rule $\{A_1, A_2\} \rightarrow \{A_3, A_4\}$ with confidence $C$ and support $S$ can be understood as 4ft association rule $A_1 \land A_2 \rightarrow_{C,S} A_3 \land A_4$. The condition $\frac{a}{a+b+c+d} \geq C \land \frac{a}{a+b+c+d} \geq S$ concerning 4ft table corresponds to 4ft quantifier $\rightarrow_{C,S}$ of "classical" association rule.

Let us emphasise differences among 4ft quantifiers $\equiv_{p,a}$, $\Leftrightarrow_{p,a,s}$, $\Rightarrow_{p,a,s}$ and $\Rightarrow_{C,S}$. We will use examples with $p = 0.9$, $s = 100$, $C = 0.9$ and $S = 0.1$, Boolean attributes $\varphi$, $\psi$ and three data matrices $M_1$, $M_2$, $M_3$ with different 4ft tables, see Tab. 3, Tab. 4 and Tab. 5. It is easy to verify that values $Val(\varphi \sim \psi \ , M)$ of particular 4ft association rules are according to Tab. 6. We suppose $M = M_1$, $M = M_2$ or $M = M_3$.

<table>
<thead>
<tr>
<th>$M_1$</th>
<th>$\varphi$</th>
<th>$\neg \varphi$</th>
<th>$\psi$</th>
<th>$\neg \psi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varphi$</td>
<td>100</td>
<td>100</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$\neg \varphi$</td>
<td>1</td>
<td>1</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

Tab. 3 - 4ft($M_1, \varphi, \psi$)

<table>
<thead>
<tr>
<th>$M_2$</th>
<th>$\varphi$</th>
<th>$\neg \varphi$</th>
<th>$\psi$</th>
<th>$\neg \psi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varphi$</td>
<td>100</td>
<td>100</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$\neg \varphi$</td>
<td>1</td>
<td>1</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

Tab. 4 - 4ft($M_2, \varphi, \psi$)

<table>
<thead>
<tr>
<th>$M_3$</th>
<th>$\varphi$</th>
<th>$\neg \varphi$</th>
<th>$\psi$</th>
<th>$\neg \psi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varphi$</td>
<td>100</td>
<td>100</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$\neg \varphi$</td>
<td>100</td>
<td>100</td>
<td>000</td>
<td>000</td>
</tr>
</tbody>
</table>

Tab. 5 - 4ft($M_3, \varphi, \psi$)
Let us remark that logical calculi formulae of which correspond to 4ft association rules were defined and studied. Various useful theoretical results were achieved concerning among others deduction rules, see [2], [4]. Several classes of 4ft association rules were also defined and studied. 4ft-quantifier $\equiv_{p,s}$ is an example of equivalence quantifiers, $\Rightarrow_{p,s}$ is an example of double implication quantifiers, $\Rightarrow_{p,\alpha,s}$ are examples of implication quantifiers.

3 Conditional 4ft Association Rules

A 4ft conditional association rule is an expression of the form $\varphi \sim \psi / \chi$ where $\varphi$, $\psi$ and $\chi$ are derived Boolean attributes. The intuitive meaning is that $\varphi$ and $\psi$ are in relation given by 4ft quantifier $\sim$ when the condition $\chi$ is satisfied.

If $M$ is data matrix and $\chi$ is a Boolean attribute derived from columns of $M$ then we mean by $M/\chi$ a data matrix containing exactly all rows of $M$ satisfying $\chi$. We say that $\varphi \sim \psi / \chi$ is true in data matrix $M$ (symbolically $Val(\varphi \sim \psi / \chi, M) = true$) if both there is the row of $M$ satisfying $\chi$ and if $Val(\varphi \sim \psi, M/\chi) = true$.

There is no general relation between $Val(\varphi \sim \psi, M)$ and $Val(\varphi \sim \psi / \chi, M)$. Let values of $\varphi$, $\psi$ and $\chi$ at data matrix $M_4$ be according to Tab. 7. Then 4ft tables $4ft(M_4, \varphi, \psi)$ and $4ft(M_4/\chi, \varphi, \psi)$ are in Tab. 8 and Tab. 9, thus $Val(\varphi \Rightarrow_{0.9,100} \psi, M_4) = 0$ and $Val(\varphi \Rightarrow_{0.9,100} \psi / \chi, M_4) = 1$. It is easy to modify $M_4$ such that $Val(\varphi \Rightarrow_{0.9,100} \psi, M_4) = 1$ and $Val(\varphi \Rightarrow_{0.9,100} \psi / \chi, M_4) = 0$.

<table>
<thead>
<tr>
<th>objects</th>
<th>$\psi$</th>
<th>$\chi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$o_1 = o_{100}$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$o_{101}$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$o_{102} = o_{200}$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$o_{201}$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$o_{202} = o_{301}$</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Tab. 7 - Data matrix $M_4$  Tab. 8 - $4ft(M_4, \varphi, \psi)$  Tab. 9 - $4ft(M_4/\chi, \varphi, \psi)$

4 Procedure 4ft-Miner

Procedure 4ft-Miner mines for 4ft association rules $\varphi \sim \psi$ and for conditional 4ft association rules $\varphi \sim \psi /$. 4ft quantifiers of fourteen types can be used, see 4.1. Derived Boolean attributes $\varphi$, $\psi$ and $\chi$ are conjunctions of basic Boolean attributes, they are called antecedent succedent and condition respectively, see paragraph 4.2. Input and output of 4ft-Miner are described in paragraphs 4.3 and 4.4. Some further features of 4ft-Miner are mentioned in paragraph 4.5.
4.1 4ft-quantifiers of 4ft-Miner

Fourteen types of 4ft quantifiers are implemented in procedure 4ft-Miner. All of them are defined by the condition concerning 4ft table. There are three implication quantifiers: $\Rightarrow_{p,s}$, $\Rightarrow_{p,\alpha,s}$ (see section 2) and the 4ft quantifier $\Rightarrow_{p,\alpha,s}$ of upper critical implication defined by a condition $\sum_{i=0}^{a_i} (a_i+b_i)p_i(1-p_\alpha)^{a_i+b_i-i} > \alpha \wedge a \geq s$.

Further there are three analogous double implication quantifiers: $\Leftrightarrow_{p,s}$ (see section 2), $\Leftrightarrow_{p,\alpha,s}$ and three analogous equivalence quantifiers: $\equiv_{p,s}$ (see section 2), $\equiv_{p,\alpha,s}$, see also [2], [4].

There are also quantifiers corresponding to $\chi^2$ test and to Fisher’s test and the 4ft quantifier defined by the condition $ad > e^\delta$ where $\delta > 0$. Quantifier $\rightarrow_{C,S}$ corresponding to "classical" association rule (see section 2) is also implemented. Last implemented type of 4ft quantifiers is the 4ft quantifier corresponding to condition $\max(\frac{b_i}{a_i+b_i}, \frac{c_i}{a_i+c_i}) < \gamma$ where $0 \leq \gamma < 1$, see [6].

4.2 Antecedent, Succedent and Condition

Antecedent, succedent and condition are conjunctions of basic Boolean attributes. Basic Boolean attribute is of the form $A[\sigma]$ where $A$ is the column and $\sigma$ is a subset of a set of possible values of $A$. Boolean attribute $A[\sigma]$ is true in the row $o$ of analysed data matrix if the value in the column $A$ and in the row $o$ belongs to the set $\sigma$. Thus, if $A_1[3,5]$ is true in row $o_1$, then $a_{1,1} = 3$ or $a_{1,1} = 5$. An example of the conditional 4ft association rule:

$A_1[1,3,4] \wedge A_3[5,6] \Leftrightarrow_{0.9} A_5[8,12] \wedge A_7[11,12,14] \wedge A_8[2] \wedge A_{10}[4,5,6,7,8]$.

4.3 Input of 4ft-Miner

An input of 4ft-Miner is given by: (i): The analysed data matrix. (ii): 4ft quantifier. (iii): Simple definition of all antecedents to be automatically generated. It consists of: (iii-a): A list of all columns of data matrix, from which basic Boolean attributes of antecedent will be automatically generated. (iii-b): Simple definition of the set of all basic Boolean attributes to be automatically generated from each particular column. (iii-c): Minimal and maximal number of basic Boolean attributes in each generated antecedent. (iv): Analogous definitions of all succedents and of all conditions to be automatically generated.

The set of all basic Boolean attributes to be generated from a particular column is given by a type of subsets and by the minimal and the maximal number of particular values in the subset. There are five types of subsets to be generated: all subsets, intervals, left cuts, right cuts and cuts.

Examples of a definition of the set of basic Boolean attributes for column $A$ with possible values $\{1,2,3,4,5\}$: (1) all subsets with 2-3 values defines basic Boolean attributes $A[1,2], A[1,3], \ldots, A[4,5], A[1,2,3], A[1,2,4], \ldots, A[3,4,5]$; (2) intervals with 2-3 values defines basic Boolean attributes $A[1,2], A[2,3], A[3,4], A[4,5], A[1,2,3], A[2,3,4]$ and $A[3,4,5]$; (3) left cuts with most 3 values defines basic Boolean attributes $A[1], A[1,2], A[1,2,3]$.
4.4 Output of 4ft-Miner

4ft-Miner automatically generates all 4ft association rules or all conditional 4ft association rules given by the conditions (ii) - (v) (usually $10^5 - 10^7$) and verifies them in data matrix given by (i). Output of 4ft-Miner is the set of all 4ft association rules (all conditional 4ft association rules) true in data matrix given by (i).

Usual output of 4ft-Miner consists of tens or hundreds of true 4ft association rules (true conditional 4ft association rules). There are strong tools for dealing with output of 4ft-Miner. It is possible to sort output 4ft association rules by various criterions. Flexible conditions can be used to define subsets of output 4ft association rules. It is also possible to export defined subsets in several formats.

4.5 Some Further Features of 4ft-Miner

4ft-Miner works under WINDOWS, analysed data matrix can be stored in a database (ODBC is applied). New values can be defined for particular columns (e.g. intervals or groups of original values). New columns of data matrix can be also defined (SQL - like) and used in conditions (iii) - (v).

4ft-Miner works very fast. Usual task (data matrix with $10^4$ rows, several millions of 4ft association rules to be generated and verified) requires only several minutes at PC with Pentium II and with 128 MB of operational memory. Several optimisation techniques and deep theoretical results are used, e.g. bit strings for representation of analysed data matrix [3] and deduction rules [4].

Let us remark that 4ft-Miner can deal with missing information [2], [5]. Let us also emphasise that 4ft-Miner is a GUHA procedure in the sense of [2].

References


This work is supported by grant 47160008 and by project LNB00107 of the Ministry of Education of the Czech Republic.
Rule Discovery Technique Using Genetic Programming Combined with Apriori Algorithm

Ayahiko Niimi¹ and Eiichiro Tazaki¹

Department of Control & Systems Engineering, Toin University of Yokohama
1614 Kurogane-cho, Aoba-ku, Yokohama 225-8502, JAPAN

tazaki@intlab.toin.ac.jp

1 Introduction

Various techniques have been proposed for rule discovery using classification learning. In general, the learning speed of a system using genetic programming (GP) [1] is slow. However, a learning system which can acquire higher-order knowledge by adjusting to the environment can be constructed, because the structure is treated at the same time.

On the other hand, there is the Apriori algorithm [2], a rule generating technique for large databases. This is an association rule algorithm. The Apriori algorithm uses two values for rule construction: a support value and a confidence value. Depending on the setting of each index threshold, the search space can be reduced, or the candidate number of association rules can be increased. However, experience is necessary for setting an effective threshold.

Both techniques have advantages and disadvantages as above. In this paper, we propose a rule discovery technique for databases using genetic programming combined with the Apriori algorithm. By using the combined rule generation learning method, it is expected to construct a system which can search for flexible rules in large databases.

2 Proposed Rule Discovery Technique

We propose a rule discovery technique which combines GP with the Apriori algorithm. By combining each technique, it is expected to increase the efficiency of the search for flexible rules in large databases.

The following steps are proposed for the rule discovery technique.

1. First, the Apriori algorithm generates the association rule.
2. Next, the generated association rules are converted into decision trees which are taken in as initial individuals of GP. The decision trees are trained by GP learning.
3. The final decision tree is converted into classification rules.

This allows effective schema to be contained in the initial individuals of GP. As a result, it is expected to improve the GP’s learning speed and its classification accuracy.

© Springer-Verlag Berlin Heidelberg 2000
For conversion from the association rule into decision trees, we use the following procedures.

1. For the first process, the route of the decision tree is constructed, assuming the conditions of the association rule as the attribute-based tests of the decision tree.
2. In the next process, the conclusions of the association rule is appended on the terminal node of this route.
3. Finally, the terminal nodes which are not defined by the association rule are assigned candidate nodes at random.

For conversion from the GP’s decision tree to the classification rule, we use the process proposed by Quinlan [3].

3 Experiments

To verify the validity of the proposed method, we applied it to the house-votes data from UCI Machine Learning Repository [4], and medical database for occurrence of hypertension [5]. From here on all occurrence of GP uses Automatically Defined Function Genetic Programming (ADF-GP) [1] including the proposed method. In the proposed method, we took the association rule generated by Apriori algorithm as initial individuals of GP. We compared the results of the proposed method against GP. We use house-votes database as small test database expressed by discrete values, and hypertension database as large test expressed by continuous values.

3.1 Application to house-votes Data

For evaluation, we used house-votes data from UCI Machine Learning Repository [4]. We compared the results of the proposed method with GP. The evaluation data contains 16 attributes and 2 classes. The attributes are for example “handicapped-infants” and “water-project-cost-sharing” etc. They are expressed by 3 values: “y”, “n”, and “?”. And the 2 classes are “democrat” and “republican”. 50 cases out of the total 435 data of house-votes were used for training data.

We extracted the association rule from the database by the Apriori algorithm. We applied the Apriori algorithm to a data set excluding data with the “?” value, because “?” value means “others”. In the following experiment, we used minimum support value (= 30) and minimum confidence value (= 90). As a result of the experiment, 75 rules were generated.

Next, the above generated 75 rules were taken into the initial individual. The result of the evaluation of each fitness value is shown in figure 1, and the result of best individual is shown in table 1.

By using GP, inference accuracy did not improve rapidly. However, the proposed method showed fast learning and achieved high accuracy. Comparing the
best individual results, the proposed method showed better results than GP, except for accuracy against the training data. Concerning the results of training data, GP may have shown overfitting, but its proof could not be obtained by only this result.

The rules were converted from the constructed decision tree removing invalid rules and meaningless rules. The rules’ total accuracy was 94.8%.

### 3.2 Application to Medical Database

We applied a medical diagnostic system for the occurrence of hypertension. We compared the results of proposed method with GP. Most of the data values are expressed as continuous values, and the size of the database is larger than the house-votes database.

The occurrence of hypertension database contains 15 input terms and 1 output term. There are 2 kinds of intermediate assumptions between the input terms and the output term[6]. Among the input terms, 10 terms are categorized into a biochemical test related to the measurement of blood pressure for past five years, and the other terms are “Sex”, “Age”, “Obesity Index”, “γ-GTP”, and “Volume of Alcohol Consumption”. 1 output term represents whether the patient has an attack of hypertension for the input record. The database has

![Fig. 1. Evaluation of Each Fitness Value](image)

**Table 1.** Experiment Best Individuals Result (House-votes).

<table>
<thead>
<tr>
<th>method</th>
<th>training (%)</th>
<th>all data (%)</th>
<th>nodes</th>
<th>depths</th>
<th>generations</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADF-GP</td>
<td>100.0</td>
<td>86.0</td>
<td>11</td>
<td>3</td>
<td>386</td>
</tr>
<tr>
<td>Apriori +</td>
<td>98.0</td>
<td>92.9</td>
<td>9</td>
<td>2</td>
<td>235</td>
</tr>
</tbody>
</table>

---

Rule Discovery Technique using Genetic Programming
1024 patient records. In this paper, we selected 100 occurrence data and 100 no-occurrence data by random sampling, and this was used as the training data.

The association rule has been extracted from the database by the Apriori algorithm. The Apriori algorithm was used after these attributes had been converted into binary attributes using the average of each data, because the continuous value attributes were included in this database. To search for the relationship between the minimum support value and the minimum confidence value and the number of rules, we experimented with the threshold patterns. (Refer to the result table 2) In the following experiment, we used minimum support value (= 30) and minimum confidence value (= 90).

