Computational Intelligence in Data Mining

Proceedings of a workshop held in conjunction with the Fifth IEEE International Conference on Data Mining

© Jose-María Serrano

Houston, USA, November 27th, 2005

Edited by

Fernando Berzal
Universidad de Granada (Spain)

Juan Carlos Cubero
Universidad de Granada (Spain)

Zbigniew W. Ras
University of North Carolina (USA)

Thomas Sudkamp
Wright State University (USA)

Ronald Yager
Iona College (USA)

IEEE COMPUTER SOCIETY

ISBN 0-9738918-5-8

ICDM'2005 Workshop on

Computational Intelligence in Data Mining

Proceedings of a workshop held in conjunction with the Fifth IEEE International Conference on Data Mining

Houston, Texas, November 27th, 2005

Table of contents

Finding Simple Disjoint Decompositions in Frequent Itemset Data Using Zero-suppressed BDDs 3
Shin-ichi Minato

Training of Kernel Fuzzy Classifiers by Dynamic Cluster Generation 13
Shigeo Abe

Generating Fuzzy Queries from Weighted Fuzzy Classifier Rules 21
Antonio C. S. Branco, Alexandre G. Evsukoff & Nelson F. F. Ebecken

Fuzzy Clustering in Parallel Universes with Noise Detection 29
Bernd Wiswedel & Michael R. Berthold

Accuracy-Complexity Tradeoff Analysis by Multiobjective Rule Selection 39
Hisao Ishibuchi & Yusuke Nojima

Intelligent Computation for Association Rule Mining 49
Hong-Cheu Liu & John Zeleznikow

Building A Security-Aware Query Answering System Based On Hierarchical Data Masking 55
Seunghyun Im, Zbigniew W. Ras & Agnieszka Dardzinska

Mining of Cell Assay Images Using Active Semi-Supervised Clustering 63
Nicolas Cebron & Michael R. Berthold

An Adaptive Intrusion Detection Systems using a Data Mining Approach 71
Sujaa Rani Mohan, E.K. Park & Yijie Han

Structured Multinomial Models of Traffic Flow 79
Juan K. Lin

ICDM'2005 Workshop on

Computational Intelligence in Data Mining

Organizing Committee

Fernando Berzal, *University of Granada (Spain)*
Juan-Carlos Cubero, *University of Granada (Spain)*
Zbigniew W. Ras, *University of North Carolina (USA)*
Thomas Sudkamp, *Wright State University (USA)*
Ronald R. Yager, *Iona College (USA)*

Program Committee

Michael R. Berthold. *University of Konstanz (Germany)*
José-Ramón Cano. *University of Jaén (Spain)*
I-Jen Chiang. *Taipei Medical School (Taiwan)*
Agnieszka Dardzinska. *Bialystok Technical University (Poland)*
Miguel Delgado. *University of Granada (Spain)*
María Ángeles Gil Álvarez. *University of Oviedo (Spain)*
Francisco Herrera. *University of Granada (Spain)*
Tzung-Pei Hong. *National University of Kaohsiung (Taiwan)*
Eyke Hüllermeier. *University of Marburg (Germany)*
Hisao Ishibuchi. *Osaka Prefecture University (Japan)*
Frank Klawonn. *University of Braunschweig (Germany)*
Peter Kokol. *University of Maribor (Slovenia)*
Vipin Kumar. *University of Minnesota (USA)*
T.Y. Lin. *San Jose State University (USA)*
Eric Louie. *IBM Almaden Research Center (USA)*
Dragos Margineantu. *Boeing (USA)*
Nicolás Marín. *University of Granada (Spain)*
Dunja Mladenic. *J Stefan Institute (Slovenia)*
Witold Pedrycz. *University of Alberta (Canada)*
James Peters. *University of Manitoba (Canada)*
Lorenza Saitta. *Università del Piemonte Orientale (Italy)*
Daniel Sánchez. *University of Granada (Spain)*
José-María Serrano. *University of Jaén (Spain)*
Andrzej Skowron. *Warsaw University (Poland)*
Dominik Slezak. *University of Regina (Canada)*
Maarten van Someren. *University of Amsterdam (Holland)*
Shusaku Tsumoto. *Shimane Medical University (Japan)*
Amparo Vila. *University of Granada (Spain)*
Yiyu Yao. *University of Regina (Canada)*
Xingquan Zhu. *University of Vermont (USA)*
Wojciech Ziarko. *University of Regina (Canada)*

Finding Simple Disjoint Decompositions in Frequent Itemset Data Using Zero-suppressed BDDs

Shin-ichi Minato
Hokkaido University
minato@ist.hokudai.ac.jp

Abstract

In this paper, we propose a method of finding simple disjoint decompositions in frequent itemset data. The techniques for decomposing Boolean functions have been studied for long time in the area of logic circuit design, and recently, there is a very efficient algorithm to find all possible simple disjoint decompositions for a given Boolean functions based on BDDs (Binary Decision Diagrams). We consider the data model called "sets of combinations" instead of Boolean functions, and present a similar efficient algorithm for finding all possible simple disjoint decompositions for a given set of combinations. Our method will be useful for extracting interesting hidden structures from the frequent itemset data on a transaction database. We show some experimental results for conventional benchmark data.

1 Introduction

Manipulation of large-scale combinatorial data is one of the fundamental technique for data mining process. In particular, frequent item set analysis is important in many tasks that try to find interesting patterns from web documents and databases, such as association rules, correlations, sequences, episodes, classifiers, and clusters. Since the introduction by Agrawal et al.[1], the frequent item set and association rule analysis have been received much attentions from many researchers, and a number of papers have been published about the new algorithms or improvements for solving such mining problems[6, 8, 22].

After generating frequent itemset data, we sometimes faced with the problem that the frequent itemsets are too large and complicated to retrieve useful information. So, it is an important technique for extracting some hidden structures from the frequent itemsets to make the data more understandable. Closed/maximal itemset mining[23, 20, 21] is one of the useful method in this approach.

In this paper, we propose a new method of finding "simple disjoint decompositions" in the frequent itemset data. Our method extracts another aspect of hidden structures from complicated itemset data, and will be useful for database analysis.

Our method is based on the Boolean function decomposition technique, which is a fundamental theory of logic circuit design. Simple disjoint decomposition is a basic and useful concept in this theory. This decomposition gives a single-output sub-block function whose input variable set is disjoint from the other part. It is a special case of decompositions and not always possible for all Boolean functions. If we find a such decomposition for a given function, it must be a good choice for optimal design, and we may proceed to the local optimization of each sub-block. There are so many studies on the method of finding simple disjoint decompositions, and currently, the method[4][10][11] based on the recursive algorithm using BDDs (Binary Decision Diagrams) is remarkably fast and powerful to find all possible simple disjoint decompositions for a given Boolean functions.

In this paper, we focus on the data model called "sets of combinations", instead of Boolean functions. A set of combinations consists of the elements each of which is a combination of multiple items. This data model often appears in many kind of combinatorial problems, and of course, it can be used for representing frequent itemset data. A set of combinations can be mapped into Boolean space of n input variables, and efficiently manipulated by using BDDs. In addition, we can enjoy more efficient manipulation using "Zero-suppressed BDDs" (ZBDD)[12], which are special type of BDDs optimized for handling sets of combinations.

As a major contribution of this paper, we present that we can define the operation of simple disjoint decomposition for sets of combinations, as well as for Boolean functions. We then show that all possible simple disjoint decompositions on sets of combinations can be extracted in a method based on ZBDDs, as well as the conventional BDD-based functional decomposition method. Our new ZBDD-based decomposition algorithm for sets of combinations is not completely same as the BDD-based one for Boolean functions, but they have similar recursive structures and complexities. We also show the experimental results of finding

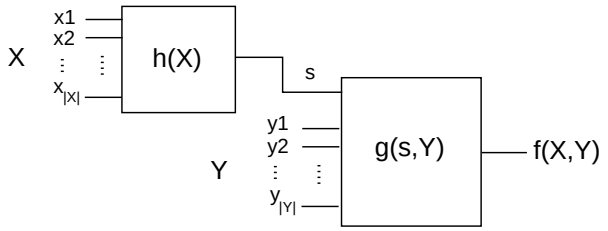


Figure 1. Simple disjoint decomposition on Boolean function.

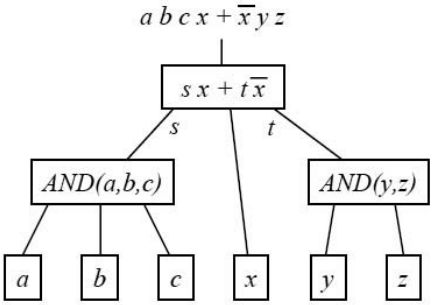


Figure 2. Tree structure of simple disjoint decompositions.

all simple disjoint decompositions in the frequent itemsets extracted from a conventional benchmark dataset.

The paper is organized as follows: First, we review the decomposition methods for Boolean functions in Section 2. In Section 3, we define the decomposition for sets of combinations and present the algorithm for finding simple disjoint decompositions. We then show the experimental results in section 4, followed by conclusion.

2 Boolean Function Decomposition

At first, we review the methods of Boolean function decomposition. If the function f can be represented as $f(X,Y) = g(h(X),Y)$, then f can be realized by the network shown in Fig. 1. We call it *simple disjoint decomposition.* It is called "simple" because h is a single-output function and "disjoint" because X and Y have no common variables. We call a trivial decomposition if X consists of only one variable. A non-trivial simple disjoint decomposition does not always exist in a given function, but if exists, it is considerably effective for logic optimization.

A function may have more than one simple disjoint decompositions. They can be nesting. For example, the function $abcx + \overline{x}yz$ has five decompositions as $X = \{a,b\}$, $\{b,c\}$, $\{a,c\}$, $\{a,b,c\}$, and$\{y,z\}$.

Multiple input logic operations (AND, OR, EXOR) may produce a number of associative sub-decompositions. In such cases, we handle those decompositions as one group, and only use the full-merged form to represent the group. On the above example, we only use $\{a,b,c\}$ to represent the group including three associative sub-decompositions $\{a,b\}$, $\{b,c\}$, $\{a,c\}$. After merging such associative ones, two simple disjoint decompositions never overlap each other. The structure of simple disjoint decompositions can be expressed by a tree graph as shown in Fig. 2. Since each input variable appears only once as a leaf of tree, the number of branching nodes never exceeds the number of input variables. The problem of finding simple disjoint decompositions is to construct such a tree structure for a given Boolean function.

There are many studies on the methods of finding simple disjoint decompositions. At first, a classical method with a *decomposition chart* is presented[3][17]. In recent years, more efficient way using a BDD-based implicit decomposition chart is discussed[9][18][19]. One proposed another approach[13] to extract all simple disjoint decompositions based on factoring of sum-of-products expressions.

In the long history of studies on Boolean function decomposition, the BDD-based recursive algorithm, which is proposed by Bertacco et al. on 1997 and improved later by Matsunaga[10][11], is now overwhelmingly effective to extract all simple disjoint decompositions for a given Boolean functions. This algorithms is based on the following two properties:

- If we consider NPN-equivalence and associativity, the tree structure of simple disjoint decompositions is canonical for a given Boolean functions.

- Basically, all simple disjoint decompositions for f can also be found in the two cofactor functions $f_{(x=0)}$ and $f_{(x=1)}$.

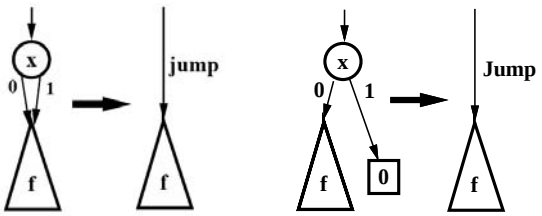
Using the two properties, the algorithm expands a given Boolean function to the two cofactor functions and call itself recursively. The final results of tree structure is obtained by checking common part of the results for the two cofactor functions. Since the algorithm has a cache mechanism to avoid duplicated traversals for the same BDD nodes, we can efficiently execute the procedure in a time bounded by the BDD size (roughly square for BDD nodes). Actually, we need only a few seconds to extract all possible simple disjoint decompositions for a benchmark function with ten thousands of BDD nodes and dozens of input variables. This decomposition method is effectively used for VLSI logic synthesis and technology mapping[16].

a	b	c	F	
0	0	0	0	
0	0	1	1	→ c
0	1	0	0	
0	1	1	1	→ bc
1	0	0	0	
1	0	1	0	
1	1	0	1	→ ab
1	1	1	1	→ abc

As a Boolean function:
$F = a\,b + \overline{a}\,c$

As a set of combinations:
$F = \{ab, abc, bc, c\}$

Figure 3. Correspondence of Boolean functions and Sets of combinations.



(a) Ordinary BDD. (b) ZBDD.

Figure 4. Different reduction rules for BDD and ZBDDs.

3 Decomposition for Sets of Combinations

In this section, we discuss simple disjoint decomposition for sets of combinations, and show a method of finding those decompositions.

3.1 Sets of combinations and Boolean functions

A set of combinations consists of the elements each of which is a combination of a number of items. There are 2^n combinations chosen from n items, so we have 2^{2^n} variations of sets of combinations. For example, for a domain of five items a, b, c, d, and e, we can show examples of sets of combinations as:
$\{ab, e\}$, $\{abc, cde, bd, acde, e\}$, $\{1, cd\}$, $\emptyset$. Here "1" denotes a combination of null items, and "$\emptyset$" means an empty set. Sets of combinations are one of the basic data structure for handling combinatorial problems. They often appear in real-life problems, such as combinations of switching devices, sets of faults, paths in the networks, etc., and of course, it can be used for representing frequent itemset data.

A set of combinations can be mapped into Boolean space of n input variables. For example, Fig. 3 shows a truth table of Boolean function $(ab + \overline{a}c)$, but also represents a set of combinations $\{ab, abc, bc, c\}$. Such Boolean functions are called *characteristic functions* for the sets of combinations. Using BDD manipulation for characteristic functions, we can implicitly represent and manipulate large-scale sets of combinations. In addition, we can enjoy more efficient manipulation using "Zero-suppressed BDDs" (ZBDD)[12], which are special type of BDDs optimized for handling sets of combinations.

ZBDDs are based on the reduction rule different from one used in ordinary BDDs. As illustrated in Fig. 4(a), the ordinary reduction rule deletes the nodes whose two edges point to the same node. However, in ZBDDs, we do not delete such nodes but delete another type of nodes whose 1-edge directly points to the 0-terminal node, as shown in Fig. 4(b).

In ZBDDs, a 0-edge points to the subset (cofactor) of combinations not including the decision variable x, and a 1-edges points to the subset (cofactor) of combinations including x. If the 1-edge directly points to the 0-terminal node, it means that the item x never appear in the set of combinations. Zero-suppressed reduction rule automatically deletes such a node with respect to the irrelevant item x, and thus ZBDDs are more compactly represent sets of combinations than using ordinary BDDs.

The detailed techniques of ZBDD manipulation are described in the articles[12][14]. A typical ZBDD package supports cofactoring operations to traverse 0-edge or 1-edge, and binary operations between two sets of combinations, such as union, intersection, and difference. the computation time for each operation is almost linear to the number of ZBDD nodes related to the operation.

3.2 Simple disjoint decompositions on sets of combinations

In this paper, we propose the definition of "simple disjoint decomposition on sets of combinations", as a similar concept as one for Boolean functions. As shown in Fig. 1 again, if a given sets of combination f can be decomposed as $f(X, Y) = g(h(X), Y)$, and X and Y has no common items, we then call it a simple disjoint decomposition. Here we explain the meaning of substitution operation on the set of combinations. At first, we consider the set of combination $g(s, Y)$. Let us extract all combinations including s (a cofactor of g w.r.t s), and then replace s with any one combination in $h(X)$. The result of substitution $g(h(X), Y)$ is the set of all possible combinations obtained by the replacements. Notice that the combinations irrelevant to s are left as is. If the substitution result exactly equals to $f(X, Y)$, it is a decomposition on f.

For instance, we consider the two sets of combinations:

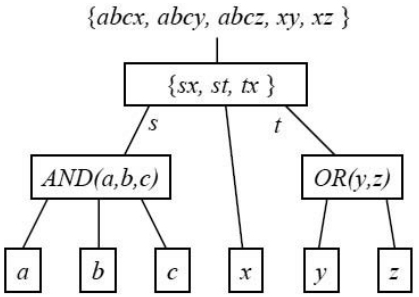


Figure 5. Tree structure of decompositions on sets of combinations.

$g(c,d,e) = \{cd, cde, de, e\}$ and $h(a,b) = \{a, ab\}$. When we substitute $h(a,b)$ for c, then

$g(h(a,b),d,e) = \{ad, abd, ade, abde, de, e\}$.

Namely, $f(a,b,d,e) = \{ad, abd, ade, abde, de, e\}$ has a simple disjoint decomposition as $g(h(a,b),d,e)$.

A set of combinations has one-to-one mapping to a Boolean function, however, the simple disjoint decompositions for the two data models do not have such direct relation. Although simple disjoint decomposition on sets of combinations is a similar concept as one for Boolean functions, we have to develop another decomposition algorithm considering particular properties on sets of combinations.

Similarly to the case for Boolean functions, a set of combinations may have more than one simple disjoint decompositions. For example, the set of combinations $\{abcx, abcy, abcz, xy, xz\}$ has five decompositions as $X = \{a,b\}$, $\{b,c\}$, $\{a,c\}$, $\{a,b,c\}$, and $\{y,z\}$. They may have a nested structure, and can be represented by a tree graph as shown in Fig. 5.

Contrary to the case for Boolean functions, sets of combinations do not have symmetric NOT operation (uses difference operation instead of NOT), there are no equivalences on input/output polarity. Only input permutation should be considered. In Matsunaga's decomposition algorithm[11] on Boolean functions uses only OR and NOT instead of AND operation to keep canonical forms, however, in the case for sets of combinations, AND (product) operation and OR (union) operation do not have dual relation, so we should treat them individually. We do not have to consider XOR operation on sets of combinations. Finally we found two kind of binary operations: AND (product) and OR (union), which may produce associative sub-decompositions on sets of combinations. As well as for Boolean functions, we handle those sub-decompositions as one group, and only use the full-merged form to represent the group.

We need one more special consideration when the set of combinations includes "1" (the element of null combination). For example, let us consider $f(a,b,c) = \{ab, c, 1\}$. If we choose $h = \{ab\}$, we can find simple disjoint decomposition as $g(h,c) = \{h, c, 1\}$. However, we may choose $h = \{ab, 1\}$ and then $g(h,c) = \{h, c\}$ or $g(h,c) = \{h, c, 1\}$ are possible. Consequently, the decomposition structure may not be unique when the set of combinations includes "1". In such cases, we need additional rule to keep the decomposition trees canonical. Basically, we may put a restriction that $h(X)$ does not include "1", namely, only the parent set $g(s,Y)$ can have "1". This rule is working well except for AND (product) operation. Let us consider $f = \{abc, ac\}$. It can be decomposed as $\{ab, b\} \times \{c\}$, $\{a\} \times \{bc, c\}$, $\{ac\} \times \{b, 1\}$, and $\{a\} \times \{b, 1\} \times \{c\}$. This example shows that AND (product) operation has the associative property no matter how "1" is included or not, so we must decompose by a set with "1" if it is possible. Fortunately, we can see that if f is AND-decomposable by h with "1", then f is not decomposable by h without "1". On the other hand, if f is AND-decomposable by h without "1", then f is not decomposable by h with "1". This means that we can keep the decomposition tree unique.

3.3 ZBDD-based algorithm for finding simple disjoint decompositions

In the following discussion, we use "$\cdot$" for AND (product) operator, and "+" for OR (union) operator.

If a given sets of combinations $F(X,Y)$ contains a simple disjoint decomposition with $P(X)$, it can be written as:

$F(X,Y) = P(X) \cdot Q(Y) + R(Y)$.

Here we choose an item v used in F, and compute the two cofactors F_0, F_1. (Namely, $F = v \cdot F_1 + F_0$.)
Since v must be included in either of X or Y, the following two cases are considered:

- In case of $v \in X$: (Let $X' = X - v$.)
  $F(X,Y) = P(X) \cdot Q(Y) + R(Y)$
  $= \{v \cdot P_1(X') + P_0(X')\} \cdot Q(Y) + R(Y)$
  $= v \cdot \{P_1(X') \cdot Q(Y)\} + \{P_0(X') \cdot Q(Y) + R(Y)\}$
  Thus,
  $F_1(X',Y) = P_1(X') \cdot Q(Y)$,
  $F_0(X',Y) = P_0(X') \cdot Q(Y) + R(Y)$.

- In case of $v \in Y$: (Let $Y' = Y - v$.)
  $F(X,Y) = P(X) \cdot Q(Y) + R(Y)$
  $= P(X) \cdot \{v \cdot Q_1(X') + Q_0(X')\}$
  $+\{v \cdot R_1(X') + R_0(X')\}$
  $= v \cdot \{P(X) \cdot Q_1(Y') + R_1(Y')\}$
  $+\{P(X) \cdot Q_0(Y') + R_0(Y')\}$
  Thus,
  $F_1(X,Y') = P(X) \cdot Q_1(Y') + R_1(Y')$,
  $F_0(X,Y') = P(X) \cdot Q_0(Y') + R_0(Y')$.

In any case, if a given sets of combination F has simple disjoint decompositions, they can be found by checking the common set of decompositions on the cofactors F_0 and

```
Decomp(F)
{
  if (F = 0) return 0 ;
  if (F = 1) return 1 ;
  H ← Cache(F) ;
  if (H exists) return H ;
  v ← F.top ; /* Top item in F */
  (F_0, F_1) ← (Cofactors of F w.r.t v) ;
  H_0 ←Decomp(F_0) ;
  H_1 ←Decomp(F_1) ;
  H ←Merge(v, H_0, H_1) ;
     /* Make H from common part of H_0, H_1 */
  Cache(F) ← H ;
  return H ;
}
```

Figure 6. Sketch of the algorithm.

F_1. Similarly to BDD-based recursive method of Boolean function decomposition, there are some cases that a part of factors, such as $P_0(X')$ and $Q_0(Y')$, may be reduced to a constant "1" or "$\emptyset$", so we need more detailed classification in actual implementation of the algorithm.

The above discussion shows that we can generate a tree structure of representing all possible simple disjoint decompositions on a set of combinations using a ZBDD-based recursive algorithm, which is a similar manner to the BDD-based method[4][10][11] for Boolean function decomposition. The sketch of algorithm is shown in Fig.6. The computation time can be roughly square to the ZBDD size as well as the previous method, by using a cache mechanism to avoid duplicated traversals for the same ZBDD nodes.

4 Experimental Results

We implemented a program for finding all simple disjoint decompositions on sets of combinations. The program is based on our own ZBDD package, and additional 1,600 lines of C++ code for the decomposition algorithm. We used a Pentium-4 PC, 800MHz, 512MB of main memory, with SuSE Linux 9. We can manipulate up to 10,000,000 nodes of ZBDDs in this PC.

For evaluating the performance, we conducted an experiment for finding all simple disjoint decompositions in th frequent itemset data extracted from two example of benchmark database[7]. The specification of the two databases are shown in Table 1. In this table, the column "#Items" shows the number of items used in the database, "#Records" is the number of records, "Total literals" means the total number of each record size. (Record size is the number of items appearing in the record.) "Literals/Records" shows average number of items appearing in a record.

In our experiment, at first we generate a ZBDD representing the histogram of all patterns seen in the database, and then we extract a set of frequent patterns that appear more than or equal to α times in the database. We conducted this frequent itemset mining procedure for various threshold α. In this process, we need about 300 second for "mushroom", and 6 second for the first 1,000 lines of "T10I4D100K", respectively. The detailed techniques in the ZBDD-based frequent itemset mining method are described in a recent articles[15].

The experimental results are summarized in Table 2. In the table, the column "Min-freq.(α)" shows the minimum frequency parameter α. "#Items" means the number of items related to the frequent itemset. "#Patterns" is the number of patterns in the frequent itemsets. "ZBDD nodes" shows the number of ZBDD nodes representing the frequent itemset data, which is the input of the decomposition algorithm. "Time(sec)" shows the CPU time for generating the decomposition tree representing all simple disjoint decompositions. (not including the time for generating initial ZBDDs.) "#Decomp." is the number of non-trivial decompositions extracted by our method. Notice that all sub-decompositions caused by associative operations are counted only once for one group, so, actual number of decompositions may exist more.

As shown in the table, our decomposition method is very powerful to deal with a huge number of frequent patterns. We succeeded in finding simple disjoint decompositions in the frequent itemset data in a feasible time. It is another interesting point that the decomposition results are different depending on threshold α.

In Fig. 7, we show an example of decomposition result for "mushroom" with threshold $\alpha = 5,000$. The upper description lists all frequent patterns of itemsets, and the lower is the decomposition result. In this format, "`AND( )`" and "`OR( )`" show the associative decomposition groups, and "`[ ]`" means another general decomposition. "`!`" is a special character to mean "+1". For example, "`AND(!x !y)`" indicates $(x+1)(y+1)$. Even for this small example, we can see that it is hard for human beings to find the hidden structures from the plain list. By using our decomposition method, the data becomes remarkably understandable.

In Fig. 8 and Fig. 9, we show the decomposition results for "mushroom" and "T10I4D100K" with various threshold α. We can observe interesting structures hidden in the frequent itemset data. Notice that the frequent itemset data handled here are too large to manipulate explicit manner. ZBDD-based implicit manipulation is a key technique

5 Conclusion

In this paper, we proposed the definition of simple disjoint decomposition for sets of combinations, and presented an efficient ZBDD-based method for finding all possible simple disjoint decompositions on a set of combinations. The experimental results shows that our method is very powerful and useful for extracting hidden interesting structures from the frequent itemset data.

Table 1. Spec. of databases

Data name	#Items	#Records	Total Literals	Literals/Records
mushroom	119	8,124	186,852	23.0
T10I4D100K (first 1000 lines)	795	1,000	10,098	10.1

Table 2. Experimental result

Data name	Min-freq.(α)	#Items	#Patterns	ZBDD nodes	Time(sec)	#Decomp.
mushroom	5,000	7	42	11	(<0.1)	5
	2,000	35	6,624	286	(<0.1)	4
	1,000	54	123,278	1,417	0.1	4
	500	67	1,442,504	4,011	0.2	5
	200	83	18,094,822	12,340	0.5	6
	100	90	66,076,586	23,068	0.8	7
	50	98	198,169,866	36,652	2.0	7
	20	113	781,458,546	53,776	4.0	8
	10	115	1,662,769,668	61,240	4.9	9
	5	117	2,844,545,896	62,389	11.7	10
	2	119	5,043,140,094	51,217	7.1	10
	1	119	5,574,930,438	40,557	5.0	10
T10I4D100K	70	1	2	1	(<0.1)	1
(first 1000 lines)	60	4	5	4	(<0.1)	1
	50	12	13	12	(<0.1)	1
	30	74	75	74	(<0.1)	1
	20	171	173	172	0.1	2
	10	378	506	430	0.8	22
	5	585	3,891	1,322	1.5	42
	2	745	30,893	9,903	13.3	13
	1	795	24,467,220	70,847	1,698.0	8

List of all frequent patterns:

```
x39 x86 x85 x34, x39 x86 x85, x39 x86 x34, x39 x86, x39 x85 x34, x39 x85, x39 x34, x39,
x90 x86 x85 x36 x34, x90 x86 x85 x36, x90 x86 x85 x34, x90 x86 x85, x90 x86 x36 x34,
x90 x86 x36, x90 x86 x34, x90 x86, x90 x85 x36 x34, x90 x85 x36, x90 x85 x34, x90 x85,
x90 x36 x34, x90 x36, x90 x34, x90, x86 x85 x36 x34, x86 x85 x36, x86 x85 x34, x86 x85,
x86 x36 x34, x86 x36, x86 x34, x86, x85 x59, x85 x36 x34, x85 x36, x85 x34, x85, x59,
x36 x34, x36, x34, 1
```

Decomposition result:

```
AND(OR(AND(OR(x39 AND(!x90 !x36)) !x86 !x34) x59) !x85)
```

Figure 7. Decomposition result for "mushroom" with $\alpha = 5,000$.

Now we have just finished implementation of decomposition algorithms, and starting experiments for data mining applications. The concept of simple disjoint decomposition will be a meaningful operation in database processing, and we hope that our result has an impact to the data mining community.

Acknowledgment

The author thanks Prof. Arimura of Hokkaido Univ. for his technical comments. This study is partly supported by Grant-in-Aid Scientific Research on Priority Area "Informatics", 2004 (Area #006) and Scientific Research (B), 2005, 17300041.

References

[1] R. Agrawal, T. Imielinski, and A. N. Swami, Mining Association rules between sets of items in large databases, In P. Buneman and S. Jajodia, edtors, *Proc. of the 1993 ACM SIGMOD International Conference on Management of Data*, Vol. 22(2) of SIGMOD Record, pp. 207–216, ACM Press, 1993.

[2] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen and A. I. Verkamo, Fast Discovery of Association Rules, In *Advances in Knowledge Discovery and Data Mining*, MIT Press, 307–328, 1996.

[3] R. L. Ashenhurst, "The decomposition of switching functions," Proc. of an International Symposium on the Theory of Switching, pp. 74–116, 1957.

[4] V. Bertacco and M. Damiani, "The disjunctive decomposition of logic functions," In Proc. of International Conference on Computer-Aided Design Design (ICCAD-97), pp. 78–82, Nov.1997.

[5] Bryant, R. E., Graph-based algorithms for Boolean function manipulation, IEEE Trans. Comput., C-35, 8 (1986), 677–691.

[6] B. Goethals, "Survey on Frequent Pattern Mining", Manuscript, 2003. `http://www.cs.helsinki.fi/u/goethals/publications/survey.ps`

[7] B. Goethals, M. Javeed Zaki (Eds.), Frequent Itemset Mining Dataset Repository, Frequent Itemset Mining Implementations (FIMI'03), 2003. `http://fimi.cs.helsinki.fi/data/`

[8] J. Han, J. Pei, Y. Yin, R. Mao, Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach, Data Mining and Knowledge Discovery, 8(1), 53–87, 2004.

[9] Y.-T. Lai, M. Pedram and S. B. K. Vrudhula, "BDD based decomposition of logic functions with application to FPGA synthesis," In Proc. of 30th ACM/IEEE Design Automation Conf. (DAC-93), pp. 642–647, 1993.

[10] Y. Matsunaga, "An exact and efficient algorithm for disjunctive decomposition," In Proc. of the Workshop on Synthesis And System Integration of Mixed Technologies (SASIMI-98), pp. 44–50, 1998.

[11] Y. Matsunaga, "An efficient algorithm finding simple disjoint decompositions using BDDs," IEICE Trans. on fund., Vol. E-85-A, No. 12, pp. 2715–2724, 2002.

[12] Minato, S., Zero-suppressed BDDs for set manipulation in combinatorial problems, In Proc. 30th ACM/IEEE Design Automation Conf. (DAC-93), (1993), 272–277.

[13] S. Minato and G. De Micheli, "Finding All Simple Disjunctive Decompositions Using Irredundant Sum-of-Products Forms," In Proc. of ACM/IEEE International Conference on Computer-Aided Design (ICCAD-98), pp. 111–117, Nov. 1998.

[14] Minato, S., Zero-suppressed BDDs and Their Applications, International Journal on Software Tools for Technology Transfer (STTT), Springer, Vol. 3, No. 2, pp. 156–170, May 2001.

[15] S. Minato and H. Arimura: "Efficient Combinatorial Item Set Analysis Based on Zero-Suppressed BDDs", IEEE/IEICE/IPSJ International Workshop on Challenges in Web Information Retrieval and Integration (WIRI-2005), pp. 3–10, Apr., 2005.

[16] A. Mishchenko, X. Wang, and T. Kam, "A New Enhanced Constructive Decomposition and Mapping Algorithm," In Proc. of 40th ACM/IEEE Design Automation Conf. (DAC-2003), pp. 143–148, 2003.

[17] J. Roth and R. Karp, "Minimization Over Boolean Graphs," IBM Journal, pp. 227-238, Apr. 1962.

[18] T. Sasao, "FPGA Design by Generalized Functional Decomposition," Logic Synthesis and Optimization, Kluwer Academic Publishers, pp. 233–258, 1993.

[19] H. Sawada and T. Suyama and A. Nagoya, "Logic Synthesis for Look-Up Table based FPGAs Using Functional Decomposition and Boolean Resubstitution," IEICE Trans. Information and Systems, Vol. E80-D, No. 10, pp. 1017-1023, Oct. 1997.

[20] T. Uno, T. Asai, Y. Uchida, H. Arimura, "LCM: An Efficient Algorithm for Enumerating Frequent Closed Item Sets," In Proc. IEEE ICDM'03 Workshop FIMI'03, 2003. (Available as CEUR Workshop Proc. series, Vol. 90, http://ceur-ws.org/vol-90)

[21] T. Uno, T. Asai, Y. Uchida and H. Arimura, "An Efficient Algorithm for Enumerating Closed Patterns in Transaction Databases," In Proc. of the 8th International Conference on Discovery Science 2004 (DS-2004), 2004.

[22] M. J. Zaki, Scalable Algorithms for Association Mining, IEEE Trans. Knowl. Data Eng. 12(2), 372–390, 2000.

[23] M. J.Zaki, C. Hsiao, "CHARM: An Efficient Algorithm for Closed Itemset Mining," 2nd SIAM International Conference on Data Mining (SDM'02), pp. 457-473, 2002.