Table 2. Relations between Thresholds and Number of Rules

<table>
<thead>
<tr>
<th>Minimum Support Value</th>
<th>Minimum Confidence Value</th>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>75</td>
<td>396</td>
</tr>
<tr>
<td>30</td>
<td>75</td>
<td>125</td>
</tr>
<tr>
<td>25</td>
<td>90</td>
<td>187</td>
</tr>
<tr>
<td>30</td>
<td>90</td>
<td>33</td>
</tr>
</tbody>
</table>

Next, the 33 rules generated by the Apriori algorithm were taken into the initial individual. The result of best individual is shown in table 3.

By using GP, inference accuracy did not improve rapidly. However, the proposed method showed fast learning and achieved high accuracy.

Table 3. Experiment Best Individuals Result (Hypertension).

<table>
<thead>
<tr>
<th></th>
<th>training (%)</th>
<th>all data (%)</th>
<th>nodes</th>
<th>depths</th>
<th>generations</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADF-GP</td>
<td>89.5</td>
<td>66.3</td>
<td>41</td>
<td>6</td>
<td>18553</td>
</tr>
<tr>
<td>Apriori + ADF-GP</td>
<td>89.5</td>
<td>74.9</td>
<td>49</td>
<td>4</td>
<td>671</td>
</tr>
</tbody>
</table>

When the rules were converted from the decision tree, invalid rules and meaningless rules were removed. Each ratio of the number of effective rules to generation rules was 37.5% (by GP) and 50.0% (by proposed method). (Table 4 shows 3 rules generated with each technique, chosen by the highest support value.)

By using GP, many invalid rules and many rules which were difficult to interpret were generated. Compared to GP, the proposed method showed decrease in the support value and improvement in accuracy. The proposed technique improved the ratio of effective rules and the accuracy.
Table 4. Comparison of Generated Rules (Size and Inference Accuracy)

<table>
<thead>
<tr>
<th>Technique</th>
<th>Size</th>
<th>Support Value (%)</th>
<th>Wrong (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADF-GP</td>
<td>4</td>
<td>41.4</td>
<td>48.4</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>36.0</td>
<td>24.1</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>22.6</td>
<td>8.2</td>
</tr>
<tr>
<td>Apriori+ADF-GP</td>
<td>2</td>
<td>17.7</td>
<td>30.2</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>15.5</td>
<td>13.2</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>13.8</td>
<td>19.2</td>
</tr>
</tbody>
</table>

4 Concluding Remarks

In this paper, we proposed a rule discovery technique for databases using genetic programming combined with association rule algorithms.

In the future, we will study the following 4 topics related to the proposed method. The first topic is to apply the method to other verifications. The second topic is to further research the conversion algorithm from the association rule to a decision tree with high accuracy. The third topic is to extend the proposed method to multi-value classification problems. The fourth topic is to do a theoretical analysis about the mechanism of the overfitting.

References

Discovery of $M$-of-$N$ Concepts for Classification

Nobuhiro Yugami

Fujitsu Laboratories
1-9-3 Nakase, Minamaku, Chiba, 261-8588, Japan
yugami@flab.fujitsu.co.jp

1 Introduction

The purpose of knowledge discovery system is to discover interesting patterns in a given database. There exist many types of patterns and this paper focuses on discovery of classification rules from a set of training instances represented by attribute values and class labels. A classification rule restricts values of attributes in its body and predicts a class of an instance that satisfies the body. In usual, a body is a conjunction of conditions on attribute values. This paper deals with a different type of rule whose body is a threshold function and requires at least $m$ of $n$ conditions in it are satisfied. Such kind of rules have much more representation power than rules with conjunctive bodies and are suitable for many real world problems such as diagnoses of diseases in which observation of more symptoms of a certain disease leads more confident diagnosis[3, 8].

For $m$-of-$n$ concepts, small threshold for many conditions, i.e. small $m$ for large $n$, easily achieves large support. This property drastically increases the number of rules with large support and high accuracy, and makes it difficult to select interesting rules with fixed lower bounds of support and accuracy. This paper extends the interestingness of classification rules proposed in [7] and tries to resolve this problem by evaluating rules’ support and accuracy with those of simpler rules.

2 Classification Rules

A classification rule is a rule that restricts values of attributes in its body and classifies an instance that satisfies the body into the class in its head. In usual, a body is a conjunction of conditions on attribute values and requires all conditions in it are satisfied. This paper deals with a different type of classification rule such as

$$R_1: m \rightarrow \alpha f = (a_1 = v_1, \ldots, a_n = v_n) \rightarrow \text{class} = c,$$

where $a_i$ is an attribute and $v_i$ is one of its possible values. This rule means that if an instance satisfies at least $m$ of $n$ conditions, $a_i = v_i$, $i = 1, \ldots, n$, then the instance belongs to class $c$. For simplicity, we use $b_i$ to represent $a_i = v_i$ and denote the body as $b_1 + \cdots + b_n \geq m$.

© Springer-Verlag Berlin Heidelberg 2000
3 Evaluation of Rules

The most basic criteria to evaluate classification rules are support and accuracy. Support is a probability that both of a body and a head are satisfied, and represents generality of a rule to be evaluated. Accuracy is a conditional probability that a head is satisfied on the condition that a body is satisfied, and represents reliability of a rule. Then, it is important to discover rules with large support and high confidence. A simple way to find such rules is to give lower bounds of support and accuracy and to select only rules whose support and accuracy are higher than the lower bounds[1].

However, fixed lower bounds are insufficient to select interesting rules and sometimes accept boring rules[7]. Let’s consider a rule \( b_1 \rightarrow c \) and its specialization \( b_1 \land b_2 \rightarrow c \). If we already know the rule \( b_1 \rightarrow c \) and its accuracy is higher than that of \( b_1 \land b_2 \rightarrow c \), then the second rule doesn’t give us any new information and is meaningless even if its accuracy is higher than the given lower bound. This problem becomes serious for rules with m-of-n bodies because rules with small threshold for many conditions, i.e. small \( m \) for large \( n \), easily achieves large support and typical lower bounds of support and accuracy tend to lead huge number of rules.

The basic idea to resolve this problem is to dynamically set lower bounds of support and accuracy by using support and accuracy of shorter rules. In other words, we require longer rules must sufficiently improve support and/or accuracy from shorter rules. To evaluate a rule with \( n \) conditions in its body,

\[
R_1 : b_1 + \cdots + b_n \geq m \rightarrow \text{class} = c,
\]

we use the following rules with \( n - 1 \) conditions in their bodies.

\[
R_2 : b_1 + \cdots + b_{i-1} + b_{i+1} + \cdots + b_n \geq m \rightarrow \text{class} = c,
R_3 : b_1 + \cdots + b_{i-1} + b_{i+1} + \cdots + b_n \geq m - 1 \rightarrow \text{class} = c.
\]

Because \( R_1 \) is a generalization of \( R_2 \) and a specialization of \( R_3 \), \( R_1 \) is not so important if its support is not sufficiently higher than \( R_2 \) or its accuracy isn’t sufficiently higher than \( R_3 \). The problem is what “sufficiently higher” means. Assuming independence between \( b_i \) and \( b_1, \ldots, b_{i-1}, b_{i+1}, \ldots, b_n \), we can calculate support and accuracy of \( R_1 \) from those of \( R_2 \) and \( R_3 \) as follows.

\[
\text{Sup}_{\text{exp}}(R_1) = P(c \land b_1 + \cdots + b_n \geq m) = P(c \land b_1 + \cdots + b_{i-1} + b_{i+1} + \cdots + b_n \geq m) + P(c \land b_i \land b_1 + \cdots + b_{i-1} + b_{i+1} + \cdots + b_n = m - 1) = \text{Sup}(R_2) + P(b_i | c) (\text{Sup}(R_3) - \text{Sup}(R_2)) ,
\]

\[
\text{Acc}_{\text{exp}}(R_1) = P(c | b_1 + \cdots + b_n \geq m) = \frac{\text{Sup}_{\text{exp}}(R_1)}{\text{Acc}(R_2) + P(b_i) \left( \frac{\text{Sup}(R_3)}{\text{Acc}(R_2)} - \frac{\text{Sup}(R_2)}{\text{Acc}(R_2)} \right)} .
\]
Table 1. The number of m-of-3 rules with sufficiently high accuracy with respect to constraint (2). We set minimum support and minimum accuracy to 2% and 90% respectively.

<table>
<thead>
<tr>
<th>database</th>
<th>$d_{acc} = -\infty$</th>
<th>$d_{acc} = 0%$</th>
<th>$d_{acc} = 10%$</th>
<th>$d_{acc} = 20%$</th>
<th>$d_{acc} = 50%$</th>
</tr>
</thead>
<tbody>
<tr>
<td>krkp</td>
<td>3,086</td>
<td>2,042</td>
<td>65</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>mushroom</td>
<td>85,770</td>
<td>63,257</td>
<td>4,716</td>
<td>2,009</td>
<td>80</td>
</tr>
<tr>
<td>soybean</td>
<td>22,853</td>
<td>18,413</td>
<td>1,809</td>
<td>1,149</td>
<td>96</td>
</tr>
<tr>
<td>voting</td>
<td>4,497</td>
<td>1,873</td>
<td>178</td>
<td>1,030</td>
<td>21</td>
</tr>
</tbody>
</table>

Table 2. The number of m-of-3 rules with sufficiently large support with respect to constraint (1). We set minimum support and minimum accuracy to 2% and 90% respectively.

<table>
<thead>
<tr>
<th>database</th>
<th>$d_{sup} = -\infty$</th>
<th>$d_{sup} = 0%$</th>
<th>$d_{sup} = 1%$</th>
<th>$d_{sup} = 2%$</th>
<th>$d_{sup} = 5%$</th>
</tr>
</thead>
<tbody>
<tr>
<td>krkp</td>
<td>3,086</td>
<td>1,175</td>
<td>101</td>
<td>13</td>
<td>0</td>
</tr>
<tr>
<td>mushroom</td>
<td>85,770</td>
<td>56,024</td>
<td>4,256</td>
<td>1,306</td>
<td>140</td>
</tr>
<tr>
<td>soybean</td>
<td>22,853</td>
<td>18,907</td>
<td>731</td>
<td>84</td>
<td>0</td>
</tr>
<tr>
<td>voting</td>
<td>4,497</td>
<td>2,123</td>
<td>504</td>
<td>249</td>
<td>27</td>
</tr>
</tbody>
</table>

If the real support and/or accuracy of $R_1$ is comparable to or smaller than above values, then $R_1$ only shows regularity that can be easily expected from shorter rules. This leads the following constraints of interesting rules.

$$Sup(R_1) > Sup_{exp}(R_1) + d_{sup}, 1 \leq \forall i \leq n,$$

(1)

$$Acc(R_1) > Acc_{exp}(R_1) + d_{acc}, 1 \leq \forall i \leq n,$$

(2)

where $d_{sup}$ and $d_{acc}$ are domain dependent parameters.

4 Experiments

We applied the constraints (1) and (2) to databases in UCI repository and explored how many rules and what kind of rules were selected. Because of the lack of space, we only report the results for m-of-3 rules. For all experiments, we first enumerated m-of-3 rules whose support and accuracy were higher than 2% and 90% respectively and filtered them with the constraints (1) and (2). Table 1 reports the number of rules when the accuracy constraint (2) was applied and table 2 reports the results with the support constraint (1). Even if the databases used in the experiments are not so large and we focused only on short rules with $n = 3$, there were thousands or more rules with 2% or larger support and 90% or higher accuracy$(d_{acc} = -\infty$ in table 1 and $d_{sup} = -\infty$ in table 2). However, we could select non-trivial rules from them by applying the constraints (1) or (2) with appropriate values of $d_{acc}$ and $d_{sup}$.
5 Related Works

For discovery of rules, one of the most important problem is what kind of rules should be discovered. Many previous works dealt with this problem and proposed various criteria of rules to filter out less interesting rules. One approach is to discover typical rules in a given database that cover many instances and classify them with high confidence[4]. Entropy-based criteria such as J-measure[5] is an example of the criteria for typical rules. Another approach is to discover exceptional rules. Because the exceptional rules cover relatively small number of instances and a different type of criteria is required. Suzuki[6] proposed a criterion to evaluate a pair of typical rule and its exception based on the difference of accuracy of the two rules. My previous work[7] proposed a criterion for rules with conjunctive bodies based on expected value of accuracy and showed the criterion could filter out trivial rules.

6 Summary

This paper focused on discovery of classification rules whose bodies are m-of-n concepts and proposed criteria to select only meaningful rules from huge number of rules with large support. The criteria compare support and accuracy of a rule to be evaluated with these expectations calculated from simpler rules.

References

Issues in Organizing a Successful Knowledge Discovery Contest

Einoshin Suzuki

Electrical and Computer Engineering, Yokohama National University, 79-5 Tokiwadai, Hodogaya, Yokohama 240-8501, Japan
suzuki@dnj.ynu.ac.jp

Abstract. This paper discusses three issues in organizing a successful knowledge discovery contest based on our experience with KDD Challenge 2000. KDD Challenge 2000 has been a success with its three unique features: four preliminary contests, four data sets, and two program committees. Based on this experience, we consider that three issues: clear motivations of a contest, supports for domain experts, and promotion for participants, are mandatory for a successful organization of a knowledge discovery contest.

1 Introduction

A knowledge discovery contest (KDC) is a systematic attempt to evaluate discovery methods of participants with a set of common data sets or common problems. The interest in KDCs has been increasing in the KDD (Knowledge Discovery in Databases) community. Major KDCs include KDD-Cup, Discovery Challenge [4], and KDD Challenge [3], which were held in conjunction with KDD conferences (International Conference on Knowledge Discovery & Data Mining), a PKDD conference (European Conference on Principles and Practice of Knowledge Discovery in Databases), and a PAKDD conference (Pacific-Asia Conference on Knowledge Discovery and Data Mining) respectively. Besides, UCI KDD Archive [5] provides various benchmark problems of KDD with domain knowledge, and can be considered as a promising source of KDCs.

Since the goal of KDD can be summarized as extraction of useful knowledge from a huge amount of data, a knowledge discovery (KD) system should be evaluated from the usefulness of its output. A KDC provides a systematic opportunity for such evaluation, and is thus highly important in KDD. However, there are several issues in organizing a successful KDC. We here discuss such issues based on our experience with KDD Challenge 2000 [3].

2 KDD Challenge 2000

As a KDC, KDD Challenge 2000 provides three unique features. First, KDD Challenge 2000 was held based on four domestic KDCs each of which had been organized by JSAI (Japanese Society for Artificial Intelligence), and was thus
carefully prepared. Second, KDD Challenge 2000 provided four data sets; Diagnosis data on meningoencephalitis, Bacteriological examination data, Treatment history data of patients under collagen diseases, and Mutagenicity data; to its challengers and was thus richest in opportunities. Each data represents a single table-formatted data set, data sets with set attributes and numerous missing values, highly irregular time-series data, and data sets with chemical structures respectively. Third, KDD Challenge 2000 was organized by a program committee with data mining specialists and a supervisory board with domain experts, and was thus supported by people with various backgrounds. Several organizers are a data mining specialist as well as a domain expert, and played a mandatory role in the contest. The concern of participants can be classified as a domain-specific problem such as mutagenic activity and a data mining problem such as rule discovery from graph structures. Professor J. M. Żytkow, who contributed many results to automated discovery from the viewpoint of history of science, is interested in promoting collaborative discovery among participants in a KDC.

3 Issues in a Knowledge Discovery Contest

First, motivations of a KDC should be clearly settled and announced. Such motivations can be decomposed into academic benefits, which mainly represent development of an effective KD method, and domain benefits, which mainly represent discovery of useful knowledge. Evaluation of KD methods depends on which motivation be emphasized. For example, suppose you have two KD methods: one discovered 99 useful rules and 1 useless rule, and another discovered 1 incredibly useful rule and 99 useless rule. Motivation for academic benefits favors the former method, while motivation for domain benefits favors the latter method.

Second, measures to support domain experts are needed. Domain experts can be reluctant due to several factors including noise in data and immaturity of the domain. The target problem of a KDC should be appropriately settled considering the evaluation process. Moreover, motivation for academic benefits necessitates clarifications of successes and failures with respect to characteristics of a KD method. It is also desirable that interestingness index is decomposed into several indices such as validness, novelty, and usefulness. All of these require considerable efforts to domain experts, but improve the quality of a contest.

Third, increasing the number of participants is mandatory for the success of a KDC. Currently, a participant should perform the whole KDD process [2] by himself, and suffers from its iterative and interactive aspects. We predict that a future KDC will allow partial participations by either combining partial results systematically or promoting collaborations among participants. The former requires standardization of a KDD process, which is considered highly important in KDD. The latter is being experimented in Discovery Challenge 2000 [1] which will be held in conjunction with PKDD-2000.
4 Conclusions

The goal of KDD can be summarized as extraction of useful knowledge from a huge amount of data, thus a KD system cannot be evaluated solely by the given data. A KDC offers a systematic opportunity for subjective evaluation by domain experts, and is considered as mandatory in KDD. This article discussed three issues from the viewpoint of organizing a successful KDC based on our experience with KDD Challenge 2000. We should steadily continue accomplishing academic and domain achievements in the coming KDCs, and also extending the application domains to a broader range.

Acknowledgement

We are grateful to the program committee members\(^1\) and the supervisory board members\(^2\), for their valuable collaboration and contribution in KDD Challenge 2000. This invaluable experience was mandatory for writing this manuscript.

References

1. Discovery Challenge 2000, \url{http://www.cwi.nl/conferences/pkdd2000/}.
5. UCI KDD Archive, \url{http://kdd.ics.uci.edu/}.

\(^1\) Professor Petr Berka (University of Economics, Czech Republic), Professor Takashi Okada (Kwansei Gakuin University, Japan), Professor Terano Takao (Tsukuba University, Japan), Professor Shusaku Tsumoto (Shimane Medical University, Japan), Professor Takashi Washio (Osaka University, Japan), Professor Takahira Yamaguchi (Shizuoka University, Japan), and Professor Jan M. Żytkow (University of North Carolina, USA)

\(^2\) Dr. Makoto Hayashi (National Institute of Health Sciences, Japan), Dr. Katsuhiko Takabayashi (Chiba University, Japan), and Professor Shusaku Tsumoto (Shimane Medical University, Japan)
Knowledge Integration of Rule Mining and Schema Discovering

Kohei Maruyama and Kuniaki Uehara

1 Graduate School of Science and Technology, Kobe University, Japan
2 Research Center for Urban Safety and Security, Kobe University, Japan

1 Introduction

Despite the growing popularity of semi-structured data such as Web documents and bibliography data, most data mining researches have focused on databases containing well structured data like RDB or OODB. In this paper, we try to find useful association rules from semi-structured data. However, some aspects of semi-structured data are not appropriate for data mining tasks.

One problem is that semi-structured data contains some degree of irregularity and it does not have fixed schema known in advance. The lack of external schema information make it a very challenging task to use standard database access method or to apply the algorithms of rule mining. Therefore, schema discovering is considered to be necessary for rule mining.

Another problem of association rule mining is computing cost. If discovered schema pattern contains redundant attributes, they affect mining efficiency. Therefore, we try to feedback knowledge that obtained from the result of association rules to schema discovering. It means rule mining and schema discovering can give benefit to each other. In this way, by integrating knowledge of both rule mining and schema discovering, we can extract useful association rules from semi-structured data efficiently.

2 Schema Discovering for Mining Association Rules

2.1 Prototype-based Model for Semi-Structured Data

In order to make it easy to manipulate semi-structured data, we begin by representing semi-structured data in prototype-based model [1]. Prototype-based model is proposed in the field of object-oriented programming, but there is no distinction between classes and instances like traditional object-oriented model. Therefore, each object can have its own structure. Furthermore, slots which store the attributes and data values are evaluated dynamically. So there is no need for system to know the type of data in advance. Such features of prototype-based model is considered to fit the features of semi-structured data.

In our approach, we use BibTeX data as a kind of semi-structured data. Structures and patterns of those bibliography data depend on users or journals, so we can consider them as semi-structured data. In our prototype-based model,
each slot of prototype objects stores the each attribute of BibTeX data such as "title" and "author." Those attributes and their values contain some irregularities, but in prototype-based model, we need not know what kind of data exists in each BibTeX data in advance. As shown in Fig. 1, we represent each BibTeX data as tree model to apply schema discovering algorithm. Detected the most typical common structure in schema discovering can help us applying the algorithm of association rule mining.

\begin{verbatim}
@InProceedings{Object-ID,
  author =  "A.SMITH and M.TOM",
  title =   "Visualizing Tool for ...",
  booktitle = "International Conference On...",
  publisher = "ACM Press, New York",
  address = "Menlo Park, CA, USA",
  keyword = "data mining, visualizing",
  month = "aug",
  year = "1996",
  note = "ftp://ftp...../**.ps",
  address = "Menlo Park, CA, USA",
  publisher = "ACM Press, New York",
  editor = "AAA & BBB...",
  author = "A.SMITH and M.TOM",
  title = "Visualizing Tool for ...",
  booktitle = "International Conference On...",
  publisher = "ACM Press, New York",
  address = "Menlo Park, CA, USA",
  keyword = "data mining, visualizing",
  month = "aug",
  year = "1996",
  note = "ftp://ftp...../**.ps",
  address = "Menlo Park, CA, USA",
  publisher = "ACM Press, New York",
  editor = "AAA & BBB...",
}
\end{verbatim}

Fig. 1. Example of BibTeX Data.

\section{Discovering Typical Schema Pattern}

In order to clean semi-structured data like BibTeX data, we detect the most typical common structure of them by using the same idea of Wang & Liu's algorithm [2], which is called schema discovery. Once a most typical schema pattern is discovered, it filters out the redundant attributes that do not match with the schema pattern. By storing all data into that pattern, we can translate them into cleaned structured data. A summary of this process is as follows:

- **Definitions**
  - Let $p_i$ denotes \textit{path expression} which is a path representation from root node to leaf node. A \textit{k-tree expression} is a tree-expression containing $k$ leaf nodes and can be represented by a sequence $p_1...p_k$.
  - Consider a tree-expression $te$. The \textit{support of $te$} is the number of the root document $d$ such that $te$ is "weaker than" $d$. Intuitively, if all structural information of $te_1$ is found in $te_2$, $te_1$ is weaker than $te_2$. \textit{MINSUP} denotes user-specified minimum support and $te$ is frequent if the \textit{support of $te$} is not less than \textit{MINSUP}. $te$ is \textit{maximally frequent} if $te$ is frequent and is not weaker than other frequent tree-expressions. The discovery problem is to find all maximally frequent tree expressions.

- **Algorithm**
  1. \textit{MINSUP} is specified by the user.
  2. For all frequent 1-tree expressions, $F_1$ are found in the form of pass-expressions.
  3. Every frequent $k$-tree expressions $p_1...p_{k-1}p_k$ is constructed by two frequent \textit{(k-1)}-tree expressions $p_1...p_{k-2}p_{k-1}$ and $p_{k-2}p_k$. We represent all frequent \textit{k-tree expressions} $p_1...p_{k-1}p_k$ as $F_k$. 

4. The actual frequent k-tree expressions in $F_k$ are found. This step prunes all non-maximally tree-expressions.

![Diagram](image)

**Fig. 2.** Constructing (k+1)-frequent Tree Expression

Fig. 2 shows the case that frequent 3-tree expression $p_1 p_2 p_3$ is constructed from two frequent 2-tree expressions $p_1 p_2$ and $p_1 p_3$. We can treat the final k-frequent tree expression as the most typical schema pattern. After we find it and clean data, we can apply the algorithm of mining association rules.

### 2.3 Mining Association Rules with Concept Hierarchy

Association rules are powerful abstractions to understand a large amount of data by finding interesting regularities which satisfy two given thresholds of support and confidence. However, one of the problem for unavailability of useful association rules is that rules are generated with no background knowledge. In order to find more interesting and informative rules, it can be considered to use concept hierarchies as the representation of background knowledge [3].

In the case of knowledge discovery from BibTeX data, the most useful attribute is “title” attribute. It is considered to contain important words which denote author’s main interest. So we associate each word in “title” with the concept in concept hierarchy, and generate not only original association rules but also their child rules which contain their child nodes in concept hierarchy.

Fig. 3 shows the example of generated child rules. These child rules can tell who is the main researcher in each field defined as a child of “data mining.” Such a tendency cannot be discovered by original rules alone.

<table>
<thead>
<tr>
<th>Original Rule</th>
<th>Child Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Title</strong></td>
<td><strong>author</strong></td>
</tr>
<tr>
<td>Data Mining</td>
<td>Han, Agrawal, Ron, Kaim</td>
</tr>
</tbody>
</table>

![Diagram](image)

**Fig. 3.** Discovered Rule with Concept Hierarchy
For another example, assume that following rules are found by using the dataset we got from bibliography search engine.

- \{ \text{Journal=}Lecture note in C.S., Year=1998 \} \Rightarrow \{ \text{Title=}data mining \}
- \{ \text{Journal=}Lecture note in C.S., Year=1999 \} \Rightarrow \{ \text{Title=}data mining \}

From these rules, we can know that papers on \textit{data mining} are constantly written in recent years. But we cannot discover any more knowledge from these rules. But by using concept hierarchy, we can discover \textit{child rules} as shown in Fig. 4.

![Fig. 4. Discovered Rules and Their Graph](image)

Generated rules show that in “1998,” some papers about \textit{association rule} on the data mining field are written, but in “1999,” papers on \textit{visualizing} appears several times while there is no paper on \textit{association rule.” It means that target of interest has been changed from \textit{association rule} to \textit{visualizing} under the same \textit{data mining} field.

As shown in Fig. 4, by using concept hierarchy, we can generate more informative rules for discovered original rules. In this way, we may find the important keywords which are frequently appeared recently but not discovered because of the largeness of the itemsets we choose or of the height of \textit{support} or \textit{confidence}. One of our purpose is to find these hidden informative rules behind the discovered rules which is meaningless at first sight and discover knowledge like such a tendency of particular researcher, research field, and so on.

### 3 Feedback from Rule Mining to Schema Discovering

As mentioned above and our previous paper [4], schema discovering helps the task of rule mining from semi-structured data, and we could discover some useful rules. One problem here is that the algorithm should compute all combinations of items and it is crucial for mining efficiency. Therefore, if we can get rid of useless attributes in advance which do not appear as a result of mining, we will be able to carry out the rule mining efficiently in case that a need for next mining arises such as update of database. Therefore, we consider to feedback the result knowledge of rule mining in turn to improve schema pattern. Fig. 5 shows the outline of how this knowledge integration works.

In Fig. 5, left tree is a typical schema pattern discovered by schema discovering. After refining BibTeX data by storing all of them in it, we extract association...
rules. If discovered rules do not include data value of some attributes, it can be pruned from the schema pattern to make it a less redundant form. For example, "address.city" attribute is discovered in typical schema pattern, but it doesn’t appear in the extracted association rule. Therefore, it can be pruned. In this way, we can generate improved schema pattern and use it for next mining. By using less redundant schema pattern, we will generate new rules efficiently. That is, rule mining and schema discovering can give mutual merit to each other in case of handling semi-structured data.

Some researchers try to feedback the benefit of data mining to information extraction system in similar approach [5]. They try to improve bottom-up information extraction which is considered to be appropriate for incremental retrieving of data. On the other hand, we try to improve top-down schema discovering which is appropriate for efficient mining for large database. Our consideration here is that computational cost must be a main concern in data mining, but we also have to examine the incremental features of semi-structured data.

4 Conclusion and Future Works

In order to manipulate semi-structured data such as bibliography data, we adopt prototype-based model and clean data by using schema discovering. Also, we generate child rules to obtain more informative rules. Furthermore, we propose that the discovered rules can be reused to improve schema pattern. That is the knowledge integration of our case.

As future works, we have to examine efficiency of our approach and use of other semi-structured data like XML data. Furthermore, we plan to see our approach from the point of information extraction field.

References
Discovery of Correlation from Multi-stream of Human Motion

Mitsuomi Shimada\textsuperscript{1} and Kuniaki Uehara\textsuperscript{2}

\textsuperscript{1} Graduate School of Science and Technology, Kobe University
\textsuperscript{2} Research Center for Urban Safety and Security, Kobe University
\{shimada, uehara\}@ai.cs.kobe-u.ac.jp

1 Introduction

Human motion data is practically used in some domains such as movies, CGs and so on. Creators use motion data to produce exciting and dangerous scenes without real actors. Human motion data has following features:

1. correlation between body parts
   We control all body parts in cooperation and motion data consists of the information of cooperation. For example, we swing both arms in turn to keep walking straight. There exists correlation between arms and legs.

2. correlation in the flow of contents
   There is tendency that certain movement likely to occurs after another movement. For example, once we raise our hands, we probably put our hands down in certain interval of time. This is correlation in time flow of motion.

Because human motion data has these kinds of features, motion data should be treated as multi-stream [4]. Multi-stream includes unexpectedly frequent or infrequent co-occurrences among different streams. This means that an event on one stream is related to another events which locate on other streams and seem to have nothing to do with the former event. Time series pattern of stock price is a good example. Rise and fall of price on some stocks obviously cause price of one stock to rise and fall. If we analyze the multi-stream of time series for some stock price and can discover correlation between all streams, the correlation help us to decide better time to buy stocks.

Correlations which are discovered from multi-stream of human motion characterize a specific motion data. Furthermore, those correlations become basic elements that can be used to construct motion with combination of themselves, just as phonemes of human voice do. We call those basic elements primitive motions. As the result, we can use primitive motions as indices to retrieve and recognize motion.

We introduce the method that finds correlations based on contents of motion and converts motion data into combinations of primitive motions. Sets of primitive motions are used as indices to retrieve and recognize motion. Discovered correlations are visually comprehensible indices for multi-stream of motion, because those are found according to contents of the multi-stream. Therefore, those correlations are effectively used as indices for multi-stream of human motion.

© Springer-Verlag Berlin Heidelberg 2000
2 Human motion data as multi-stream

Motion data, captured by motion capture system, is one of the multi-stream data. The data is mass of information for body parts. This means that motion captured data consists of streams with 3-D time series patterns, representing positions of major body joints. The data is used mainly for entertainment and research domains. In movies and CGs, captured data is effectively used to create scenes that real actors could never play.

![Fig. 1. The motion capture system](image)

The multi-stream of motion includes correlations discussed in Sec. 1. Fig. 2 shows an example of motion data which represents that after one raised one's right hand, one starts putting one's left hand down.

![Fig. 2. An example of correlation in multi-stream of human motion.](image)

The example shows that motion data has features as multi-stream, such as:

- temporal distance on each stream
  Consistent contents on each stream do not always occur in fixed temporal distance. In Fig. 2, “Raising right hand” and “Putting left hand down” occur twice in each stream, however, temporal distance of occurrences between “Raising right hand” and “Putting left hand down” on left side differs from that on right side. There is always variety of those distances.
- period among contents
  Segments with content that characterize motion do not always occur in fixed
period. In Fig. 2, “Raising right hand” and “Putting left hand down” occur twice in streams with period of certain time length and that is a characteristic on motion data, however, the period is not always the same. There also exists the variety.

A personal characteristic of the actor in performance can cause the variety, and it is impossible to prevent the variety from occurring. In order to find correlations with easy analysis and with consideration of the variety, we convert motion data into strings of characters that symbolize contents and process those strings as patterns of characters. We can find patterns decreasing the effect of the variety, by adjusting the size of patterns. The detail is written in Sec. 3.

3 Primitive motion: the index for human motion

Our system converts multi-stream of motion into symbol multi-stream, because of running time for analysis algorithm on huge amount of data, such as motion data. Those symbols should:

- be given to segments in motion multi-stream with consistent content
- not be given by hand because the expression of human motion made by each person depends on his/her personal sense so much.

First of all, our system executes the process, and we call this conversion content-based automatic symbolization [2]. Next, correlations are discovered from symbol multi-stream for body parts and combination of those are used as indices for motion. We call correlations primitive motions.

3.1 Content-based automatic symbolization

For the content-based automatic symbolization, we focused on that the change of content and the change of velocity on each body part happens simultaneously. The motion data is divided into segments [3] at those points where velocity changes. Those detected points are candidates for border points of a segment with consistent content. However, those detected points include noise made by unconscious movement which has nothing to do with the changes of contents. The unconscious movement is mainly caused by vibration of body parts. The noise is removed by considering the 3-D distances between points.