$\alpha = 2,000$:

```
AND(![x102 x58 OR(x66 x61) x29 x116 x56 x6 x11 x110 x94 x53 x28 x24 x10 x114 x39 x2 x93
x90 AND(!x86 !x34) x76 x67 x63 x59 x52 x38 x36 x23 x13 x9 x3 x1] !x85)
```

$\alpha = 1,000$:

```
AND(![x48 x102 x58 AND(!x101 !x95) x66 x61 x29 x17 OR(x69 x77) x117 x116 x56 x6 x111 x11
x44 x110 x43 x94 x53 x37 x28 x24 x16 x10 x41 x15 x114 x99 x39 x14 x2 x107 x98 x93 x90 x86
x76 x67 x63 x59 x54 x52 x38 x36 x34 x23 x13 x9 x3 x1] !x85)
```

$\alpha = 500$:

```
AND(![OR(x32 x7 x31) x119 x48 x102 x91 x58 x80 x101 x95 x66 x61 x29 x17 OR(x78 x68) x69
x77 x45 OR(x60 x64) x117 x116 x56 x6 x111 x11 x44 x110 x43 x42 x94 x53 x37 x28 x24 x16
x10 x41 x15 x114 x99 x55 x39 x14 x2 x107 x98 x93 x90 x86 x76 x67 x63 x59 x54 x52 x38 x36
x34 x23 x13 x9 x3 x1] !x85)
```

$\alpha = 200$:

```
AND(![x35 OR(x32 x31) x7 x119 x48 x112 x102 x91 x58 OR(x80 x71 x79 x70) x101 x95 x66 x61
x29 x17 x46 OR(x78 x68) x69 x77 x45 x60 x117 x65 x116 x56 x6 x111 x64 x11 x44 x110 x43
x42 x109 x94 x53 x37 x28 x24 x16 x10 x115 x41 OR(x27 x26) x15 x4 x114 x108 x99 x55 x39
x14 x2 x113 x107 x98 x93 x90 x86 x76 x67 x63 x59 x54 x52 x40 x38 x36 x34 x25 x23 x13 x9
x3 x1] !x85)
```

Figure 8. Decomposition results for "mushroom" with various α.

$\alpha = 60$:

```
!OR(x354 x177 x217 x368)
```

$\alpha = 50$:

```
!OR(x438 x460 x829 x684 x354 x177 x217 x529 x766 x283 x120 x368)
```

$\alpha = 20$:

```
!OR(x893 x634 x661 x826 x510 x886 x38 x440 x75 x438 x998 x752 x812 x57 x694 x692 x982
x413 x8 x349 x912 x906 x162 x275 x55 x470 x598 x793 x989 x33 x995 x509 x797 x69 x516
x593 x236 x93 x140 x112 x48 x348 x239 x32 x21 x132 x10 x72 x871 x832 x580 x168 x154
x573 x472 x31 x362 x494 x489 x196 x919 x744 x28 x600 x460 x829 x605 x477 AND(!x346
!x217) x684 x354 x296 x12 x617 x522 x885 x676 x145 x956 x758 x935 x780 x638 x631 x487
x541 x116 x789 x788 x526 x403 x918 x844 x722 x419 x310 x73 x151 x874 x480 x204 x70 x43
x944 x888 x614 x523 x803 x778 x579 x411 x147 x921 x78 x27 x862 x392 x960 x795 x623 x571
x177 x175 x161 x878 x653 x276 x183 x914 x720 x675 x597 x280 x279 x192 x71 x390 x970
x947 x809 x782 x682 x658 x529 x350 x798 x569 x966 x883 x766 x738 x381 x283 x229 x937
x895 x449 x854 x674 x581 x401 x205 x120 x39 x825 x775 x561 x538 x368 x274)
```

$\alpha = 10$:

```
OR(x207 x820 x732 x769 x550 x428 x450 x258 x173 x922 x893 x949 x405 x351 x215 x634 [x661
x394 x510 AND(!x515 !x33) x346 x780 x487 AND(!x888 !x561) x217 x720 x71 x766 x283] x308
x948 x815 x838 x707 x826 x804 x309 x887 x318 x860 x68 x241 x129 x843 x429 x886 [x819 x75
x438 AND(!x598 !x782) x460 x829 x684 x789 x70 x529 x937 x368] x468 x686 x265 x784 x252 x38
x440 x486 x108 x322 x361 [x357 AND(!x752 !x58) AND(!x158 !x617) x583 x354 x480 x27] x563
x170 x867 x710 x595 x899 x998 x991 x852 x160 x923 x812 x800 x57 x567 x694 x692 x984 x982
x94 x413 x197 x8 x349 x963 x665 x546 x406 x373 x117 AND(!x912 !x348) x906 x711 AND(!x534
!x470 !x995) x378 x162 x45 x259 x275 x897 x95 x55 x37 x427 x793 x97 x989 x336 x527 x343
x983 x611 [x509 x862 x801 x461 x392 x569] x110 x577 x913 x797 x415 x69 x6 AND(!x516 !x744)
x423 x1 x122 x593 x90 x952 x236 x606 x93 x594 x387 x285 x140 x112 x521 x765 x639 x319 x126
x48 x500 x100 x239 x136 x54 x32 x21 x464 x172 x132 x10 x981 x72 x988 x871 x832 x580 x168
x154 x111 x651 x628 x573 x472 x329 x181 x31 x591 x362 x673 x641 x494 x489 x196 x919 x736
AND(!x517 !x883) x115 x5 x742 x28 AND(!OR(x709 x177 x970) !x310) x600 x746 x649 x355 x234
x884 x605 x477 x210 x841 x740 x548 x296 x12 x522 x885 x792 x790 x731 x676 x385 x145
AND(!x956 !x788) x763 x758 x242 x17 x935 x735 x638 x631 x471 x946 x805 x701 x541 x171 x395
x201 x198 x116 x975 x774 x526 x403 x326 x967 x918 x846 x844 x810 x722 x484 x469 x419 x118
x73 x890 x830 x504 x151 x874 x334 x204 x43 x944 x614 x523 x290 x266 x903 x842 x803 x778
x579 x572 x411 x147 x921 x78 x839 x130 x125 x960 x910 AND(OR(x795 AND(!x623 !x853)) !x571)
x490 x424 x175 x161 AND(!x878 !x538) x706 x653 x277 x276 x256 x193 x183 x932 x914 x675 x618
x597 x530 x496 x280 x279 x272 x192 x390 x227 x947 x809 x682 x658 x350 x214 x798 x620 x143
x104 x978 x966 x738 x708 x381 x352 x294 x229 x964 x895 x857 x449 x422 x950 x854 x733 x674 x35
x814 x704 x581 x401 x205 x120 AND(!x39 !x825) x834 x775 x687 x448 x274 x240 x164 x52 x25)
```

Figure 9. Decomposition results for "T10I4D100K" (first 1000 lines) with various α.

Training of Kernel Fuzzy Classifiers by Dynamic Cluster Generation

Shigeo Abe
Graduate School of Science and Technology
Kobe University
Nada, Kobe, Japan
abe@eedept.kobe-u.ac.jp

Abstract

We discuss kernel fuzzy classifiers with hypersphere regions, which are defined in the feature space mapped from the input space. Starting from a hypersphere with a small radius defined at a datum, we expand the hypersphere if the new datum is within the prescribed distance from the center. And if not, we define a new hypersphere. After rule generation, we resolve overlaps of different classes contracting the hyperspheres. We then define a truncated conical membership function for each hypersphere. We demonstrate the usefulness of the kernel version of fuzzy classifiers with hypersphere regions with several benchmark data sets.

1 Introduction

In support vector machines (SVMs) [1], the input space is mapped into a high dimensional feature space, and in the space, the optimal hyperplane is determined so that two classes are separated with the maximum margin. According to performance comparisons in a wide range of applications, support vector machines have shown to have high generalization ability.

Inspired by the success of support vector machines, to improve generalization ability and classification ability, conventional pattern classification techniques are extended to incorporate maximizing margins and mapping to a feature space. For example, online perceptron algorithms, neural networks, and fuzzy systems have incorporated maximizing margins [2].

There are numerous conventional techniques that are extended to be used in the high-dimensional feature space, e.g., kernel perceptrons, the k-means clustering algorithm, the kernel self organizing feature map, kernel discriminant analysis, kernel principal component analysis, kernel Mahalanobis distance, and kernel least-squares [2].

One of the problems of support vector machines is that it is difficult to analyze the behavior of support vector machines because the input space is mapped to the high-dimensional feature space. There are several approaches to visualize support vector machines [3, 4, 5]. Another approach is to extend fuzzy classifiers to kernel fuzzy classifiers that are defined in the feature space [6]. Although fuzzy classifiers with hyperbox regions such as discussed in [7, 8] are difficult to extend to the feature space, classifiers with hyperspheres [9, 10, 11] are relatively easily extended.

In this paper, we define fuzzy rules in the feature space in the way similar to fuzzy min-max classifiers [7]. Instead of generating and contracting hyperboxes, we generate and contract hyperspheres in the feature space. To facilitate efficient calculations in the feature space, we use kernel tricks. Namely, we use kernel functions associated with a mapping function to avoid explicit treatment of variables in the feature space.

Suppose there are training data belonging to one of n classes. We scan the training data and for a training datum with no associated hyperspheres we define the hypersphere at the training datum with a predefined small radius R_ε. If there is a hypersphere that includes the training datum or whose center is within the prescribed maximum radius $R_{\max}$, we modify the center and the radius of the hypersphere. If not, we generate the hypersphere at the training datum with radius R_ε. After generating the hyperspheres, we check whether the hyperspheres of different classes overlap. If there is an overlap, we resolve the overlap contracting the hyperspheres.

In Section 2, we discuss the rule generation of kernel fuzzy classifiers. In Section 3, we compare the generalization performance of the method with that of other classifiers using two-class and multiclass data sets.

2 Dynamic Rule Generation

2.1 Concept

In the first stage we scan the training data and generate hyperspheres of each class without resolving overlaps be-

tween classes. Then in the second stage, we resolve overlaps between classes, contracting hyperspheres.

We explain the idea of hypersphere generation using the two-dimensional example shown in Fig. 1. In the figure, assume that Data 1, 2, and 3 belonging to the same class are scanned in this order. For Datum 1, we generate the circle with Datum 1 being the center and with radius R_ε. Then for Datum 2, we check if the circle is expandable. Assume that distance between Data 1 and 2 is shorter than $R_{\max}$. Then we generate the dotted circle shown in Figure 1, whose center is at the middle of data 1 and 2 and whose diameter is the distance between Data 1 and 2.

If Datum 3 is inside of the dotted circle, we update the center and the radius. But because Datum 3 is outside of the circle, we check if the distance between the center and the datum is within $R_{\max}$. If so, we update the center adding Datum 3 in addition to Datum 1 and 2 and calculate the minimum radius that includes Data 1, 2, and 3. If the circle is not expandable, we generate the circle with Datum 3 being the center and with radius R_ε.

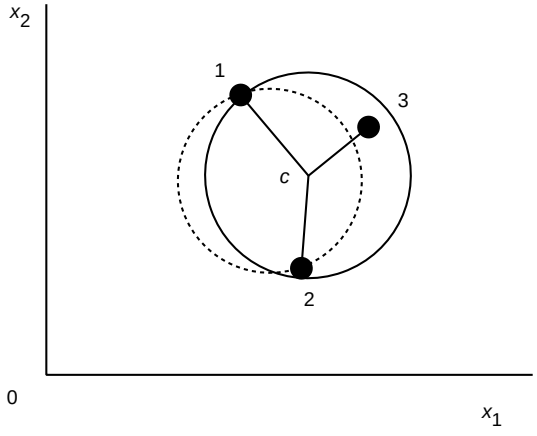


Figure 1. Generation of circles

After generating hyperspheres for each class without considering the overlap between classes, we resolve overlaps by contracting hyperspheres. In Fig. 2, two circles belonging to classes i and j are overlapped. We resolve this overlap, contracting the circles as shown in the figure.

For each hypersphere we define a membership function for datum $\mathbf{x}$, in which the degree of membership is 1 if $\mathbf{x}$ is in the sphere and the degree of membership decreases as $\mathbf{x}$ moves away from the hypersphere. Fig. 3 shows an example for a two dimensional case. The shape of the membership function is a truncated conical. Usually, the degree of membership is defined between 0 and 1, but to avoid unclassifiable regions, we assume negative degree of membership.

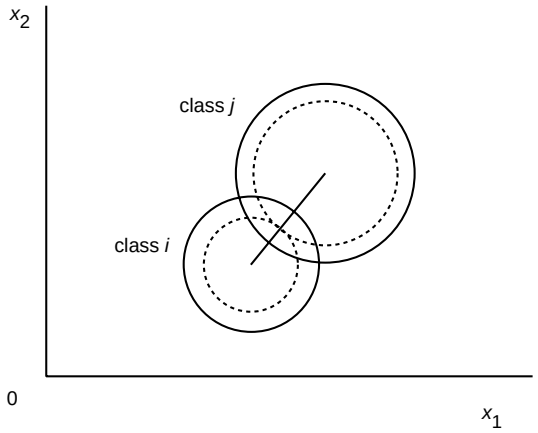


Figure 2. Resolution of overlap

2.2 Rule Generation

In the following we discuss the procedure of rule generation in detail.

Let the training inputs for class i be $\mathbf{x}_{ij}$ for $j = 1, \ldots, M_i$, where $\mathbf{x}_{ij}$ is the jth training datum for class i and M_i is the number of data for class i. And let $\mathbf{g}(\mathbf{x})$ be the mapping function that maps the input space into the feature space.

The procedure for generating hyperspheres for class i is as follows.

1. Generate the hypersphere S_{i1} with center $\mathbf{c}_{i1} = \mathbf{g}(\mathbf{x}_{i1})$ and with radius $R_{i1} = R_\varepsilon$. Set $N_i^{\mathrm{s}} = 1$, $X_{i1} = \{1\}$, and $j = 2$, where N_i^{s} is the number of generated hyperspheres for class i and X_{i1} is the set of indices of data for calculating the center $\mathbf{c}_{i1}$.

2. Check if $\mathbf{x}_{ij}$ is in a sphere. Namely if there exists such k ($1 \le k \le N_i^{\mathrm{s}}$) that satisfies

$$\|\mathbf{g}(\mathbf{x}_{ij}) - \mathbf{c}_{ik}\| \le R_{ik}, \tag{1}$$

$\mathbf{x}_{ij}$ is in hypersphere S_{ik}, where

$$\mathbf{c}_{ik} = \frac{1}{|X_{ik}|} \sum_{j' \in X_{ik}} \mathbf{g}(\mathbf{x}_{ij'}). \tag{2}$$

Here, $|X_{ik}|$ is the number of elements in X_{ik}. If there are more than one hypersphere that satisfy (1), we select one whose value on the left-hand side of (1) is the smallest. Update the center given by (2) adding $\mathbf{x}_{ij}$, update R_{ik}, and go to Step 4. Otherwise, find hypersphere S_{ik} whose center is nearest to $\mathbf{x}_{ij}$:

$$k = \arg\min_{k'} \|\mathbf{g}(\mathbf{x}_{ij}) - \mathbf{c}_{ik'}\|. \tag{3}$$

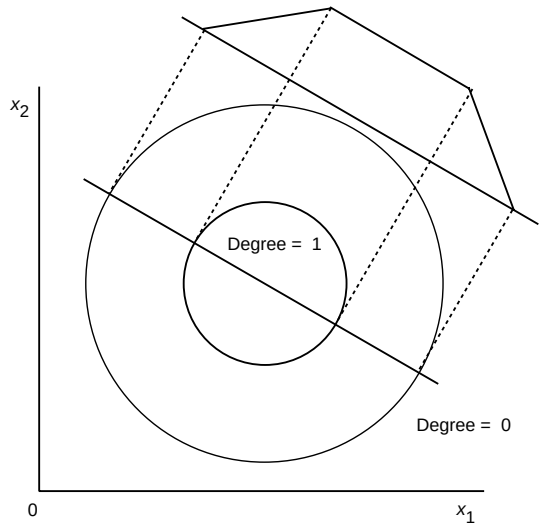


Figure 3. Definition of a membership function

3. Check if the hypersphere S_{ik} can be expandable for the inclusion of $\mathbf{x}_{ij}$. If

$$\|\mathbf{g}(\mathbf{x}_{ij'}) - \mathbf{c}_{ik}\| \leq R_{\max} \quad \text{for } j \in X_{ik} \tag{4}$$

is satisfied, hypersphere S_{ik} is expandable.

Then, we set

$$X_{ik} \leftarrow X_{ik} \cup \{\mathbf{x}_{ij}\}, \tag{5}$$
$$R_{ik} = \max_{j' \in X_{ik}} \|\mathbf{x}_{ij'} - \mathbf{c}_{ij}\|. \tag{6}$$

Otherwise, we set

$$N_i^{\mathrm{s}} \leftarrow N_i^{\mathrm{s}} + 1, \tag{7}$$
$$X_{i,N_i^{\mathrm{s}}} = \{j\}, \tag{8}$$

and generate the hypersphere $S_{i,N_i^{\mathrm{s}}}$ with center $\mathbf{c}_{i,N_i^{\mathrm{s}}} = \mathbf{g}(\mathbf{x}_{ij})$ and with radius $R_{i,N_i^{\mathrm{s}}} = R_{\varepsilon}$.

4. If $j = M_i$, terminate the algorithm. Otherwise, $j \leftarrow j + 1$ and go to Step 2.

2.3 Overlap Resolution

We can resolve overlaps by contracting hyperspheres or tuning membership functions, which are defined in the next section. But since we are going to define truncated conical membership functions shown in Fig. 3, we need to resolve overlaps between hyperspheres.

The ratio of contraction directly influences the generalization ability. But here, we resolve overlaps of hyperspheres S_{ij} and S_{ko} $(i \neq k)$ in the following way:

1. For $R_{ij} \leq \|\mathbf{c}_{ij} - \mathbf{c}_{ko}\| < R_{ij} + R_{ko}$ and $R_{ko} \leq \|\mathbf{c}_{ij} - \mathbf{c}_{ko}\| < R_{ij} + R_{ko}$

   As shown in Fig. 2, two hyperspheres overlap but each center is outside of the other hypersphere. We resolve the overlap contracting the hyperspheres by

$$\Delta r = \frac{R_{ij} + R_{ko} - \|\mathbf{c}_{ij} - \mathbf{c}_{ko}\|}{2}, \tag{9}$$
$$R_{ij} \leftarrow R_{ij} - \Delta r, \tag{10}$$
$$R_{ko} \leftarrow R_{ko} - \Delta r. \tag{11}$$

2. For $R_{ij} < \|\mathbf{c}_{ij} - \mathbf{c}_{ko}\| \leq R_{ko}$

   One of the hypersphere centers is inside of the other hypersphere but the other is not. We resolve the overlap contracting the hyperspheres by

$$R_{ko} \leftarrow \|\mathbf{c}_{ij} - \mathbf{c}_{ko}\| - \frac{R_{ij}}{2}, \tag{12}$$
$$R_{ij} \leftarrow \frac{R_{ij}}{2}. \tag{13}$$

3. For $\|\mathbf{c}_{ij} - \mathbf{c}_{ko}\| < R_{ij} \leq R_{ko}$

   Both centers of the hyperspheres are inside of the other hyperspheres. We resolve the overlap contracting the hyperspheres by

$$R_{ij} \leftarrow \frac{\|\mathbf{c}_{ij} - \mathbf{c}_{ko}\|}{2}, \tag{14}$$
$$R_{ko} \leftarrow \frac{\|\mathbf{c}_{ij} - \mathbf{c}_{ko}\|}{2}. \tag{15}$$

Contraction of hyperspheres may result in misclassification of the correctly classified training data, and thus may worsen the generalization ability. This is avoided by tuning membership functions. But here, we do not consider tuning. Instead, we optimize the value of $R_{\max}$ by cross validation.

2.4 Definition of Membership Functions

We define the membership function of S_{ij} for $\mathbf{x}$, $m_{ij}(\mathbf{x})$, by

$$m_{ij}(\mathbf{x}) = \begin{cases} 1 & \text{for} \quad d_{ij}(\mathbf{x}) \leq R_{ij}, \\ 1 - d_{ij}(\mathbf{x}) + R_{ij} & \text{for} \quad d_{ij}(\mathbf{x}) > R_{ij}, \end{cases} \tag{16}$$

where $d_{ij}(\mathbf{x})$ is the distance between $\mathbf{g}(\mathbf{x})$ and $\mathbf{c}_{ij}$ and is given by

$$d_{ij}(\mathbf{x}) = \|\mathbf{c}_{ij} - \mathbf{g}(\mathbf{x})\|. \tag{17}$$

The membership function given by (16) takes negative value so that any data are classified into a definite class unless they are on the class boundaries. Since the slopes of the membership functions are the same for all the hyperspheres, the class boundary on the line segment between $\mathbf{c}_{ij}$ and $\mathbf{c}_{ko}$

is the middle point of the line segment between the two hyperspheres S_{ij} and S_{ko}. This is a similar idea of the optimal separating hyperplane of a support vector machine. But if there are misclassified data belonging to class i or j, we can tune the slopes of membership functions [12]. But here we do not consider this.

2.5 Kernel Methods

In kernel methods, we treat the variables in the feature space implicitly using kernels. In our proposed method, we need to calculate $\|\mathbf{c}_{ij} - \mathbf{c}_{ko}\|$ and $\|\mathbf{g}(\mathbf{x}) - \mathbf{c}_{ij}\|$ using $H(\mathbf{x}, \mathbf{x}') = \mathbf{g}^T(\mathbf{x})\,\mathbf{g}(\mathbf{x})$. Namely,

$$\begin{aligned}\|\mathbf{c}_{ij} - \mathbf{c}_{ko}\|^2 &= (\mathbf{c}_{ij} - \mathbf{c}_{ko})^T(\mathbf{c}_{ij} - \mathbf{c}_{ko}) \\ &= \frac{1}{|X_{ij}|^2}\sum_{\mathbf{x},\mathbf{x}'\in X_{ij}} H(\mathbf{x}, \mathbf{x}') \\ &\quad -\frac{2}{|X_{ij}|\,|X_{ko}|}\sum_{\mathbf{x}\in X_{ij},\mathbf{x}'\in X_{ko}} H(\mathbf{x}, \mathbf{x}') \\ &\quad +\frac{1}{|X_{ko}|^2}\sum_{\mathbf{x},\mathbf{x}'\in X_{ko}} H(\mathbf{x}, \mathbf{x}'), \qquad (18)\end{aligned}$$

$$\begin{aligned}\|\mathbf{g}(\mathbf{x}) - \mathbf{c}_{ij}\|^2 &= H(\mathbf{x}, \mathbf{x}) - \frac{2}{|X_{ij}|}\sum_{\mathbf{x}'\in X_{ij}} H(\mathbf{x}, \mathbf{x}') \\ &\quad +\frac{1}{|X_{ij}|^2}\sum_{\mathbf{x}',\mathbf{x}''\in X_{ij}} H(\mathbf{x}', \mathbf{x}''). \qquad (19)\end{aligned}$$

In our study, we use the following kernels:

1. linear kernels: $H(\mathbf{x}, \mathbf{x}') = \mathbf{x}^t\mathbf{x}'$,
2. polynomial kernels:
$$H(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^t\mathbf{x}' + 1)^d,$$
where d is a positive integer,
3. RBF kernels:
$$H(\mathbf{x}, \mathbf{x}') = \exp(-\gamma||\mathbf{x} - \mathbf{x}'||^2), \qquad (20)$$
where γ is a positive parameter.
4. Mahalanobis kernels:
$$H(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\delta}{m}(\mathbf{x} - \mathbf{x}')^T Q^{-1}(\mathbf{x} - \mathbf{x}')\right),$$
where δ is a positive parameter and Q is the diagonal covariance matrix calculated using the training data [13].

Because, except for linear kernels,

$$\sum_i \mathbf{g}(\mathbf{x}_i) \neq \mathbf{g}(\sum_i \mathbf{x}_i) \qquad (21)$$

holds, we need to calculate the second term in the right hand side of (19) for $\mathbf{x}$ but the result of the third term may be saved to reduce computation time. Likewise, by saving the result of (18), computation time is reduced.

3 Performance Evaluation

We compared the generalization ability of the kernel fuzzy classifier and other methods using two groups of data sets: (1) some of the two-class data sets used in [14, 15][1] and (2) multiclass data sets used in [2, 12]. We used the linear and Mahalanobis kernels for the kernel fuzzy classifier.

Throughout the experiments we set $R_\varepsilon = 0.01$ and selected the value of $R_{\max}$ from 0.05 to 1.5 by the increment of 0.05 by 10-fold cross validation. In addition to this we set a large value to $R_{\max}$ to avoid clustering the class data. For Mahalanobis kernels we need to determine $R_{\max}$ and δ. Like Mahalanobis kernels for support vector machines [13], we determine the values of $R_{\max}$ and δ by line search. Namely, first we determine the value of $R_{\max}$ with $\delta = 1$ by cross validation. Then with the determined value of $R_{\max}$, we determine the value of δ by cross validation changing δ from 0.1 to 2 with the increment of 0.1.

3.1 Evaluation for Two-class Problems

Table 1 lists the specifications of two-class problems. Each problem has 100 or 20 training data sets and their corresponding test data sets. We determined the optimal values of $R_{\max}$ by 10-fold cross validation. For comparison, we used the support vector machine (SVM) with Mahalanobis kernels and the kernel fuzzy classifier with ellipsoidal regions [6]. For the support vector machine, we determined the optimum value of the margin parameter C and δ by line search. For the kernel fuzzy classifier with ellipsoidal regions (KFC-ER), we used RBF kernels and determined the parameters by 5-fold cross validation.

We performed cross validation of the first five training data sets, and selected the median of the best values. Then, for the optimal values, we trained the classifier for 100 or 20 training data sets and calculated the average recognition error and the standard deviation for the test data sets.

Table 2 lists the parameter values for the kernel fuzzy classifiers determined by cross validation. The symbol ∞ denotes that the highest recognition rate for the validation data set was obtained when the class data were not clustered. Except for the banana and image data sets relatively large values were selected for $R_{\max}$.

Table 3 lists the average classification errors and the standard deviations with the $\pm$ symbol. The "KFC-L" and

[1] http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm

Table 3. Average error rates and standard deviations of the test data sets.

Data	KFC-L	KFC	KFC-ER	SVM
Banana	12.6 $\pm$0.9	12.4 $\pm$0.8	10.9 $\pm$0.6	**10.4**$\pm$**0.5**
B. Cancer	31.6 $\pm$5.3	31.0 $\pm$4.6	26.5 $\pm$4.4	**25.6**$\pm$4.4
Diabetes	33.2 $\pm$5.9	28.7 $\pm$2.5	25.3$\pm$2.0	**23.7**$\pm$**1.7**
German	26.5 $\pm$2.4	27.4 $\pm$2.2	25.2$\pm$2.4	**23.9**$\pm$**2.1**
Heart	18.7 $\pm$3.5	16.2 $\pm$3.3	**15.6**$\pm$3.6	15.7 $\pm$3.2
Image	**2.8**$\pm$0.7	3.7 $\pm$0.5	2.9$\pm$0.7	3.0$\pm$0.5
Ringnorm	24.4$\pm$0.6	**1.4**$\pm$0.1	3.2 $\pm$0.3	1.7$\pm$0.1

Table 1. Benchmark data sets for two-class problems.

Data	Inputs	Train.	Test	Sets
Banana	2	400	4900	100
B. cancer	9	200	77	100
Diabetes	8	468	300	100
German	20	700	300	100
Heart	13	170	100	100
Image	18	1300	1010	20
Ringnorm	20	400	7000	100

Table 2. Parameter values selected by cross validation.

Data	Linear	Mahalanobis	
	$R_{\max}$	$R_{\max}$	δ
Banana	0.1	0.35	0.7
B. Cancer	0.8	0.9	1.0
Diabetes	0.65	0.6	0.5
German	0.8	0.5	0.5
Heart	∞	∞	0.1
Image	0.05	0.1	1.4
Ringnorm	∞	∞	1.2

"KFC" columns list the values for the kernel fuzzy classifier with linear kernels and that with Mahalanobis kernels, respectively.

The best performance in the row is shown in boldface. For the german and image data sets, the KFC-L performed better than the KFC, but for the other four data sets, the KFC performed better.

The KFC-L showed the best performance for the image data set, but for the ringnorm data set, the KFC-L showed a large error compared to other methods. Table 4 shows the best recognition rates of the ringnorm validation data sets. The recognition rates of the KFC-L are very low compared to those of the KFC. Because smaller $R_{\max}$ results in overfitting, the large value of $R_{\max}$ was selected as optimal. And the large value caused large contraction of hyperspheres. Thus, in such a situation we need to optimize the membership functions as discussed in [12]. Except for the heart, image, and ringnorm data sets, KFC or KFC-L performed poorly compared to KFC-ER and SVM. For these data sets to improve the generalization ability we need to optimize the membership functions.

3.2 Evaluation for Multiclass Problems

As multiclass data sets, we used the data sets in [2, 12]. They were the iris data [16, 17], the numeral data for license plate recognition [18], the thyroid data [19],[2] the blood cell data [20], and hiragana data [12, 21]. Table 5 lists the specifications of the data sets.

We used pairwise support vector machines. To resolve unclassifiable regions, we used fuzzy support vector machines with minimum operators [2].

Table 6 shows the parameter values determined by cross validation. Unlike two-class problems, a relatively small value was set to $R_{\max}$.

[2] ftp://ftp.ics.uci.edu/pub/machine-learning-databases/

Table 4. Cross validation results (%) for the ringnorm data set.

Data	KFC-L		KFC		
	$R_{\max}$	Rec.	$R_{\max}$	δ	Rec.
1	∞	75.75	∞	1.0	99.00
2	∞	76.75	∞	1.6	98.75
3	∞	74.00	∞	1.2	98.75
4	1.2	77.50	∞	1.1	99.00
5	∞	75.00	∞	1.4	98.75

Table 5. Benchmark data sets for multi-class problems.

Data	Inputs	Classes	Train.	Test
Iris	4	3	75	75
Numeral	12	10	810	820
Thyroid	21	3	3772	3428
Blood cell	13	12	3097	3100
Hiragana-50	50	39	4610	4610
Hiragana-105	105	38	8375	8356
Hiragana-13	13	38	8375	8356

Table 6. Parameter values selected by cross validation.

Data	Linear	Mahalanobis	
	$R_{\max}$	$R_{\max}$	δ
Iris	0.3	0.35	0.6
Numeral	0.05	0.65	0.9
Thyroid	0.1	0.25	0.6
Blood cell	0.1	0.05	0.2
Hiragana-50	0.05	0.15	1.0
Hiragana-13	0.05	0.05	0.3
Hiragana-105	0.05	0.10	0.5

Table 7 lists the recognition rates of the test data sets for 6 classifiers. The results of the fuzzy mini-max classifier (FMM) and the k-nearest neighbor classifier (k-NN) are from [12]. They showed the best performance for the test data. For k-NN, the best k was selected from 1, 3, 5, and 7. But for other classifiers, the parameters were determined by cross validation. The SVM used Mahalanobis kernels and the margin parameter and δ were determined by cross validation. For the KFC-ER, polynomial kernels were used and the polynomial degree was determined by 5-fold cross validation.

The performance difference between the KFC-L and KFC is small. The reason may be that because the multi-class data sets are relatively easily classified compared to the two-class problems, the improvement of separability by mapping to the feature space resulted in overfitting. For the thyroid data, the KFC-L and KFC show poor recognition rates. This is the same tendency with that of the least-squares support vector machines [2]. But for other data sets, performance is comparable.

4 Conclusions

In this paper we discussed kernel fuzzy classifiers with hypersphere regions. We scan the training data and if no hyperspheres are defined for the class associated with a datum we define the hypersphere at the training datum with a predefined small radius. If there is a hypersphere that can include the training data within the prescribed maximum radius, we expand the hypersphere. If not, we generate the hypersphere at the training data with the small radius. After generating the hyperspheres, we resolve the overlap contracting the hyperspheres.

For some two-class data sets the proposed classifiers showed performance inferior to support vector machines and kernel fuzzy classifiers with ellipsoidal regions, but for multiclass problems, their performance was comparable to other methods.

To further improve the generalization ability, we need to tune the membership functions.

References

[1] V. N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, New York, 1998.

[2] S. Abe. *Support Vector Machines for Pattern Classification*. Springer-Verlag, London, 2005.

[3] D. Caragea, D. Cook, and V. Honavar. Towards simple, easy-to-understand, yet accurate classifiers. In *Proceedings of the Third IEEE International Conference on Data Mining (ICDM 2003)*, pages 497–500, Melbourne, FL, 2003.

Table 7. Recognition rates of the test data sets.

Data	KFC-L	KFC	FMM	KFC-ER	k-NN	SVM
Iris	**97.33**	96.00	**97.33**	**97.33**	94.67	96.00
Numeral	99.27	99.02	**99.76**	99.51	99.51	**99.76**
Thyroid	92.82	93.76	**99.42**	96.82	93.67	98.02
Blood cell	90.84	90.90	90.45	**93.16**	90.13	92.74
Hiragana-50	96.75	96.98	94.64	97.61	97.16	**98.52**
Hiragana-13	99.52	99.58	98.22	**99.86**	99.55	99.83
Hiragana-105	99.99	99.99	99.44	**100**	99.99	**100**

[4] X. Fu, C.-J. Ong, S. Keerthi, G. G. Hung, and L. Goh. Extracting the knowledge embedded in support vector machines. In *Proceedings of International Joint Conference on Neural Networks (IJCNN 2004)*, volume 1, pages 291–296, Budapest, Hungary, 2004.

[5] H. Núñez, C. Angulo, and Català. Rule extraction from support vector machines. In *Proceedings of the Tenth European Symposium on Artificial Neural Networks (ESANN 2002)*, pages 107–112, Bruges, Belgium, 2002.

[6] K. Kaieda and S. Abe. KPCA-based training of a kernel fuzzy classifier with ellipsoidal regions. *International Journal of Approximate Reasoning*, 37(3):145–253, 2004.

[7] P. K. Simpson. Fuzzy min-max neural networks—Part 1: Classification. *IEEE Transactions on Neural Networks*, 3(5):776–786, 1992.

[8] S. Abe and M.-S. Lan. A method for fuzzy rules extraction directly from numerical data and its application to pattern classification. *IEEE Transactions on Fuzzy Systems*, 3(1):18–28, 1995.