Segmented data is clustered into groups according to their similarity. However, even segmented data with same content have different time length, because nobody can acts in exactly the same manner. In order to calculates the similarity between time series patterns with different length, we employ Dynamic Time Warping (DTW) [1], which was developed in speech recognition domain. DTW calculates the similarity between patterns absorbing the difference on the time scale.

Human voice has fixed number of consistent contents, phonemes, but human motion does not have pre-defined patterns. So it is unknown that how many
consistent contents exist. For the reason, we use a simple and powerful unsupervised clustering algorithm, Nearest Neighbor (NN) algorithm [3]. The system clusters segments by using NN algorithm with DTW for its evaluation function, and motion data is automatically converted into symbol streams based on its content by using symbols that are given to clusters.

3.2 Correlations in symbol multi-stream

Correlations of contents exists on flow of time, so the symbol multi-stream is synchronized on time scale as shown in Fig. 3 (a).

![Diagram](image)

Fig. 3. The process of multi-stream.

Discovered primitive motions show frequency of occurrence of two symbol patterns. That means the system calculates probability of occurrence for a pattern B which occurs after a pattern A in certain blocks of interval. See the right hand side of Fig. 3 (a). Our system finds the pattern A which occurs more times than others and includes symbol “B”, “B”, “L”, “M” in two streams with right hand and elbow. After two blocks of symbols from the beginning of A, the system finds the same kind of pattern B which consists of “U”, “U”, “G”, “G” in streams of left hand and elbow. The combination of occurrence of A and B occurs frequently in symbol multi-stream, and it means “Contents “U” and “G” occur after contents “B”, “L” and “M” occur in the motion at higher probability”, because symbols represent contents of motion(Fig. 3 (b)).

In order to decrease the influence of the variety which we discussed in Sec. 2, we set the size of patterns A (A_size) and B (B_size), and, the interval of A and B (int). Those sizes have a flexibility and the flexibility allows the system to find patterns A and B with the same interval, even if those sizes are set A_size = 2, B_size = 2 and int = 2, or, A_size = 3, B_size = 3 and int = 1. This flexibility decreases the influence of the variety.

The system discovers many correlations from motion multi-stream and each of them has a degree of strongness to characterize the motion. Probability is an influential parameter to define the strongness and to select correlations as primitive motions for the motion. However, it still needs a deep consideration to define the strongness.

4 Experiments

We prepared 8 kinds of Japanese traditional physical exercise data as test data to discover primitive motions. Every motion consists of repetition of 2 or 3 times for
one kind. Test data are about 10 to 20 seconds long, and the sampling frequency is 120 times/seconds. Main movement of those test data concentrates on both arms and we had experiments for 4 streams of following body parts: right and left hands, and, right and left elbows.

The system discovered primitive motions mainly in streams of right and left elbow. This is because elbows have less number of clusters than that of hands. To be precise, elbows have about 100 clusters and hands have about 200 clusters. The difference of numbers mean hands can express motion with wider variety than elbows do. The system discovered primitive motions such as: a) left elbow is on the way to go, after right elbow starts going down, b) after right elbow stops going up, left elbow starts going down and so on. These primitives occurred over 40 or 50 times, and the probability of occurrence is from 0.3 to 0.4.

Some other primitive motions kept the probability as high as 0.3, but those primitives occurred under 40 or 50 times. This shows that primitive motions with higher probability occurrence and with more frequent occurrence than others should be given to motion as indices. It is difficult to express features of motion exactly only by words, however, those discovered correlations were visually understandable. This means our method is effective to discover indices for multi-stream of motion.

5 Conclusion

We proposed the method that finds correlations between body parts and between contents in time flow of motion. Those correlations appears many times in multi-stream and motion data is represented with the combination of correlations. Correlations is basic elements of motion, and we call correlations primitive motions. Primitive motions can be indices for motion retrieval and recognition.

In motion recognition domain, many researchers have focused on the extraction of human motion information from time series of 2-D images. The extracted information does not possesses correlation and they do not process the information as multi-stream. From this point of view, our method is more effective to recognize and retrieve human motion data.

References

An Appropriate Abstraction for Constructing a Compact Decision Tree

Yoshimitsu Kudoh and Makoto Haraguchi

Division of Electronics and Information Engineering,
Hokkaido University, N 13 W 8, Sapporo 060-8628 JAPAN
{kudo,makoto}@db-ei.eng.hokudai.ac.jp

Abstract. In general, it is considered that pre-processing for data mining are necessary techniques to remove irrelevant and meaningless aspects of data before applying data mining algorithms. From this viewpoint, we have considered pre-processing for detecting a decision tree, and already proposed a notion of Information Theoretical Abstraction, and implemented a system ITA. Given a relational database and a family of possible abstractions for its attribute values, called an abstraction hierarchy, our system ITA selects the best abstraction among the possible ones so that class distributions needed to perform our classification task are preserved, and generalizes database according to the best abstraction. According to our previous experiment, just one application of abstraction for the whole database has shown its effectiveness in reducing the size of the detected decision tree, without making the classification accuracy worse. However, since such classification systems as C4.5 perform serial attribute-selection repeatedly, ITA does not generally guarantee the preservingness of class distributions, given a sequence of attribute-selections. For this reason, in this paper, we propose a new version of ITA, called iterative ITA, so that it tries to keep the class distributions in each attribute selection step as possibly as we can.

1 Introduction

Many studies on KDD, knowledge discovery in databases have concentrated on developing data mining algorithms for detecting useful rules from very large databases effectively. However, those detected rules include even meaningless rules as well as meaningful ones. Thus, pre-processing for data mining are needed to exclude irrelevant and meaningless rules. There exists some techniques commonly used in the pre-processing[1]. For instance, an attribute-oriented induction used in DBMiner[2] is a powerful technique not only for preventing the mining task from extracting meaningless rules but also making the detected rules more understandable.

We have already developed a system ITA (Information Theoretical Abstraction)[3] based on the attribute-oriented induction and the information theory. ITA is based on the idea that the information gain ratio used in C4.5[4] can be also applicable to determine which abstraction preserves the necessary information in the attribute-oriented induction[2]. Given a relational database and a
set of possible abstractions for its attribute values, ITA selects an appropriate abstraction among possible ones and generalizes the database according to the selected abstraction in order to reduce its size, where an abstraction is said to be appropriate if for given target classes, the class distribution in the original database can be preserved in the resultant of generalization as possibly as we can. Therefore, we can obtain a compact (generalized) database still having an abstract class distribution closer to the original one. In our previous work, the original database $D$ is first generalized according to an appropriate abstraction, and then the generalized one $D'$ inputs to C4.5 to construct a decision tree $DT'$ [3]. It has already empirically shown that the classification error of $DT'$ is almost the same as one of a decision tree directly computed by C4.5 from the original database $D$. Nevertheless, the size of $DT'$ is drastically reduced, compared with $DT$.

Thus, just one application of abstraction for whole database has been experimentally shown its effectiveness in reducing the size of detected rules, without making the classification error worse. However, as C4.5 performs serial attribute selections repeatedly, ITA does not generally guarantee the preservingness of class distribution in each selection step. Hence, when we require that the classification accuracy of $DT'$ must be almost equal to $DT$, we cannot allow the classification accuracy to go down even slightly. For this reason, in this paper, we propose a new version of ITA, called iterative ITA so that it tries to keep the class distribution for each attribute selection step as possibly as we can.

As the result, the precision of detected rules will become much closer to one of C4.5, while keeping the same property of reducing the size of detected rules as to non-iterative ITA.

2 Iterative ITA

Figure 2.1 illustrates a generalization process in iterative ITA. Iterative ITA selects an appropriate abstraction in each attribute selection step and constructs a compact decision tree, called an abstract decision tree. That is, we propose to perform our generalization process in each attribute-selection step in C4.5, where an attribute based on which a decision tree is expanded. Each node $N_j$ in a tree has a corresponding sub-database $D_{N_j}$ of the original $D$, obtained by selecting tuples. For such a sub-database $D_{N_j}$, C4.5 selects another attribute $A_i$ to furthermore expand the tree. We try to find an appropriate abstraction $\varphi$ for that $A_i$, so that the target class distribution given $A_i$ values can be preserved even after generalizing the $A_i$ values $v$ to more abstract value $\varphi(v) = \tilde{a}_k$. The generalized database is also denoted by $D_{N_i} = \varphi(D_{N_i})$. Iterative ITA has the following features.

- The process of constructing the decision tree in C4.5 divides the current node corresponds to $D$ according to all attribute values $\{v^1_1, \ldots, v^1_n, \ldots, v^m_1, \ldots, v^m_n\}$ in $A_i$. On the other hand, iterative ITA divides the current node according to all abstract concepts in the grouping $\{\{a_1, \ldots, a_n\}, \ldots, \{a^m_1, \ldots, a^m_n\}\} = \{\tilde{a}_1, \ldots, \tilde{a}_m\} = \varphi(A_i)$ selected in each attribute selection step.
An Appropriate Abstraction for Construction a Compact Decision Tree

Therefore the number of the branches is \( m \) and iterative ITA reduces the branches in the original decision tree constructed by C4.5.

- Roughly speaking, we can say that a condition of terminating the expansion of the decision tree is concerned with a question of whether an expected classification error rate in the child nodes is small or not compared with the current node. A process of constructing the decision tree in our iterative ITA is similar to C4.5. Suppose that the expected classification error rate in some child node is smaller than the current node, and this relation between some child node and the current node holds after abstraction. Then the classification accuracy in \( D_{N_k} \) is improved. In such a case, iterative ITA continues a process of constructing the decision tree. Otherwise it terminates the expansion process. Since the expected classification error rate at abstract level is the average of those at concrete level, this error rate at abstract level turns out to be larger than the error rate at concrete level. In other words, a chance of improving the classification accuracy is lost by applying abstraction.

Furthermore, suppose that the expected classification error rate in some child node is larger than the current node. This means that the condition for terminating the expansion process holds. Then, for any abstraction, the expected classification error rate at abstract level is also larger than the error rate at concrete level. In this case, iterative ITA terminates the expansion process. This is again because the expected classification error rate at abstract level is defined as the average of the error rate at concrete level. As a result, we can say that stopping condition for expansion at abstract level is satisfied whenever it does at concrete level, and the condition at abstract level tends to hold earlier than the concrete level.
3 Experiment on Census Database

We have made some experimentations using iterative ITA system. In our experimentation, we try to generate decision trees from a Census Database in US Census Bureau found in UCI repository. Iterative ITA generates various compact decision trees, called abstract decision trees, by adjusting a threshold of a change ratio. The change ratio is the ratio of the information gain ratio after applying generalization one before applying generalization. We compare our abstract decision trees with a decision tree generated by C4.5. A size and an error rate of their decision trees are shown in Figure 3.2.

From this observation, we consider that iterative ITA is useful for constructing a compact decision tree whose error rate is approximately equal to one before generalization because the size has drastically decreased to about 1200 from about 6000, at the sacrifice of slightly increasing error rate to 0.167 from 0.132 in the best case.

4 Conclusion

We can consider that the generalization method used in iterative ITA is useful for generating a very compact abstract decision tree, that is more understandable for almost users, whose regression of error rate is minimized among a given class of abstractions.

References

2. Han, J. and Fu, Y.: Attribute-Oriented Induction in Data Mining. In [1], pp.389-421, 1996.
Extracting Positive and Negative Keywords for Web Communities

Daisuke Ikeda and Sachio Hirokawa

Computing and Communications Center, Kyushu University,
Fukuoka 812-8581, Japan
{daisuke.hirokawa}@cc.kyushu-u.ac.jp

1 Introduction

The linkage information is shown to be useful to find good Web pages at a search engine [5, 10]. But, in general, a search result contains several topics. Clustering Web pages enables a user to browse them easily. There are several works on clustering Web pages [7, 10–12]. In [9], we visualized Web graphs using spring model. But clustering is not enough to understand the topics of the clusters. Extraction of meta-data that explains communities is an important subject. Chakrabarti et al [6] used the terms in the small neighborhood around a document. Our approach is to combine the clustering and keyword extraction to interpret the communities.

To find communities, we solve the eigensystem of the matrix made from the link structure of Web pages. To get characteristic keywords from found communities, we use the algorithm developed in [1, 2, 8]. The input for the algorithm are two sets of documents — positive and negative documents. The algorithm outputs a pattern which well classifies them. This algorithm is robust for errors and noises, so that it is suitable for Web pages. The novelty of the keyword extraction algorithm is that keywords not only characterize one community but also distinguish the community from others. Thus, even if we fix a community, we have different characteristic keywords for the community according to the counter part.

We found good characteristic keywords from two communities without seeing Web pages in them. We also show an experimental result in which different keywords are extracted according to the counter part.

2 Preliminaries

Our method has three basic steps. First, we collect a large number of Web pages. The second step is to find communities by solving eigensystem of the matrix made from the link structure of the collected pages. Kleinberg [10] showed some communities extracted from eigenvectors. The final step is to find characteristic keywords from communities found in the previous step.

First, we define a Web graph and its matrix representation. A Web graph $G = (V, E)$ is a directed graph such that (1) each node $v \in V$ is labeled with a URL $u$.

© Springer-Verlag Berlin Heidelberg 2000
(label(v) = u), and (2) there exists an edge (v₁, v₂) ∈ E if and only if there exists a link from label(v₁) to label(v₂). The adjacency matrix M_G = (m_{i,j}) for a Web graph G is defined by m_{i,j} = 1 if (v_i, v_j) ∈ E and m_{i,j} = 0 otherwise. For a matrix M, M^T denotes its transposed matrix.

### 2.1 Community Discovery

In [10], Kleinberg introduced the notion of authority and hub. Let G = (V, E) be a Web graph, and x_i and y_i be an authority and hub weights, respectively, for a node v_i ∈ V. His simple iteration algorithm updates both weights by $x_i \leftarrow \sum_{(v_i,v) \in E} y_j$ and $y_i \leftarrow \sum_{(v,v_i) \in E} x_j$, and normalizes at each iteration step. He shows that this iteration converges.

**Theorem 1 (Kleinberg [10]).** $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_n)$ converge the principle eigenvectors of $A(G)^T A(G)$ and $A(G) A(G)^T$, respectively, where $A(G)$ is the adjacency matrix of G and the principle eigenvector is one which has the largest eigenvalue.

Let $(x^+_1, \ldots, x^+_n)$ be the principle eigenvector. Then, the page with large value $x^+_i$ is a better page. Kleinberg [10] showed that $(X^+_i, Y^+_i)$ and $(X^-_i, Y^-_i)$ form two communities, where $(X^+_i, Y^+_i)$ are the most positive coordinates and $(X^-_i, Y^-_i)$ are the most negative ones in some non-principal eigenvectors of $A(G)^T A(G)$ and $A(G) A(G)^T$.

### 2.2 Keyword Discovery

Arimura et al. [1, 2] formulated the optimal pattern discovery problem and develop an efficient algorithm for it. The problem is, given two sets of documents, to find a pattern that classifies them.

We briefly give a formulation of the problem. Let $S = \{s_1, \ldots, s_n\}$ be a set of documents. An objective condition over S is a binary labeling function $\xi : S \to \{0, 1\}$. For a document s, we call it a positive document if $\xi(s) = 1$ and a negative document if $\xi(s) = 0$. Pos and Neg denotes the sets of positive and negative documents, respectively. For a pattern $\pi$ and a document s, we define $\pi(s) = 1$ if $\pi$ matches $s$ and $\pi(s) = 0$ otherwise. Let $S_i = \{s \in S|\pi(s) = i\}(i = 0, 1)$. We define $G(\pi) = \psi(\theta_i)|S_i| + \psi(\theta_i)|S_i|$, where $\theta_i = |S_i \cap \text{Pos}|(i = 0, 1)$ and $\psi$ is an impurity function. In our experiments, we use $\psi(\theta) = -\theta \log \theta - (1-\theta) \log (1-\theta)$.

**Definition 1.** The optimal pattern discovery problem is, given a set S of documents and an objective condition $\xi$ over S, to find a pattern $\pi$ that minimizes $G(\pi)$.

Although we can use a complex pattern in the above problem, in this paper, we simply use a substring of a Web page as a pattern because it requires small computational complexity.
3 Experiments

To collect Web pages, we give a query to a search engine and get a set, called root set, of the top 200 Web pages. We use AltaVista. Then, we enlarge the root set to base set according to [10]. For this purpose, we use the URL database in KN (Knowledge Network) [9]. We use the following restrictions when enlarge to base set: (1) only a URL whose prefix is “http” is used. (2) a link between pages on the same host is ignored. (3) for a URL in the root set, if there exist more than 50 pages that point to the URL, then we choose only 50 pages randomly. In our experiments, we give a query of “+java +introduction” to AltaVista. Then, we enlarge the root set into the base set of 9,159 Web pages.

3.1 Principal Eigenvector

We implement the simple iteration algorithm in [10], which outputs the ranked lists of authorities and hubs. The lists correspond to the principal eigenvectors.

The following table is a list of extracted keywords with the best 98(1/100)' authorities as positive documents. As negative documents, we use the worst 90(1/100) authorities and the best 34(1/50) hubs. In general, the set of the authorities does not form a single community, but we expect that this experiment shows the power of the keyword extraction algorithm.

<table>
<thead>
<tr>
<th>negative documents</th>
<th>positive keywords</th>
<th>negative keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>worst 90 authorities</td>
<td>java, introduction</td>
<td>national, institute, association</td>
</tr>
<tr>
<td>best 34 hubs</td>
<td></td>
<td>developer’s, resource, guide, library</td>
</tr>
</tbody>
</table>

In the first case, the algorithm finds positive and negative keywords that classifies two sets of documents. Especially, the query terms “java” and “introduction” are discovered from the positive documents. But, in the second case, the algorithm does not find good positive keywords although it finds good negative keywords. This is due to the number of negative documents. But, found keywords from the negative documents seem to show a property of hub pages.

3.2 Non-Principal Eigenvector

We also implement an algorithm in Fortran that solves the eigensystem. The table 1 shows extracted keywords from the two communities of the 2nd eigenvectors. The positive documents are 200 Web pages which are 100 pages of the most positive coordinates in the 2nd eigenvectors of both $A(G)'A(G)$ and $A(G)A(G)'$, and the negative ones are also 200 pages which are 100 pages of the most negative coordinates of them. We see many keywords related to the operation system name “OS/2 Warp” in the negative keywords.

The two communities of the 2nd eigenvectors form the Web graph in Figure 1. The nodes at the right-hand side are the positive documents and those at the left-hand side are negative.

---

2. We can get only 98 pages from the list of 100 URLs.
### Table 1. Keywords extracted from the 2nd eigenvectors.

<table>
<thead>
<tr>
<th>Rank</th>
<th>Score</th>
<th>Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>153</td>
<td>“os/2”</td>
</tr>
<tr>
<td>2</td>
<td>37</td>
<td>“warp”</td>
</tr>
<tr>
<td>3</td>
<td>26</td>
<td>“for os/2”</td>
</tr>
<tr>
<td>4</td>
<td>27</td>
<td>“os/2 warp”</td>
</tr>
<tr>
<td>5</td>
<td>21</td>
<td>“os/2&lt;/a&gt;”</td>
</tr>
<tr>
<td>6</td>
<td>24</td>
<td>“driver”</td>
</tr>
<tr>
<td>7</td>
<td>19</td>
<td>“#00000F”</td>
</tr>
<tr>
<td>8</td>
<td>18</td>
<td>“os/2 user”</td>
</tr>
<tr>
<td>9</td>
<td>18</td>
<td>“of os/2”</td>
</tr>
<tr>
<td>10</td>
<td>18</td>
<td>“device driver”</td>
</tr>
<tr>
<td>11</td>
<td>17</td>
<td>“freeware”</td>
</tr>
<tr>
<td>12</td>
<td>16</td>
<td>“unofficial”</td>
</tr>
<tr>
<td>13</td>
<td>16</td>
<td>“national accelerator”</td>
</tr>
<tr>
<td>14</td>
<td>16</td>
<td>“developer magazine”</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td>“warped”</td>
</tr>
<tr>
<td>16</td>
<td>15</td>
<td>“warp 4”</td>
</tr>
<tr>
<td>17</td>
<td>15</td>
<td>“electronic developer”</td>
</tr>
<tr>
<td>18</td>
<td>15</td>
<td>“&lt;/a&gt;”</td>
</tr>
<tr>
<td>19</td>
<td>14</td>
<td>“warp&lt;/a&gt;”</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>“user group”</td>
</tr>
</tbody>
</table>

![Fig. 1. Community extracted from the 2nd eigenvectors.](image-url)
4 Conclusion and Discussion

We showed some experimental results. We found good characteristic keywords from two communities without seeing Web pages in them. We also showed an experimental result in which different keywords are extracted according to the counter part.

To obtain a complete list of communities, we need to solve eigensystem for 9,169 \times 9,169 matrix, which requires too much time and space. Applying linear algebra [3, 4] to reduce the rank of adjacency matrix for clustering is further research.

References

Nonequilibrium Thermodynamics from Time Series Data Analysis

H. H. Hasegawa*, T. Washio+, Y. Ishimiya*, T. Saito*

Graduate School of Science and Technology, Ibaraki University*
Mito, 310-8512
hiroshih@mito.ipc.ibaraki.ac.jp
I.S.I.R., Osaka University+
8-1 Mihogaoka, Ibaragi, 567-0047, Japan
washio@sanken.osaka-u.ac.jp

We propose nonequilibrium thermodynamics of non-physical systems such as biological and economical ones. We inductively construct it based on the Auto Regressive type models derived from time series data. Since the detailed balance is generally broken in such non-physical systems, we extend the Sekimoto-Sasa theory for systems with the circulation of fluctuations.

We apply our arguments to the time series data of nuclear reactor noise.

We define effective potential, $U(x, a, c)$, and free potential, $F(a, c)$, from the probability density of nonequilibrium steady state, $P_s(x, a, c) = \exp[F(a, c) - U(x, a, c)] = \exp[F(a, c) - x \cdot G_s(a, c)x/2]$ where $a, c$ are coefficient vectors of ARMA model. While the reactivity, $\rho$, so the coefficient vectors, are varying slowly, the first law is given as follows:

$$\langle U(t_i) \rangle_{t_i} - \langle U(t_f) \rangle_{t_f} = W + D,$$

$$W = \sum_{t=t_i}^{t_f-1} \int dx \frac{[U(t+1) - U(t)]}[P(x, t+1) + P(x, t)]/2.$$

---


\[
D \equiv \sum_{t_i}^{t_f-1} \int dx [U(t+1) + U(t)][P(x,t+1) - P(x,t)]/2,
\]

where \( \langle \cdot \rangle \equiv \int dx \cdot P(x,t) \) is an average over an ensemble of the time series data. We can approximate the average as a time average over a single time series when the coefficient vectors \( a(t), c(t) \) change quite slowly in time.

The work in the quasi-static process, \( W_{QS} \), is given as difference between the initial free potential and the final one. The work in the near quasi-static process, \( W_{ARMA} \), can be estimated using the ARMA model. We showed that \( W_{ARMA} \) is bounded from the below by \( W_{QS} \). The ARMA model is not valid in the nonstationary process, since many degrees of freedom are activated. We can expect the work in nonstationary process, \( W_{NS} \), is greater than \( W_{ARMA} \).

We calculated the work from actual time series data of nuclear reactor noise. We estimated the effective potential using the time average in the stationary process, \( \langle x_1^2 \rangle_{ns} = \langle x_2^2 \rangle_{ns} \) and \( \langle x_1 x_2 \rangle_{ns} \). We assumed that the matrix \( G(t) \) in the potential linearly depends on time and the ensemble average can be approximated as the time average over the single time series. We estimated the work as \( W_{NS} = [G_{11}(t_f) - G_{11}(t_i)] < x_1^2 >_{ns} - [G_{12}(t_f) - G_{12}(t_i)] < x_1 x_2 >_{ns} \). \( < x_1^2 >_{ns} \) and \( < x_1 x_2 >_{ns} \) are the time average over the time series data in the nonstationary process.

We obtained the following results:

| \( \rho \) | \(-2.19 \rightarrow -1.25\) | \(-0.588 \rightarrow -0.289\) |
| \( W_{QS} \) | \(-0.8644\) | \(-0.7764\) |
| \( W_{ARMA} \) | \(-0.8639\) | \(-0.7733\) |
| \( W_{NS} \) | \(-0.8377\) | \(-0.7001\) |
| \( t_f - t_i \) | \(41556\) | \(14667\) |

These results are consistent with the second law given as the following inequality,

\[
W_{NS} \geq W_{ARMA} \geq W_{QS} = F(t_f) - F(t_i).
\]

In this paper we proposed the nonequilibrium thermodynamics of non-physical systems such as biological and economical ones. Random fluctuations in such system generally do not have to do with the thermal ones. We showed that it was possible to introduce a concept of effective potential and construct nonequilibrium thermodynamics. We inductively constructed it based on the Auto Regressive type models derived from time series data.

We applied our arguments to the actual time series data of nuclear reactor noise. Since the detailed balance is broken in the system of nuclear reactor, we used the extended Sekimoto-Sasa theory for the circulation of fluctuations. From the time series data in the stationary process, we define the effective potential and the free potential. The effective potential includes the effect of the circulation of fluctuations as the house keeping heat. In the thermodynamics of nonequilibrium steady states, the work in nonstationary process is bounded from the below by the difference of the initial free potential and final one.

We thank Prof. S. Yamada for providing us the time series data of nuclear reactor noise. This work is partly supported by a Grant-in-Aid for Scientific Research on Priority Areas “Discovery Science” from the Ministry of Education, Science and Culture, Japan.
Automatic Determination Algorithm for the Optimum Number of States in NL-HMnet

Motoyuki Suzuki and Shozo Makino

Computer Center / Graduate School of Information Sciences, Tohoku Univ., 980-8578, Japan.

1 Introduction

Hidden Markov Network (HMnet)[1] is a kind of statistical finite-state automata. Discrete-type HMnet and NL-HMnet[2] can be used as a language model in a speech recognition system. NL-HMnet has no self-loop transition. The proposed algorithms for constructing Discrete-type HMnet and NL-HMnet require the pre-defined total number of states in HMnet. The optimum number of states can be changed by various factors such as kind of task, number of training samples, and so on. It is desirable to automatically determine the optimum number of states for each condition.

The HMnet with the optimum number of states means one showing the minimum perplexity for the test samples. In other words, we can choose the HMnet with the minimum perplexity (is equal to the maximum likelihood) for the test samples. If we can estimate the likelihood for the test samples using the training samples, the HMnet with the optimum number of states can be chosen.

In this paper, we have proposed an automatic determination algorithm for the optimum number of states. It can estimate the test set perplexity using only the training samples, so we can determine the optimum number of states in HMnet.

2 The determination algorithm

In order to estimate the test set perplexity using only the training samples, it can be thought to use a part of training samples as test samples. This idea was already used in the deleted interpolation algorithm[3] which was one of smoothing algorithms of n-gram. In this algorithm, the training samples are divided to two groups, a larger set and the remaining smaller set, n-gram is trained using the larger set, and the test set perplexity is calculated using the remaining smaller set. These steps are repeated by changing the training set and the test set, and then the optimum coefficients for all trials is determined.

If we use this idea to estimate the test set perplexity of the HMnet, we need to re-estimate the HMnet for each trial. However, as the re-estimation of the HMnet needs a large amount of calculation time, we cannot employ this idea. In this paper, instead of this idea, we propose another estimation algorithm of the test set perplexity. The new idea is to estimate the occurrence probability of
a single test sample generated from the unrelated path of the HMMnet with the test sample.

In the HMMnet, each training sample is assigned to a single path. We calculate \( \bar{P}(x) \) for the training sample \( x \) using Eq. (1).

\[
\bar{P}(x) = \frac{1}{N-1} \sum_{y \neq x}^{N} \text{Prob}_{\text{Path}(y)}(x)
\]  

(1)

where, \( \text{Path}(y) \) denotes a single path to which a sample \( y \) is assigned, \( \text{Prob}_{p}(x) \) denotes the occurrence probability of a sample \( x \) generated from the path \( p \). It can be assumed that \( \bar{P}(x) \) is the occurrence probability of a single “test” sample \( x \) generated from the HMMnet. Entropy \( H \) is calculated using Eq. (2).

\[
H = -\frac{1}{W} \sum_{x}^{N} \log(\bar{P}(x))
\]  

(2)

where, \( W \) denotes the total number of words in the training samples. We can estimate the perplexity using Eq. (3).

\[
P_{\text{crp}} = 2^{H}
\]  

(3)

In order to reduce the calculation time, we use Eq. (4) instead of Eq. (1) for \( \bar{P}(x) \), because the total number of paths is less or equal to the total number of training samples.

\[
\bar{P}(x) = \frac{1}{N-1} \sum_{p \neq \text{Path}(x)}^{P} \text{Prob}_{p}(x) \times \text{num}(p)
\]  

(4)

where, \( \text{num}(p) \) denotes the number of training samples assigned to a path \( p \), \( P \) denotes the total number of paths.

3 Experiments

The training and test samples were randomly generated from a finite-state automaton. We used two automata. One describes control commands for editing program (in Fig. 1), and it has 24 states and vocabulary size is 36. Another automaton describes an airport-traffic control commands, and it has 64 states and vocabulary size is 59. The number of training samples were set to 2000, 5000 and 10000, and the number of test samples was set to 2000.

MDL (Minimum Description Length) criterion[4] is frequently used for choosing the optimum order of a statistical model. We compared performance of the proposed algorithm with MDL. MDL is calculated using Eq. (5).

\[
\text{MDL} = -l + \frac{\alpha}{2} \log N + \log I
\]  

(5)
Table 1. Number of states and its perplexity (shown in a parenthesis) determined by each method. (Discrete-type HMnet)

<table>
<thead>
<tr>
<th>task</th>
<th># samples</th>
<th>minimum</th>
<th>estimated</th>
<th>MDL</th>
</tr>
</thead>
<tbody>
<tr>
<td>control</td>
<td>2000</td>
<td>60 (3.09)</td>
<td>40 (3.09)</td>
<td>31 (3.26)</td>
</tr>
<tr>
<td>command</td>
<td>5000</td>
<td>40 (3.01)</td>
<td>40 (3.01)</td>
<td>28 (3.09)</td>
</tr>
<tr>
<td>for editor</td>
<td>10000</td>
<td>30 (3.00)</td>
<td>31 (3.00)</td>
<td>76 (3.32)</td>
</tr>
<tr>
<td>airport</td>
<td>2000</td>
<td>110 (7.85)</td>
<td>108 (7.85)</td>
<td>32 (12.3)</td>
</tr>
<tr>
<td>control</td>
<td>5000</td>
<td>125 (7.72)</td>
<td>126 (7.72)</td>
<td>61 (9.04)</td>
</tr>
<tr>
<td>command</td>
<td>10000</td>
<td>140 (7.51)</td>
<td>132 (7.51)</td>
<td>110 (7.78)</td>
</tr>
</tbody>
</table>

where, $l$ denotes a log likelihood for training samples, $\alpha$ denotes number of free parameters, $N$ denotes number of training samples. $I$ denotes number of model and is set to be constant in this experiments.

Figure 2 shows the test set perplexity and the estimated one for the control commands for editing program. The number of training samples was set to 10000 in this experiment. From these results, the test set perplexity was correctly estimated by the proposed algorithm. The minimum perpextities were 50 states for Discrete-type HMnet, and 25 states for NL-HMnet. We can choose the optimum number of states using the estimated perplexity.

Figure 3 shows results with MDL in the same conditions. MDL gave 75 states for Discrete-type HMnet, and 40 states for NL-HMnet. The MDL shows larger numbers than the optimum numbers of states.