[9] C.-W. V. Chen and T.-Z. Liu. A competitive learning process for hyperspherical classifiers. *Neurocomputing*, 17:99–110, 1997.

[10] P. M. Patil, U. V. Kulkarni, and T. R. Sontakke. Modular fuzzy hypersphere neural network. In *Proceedings of the Twelfth IEEE International Conference on Fuzzy Systems*, volume 1, pages 232–236, 2003.

[11] G. C. Anagnostopoulos and M. Georgiopoulos. Hypersphere ART and ARTMAP for unsupervised and supervised, incremental learning. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN 2000)*, volume 6, pages 59–64, Como, Italy.

[12] S. Abe. *Pattern Classification: Neuro-Fuzzy Methods and Their Comparison*. Springer-Verlag, London, 2001.

[13] S. Abe. Training of support vector machines with mahalanobis kernels. In W. Duch, J. Kacprzyk, E. Oja, and S. Zadrozny, editors, *Artificial Neural Networks: Formal Models and Their Applications (ICANN 2005)—Proceedings of International Conference, Warsaw, Poland*, pages 571–576. Springer-Verlag, Berlin, Germany, 2005.

[14] G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for AdaBoost. *Machine Learning*, 42(3):287–320, 2001.

[15] K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–201, 2001.

[16] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

[17] J. C. Bezdek, J. M. Keller, R. Krishnapuram, L. I. Kuncheva, and N. R. Pal. Will the real iris data please stand up? *IEEE Transactions on Fuzzy Systems*, 7(3):368–369, 1999.

[18] H. Takenaga, S. Abe, M. Takatoo, M. Kayama, T. Kitamura, and Y. Okuyama. Input layer optimization of neural networks by sensitivity analysis and its application to recognition of numerals. *Electrical Engineering in Japan*, 111(4):130–138, 1991.

[19] S. M. Weiss and I. Kapouleas. An empirical comparison of pattern recognition, neural nets, and machine learning classification methods. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 781–787, Detroit, 1989.

[20] A. Hashizume, J. Motoike, and R. Yabe. Fully automated blood cell differential system and its application. In *Proceedings of the IUPAC Third International Congress on Automation and New Technology in the Clinical Laboratory*, pages 297–302, Kobe, Japan, 1988.

[21] M.-S. Lan, H. Takenaga, and S. Abe. Character recognition using fuzzy rules extracted from data. In *Proceedings of the Third IEEE International Conference on Fuzzy Systems*, volume 1, pages 415–420, Orlando, 1994.

Generating Fuzzy Queries from Weighted Fuzzy Classifier Rules

Antonio C. S. Branco
COPPE/Federal University of Rio de Janeiro, Brazil
saraivabranco@gmail.com

Alexandre G. Evsukoff
COPPE/Federal University of Rio de Janeiro, Brazil
evsukoff@coc.ufrj.br

Nelson F. F. Ebecken
COPPE/Federal University of Rio de Janeiro, Brazil
nelson@ntt.ufrj.br

Abstract

This work presents a methodology to generate automatically a set of fuzzy queries that are translated from a set of fuzzy rules learned from a data set. A rule simplification approach is proposed, which allows a better interpretability of the fuzzy query sentence. The fuzzy query sentences are generated in such a way that they can be processed by a standard fuzzy query system. The methodology is applied to a benchmark data set considering a scenario of target selection in a direct market application. The results obtained by the current approach are compared with the results of a standard a decision tree induction algorithm.

1. Introduction

The integration of Data Mining (DM) tools with Data Base Management Systems (DBMS) is now more than trend, it is a reality. The major DBMS vendors have already integrated DM solutions within their products. On the other hand, the main DM suites have also provided the integration of DM models into DBMS trough modeling languages such as the Predictive Model Markup Language (PMML). It is thus a fact that future research on new DM tools and methods must consider their integration with DBMS.

On the application side, data intensive industries, such as insurance, accounting and telecommunications, among others, need frequently to retrieve their costumers for different marketing relationship actions. Nevertheless, defining a query sentence that captures the main features of a subset of records is a non trivial task. The queries generation can thus be regarded as a DM task.

Structured Query Languages (SQL) provides a structured description of the retrieved records. A standard SQL sentence representing a concept can be translated from rules generated by an algorithm for decision-tree or rule induction. The translation of machine learning rules into SQL sentences is straightforward: for each class, each rule corresponds to a sentence (in the WHERE clause), and all rules related to the same class are aggregated with a disjunctive (OR) operator. A standard query, however, will return all the records that match the query sentence. However, it is frequently desirable to rank those records, such that the manager is able to define priorities.

Neural networks and Bayesian classifiers are also frequently found in most of DM suites. Such models may be coded into DBMS, via PMML, to retrieve a ranked list of records. Nevertheless, neural networks and Bayesian classifiers models are not linguistically understandable, such that managers cannot directly validate the knowledge extracted by the DM algorithm.

Fuzzy queries have emerged in the last 25 years to deal with the necessity to soften the Boolean logic in relational databases. A fuzzy query system is an interface to human users to get information from database using (quasi) natural language sentences [1][2][3]. Many fuzzy queries implementations have been proposed, resulting in slightly different languages such as SQLf [4], FQUERY [5] and Summary SQL [6][7], among others. Although some variations according to the particularities of different implementations, the answer to a fuzzy query sentence is a generally a list of records, ranked by the degree of matching.

The focus of attention of the fuzzy query research has been on the expressive power of the fuzzy query language, since the query is usually provided by the user. Nevertheless, the definition of a query to retrieve the records according to their features' values is a very difficult task. Not due to the lack of expressive power of the query language, but due to the difficulty on defining the concept itself.

Fuzzy queries sentences are structured definitions of fuzzy concepts. Under this assumption, fuzzy queries can be automatically generated by fuzzy rule based classifiers.

Fuzzy rule based classifiers are a very active research field [8][9][10]. The weighted fuzzy rule based approach allows better classification results since it allows to define more precisely the decision boundary among the classes [11][12]. The translation of weighted fuzzy rules into fuzzy queries is not straightforward since most of fuzzy queries languages do not support weights.

In this work, it is proposed a methodology to translate a set of rules computed by a weighted fuzzy classifier into fuzzy queries, such that they can be coded into most of available fuzzy query languages. The whole procedure is sketched in Figure 1. The left side of the figure shows an existing fuzzy query system (denoted Fuzzy SQL) used to

process the fuzzy queries connected to a DBMS. The proposed methodology is showed in the right side of the figure. A training set is selected from the database and used to generate a set of fuzzy rules by the fuzzy classifier. The set of rules generated by the fuzzy classifier may be very large, containing many useless rules. The fuzzy rule base is thus pruned and then translated into a set of fuzzy query sentences. The fuzzy query system is used to retrieve an ordered set of records from the database, corresponding to a given class.

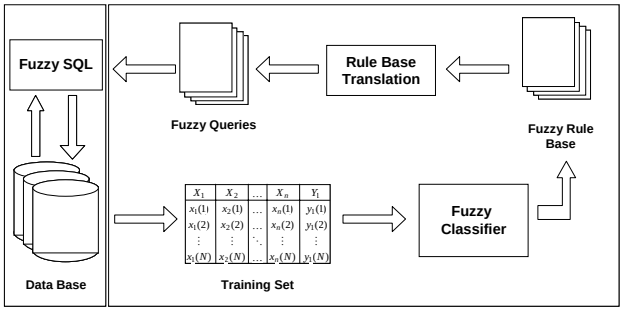


Figure 1. The proposed approach.

The remainder of this paper is organized as follows. Next section presents the weighted fuzzy rule-based classifier and the translation of fuzzy rules into fuzzy queries. Section 3 describes the learning algorithm of fuzzy rules and the pruning procedure. Section 4 presents the results of the application of the proposed methodology for a set of benchmark data sets and a case study using the Credit Card data set. The fuzzy queries results are compared to standard queries generated by a standard decision-tree induction algorithm. Finally some conclusions are drawn.

2. Fuzzy rule-based classifier

Consider the standard classification problem where input variables are presented as a *p*-dimensional vector **x** in the input variable domain $X_1 \times X_2 \times \ldots \times X_p = X^p$ and the output variables represented by the classes' set $\mathbf{C} = \{C_1, \ldots, C_m\}$. The solution to the classification problem is to assign a class label $C_k \in \mathbf{C}$ to an observation $\mathbf{x}(t) \in X^p$, where t represents a record in the database.

Fuzzy classifiers attempt to compute, for each class, a fuzzy membership value $\mu_{C_k}(\mathbf{x}(t))$ that correspond to the degree of matching of the observation $\mathbf{x}(t)$ to the class C_k.

This section describes the weighted fuzzy rule-based classifier approach, which is based on the fuzzy pattern matching approach [8]. Under this approach, the output of the fuzzy classifiers is computed in two steps:

1. for each input, compute partial outputs as the degree of matching of the observed input value to each class,
2. compute the final output by the aggregation of all partial outputs.

The following subsections describe the main steps of the fuzzy rule-based classifier approach.

2.1. Fuzzification

In a general application, the input variables may be numeric (discrete or continuous) or nominal. Fuzzy sets allow a unified representation for nominal and numeric variables as fuzzy sets. Fuzzification is thus an important issue in fuzzy query generation since it provides the numeric-to-linguistic interface that allows dealing with numeric values as linguistic terms.

Generally, each input variable x_i can be described using ordered linguistic terms in a *descriptor set* $\mathbf{A}_i = \{A_{i1}, \ldots, A_{in_i}\}$. When the variable is nominal, the descriptor set is the set of possible values for the variable (or a combination of them). When the variable is numeric, the *meaning* of each term $A_{ij} \in \mathbf{A}_i$ is given by a fuzzy set defined on the variable domain.

For a single variable input $x_i(t)$, the fuzzification vector $\mathbf{u}_i(t) = (u_{i1}(t), \ldots, u_{in_i}(t))$ is computed by the fuzzy sets in the fuzzy partition of the input variable domain as:

$$\mathbf{u}_i(t) = \left(\mu_{A_{i1}}(x_i(t)), \ldots, \mu_{A_{in_i}}(x_i(t))\right) . \quad (1)$$

An easy way to parameterize fuzzy sets is to use triangular membership functions that are completely determined by the centers of triangles, which may be considered as prototypes values for the corresponding fuzzy sets (see Figure 2).

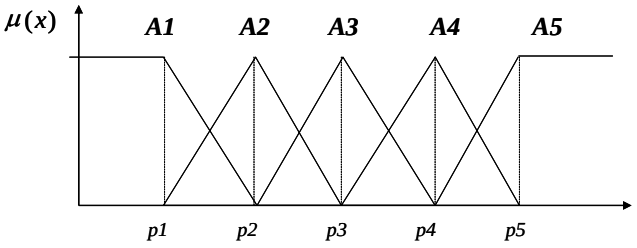


Figure 2: Example of fuzzy partition.

The fuzzification vector generalizes the information contained in the input variable and is computed in the same way if the variable is numeric or nominal. For nominal variables, there is no fuzziness and the fuzzification vector is a binary vector, indicating which nominal value is present on the record.

In a general-purpose fuzzy classifier, multidimensional fuzzification may also be considered and computed by fuzzy clustering methods. Nevertheless, multidimensional

fuzzy sets often do not represent linguistic concepts and are not supported by some fuzzy query languages. Multidimensional fuzzy sets are thus out of the scope of this paper.

2.2. Fuzzy rules

A fuzzy rule relates input linguistic terms $A_{ij} \in \mathbf{A}_i$ to the classes $C_k \in \mathbf{C}$ in rules like:

$$\textit{if } x_i(t) \textit{ is } A_{ij} \textit{ then class is } C_k \textit{ with } cf = \varphi^i_{jk}. \quad (2)$$

where $\varphi^i_{jk} \in [0,1]$ is a confidence factor that represents the rule certainty.

The rule (2) describes a flexible constraint on the values of the variable x_i that can be related to the class (or concept) C_k. For instance, if the variable x_i represents the "salary", then an example of the rule could be: "*if the customer's salary is high then the customer's class is Gold*". The confidence factor represents how much of this relation is true, for instance 95%. The rule's certainty factor may be considered as a relative quantifier [13]. The above example could be interpreted as "*Most of the high salary customers are Gold class customers*", where "most" is the linguistic quantifier that represent 95% of the records in the database. Linguistic quantifiers can be represented as a fuzzy set defined over the unit interval and have been used to define fuzzy summaries in the language Summary SQL [6].

In this work, the confidence factor φ^i_{jk} represents how much the term $A_{ij} \in \mathbf{A}_i$ is linked to the class $C_k \in \mathbf{C}$ in the model defined by the rule (2). A value $\varphi^i_{jk} > 0$ means that the observation of the term A_{ij} is related with the occurrence of the class C_k in φ^i_{jk} of the records.

A set of rules (or a rule base) for each input variable defines a sub-model that is represented by the matrix Φ_i as shown in Table 1. In such matrix, the lines $j = 1...n_i$ are related to the terms in the input variable descriptor set $\mathbf{A}_i$ and the columns $k = 1...m$ are related to classes in the set $\mathbf{C}$, such that $\Phi_i(A_{ij}, C_k) = \varphi^i_{jk}$.

Table 1: Rule base weights' matrix.

Φ_i	C_1	…	C_m
A_{i1}	$\Phi_i(A_{i1}, C_1)$	…	$\Phi_i(A_{i1}, C_m)$
⋮	⋮	⋱	⋮
A_{in_i}	$\Phi_i(A_{in_i}, C_1)$	…	$\Phi_i(A_{in_i}, C_m)$

A rule base is defined for each input variable and used to compute partial outputs by fuzzy inference. The final outputs are computed as the aggregation of all partial outputs as described next.

2.3. Fuzzy inference

The fuzzy classifier output is represented by the class membership vector $\mathbf{y}(t) = \left(\mu_{C_1}(\mathbf{x}(t)), \ldots, \mu_{C_m}(\mathbf{x}(t))\right)$. Each component $\mu_{C_k}(\mathbf{x}(t))$ is the membership of a given input record $\mathbf{x}(t)$ to the class C_k.

The vector $\mathbf{y}_i(t) = \left(\mu_{C_1}(x_i(t)), \ldots, \mu_{C_m}(x_i(t))\right)$ is the partial output membership vector whose components are the classes' membership values considering only the information in the input variable i.

The output of each sub-model is computed by composition-projection inference:

$$\mathbf{y}_i(t) = \mathbf{u}_i(t) \circ \Phi_i. \quad (3)$$

The composition-projection operation computed by the standard max-min composition operator as:

$$\mu_{Ck}(x_i(t)) = \max_{j=1...n_i} \left(\min\left(\mu_{A_{ij}}(x_i(t)), \Phi_i(A_{ij}, C_k)\right)\right). \quad (4)$$

Equation (4) computes the degree of matching of the fuzzification vector $\mathbf{u}_i(t)$ with the prototype of the class C_k, represented by the corresponding column of the rule base matrix Φ_i.

The final output is computed by the aggregation of all partial outputs by an aggregation operator $\mathbf{H}: [0,1]^p \to [0,1]$ as:

$$\mu_{C_k}(\mathbf{x}(t)) = \mathbf{H}\left(\mu_{C_k}(x_1(t)), \ldots, \mu_{C_k}(x_p(t))\right). \quad (5)$$

The best aggregation operator must be chosen according to the semantics of the application. A t-norm operator, such as the "minimum", gives good results to express that all partial conclusions must agree. In classification problems, the final decision is computed by a decision rule. The most usual decision rule is the "maximum rule", where the class is chosen as the one with greatest membership value (Figure 3).

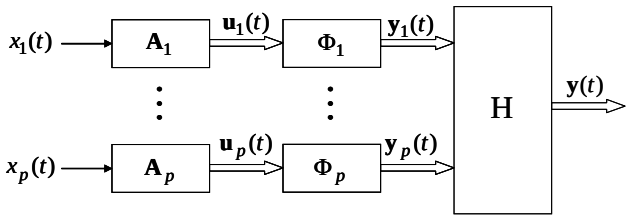


Figure 3. The weighted fuzzy classifier.

The translation of fuzzy rules in to fuzzy queries is presented next.

2.4. Writing fuzzy queries from weighted fuzzy rules

Most of the fuzzy query languages are structured to represent sentences in the form [4][5][6]:

```
SELECT <Atributes> FROM <table>
WHERE:
  <expression>
  <expression>
  …
  <expression>
```

The *attribute* list and the *table* list are usually similar to standard SQL. The *expression* is a fuzzy logic sentence that is translated from the fuzzy classifier's rules in the proposed method.

Non-weighted fuzzy rules may be directly translated into fuzzy queries. Weighted fuzzy rules may not be directly translated since most of fuzzy query languages do not support weights. The weighted fuzzy rules must be converted into non-weighted fuzzy rules. This conversion is based on the conjunctive interpretation of fuzzy if-then rules, where the implication operator is computed as a t-norm operator. Under this interpretation, a weighted rule may be converted into a non-weighted one by introducing the confidence into the rule premise, such that the rule originally written as (2) is converted to:

$$\textit{if } x_i(t) \textit{ is } A_{ij} \textit{ and } w_{ijk} \textit{ is TRUE then class is } C_k \qquad (6)$$

where the auxiliary variable w_{ijk} is defined for each rule such that:

$$\mu_{TRUE}(w_{ijk}) = \phi^i_{jk} = \Phi_i(A_{ij}, C_k) \ . \qquad (7)$$

The membership value $\mu_{TRUE}(w_{ijk})$ is thus independent of the input variable observation $x_i(t)$ and can be stored in the database.

The AND operator in rule (6) is computed by a t-norm operator and a set of rules in the rule base is aggregated using a t-conorm operator, resulting in the max-min composition operator (4).

The translation of a set of fuzzy rules (6) into a fuzzy query sentence must ensure that the result of the query is equivalent to the classifier result. A set of fuzzy rules like (6) are translated to a set of expressions aggregated by disjunctive (OR) operators as:

```
((x1 is A11 AND w11k is TRUE) OR ...OR
   (x1 is A1n1 AND w1n1k is TRUE))
```

The aggregation between rules is computed by an aggregation operator (5), which is a conjunctive operator (AND) between sentences. The fuzzy query evaluation is thus equivalent to the classifier result. The fuzzy query sentence for the class C_k is:

```
SELECT <Attributes> FROM <Data Base>
WHERE:
((x1 is A11 AND w11k is TRUE) OR ...OR
 (x1 is A1n1 AND w1n1k is TRUE)) AND ...
... AND ...
((xi is Ai1 AND wi1k is TRUE) OR ... OR
 (xi is Ain1 AND win1k is TRUE)) AND ...
... AND ...
((xp is Ap1 AND wp1k is TRUE) OR ... OR
 (xp is Apnp AND wpnpk is TRUE)).
```

This solution can be easily implemented in most of fuzzy query languages.

The rule base weights are the core of the model described by the fuzzy classifier and their determination from a data set is described in the next section.

3. Rule Base Learning

This section comments the estimation of the rule base weights from a data set. The weights estimation procedure computes weights for all possible rules. Generally a large number of fuzzy rules is generated, which makes difficult the linguistic interpretation of the model. A prune procedure is thus also proposed to reduce the rule base size and, consequently, the size of the corresponding fuzzy query.

3.1. Weights estimation

The rule base weights are computed from a training data set T, where each sample $t = 1..N$ is a pair $(\mathbf{x}(t), \mathbf{v}(t))$, of which $\mathbf{x}(t)$ is the input variables vector for each record and $\mathbf{v}(t) = (v_1(t), \ldots, v_m(t))$ is the vector containing the correct membership values of $\mathbf{x}(t)$ to each class. In most of practical applications, the correct output $\mathbf{v}(t)$ is binary, like the fuzzification of nominal variables.

Fuzzy rule base weights may be computed in many ways [10]. In this work, for each input variable, each component $\Phi_i(A_{ij}, C_k)$ of the rule base Φ_i is computed as [14]:

$$\Phi_i(A_{ij}, C_k) = \frac{\sum_{t=1..N} u_{ij}(t).v_k(t)}{\sum_{t=1..N} u_{ij}(t)} \ . \qquad (8)$$

where $u_{ij}(t) = \mu_{A_{ij}}(x_i(t))$ and $v_k(t)$ is the correct membership of the sample t to the class C_k.

In a probabilistic framework, equation (8) for a nominal input variable x_i is computed as:

$$\Phi_i(A_{ij}, C_k) = \frac{N_{(A_{ij} \cap C_k)}}{N_{A_{ij}}} \ . \qquad (9)$$

where $N_{(A_{ij} \cap C_k)}$ is the number of samples in the training set that have the value A_{ij} for the variable x_i and are classified as class C_k; and $N_{A_{ij}}$ is the total number of samples that have the value A_{ij} for the variable x_i.

The conditional probability of the class C_k, given the observation of the value A_{ij}, computed from Bayes rule is:

$$P(C_k \mid A_{ij}) = \frac{P(A_{ij} \mid C_k).P(C_k)}{P(A_{ij})} = \frac{\frac{N_{(A_{ij} \cap C_k)}}{N_{C_k}} \frac{N_{C_k}}{N}}{\frac{N_{A_{ij}}}{N}} \quad (10)$$

where N_{C_k} is the total number of samples classified as class C_k and N the total number of samples in the training set.

Comparing equations (8) and (10), it can be seen that the rule base weights are an estimation of the *a posteriori* probability of the class occurrence given by the observation of the value A_{ij}, and:

$$\Phi_i(A_{ij}, C_k) \approx P(C_k \mid A_{ij}). \quad (11)$$

Equality in (11) is achieved for nominal variables. For continuous input variables, the result for rules' output provides the interpolation of the conditional probabilities, weighted by the membership to the corresponding fuzzy set.

The final output is an aggregation of all input variables to compute the final class membership. When this aggregation is computed by a t-norm operator (like the minimum or product) the final class membership is a rough estimation of the joint conditional probability of each class, given the observation of all input variables.

The expression in (8) has been referred as the $\sum$count, and has been used to compute the value of relative quantifiers [13]. Linguistic quantifiers can be processed by some fuzzy query languages such as Summary SQL [7] to compute fuzzy summaries. In this work, the fuzzy rules are translated into fuzzy queries in such a way that they can be processed by any fuzzy query language with an equivalent result to the classifier.

The rule base weights are computed for all possible rules. There are generally a large number of possible rules, many of that are useless for the classification, which makes difficult the interpretation of the classifier model and the corresponding query sentence. A pruning procedure must consider the most important rules to generate a compact set of sentences in the fuzzy query.

3.2. Rule Base Pruning

Fuzzy rules' pruning and simplification is a very active research area [15][16]. In this work, fuzzy rules' pruning is necessary to allow a more compact set of query sentences. The pruning procedure is based on the values of two indexes:

- The Horizontal index (I_H), computed as:

$$I_H(x_i, A_{ij}) = \frac{\sum_{k=1}^{m} \left(\max_{k=1..m} \left(\Phi_i(A_{ij}, C_k) \right) - \Phi_i(A_{ij}, C_k) \right)}{m-1} \quad (12)$$

- The Vertical index (I_V), computed as:

$$I_V(x_i, C_k) = \frac{\sum_{j=1}^{n_i} \left(\max_{j=1..n_i} \left(\Phi_i(A_{ij}, C_k) \right) - \Phi_i(A_{ij}, C_k) \right)}{n_i - 1} \quad (13)$$

The horizontal index $I_H \in [0,1]$ measures how much the fuzzy set A_{ij} of the variable x_i is able to discriminate among the classes. The vertical index $I_V \in [0,1]$ measures how much a class C_k is detected according to the possible values of the variable x_i. Fuzzy rules can be pruned by selecting a threshold values for one of these indexes.

For general-purpose classifiers, the horizontal index should be used since it allows selecting the fuzzy rules that will result in a better classification performance. For fuzzy queries, however, the queries are executed for a given class independently to retrieve the best representatives' records of that class. Thus, the vertical index should be used to select the rules that best discriminates the records of a given class.

3.3. Fuzzy Query Evaluation

The fuzzy query evaluation is based on the standard metrics for evaluation of information retrieval systems: Precision and Recall. The Precision metric is defined as the number of relevant records selected as a fraction of the total number of selected records. The Recall metric is defined as the number of relevant records selected as a fraction of the total number of relevant records.

The Precision and Recall metrics can be computed directly from the confusion matrix, presented in Table 2, which is similar to the one usually used to evaluate classification systems. The rows represent the results of the query system and the columns represent the true information about the selected records.

Table 2: Confusion matrix

	Relevant	**Not Relevant**
Selected	*TP*	*FN*
Not Selected	*FP*	*TN*

The values in the Table 2 are the standard ones: *TP* stands for the number of true positive samples, *FP* is the number of false positive samples, *TN* is the number of

true negative samples and *FN* is the number of false negative samples.

The Precision and Recall metrics are computed as:

$$P = \frac{TP}{(TP + FN)}. \quad (14)$$

$$R = \frac{TP}{(TP + FP)}. \quad (15)$$

Precision and Recall rates are both desired, but it is generally very difficult to achieve high values of both metrics simultaneously: as the Recall rate increases, the Precision usually decreases and vice-versa.

Differently from a standard SQL query, a fuzzy query returns all the records in the database, even those associated to a very small membership value. In a practical application, it is necessary to set a membership threshold or the maximum number of returned records. These parameters must be set to run the query.

The Precision and Recall metrics, when computed over the training set, are useful to give the user an insight of the membership threshold to be set in the fuzzy query. Moreover the user can adjust the threshold to their own needs, based on the results of the training set, which is an impossible using a standard SQL query.

4. Results and Discussion

The evaluation of the current approach is performed under two perspectives. The fuzzy weighted classifier is evaluated from a set of benchmark classification data sets and results are compared with the J4.8 decision tree induction and the Naïve Bayes algorithms, both computed within the Weka suite [17]. The evaluation of the fuzzy query generated by the proposed approach is performed under a target marketing scenario, using the Australian Credit Card data set. The fuzzy query results are compared with the results of the J4.8 decision tree induction algorithm.

4.1. Fuzzy Weighted Classifier Evaluation

A fuzzy weighted classifier, as described above, was generated for a set of benchmarks data sets. For all data sets the fuzzy classifier was generated using $n = 5$ fuzzy membership function of all numerical variables.

Table 3 presents, for each benchmark data, the number of variables, the number of classes and the error rates for the Fuzzy Weighted Classifier (FWC), for the J4.8 algorithm and for the Naïve Bayes (NB) algorithm. The J4.8 and NB algorithms were computed with all default parameters and the results in Table 3 are the average values of the 10-fold cross validation for both algorithms. Bold values indicate the best values for each benchmark.

The FWC will have a better accuracy than the decision tree induction algorithm when numerical variables are predominant and variables are roughly statistically independent [14]. The NB algorithm performs better when the variables are independent and probabilities' densities may be well approximated by the normal probability distribution function. Nevertheless, in general, the classifiers' performance depends mainly on the characteristics of the problem.

Table 3: Classifiers' results

Benchmark	# Vars	# Class	FWC	J4.8	NB
Abalone	8	3	44.00	**39.70**	42.25
Balance Scale	4	3	**36.46**	43.20	36.48
Credit Card	15	2	**13.63**	14.20	23.27
Prima indians	8	2	31.38	25.90	**23.69**
Ionosphere	34	2	**6.27**	8.55	29.91
Breast Cancer	9	2	**5.76**	8.40	5.75
Glass	9	7	47.64	**29.40**	51.87
Wine	13	3	5.62	7.90	**2.80**

The results in Table 3 were obtained without rule base pruning. Fuzzy rule base pruning may improve the performance of the FWC [12][16].

4.2. Fuzzy Query Generation

In order to evaluate the fuzzy queries generated by the proposed approach, the Credit Card approval data set was used. This data set contains 690 examples, each with 6 numeric and 9 categorical attributes. For confidentiality reasons, the attributes' descriptions have been omitted. The records are originally classified into two classes: "approved" (C_1) or "rejected" (C_2), respectively with 45.3% and 54.7% of *apriori* probability. The data set was randomly divided into a training data set and a testing data set containing respectively 460 and 230 records with the same class distribution.

As a "proof of concept", the Fuzzy Query™ tool [20] is used to implement the fuzzy queries generated by the method. The software is a Win32-based application designed to run in conjunction with any ODBC-compliant database management systems.

The fuzzy rule based classifier computes a solution with all rules. The number of rules generated by the classifier is very large, which makes difficult the interpretation. The pruning procedure presented above was thus applied to the fuzzy rule base. The original rule base with 75 rules for each class was thus reduced to 10 rules by selecting the rules where $I_V > 0.5$. The decision tree induced by the J4.8 algorithm has 30 rules.

The simplified rule base was translated into a fuzzy query for class C_1. The resulting fuzzy query is very simple and represents an understandable customer model generated by the fuzzy classifier.

4.3. Fuzzy Query Evaluation

Direct marketing has become one of potential applications of data mining techniques, due to the great volume of available data from transactional databases [18][19]. The objective of direct marketing is to contact customers to offer products and services according to their specific needs. In a direct marketing campaign, the potential customers must be selected from the entire database in order to maximize the profit and minimize the losses due to an overcharge of marketing material sent to wrong customers. Customers' selection or generally "target" selection is thus an important issue in direct marketing and several data mining techniques may be applied to model customer profile according to the objective of the campaign.

The evaluation of the fuzzy queries generated by the proposed approach is done for a target marketing scenario, using the Credit Card data set. In this data set, the relevant class for the application discussed in this section is the class C_1, which represents credit approvals, such that the fuzzy query designed for the class C_1 should not select a record from the class C_2.

The Precision and the Recall metrics as a function of the membership values of the returned records for the training set are shown in Figure 4. The results show that, as expected, as grater the membership value is, the higher is the Precision of the selected records, but the lower is the Recall. An optimal threshold value can thus be chosen to allow high Precision and Recall rates. In this application, the optimal threshold value must be chosen around 0.25.

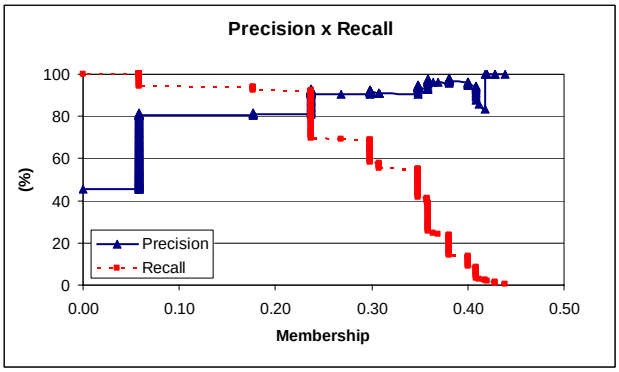


Figure 4: Precision and Recall for the training set.

The Precision and Recall measures computed by the fuzzy weighted query (Fuzzy SQL) with two different thresholds values over the testing set are shown in Table 4. The Precision and Recall metrics computed by the J4.8 algorithm over the same testing set are also shown in Table 4. It can be seen that the fuzzy query results over the testing set correspond roughly to the ones computed from the training set. Moreover the selection of the threshold allows the user to choose between a greater Precision and Recall value, while the J4.8 result is fixed.

The relative importance of Precision and Recall for a certain application depends on the user. Experienced users can work with high recall and low precision, as they are capable to reject irrelevant information. Novice users, on the other hand, need higher precision because they lack the judgment of the experienced users.

Table 4: Fuzzy query evaluation in the testing set

	Precision (%)	Recall (%)
Fuzzy SQL $\delta = 0.25$	92.30	60.60
Fuzzy SQL $\delta = 0.20$	79.27	88.88
J4.8	89.10	75.20

In most of applications in data intensive industries, the number of returned records is very large. Thus it is necessary to select the more suitable record for a given application. As it can be seen from Figure 4, the precision of the records returned by the fuzzy query with grater membership values is very large.

The decision tree induced by the J4.8 algorithm could be used to construct a standard SQL query to retrieve potential customers in the data set. Nevertheless, the query result would not be ordered as the fuzzy query result.

5. Conclusions

This work has shown a methodology that automatically generates fuzzy queries from a training data set. The fuzzy queries are translated from a set of fuzzy weighted classifier rules. A pruning procedure to simplify the fuzzy rule base and the resulting set of fuzzy queries was also proposed.

This methodology has been implemented in a general-purpose fuzzy query tool, which can be connected to any ODBC-compliant database management system.

The fuzzy queries are able to retrieve an ordered list of the records according to the fuzzy rule based model learned by the fuzzy classifier. The Precision and Recall analysis of the selected records of the fuzzy query allow the user to select an optimal threshold for other data.

The proposed methodology was developed focusing direct marketing application, but may be useful in other kind of application such as case based reasoning or information retrieval.

The future directions of this work are to apply this methodology in unstructured data for application to text mining and web mining.

Acknowledgments

The authors are grateful to the Brazilian Research Agencies FINEP, CNPq and CAPES for the financial support for this research.

References

[1] R. A. Ribeiro and A. M. Moreira. Fuzzy query interface for a business database. *Int. J. Human-Computer Studies* 58 pp. 363–391, 2003.

[2] Y. Takahashi. A fuzzy query language for relational databases. *IEEE Trans. on Systems, Man and Cybernetics*, Vol. 21 No. 6, pp. 1576 - 1579, 1991.

[3] D.-Y. Choi. Enhancing the power of Web search engines by means of fuzzy query. *Decision Support Systems*, 35, pp. 31-44, 2003

[4] P. Bosc and O. Pivert. SQLf: a relational database language for fuzzy querying. *IEEE Transactions on Fuzzy Systems*, Vol. 3 No. 1 pp. 1-17, 1995.

[5] J. Kacprzyk and S. Zadrozny. SQLf and FQUERY for Access. *Proceedings of the Joint 9th IFSA World Congress and 20th NAFIPS International Conference*, Vol. 4 pp. 2464 - 2469, 2001.

[6] D. Rasmussen and R. R. Yager. Summary SQL - A Fuzzy Tool for Data Mining. *Intelligent Data Analysis*, 1, pp. 49-58, 1997.

[7] D. Rasmussen and R. R. Yager. Finding fuzzy and gradual functional dependencies with Summary SQL. *Fuzzy Sets and Systems*, 106, pp. 131-142, 1999.

[8] D. Dubois, H. Prade, C. Testemale, Weighted fuzzy pattern matching, *Fuzzy Sets and Systems* 28, pp. 313–331, 1988.

[9] K. Nozaki, H. Ishibuchi and H. Tanaka. Adaptive fuzzy rule-based classification systems. *IEEE Transactions on Fuzzy Systems*, Vol. 4, No. 3, pp. 238 – 250, 1996.

[10] A. Devillez. Four fuzzy supervised classification methods for discriminating classes of non-convex shape. *Fuzzy Sets and Systems*, Vol. 141, No. 2, pp. 219-240, 2004.

[11] H. Ishibuchi and T. Nakashima. Effect of rule weights in fuzzy rule-based classification systems. *IEEE Transactions on Fuzzy Systems*, Vol. 9, No. 4, pp. 506 – 515, 2001.

[12] H. Ishibuchi and T. Yamamoto. Rule Weight Specification in Fuzzy Rule-Based Classification Systems, *IEEE Transactions on Fuzzy Systems*, Vol. 13, No. 4, pp. 428 – 435, 2005.

[13] L. A. Zadeh. A computational approach to fuzzy quantifiers in natural languages. *Computers and Mathematics with Application*. 9, pp. 149-184, 1983.

[14] A. G. Evsukoff, A. C. S. Branco and S. Gentil. A knowledge acquisition method for fuzzy expert systems in diagnosis problems, *IEEE International Conference on Fuzzy Systems, FUZZ-IEEE'97*, Barcelona, July 1997.

[15] W. E. Combs, J. E Andrews. Combinatorial explosion eliminated by a fuzzy rule configuration, *IEEE Trans. on Fuzzy Systems*, Vol. 6 No. 1, pp. 1-11, 1998.

[16] R. P. Espíndola, N. F. F. Ebecken. Data classification by a fuzzy genetic system approach, *Third International Conference on Data Mining*, Bologna, 2002.

[17] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*, 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

[18] M. J. A. Berry, G. Linoff. *Data Mining Techniques: for Marketing, Sales, and Customer Support*, Wiley Computer Publishing, 1997.

[19] U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth and R. Uthurusamy. *Advances in Knowledge Discovery and Data Mining*, AAAI Press/The MIT Press, 1996.

[20] Sonalysts Inc. Fuzzy Query for Windows ((http://fuzzy.sonalysts.com/fuzzyquery.htm), 1998.

Fuzzy Clustering in Parallel Universes with Noise Detection

Bernd Wiswedel and Michael R. Berthold
Department of Computer and Information Science, University of Konstanz
78457 Konstanz, Germany
{wiswedel,berthold}@inf.uni-konstanz.de

Abstract

We present an extension of the fuzzy c-Means algorithm that operates on different feature spaces, so-called parallel universes, simultaneously and also incorporates noise detection. The method assigns membership values of patterns to different universes, which are then adopted throughout the training. This leads to better clustering results since patterns not contributing to clustering in a universe are (completely or partially) ignored. The method also uses an auxiliary universe to capture patterns that do not contribute to any of the clusters in the real universes and therefore likely represent noise. The outcome of the algorithm are clusters distributed over different parallel universes, each modeling a particular, potentially overlapping, subset of the data and a set of patterns detected as noise. One potential target application of the proposed method is biological data analysis where different descriptors for molecules are available but none of them by itself shows global satisfactory prediction results. In this paper we show how the fuzzy c-Means algorithm can be extended to operate in parallel universes and illustrate the usefulness of this method using results on artificial data sets.

1 Introduction

In recent years, researchers have worked extensively in the field of cluster analysis, which has resulted in a wide range of (fuzzy) clustering algorithms [7, 8]. Most of the methods assume the data to be given in a single (mostly high-dimensional numeric) feature space. In some applications, however, it is common to have multiple representations of the data available. Such applications include biological data analysis, in which, e. g. molecular similarity can be defined in various ways. Fingerprints are the most commonly used similarity measure. A fingerprint in a molecular sense is a binary vector, whereby each bit indicates the presence or absence of a molecular feature. The similarity of two compounds can be expressed based on their bit vectors using the Tanimoto coefficient for example. Other descriptors encode numerical features derived from 3D maps, incorporating the molecular size and shape, hydrophilic and hydrophobic regions quantification, surface charge distribution, etc. [5]. Further similarities involve the comparison of chemical graphs, inter-atomic distances, and molecular field descriptors. However, it has been shown that often a single descriptor fails to show satisfactory prediction results [13].

Other application domains include web mining where a document can be described based on its content and on anchor texts of hyperlinks pointing to it [4]. Parts in CAD-catalogues can be represented by 3D models, polygon meshes or textual descriptions. Image descriptors can rely on textual keywords, color information, or other properties [9].

In the following we denote these multiple representations, i. e. different descriptor spaces, as *Parallel Universes* [11, 16], each of which having representations of all objects of the data set. The challenge that we are facing here is to take advantage of the information encoded in the different universes to find clusters that reside in one or more universes each modeling one particular subset of the data. In this paper, we develop an extended fuzzy c-Means (FCM) algorithm [1] with noise detection that is applicable to parallel universes, by assigning membership values from objects to universes. The optimization of the objective function is similar to the original FCM but also includes the learning of the membership values to compute the impact of objects to universes.

In the next section, we will discuss in more detail the concept of parallel universes; section 3 presents related work. We formulate our new clustering scheme in section 4 and illustrate its usefulness with some numeric examples in section 5.

2 Parallel Universes

We consider parallel universes to be a set of feature spaces for a given set of objects. Each object is assigned

a representation in each single universe. Typically, parallel universes encode different properties of the data and thus lead to different measures of similarity. (For instance, similarity of molecular compounds can be based on surface charge distribution or fingerprint representation.) Note, due to these individual measurements they can also show different structural information and therefore exhibit distinctive clustering. This property differs from the problem setting in the so-called *Multi-View Clustering* [3] where a single universe, i. e. view, suffices for learning but the aim is on binding different views to improve the classification accuracy and/or accelerating the learning process.

Note, the concept of parallel universes is not related to *Subspace Clustering* [10], even though it seems so at first. Subspace clustering algorithms seek to identify different subspaces, i. e. subsets of input features, in a dataset. This becomes particularly useful when dealing with high-dimensional data, where often, many dimensions are irrelevant and can mask existing clusters in noise. The main goal of such algorithms is therefore to uncover clusters and subspaces containing only a small, but dense fraction of the data, whereas the clustering in parallel universes is given the definition of all data in all universes and the goal is to exploit this information.

The objective for our problem definition is on identifying clusters located in different universes whereby each cluster models a subset of the data based on some underlying property.

Since standard clustering techniques are not able to cope with parallel universes, one could either restrict the analysis to a single universe at a time or define a descriptor space comprising all universes. However, using only one particular universe omits information encoded in the other representations and the construction of a joint feature space and the derivation of an appropriate distance measure are cumbersome and require great care as it can introduce artifacts.

3 Related Work

Clustering in parallel universes is a relatively new field of research. In [9], the DBSCAN algorithm is extended and applied to parallel universes. DBSCAN uses the notion of dense regions by means of core objects, i. e. objects that have a minimum number k of objects in their (ϵ-) neighborhood. A cluster is then defined as a set of (connected) dense regions. The authors extend this concept in two different ways: They define an object as a neighbor of a core object if it is in the ϵ-neighborhood of this core object either (1) in any of the representations or (2) in all of them. The cluster size is finally determined through appropriate values of ϵ and k. Case (1) seems rather weak, having objects in one cluster even though they might not be similar in any of the representational feature spaces. Case (2), in comparison, is very conservative since it does not reveal local clusters, i. e. subsets of the data that only group in a single universe. However, the results in [9] are promising.

Another clustering scheme called "Collaborative fuzzy clustering" is based on the FCM algorithm and was introduced in [12]. The author proposes an architecture in which objects described in parallel universes can be processed together with the objective of finding structures that are common to all universes. Clustering is carried out by applying the c-Means algorithm to all universes individually and then by exchanging information from the local clustering results based on the partitioning matrices. Note, the objective function, as introduced in [12], assumes the same number of clusters in each universe and, moreover, a global order on the clusters which is very restrictive due to the random initialization of FCM.

A supervised clustering technique for parallel universes was given in [11]. It focuses on a model for a particular (minor) class of interest by constructing local neighborhood histograms, so-called Neighborgrams for each object of interest in each universe. The algorithm assigns a quality value to each Neighborgram and greedily includes the best Neighborgram, no matter from which universe it stems, in the global prediction model. Objects that are covered by this Neighborgram are finally removed from consideration in a sequential covering manner. This process is repeated until the global model has sufficient predictive power.

Blum and Mitchell [4] introduced co-training as a semi-supervised procedure whereby two different hypotheses are trained on two distinct representations and then bootstrap each other. In particular they consider the problem of classifying web pages based on the document itself and on anchor texts of inbound hyperlinks. They require a conditional independence of both universes and state that each representation should suffice for learning if enough labeled data were available. The benefit of their strategy is that (inexpensive) unlabeled data augment the (expensive) labeled data by using the prediction in one universe to support the decision making in the other.

Other related work includes reinforcement clustering [15] and extensions of partitioning methods—such as k-Means, k-Medoids, and EM—and hierarchical, agglomerative methods, all in [3].

4 Clustering Algorithm

In this section, we introduce all necessary notation, review the FCM [1, 6] algorithm and formulate a new objective function that is suitable to be used for parallel universes. The technical details, i. e. the derivation of the objective function, can be found in the appendix.

In the following, we consider U, $1 \leq u \leq U$, parallel universe, each having representational feature vec-

tors for all objects $\vec{x}_{i,u} = (x_{i,u,1}, \ldots, x_{i,u,a}, \ldots x_{i,u,A_u})$ with A_u the dimensionality of the u-th universe. We depict the overall number of objects as $|T|$, $1 \leq i \leq |T|$. We are interested in identifying c_u clusters in universe u. We further assume appropriate definitions of distance functions for each universe $d_u\left(\vec{w}_{k,u}, \vec{x}_{i,u}\right)^2$ where $\vec{w}_{k,u} = (\vec{w}_{k,u,1}, \ldots, \vec{w}_{k,u,a}, \ldots \vec{w}_{k,u,A_u})$ denotes the k-th prototype in the u-th universe.

We confine ourselves to the Euclidean distance in the following. In general, there are no restrictions to the distance metrics other than the differentiability. In particular, they do not need to be of the same type in all universes. This is important to note, since we can use the proposed algorithm in the same feature space, i. e. $\vec{x}_{i,u_1} = \vec{x}_{i,u_2}$ for any u_1 and u_2, but different distance measure across the universes.

4.1 Formulation of new objective function

A standard FCM algorithm relies on one feature space only and minimizes the accumulated sum of distances between patterns $\vec{x}_i$ and cluster centers $\vec{w}_k$, weighted by the degree of membership to which a pattern belongs to a cluster. We refer here to an objective function that also includes noise detection [6]. Note that we omit the subscript u here, as we consider only one universe:

$$J_m = \sum_{i=1}^{|T|} \sum_{k=1}^{c} v_{i,k}^m \, d\left(\vec{w}_k, \vec{x}_i\right)^2 + \delta^2 \sum_{i=1}^{|T|} \left(1 - \sum_{k=1}^{c} v_{i,k}\right)^m . \quad (1)$$

The coefficient $m \in (1, \infty)$ is a fuzzyfication parameter, and $v_{i,k}$ the respective value from the partition matrix, i. e. the degree to which pattern $\vec{x}_i$ belongs to cluster k. The last term serves as a noise cluster; all objects have an fixed, user-defined distance δ^2 to it. Objects that are not close to any cluster center $\vec{w}_k$ are therefore detected as noise.

This function is subject to minimization under the constraint

$$\forall\, i : \sum_{k=1}^{c} v_{i,k} \leq 1\,, \quad (2)$$

requiring that the coverage of any pattern i needs to accumulate to at most 1 (the remainder to 1 represents the membership to the noise cluster).

The above objective function assumes all cluster candidates to be located in the same feature space and is therefore not directly applicable to parallel universes. To overcome this, we introduce a matrix $(z_{i,u})$, $1 \leq i \leq |T|$, $1 \leq u \leq U$, encoding the membership of patterns to universes. A value $z_{i,u}$ close to 1 denotes a strong contribution of pattern $\vec{x}_i$ to the clustering in universe u, and a smaller value, a respectively lesser degree. $z_{i,u}$ has to satisfy standard requirements for membership degrees: it must accumulate to at most 1 considering all universes and must be in the unit interval.

The new objective function is given with

$$J_{m,n} = \sum_{i=1}^{|T|} \sum_{u=1}^{U} z_{i,u}^n \sum_{k=1}^{c_u} v_{i,k,u}^m d_u\left(\vec{w}_{k,u}, \vec{x}_{i,u}\right)^2 + \delta^2 \sum_{i=1}^{|T|} \left(1 - \sum_{u=1}^{U} z_{i,u}\right)^n . \quad (3)$$

Parameter $n \in (1, \infty)$ allows (analogous to m) to have impact on the fuzzyfication of $z_{i,u}$: The larger n the more equal the distribution of $z_{i,u}$, giving each pattern an equal impact to all universes. A value close to 1 will strengthen the composition of $z_{i,u}$ and assign high values to universes where a pattern shows good clustering behavior and small values to those where it does not. Note, we now have U different partition matrices $(v_{i,k})$ to assign membership degrees of objects to cluster prototypes. Similar to the objective function (1), the last term's role is to "localize" the noise and place it in a single auxiliary universe. By assigning patterns to this noise universe, we declare them to be outliers in the data set. The parameter δ^2 reflects the fixed distance between a virtual cluster in the noise universe and all data points. Hence, if the minimum distance between a data point and any cluster in one of the universes becomes greater than δ^2, the pattern is labeled as noise.

As in the standard FCM algorithm, the objective function has to fulfill side constraints. The coverage of a pattern among the partitions in each universe must accumulate to 1:

$$\forall\, i,\, u : \sum_{k=1}^{c_u} v_{i,k,u} = 1\,. \quad (4)$$

This is similar to the constraint of the single universe FCM in (2) but requires to a strict sum of 1 since we do not have a noise cluster in each universe.

Additionally, as mentioned above, the membership of a pattern to different universes has to be at most 1, i. e.

$$\forall\, i : \sum_{u=1}^{U} z_{i,u} \leq 1\,. \quad (5)$$

The remainder to 1 encodes the membership to the noise cluster mentioned above.

The minimization is done with respect to the parameters $v_{i,k,u}$, $z_{i,u}$, and $\vec{w}_{k,u}$. Since the derivation of the objective function is more of technical interest, please refer to the appendix for details.

The optimization splits into three parts. The optimization of the partition values $v_{i,k,u}$ for each universe; determining

the membership degrees of patterns to universes $z_{i,u}$ and finally the adaption of the center vectors of the cluster representatives $\vec{w}_{k,u}$.

The update equations of these parameters are given in (6), (7), and (8). For the partition values $v_{i,k,u}$, it follows

$$v_{i,k,u} = \frac{1}{\sum_{\bar{k}=1}^{c_u} \left(\frac{d_u(\vec{w}_{k,u}, \vec{x}_{i,u})^2}{d_u(\vec{w}_{\bar{k},u}, \vec{x}_{i,u})^2} \right)^{\frac{1}{m-1}}}. \quad (6)$$

Note, this equation is independent of the values $z_{i,u}$ and is therefore identical to the update expression in the single universe FCM. The optimization with respect to $z_{i,u}$ yields

$$z_{i,u} = \frac{1}{\sum_{\bar{u}=1}^{U} \left(\frac{\sum_{k=1}^{c_u} v_{i,k,u}^m d_u(\vec{w}_{k,u}, \vec{x}_{i,u})^2}{\sum_{k=1}^{c_{\bar{u}}} v_{i,k,\bar{u}}^m d_{\bar{u}}(\vec{w}_{k,\bar{u}}, \vec{x}_{i,\bar{u}})^2 + \delta^2} \right)^{\frac{1}{n-1}}}, \quad (7)$$

and update equation for the adaption of the prototype vectors $\vec{w}_{k,u}$ is of the form

$$w_{k,u,a} = \frac{\sum_{i=1}^{|T|} z_{i,u}^n v_{i,k,u}^m x_{i,u,a}}{\sum_{i=1}^{|T|} z_{i,u}^n v_{i,k,u}^m}. \quad (8)$$

Thus, the update of the prototypes depends not only on the partitioning value $v_{i,k,u}$, i. e. the degree to which pattern i belongs to cluster k in universe u, but also to $z_{i,u}$ representing the membership degrees of patterns to the current universe of interest. Patterns with larger values $z_{i,u}$ will contribute more to the adaption of the prototype vectors, while patterns with a smaller degree accordingly to a lesser extent.

Equipped with these update equations, we can introduce the overall clustering scheme in the next section.

4.2 Clustering algorithm

Similar to the standard FCM algorithm, clustering is carried out in an iterative manner, involving three steps:

1. Update of the partition matrices (v)
2. Update of the membership degrees (z)
3. Update of the prototypes ($\vec{w}$)

More precisely, the clustering procedure is given as:

(1)	*Given:* Input pattern set described in U parallel universes: $\vec{x}_{i,u}$, $1 \leq i \leq \|T\|$, $1 \leq u \leq U$
(2)	*Select:* A set of distance metrics $d_u(\cdot,\cdot)^2$, and the number of clusters for each universe c_u, $1 \leq u \leq U$, define parameter m and n
(3)	*Initialize:* Partition parameters $v_{i,k,u}$ with random values and the cluster prototypes by drawing samples from the data. Assign equal weight to all membership degrees $z_{i,u} = \frac{1}{U}$.
(4)	*Train:*
(5)	*Repeat*
(6)	Update partitioning values $v_{i,k,u}$ according to (6)
(7)	Update membership degrees $z_{i,u}$ according to (7)
(8)	Compute prototypes $\vec{w}_{i,u}$ using (8)
(9)	*until* a termination criterion has been satisfied

The algorithm starts with a given set of universe definitions and the specification of the distance metrics to use. Also, the number of clusters in each universe needs to be defined in advance. The membership degrees $z_{i,u}$ are initialized with equal weight (line (3)), thus having the same impact on all universes. The optimization phase in line (5) to (9) is—in comparison to the standard FCM algorithm—extended by the optimization of the membership degrees, line (7). The possibilities for the termination criterion in line (9) are manifold. One can stop after a certain number of iterations or use the change of the value of the objective function (3) between two successive iterations as stopping criteria. There are also more sophisticated approaches, for instance the change to the partition matrices during the optimization.

Just like the FCM algorithm, this method suffers from the fact that the user has to specify the number of prototypes to be found. Furthermore, our approach even requires the definition of cluster counts *per* universe. There are numerous approaches to suggest the number of clusters in the case of the standard FCM, [17, 14, 2] to name but a few. Although we have not yet studied their applicability to our problem definition we do believe that some of them can be adapted to be used in our context as well.

5 Experimental Results

In order to demonstrate this approach, we generated synthetic data sets with different numbers of parallel universes. For simplicity we restricted the size of a universe to 2 dimensions and generated 2 Gaussian distributed clusters per universe. We used 70% of the data (overall cardinality 2000 patterns) to build groupings by assigning each object to one of the universes and drawing its features in that universe according to the distribution of the cluster (randomly picking one of the two). The features of that object in the other

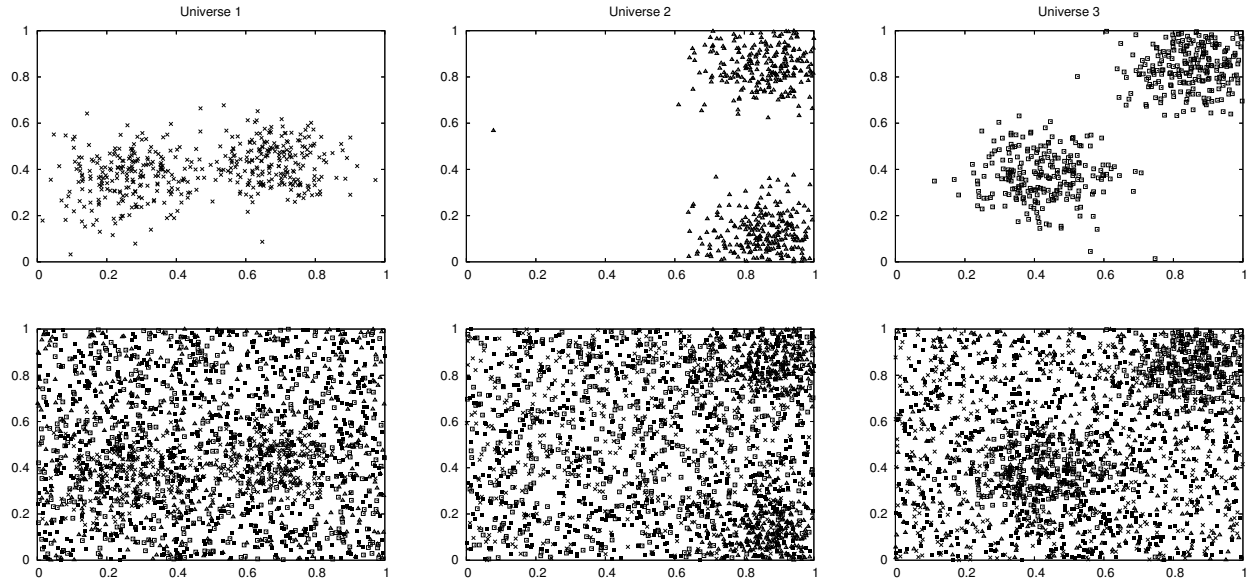


Figure 1. Three universes of a synthetic data set. The top figures show only objects that were generated within the respective universe (using two clusters per universe). The bottom figures show all patterns; note that most of them (i. e. the ones from the other two universes), are noise in this particular universe. They also show the patterns that were not assigned to any of the cluster and represent noise in all of the universes. For clarification we use different shapes for objects that originate from different universes.

universes were drawn from a uniform distribution, i. e. they represent noise in these universes. The remaining 30% of the overall data was generated to be noise in all universes to test the ability of the algorithm to identify patterns that do not cluster at all. Figure 1 shows an example data set with three universes. The top figures show only the objects that were generated to cluster in the respective universe. They define the reference clustering. The bottom figures include all objects, i. e. patterns that cluster in any of the universes plus 30% noise, and show the universes as they are presented to the clustering algorithm. In this example, when looking solely at one universe, about 3/4 of the data does not contribute to clustering and therefore are noise in that universe[1].

To compare the results we applied the FCM algorithm with an auxiliary noise cluster as presented in [6] to the joint feature space of all universes and set the number of desired clusters to the overall number of generated clusters. Thus, the numbers of dimensions and clusters were two times the number of universes. In order to test the ability of noise detection we also applied the fuzzy clustering algorithm for parallel universes without noise universe [16]. The objective function is similar to (3) but with no explicit notion of noise. The algorithm partitions the data such that each pattern is assigned to one of the clusters.

The cluster membership decision for the single-universe FCM is based on the highest value of the partition values, i. e. the cluster to a pattern i is determined by $\bar{k} = \arg\max_{1\leq k\leq c}\{v_{i,k}\}$. If this value is less than the membership to the noise cluster, $v_{i,\bar{k}} < 1 - \sum_k^{c_u} v_{i,k}$, the pattern is labeled as noise.

When the universe information is taken into account, a cluster decision is based on the memberships to universes $z_{i,u}$ and memberships to clusters $v_{i,k,u}$. The "winning" universe is determined by $\bar{u} = \arg\max_{1\leq u\leq U}\{z_{i,u}\}$. If this value is less than the membership degree to the noise universe, $z_{i,\bar{u}} < 1 - \sum_u^U z_{i,u}$, the pattern is labeled as noise, otherwise the cluster is calculated as $\bar{k} = \arg\max_{1\leq k\leq c_{\bar{u}}}\{v_{i,k,c_{\bar{u}}}\}$.

We used the following quality measure to compare different clustering results [9]:

$$Q_K(C) = \sum_{C_i \in C} \frac{|C_i|}{|T|} \cdot (1 - \mathrm{entropy}_K(C_i)) \ ,$$

where K is the reference clustering, i. e. the clusters as generated, C the clustering to evaluate, and $\mathrm{entropy}_K(C_i)$ the entropy of cluster C_i with respect to K. This function is 1

[1] More precisely 77% which is 2/3 of 70% clustering in other universes plus 30% overall noise.

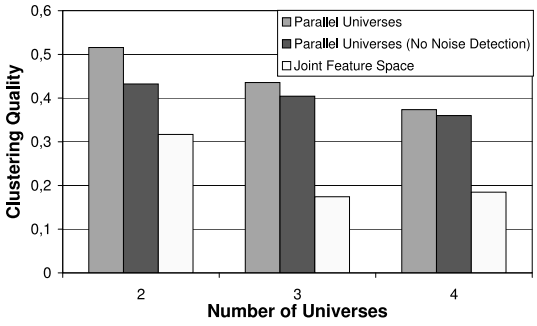


Figure 2. Clustering quality for 3 different data sets. The number of universes ranges from 2 to 4 universes. Note how the cluster quality of the joint feature space drops sharply whereas the parallel universe approach seems less affected. An overall decline of cluster quality is to be expected since the number of clusters to be detected increases.

if C equals K and 0 if all clusters are completely puzzled such that they all contain an equal fraction of the clusters in K or no clusters are detected at all. Thus, the higher the value, the better the clustering.

Figure 2 summarizes the quality values for 3 experiments compared to the FCM with noise detection [6] and the fuzzy clustering in parallel universes with no noise handling [16]. The number of universes ranges from 2 to 4. Clearly, for this data set, our algorithm takes advantage of the information encoded in different universes and identifies the major parts of the original clusters. However, when applying FCM to the joint feature space, most of the data was labeled as noise. It was noticeable, that the noise detection (30% of the data was generated such that it does not cluster in any universe) decreased when having more universes since the number of clusters—and therefore the chance to "hit" one of them when drawing the features of a noise object—increased for this artifical data. As a result, the difference in quality between our new clustering algorithm which allows noise detection and the clustering algorithm that forces a cluster prediction declines when having more universes. This effect occurs no matter how carefully the noise distance parameter δ^2 is chosen.

However, if we have only few universes, the difference is very obvious. Figure 3 visually demonstrates the clusters from the foregoing example as they are determined by the fuzzy c-Means algorithm in parallel universes (the three top figures) and our new algorithm, i. e. with noise detection (bottom figures). The figures show only the patterns that build clusters in the respective universe; other patterns, either covered by clusters in the remaining universes or detected as noise, are filtered out. Note how the clusters in the top figures are spread and contain patterns that obviously do not make much sense for this clustering. This is due to the fact that the algorithm is not allowed to declare such patterns as outliers: each pattern must be assigned to a cluster. The bottom figures, in comparison, show the clusters as round-shaped, dense regions. They have been generated using the new objective function. Patterns that in the top figures distort the clusters are not included here. It shows nicely that the algorithm does not force a cluster prediction and will recognize these patterns as being noise.

We chose this kind of data generation to test the ability to detect clusters that are blurred by noise. Particularly in biological data analysis it is common to have noisy data for which different descriptors are available and each by itself exhibits only little clustering power. Obviously this is by no means proof that the method will always detect clusters spread out over parallel universes but these early results are quite promising.

6 Conclusion

We considered the problem of unsupervised clustering in parallel universes, i. e. problems where multiple representations are available for each object. We developed an extension of the fuzzy c-Means algorithm with noise detection that uses membership degrees to model the impact of objects to the clustering in a particular universe. By incorporating these membership values into the objective function, we were able to derive update equations which minimize the objective with respect to these values, the partition matrices, and the prototype center vectors. In order to model the concept of noise, i. e. patterns that apparently are not contained in any of the cluster, we introduced an auxiliary noise universe that has one single cluster to which all objects have a fixed, pre-defined distance. Patterns that are not covered by any of the clusters will get assigned a high membership to this universe and can therefore be revealed as noise.

The clustering algorithm itself works in an iterative manner using the above update equations to compute a (local) minimum. The result are clusters located in different parallel universes, each modeling only a subset of the overall data and ignoring data that do not contribute to clustering in a universe.

We demonstrated that the algorithm performs well on a synthetic data set and nicely exploits the information of having different universes.

Further studies will concentrate on the overlap of clusters. The proposed objective function rewards clusters that

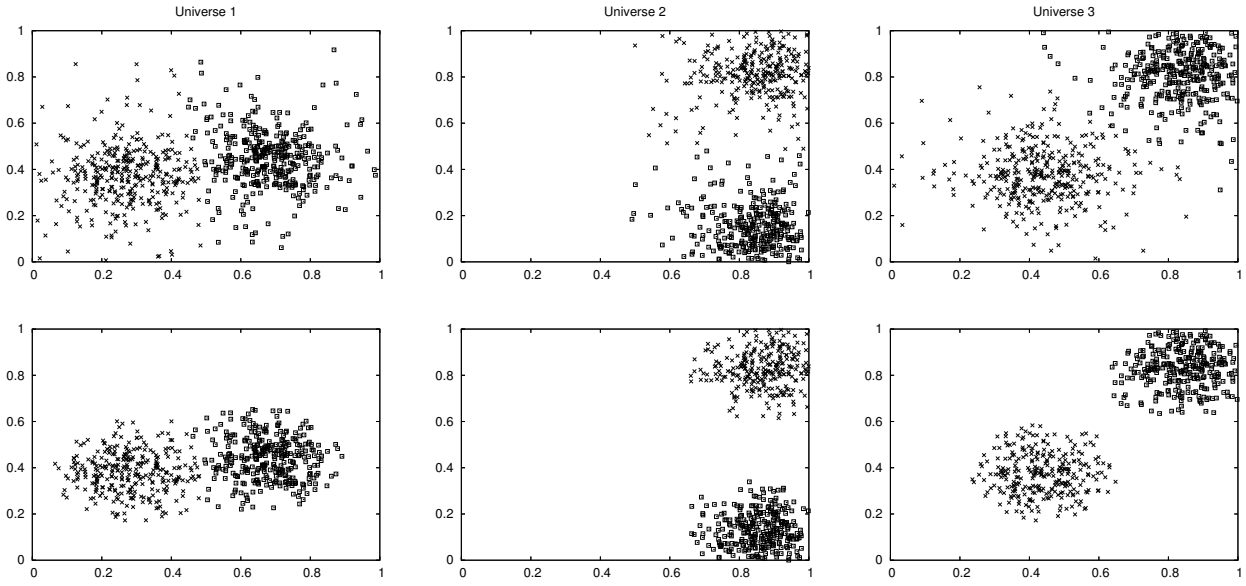


Figure 3. The top figures show the clusters as they are found when applying the algorithm with no noise detection [16]. The bottom figures show the clusters found by the algorithm using noise detection. While the clusters in the top figures contain patterns that do not appear natural for this clustering, the clustering with noise detection reveals those patterns and builds up clear groupings.

only occur in one universe. Objects that cluster well in more than one universe could possibly be identified when having balanced membership values to the universes but very unbalanced partitioning values for the cluster memberships.

Other studies will focus on the applicability of the proposed method to real world data and heuristics that adjust the number of clusters per universe.

Acknowledgment

This work was supported by the DFG Research Training Group GK-1042 "Explorative Analysis and Visualization of large Information Spaces".

Appendix

In order to compute a minimum of the objective function (3) with respect to (4) and (5), we exploit a Lagrange technique to merge the constrained part of the optimization problem with the unconstrained one. Note we skip the extra notation of the noise universe in (3) as one can think of an additional universe, i. e. the number of universe is $U+1$, that has one cluster to which all patterns have a fixed distance of δ^2. The derivation can then be applied as follows.