![Fig. 1. Grammar structure of control commands for editing program](image-url)
Table 2. Number of states and its perplexity (shown in a parenthesis) determined by each method. (NL-HMnet)

<table>
<thead>
<tr>
<th>task</th>
<th># samples</th>
<th>minimum</th>
<th>estimated</th>
<th>MDL</th>
</tr>
</thead>
<tbody>
<tr>
<td>control</td>
<td>2000</td>
<td>25 (3.00)</td>
<td>25 (3.00)</td>
<td>26 (3.09)</td>
</tr>
<tr>
<td>command</td>
<td>5000</td>
<td>25 (3.02)</td>
<td>24 (3.02)</td>
<td>28 (3.08)</td>
</tr>
<tr>
<td>for editor</td>
<td>10000</td>
<td>25 (3.00)</td>
<td>25 (3.00)</td>
<td>42 (3.23)</td>
</tr>
<tr>
<td>airport</td>
<td>2000</td>
<td>55 (7.02)</td>
<td>52 (7.19)</td>
<td>41 (7.56)</td>
</tr>
<tr>
<td>control</td>
<td>5000</td>
<td>55 (6.84)</td>
<td>55 (6.84)</td>
<td>99 (34.5)</td>
</tr>
<tr>
<td>command</td>
<td>10000</td>
<td>55 (6.82)</td>
<td>57 (6.82)</td>
<td>100 * (24.5)</td>
</tr>
</tbody>
</table>

Table 1 shows the number of states and its perplexity (shown in a parenthesis) determined by each method for Discrete-type HMnet, and table 2 shows those for NL-HMnet. “minimum” means the minimum perplexity computed using the test samples, “estimated” means the minimum perplexity computed by the proposed algorithm. From these tables, the proposed algorithm can determine the optimum number of states in every condition. On the other hand, MDL shows larger difference from the optimum number of stated shown in “minimum”. In table 1, there is small difference between the perplexity with MDL and the minimum perplexity. However, table 2 shows larger difference between them.

Fig. 2. Test set perplexity and estimated one.
4 Conclusion

We have proposed a method to determine the optimum number of states in Discrete-type HMnet and NL-HMnet. This algorithm can correctly estimate the test set perplexity. From experimental results, it can determine the optimum number of states even if MDL criterion cannot work.

References


Fig. 3. Relation between the number of states and MDL criterion.
Comparative Study of Automatic Acquisition Methods of Image Processing Procedures.

Toshihiro Hamada¹, Akinobu Shimizu², Toyofumi Saito³, Jun-ichi Hasegawa⁴, and Jun-ichiro Toriwaki³

¹ Kagawa Prefectural Industrial Technology Center,
Takamatsu, Kagawa 761-8031 hamada@itc.pref.kagawa.jp,
² Tokyo University of Agriculture and Technology, Kaganei, Tokyo 184-8588
³ Nagoya University, Nagoya, Aichi 464-8603
⁴ Chukyo University, Toyota, Aichi 470-0303

1 Introduction

Two image processing expert systems, IMPRESS[1] and IMPRESS-Pro[2] have already been developed by our research group. These systems automatically generate image processing procedures from sample pairs of an image and a sketch representing an object to be extracted from it. Automatic acquisition of the image processing procedure is a kind of knowledge discovery process, because an image processing procedure which can extract a figure like the given sketch from the image can be regarded as a representation of knowledge about the object in the image. In this paper, we examine above two expert systems comparatively from the knowledge discovery viewpoint.

2 Comparison between IMPRESS and IMPRESS-Pro

In the comparison, we use normal and abnormal images of LSI packages and sketch figures representing defect parts of the abnormal images. The outlines of IMPRESS and IMPRESS-Pro are shown in Fig.1. The input of IMPRESS is a set of sample pairs of an abnormal image and its sketch figure, and that of IMPRESS-Pro is a set of sample images consisting of normal and abnormal images and sketch figures. Each sketch figure is a binary image. IMPRESS can generate a procedure to extract the shape of sketch as precisely as possible. On the other hand, IMPRESS-Pro can generate a procedure which meets the demand about the misclassification rate per image.

Each system has in its database a sequence of local processes with kind of filters and parameters unified. While the sequence of local processes in IMPRESS consists of [smoothing-differentiation]-[binarization]-[connected component processing], that in IMPRESS-Pro consists of [smoothing-differentiation]-[binarization]-[classification of connected components]. In the process of each system, filters and parameters are fixed sequentially with evaluating the realizable performance for each stage. Finally, they generate their own best image processing procedures based on the each criterion for evaluation. The procedures acquired by two systems are expected to be different each other.

© Springer-Verlag Berlin Heidelberg 2000
3 Experiments

The image processing procedures acquired by IMPRESS and IMPRESS-Pro from a sample set (eighteen abnormal and normal images of LSI packages) are shown in Table 1. The requirements for IMPRESS-Pro are (False Positive rate) ≤ 7% and (False Negative rate) ≤ 7%. The procedures acquired by two systems are different each other. Fig. 2 and table 2 show the results of application of two procedures in table 1 to a test set(eighteen abnormal and normal real images). This result indicates that the procedure acquired by IMPRESS is more suitable than that by IMPRESS-Pro to extract the sketch figure with its shape as precisely as possible. On the other hand, the procedure acquired by IMPRESS-Pro is more effective than that by IMPRESS to control the misclassification rates.

The criterion used in IMPRESS in selecting [smoothing-differentiation] process is the index of separation calculated by comparing filtered pixel values in the sample figure with those in the background area. In IMPRESS-Pro, the number of false positive connected components is used as the criterion. Fig. 3 shows the joint distribution of evaluation values calculated by these criteria for all procedures. The procedures acquired by both systems are considered to be basically different, because the correlation between the values evaluated by two criteria is low.
4 Conclusion

In this paper, we compared two image processing expert systems from the viewpoint of automatic knowledge acquisition from images. It was confirmed that IMPRESS and IMPRESS-Pro generate the different procedures each other because of the different criteria for evaluation. Future research will be to examine the effect of search space reduction theoretically.

References

Table 1 The acquired procedures.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>IMPRESS</th>
<th>IMPRESS-Pro</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smoothing</td>
<td>5x5</td>
<td>3x3</td>
</tr>
<tr>
<td>Differentiation</td>
<td>8-Laplacian (r:41)</td>
<td>8-Laplacian (r:60)</td>
</tr>
<tr>
<td>Binarization</td>
<td>22</td>
<td>36</td>
</tr>
</tbody>
</table>

- Close-Open, Small component elimination: 34 pixels
- threshold of likelihood ratio: 0.161

Table 2 The performance of acquired procedures.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>IMPRESS</th>
<th>IMPRESS-Pro</th>
</tr>
</thead>
<tbody>
<tr>
<td>degree of coincidence</td>
<td>64%</td>
<td>47%</td>
</tr>
<tr>
<td>False Positive image</td>
<td>33%</td>
<td>0%</td>
</tr>
<tr>
<td>False Negative image</td>
<td>6%</td>
<td>17%</td>
</tr>
</tbody>
</table>

Fig. 3 The distribution of evaluation values for all procedures searched by the system.
Extraction of Authors’ Characteristics from Japanese Modern Sentences via N-gram Distribution

Tsukasa MATSUURA\(^1\) and Yasumasa KANADA\(^2\)

\(^1\) Department of Information Science, Faculty of Science, Graduate School of the University of Tokyo, 7-3-1, Hongo, Bunkyo, Tokyo, Japan (tsur@pci.cc.u-tokyo.ac.jp)

\(^2\) Information Technology Center, the University of Tokyo, 2-11-16, Yayoi, Bunkyo, Tokyo, Japan (kanada@pci.cc.u-tokyo.ac.jp)

1 Introduction

Objects of many studies of authorship attribution have been text data in which boundaries between words are obvious [1] [2]. When we apply these studies to languages in which sentences could not be divided obviously into words, such as Japanese or Chinese, preliminary processing of text data such as morphological analysis is required and may influence the final results. The methods which make use of characteristics of particular languages or particular compositions also have limited coverage [3]. Extracting authors’ characteristics from sentences is generally an unsolved problem. Therefore, we propose a method for authorship attribution based on distribution of n-grams of characters in sentences. The proposed method can analyze sentences without any additional information, i.e. preliminary analyses. The experiments, where 3-grams to represent author’s characteristics were deduced on the basis of their distributions, are also reported in the following.

2 Authors’ Characteristics in N-gram Distribution

Firstly, we introduce a measure of dissimilarity between two sets of text data. Let us assume that a small value of this measure suggests the high probability that a single author wrote the texts which are compared. A string \(x\) of \(n\) characters, i.e. an n-gram, is expressed as \(x=x_1x_2\cdots x_n\), where \(x_i\) is in an alphabet of the target language for \(i=1,\ldots, n\). The function \(P(x)\) represents a probability distribution function of n-grams \(x\) in text \(P\). We take notice of probability of appearance, not conditional probability at which \(x_n\) appears after \(x_1\cdots x_{n-1}\). Where text \(P\) and \(Q\) are given, we define set \(C\) by \(C=\{x|P(x)\cdot Q(x)\neq 0\}\). \(C\) is a set of n-grams which appear in both of text \(P\) and \(Q\), i.e. common n-grams. We measure resemblance between text \(P\) and text \(Q\) with function \(\text{dissim}\) defined by \(\text{dissim}(P; Q) = \frac{1}{\text{card}(C)} \sum_{x \in C} \log \frac{P(x)}{Q(x)}\). Here, “card(\(C\))” means the

© Springer-Verlag Berlin Heidelberg 2000
number of elements belonging to the set \( C \). *Dissim* is symmetrical concerning \( P \) and \( Q \). If only \( P(x) \) is the same as \( Q(x) \), the value of *dissim* becomes 0. Though there are many studies on functions to measure a resemblance between probability distribution, we take divergence (Kullback-Leibler information) as a comparable measure, which is defined as

\[
D(P||Q) = \sum_{x \in C} P(x) \log(\frac{P(x)}{Q(x)})
\]

3 Experiments on Authorship Attribution

The objects of analyses are 92 works in total, i.e., 72 novels, 9 essays, 5 letters, 3 scenarios, 2 diaries, and 1 speech, written from the *Meiji* period\(^1\) to the early *Showa* period\(^2\). The following list represents the authors' names and the numbers of their works analyzed.


The total number of characters included in the 92 works is 3,001,650. All the works are downloaded from the Aozora Bunko\(^3\). The 67 works of them follow the modern prescription of notation, i.e. modern kana\(^4\). The others are written in the old prescription, i.e. historical kana.

A blank and a line feed are counted as a character. Redundancies, i.e. lines which don’t include any characters, blanks after line feeds, are ignored. Characters which can be encoded in the JIS X 0208 are all counted. Each of the other characters is replaced with a single tag, which is counted as a character.

Credibility of *dissim* declines when volumes of texts to be compared differ greatly. Works are randomly combined until volumes of connected texts become 30,000 characters. The works longer than 30,000 characters were analyzed without combination. After consideration on the result of pre-experiments reported in [1], the volume of 30,000 characters was chosen. In order to observe the relation between volumes of texts and accuracy of authorship attribution, accuracy was also counted when texts from of 10,000 to of 30,000 characters were compared. Because accuracy might depend on way of combination of works, 50 ways of the random combination were tried and 50 text-sets to analyze were generated. To discuss what \( n \) of \( n \)-gram distribution is effective, the results of analyses from via 1-gram distribution to via 10-gram distribution are to be reported.

Here is a measure of accuracy of authorship attribution via percentage of success cases to all the comparisons in a text set. The number of comparisons is equal to the number of ways to choose a text as an origin in a comparison, i.e. the number of texts to be compared. A success case is defined as a case where all

\(^1\) The *Meiji* period is from 1868 to 1912.
\(^2\) The *Showa* period is from 1926 to 1989
\(^3\) http://www.aozora.gr.jp/main.html
\(^4\) A virtual library on the WWW which provides mainly literatures whose copyrights have terminated.

Japanese phonogram
the values of dissimilarity between the origin text and the same author’s texts are smaller than the minimum dissimilarity between the origin text and different authors’ texts. In a success case, when the dissimilarity of a text T is smaller than that of a text of the origin author, i.e. the author of the origin text, we can tell that text T is by the origin author. When a text-set has pairs of texts where any common n-grams are not found, authorship attributions which deal with the text-set are considered as fail cases.

In Figure 1, we can see the arithmetic means of accuracy via dissim and via divergence through the test on all the 50 text-sets. On dissim, 3-gram Distribution shows the highest scores and the maximum value was 96.0%, which was achieved with 30,000 character-long texts. Though no techniques equivalent to smoothing or flooring are adopted, accuracy of divergence is quite low. Dissim from with 2-gram distribution to with 4-gram distribution always shows better results than divergence with any n-gram distribution. Average accuracy via dissim tends to increase according to the increase of volumes of texts to be compared. Therefore, the credibility of dissim is supposed to improve when longer texts are analyzed.

![Graphs showing average accuracy via dissim and via divergence.](image)

**Fig. 1.** Average accuracy of authorship attribution via dissim and via divergence
(The arithmetic means of the accuracy on 50 text-set are represented.)

4 Extraction of Authors Characteristic N-grams

In this section, we try to extract 3-grams which represent characteristics of Ryunosuke AKUTAGAWA and Hiroshi KIKUCHI from a set of combined texts of 12,500-character-long. The text-set is one of the shortest sets where perfect authorship attributions were attained via 3-gram distribution in the experiments in section 3. Because the works analyzed in this section are published in 11 years from 1917 to 1927, the general change of literary style did not have a great effect on differences between these two authors’ styles.

The effect on dissim by n-gram x is defined by \( \text{share}(x) = | \log P(x)/Q(x) | \). N-grams whose distribution is distinctive of an author are supposed to represent
the author's characteristics. When we sorted 3-grams in order of $\text{share}(x)$, the value of $\text{share}(x)$ changed drastically around the mode\(^5\). Though the percentage of n-grams which show $\text{share}(x)$ less than or equal to the mode was 38.7% in average (standard variance was 2.63 points), the sum of their $\text{share}(x)$ was only 0.23% to value of $\text{dissim}$ in average (standard variance was 0.14 points). Therefore, we define characteristic n-grams in a comparison between texts as n-grams x such that $\text{share}(x)$ is greater than the mode.

The sets of characteristic n-grams in comparisons among texts of a single author represent the differences in contents, not in styles. They are named content constrained n-grams and should be eliminated from authors' characteristic n-grams, which represent authors' characteristics. If the author who uses the n-grams more frequently alters case by case, the n-grams should also be eliminated because they are difficult to represent author's characteristics. N-grams which appear as characteristic n-grams in only one comparison are illfounded, so that they should be also eliminated.

Set A is defined by the union of the sets of characteristic n-grams in comparisons between the two authors. Set B is the union of the sets of content constrained n-grams. When $A \cap B$ is considered as authors' characteristic 3-grams, the 196 kinds of Authors' characteristic 3-grams were educed. The top 20 kinds of 3-grams of the greatest $\text{share}(x)$ are shown in Figure 1. The top 20 kinds of 3-grams whose probabilities of appearance are the greatest in sentences of each author are also listed in Figure 1. If differences between the modern prescription of notation and old one are taken into consideration, the 12 kinds of 3-gram of the greatest probabilities are common to the two authors. Furthermore, they consist of parts of functional words which are supposed to appear frequently in general sentences. 3-grams of greater probabilities cannot be clues to distinguish each author's characteristics. On the other hand, Authors' characteristic 3-grams include parts of meaningful words and don't contain n-grams common to the 40 kinds of 3-grams of the greatest probabilities. These Authors' characteristic 3-grams will contribute not only to the improvement of authorship attribution methods but also to the studies of literature or stylistics.

5 Conclusion and Future Work

$\text{Dissim}$ achieved average accuracy of 96.0% at highest via 3-gram distribution. $\text{Dissim}$ could extract authors' characteristics common to works in various genres. Divergence was not effective in these experiments. We suppose that $\text{dissim}$ achieved high accuracy because (1) $\text{dissim}$ is normalized on the number of common n-grams, (2) absolute values of ratios of $P(x)$ to $Q(x)$ are added in $\text{dissim}$.

Our study on extraction of authors' fingerprints is in the stage of examining validity of the proposed method. Texts of more authors, of more languages and of more kinds should be taken into experiments. More random combinations to connect compositions should also be experimented. Thereafter, we will

\(^5\) a value which appears most frequently in distribution
Table 1. Author’s characteristic 3-grams and 3-grams of the greatest appearance probabilities (*@n* means a carriage return.)