It leads to a new objective function F_i that we minimize for each pattern $\vec{x}_i$ individually,

$$F_i = \sum_{u=1}^{U} z_{i,u}^n \sum_{k=1}^{c_u} v_{i,k,u}^m d_u\left(\vec{w}_{k,u}, \vec{x}_{i,u}\right)^2 + \sum_{u=1}^{U} \mu_u \left(1 - \sum_{k=1}^{c_u} v_{i,k,u}\right) + \lambda \left(1 - \sum_{u=1}^{U} z_{i,u}\right). \quad (9)$$

The parameters λ and μ_u, $1 \le u \le U$, denote the Lagrange multiplier to take (4) and (5) into account. The necessary conditions leading to local minima of F_i read as

$$\frac{\partial F_i}{\partial z_{i,u}} = 0, \quad \frac{\partial F_i}{\partial v_{i,k,u}} = 0, \quad \frac{\partial F_i}{\partial \lambda} = 0, \quad \frac{\partial F_i}{\partial \mu_u} = 0\,, \quad 1 \le u \le U, \quad 1 \le k \le c_u. \quad (10)$$

In the following we will derive update equations for the z and v parameters. Evaluating the first derivative of the equations in (10) yields the expression

$$\frac{\partial F_i}{\partial z_{i,u}} = n\, z_{i,u}^{n-1} \sum_{k=1}^{c_u} v_{i,k,u}^m d_u\left(\vec{w}_{k,u}, \vec{x}_{i,u}\right)^2 - \lambda = 0,$$

and hence

$$z_{i,u} = \left(\frac{\lambda}{n}\right)^{\frac{1}{n-1}} \left(\frac{1}{\sum_{k=1}^{c_u} v_{i,k,u}^m d_u\left(\vec{w}_{k,u}, \vec{x}_{i,u}\right)^2}\right)^{\frac{1}{n-1}}. \quad (11)$$

We can rewrite the above equation

$$\left(\frac{\lambda}{n}\right)^{\frac{1}{n-1}} = z_{i,u}\left(\sum_{k=1}^{c_u} v_{i,k,u}^m d_u\left(\vec{w}_{k,u},\vec{x}_{i,u}\right)^2\right)^{\frac{1}{n-1}}. \quad (12)$$

From the derivative of F_i w. r. t. λ in (10), it follows

$$\begin{aligned}\frac{\partial F_i}{\partial \lambda} &= 1-\sum_{u=1}^{U} z_{i,u} = 0\\ \sum_{u=1}^{U} z_{i,u} &= 1\,, \end{aligned} \quad (13)$$

which returns the normalization condition as in (5). Using the formula for $z_{i,u}$ in (11) and integrating it into expression (13) we compute

$$\sum_{u=1}^{U}\left(\frac{\lambda}{n}\right)^{\frac{1}{n-1}}\left(\frac{1}{\sum_{k=1}^{c_u} v_{i,k,u}^m d_u\left(\vec{w}_{k,u},\vec{x}_{i,u}\right)^2}\right)^{\frac{1}{n-1}} = 1$$

$$\left(\frac{\lambda}{n}\right)^{\frac{1}{n-1}}\sum_{u=1}^{U}\left(\frac{1}{\sum_{k=1}^{c_u} v_{i,k,u}^m d_u\left(\vec{w}_{k,u},\vec{x}_{i,u}\right)^2}\right)^{\frac{1}{n-1}} = 1 \quad (14)$$

We make use of (12) and substitute $\left(\frac{\lambda}{n}\right)^{\frac{1}{n-1}}$ in (14). Note, we use $\bar{u}$ as parameter index of the sum to address the fact that it covers all universes, whereas u denotes the current universe of interest. It follows

$$\begin{aligned}1 &= z_{i,u}\left(\sum_{k=1}^{c_u} v_{i,k,u}^m d_u\left(\vec{w}_{k,u},\vec{x}_{i,u}\right)^2\right)^{\frac{1}{n-1}}\\ &\quad\times\sum_{\bar{u}=1}^{U}\left(\frac{1}{\sum_{k=1}^{c_{\bar{u}}} v_{i,k,\bar{u}}^m d_{\bar{u}}\left(\vec{w}_{k,\bar{u}},\vec{x}_{i,\bar{u}}\right)^2}\right)^{\frac{1}{n-1}},\end{aligned}$$

which can be simplified to

$$1 = z_{i,u}\sum_{\bar{u}=1}^{U}\left(\frac{\sum_{k=1}^{c_u} v_{i,k,u}^m d_u\left(\vec{w}_{k,u},\vec{x}_{i,u}\right)^2}{\sum_{k=1}^{c_{\bar{u}}} v_{i,k,\bar{u}}^m d_{\bar{u}}\left(\vec{w}_{k,\bar{u}},\vec{x}_{i,\bar{u}}\right)^2}\right)^{\frac{1}{n-1}},$$

and returns an immediate update expression for the membership $z_{i,u}$ of pattern i to universe u (see also (7)):

$$z_{i,u} = \frac{1}{\sum_{\bar{u}=1}^{U}\left(\frac{\sum_{k=1}^{c_u} v_{i,k,u}^m d_u(\vec{w}_{k,u},\vec{x}_{i,u})^2}{\sum_{k=1}^{c_{\bar{u}}} v_{i,k,\bar{u}}^m d_{\bar{u}}(\vec{w}_{k,\bar{u}},\vec{x}_{i,\bar{u}})^2}\right)^{\frac{1}{n-1}}}.$$

Analogous to the calculations above we can derive the update equation for value $v_{i,k,u}$which represents the partitioning value of pattern i to cluster k in universe u. From (10) it follows

$$\frac{\partial F_i}{\partial v_{i,k,u}} = z_{i,u}^n\, m\, v_{i,k,u}^{m-1} d_u\left(\vec{w}_{k,u},\vec{x}_{i,u}\right)^2 - \mu_u = 0,$$

and thus

$$\begin{aligned} v_{i,k,u} &= \left(\frac{\mu_u}{m\, z_{i,u}^n\, d_u\left(\vec{w}_{k,u},\vec{x}_{i,u}\right)^2}\right)^{\frac{1}{m-1}} \quad (15)\\ \left(\frac{\mu_u}{m\, z_{i,u}^n}\right)^{\frac{1}{m-1}} &= v_{i,k,u}\left(d_u\left(\vec{w}_{k,u},\vec{x}_{i,u}\right)^2\right)^{\frac{1}{m-1}}. \quad (16)\end{aligned}$$

Zeroing the derivative of F_i w. r. t. μ_u will result in condition (4), ensuring that the partition values sum to 1, i. e.

$$\frac{\partial F_i}{\partial \mu_u} = 1-\sum_{k=1}^{c_u} v_{i,k,u} = 0\,. \quad (17)$$

We use (15) and (17) to come up with

$$\begin{aligned}1 &= \sum_{k=1}^{c_u}\left(\frac{\mu_u}{m\, z_{i,u}^n\, d_u\left(\vec{w}_{k,u},\vec{x}_{i,u}\right)^2}\right)^{\frac{1}{m-1}},\\ 1 &= \left(\frac{\mu_u}{m\, z_{i,u}^n}\right)^{\frac{1}{m-1}}\sum_{k=1}^{c_u}\left(\frac{1}{d_u\left(\vec{w}_{k,u},\vec{x}_{i,u}\right)^2}\right)^{\frac{1}{m-1}} \quad (18)\end{aligned}$$

Equation (16) allows us to replace the first multiplier in (18). We will use the $\bar{k}$ notation to point out that the sum in (18) considers all partitions in a universe and k to denote one particular cluster coming from (15),

$$\begin{aligned}1 &= v_{i,k,u}\left(d_u\left(\vec{w}_{k,u},\vec{x}_{i,u}\right)^2\right)^{\frac{1}{m-1}}\\ &\quad\times\sum_{\bar{k}=1}^{c_u}\left(\frac{1}{d_u\left(\vec{w}_{\bar{k},u},\vec{x}_{i,u}\right)^2}\right)^{\frac{1}{m-1}}\\ 1 &= v_{i,k,u}\sum_{\bar{k}=1}^{c_u}\left(\frac{d_u\left(\vec{w}_{k,u},\vec{x}_{i,u}\right)^2}{d_u\left(\vec{w}_{\bar{k},u},\vec{x}_{i,u}\right)^2}\right)^{\frac{1}{m-1}}\end{aligned}$$

Finally, the update rule for $v_{i,k,u}$ arises as (see also 6):

$$v_{i,k,u} = \frac{1}{\sum_{\bar{k}=1}^{c_u}\left(\frac{d_u(\vec{w}_{k,u},\vec{x}_{i,u})^2}{d_u(\vec{w}_{\bar{k},u},\vec{x}_{i,u})^2}\right)^{\frac{1}{m-1}}}.$$

For the sake of completeness we also derive the update rules for the cluster prototypes $\vec{w}_{k,u}$. We confine ourselves to the Euclidean distance here, assuming the data is normalized[2]:

$$d_u\left(\vec{w}_{k,u},\vec{x}_{i,u}\right)^2 = \sum_{a=1}^{A_u}\left(w_{k,u,a}-x_{i,u,a}\right)^2, \quad (19)$$

[2]The derivation of the updates using other than the Euclidean distance works in a similar manner.

with A_u the number of dimensions in universe u and $w_{k,u,a}$ the value of the prototype in dimension a. $x_{i,u,a}$ is the value of the a-th attribute of pattern i in universe u, respectively. The necessary condition for a minimum of the objective function (3) is of the form $\nabla_{\vec{w}_{k,u}} J = 0$. Using the Euclidean distance as given in (19) we obtain

$$\begin{aligned}
\frac{\partial J_{m,n}}{\partial w_{k,u,a}} = 0 &= 2 \sum_{i=1}^{|T|} z_{i,u}^n \, v_{i,k,u}^m \, (w_{k,u,a} - x_{i,u,a}) \\
w_{k,u,a} \sum_{i=1}^{|T|} z_{i,u}^n \, v_{i,k,u}^m &= \sum_{i=1}^{|T|} z_{i,u}^n \, v_{i,k,u}^m \, x_{i,u,a} \\
w_{k,u,a} &= \frac{\sum_{i=1}^{|T|} z_{i,u}^n \, v_{i,k,u}^m \, x_{i,u,a}}{\sum_{i=1}^{|T|} z_{i,u}^n \, v_{i,k,u}^m} ,
\end{aligned}$$

which is also given with (8).

References

[1] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York, 1981.

[2] J. C. Bezdek and R. J. Hathaway. VAT: a tool for visual assessment of (cluster) tendency. In *Proceedings of the 2002 International Joint Conference on Neural Networks (IJCNN '02)*, pages 2225–2230, 2002.

[3] S. Bickel and T. Scheffer. Multi-view clustering. In *Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM'04)*, pages 19–26, 2004.

[4] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual Conference on Computational Learning Theory (COLT'98)*, pages 92–100. ACM Press, 1998.

[5] G. Cruciani, P. Crivori, P.-A. Carrupt, and B. Testa. Molecular fields in quantitative structure-permeation relationships: the VolSurf approach. *Journal of Molecular Structure*, 503:17–30, 2000.

[6] R. N. Dave. Characterization and detection of noise in clustering. *Pattern Recognition Letters*, 12:657–664, 1991.

[7] D. J. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining*. MIT Press, 2001.

[8] F. Höppner, F. Klawoon, R. Kruse, and T. Runkler. *Fuzzy Cluster Analysis*. John Wiley, Chichester, England, 1999.

[9] K. Kailing, H.-P. Kriegel, A. Pryakhin, and M. Schubert. Clustering multi-represented objects with noise. In *PAKDD*, pages 394–403, 2004.

[10] L. Parsons, E. Haque, and H. Liu. Subspace clustering for high dimensional data: a review. *SIGKDD Explor. Newsl.*, 6(1):90–105, 2004.

[11] D. E. Patterson and M. R. Berthold. Clustering in parallel universes. In *Proceedings of the 2001 IEEE Conference in Systems, Man and Cybernetics*. IEEE Press, 2001.

[12] W. Pedrycz. Collaborative fuzzy clustering. *Pattern Recognition Letters*, 23(14):1675–1686, 2002.

[13] A. Schuffenhauer, V. J. Gillet, and P. Willett. Similarity searching in files of three-dimensional chemical structures: Analysis of the bioster database using two-dimensional fingerprints and molecular field descriptors. *Journal of Chemical Information and Computer Sciences*, 40(2):295–307, 2000.

[14] N. B. Venkateswarlu and P. S. V. S. K. Raju. Fast ISODATA clustering algorithms. *Pattern Recognition*, 25(3):335–342, 1992.

[15] J. Wang, H.-J. Zeng, Z. Chen, H. Lu, L. Tao, and W.-Y. Ma. ReCoM: Reinforcement clustering of multi-type interrelated data objects. In *In Proceedings of the 26th annual international ACM SIGIR conference on research and development in information retrieval (SIGIR'03)*, pages 274–281, 2003.

[16] B. Wiswedel and M. R. Berthold. Fuzzy clustering in parallel universes. In *Proc. Conf. North American Fuzzy Information Processing Society (NAFIPS 2005)*, pages 567–572, 2005.

[17] R. R. Yager and D. P. Filev. Approximate clustering via the mountain method. *IEEE Trans. Systems Man Cybernet.*, 24(8):1279–1284, August 1994.

Accuracy-Complexity Tradeoff Analysis by Multiobjective Rule Selection

Hisao Ishibuchi
Osaka Prefecture University
hisaoi@cs.osakafu-u.ac.jp

Yusuke Nojima
Osaka Prefecture University
nojima@cs.osakafu-u.ac.jp

Abstract

In this paper, we clearly demonstrate that genetics-based multiobjective rule selection can significantly improve the accuracy-complexity tradeoff curve of extracted rule sets for classification problems. First a prespecified number of rules are extracted from numerical data with continuous attributes using a heuristic rule extraction criterion. Then genetics-based multiobjective rule selection is applied to the extracted rule set to find a number of non-dominated rule subsets with respect to the classification accuracy and the number of rules. Experimental results clearly show that multiobjective rule selection finds a number of smaller rule subsets with higher accuracy than heuristically extracted rule sets. That is, the accuracy-complexity tradeoff curve is improved by multiobjective rule selection.

1. Introduction

Almost all real-world decision making problems involve multiple objectives. These objectives usually conflict with each other. In the case of knowledge extraction, we want to maximize the accuracy of extracted rules. At the same time, we want to minimize their complexity (i.e., maximize their interpretability). Evolutionary multiobjective optimization (EMO) is an active research area in the field of evolutionary computation (see Deb [1] and Coello et al. [2]). The main advantage of EMO approaches over conventional optimization techniques is that a number of non-dominated solutions are simultaneously obtained by their single run. The obtained non-dominated solutions help the decision maker to understand the tradeoff structure between conflicting objectives (e.g., through their visualization). Such knowledge about the tradeoff structure in turn helps the decision maker to choose the final solution from the obtained non-dominated ones.

In some conventional (i.e., non-evolutionary) approaches to multiobjective optimization, the decision maker is supposed to integrate multiple objectives into a single scalar objective function by assigning a relative weight to each objective. The assessment of the relative weight, however, is usually very difficult because the decision maker has no *a priori* information about the tradeoff between conflicting objectives. For example, it is very difficult to assign relative weights to the two major goals in knowledge extraction: accuracy maximization and complexity minimization. In other conventional approaches, the decision maker is requested to assign the target value to each objective. The specification of the target value is also difficult for the decision maker. For example, it is not easy to specify the target values for the classification accuracy and the number of extracted rules before the decision maker knows the tradeoff structure between the accuracy and the complexity of rule sets for a particular classification problem at hand.

Recently EMO approaches have been employed in some studies on modeling and classification. For example, Kupinski & Anastasio [3] used an EMO algorithm to generate non-dominated neural networks on a receiver operating characteristic curve. Gonzalez et al. [4] generated non-dominated radial basis function networks of different sizes. Abbass [5] used a memetic EMO algorithm (i.e., a hybrid EMO algorithm with local search) to speed up the back-propagation algorithm where multiple neural networks of different sizes were evolved to find an appropriate network structure. Non-dominated neural networks were combined into a single ensemble classifier in [6]-[8]. The use of EMO algorithms to design ensemble classifiers was also proposed in Ishibuchi & Yamamoto [9] where multiple fuzzy rule-based classifiers of different sizes were generated. In some studies on fuzzy rule-based systems [10]-[17], EMO algorithms were used to analyze the tradeoff between accuracy and interpretability.

In this paper, we intend to clearly demonstrate the effectiveness of EMO approaches to knowledge extraction from numerical data for classification problems with many continuous attributes. First we briefly explain some basic concepts in multiobjective optimization in Section 2. Next we explain our EMO approach to knowledge extraction. Our approach consists of two stages: heuristic rule extraction (i.e., data mining stage) and genetics-based multiobjective rule selection (i.e., optimization stage). These two stages are described in Section 3 and Section 4, respectively. In the second stage of our EMO approach, knowledge extraction is formulated as a two-objective rule

selection problem. The two objectives are to maximize the classification accuracy and to minimize the number of rules. An EMO algorithm is employed to efficiently find a number of non-dominated rule sets with respect to these two objectives for classification problems with many continuous attributes. In Section 5, obtained non-dominated rule sets are compared with heuristically extracted rule sets. Finally Section 6 concludes this paper.

2. Multiobjective Optimization

We explain some basic concepts in multiobjective optimization using the following *k*-objective problem:

$$\text{Minimize } \mathbf{z} = (f_1(\mathbf{y}),\ f_2(\mathbf{y}),\ ...,\ f_k(\mathbf{y})), \quad (1)$$

$$\text{subject to } \mathbf{y} \in \mathbf{Y}, \quad (2)$$

where **z** is the objective vector, **y** is the decision vector, and **Y** is the feasible region in the decision space. Since the *k* objectives usually conflict with each other, there is no absolutely optimal solution $\mathbf{y}^*$ ($\mathbf{y}^* \in \mathbf{Y}$) that satisfies the following relation with respect to all objectives:

$$\forall i \quad f_i(\mathbf{y}^*) = \min\{f_i(\mathbf{y}) : \mathbf{y} \in \mathbf{Y}\}. \quad (3)$$

In general, multiobjective optimization problems have a number of non-dominated (i.e., Pareto-optimal) solutions. Now we briefly explain the concept of Pareto-optimality. Let **a** and **b** be two feasible solutions of the *k*-objective problem in (1)-(2). When the following condition holds, **a** can be viewed as being better than **b**:

$$\forall i \quad f_i(\mathbf{a}) \le f_i(\mathbf{b}) \quad \text{and} \quad \exists j \quad f_j(\mathbf{a}) < f_j(\mathbf{b}). \quad (4)$$

In this case, we say that **a** dominates **b** (equivalently **b** is dominated by **a**). This dominance relation between **a** and **b** in (4) is sometimes denoted as $\mathbf{a} \prec \mathbf{b}$.

When **b** is not dominated by any other feasible solutions, **b** is referred to as a non-dominated (i.e., Pareto-optimal) solution of the *k*-objective problem in (1)-(2). That is, **b** is a Pareto-optimal solution when there is no feasible solution **a** that satisfies $\mathbf{a} \prec \mathbf{b}$. The set of all Pareto-optimal solutions forms a tradeoff surface in the *k*-dimensional objective space. This tradeoff surface in the objective space is referred to as the Pareto-front. Various EMO algorithms have been proposed to efficiently find a number of Pareto-optimal (or near Pareto-optimal) solutions that are uniformly distributed on the Pareto-front [1]-[2].

3. Heuristic Extraction of Classification Rules

Our EMO approach to knowledge extraction consists of two stages: heuristic rule extraction and genetics-based multiobjective rule selection. In the first stage (i.e., data mining stage), a prespecified number of promising rules are efficiently extracted in a heuristic manner. Then a number of non-dominated rule sets, which are subsets of the extracted rules, are found by an EMO algorithm in the second stage (i.e., optimization stage). These two stages are explained in this section and the next section, respectively.

Let us assume that we have *m* training (i.e., labeled) patterns $\mathbf{x}_p = (x_{p1}, ..., x_{pn})$, $p = 1, 2, ..., m$ from *M* classes in the *n*-dimensional continuous pattern space where x_{pi} is the attribute value of the *p*-th training pattern for the *i*-th attribute ($i = 1, 2, ..., n$). We denote these training patterns by *D* (i.e., $D = \{\mathbf{x}_1, ..., \mathbf{x}_m\}$). We also denote training patterns from Class *h* as *D*(Class *h*) where $h = 1, 2, ..., M$.

For our *n*-dimensional *M*-class classification problem, we use the following classification rule:

$$\text{Rule } R_q: \text{If } x_1 \text{ is } A_{q1} \text{ and } ... \text{ and } x_n \text{ is } A_{qn} \text{ then Class } C_q \text{ with } CF_q, \quad (5)$$

where R_q is the label of the *q*-th rule, $\mathbf{x} = (x_1, ..., x_n)$ is an *n*-dimensional pattern vector, A_{qi} is an antecedent interval, C_q is a class label, and CF_q is a rule weight (i.e., certainty grade). Each antecedent condition "x_i is A_{qi}" means the inclusion relation $x_i \in A_{qi}$ (i.e., the inequality relation $A_{qi}^L \le x_i \le A_{qi}^U$ where $A_{qi} = [A_{qi}^L, A_{qi}^U]$). We denote the antecedent part of the classification rule R_q in (5) by the interval vector $\mathbf{A}_q$ where $\mathbf{A}_q = (A_{q1}, ..., A_{qn})$. Thus R_q is denoted as "$\mathbf{A}_q \Rightarrow$ Class C_q".

The first step to heuristic rule extraction is the discretization of the domain interval of each continuous attribute into antecedent intervals. Since we usually have no *a priori* information about an appropriate granularity of the discretization for each attribute, we simultaneously use multiple partitions with different granularities (i.e., from coarse partitions into a few intervals to fine partitions into many intervals). This is one characteristic feature of our approach to knowledge extraction. Since we simultaneously use multiple partitions with different granularities, we need no heuristic criteria to compare different granularities (i.e., to determine the number of intervals for each attribute). In computational experiments, we use five partitions into *K* intervals where $K = 1, 2, 3, 4, 5$ (see Fig. 1). It should be noted that $K = 1$ corresponds to the whole domain interval.

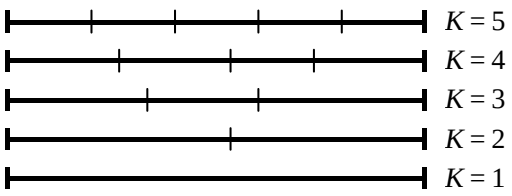


Fig. 1. Five partitions with different granularities used in our computational experiments.

As shown in Fig. 1, the whole domain interval is divided

into K intervals. To specify $(K-1)$ cutting points for each attribute, we use an optimal splitting method [18] based on the class entropy measure [19]:

$$H(A_1, ..., A_K) = -\sum_{j=1}^{K} \frac{|D_j|}{|D|} \sum_{h=1}^{M} \left(\frac{|D_{jh}|}{|D_j|} \cdot \log_2 \frac{|D_{jh}|}{|D_j|} \right), \quad (6)$$

where $(A_1, ..., A_K)$ is K intervals generated by the discretization of an attribute, D_j is the set of training patterns in the interval A_j, and D_{jh} is the set of training patterns from Class h in D_j. Using the optimal splitting method [18], we can efficiently find the optimal $(K-1)$ cutting points that minimize the class entropy measure in (6). In this manner, we can obtain multiple partitions for various values of K for each attribute.

When we use five partitions with $K = 1, 2, 3, 4, 5$ in Fig. 1, we have 15 antecedent intervals for each attribute. This means that we have 15^n combinations of the antecedent intervals for our n-dimensional classification problem. Such a combination corresponds to the antecedent part of each classification rule in (5).

The next step to heuristic rule extraction is the determination of the consequent class C_q and the rule weight CF_q for each combination $\mathbf{A}_q$ of the antecedent intervals. This is performed by calculating the confidence of the classification rule "$\mathbf{A}_q \Rightarrow$ Class h" for each class h (see [20] for the confidence measure). Let $D(\mathbf{A}_q)$ be the set of compatible training patterns with the antecedent part $\mathbf{A}_q$:

$$D(\mathbf{A}_q) = \{\mathbf{x}_p \mid x_{p1} \in A_{q1}, ..., x_{pn} \in A_{qn}\}. \quad (7)$$

When $D(\mathbf{A}_q)$ is empty, we do not generate any rule with the antecedent part $\mathbf{A}_q$.

The confidence of "$\mathbf{A}_q \Rightarrow$ Class h" is calculated as

$$c(\mathbf{A}_q \Rightarrow \text{Class } h) = \left| D(\mathbf{A}_q) \cap D(\text{Class } h) \right| \Big/ \left| D(\mathbf{A}_q) \right|, \quad h = 1, 2, ..., M. \quad (8)$$

The confidence of "$\mathbf{A}_q \Rightarrow$ Class h" in (8) is the ratio of compatible training patterns with $\mathbf{A}_q$ from Class h to all the compatible training patterns. Another measure called support has also been frequently used in the literature [20]. The support of "$\mathbf{A}_q \Rightarrow$ Class h" is calculated as

$$s(\mathbf{A}_q \Rightarrow \text{Class } h) = \left| D(\mathbf{A}_q) \cap D(\text{Class } h) \right| \Big/ \left| D \right|, \quad h = 1, 2, ..., M. \quad (9)$$

The consequent class C_q is specified as the class with the maximum confidence:

$$c(\mathbf{A}_q \Rightarrow \text{Class } C_q) = \max\{c(\mathbf{A}_q \Rightarrow \text{Class } h) \mid h = 1, 2, ..., M\}. \quad (10)$$

We have the same consequent class as in (10) even when we use the support in (9) instead of the confidence in (8). The consequent class C_q is the dominant class among the compatible training patterns with the antecedent part $\mathbf{A}_q$. As we have already mentioned, we do not generate any rule with the antecedent part $\mathbf{A}_q$ when there is no compatible training patterns with $\mathbf{A}_q$.

We specify the rule weight CF_q by the confidence as

$$CF_q = c(\mathbf{A}_q \Rightarrow \text{Class } C_q). \quad (11)$$

The rule weight CF_q is used in the classification phase of new patterns in the following manner. When a new pattern is to be classified by a rule-based classification system, first all compatible rules with the new pattern are found. Then a single winner rule with the largest rule weight is identified among the compatible rules. Finally the new pattern is classified as the consequent class of the winner rule.

Using the above-mentioned rule generation procedure, we can generate a huge number of classification rules by examining the 15^n combinations of the antecedent intervals. For high-dimensional classification problems, it may be impractical to examine all the 15^n combinations. Thus we only examine short rules with a small number of antecedent conditions. It should be noted that the antecedent interval corresponding to $K = 1$ in Fig. 1 is actually equivalent to a *"don't care"* condition. Thus all *don't care* conditions with the antecedent interval for $K = 1$ can be omitted. In this paper, the number of antecedent conditions excluding *don't care* conditions is referred to as the rule length. We only examine short rules of length $L_{\max}$ or less (e.g., $L_{\max} = 3$). This restriction on the rule length is to find simple classification rules with high interpretability.

We further decrease the number of rules by choosing only good rules with respect to a heuristic rule extraction criterion. That is, we choose a prespecified number of short rules for each class using a heuristic criterion. In our computational experiments, we use the following three heuristic criteria:

Support with the minimum confidence level: Each rule is evaluated based on its support value when its confidence value is larger than the prespecified minimum confidence level. This criterion never extracts unqualified rules whose confidence values are smaller than the minimum confidence level. Five minimum confidence levels (0.5, 0.6, 0.7, 0.8, 0.9) are examined in computational experiments.

Product of confidence and support: Each rule is evaluated based on the product of its confidence and support values.

Difference in support: Each rule is evaluated based on the difference between its support value and the total support value of the other rules with the same antecedent condition and different consequent classes. More specifically, the rule R_q with the antecedent condition $\mathbf{A}_q$ and the consequent class C_q is evaluated as follows:

$$f(R_q) = s(\mathbf{A}_q \Rightarrow \text{Class } C_q) - \sum_{\substack{h=1 \\ h \neq C_q}}^{M} s(\mathbf{A}_q \Rightarrow \text{Class } h) . \quad (12)$$

This is a modified version of a heuristic rule evaluation criterion used in an iterative fuzzy genetics-based machine learning algorithm called SLAVE [21].

We choose a prespecified number of promising rules with the largest values of each criterion in a greedy manner for each class. As we have already mentioned, only short rules of length $L_{\max}$ or less are examined in the heuristic rule extraction stage in order to find interpretable rules.

4. Multiobjective Rule Selection

Let us assume that we have N rules extracted from numerical data by heuristic rule extraction in the previous section (i.e., N/M rules for each class). Genetics-based multiobjective rule selection is used to find non-dominated rule sets from these N rules with respect to the accuracy and the complexity (i.e., to find non-dominated subsets of the N rules). The accuracy maximization of a rule set S is performed by minimizing the error rate on training patterns by S. We include the rejection rate into the error rate (i.e., training patterns with no compatible rules in S are counted among errors in this paper). On the other hand, we measure the complexity of the rule set S by the number of rules in S. Thus our rule selection problem is formulated as follows:

$$\text{Minimize } f_1(S) \text{ and } f_2(S), \quad (13)$$

where $f_1(S)$ is the error rate on training patterns by the rule set S and $f_2(S)$ is the number of rules in S.

Any subset S of the N candidate rules can be represented by a binary string of length N as

$$S = s_1 s_2 \cdots s_N , \quad (14)$$

where $s_j = 1$ and $s_j = 0$ mean that the j-th candidate rule is included in S and excluded from S, respectively. Such a binary string is handled as an individual in our EMO approach.

Since feasible solutions (i.e., subsets of the extracted N rules) are represented by binary strings, we can directly apply almost all EMO algorithms to our rule selection problem in (13) using standard crossover and mutation operations. In this paper, we use an elitist non-dominated sorting genetic algorithm (NSGA-II) of Deb et al. [22] because it is a state-of-the-art well-known EMO algorithm with high search ability.

The NSGA-II algorithm randomly generates a prespecified number of binary strings of length N (say N_{pop} strings) as an initial population. Each string is evaluated using Pareto ranking and a crowding measure. N_{pop} new strings are generated by genetic operations (i.e., selection, crossover, and mutation). The generated offspring population is merged with the parent population. The next population is constructed by choosing N_{pop} best strings from the merged population with $2 \times N_{\text{pop}}$ strings using Pareto ranking and a crowding measure as in the selection of parent strings. In this manner, the generation update is iterated until a prespecified stopping condition is satisfied. Non-dominated strings are chosen from the merged population at the final generation. These strings are presented to the human user as non-dominated rule sets. See Deb et al. [22] for details of the NSGA-II algorithm.

In the application of the NSGA-II algorithm to our rule selection problem, we use two problem-specific heuristic tricks in order to efficiently find small rule sets with high accuracy. One trick is biased mutation where a larger probability is assigned to the mutation from 1 to 0 than that from 0 to 1. This is for efficiently decreasing the number of rules in each rule set. The other trick is the removal of unnecessary rules, which is a kind of local search. Since we use the single winner-based method for classifying each pattern by the rule set S, some rules in S may be chosen as winner rules for no training patterns. We can remove these rules without degrading the first objective (i.e., the number of correctly classified training patterns). At the same time, the removal of unnecessary rules leads to the improvement in the other objectives. Thus we remove all rules that are not selected as winner rules for any training patterns from the rule set S. The removal of unnecessary rules is performed after the first objective is calculated and before the second and third objectives are calculated.

5. Computational Experiments

5.1. Settings of Computational Experiments

We use six data sets in Table 1: Wisconsin breast cancer (Breast W), diabetes (Diabetes), glass identification (Glass), Cleveland heart disease (Heart C), sonar (Sonar), and wine recognition (Wine) data sets. These six data sets are available from the UC Irvine machine learning repository (http://www.ics.uci.edu/~mlearn/). Data sets with missing values are marked by "*" in the third column of Table 1. Since we do not use incomplete patterns with missing values, the number of patterns in the third column does not include those patterns with missing values. All attributes are handled as continuous attributes in this paper.

We evaluate the performance of our EMO approach in comparison with the reported results on the same data sets in Elomaa & Rousu [18] where six variants of the C4.5 algorithm were examined. The performance of each variant was evaluated by ten independent executions (with different data partitions) of the whole ten-fold cross-validation (10CV) procedure (i.e., $10 \times 10\text{CV}$) in [18]. We show in the

last two columns of Table 1 the best and worst error rates on test patterns among the six variants reported in [18] for each data set.

Table 1. Data sets used in our computational experiments.

Data set	Attributes	Patterns	Classes	C4.5 in [18]	
				Best	Worst
Breast W	9	683*	2	5.1	6.0
Diabetes	8	768	2	25.0	27.2
Glass	9	214	6	27.3	32.2
Heart C	13	297*	5	46.3	47.9
Sonar	60	208	2	24.6	35.8
Wine	13	178	3	5.6	8.8

* Incomplete patterns with missing values are not included.

In this section, we examine the accuracy of extracted rules by heuristic rule extraction and non-dominated rule sets obtained by genetics-based multiobjective rule selection for training patterns and test patterns. When the classification accuracy on training patterns is discussed, all the given patterns (excluding incomplete patterns with missing values) are used in heuristic rule extraction and multiobjective rule selection. On the other hand, we use the 10CV procedure (i.e., 90% training patterns and 10% test patterns) when we examine the accuracy on test patterns.

We first explain computational experiments for examining the accuracy on training patterns where all the given patterns are used as training patterns. The accuracy of rules is evaluated on the same training patterns.

As in Fig. 1, we simultaneously use five partitions for each attribute. In the heuristic rule extraction stage, various specifications are used as the number of extracted rules for each class in order to examine the relation between the number of extracted rules and their accuracy. The number of extracted rules is specified as 1, 2, 3, 4, 5, 10, 20, 30, 40, 50, and 100. The three heuristic criteria in Section 3 are used in the heuristic rule extraction stage. When multiple rules have the same value of a heuristic criterion, those rules are randomly ordered (i.e., random tie break). As we have already mentioned, the five specifications of the minimum confidence level (i.e., 0.5, 0.6, 0.7, 0.8, 0.9) are examined in the support criterion with the minimum confidence level.

The maximum rule length $L_{\max}$ is specified as $L_{\max} = 2$ for the sonar data set and $L_{\max} = 3$ for the other data sets. That is, candidate rules of length 2 or less are examined for the sonar data set while those of length 3 or less are examined for the other data sets. We use such a different specification because only the sonar data set involves a large number of attributes (i.e., it has a huge number of possible combinations of antecedent intervals).

For each specification of the heuristic rule extraction criterion, average results are calculated over 20 runs for each data set in order to decrease the possible effect of the random tie break. Then we choose the heuristic rule extraction criterion from which the best average error rate on training patterns is obtained among various criteria in the case of 100 rules for each class. The chosen heuristic rule extraction criterion is used to extract candidate rules for the genetics-based multiobjective rule selection stage. It should be noted that a different criterion is chosen for each data set.

As candidate rules in multiobjective rule selection, we extract 300 rules for each class from training patterns. Thus $300M$ rules are used as candidate rules where M is the number of classes. The NSGA-II algorithm is applied to the extracted $300M$ rules using the following parameter values to find non-dominated rule sets with respect to the two objectives of our rule selection problem:

Population size: 200 strings,
Crossover probability: 0.8 (uniform crossover),
Biased mutation probabilities:
$p_{\mathrm{m}}(0 \to 1) = 1/300M$ and $p_{\mathrm{m}}(1 \to 0) = 0.1$,
Stopping condition: 5000 generations.

The extraction of $300M$ rules and the application of the NSGA-II algorithm are executed 20 times for each data set. Multiple non-dominated rule sets are obtained from each run of the NSGA-II algorithm. We calculate the error rate of each rule set on training patterns. Then the average error rate is calculated over rule sets with the same number of rules among 20 runs. Only when rule sets with the same number of rules are found in all the 20 runs, we report the average error rate for that number of rules in this section.

On the other hand, the 10CV procedure is used for examining the accuracy of rules on test patterns. First the 10CV procedure is iterated three times (i.e., $3 \times 10\mathrm{CV}$) using various criteria in heuristic rule extraction. The average error rates on test patterns are calculated over the three iterations of the 10CV procedure for various specifications of a heuristic rule extraction criterion and the number of extracted rules.