<table>
<thead>
<tr>
<th>rank</th>
<th>Authors’ Characteristic 3-grams</th>
<th>3-gram share</th>
<th>author who use 3-gram more frequently</th>
<th>3-grams of greater appearance probabilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>れは彼</td>
<td>6.82E-04</td>
<td>AKUTAGAWA</td>
<td>つた。</td>
</tr>
<tr>
<td>2</td>
<td>き上げ</td>
<td>4.05E-04</td>
<td>KIKUCHI</td>
<td>した。</td>
</tr>
<tr>
<td>3</td>
<td>同人雑</td>
<td>3.23E-04</td>
<td>KIKUCHI</td>
<td>た。@n</td>
</tr>
<tr>
<td>4</td>
<td>人雑誌</td>
<td>3.23E-04</td>
<td>KIKUCHI</td>
<td>てるた</td>
</tr>
<tr>
<td>5</td>
<td>の仕事</td>
<td>3.04E-04</td>
<td>KIKUCHI</td>
<td>た。@n</td>
</tr>
<tr>
<td>6</td>
<td>に似れ</td>
<td>2.89E-04</td>
<td>KIKUCHI</td>
<td>るた。</td>
</tr>
<tr>
<td>7</td>
<td>完全に</td>
<td>2.83E-04</td>
<td>AKUTAGAWA</td>
<td>てるる</td>
</tr>
<tr>
<td>8</td>
<td>たれは</td>
<td>2.45E-04</td>
<td>AKUTAGAWA</td>
<td>だつた</td>
</tr>
<tr>
<td>9</td>
<td>ようで</td>
<td>2.34E-04</td>
<td>AKUTAGAWA</td>
<td>かつた</td>
</tr>
<tr>
<td>10</td>
<td>よし</td>
<td>2.31E-04</td>
<td>AKUTAGAWA</td>
<td>が。</td>
</tr>
<tr>
<td>11</td>
<td>やつ</td>
<td>2.26E-04</td>
<td>KIKUCHI</td>
<td>ながら</td>
</tr>
<tr>
<td>12</td>
<td>現して</td>
<td>2.23E-04</td>
<td>AKUTAGAWA</td>
<td>た。</td>
</tr>
<tr>
<td>13</td>
<td>書き上</td>
<td>2.19E-04</td>
<td>KIKUCHI</td>
<td>がら、</td>
</tr>
<tr>
<td>14</td>
<td>生活で</td>
<td>2.15E-04</td>
<td>AKUTAGAWA</td>
<td>つてる</td>
</tr>
<tr>
<td>15</td>
<td>か？</td>
<td>2.15E-04</td>
<td>AKUTAGAWA</td>
<td>それ</td>
</tr>
<tr>
<td>16</td>
<td>人とも</td>
<td>2.13E-04</td>
<td>AKUTAGAWA</td>
<td>@n「</td>
</tr>
<tr>
<td>17</td>
<td>が。</td>
<td>2.06E-04</td>
<td>AKUTAGAWA</td>
<td>なかつ</td>
</tr>
<tr>
<td>18</td>
<td>も。</td>
<td>2.04E-04</td>
<td>AKUTAGAWA</td>
<td>な。</td>
</tr>
<tr>
<td>19</td>
<td>なんか</td>
<td>2.00E-04</td>
<td>AKUTAGAWA</td>
<td>た。彼</td>
</tr>
<tr>
<td>20</td>
<td>のつい</td>
<td>2.00E-04</td>
<td>AKUTAGAWA</td>
<td>た。</td>
</tr>
</tbody>
</table>

be able to give answers to the question that why dissim represented the difference between authors’ styles better than divergence. These conclusions will be meaningful clues when we consider what are desirable characteristics for similarity measures among probability distribution in the area of natural language processing. The final aim of our study is automatic discovery of methods for grasping characteristics from general or particular sentences.

References

Combination Retrieval for Creating Knowledge from Sparse Document Collection

Naohiro Matsumura¹ and Yukio Ohsawa²

¹ Graduate School of Electronic Engineering, University of Tokyo
7-3-1 Bunkyo-ku, Tokyo, 113-0013, Japan
² Graduate School of Systems Management, University of Tsukuba
3-29-1 Otsuka, Bunkyo-ku, Tokyo, 112-0012, Japan

Abstract. With the variety of human life, people are interested in various matters for each one's unique reason, for which a machine maybe a better counselor than a human. This paper proposes to help user create novel knowledge by combining multiple existing documents, even if the document collection is sparse i.e. if a query in the domain has no corresponding answer in the collection. This novel knowledge realizes an answer to a user’s unique question, which can not be answered by a single recorded document. In the Combination Retriever implemented here, cost-based abduction is employed for selecting and combining appropriate documents for making a readable and context-reflecting answer. Empirically, Combination Retriever obtained satisfactory answers to user’s unique questions.

1 Introduction

People are interested in personal and unique matters, e.g. very rare health condition, friction with friends, etc. They often hesitate to consult a human about such unique matters, and worry in their own minds. In such a case, entering such interests to a search engine and reading the output documents is a convenient way which may serve satisfactory information.

However, a document collection of a search engine, even though they may seem to include a lot of documents, is too sparse for answering a unique question: They have only past information not satisfactory for answering novel queries. For overcoming this situation, a search engine is desired to help user create knowledge from sparse documents.

For this purpose, we propose a novel information retrieval method named combination retrieval. The basic idea is that an appropriate combination of existing documents may lead to creating novel knowledge, although each one document may be short of answering the novel query. Based on the principle that combining ideas triggers the creation of new ideas [1], we present a system to obtain and present an optimal combination of documents to the user,
optimal in that the solution forms a document-set which is the most readable (understandable) and reflecting the user’s context.

The remainder of this paper goes as follows: In Section 2, the mechanism of the implemented system Combination Retriever is described. We show the experiments and the results in Section 3, showing the performance of Combination Retriever for medical counseling question-and-answer documents.

2 The Process of Combination Retriever

For realizing combination retrieval, we need a method for selecting meaningful documents which, as a set, serve a good (readable and meaningful) answer to the user. Here we show our approach implemented as a system called Combination Retriever, where abductive inference is used for the selection of documents to be combined. The process of Combination Retriever is as follows:

The process of Combination Retriever

Step 1) Accept user’s query $Q_g$.
Step 2) Obtain $G$, a word-set representing the goal user wants to understand, from $Q_g$ ($G = Q_g$ if $Q_g$ is given simply as a word-set).
Step 3) Make knowledge-base $\Sigma$ for the abduction of Step 4). For each document $D_x$ in the document-collection $C_{doc}$, a Horn clause is made as to describe the condition (words needed to be understood for reading $D_x$) for and the effect (words to be subsequently understood by reading $D_x$) of reading document $D_x$.
Step 4) Obtain $h$, the optimal hypothesis-set which derives $G$ by being combined with $\Sigma$, by cost-based abduction (CBA, hereafter)[2]. $h$ obtained here represents the union of following information, of the least size of $K$.

- $S$: The document-set the user should read.
- $K$: The keyword-set the user should understand by other information source than the document collection $C_{doc}$, for reading the documents in $S$.
Step 5) Show the documents in $S$ to the user.

The intuitive meaning of the abductive inference is to obtain the conditions for understanding user’s goal $G$. Those conditions include the documents to read ($S$) for understanding $G$, and necessary knowledge ($K$) for reading those documents. That is, $S$ means the document-combination we aim to present to the user.

2.1 An Example of Combination Retriever’s Execution

For example, Combination Retriever runs as follows.

Step 1) $Q_g = \text{“Does alcohol cause a liver cancer ?”}$
Step 2) $G$ is obtained from $Q_g$ as $\{\text{alcohol, liver, cancer}\}$. 
Step 3) From $C_{doc}$, documents $D_1, D_2$, and $D_3$ are taken, each including terms in $G$, and put into Horn clauses as:

- $alcohol : \neg \text{cirrhosis, cell, disease, } D_1$
- $liver : \neg \text{cirrhosis, cell, disease, } D_1$
- $alcohol : \neg \text{marijuana, drug, health, } D_2$
- $liver : \neg \text{marijuana, drug, health, } D_2$
- $alcohol : \neg \text{cell, disease, organ, } D_3$
- $cancer : \neg \text{cell, disease, organ, } D_3$

Hypothesis-set $H$ is formed of the conditional parts here, of $D_1$, $D_2$ and $D_3$ of Type 1 \(^1\) each weighted 0, and “cirrhosis,” “cell,” “disease,” “marijuana,” “drug,” “health,” and “organ” of Type 2 \(^2\) each weighted 1.

Step 4) $h$ is obtained as $S \cup K$, where

\[
S = \{ D_1, D_3 \} \quad \text{and} \quad K = \{ \text{cirrhosis, cell, disease, organ} \}.
\]

meaning that user should understand “cirrhosis,” “cell,” “disease” and “organ” for reading $D_1$ and $D_3$, served as the answer to $Q_g$. This solution is selected because $cost(h)$ takes the values of 4, less than 6 of the only alternative feasible solution, i.e. $\{ \text{marijuana, drug, health, cell, disease, organ} \}$ plus $\{ D_2, D_3 \}$.

Step 5) User now reads the two documents presented as:

$D_1$ (including alcohol and liver) stating that alcohol alters the liver function by changing liver cells into cirrhosis.

$D_3$ (including alcohol and cancer) showing the causes of cancer in various organs, including a lot of alcohol. This document recommends drinkers to limit to one ounce of pure alcohol per day.

As a result, the subject learns that he should limit drinking to keep liver healthy and avoid cancer, and also came to know that other tissues than liver get cancer from alcohol.

Thus, user can understand the answer by learning a small number of words from outside of $C_{doc}$, as we aimed in employing CBA. More importantly than this major effect of Combination Retriever, a by-product is that the common hypotheses between $D_1$ and $D_3$, i.e., $\{ \text{cell, disease} \}$ of Type 2 are discovered as the context of user’s interest underlying the entered words. This effect is due to CBA which obtains the smallest number of involved contexts, for explaining the goal (i.e. answering the query), as solution hypotheses. Presenting such a novel and meaningful context to the user induces the user to create new knowledge \(^5\), to satisfy his/her novel interest.

\(^1\) Hypothesis that user reads a document in $C_{doc}$.
\(^2\) Hypothesis that user knows (learns) a conditional term in $C_{doc}$. 

---

Naohiro Matsumura and Yukio Ohsawa
3 Experimental evaluations

3.1 The experimental conditions

Combination Retriever was applied to $C_{\text{doc}}$ of 1320 question-answer pairs from a health care question answering service on WWW (*Alice*, http://www.alice.columbia.edu). Past clients of *Alice* asked about personal anxiety or interest in health and a medical counselor answered them.

![Image](image_url)

**Fig. 1.** An output of Combination Retriever, showing two past answers 0323 and 0613 (document IDs in $C_{\text{doc}}$) for input query `{alcohol, cancer, liver}`.

3.2 Result Statistics

The test was executed for 5 subjects from 21 to 30 years old accustomed to using a Web browser. This means that the subjects were of the near age to the past
question askers of Alice and entered queries into the CGI interface and read the answers of Combination Retriever smoothly.

Here, 63 queries were entered. This seems to be quite a small number for the evaluation data. However, we compromised with this size of data for two reasons. First, we aimed at having each subject evaluate the returned answer in a natural manner. That is, in order to have the subject report whether he/she was really satisfied with the output of Combination Retriever, the subject must enter his/her real anxiety or interest. Otherwise, the subject has to imagine an unreal person who asks the query and imagine what the unreal person feels with the returned answers. Therefore we restricted to a small number of queries entered from real interests. Second, the results show significant superiority of Combination Retriever as follows, even though the test data was small.

The overall result was that Combination Retriever satisfied 43 of the 63 queries, while VFAQ satisfied only 26 queries. Next, let us show more detailed results, which show that Combination Retriever works especially for novel queries.

4 Conclusions

We proposed to help user create novel knowledge, by combining and presenting multiple existing documents. This novel knowledge realizes an answer to user’s unique question, which can not be answered by a single document.

This high-performance comes from obtaining minimal-cost hypothesis in CBA. That is, a document-set in a meaningful context can be obtained, because CBA discovers relevant context according to user’s query, by minimizing the number of conditional terms for reading output documents. This means that the user and the system can ask and answer under a meaningful context, which supports a meaningful communication. From such a novel and meaningful context presented, the user can create new knowledge which realizes a satisfaction of his/her unique interest. This is a significant by-product of minimizing the cost of output-documents for obtaining an answer easy to read.

References

Discovery of Nominally Conditioned Polynomials using Neural Networks, Vector Quantizers and Decision Trees

Kazumi Saito¹ and Ryohei Nakano²

¹ NTT Communication Science Laboratories
2-4 Hikaridai, Seika, Soraku, Kyoto 619-0237 Japan
saito@cslab.kecl.ntt.co.jp

² Nagoya Institute of Technology
Gokiso-cho, Showa-ku, Nagoya 466-8555 Japan
nakano@ics.nitech.ac.jp

1 Introduction
The discovery of a numeric law, e.g., Kepler's third law \( T = kr^{3/2} \), from data is the central part of scientific discovery systems. The data considered in many real fields usually contains both nominal and numeric values. Thus, we consider discovering a law that governs such data in the form of a rule set of nominally conditioned polynomials. Recently, a connectionist method called RF6 [1] was proposed to solve problems of this type. RF6 can learn multiple nominally conditioned polynomials with single neural networks; besides, RF6 can discover generalized polynomials whose power values are not restricted to integers. However, for real complex problems, RF6 will suffer from a combinatorial explosion in the process of restoring rules from a trained neural network. Therefore, this paper proposes a new version of RF6 by greatly improving its procedure of restoring nominally conditioned polynomials from a trained neural network.

2 Restoring Nominally Conditioned Polynomials
Let \( \{q_1, \cdots, q_{K_1}, x_1, \cdots, x_{K_2}, y\} \) be a set of variables describing an example, where \( q_k \) is a nominal explanatory variable, \( x_k \) is a numeric explanatory variable and \( y \) is a numeric target variable. For each \( q_k \) we introduce a dummy variable expressed by \( q_{kl} \), i.e., \( q_{kl} = 1 \) if \( q_k \) matches the \( l \)-th category; otherwise 0, where \( l = 1, \cdots, L_k \), and \( L_k \) is the number of distinct categories appearing in \( q_k \). As a true model governing data, we consider the following set of multiple nominally conditioned polynomials whose power values are not restricted to integers.

\[
\text{if} \bigwedge_{q_{kl} \in Q^i} q_{kl} = 1 \text{ then } y(x; \Theta^i) = w^i_0 + \sum_{j=1}^{J^i} w^i_j \prod_{k=1}^{K_2} x^j_k, \quad i = 1, \cdots, I^* \quad (1)
\]

where \( I^* \) is the number of rules, \( Q^i \) denotes a set of dummy variables corresponding to the \( i \)-th nominal condition and \( \Theta^i \) is a parameter vector used in the \( i \)-th generalized polynomial. Here, each parameter \( w^i_j \) or \( w^i_{jk} \) is a real number, and \( J^i \) is an integer corresponding to the number of terms.

© Springer-Verlag Berlin Heidelberg 2000
Consider a function $c(q; V) = e^{\sum_{k=1}^{K_1} \sum_{l=1}^{L_k} v_{kl} q_{kl}}$, where $V$ denotes a vector of parameters $v_{kl}$. We can show [1] that with an adequate number $J$, certain type of neural network $y(q, x; \Theta) = w_0 + \sum_{j=1}^{J} w_j \exp(\sum_{k=1}^{K_1} \sum_{l=1}^{L_k} v_{jkl} q_{kl} + \sum_{k=1}^{K_2} w_{jkl} \ln x_k)$ can closely approximate Eq. (1). Let $D = \{(q^\mu, x^\mu, y^\mu) : \mu = 1, \ldots, N\}$ be a set of training data, where $N$ is the number of examples. Then, each parameter can be estimated by minimizing an objective function $E(\Theta) = \sum_{\mu=1}^{N} (y^\mu - y(q^\mu, x^\mu; \Theta))^2 + \Theta^T \Lambda \Theta$, where a penalty term is added to improve both the generalization performance and the readability of the learning results.

### 2.1 Restoring Procedures

Assume that we have already obtained a neural network trained as the best law-candidate. In order to restore a set of nominally conditioned polynomials as described in Eq. (1), we need a suitable efficient procedure.

RF6 [1] has a decomposition procedure for this purpose; i.e., a set of nominally conditioned terms is extracted from each hidden unit, and then each of these terms is in turn combined through all of the hidden units. When $\alpha$ denotes the average number of terms over each hidden unit, the total number of these combined terms approximately amounts to $\alpha J$. Thus, as the number of hidden units or the number of nominal variables increases, this procedure comes to suffer from a combinatorial explosion.