We choose the heuristic rule extraction criterion from which the best average error rate on test patterns is obtained among various criteria in the case of 100 rules for each class. The chosen heuristic rule extraction criterion is used to extract candidate rules for the genetics-based multiobjective rule selection stage as in the computational experiments for examining the accuracy on training patterns.

Using the chosen heuristic rule extraction criterion for each data set, the 10CV procedure is iterated three times (i.e., $3 \times 10\mathrm{CV}$). In each run of $3 \times 10\mathrm{CV}$ for each data set, 300 candidate rules are extracted for each class from training patterns. The NSGA-II algorithm is applied to the $300M$ candidate rules. The error rate on test patterns is calculated for each of the obtained non-dominated rule sets. The average error rate on test patterns is calculated for rule

sets with the same number of rules over 30 runs in 3×10CV . Only when rule sets with the same number of rules are obtained from all the 30 runs, we report the average error rate for that number of rules in this section.

5.2. Results on Training Patterns

In this subsection, we report experimental results on training patterns where average error rates are calculated on training patterns.

Wisconsin Breast Cancer Data: Experimental results by heuristic rule extraction are summarized in Table 2 where the average error rate over 20 runs is shown for each combination of a heuristic rule extraction criterion and the number of extracted rules for each class. The best error rate in each row is indicated by bold face. Since the best result for the case of 100 rules for each class is obtained by the support with the minimum confidence level 0.6 in Table 2 (see the last row), this heuristic rule extraction criterion is used in genetics-based multiobjective rule selection to extract 300 candidate rules for each class.

Table 2. Average error rates on training patterns of extracted rules by heuristic rule extraction (Breast W).

Rules for each class	Support with minimum confidence 0.5	0.6	0.7	0.8	0.9	Product	Diff.
1	8.78	8.78	8.78	8.78	7.91	**7.47**	7.50
2	8.71	8.71	8.71	7.03	**5.71**	6.59	6.59
3	7.03	7.03	7.03	5.56	6.15	**4.98**	6.44
4	5.83	5.71	5.71	5.71	**4.10**	4.39	4.74
5	**4.98**	**4.98**	**4.98**	5.27	5.71	6.00	5.23
10	**5.56**	**5.56**	**5.56**	5.83	6.37	6.73	6.73
20	**6.76**	6.84	6.92	7.04	8.82	8.78	8.78
30	10.40	10.40	10.40	10.40	9.66	9.46	**8.40**
40	**6.49**	6.56	6.52	6.60	7.38	7.61	7.61
50	8.17	8.18	8.13	8.20	8.20	**7.76**	7.78
100	7.22	**7.17**	7.22	7.26	7.47	7.47	7.47

In Fig. 2, we compare the average error rates between heuristic rule extraction and multiobjective rule selection. All the experimental results in Table 2 by heuristic rule extraction are depicted by closed circles whereas the average error rates of selected rules by multiobjective rule selection are shown by open circles. It should be noted that the horizontal axis in Fig. 2 is the total number of rules while the first column of Table 2 shows the number of rules for each class. From Fig. 2, we can see that smaller rule sets with lower error rates are found by multiobjective rule selection than heuristic rule extraction. That is, multiobjective rule selection improves the accuracy-complexity tradeoff curve in Fig. 2. We can observe a clear tradeoff structure between the average error rate and the number of rules from the experimental results by multiobjective rule selection (i.e., open circles in Fig. 2).

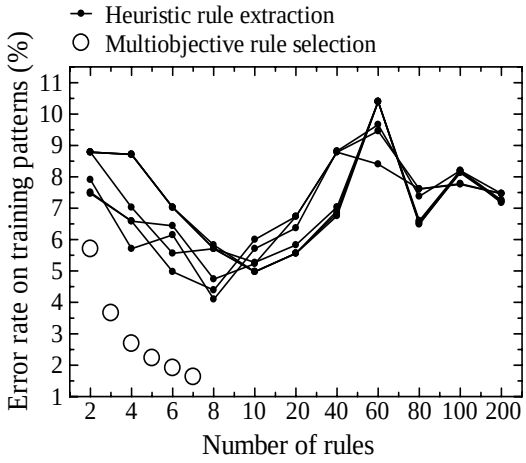


Fig. 2. Comparison between heuristic rule extraction and genetics-based multiobjective rule selection with respect to the average error rates on training patterns (Breast W).

Diabetes Data: Experimental results by heuristic rule extraction are summarized in Table 3. An interesting observation in Table 3 (and also in Table 2) is that the increase in the number of extracted rules does not always lead to the improvement in the average error rates. Another interesting observation from the comparison between Table 2 and Table 3 is that good results are obtained from different heuristic rule extraction criteria (e.g., see the sixth column with the label "0.9" of each table). That is, the choice of an appropriate criterion is problem-dependent.

Table 3. Average error rates on training patterns of extracted rules by heuristic rule extraction (Diabetes).

Rules for each class	Support with minimum confidence 0.5	0.6	0.7	0.8	0.9	Product	Diff.
1	32.68	36.59	30.99	36.33	62.63	**28.39**	29.43
2	31.38	32.16	**23.57**	36.33	59.90	23.96	28.39
3	31.77	**22.79**	23.28	36.33	58.33	23.18	28.39
4	32.68	**22.79**	23.44	36.33	49.87	**22.79**	24.09
5	33.85	23.05	23.44	30.86	49.78	**22.79**	23.44
10	29.69	27.60	**22.27**	30.73	47.33	26.95	22.46
20	31.51	24.74	**22.01**	28.36	38.63	24.48	22.14
30	30.83	24.55	**22.01**	26.43	33.52	22.79	22.87
40	28.65	**21.88**	22.79	24.35	32.29	22.66	22.79
50	27.47	**22.66**	22.79	24.48	30.08	**22.66**	23.31
100	26.95	23.70	23.70	**23.57**	26.56	23.96	23.78

In the same manner as Fig. 2, we compare heuristic rule extraction with multiobjective rule selection in Fig. 3.

Whereas multiobjective rule selection does not always outperform heuristic rule selection when the number of rule is small, it finds good rule sets with 8-20 rules. The relatively poor performance of multiobjective rule selection in the case of small rule sets with 2-6 rules is due to the use of candidate rules extracted by the support criterion with the minimum confidence level 0.8. As shown in Table 3, the performance of this criterion is not good when the number of extracted rules is small. Better results will be obtained from multiobjective rule selection if we use other criteria such as the product of confidence and support to extract candidate rules.

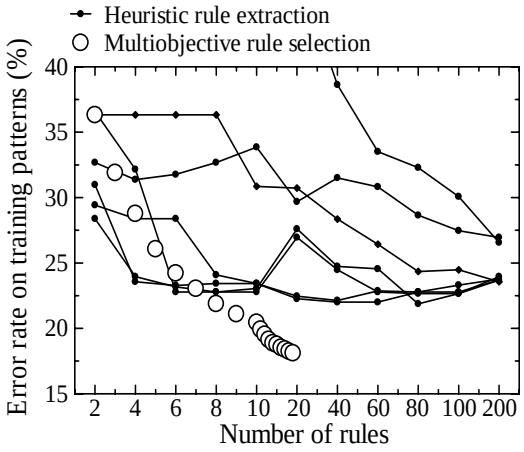


Fig. 3. Comparison between heuristic rule extraction and genetics-based multiobjective rule selection with respect to the average error rates on training patterns (Diabetes).

Glass Identification Data: In the same manner as Fig. 2 and Fig. 3, we compare heuristic rule extraction with multiobjective rule selection in Fig. 4. We can see from Fig. 4 that much better results are obtained from multiobjective rule selection than heuristic rule extraction. That is, much better tradeoffs between the accuracy and the complexity are obtained from multiobjective rule selection.

Cleveland Heart Disease Data: In the same manner as Figs. 2-4, experimental results are summarized in Fig. 5. Multiobjective rule extraction does not always outperform heuristic rule extraction when the number of rules is small. Multiobjective rule selection, however, finds much better rule sets than heuristic rule selection when the number of rules is large (e.g., 15-50 rules). We obtained a similar observation in Fig. 3 for the Diabetes data set.

Sonar Data: Experimental results are summarized in Fig. 6. We can see that much lower error rates are obtained by multiobjective rule selection than heuristic rule extraction for those rule sets with 6-15 rules.

Wine Data: Experimental results are summarized in Fig. 7. All the 20 runs of the NSGA-II algorithm find rule sets with only 5 rules that can correctly classify all the given patterns. On the other hand, 30 rules can not correctly classify all the given patterns in the case of heuristic rule extraction.

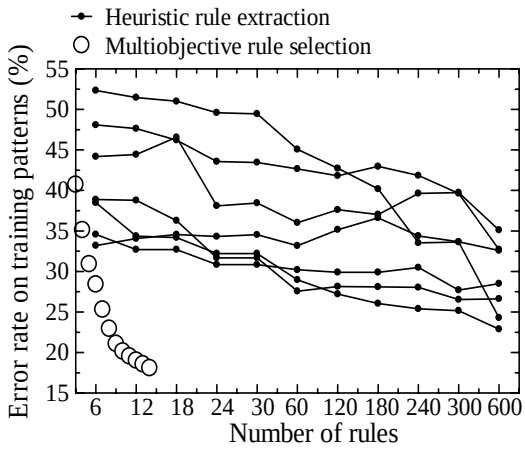


Fig. 4. Comparison between heuristic rule extraction and genetics-bases multiobjective rule selection with respect to the average error rates on training patterns (Glass).

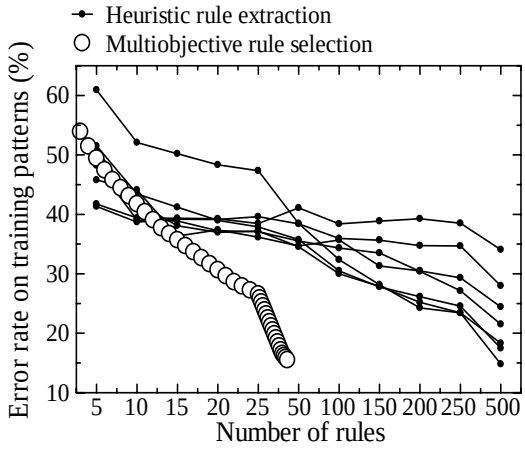


Fig. 5. Comparison between heuristic rule extraction and genetics-bases multiobjective rule selection with respect to the average error rates on training patterns (Heart C).

5.3. Results on Test Patterns

In this subsection, we report experimental results on test patterns where average error rates on test patterns are calculated by three iterations of the 10CV procedure. Heuristic rule extraction and genetics-based multiobjective rule selection are compared with each other. Our experimental results are also compared with the reported results of the C4.5 algorithm in Elomaa and Rousu [18].

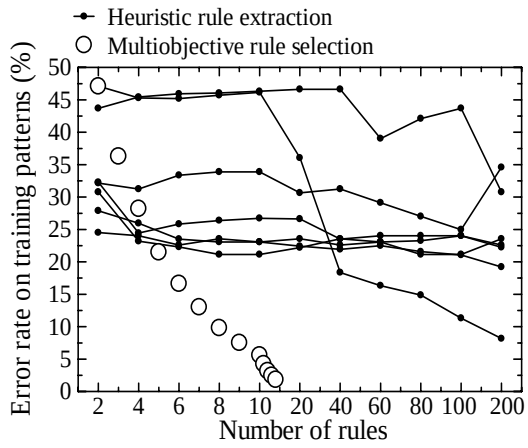


Fig. 6. Comparison between heuristic rule extraction and genetics-bases multiobjective rule selection with respect to the average error rates on training patterns (Sonar).

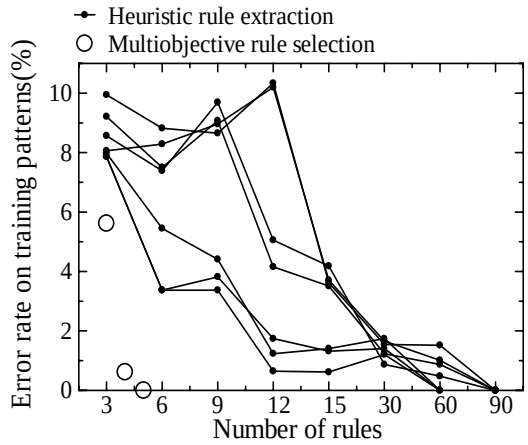


Fig. 7. Comparison between heuristic rule extraction and genetics-bases multiobjective rule selection with respect to the average error rates on training patterns (Wine).

Wisconsin Breast Cancer Data: Experimental results by heuristic rule extraction are summarized in Table 4 where the average error rate on test patterns over three iterations of the 10CV procedure is shown for each combination of a heuristic rule extraction criterion and the number of extracted rules for each class. The best error rate in each row is indicated by bold face. The best result for the case of 100 rules for each class is obtained by the difference criterion in support in Table 3. So we use this heuristic rule extraction criterion in genetics-based multiobjective rule selection to extract 300 candidate rules for each class from training patterns in each run of the 10CV procedure.

In Fig. 8, we compare heuristic rule extraction with multiobjective rule selection by depicting the average error rates on test patterns. Much better results are obtained by multiobjective rule selection. The dotted and dashed lines show the worst and best results of the C4.5 algorithm in Elomaa and Rousu [18], respectively. We can see that multiobjective rule selection outperforms the best result of the C4.5 algorithm with respect to the generalization ability.

Table 4. Average error rates on test patterns of extracted rules by heuristic rule extraction (Breast W).

Rules for each class	Support with minimum confidence 0.5	0.6	0.7	0.8	0.9	Product	Diff.
1	11.19	11.19	11.19	11.19	7.66	**7.42**	7.45
2	8.30	8.52	8.67	8.12	**5.77**	6.53	6.50
3	6.71	6.71	6.71	**5.55**	5.89	5.95	6.23
4	5.31	5.50	5.29	5.53	5.29	**5.17**	5.78
5	**4.97**	5.09	5.02	5.36	5.83	5.74	5.53
10	5.70	**5.69**	5.73	6.05	6.09	6.69	6.61
20	6.50	**6.45**	6.47	6.75	8.64	8.40	8.17
30	9.52	9.51	9.52	9.66	8.91	8.09	**7.54**
40	7.49	7.52	7.48	7.61	7.14	7.18	**7.13**
50	7.56	**7.55**	**7.55**	7.56	7.73	**7.55**	7.56
100	7.53	7.52	7.54	7.54	7.38	**7.09**	**7.09**

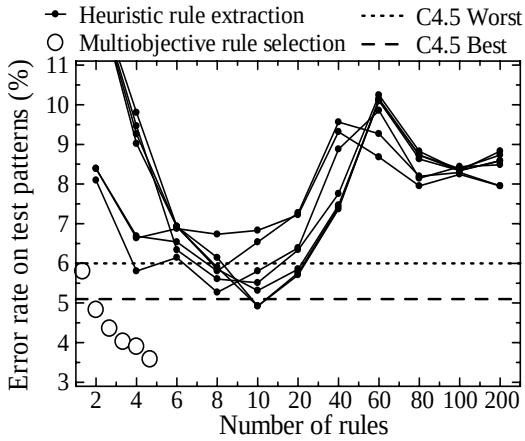


Fig. 8. Experimental results of the 10CV procedure (Breast W).

Diabetes Data: In the same manner as Fig. 8, we compare the average error rates on test patterns between heuristic rule extraction and multiobjective rule selection in Fig. 9. Experimental results show that multiobjective rule selection does not outperform heuristic rule extraction in terms of error rates on test patterns for the diabetes data set.

Glass Identification Data: Experimental results are summarized in Fig. 10. Fig. 10 clearly shows that better results are obtained from multiobjective rule selection than heuristic rule extraction.

Cleveland Heart Disease Data: Experimental results are summarized in Fig. 11 where multiobjective rule selection does not always outperform heuristic rule extraction.

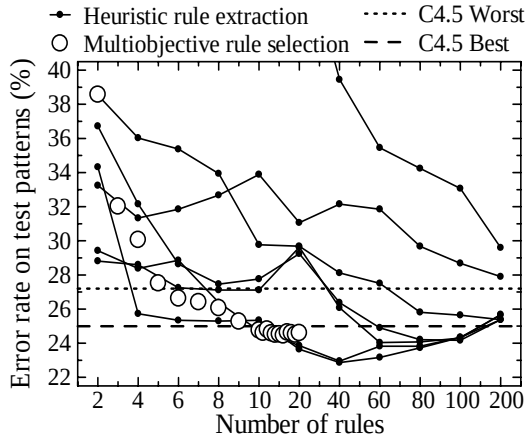


Fig. 9. Experimental results of the 10CV procedure (Diabetes).

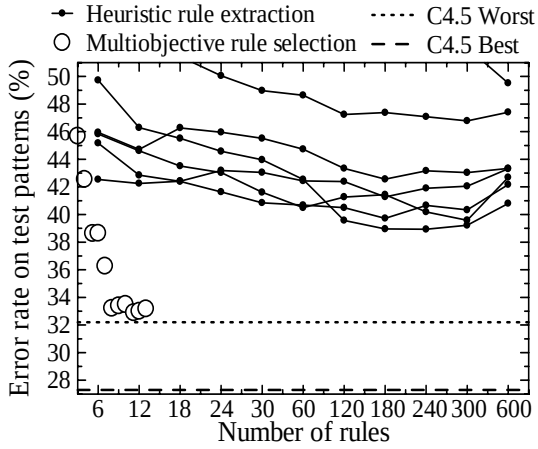


Fig. 10. Experimental results of the 10CV procedure (Glass).

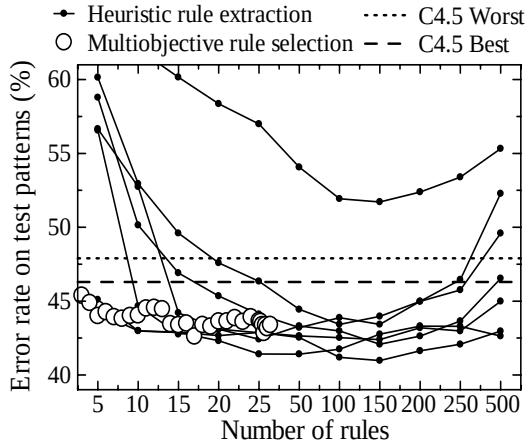


Fig. 11. Experimental results of the 10CV procedure (Heart C).

Sonar Data: Experimental results are summarized in Fig. 12. We can see from Fig. 12 that lower error rates are obtained by multiobjective rule selection than heuristic rule extraction when the number of rules is 9-12.

Wine Data: Experimental results are summarized in Fig. 13. We can see from Fig. 13 that very small rule sets of only 3 or 4 rules obtained by multiobjective rule selection have almost the same generalization ability as much larger rule sets of 30-150 rules obtained by heuristic rule extraction.

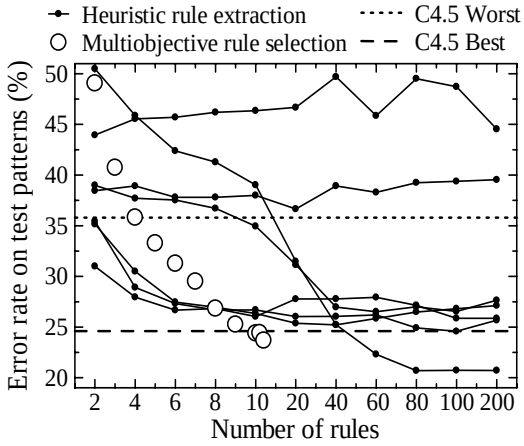


Fig. 12. Experimental results of the 10CV procedure (Sonar).

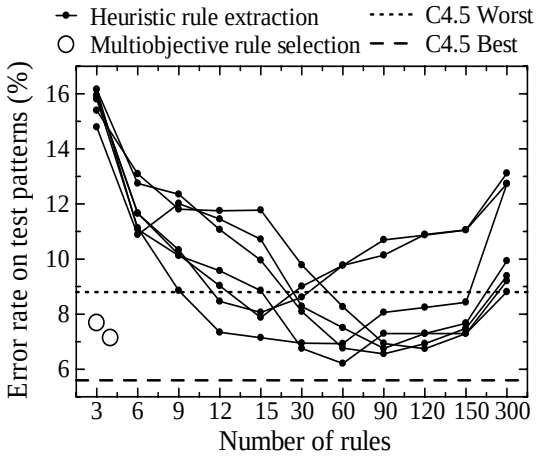


Fig. 13. Experimental results of the 10CV procedure (Wine).

6. Conclusions

We compared heuristic rule extraction with genetics-based multiobjective rule selection through computational experiments on six data sets from the UC Irvine machine learning repository. Experimental results showed that

multiobjective rule selection improved the accuracy-complexity tradeoff curve of heuristically extracted rules by searching for good combinations of a small number of rules. This improvement was observed in all experiments with respect to the accuracy on training patterns and most experiments with respect to the accuracy on test patterns. Except for the glass data set, multiobjective rule selection was comparable to or outperformed the C4.5 algorithm in terms of the generalization ability of obtained rule sets.

Since a large number of rules are usually obtained from data mining, multiobjective rule selection seems to be a promising direction to decrease the complexity of extracted rules. One difficulty of our EMO approach is its large computational load when it is applied to large data sets.

Acknowledgement

This work was partially supported by Japan Society for the Promotion of Science (JSPS) through Grand-in-Aid for Scientific Research (B): KAKENHI (17300075).

References

[1] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*, John Wiley & Sons, Chichester, 2001.

[2] C. A. Coello Coello, D. A. van Veldhuizen, and G. B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*, Kluwer Academic Publishers, Boston, 2002.

[3] M. A. Kupinski and M. A. Anastasio, "Multiobjective genetic optimization of diagnostic classifiers with implications for generating receiver operating characteristic curve," *IEEE Trans. on Medical Imaging*, vol. 18, no. 8, pp. 675-685, August 1999.

[4] J. Gonzalez, I. Rojas, J. Ortega, H. Pomares, F. J. Fernandez, and A. F. Diaz, "Multiobjective evolutionary optimization of the size, shape, and position parameters of radial basis function networks for function approximation," *IEEE Trans. on Neural Networks*, vol. 14, no. 6, pp. 1478-1495, November 2003.

[5] H. A. Abbass, "Speeding up back-propagation using multiobjective evolutionary algorithms," *Neural Computation*, vol. 15, no. 11, pp. 2705-2726, November 2003.

[6] H. A. Abbass, "Pareto neuro-evolution: Constructing ensemble of neural networks using multi-objective optimization," *Proc. of Congress on Evolutionary Computation*, pp. 2074-2080, Canberra, Australia, December 8-12, 2003.

[7] A. Chandra and X. Yao, "DIVACE: Diverse and accurate ensemble learning algorithm," *Lecture Notes in Computer Science 3177: Intelligent Data Engineering and Automated Learning - IDEAL 2004*, Springer, Berlin, pp 619-625, August 2004.

[8] A. Chandra and X. Yao, "Evolutionary framework for the construction of diverse hybrid ensemble," *Proc. of the 13th European Symposium on Artificial Neural Networks - ESANN 2005*, pp 253-258, Brugge, Belgium, April 27-29, 2005.

[9] H. Ishibuchi and T. Yamamoto, "Evolutionary multiobjective optimization for generating an ensemble of fuzzy rule-based classifiers," *Lecture Notes in Computer Science*, vol. 2723, *Genetic and Evolutionary Computation - GECCO 2003*, pp. 1077-1088, Springer, Berlin, July 2003.

[10] H. Ishibuchi, T. Murata, and I. B. Turksen, "Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems," *Fuzzy Sets and Systems*, vol. 89, no. 2, pp. 135-150, July 1997.

[11] H. Ishibuchi, T. Nakashima, and T. Murata, "Three-objective genetics-based machine learning for linguistic rule extraction," *Information Sciences*, vol. 136, no. 1-4, pp. 109-133, August 2001.

[12] O. Cordon, M. J. del Jesus, F. Herrera, L. Magdalena, and P. Villar, "A multiobjective genetic learning process for joint feature selection and granularity and contexts learning in fuzzy rule-based classification systems," in J. Casillas, O. Cordon, F. Herrera, and L. Magdalena (eds.), *Interpretability Issues in Fuzzy Modeling*, pp. 79-99, Springer, Berlin, 2003.

[13] F. Jimenez, A. F. Gomez-Skarmeta, G. Sanchez, H. Roubos, and R. Babuska, "Accurate, transparent and compact fuzzy models by multi-objective evolutionary algorithms," in J. Casillas, O. Cordon, F. Herrera, and L. Magdalena (eds.), *Interpretability Issues in Fuzzy Modeling*, pp. 431-451, Springer, Berlin, 2003.

[14] H. Ishibuchi and T. Yamamoto, "Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining," *Fuzzy Sets and Systems*, vol. 141, no. 1, pp. 59-88, January 2004.

[15] H. Ishibuchi, T. Nakashima, M. Nii, *Classification and Modeling with Linguistic Information Granules: Advanced Approaches to Linguistic Data Mining*, Springer, Berlin, November 2004.

[16] H. Wang, S. Kwong, Y. Jin, W. Wei, and K. F. Man, "Agent-based evolutionary approach for interpretable rule-based knowledge extraction," *IEEE Trans. on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, vol. 35, no. 2, pp. 143-155, May 2005.

[17] H. Wang, S. Kwong, Y. Jin, W. Wei, and K. F. Man, "Multi-objective hierarchical genetic algorithm for interpretable fuzzy rule-based knowledge extraction," *Fuzzy Sets and Systems*, vol. 149, no. 1, pp. 149-186, January 2005.

[18] T. Elomaa and J. Rousu, "General and efficient multisplitting of numerical attributes," *Machine Learning*, vol. 36, no. 3, pp. 201-244, September 1999.

[19] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[20] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo, "Fast discovery of association rules," in U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy (eds.), *Advances in Knowledge Discovery and Data Mining*, AAAI Press, Menlo Park, pp. 307-328, 1996.

[21] A. Gonzalez and R. Perez, "SLAVE: A genetic learning system based on an iterative approach," *IEEE Trans. on Fuzzy Systems*, vol. 7, no. 2, pp. 176-191, April 1999.

[22] K. Deb, A. Pratap, S. Agrawal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. on Evolutionary Computation*, vol. 6, no. 2, pp. 182-197, April 2002.

Intelligent Computation for Association Rule Mining

Hong-Cheu Liu
School of Economics and Information Systems
University of Wollongong
Wollongong, NSW 2522, Australia
hongcheu@uow.edu.au

John Zeleznikow
School of Information Systems
Victoria University
Melbourne, Vic. 8001 Australia
John.Zeleznikow@vu.edu.au

Abstract

Although there have been several encouraging attempts at developing SQL-based methods for data mining, simplicity and efficiency still remain significant impediments for further development. In this paper, we develop a fixpoint operator for computing frequent itemsets and demonstrate three query paradigm solutions for association rule mining that use the idea of least fixpoint computation. We consider the generate-and-test and the frequent-pattern growth approaches and propose an novel method to represent a frequent-pattern tree in an object-relational table and exploit a new join operator developed in the paper. The results of our research provide theoretical foundation for intelligent computation of association rules and could be useful for data mining query language design in the development of next generation of database management systems.

1 Introduction

Knowledge discovery from large databases has gained popularity and its importance is well recognized. Most efforts have focused on developing novel algorithms and data structures to aid efficient computation of such rules. Much work has also been performed on data cleaning, preparation and transformation. While research into such procedural computation of association rules has been extensive, little object-relational technology has yet been significantly exploited in data mining even though data is often stored in (object)-relational databases.

Several encouraging attempts at developing methods for mining object-relational data have been proposed. In principle, we can express and implement association rule mining in conventional SQL language (transaction databases) or XQuery (XML data). This approach was examined by [4, 8], for instance. However, the resulting SQL (XQuery) code is less than intuitive, unnecessarily long and complicated. There has no relational optimization yet been exploited in their proposals. It was pointed out in the literature that current SQL systems are unable to compete with ad-hoc file processing algorithms in general purpose data mining systems such as the well known Apriori algorithm and its variants [7]. However most data is stored in (object-) relational database systems, it is meaningful to investigate intelligent computational methods for association rule mining by exploiting object-relational technology.

The integration of data mining functionality with database management systems is an essential component of advanced data retrieval and analysis applications. The main idea is to combine relational query languages with data mining primitives in an overall framework capable of specifying data mining tasks as object-relational queries. Logic-based database languages provide a flexible model of representing, maintaining and utilizing high-level knowledge. This motivates us to study a logic-based framework and develop relational operators (fixpoint and fp-join operators) for intelligent data analysis.

In this paper, we focus on computational methods from three paradigms that have been developed for querying relational databases. We demonstrate three paradigm solutions for association rule mining that use the idea of least fixpoint computation. We consider the generate-and-test and the frequent-pattern growth approaches and propose an novel method to represent a frequent-pattern tree in an object-relational table and exploit a new join operator developed in the paper. The results of our research provide theoretical foundation for intelligent computation of association rules and could be useful for the development of next generation of database management systems with data mining functionality.

The presentation of the paper is organized as follows. We briefly review the basic concepts in Section 2. In Section 3 we present three paradigms for data mining query languages. We develop a fixpoint operator for computing frequent itemsets and show how it is expressed in the three different paradigms. We then present datalog implementation for frequent itemset mining by using the frequent-pattern

growth approach in Section 4. Finally we give a conclusion in Section 5.

2 Basic Concepts

In this section, we briefly review the basic concepts of association rule mining and data mining query languages.

2.1 Association Rules

While many forms of rule inductions are interesting, association rules were found to be appealing because of their simplicity and intuitiveness. In this paradigm, the rule mining process is divided into two distinct steps - discovering *large item sets* and generating rules.

The first work on mining association rules from large databases is the support-confidence framework established by Agrawal et al. [1]. Let $I = \{i_1, ...i_n\}$ be a set of item identifiers. An association rule is an implication of the form

$$X \Rightarrow Y, \text{ where } X, Y \subseteq I, \text{ and } X \cap Y = \emptyset$$

Association rules are characterized by two measures. The rule $A \Rightarrow B$ holds in the transaction set D with support s, where s is the percentage of transactions in D that contain $A \cup B$. This is taken to be the probability, $P(A \cup B)$. The rule $A \Rightarrow B$ has confidence c in the transaction set D if c is the percentage of transactions in D containing A that also contain B. This is taken to be the conditional probability, $P(B \mid A)$. That is,

$$\begin{aligned} support(A \Rightarrow B) &= P(A \cup B) \\ confidence(A \Rightarrow B) &= P(B \mid A) \end{aligned}$$

The task of mining association rules is to generate all association rules that satisfy two user-defined threshold values: a minimum support and a minimum confidence.

2.2 Data Mining Query Languages

A desired feature of data mining systems is the ability to support ad hoc and interactive data mining in order to facilitate flexible and effective knowledge discovery. Data mining query languages can be designed to support such a feature [3]. In particular, declarative query language support acts an important role in the next generation of Web database systems with data mining functionality. Query systems should provide mechanism of obtaining, maintaining, representing and utilizing high level knowledge in a unified framework. A knowledge discovery support environment should be an integrated mining and querying system capable of representing domain knowledge, extracting useful knowledge and organizing in ontologies [2].

We will introduce three query language paradigms for association rule mining in the next section. The first paradigm is logic based. It is a variant of the complex value calculus. The second paradigm provides basic algebraic operations for manipulating (object-)relations to construct mining results to queries. It uses an aggregation operator in addition to the basic relational operations. The third paradigm stems from logic programming. We use Datalog^{cv} with negation as a representative.

Designing a comprehensive data mining language is challenging because data mining covers a wide spectrum of tasks and each task has different requirements. In this paper we provide some theoretical foundations for relational computation of association rule mining.

3 Relational Computation for Association Rules

As most data is stored in (object-)relational databases, we can exploit object-relational technology to manage and mine interesting information from those data. In this section, we investigate relational computation methods and demonstrate three query paradigm solutions for association rule mining that use the idea of least fixpoint computation. The three query language paradigms, namely calculus, algebra and deductive rules, continue to play an important role in query languages of the next generation database systems.

3.1 Calculus+*Fixpoint*

We provide a noninflationary extension of the complex value calculus with recursion and aggregate operation. We define a fixpoint operator which allows the iteration of calculus formulas up to a fixpoint. In effect, this allows us to define frequent itemsets inductively using calculus formulas.

The motivation of defining a fixpoint operator in a data mining query language is to provide an alternative way to achieve association rule mining and to assist the development of a logic database language with data mining mechanisms for modeling extraction, representation and utilization of both induced and deduced knowledge.

Definition 1 *Let $\mathcal{S}^k(V)$ denote the set of all degree-k subset of V. For any two sets S and s, s is said to be a degree-k subset of S if $s \in \mathcal{P}(S)$ and $|s| = k$. $\mathcal{P}(S)$ denotes the powerset of S.*

The noninflationary version of the fixpoint operator is presented as follows. Consider association rule mining from object-relational data. Suppose that raw data is first preprocessed to transform to an object-relational database. Let $D = (x, y)$ be a nested table in the mapped object-relational

database. For example, $x = items$, $y = count$. *Items* is a set valued attribute. Let $S^k_x(D) = \{t \mid \exists u \in D, v = S^k(u[x]), t = (v,y)\}$. We develop a fixpoint operator for computing the frequent itemsets as follows. The relation J_n holding the frequent itemsets with support value greater than a threshold δ can be defined inductively using the following formulas:

$$\begin{aligned}\varphi(T,k) &= \sigma_{y\geq\delta}({}_x\mathcal{G}_{sum(y)}\mathcal{S}^k_x(D)(x,y)) \longrightarrow T(x,y), \\ &\quad \text{if } k = 1 \\ \varphi(T,k) &= T(x,y) \vee \sigma_{y\geq\delta}({}_x\mathcal{G}_{sum(y)}(\exists u, v\{T(u,v) \\ &\quad \wedge(\mathcal{S}^k_x(D)(x,y)) \wedge u \subset x \longrightarrow T(x,y)\})), \text{if } k \geq 1\end{aligned}$$

as follows: $J_0 = \emptyset$; $J_n = \varphi(J_{n-1}, n), n > 0$. Where $\mathcal{G}$ is the aggregation operator. Here $\varphi(J_{n-1}, n)$ denotes the result of evaluating $\varphi(T,k)$ when the value of T is J_{n-1} and the value of k is n. Note that, for each input database D, and the support threshold δ, the sequence $\{J_n\}_{n\geq 0}$ converges. That is, there exists some k for which $J_k = J_j$ for every $j > k$. Clearly, J_k holds the set of frequent itemsets of D. Thus the frequent itemsets can be defined as the limit of the forgoing sequence. Note that $J_k = \varphi(J_k, k+1)$, so J_k is also a fixpoint of $\varphi(T,k)$. The relation J_k thereby obtained is denoted by $\mu_T(\varphi(T,k))$. By definition, μ_T is an operator that produces a new nested relation (the fixpoint J_k) when applied to $\varphi(T,k)$.

In [6], the author proposed a fixpoint operator for computing frequent itemsets which is different from our definition. The least fixpoint operator of [6] is based on bottom-up computation approach which starts from the 'distance-1' subsets of the input database. We believe that our fixpoint operator is more appropriate as it can take advantage of anti-monotonicity property to do cross examination and make the computation method more efficient.

3.2 Algebra+*While*

Relational algebra is essentially a procedural language. The extension of the complex value algebra with recursion and incorporated with a *while* construct is consistent with the imperative paradigm and can express association rule mining queries.

We expect to have a function **sub** available in the next generation database systems that takes three arguments, two sets of values (Items) V_1 and V_2, and a natural number k such that $|V_2| \leq k \leq |V_1|$, and returns the degree-k subsets of the set V_1 that include V_2. We define a new join operator called *sub-join*.

Definition 2 *Let us consider two relations with the same schemes* $\{Item, Count\}$. $r \bowtie^{sub,k} s = \{t \mid \exists u \in r, v \in s$ *such that* $u[Item] \subseteq v[Item] \wedge \exists t'$ *such that* $(u[Item] \subseteq t' \subseteq v[Item] \wedge |t'| = k), t = <t', v[Count]>\}$

Here, we treat the result of $r \bowtie^{sub,k} s$ as multiset meaning, as it may produce two tuples of t' with the same support value. In the mining process we need to add all support values for each item.

Example 1 *Given two relations r and s, the result of* $r \bowtie^{sub,2} s$ *is shown as follows.*

r

Items	*Support*
$\{a\}$	0
$\{b,f\}$	0
$\{d,f\}$	0

s

Items	*Support*
$\{a,b,c\}$	3
$\{b,c,f\}$	4
$\{d,e\}$	2

$r \bowtie^{sub,2} s$

Items	*Support*
$\{a,b\}$	3
$\{a,c\}$	3
$\{b,f\}$	4

Figure 1. An example of sub-join

Given a database $D = (Item, Support)$ and support threshold δ, the following fixpoint algorithm computes frequent itemset of D.

Algorithm *fixpoint*
Input: An object-relational database D and support threshold δ
Output: The frequent itemsets of D

begin
$k := 1$
$T := \sigma_{Support\geq\delta}({}_{Item}\mathcal{G}_{sum(Support)}\mathcal{S}^k_{Item}(D)))$
$P := \emptyset$
$L := T$
While $(L - P) \neq \emptyset$ do
$P := L$
$k := k + 1$
$T := \sigma_{Support\geq\delta}({}_{Item}\mathcal{G}_{sum(Support)}(T \bowtie^{sub,k} (D))$
$L := L \cup T$
endwhile
end

Example 2 *Let's look at an example of fixpoint algorithm, based on the transaction table, D, of figure 2.*

The figure 3 shows the computation steps of the *fixpoint* algorithm. It is easy to show that the above algorithm computes the fixpoint defined in the Calculus + μ language and hence the result below follows.

Theorem 1 *For any object-relational database and minimum threshold δ, the fixpoint defined in the association rule mining expressed in Calculus + μ and the fixpoint algorithm compute the identical frequent itemsets.*

D

TID	*item_IDs*
T_1	$\{i_1, i_2, i_5\}$
T_2	$\{i_2, i_4\}$
T_3	$\{i_2, i_3\}$
T_4	$\{i_1, i_2, i_4\}$
T_5	$\{i_1, i_3\}$
T_6	$\{i_2, i_3\}$
T_7	$\{i_1, i_3\}$
T_8	$\{i_1, i_2, i_3, i_5\}$
T_9	$\{i_1, i_2, i_3\}$

Figure 2. Transaction data

*step*1

Items	*Support*
i_1	6
i_2	7
i_3	6
i_4	2
i_5	2

*step*2

Items	*Support*
i_1	6
i_2	7
i_3	6
i_4	2
i_5	2
$\{i_1, i_2\}$	4
$\{i_1, i_5\}$	2
$\{i_1, i_3\}$	4
$\{i_2, i_5\}$	2
$\{i_2, i_4\}$	2
$\{i_2, i_3\}$	4

*step*3

Items	*Support*
i_1	6
i_2	7
i_3	6
i_4	2
i_5	2
$\{i_1, i_2\}$	4
$\{i_1, i_5\}$	2
$\{i_1, i_3\}$	4
$\{i_2, i_5\}$	2
$\{i_2, i_4\}$	2
$\{i_2, i_3\}$	4
$\{i_1, i_2, i_5\}$	2
$\{i_1, i_2, i_3\}$	2

Figure 3. Computation steps of *fixpoint* algorithm

Proof Sketch. The fixpoint operator in Calculus defines frequent itemsets inductively by using the basic definitions of the used-defined threshold and the aggregation operator. The *fixpoint* algorithm computes the frequent itemsets by performing the sub-join operation. This yields the result which is equivalent to the fixpoint defined in the Calculus.□

TABLE Corporation = (Doc-id, Sub-doc)
TABLE Products = (Product-id, prod-name, Warranty, Composition, Distributor)
Warranty = (premium, country, w-period)
Composition = (Composition-id, c-name, Component)
Component = (part, quantity)
Distributor = (company, fee)
TABLE Parts = (Part-id, part-name, weight, Warranty, Source)
Warranty = (country, w-period)
Source = (company, cost)

Figure 4. Three mapped nested relational schemes.

Consider the object-relational database shown in figure 4. An association rule describes regularities of component parts contained in products. For example, the rule $\{p_1, p_2, p_3\} \Rightarrow \{p_4\}$ states that if a product containing parts $\{p_1, p_2\ p_3\}$ is likely to also contain part $\{p_4\}$. We can apply the above *fixpoint* Algorithm to find frequent patterns and then generate such association rules.

3.3 Datalog$^{cv,\neg}$

In this section, we present an operational semantics for association rule mining queries expressed in Datalog$^{cv,\neg}$ program from fixpoint theory. The formal syntax and semantics of Datalog$^{cv,\neg}$ are straightforward extensions of those for Datalog^{cv}. A Datalog $^{cv,\neg}$ rule is an expression of the form $A \leftarrow L_1, ..., L_n$, where A is an atom and each L_i is either an positive atom B_i or a negated atom $\neg B_i$. A Datalog$^{cv,\neg}$ program is a nonempty finite set of Datalog$^{cv,\neg}$ rules.

The challenge is to develop declarative means of computing association rules so that we can mine interesting information from object-relational databases. It is difficult to cast inherent procedurality into the declarativity of logic-based systems.

We present a Datalog program as shown in the figure 5 which can compute the frequent itemsets. The rule 1 generates the set of *1-itemset* from the input frequency table. The rule 2 selects the frequent *1-itemset* whose support is greater than the threshold. Let us assume that we have a *sub-join* relation, where $sub\_join(J, I, k, x)$ is interpreted as 'x is obtained by applying **sub** function to two operands J and I, i.e., $x = J \bowtie^{sub,k} I$. The rule 3 performs the *sub-join* operation on the table *large* generated in the rule 2 and the

1. $cand(J,C) \leftarrow freq(I,C), J \subset I, |J| = 1$
2. $large(J,C) \leftarrow cand(J,C), C > \delta$
3. $T(genid(),x,C_2) \leftarrow large(J,C_1), freq(I,C_2), k = max(|J|)+1, sub\_join(J,I,k,x)$
4. $cand(x,sum < C >) \leftarrow T(id,x,C)$
5. $large(x,y) \leftarrow cand(x,y), y > \delta$

Figure 5. Deductive association rule mining program

input frequency table.

Datalog system is of set semantics. In the above program, we treat T facts as multisets, i.e., bag semantics, by using system generated *id* to simulate multiset operation. The rule 4 counts the sum total of all supports corresponding to each candidate item set generated in table T so far. Finally, rule 5 computes the frequent itemsets by selecting the itemsets in the candidate set whose support is greater than the threshold. Suppose that n is the maximum cardinality of the itemsets in the frequency table. The above program is bounded by n.

We now show the program that defines *sub-join*:

$to\_join(J,I) \leftarrow A(J), B(I), J \subset I$
$sub\_join(J,I,k,x) \leftarrow to\_join(J,I), J \subset I, x \subset I, |x| = k$

Once the frequent itemset table has been generated, we can easily apply the following rule, which was proposed in [5], to produce all association rules.

$$rules(I,J-I,support,conf) \leftarrow large(I,C_I), large(J,C_J), support = C_J, conf = C_J/C_I, conf > \delta$$

In the final step, the above generated rules will be represented in the output object-relational table.

4 The frequent-pattern growth approach

The frequent-pattern growth mining process consists of two steps [3]:

- Construct a compact frequent-pattern tree which retains the itemset association information in less space.
- Mine the FP-tree to find all frequent patterns recursively.

When the database is large, it is unrealistic to construct a main memory-based FP-tree. An interesting alternative is to store a FP-tree in an object-relational table. See Figure 6. The mining of the FP-tree proceeds as follows. Start from each frequent 1-itemset (as an initial suffix pattern), perform

FP

part	*count*	pattern-base	
		pattern	count
p_5	2	$< p_2, p_1 >$	1
		$< p_2, p_1, p_3 >$	1
p_4	2	$< p_2, p_1 >$	1
		$< p_2 >$	1
p_3	6	...	...
p_1	6	...	...
p_2	7	...	...

Figure 6. An object-relational table representing FP-tree

mining by applying a special kind of join, called fp-join which is defined below, on the pattern base attribute in the FP-tree table.

Definition 3 *Given two arrays* $a =< a_1,...,a_m >$ *and* $b =< b_1,...,b_n >$, *where* $m \leq n$, *the join of two arrays is defined as* $a \bowtie b =$

- $< a_1,...,a_j >$, *if* $(a_1 = b_1,...,a_j = b_j)$ *and* $a_{j+1} \neq b_{j+1}$ *where* $j < m$; *or*
- $< a_1,...,a_m >$, *if* $a_1 = b_1,...,a_m = b_m$

For example, given two arrays $< i_2,i_1,i_5 >$ and $< i_2,i_1 >$, then $< i_2,i_1,i_5 > \bowtie < i_2,i_1 > = < i_2,i_1 >$. Then we define fp-join for the conditional pattern base attribute in the FP-tree table.

Definition 4 *Given two relations* u_1 *and* u_2 *with schemas* $\{< pattern : array, count : integer >\}$, *the fp-join of two relations is defined as follows:*

$$\begin{aligned} u_1 \bowtie^{fp} u_2 = \{t \mid \exists t_1 \in u_1 \text{ and } t_2 \in u_2 \text{ such that } & (t[pattern] = t_1[pattern] \bowtie t_2[pattern] \\ & \wedge t[count] = t_1[count] + t_2[count]) \\ & \vee (t \in u_1 \wedge (\forall t' \in u_2, t[pattern] \bowtie t'[pattern] = \emptyset) \\ & \vee (t \in u_2 \wedge (\forall t' \in u_1, t[pattern] \bowtie t'[pattern] = \emptyset) \end{aligned}$$

Example 3 *Suppose there is a relation* $R = \{<< i_2,i_1 >,2 >,<< i_2 >,2 >,<< i_1 >,2 >\}$. $R \bowtie^{fp} R = \{<< i_2,i_1 >,2 >,<< i_2 >,4 >,<< i_1 >,2 >\}$

We present a Datalog program as shown in the figure 7 which can compute the frequent itemsets by using the FP-growth approach. Similar to the candidate generate-and-test approach, the rules 1 and 2 produce the frequent 1-itemset L_1. The rule 3 produces the prefix patterns for each item (i.e., part). The rule 4 counts the number of patterns for each prefix. The nest operator is applied to create nested schema $FP\text{-}base(J,C,pattern\text{-}base < K,PC >)$ in rule 5.

1. $freq(parts, count < company >)$
   $\leftarrow D(company, parts)$
2. $L_1(J,C)$
   $\leftarrow freq(I,C)$, $J \subset I$, $|J| = 1$, $C > \delta$
3. $FP\text{-}pattern(J,C,T,K)$
   $\leftarrow L_1(J,C)$, $D(T,I)$, $J \subset I$, $K = I - J$
4. $FP\text{-}tree(J,C,K,count < T >)$
   $\leftarrow FP\text{-}pattern(J,C,T,K)$
5. $FP\text{-}base(J,C,pattern\text{-}base < K,PC >$
   $\leftarrow FP\text{-}tree(J,C,K,PC)$
6. $Cand\text{-}FP(J,C,CondFP < base,count >)$
   $\leftarrow FP\text{-}base(J,C,B)$, $B \bowtie^{fp} B = CondFP$
7. $FP(I,PC)$
   $\leftarrow Cand\text{-}FP(J,C,CondFP < K,C >)$,
   $Powerset(CondFP.K) \cup J = I$, $PC = C$, $C > \delta$
8. $FP(I,min(PC))$
   $\leftarrow FP(I,PC)$

Figure 7. The FP-growth approach to frequent pattern mining

The rule 6 applies the fp-join operator defined before to create the conditional pattern base, called $CondFP$. Finally, rules 7 and 8 form the frequent patterns by concatenating with the suffix pattern. In the program we use *Powerset* function which can be implemented in a sub-program and an aggregate function *min* to select the minimum support of the prefix patterns.

5 Conclusion

We have investigated data mining query languages from three paradigms that have been developed for querying relational databases. Three paradigm solutions for association rule mining that use the idea of least fixpoint computation have been demonstrated. In this paper, we have also shown that object-relational data can be mined in a declarative way so that extensive optimization task can be done in the underlying object-relational database engine. The main disadvantage of the deductive approach to data mining query languages is the concern of its performance. However, optimization techniques from deductive databases can be utilized and the most computationally intensive operations can be modularized. We have presented our preliminary ideas first and comprehensive query optimization and experimental work will be carried out at a later stage. The results of our research provide theoretical foundations for intelligent computation of association rules and could be useful for data mining query language design in the next generation of database systems.

References

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of ACM SIGMOD conference on management of data*, pages 207–216, 1993.

[2] F. Giannotti, G. Manco, and F. Turini. Towards a logic query language for data mining. *Lecture Notes in Artificial Intelligence*, 2682:76–94, 2004.

[3] J. Han. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 2000.

[4] H. M. Jamil. Ad hoc association rule mining as sql3 queries. In *Proceedings of international conference on data mining*, pages 609–612, 2001.

[5] H. M. Jamil. Mining first-order knowledge bases for association rules. In *Proceedings of 13th IEEE International conference on tools with Artificial intelligence*, 2001.

[6] H. M. Jamil. On the equivalence of top-down and bottom-up data mining in relational databases. In *Proceedings of the 4th international conference on data warehousing and knowledge discovery*, pages 41–50, 2001.

[7] D. Tsur, J. D. Ullman, S. Abiteboul, C. Clifton, R. Motwani, S. Nestorov, and A. Rosenthal. Query flocks: A generalization of association-rule mining. In *Proceedings of ACM SIGMOD*, pages 1–12, 1998.

[8] J. W.W.Wan and G. Dobbie. Mining association rule from xml data using xquery. In *Proceedings of the Fifth International Workshop on Web Information and Data Management*, 2003.

Building A Security-Aware Query Answering System Based On Hierarchical Data Masking

Seunghyun Im
[1] University of North Carolina at Charlotte
Department of Computer Science,
Charlotte, N.C. 28223, USA
sim@uncc.edu

Zbigniew W. Ras[1,2]
[2] Polish Academy of Sciences
Institute of Computer Science,
Ordona 21, 01-237 Warsaw, Poland
ras@uncc.edu

Agnieszka Dardzińska[3,1]
[3] Bialystok Technical University
Mathematics Dept.,
15-351 Bialystok, Poland
adardzin@uncc.edu

Abstract

Knowledge based query answering system takes advantage of data mining techniques to provide answers to user queries that involve incomplete or foreign attribute values. However, such process may cause a security issue when the system contains confidential data required to be protected. The confidential data as well as other values can be treated as missing or incomplete, and vulnerable to improper disclosure of their true values. To minimize such vulnerability, data transformation techniques are often utilized. In this paper, we present a method that exploits hierarchical structure of attributes to minimize the changes from the original information system while accommodating a given security requirement. In particular, our method replaces the existing data with more generalized ones in such a way that the value replacement cannot be used to predict the confidential data.

1 Introduction

Knowledge based Query Answering Systems (*QAS*) is to discover rules either locally or at remote sites (if system is distributed) and use these rules in a query answering process. There are two different situations within this objective. The first is when attributes are incomplete and we may need rules to approximate the incomplete values to answer to a query. The second is when users want to ask queries based on some attributes which are not listed in a local domain. Since these attributes are locally not available, we can only search for their definitions at remote sites and use them to approximate given queries [6] [8]. One way to design query answering systems more flexible is to apply a hierarchical structure to their attributes [12]. Unlike single-level attribute system, data collected with different granularity levels can be assigned into an information system with their semantic relations. For example, when the age of a person is recorded, the value can be *17* or *young* .

We can expect higher probability of answering user queries successfully by using a *QAS*. However, it may create a security problem when a set of data is confidential (e.g. age or salary) and protection is required. The exact value of confidential data can be concealed from an information system, for example, by replacing them with null values. However, users can send a query to *QAS* asking the confidential data as incomplete or foreign, and *QAS* returns the hidden data.

When we design a protection method against such improper disclosure, one approach is to transform a set of data to null values [5]. In other word, we completely hide a set of existing values that are used to predict the confidential data. Another approach, which we will discuss in this paper, is to mask the exact value by substituting it with more generalized values at higher level of a hierarchical attribute structure. For example, instead of showing a person is *17* years old we may show that she is *young* if disclosure of the value *young* does not compromise the privacy of the person. The advantage of the second approach that users will be able to acquire more explicit answers to their queries.

Clearly, we need to assume that a hierarchical attribute

structure is given to each attribute and they are part of the common ontology which is large and approximately the same among sites. They should come from the same world (e.g. medical information) and, consequently, rules generated from different sites are close in terms of their meanings. In addition, each site is forced to accept a new version of ontology if any change has been made. Also the hierarchical structure must be seen by users. Users have freedom of querying any level of values in the hierarchy.

The tradeoff between security risk and information availability is relatively clear. As the amount of hidden or rough data continues to grow the disclosure of confidential data drops. However, it is important to retain the original sources of information to the maximum extent possible to maintain the *QAS* to return more precise answers. In that respect, the method presented in this paper aims to minimize the amount of data replacement in terms of value granularity while making sure that *QAS* will not reveal the data below the safe levels of granularity.

2 Null Value Imputation in Distributed Query Answering System

2.1 Distributed Query Answering System and Chase

In real life, data are often collected and stored in information systems residing at many different locations, built independently, instead of collecting them and storing at only one single location. In such cases we talk about distributed (autonomous) information systems. It is very possible that an attribute is missing or hidden in one of them while it occurs in many others. Also, in one information system, an attribute might be partially hidden, while in other systems the same attribute is either complete or close to being complete. Assume that user submits a query to one of the information systems (called a client) which involves some hidden or non-local attributes. In such a case, network communication technology is used to get definitions of these unknown or hidden attributes from other information systems (called servers). All these new definitions form a knowledge base which can be used to chase both missing and hidden attributes at the client site.

In Figure 1, we present two consecutive states of a distributed information system consisting of S_1, S_2, S_3. In the first state, all values of all hidden attributes in all three information systems have to be identified. System S_1 sends request q_{S_1} to the other two information systems asking them for definitions of its hidden attributes. Similarly, system S_2 sends request q_{S_2} to the other two information systems asking them for definitions of its hidden attributes. Now, system S_3 sends request q_{S_3} to the other two information systems also asking them for definitions of its hidden attributes. Next, rules describing the requested definitions are extracted from each of these three information systems and sent to the systems which requested them. It means, the set $L(D_1)$ is sent to S_2 and S_3, the set $L(D_2)$ is sent to S_1 and S_3, and the set $L(D_3)$ is sent to S_1 and S_2.

The second state of a distributed information system, presented in Figure 1, shows all three information systems with the corresponding $L(D_i)$ sets, $i \in \{1,2,3\}$, all abbreviated as KB. Now, the $Chase$ algorithm [9] is run independently at each of our three sites. Resulting information systems are: $Chase(S_1)$, $Chase(S_2)$, and $Chase(S_3)$. Now, the whole process is recursively repeated. It means, both hidden and incomplete attributes in all three new information systems are identified again. Next, each of these three systems is sending requests to the other two systems asking for definitions of its either hidden or incomplete attributes and when these definitions are received, they are stored in the corresponding KB sets. Now, $Chase$ algorithm is run again at each of these three sites. The whole process is repeated till some fixed point is reached (no changes in attribute values assigned to objects are observed in all 3 systems). When this step is accomplished, a query containing some hidden attribute values can be submitted to any S_i, $i \in \{1,2,3\}$ and processed in a standard way.

2.2 Null Value Imputation Algorithm Chase

Let us examine in more detail the *Chase* algorithm. Assume that information is stored in an information system $S = (X, A, V)$, where X is a set of objects, A is a finite set of attributes, and V is a finite set of their values. In particular, we say that $S = (X, A, V)$ is an incomplete information system of type λ if the following three conditions hold:

- $a_S(x)$ is defined for any $x \in X$, $a \in A$,
- $(\forall x \in X)(\forall a \in A)[(a_S(x) = \{(a_i, p_i) : 1 \leq i \leq m\}) \rightarrow \sum_{i=1}^{m} p_i = 1]$,
- $(\forall x \in X)(\forall a \in A)[(a_S(x) = \{(a_i, p_i) : 1 \leq i \leq m\}) \rightarrow (\forall i)(p_i \geq \lambda)]$.

Incompleteness is understood by having a set of weighted attribute values as a value of an attribute. Now, suppose that $L(D) = \{(t \rightarrow v_c) \in D : c \in In(A)\}$ (called a knowledge-base) is a set of all rules extracted from $S = (X, A, V)$ by $ERID(S, \lambda_1, \lambda_2)$, where $In(A)$ is the set of incomplete attributes in S and λ_1, λ_2 are thresholds for minimum support and minimum confidence, correspondingly. *ERID* [11][2] is the algorithm for discovering rules from incomplete information systems, and used as a part of null value imputation algorithm *Chase* [8]. Assume now that a

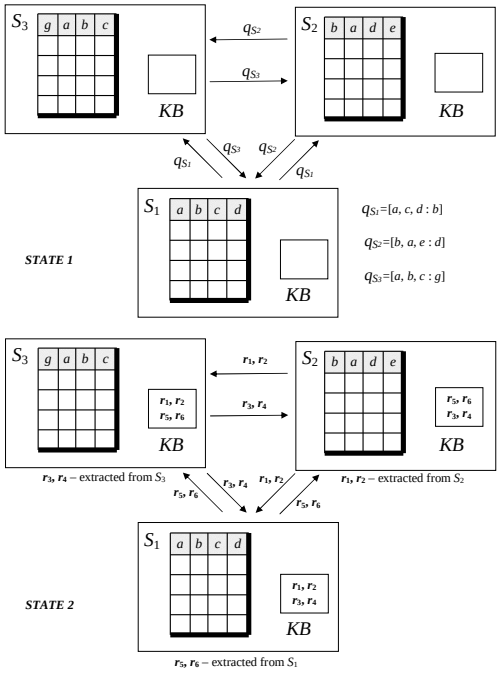


Figure 1. Global extraction and exchange of knowledge

query $q(B)$ is submitted to system $S = (X, A, V)$, where B is the set of all attributes used in $q(B)$ and that $A \cap B \neq \emptyset$. Attributes in $B - [A \cap B]$ are called either foreign or hidden in S. If S is a part of a distributed information system, definitions of such attributes can be extracted at remote sites for S [7].

The new definition replaced by the imputation algorithm is computed as following. Let $R_s(x_i) \subseteq L(D)$ be the set of rules that the conditional part of the rules is equal to the attribute values in $x_i \in S$, and d be a null value. Then there are three cases:

- $R_s(x_i) = \phi$ In this case, d cannot be predicted.
- $R_s(x) = \{r_1 = [t_1 \rightarrow d_1], r_2 = [t_2 \rightarrow d_1], ..., r_k = [t_k \rightarrow d_1]\}$ In this case, every rule implies a single decision attribute value, and $d = d_1$.
- $R_s(x_i) = \{r_1 = [t_1 \rightarrow d_1], r_2 = [t_2 \rightarrow d_2], ..., r_k = [t_k \rightarrow d_k]\}$ In this case, rules imply multiple decision values.

The confidence for the attribute value d for x_i driven by KB is defined as following [5]. Assuming that support and confidence of a rule r_i is $[s_i, c_i]$, and the product of the weight of each attribute value that matches to $a(x) \in t_i$ is $\prod p_{a(t_i)}$, for $i \leq k$.

$$conf(d') = \frac{\sum\{[\prod p_{a(t_i)}] \cdot s_i \cdot c_i : [d' = d_i]\}}{\sum\{[\prod p_{a(t_i)}] \cdot s_i \cdot c_i\}}, 1 \leq i \leq k$$

We replace the null value with d' when each $conf(d') > \lambda$.

2.3 Inconsistency

As we already pointed out, the knowledge base $L(D)$, contains rules extracted locally at the client site (information system queried by user) as well as rules extracted from information systems at its remote sites. Since rules are extracted from different information systems, inconsistencies in semantics, if any, have to be resolved before any query can be processed. There are two options:

- a knowledge base $L(D)$ at the client site is kept consistent (in this scenario all inconsistencies have to be resolved before rules are stored in the knowledge base)
- a knowledge base at the client site is inconsistent (values of the same attribute used in two rules extracted at different sites may be of different granularity levels and may have different semantics associated with them).

In general, we assume that the information stored in ontologies [4], [14] and, if needed, in inter-ontologies (if they are provided) is sufficient to resolve inconsistencies in semantics of all sites involved in $Chase$. Inconsistencies related to the confidence of conflicting rules stored in $L(D)$ do not have to be resolved at all (algorithm $Chase$ does not have such a requirement). The fact, that rules stored in $L(D)$ can be extracted at different sites and under different interpretations of incomplete values, is not pleasant assuming that we need to use them in $Chase$. In all such cases, following the same approach as in [7], rough semantics can be used for interpreting rules in $L(D)$.

One of the problems related to an incomplete information system $S = (X, A, V)$ is the freedom how new values are constructed to replace incomplete values in S, before any rule extraction process begins. This replacement of incomplete attribute values can be done either by $Chase$ or/and by a number of available statistical methods [3]. This implies that semantics of queries submitted to S and queries processed by the query answering system QAS based on $Chase$, may often differ. In such cases, following again the approach in [7], rough semantics can be used by QAS

to handle this problem. In this paper we assume that the semantic of attribute hierarchy is consistent among all the sites. For example, if $a \in A_i \cap A_j$, then only the granularity levels of a in S_i and S_j may differ but conceptually its meaning, both in S_i and S_j is the same.

2.4 Rule Extraction and Hierarchical Attribute

Before we discuss *Chase* and data security, we need to examine how rules are generated from S that is represented in hierarchical attribute structures. Assume that an information system $S = (X, A, V)$ is a partially incomplete information system of type λ, and a set of tree-like attribute hierarchy H_S is assigned to S where $h_a \in H_S$ represents all possible values of an attribute $a \in A$. If we denote a node in t_a as a_i, the set $\{a_{ik} : 1 \leq k \leq m\}$ contains all the children of a_i as shown in Figure 2.

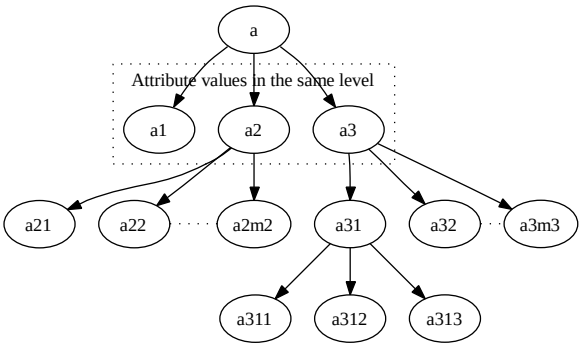


Figure 2. Hierarchical Attribute Structure

Many different combinations of attribute levels can be chosen for rule extraction. To extract rules at particular levels of interest in S, we need to transform attribute values before the rule extraction algorithm *ERID* is executed. In the following, we will use the term 'generalization' of $a(x)$ to refer to the transformation of $a(x)$ to a node value on the path from $a(x)$ to the root node in the hierarchy, and 'specification' to mean a transformation of $a(x)$ to a node value on the path to the leaf node. As defined, each attribute value in an incomplete information system is a value/weight pair $(a(x), p)$. When attribute values are transformed, the new value and weight are interpreted as the following,

- if $a(x)$ is specialized, it is replaced by a null value. This means that a parent node is considered as a null value for any child node.
- if $a(x)$ is generalized, it is replaced by $a_i \in t_a$ at the given level on the path. The weight of the new value is the sum of the children nodes. Intermediate nodes placed along the path, if exist, are computed in the same way. That is $p'_{a(x)} = \sum p_{a(x)ik}, (1 \leq k \leq m, p_{a(x)ik} \geq \lambda)$.

Clearly, the root node in each tree is an attribute name, and it is equivalent to a null value. Null value assigned to an object is interpreted as all possible values of an attribute with equal confidence assigned to all of them. Now, let L_H be the set of level of attributes to be used, λ_1 be the support, and λ_2 be the confidence value. *ERID* for hierarchical attributes is represented as *ERID-H*$(S, H_S, L_H, \lambda_1, \lambda_2)$.

3 Data Security and Chase Applicability

3.1 Problem of Data Confidentiality

To illustrate the data confidentiality problem, let's consider the following example. Suppose a local information system $S \in S_i$ for $i \in I$ operates in a distributed *QAS* as shown in Table 1. We assume that values in S are stored under hierarchical attribute structures H_S that is part of global ontology [7] for *QAS*. Now, an attribute d in S contains a set of confidential data, and S_d (see Table 2) has been built by replacing the exact values of d with values at a higher level in the hierarchical attributes structure that is considered as a secure level. User queries are now responded by S_d in replace of S. However, disclosure risk still remains because users may treat the confidential data as incomplete or foreign and contact remote sites in *QAS*. Clearly, $d_{S_d}(x)$ predicted by *Chase* can be equal to $d_S(x)$ for a number of objects [10]. Another vulnerability may be present in a similar way that users employ locally generated rules to predict attribute values contained in the rules that are extracted from remote sites [5]. For example, suppose we have generalized a set of additional attribute values to block the rules extracted from remote sites. However, there may be some local rules that reconstruct those additionally replaced values, and d can be predicted again.

3.2 Chase Applicability

Suppose that a knowledge base *KB* for S contains a set of rules. In order for the $Chase$ algorithm to be applicable to S, it has to satisfy the following conditions [7]:

- attribute value used in the decision part of a rule form *KB* has the granularity level either equal to or finer than the granularity level of the corresponding attribute in S.
- the granularity level of any attribute used in the classification part of a rule from *KB* should be either equal or softer than the granularity level of the corresponding attribute in S.

X	A	B	C	D	E	F
x_1	$(a_{[1,1,1]},\frac{2}{3})(a_{[2,1,1]},\frac{1}{3})$	$b_{[1,1,1]}$	$c_{[1,1,1]}$	$d_{[1,1,2]}$	$e_{[1,1,1]}$	$f_{[1,1,1]}$
x_2	$a_{[2,2,1]}$	$b_{[2,1,1]}$	$c_{[1,1,1]}$	$d_{[2,3,2]}$	$e_{[2,1,1]}$	$f_{[1,3,2]}$
x_3	$a_{[1,1,2]}$	$(b_{[1,1,1]},\frac{1}{2})(b_{[2,1,2]},\frac{1}{2})$	$c_{[1,1,1]}$	$d_{[1,3,2]}$	$e_{[1,1,1]}$	$f_{[1,3,1]}$
x_4	$a_{[2,2,1]}$	$b_{[2,1,2]}$	$c_{[2,1]}$	$(d_{[1,1,1]},\frac{2}{3})(d_{[2,1,1]},\frac{1}{3})$	$e_{[2,1,1]}$	$f_{[1,1,1]}$
x_5	$a_{[1,1,2]}$	$b_{[2,3,2]}$	$c_{[1,2,1]}$	$d_{[1,2]}$	$e_{[2,3,2]}$	$f_{[1,2,2]}$
x_6	$(a_{[1,1,1]},\frac{2}{3})(a_{[2,1,1]},\frac{1}{3})$	$b_{[2,2,1]}$	$c_{[1,3,1]}$	$d_{[1,1,2]}$	$e_{[2,3,2]}$	$f_{[1,1,2]}$
			.			
			.			
x_i	$a_{[1,2,1]}$	$b_{[2,1,1]}$	$c_{[1,1]}$	$d_{[1,2,2]}$	$e_{[1,1,1]}$	$f_{[1,1,1]}$

Table 1. Information System S

The set of values predicted by *Chase* may consist of horizontally and vertically different values. That means when two or more rules are supported by an object x_i, the predicted set can contain multiple values which granularity levels in H_S are the same (e.g. $\{d_{[1]}, d_{[2]}\}$), or different (e.g. $\{d_{[1]}, d_{[1,2]}\}$). We use the same confidence calculation function discussed in section 2.2 and allow all such cases to be valid predictions.