As another approach, we can extract nominally conditioned polynomials for each training example, and simply assemble them to obtain a final set of rules as a law. Then, the following set of nominally conditioned polynomials can be obtained directly from the training data and the trained neural network.

\[
\text{if } \bigwedge_{\{k,l|q_{kl}=1\}} \text{ then } y = \hat{w}_0 + \sum_{j=1}^{J} c_{ij} \prod_{k=1}^{K_2} x_{ik}^{\hat{w}_{jk}}, \quad \mu = 1, \ldots, N, \quad (2)
\]

where $c_{ij}^\mu$ denotes the $j$-th coefficient calculated from the nominal values of the $\mu$-th training example, i.e., $c_{ij}^\mu = \hat{w}_j \exp(\sum_{k=1}^{K_1} \sum_{l=1}^{L_k} \hat{v}_{jkl} q_{kl}^\mu)$. However, in comparison with the true model governing the data defined in Eq. (1), the results of this naive procedure can be far from desirable because they will contain a large number of similar polynomials, and each nominal condition will be too specific in terms of representing only one training example.

Based on the above considerations, we propose a new restoring procedure.

**step1. finding subspace representatives**

In order to find subspace representatives, a set of coefficient vectors $\{c^\mu = (c_{1}^\mu, \cdots, c_{J}^\mu)^T : \mu = 1, \cdots, N\}$ calculated from the training data is quantized into a set of representative vectors $\{r^i = (r_{1i}, \cdots, r_{Ji})^T : i = 1, \cdots, I\}$, where $I$ is the number of representatives. Among several vector quantization (VQ) algorithms, we employ the k-means algorithm due to its simplicity.

**step2. criterion for model selection**

Consider the following set of rules using the representative vectors.

\[
\text{if } i(q) = i \text{ then } y = \hat{w}_0 + \sum_{j=1}^{J} r_{ij} \prod_{k=1}^{K_2} x_{ik}^{\hat{w}_{jk}}, \quad i = 1, \cdots, I, \quad (3)
\]
where \( i(q) \) denotes a function that returns the index of the representative vector minimizing the distance, i.e., \( i(q) = \arg\min_j \sum_{i=1}^K (c_j - r_j)^2 \). Here, since each element of \( c \) is calculated as \( c_j = \hat{w}_j \exp(\sum_{k=1}^K \sum_{l=1}^{I_q} \hat{w}_{jl}(q_{kl})) \), Eq. (3) can be applied to a new example, as well as the training examples. Thus, to determine an adequate number \( I \) of representatives, we employ the procedure of cross-validation which divides the data \( D \) at random into \( S \) distinct segments \( \{D^s : s = 1, \cdots, S\} \). Namely, by using the final weights \( \Theta^\top \) trained without data segment \( D^s \), we can define a cross-validation error function \( CV = N^{-1} \sum_{s=1}^S \sum_{i \in D^s} (y^i - \hat{w}_0^i + \sum_{j=1}^J r_j^i(q^i) \prod_{k=1}^{K^2} (x_{kl}(q^i)_k))^2 \).

**step 3. generating conditional parts**

The indexing functions \( \{i(q)\} \) described in Eq. (3) must be transformed into a set of nominal conditions as described in Eq. (1). One reasonable approach is to perform this transformation by solving a classification problem whose training examples are \( \{(q^\mu, i(q^\mu)) : \mu = 1, \cdots, N\} \), where \( i(q^\mu) \) indicates the class label of a training example \( q^\mu \). For this classification problem, we employ the c4.5 decision tree generation program due to its wide availability. From the generated decision tree, we can easily obtain the final rule set as described in Eq. (1).

Clearly, these steps can be executed within the computational complexity of linear order with respect to the numbers of training examples, variables, hidden units, representatives, iterations performed by the k-means algorithm, and data segments used concerning cross-validation; i.e., this new restoring procedure can be much more efficient than the old decomposition procedure which requires the computational complexity of exponential order. Hereafter, the law discovery method using the above restoring procedure is called RF6.2.

**2.2 Evaluation by Experiments**

By using three data sets, we evaluated the performance of RF6.2. In the k-means algorithm, initial representative vectors \( \{r^I\} \) are randomly selected as a subset of coefficient vectors \( \{c^I\} \). For each \( I \), trials are repeated 100 times with different initial values, and the best result is reported. The cross-validation error is calculated by using the leave-one-out method, i.e., \( S = N \). The candidate number \( I \) of representative vectors is incremented in turn from 1 until the cross-validation error increases. The c4.5 program is used with the initial settings.

**Artificial data set.**

We consider an artificial law described by

\[
\begin{align*}
\text{if } q_{21} = 1 \land (q_{31} = 1 \lor q_{33} = 1) \text{ then } y &= 2 + 3x_1^{-1}x_2^3 + 4x_3x_4^{1/2}x_5^{-1/3} \\
\text{if } q_{21} = 0 \land (q_{32} = 1 \lor q_{34} = 1) \text{ then } y &= 2 + 5x_1^{-1}x_2^3 + 2x_3x_4^{1/2}x_5^{-1/3} \\
\text{else } \text{then } y &= 2 + 4x_1^{-1}x_2^3 + 3x_3x_4^{1/2}x_5^{-1/3}
\end{align*}
\]

where we have three nominal and nine numeric explanatory variables, and the numbers of categories of \( q_1, q_2 \) and \( q_3 \) are set as \( L_1 = 2, L_2 = 3 \) and \( L_3 = 4 \), respectively. Clearly, variables \( q_1, x_6, \cdots, x_9 \) are irrelevant to Eq. (4). Each value of nominal variables \( q_1, q_2, q_3 \) is randomly generated so that only one dummy variable becomes 1, each value of numeric variables \( x_1, \cdots, x_9 \) is randomly generated in the range of \((0, 1)\), and we get the corresponding value of \( y \) by calculating
Eq. (4) and adding Gaussian noise with a mean of 0 and a standard deviation of 0.1. The number of examples is set to 400.

In this experiment, a neural network was trained by setting the number of hidden units $J$ to 2. We examined the performance of the experimental results obtained by applying the k-means algorithm with the different number of representative vectors, where the RMSE (root mean squared error) was used for the evaluation; the training error was evaluated as a rule set by using Eq. (3); the cross-validation error was calculated by using the function $CV$; and the generalization error was also evaluated as a rule set and measured by using a set of noise-free 10,000 test examples generated independently of the training examples. The experimental results showed that the training error almost monotonically decreased ($2.090, 0.828, 0.142$, and $0.142$ for $I = 1, 2, 3,$ and $4$, respectively); the cross-validation error was minimized when $I = 3$ ($2.097, 0.841, 0.156,$ and $0.160$ for $I = 1, 2, 3,$ and $4$, respectively, i.e., indicating that an adequate number of representative vectors is 3); and the generalization error was also minimized when $I = 3$ ($2.814, 1.437, 0.320,$ and $0.322$ for $I = 1, 2, 3,$ and $4$, respectively). Since the cross-validation and generalization errors were minimized with the same number of representative vectors, we can consequently see that the desirable model was selected by using the cross-validation.

By applying the c4.5 program, we obtained the following decision tree whose leaf nodes correspond to the following.

$$q_{21} = 0: q_{34} = 1: 2 (83.0) \quad \Leftrightarrow \quad r^2 = (+5.04, +2.13)$$

$$\quad \quad | \quad q_{34} = 0: q_{32} = 0: 3 (129.0) \quad \Leftrightarrow \quad r^3 = (+3.96, +2.97)$$

$$\quad \quad \quad \quad | \quad q_{32} = 1: 2 (53.0) \quad \Leftrightarrow \quad r^2 = (+5.04, +2.13)$$

$$q_{21} = 1: q_{34} = 1: 3 (36.0) \quad \Leftrightarrow \quad r^3 = (+3.96, +2.97)$$

$$\quad \quad | \quad q_{34} = 0: q_{32} = 0: 1 (73.0) \quad \Leftrightarrow \quad r^1 = (+3.10, +4.07)$$

$$\quad \quad \quad \quad | \quad q_{32} = 1: 3 (26.0) \quad \Leftrightarrow \quad r^3 = (+3.96, +2.97)$$

where the coefficient values were rounded off to the second decimal place; each number of training examples arriving at the corresponding leaf node is shown in parenthesis. Then, the following rule set was straightforwardly obtained.

$$\begin{cases} 
if \quad q_{21} = 1 \land (q_{31} = 1 \lor q_{33} = 1) \\
then \quad y = 2.01 + 3.10x_1^{-1.00}x_2^{+3.01} + 4.07x_3^{+1.02}x_4^{+0.51}x_5^{-0.33} \\
\end{cases}$$

$$\begin{cases} 
if \quad q_{21} = 0 \land (q_{32} = 1 \lor q_{34} = 1) \\
then \quad y = 2.01 + 5.04x_1^{-1.00}x_2^{+3.01} + 2.13x_3^{+1.02}x_4^{+0.51}x_5^{-0.33} \\
else \quad then \quad y = 2.01 + 3.96x_1^{-1.00}x_2^{+3.01} + 2.97x_3^{+1.02}x_4^{+0.51}x_5^{+0.33}. \\
\end{cases} \quad (5)$$

Recall that each nominal variable matches only one category, e.g., $(q_{32} = 1 \land q_{34} = 0) \equiv (q_{32} = 1)$. Therefore, although some of the weight values were slightly different, we can see that a law almost equivalent to the true one was found.

**Financial data set.**

We performed an experimental study to discover underlying laws of market capitalization from six fundamental BS (Balance Sheet) items and the type of industry (the 33 classifications of the Tokyo Stock Exchange). Our experiments used data from 953 companies listed on the first section of the TSE, where banks,
and insurance, securities and recently listed companies were excluded. In order to understand the effect of the nominal variable intuitively, the number of hidden units was fixed at 1. The cross-validation error was minimized at \( I = 3 \). Then, the following rule set was obtained.

\[
\text{if } \bigvee_{q_l \in Q_i} q_l = 1 \text{ then } y = 12891.6 + r^1 x_2^{+0.668} x_3^{+1.043} x_6^{-0.747}, \quad i = 1, 2, 3 \quad (6)
\]

where \( r^1 = +1.907, r^2 = +1.122, r^3 = +0.657 \) and each of the nominal conditions was as follow: \( Q^1 = \{ \text{"Pharmaceuticals"}, \text{"Rubber Products"}, \text{"Metal Products"}, \text{"Machinery"}, \text{"Electrical Machinery"}, \text{"Transport Equipment"}, \text{"Precision Instruments"}, \text{"Other Products"}, \text{"Communications"}, \text{"Services"} \}; Q^2 = \{ \text{"Foods"}, \text{"Textiles"}, \text{"Pulp & Paper"}, \text{"Chemicals"}, \text{"Glass & Ceramics"}, \text{"Non-ferrous Metals"}, \text{"Maritime Transport"}, \text{"Retail Trade"} \}; \) \( Q^3 = \{ \text{"Fisheries"}, \text{"Mining"}, \text{"Construction"}, \text{"Oil & Coal Products"}, \text{"Iron & Steel"}, \text{"Electricity & Gas"}, \text{"Land Transport"}, \text{"Air Transport"}, \text{"Wearhousing"}, \text{"Wholesale"}, \text{"Other Financing Business"}, \text{"Real Estate"} \}. \) Since the second term on the right hand side of the polynomials appearing in Eq. (6) is always positive, each of the coefficient values \( r^i \) can indicate the stock price setting tendency of industry groups in similar BS situations, i.e., the discovered law tells us that industries appearing in \( Q^1 \) are likely to have a high setting, while those in \( Q^3 \) are likely to have a low setting.

**Automobile data set.**

The Automobile data set contained data on the car and truck specifications in 1985, and was used to predict prices based on these specifications. The data set had 159 examples with no missing values, and consisted of 10 nominal and 14 numeric explanatory variables and one target variable (price). In this experiment, since the number of examples was small, the number of hidden units was also set to 1. The cross-validation error was minimized at \( I = 3 \). The polynomial part of the discovered law was as follows:

\[
y = 1163.16 + r^1 x_2^{+1.638} x_4^{+0.046} x_5^{-1.436} x_6^{+0.997} x_9^{-0.245} x_{13}^{-0.071} \quad (7)
\]

where \( r^1 = +1.453, r^2 = +1.038, r^3 = +0.763 \) and the relatively simple nominal conditions were obtained. Similarly as described for the experiments using the financial data set, since the second term on the right hand side of Eq. (7) is always positive, the coefficient value \( r^1 \) can indicate the car price setting tendency for similar specifications. Actually, the discovered law verbally told us that cars of a high price setting are: “5-cylinder ones”, “BMW’s”, “convertibles”, “VOLVO turbos”, “SAAB turbos”, and “6-cylinder turbos”; cars of a middle price setting are: “PEUGOT’s”, “VOLVO non-turbos”, “SAAB non-turbos”, “HONDA 1bbl-fuel-system ones”, “MAZDA fair-risk-level ones”, “non-BMW non-turbos & 6-cylinder ones”, “non-5-cylinder turbos & fair-risk-level ones”; and other cars are of a low price setting.

**Reference**

Author Index

Akimoto, Yousuke, 255
Akutsu, Tatsuya, 86
Amano, Kazuyuki, 155
Araki, Tohru, 197
Arikawa, Setsuo, 141

Belkina, Natalya V., 240
Devaney, Judith, 247
Elouedi, Zied, 127
Fukuda, Tomoko, 112
Goto, Yoshitaka, 258
Grieser, Gunter, 50
Hagiya, Masami, 17
Halova, Jaroslava, 261
Hamada, Toshihiro, 311
Hamuro, Yukinobu, 263
Haraguchi, Makoto, 295
Hasegawa, Hiroshi H., 304
Hasegawa, Jun-ichi, 311
Hatakeyama, Eiko, 212
Higuchi, Tomoyuki, 197, 242
Hirao, Masahiro, 141
Hirokawa, Sachio, 299
Horiiuchi, Tadashi, 99
Hoshino, Hiromasa, 141
Ikeda, Daisuke, 299
Ishimiya, Yukari, 304
Jantke, Klaus P., 50

Kakimoto, Mitsuru, 212
Kanada, Yasumasa, 315
Kasahara, Yoshiya, 255, 258
Katoh, Naoki, 263
Kikuchi, Yoshiaki, 212
Kim, Hyeoncheol, 170
Kohonon, Teuvo, 76
Krepets, Vladimir V., 240
Kudoh, Yoshimitsu, 295
Kuhara, Satoru, 86
Lange, Steffen, 50
Machida, Shinobu, 197
Makino, Shozo, 306
Maruoka, Akira, 155
Maruyama, Kohei, 285
Matsuda, Takashi, 99
Matsumoto, Tetsuya, 112
Matsumura, Naohiro, 320
Matsuura, Tsukasa, 315
Mellouli, Khaled, 127
Miwa, Kazuhiisa, 225
Miyano, Satoru, 86
Miyazaki, Yoshiumi, 212
Mizoguchi, Fumio, 38
Mizuno, Jun, 155
Morita, Chie, 212
Motoda, Hiroshi, 99
Mukouchi, Yasuhito, 183
Murata, Tsuyoshi, 65
Nakamura, Nagatomo, 197
Nakano, Ryohei, 325
Nanri, Ichiro, 112
Niimi, Ayahiko, 273
Niitsu, Ryotaro, 255
Ohsawa, Yukio, 320
Ohwada, Hayato, 38
Rauch, Jan, 268
Saito, Kazumi, 325
Saito, Takeshi, 304
Saito, Toyofumi, 311
Sato, Masako, 183
Sato, Toru, 255, 258
Shimada, Mitsuomi, 290
Shimizu, Akinobu, 311
Shinohara, Ayumi, 141
Simunek, Milan, 268
Somervuo, Panu, 76
Suzuki, Einoshin, 282
Suzuki, Motoyuki, 306
Takahashi, Koichi, 17
Takeda, Masayuki, 112, 141
Takimoto, Eiji, 155
Tanaka, Takehiko, 252
Tazaki, Eiichiro, 273
Thomas, Bernd, 50
Toriwaki, Jun-ichiro, 311
Tsuchiya, Takashi, 197
Tsukimoto, Hiroshi, 212
Uehara, Kuniaki, 285, 290

Ueki, Kazuya, 155
Ueno, Genta, 197
Ullman, Jeffrey D., 1
Washio, Takashi, 99, 304
Watanabe, Tatsuya, 155

Yada, Katsutoshi, 263
Yugami, Nobuhiro, 278
Zak, Premysl, 261