4 Method Description

We present an algorithm that protects values of a confidential attribute from $Chase$ algorithm. We continue to assume that attribute $d \in S$ contains confidential values, and all $d(x_i)$ are generalized to the values at the 2^{nd} level of H_S. The new information system S_d is shown in Table 2. The structure of each attribute hierarchy is same as that of $a \in A$ as illustrated in Figure 1 that has four levels and one to three nodes in each level. The rules in the knowledge base *KB* are summarized in table 3. For instance $r_1 = [c_{[1,1,1]} \rightarrow d_{[1,1,2]}]$ is an example of a rule belonging to *KB*. We use $\lambda = 0.3$ and $\tau = 0.8$. Based on above assumptions, we define the following sets:

- $\alpha(x)$, the set of attribute values used to describe x in S_d
- $\alpha(t)$, the set of attribute values used in t, where t is their conjunction
- $R(x) = \{(t \rightarrow d) : \alpha(t) \subseteq \alpha(x)\} \subseteq KB$, the set of rules in *KB* where the attribute values used in t are contained in $\alpha(x)$
- $\beta(x) = \cup\{\alpha(t) \cup \{d\} : [t \rightarrow d] \in R(x)\}$.

Our protection strategy consists of two phases: First, we identify the set of attribute values that are used for prediction for the confidential values. In the second phase, the values identified in the first phase are generalized. Before we discuss the phases in detail, we introduce the notion of chase closure and validity of prediction.

4.1 Chase Closure and Validity of Prediction

To find the minimum amount of values that are used for prediction for the confidential values, a bottom up approach has been adapted. We check the values that will remain unchanged starting from a singleton set containing attribute value a by using *chase closure* and increase the initial set size as much as possible. Chase closure is similar to transitive closure [1] except that if the weight of a predicted value is less than λ, the value is not added to the closure. For example, an object x_5 supports three rules, $\{r_6, r_7, r_8\} \in KB$ that predict $\{b_{[2,3,1]}, b_{[2,3,2]}\}$. In this case, $b_{[2,3,1]}$ is not included in the closure because $p'_{d_{[2,3,1]}} = \frac{45}{235} < 0.3$.

Identification method based on chase closure automatically rules out any superset of must-be-hidden values, and minimizes the computational cost. The justification of this is quite simple. Chase closure has the property that the superset of a set s also contains s. Clearly, if a set of attribute values predicts d_1, then the set must be hidden regardless of the presence/abscence of other attribute values.

When a confidential value has been predicted by *Chase*, the weight of the predicted value may be substantially different from that of the actual value. If this is the case, protection is not required because an adversary cannot have enough confidence in the confidential value. In order to determine whether a prediction is valid we define a measurement function and compare it to the threshold value τ. Suppose that the weight of an actual confidential value d_i is denoted as $p_{d_{[i]}}$ and weight of the predicted value is denoted as $p'_{d_{[i]}}$. The degree of validity associated with the prediction is defined as,

X	*A*	*B*	*C*	*D*	*E*	*F*
x_1	$(a_{[1,1,1]},\frac{2}{3})(a_{[2,1,1]},\frac{1}{3})$	$b_{[1,1,1]}$	$c_{[1,1,1]}$	$d_{[1,1]}$	$e_{[1,1,1]}$	$f_{[1,1,1]}$
x_2	$a_{[2,2,1]}$	$b_{[2,1,1]}$	$c_{[1,1,1]}$	$d_{[2,3]}$	$e_{[2,1,1]}$	$f_{[1,3,2]}$
x_3	$a_{[1,1,2]}$	$(b_{[1,1,1]},\frac{1}{2})(b_{[2,1,2]},\frac{1}{2})$	$c_{[1,1,1]}$	$d_{[1,3]}$	$e_{[1,1,1]}$	$f_{[1,3,1]}$
x_4	$a_{[2,2,1]}$	$b_{[2,1,2]}$	$c_{[2,1]}$	$(d_{[1,1]},\frac{2}{3})(d_{[2,1]},\frac{1}{3})$	$e_{[2,1,1]}$	$f_{[1,1,1]}$
x_5	$a_{[1,1,2]}$	$b_{[2,3,2]}$	$c_{[1,2,1]}$	$d_{[1,2]}$	$e_{[2,3,2]}$	$f_{[1,2,2]}$
x_6	$(a_{[1,1,1]},\frac{2}{3})(a_{[2,1,1]},\frac{1}{3})$	$b_{[2,2,1]}$	$c_{[1,3,1]}$	$d_{[1,1]}$	$e_{[2,3,2]}$	$f_{[1,1,2]}$
			.			
			.			
x_i	$a_{[1,2,1]}$	$b_{[2,1,1]}$	$c_{[1,1]}$	$d_{[1,2]}$	$e_{[1,1,1]}$	$f_{[1,1,1]}$

Table 2. Information System S_d

Rule	*A*	*B*	*C*	*D*	*E*	*F*	*Sup*	*Conf*
r_1			$c_{[1,1,1]}$	$(d_{[1,1,2]})$			100	0.9
r_2	$a_{[1,1,1]}$			$(d_{[1,1,2]})$		$f_{[1,1,1]}$	110	1
r_3		$b_{[1,1,1]}$			$(e_{[1,1,1]})$		120	1
r_4	$(a_{[1,1,1]})$				$e_{[1,1,1]}$		100	1
r_5	$(a_{[1,1,1]})$	$b_{[1,1,1]}$				$f_{[1,1,1]}$	90	1
r_6	$a_{[1,1,2]}$	$(b_{[2,3,1]})$					50	0.9
r_7		$(b_{[2,3,2]})$	$c_{[2,3,2]}$				100	1
r_8		$(b_{[2,3,2]})$			$e_{[2,3,2]}$	$f_{[1,2,2]}$	100	0.9
r_9	$a_{[1,1,1]}$		$c_{[1,1]}$	$(d_{[1,1,2]})$			100	0.9
r_{10}	$a_{[1,1,1]}$			$(d_{[1,1,2]})$		$f_{[1,1]}$	100	0.9

Table 3. Rules in KB

$$v = 1 - \frac{p_{d_{[i]}} - p'_{d_{[i]}}}{p_{d_{[i]}}}$$

and, we say d_i is secure against *Chase* if $v < \tau$. For example, assume that $\tau = 0.8$ for a confidential attribute d. A confidential attribute value is $\{(d_{[1]}, \frac{3}{4})), (d_{[2]}, \frac{1}{4})\}$ and it is predicted as $\{(d_{[1]}, \frac{1}{4})), (d_{[3]}, \frac{4}{4})\}$. In this case, $d_{[1]}$ is not considered as a valid prediction because $1 - (0.5/0.75) < 0.8$.

4.2 Phase One : Identification

We start phase one with a set $\beta(x)$ for the object x_1 which construction is supported by 5 rules $\{r_1, r_2, r_3, r_4, r_5\}$ from *KB*, and check the chase closure of each singleton subset $\delta(x)$ of that set. If the chase closure of $\delta(x)$ contains classified attribute value d_1, then $\delta(x)$ does not sustain, it is marked, and it is not considered in later steps. Otherwise, the set remains unmarked. In the second iteration of the algorithm, all two-element subsets of $\beta(x)$ built only from unmarked sets are considered. If the chase closure of any of these sets does not contain d_1, then such a set remains unmarked and it is used in the later steps of the algorithm. Otherwise, the set is getting marked. If either all sets in a currently executed iteration step are marked or we have reached the set $\beta(x)$, then the algorithm stops. Since only subsets of $\beta(x)$ are considered, the number of iterations will be usually not large.

So, in our example the following singleton sets are considered:

$\{a_{[1,1,1]}\}^+ = \{a_{[1,1,1]}\}$, unmarked
$\{b_{[1,1,1]}\}^+ = \{b_{[1,1,1]}, e_{[1,1,1]}, a_{[1,1,1]}\}$, unmarked
$\{c_{[1,1,1]}\}^+ = \{c_{[1,1,1]}, d_{[1,1,2]}\} \supseteq \{d_{[1,1,2]}\}, v_{[1,1,2]} = 1 > 0.8$ marked *
$\{e_{[1,1,1]}\}^+ = \{e_{[1,1,1]}, a_{[1,1,1]}\}$, unmarked
$\{f_{[1,1,1]}\}^+ = \{f_{[1,1,1]}\}$, unmarked.

Clearly, $\{c_{[1,1,1]}\}$ has to be concealed. The next step is to build terms of length 2 and determine which of the set can remain.

$\{a_{[1,1,1]}, b_{[1,1,1]}\}^+ = \{a_{[1,1,1]}, b_{[1,1,1]}\}$ unmarked
$\{a_{[1,1,1]}, e_{[1,1,1]}\}^+ = \{a_{[1,1,1]}, e_{[1,1,1]}\}$ unmarked
$\{a_{[1,1,1]}, f_{[1,1,1]}\}^+ = \{a_{[1,1,1]}, f_{[1,1,1]}, d_{[1,1,2]}\} \supseteq \{d_{[1,1,2]}\}$, $v_{[1,1,2]} = 1 > 0.8$ marked *
$\{b_{[1,1,1]}, e_{[1,1,1]}\}^+ = \{b_{[1,1,1]}, e_{[1,1,1]}\}$ unmarked

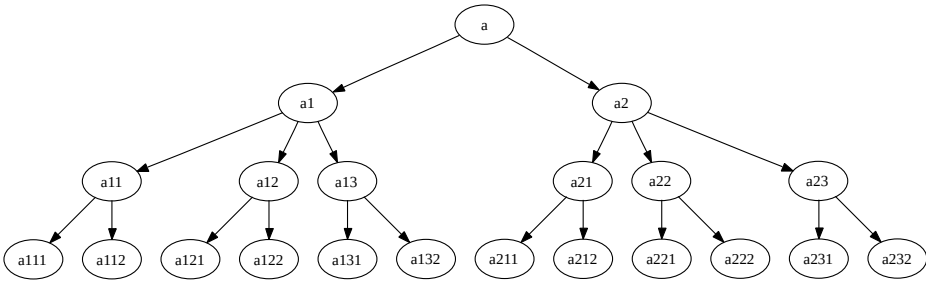


Figure 3. Attribute Hierarchy for $a \in S$

$\{b_{[1,1,1]}, f_{[1,1,1]}\}^+ = \{b_{[1,1,1]}, f_{[1,1,1]}, a_{[1,1,1]}, d_{[1,1,2]}\} \supseteq \{d_{[1,1,2]}\}, v_{[1,1,2]} = 1 > 0.8$ marked *
$\{e_{[1,1,1]}, f_{[1,1,1]}\}^+ = \{e_{[1,1,1]}, f_{[1,1,1]}\}$ unmarked

Now we build 3-element sets from previous sets that have not been marked.

$\{a_{[1,1,1]}, b_{[1,1,1]}, e_{[1,1,1]}\}^+ = \{a_{[1,1,1]}, b_{[1,1,1]}, e_{[1,1,1]}\}$ unmarked

We have $\gamma(x_1) = \{a_{[1,1,1]}, b_{[1,1,1]}, e_{[1,1,1]}\}$ as unmarked set that contains the maximum number of elements and does not have the chase closure containing d.

4.3 Phase Two : Generalization

Now, we need to generalize attribute values in $\epsilon(x_1) = \{\alpha(x_1) - \gamma(x_1)\}$. There are two issues. One is that we may have more than one $\gamma(x_1)$. If that is the case, we have to choose one of them. The other issue is that each of $\gamma(x_1)$ contains multiple attribute values that can be generalized. Several strategies can be considered to reduce the amount of generalization. One approach is to compute the minimum amount of generalization for each $\gamma(x_1)_i$ in terms of the number of layer transformations, and compare the result. However, it is difficult to say that the approach will produce the best result because (1) the granularity distance between a parent and a child can be different among $h_a \in H_S$, and (2) some attribute values are semantically more significant than others in QAS. In this case, a priority can be given by the system. In this paper, we assume that the amount of abstraction between a parent and a child is identical across all the attributes, and generalization is applied equally to each attribute value.

From phase one, we acquired the set $\epsilon(x_1) = \{c_{[1,1,1]}, f_{[1,1,1]}\}$. Our strategy in the second phase is that attribute values in $\epsilon(x_1) \subseteq (\epsilon(x_1) \cup \gamma(x_1))$ are generalized against the rules in *KB*, without modifying $\gamma(x_1)$, until the chase closure of the newly created set does not contain $d_{[1,1,2]}$. This strategy works because, as was exhibited in the first phase, if we replace all attribute values in $\epsilon(x_1)$ with null values, $d_{[1,1,1]}$ cannot be predicted. So, between a node connected to the root node (null value) and the current node, $d_{[1,1,1]}$ will not be predicted by *Chase*.

$\{\{c_{[1,1,1]}, f_{[1,1,1]}\}, \{a_{[1,1,1]}, b_{[1,1,1]}, e_{[1,1,1]}\}\}^+$
$= \{a_{[1,1,1]}, b_{[1,1,1]}, e_{[1,1,1]}, c_{[1,1,1]}, f_{[1,1,1]}, d_{[1,1,1]}\}$
$\supseteq \{d_{[1,1,2]}\}, v_{[1,1,2]} = 1$ marked*

We start from c by generalizing $c_{[1,1,1]}$ to $c_{[1,1]}$.

$\{\{c_{[1,1]}, f_{[1,1,1]}\}, \{a_{[1,1,1]}, b_{[1,1,1]}, e_{[1,1,1]}\}\}^+$
$= \{a_{[1,1,1]}, b_{[1,1,1]}, e_{[1,1,1]}, c_{[1,1]}, f_{[1,1,1]}, d_{[1,1,1]}, d_{[1,1,2]}\}$
$\supseteq \{d_{[1,1,2]}\}, v_{[1,1,2]} = 1$ marked*

$\{\{c_{[1,1]}, f_{[1,1]}\}, \{a_{[1,1,1]}, b_{[1,1,1]}, e_{[1,1,1]}\}\}^+$
$= \{a_{[1,1,1]}, b_{[1,1,1]}, e_{[1,1,1]}, c_{[1,1]}, f_{[1,1]}, d_{[1,1]}\}$
$\supseteq \{d_{[1,1,2]}\}, v_{[1,1,2]} = 1$ marked*

$\{\{c_{[1]}, f_{[1,1]}\}, \{a_{[1,1,1]}, b_{[1,1,1]}, e_{[1,1,1]}\}\}^+$
$= \{a_{[1,1,1]}, b_{[1,1,1]}, e_{[1,1,1]}, c_{[1]}, f_{[1,1]}, d_{[1,1]}\}$ unmarked

We have $\{c_{[1]}, f_{[1,1]}, a_{[1,1,1]}, b_{[1,1,1]}, e_{[1,1,1]}\}$ as a set that cannot be used to predict $d_{[1,1,2]}$. In a similar way, we compute the maximal sets for any object x_i.

5 Implementation and Conclusion

The algorithm was written in *PL/SQL* language, and executed in the Oracle 10g database running on Windows XP. A web based user interface was implemented using *HTML DB* as shown in Figure 4. The sample table that contains 3,000 objects with 7 attributes was randomly extracted from the census bureau database of the UCI Knowledge Discovery in Databases Archive [13]. A set of simple hierarchical attribute structure with a maximal depth of 3 was built on the basis of the interpretation of data. Each level of the hierarchical tree contains one to three nodes. The table was randomly partitioned into 3 tables that each have 1,000 tuples. One of these tables is called a client and the remaining 2 are called servers. We extracted 26 rules that describe the values of a confidential attribute from the servers, and 33

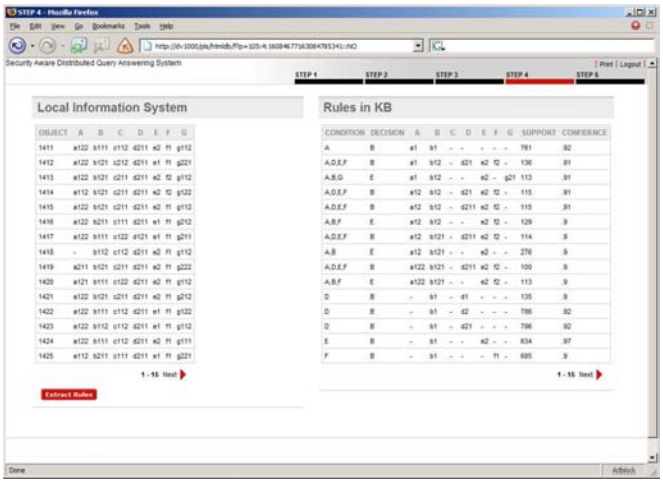


Figure 4. User Interface

local rules that are used to describe the values of remaining attributes.

To evaluate whether the use of hierarchical attributes is effective, we compared the total number of slots replaced by null values with that obtained by the same method without using hierarchical attribute structures. This was achieved by running the program without the execution of the generalization step. Without generalization step, 570 slots are replaced with null values. When the attribute hierarchy was applied, 187 slots are replaced by null values with 952 level transformations that include the number of transformations to null values. The result shows that 383 more attribute values can be shown to users. These values consist of values at the same level or values at higher levels. Clearly, the amount of improvements may not be the same when different set of rules and information systems are used. However, the use of hierarchical attributes will reduce the number of null values.

References

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison Wesley, 1995.

[2] A. Dardzińska and Z. Raś. On rules discovery from incomplete information systems. In *Proceedings of the ICDM'03 Workshop on Foundations and New Directions of Data Mining*, Melbourne, Florida, November 2003.

[3] P. Giudici. *Applied Data Mining, Statistical Methods for Business and Industry*. Wiley, 2003.

[4] N. Guarino and P. Giaretta. *Ontologies and knowledge bases, towards a terminological clarifi cation*, pages 25–32. IOS Press, 1995.

[5] S. Im and Z. Ras. Ensuring data security against knowledge discovery in distributed information system. In *Proceedings of the 10th International Conference on Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, Regina, Canada, September 2005.

[6] Z. Raś and A. Dardzińska. Collaborative query processing in dks controlled by reducts. In *Proceedings of Rough Sets and Current Trends in Computing 2002 Symposium*, Malvern, PA, October 2002.

[7] Z. Raś and A. Dardzińska. Ontology based distributed autonomous knowledge systems. *Information Systems International Journal*, 29(1):47–58, 2004.

[8] Z. Raś and A. Dardzińska. Query answering based on collaboration and chase. In *Proceedings of the 6th International Conference On Flexible Query Answering Systems*, Lyon, France, June 2004.

[9] Z. Raś and A. Dardzińska. Chase-2: Rule based chase algorithm for information systems of type lambda. In *Proceedings of the Second International Workshop on Active Mining*, Maebashi City, Japan, October 2005.

[10] Z. Raś and A. Dardzińska. *Data security and null value imputation in distributed information systems*, pages 133–146. Springer-Verlag, 2005.

[11] Z. Raś and A. Dardzińska. *Extracting Rules from Incomplete Decision Systems: System ERID*, pages 143–154. Springer, 2005.

[12] Z. Raś, A. Dardzińska, and O. Gurdal. Knowledge discovery based query answering in hierarchical information systems. In *Proceedings of the 10th International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing*, Regina, Canada, September 2005.

[13] C. B. S. Hettich and C. Merz. UCI repository of machine learning databases. 1998.

[14] G. Van Heijst, A. Schreiber, and B. Wielinga. Using explicit ontologies in kbs development. *International Journal of Human and Computer Studies*, 46(2/3), 1997.

Mining of Cell Assay Images Using Active Semi-Supervised Clustering *

Nicolas Cebron and Michael R. Berthold
ALTANA Chair for Bioinformatics and Information Mining
Department of Computer and Information Science
Konstanz University, Box M 712, 78457 Konstanz, Germany
{cebron,berthold}@inf.uni-konstanz.de

Abstract

Classifying large datasets without any a-priori information poses a problem especially in the field of bioinformatics. In this work, we explore the problem of classifying hundreds of thousands of cell assay images obtained by a high-throughput screening camera. The goal is to label a few selected examples by hand and to automatically label the rest of the images afterwards. We deal with three major requirements: first, the model should be easy to understand, second it should offer the possibility to be adjusted by a domain expert, and third the interaction with the user should be kept to a minimum. We propose a new active clustering scheme, based on an initial Fuzzy c-means clustering and Learning Vector Quantization. This scheme can initially cluster large datasets unsupervised and then allows for adjustment of the classification by the user. Furthermore, we introduce a framework for the classification of cell assay images based on this technique. Early experiments show promising results.

1. Introduction

The development of high-throughput imaging instruments, e. g. fluorescence microscope cameras, resulted in them becoming the major tool to study the effect of agents on different cell types. These devices are able to produce more than 50.000 images per day; up to now, cell images are classified by a biological expert who writes a script to analyze a cell assay. As the appearance of the cells in different assays change, the scripts must be adapted individually. Finding the relevant features to classify the cell types correctly can be difficult and time-consuming for the user.

The aim of our work is to design classifiers that are both able to learn the differences between cell types and are easy to interpret. As we are dealing with non-computer experts, we need models that can be grasped easily. We use the concept of clustering to reduce the complexity of our image dataset. Cluster analysis techniques have been widely used in the area of image database categorization.

Especially in our case, we have many single cell images with similar appearance that may nevertheless be categorized in different classes. Another case might be that the decision boundary between active and inactive is not reflected in the numerical data that is extracted from the cell image. Furthermore, the distribution of the different cell types in the whole image dataset is very likely to be biased. Therefore, the results of an automatic classification based on an unsupervised clustering may not be satisfactory, thus we need to adapt the clustering so that it reflects the desired classification of the user.

As we are dealing with a large amount of unlabeled data, the user should label only a small subset to train the classifier. Choosing randomly drawn examples from the dataset would render the classifier biased toward the underlying distribution of different kinds of cells in the cell assay images. Instead of picking redundant examples, it would be better to pick those that can "help" to train the classifier.

This is why we try to apply the concept of active learning to this task, where our learning algorithm has control over which parts of the input domain it receives information about from the user. This concept is very similar to the human form of learning, whereby problem domains are examined in an active manner.

To this date, research on approaches that combine clustering and active learning is sparse. In [1], a clustering of the dataset is obtained by first exploring the dataset with a *Farthest-First-Traversal* and providing *must-link* and *cannot-link* constraints. In the second *Consolidate*-phase, the initial neighborhoods are stabilized by picking new examples randomly from the dataset and again by providing constraints for a pair of data points.

In [7], an approach for active semi-supervised clustering for image database categorization is investigated. It in-

*This work was supported by the DFG Research Training Group GK-1042 "Explorative Analysis and Visualization of Large Information Spaces".

cludes a cost-factor for violating pairwise constraints in the objective function of the Fuzzy c-means algorithm. The active selection of constrains looks for samples at the border of the least-well defined cluster in the current iteration.

Our approach is similar to the latter one, although we do not update the cluster centers in each iteration but after an initial fuzzy c-means clustering.

In Section 2, we briefly recapitulate the fuzzy c-means algorithm, Section 3 describes our approach for the active selection of constraints, and the moving of the cluster prototypes. In Section 4, we introduce our prototype of a Cell Assay Image Mining System with its subcomponents for the image processing, before presenting first experimental results in Section 5.

2. Fuzzy c-means

The fuzzy c-means (FCM) algorithm [2] is a well-known unsupervised learning technique that can be used to reveal the underlying structure of the data. Fuzzy clustering allows each data point to belong to several clusters, with a degree of membership to each one.

Let $T = \vec{x}_i \ , i = 1, \ldots, m$ be a set of feature vectors for the data items to be clustered, $W = \vec{w}_k, k = 1, \ldots, c$ a set of c clusters. V is the matrix with coefficients where $v_{i,k}$ denotes the membership of $\vec{x}_i$ to cluster k. Given a distance function d, the fuzzy c-means algorithm iteratively minimizes the following objective function with respect to v and w:

$$J_m = \sum_{i=1}^{|T|} \sum_{k=1}^{c} v_{i,k}^m d(\vec{w}_k, \vec{x}_i)^2 \tag{1}$$

$m \in (1, \infty)$ is the fuzzification parameter and indicates how much the clusters are allowed to overlap each other. J_m is subject to minimization under the constraint

$$\forall i : \sum_{k=1}^{c} v_{i,k} = 1 \tag{2}$$

FCM is often used when there is no a-priori information available and thus can serve as an exploratory technique. A common problem is that the cluster structure does not necessarily correspond to the classes in the dataset. This is why the FCM algorithm is used only as a preprocessing technique. The fuzzy memberships $v_{i,k}$ prove useful for the selection of datapoints at the border between clusters as we will see in Section 3.1.

3. Active Learning

In order to improve the performance of the classification based on the initial, unsupervised clustering, we aim to guide the clustering process. Because we have no a-priori information about the class distribution in the dataset, we need to adapt the cluster prototypes so that they closer model the boundaries between the classes. This is done in two steps: 1. Labeling of a few "interesting" examples and 2. moving the prototypes according to these labels. These steps are discussed in detail in the following sections.

3.1. Selection of Constraints

We presume that we have access to a user (in our case the biological expert) who can give us labels for different data points. Another option would be that the user can define a constraint between a given pair (x_i, x_j) of data points. The assets and drawbacks of giving labels vs. constraints are discussed in [3].

We assume that the most informative data points lie between clusters that are not well separated from each other, so-called areas of possible confusion. This coincides with the findings and results in [6] and [13]. The prior data distribution plays an important role, [4] proposes to minimize the expected error of the learner:

$$\int_x E_T \left[(\widehat{y}(x; D) - y(x))^2 | x \right] P(x) dx \tag{3}$$

where E_T denotes the expectation over $P(y|x)$ and $\widehat{y}(x; D)$ the learner's output on input x given training set D. If we act on the assumption that the underlying structure found by the FCM algorithm already inheres an approximate categorization, we can select better examples by querying data points at the classification boundaries. That means we take into account the prior data distribution $P(x)$.

In order to have information about the general class label of the cluster itself, we let the user label the cluster centers using for each the nearest neighbor in the dataset, a technique known as "Cluster Mean selection" [6]. If more than one example per cluster shall be labeled, one can either split the corresponding cluster into subclusters, or alternatively select prototypes near to the one that was selected first.

To identify the data points that lie on the frontier between two clusters, we propose a new procedure that is easily applicable in the fuzzy setting. Rather than dynamically choosing one example for the labeling procedure, we focus on a selection technique that selects a whole batch of N samples to be labeled. Note that a data item $\vec{x}_i$ is considered as belonging to cluster k if $v_{i,k}$ is the highest among its membership values. If we consider the data points between two clusters, they must have an almost equal membership to both of them. Given a threshold $t \in [0, 1]$ the condition can be expressed as follows:

$$|v_{i,k} - v_{i,l}| < t, \quad v_{i,k}, v_{i,l} \geq \frac{1}{c} \tag{4}$$

In order to find N samples to query, we start with a high value for t and reduce it iteratively until we have obtained a set of $\leq N$ samples. Figure 1 shows an example of three clusters and the selected examples generated by this procedure.

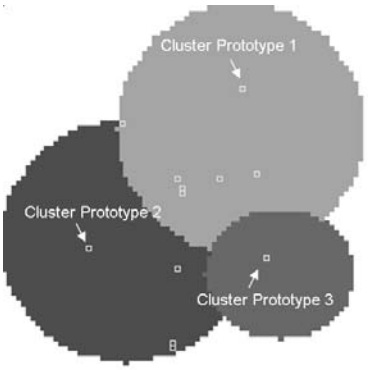


Figure 1. Three initial clusters and the selected examples

Having obtained the labels for the cluster centers and for a few confusing examples between them, we propose a new method in the next section to adapt the actual clusters to better match the underlying decision boundaries.

3.2. Learning Vector Quantization

The learning vector quantization algorithm [12] is a well-known competitive learning method. The outline of the algorithm is given in Algorithm 1. The LVQ algorithm needs the class information for all training examples. Since we can provide the class information only for a few selected examples, we need to optimize the selection of them.

3.3. Adaptive Active Learning

Our approach to optimize the LVQ algorithm includes the selective sampling scheme given in the previous section. It is dependent on an initial clustering and constitutes an extension to the LVQ-algorithm to choose the best examples. The total active clustering process is outlined in Algorithm 2.

Our initial prototypes in step 1 are the ones obtained from a fuzzy c-means clustering. Having the labels for each cluster prototype, we can select the next candidates for the query procedure along each border between two clusters. Which datapoints are selected depends on the chosen number N of examples we want to query. If N is small (approximately the number of clusters), some inter-cluster relationships will not be queried because the corresponding clusters

Algorithm 1 LVQ algorithm

1: Choose R initial prototypes for each class $m_1(k)$, $m_2(k), \ldots, m_R(k), k = 1, 2, \ldots, K$, e. g. by sampling R training points at random from each class.
2: Sample a training point $\vec{x}_i$ randomly (with replacement) and let $m_j(k)$ denote the closest prototype to $\vec{x}_i$. Let g_i denote the class label of $\vec{x}_i$ and g_j the class label of the prototype.
3: **if** $g_i = g_j$ **then** {that is they belong to the same class}
4: move the prototype toward the training point: $m_j(k) \leftarrow m_j(k) + \epsilon(\vec{x}_i - m_j(k))$, where ϵ is the learning rate.
5: **end if**
6: **if** $g_i \neq g_j$ **then** {that is they belong to different classes}
7: move the prototype away from the training point: $m_j(k) \leftarrow m_j(k) + \epsilon(\vec{x}_i - m_j(k))$
8: **end if**
9: Repeat step 2, decreasing the learning rate ϵ to zero with each iteration.

Algorithm 2 Adaptive Active Clustering Procedure

1: Perform the fuzzy c-means algorithm (unsupervised).
2: Select N training examples with the most similar membership to several clusters.
3: Ask the user for the labels of these samples.
4: Move the prototypes according to the label of the prototype and the samples.
5: Evaluation: If classification is better or matches the expected error then stop.
6: Repeat step 2, decreasing the learning rate ϵ to zero with each iteration.

are clearly separated in the feature space. As our approach focuses on separating classes given an initial "meaningful" clustering, this does not pose a problem. In each phase of our adaptation of the LVQ algorithm, we move the cluster centers according to a batch of N training points. For each point from the sample set we determine the two clusters that it lies in between and let the user label this point. We only update the two cluster prototypes that are involved at this point and leave the rest unchanged. We repeat the step of selecting a batch of training examples, then move the cluster centers for each point, decreasing the learning rate ϵ in each iteration. The process of selected examples and their influence on the prototypes can be seen clearly in Figure 2, where we assume that the majority of examples selected have the class label of cluster 2. The question is when to stop the movement of the cluster centers. The simulated annealing in the LVQ algorithm will stop the movement after a certain number of iterations. However, an acceptable solution may be found earlier, that is why a second stopping criterion is introduced.

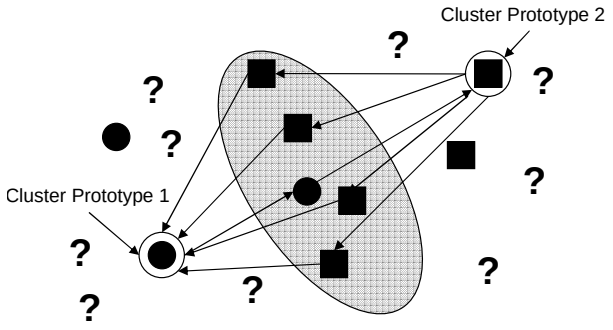


Figure 2. Points selected and their influence on the cluster prototypes

We make use of the already labeled examples to compare the previous to the newly obtained results. After the labels of the samples between cluster centers have been obtained, the cluster prototypes are moved. The new classification of the dataset is derived by assigning to each data point the class of its closest cluster prototype. By comparing the labels given by the user to the newly obtained labels from the classification, we can calculate the ratio of the number of correctly labeled samples to the number of falsely labeled examples.

4. Application: Cell Assay Image Mining

Our Cell Assay Image Mining System consists of three major elements: The segmentation module, the feature extraction module, and the classification element. Based on a modular workflow system, the user can choose and interact with the different modules and create a dataflow. This allows the user to enable and try out different settings interactively. Different modules for feature extraction or segmentation can be integrated. Figure 3 gives an overview of a typical workflow.

In the following sections, we focus on the different modules in more detail.

4.1. Segmentation

In order to calculate the features for each cell individually, the cell assay image has to be segmented. We prefer this approach in contrast to [10], because we need to identify interesting substructures in one image. The segmentation allows us to consider the cells separately in order to distinguish between different reactions of cells in the same image.

Unfortunately, the appearance of different cell types can vary dramatically. Therefore, different methods for segmentation have to be applied according to the different cell types. Work to segment and subdivide cells into the cell nucleus and cytoplasm based on seeded region growing is currently under progress. We follow the same assumption as in the approach from [11] that is, the nucleus can be detected more easily.

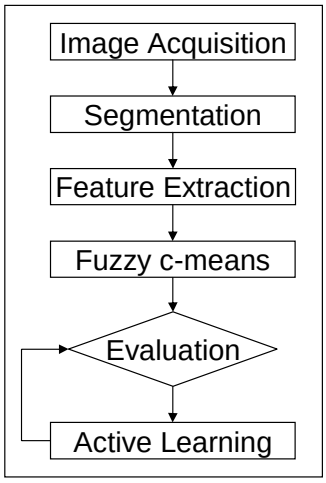


Figure 3. Workflow

4.2. Feature Extraction

The feature extraction module calculates features of a cell image based on the histogram (first order statistics) and based on the texture (second order statistics). The histogram features comprise the mean, variance, skewness, kurtosis, and entropy of the histogram.

The 14 texture features from Haralick [8] represent statistics of the co-occurrence matrix of the gray level image. Four co-occurrence matrices from horizontal, vertical, diagonal, and antidiagonal direction are averaged to achieve rotation invariance. These features provide information about the smoothness, contrast or randomness of the image - or more general statistics about the relative positions of the gray levels within the image.

4.3. Classification

The classification module comprises the initial fuzzy c-means clustering, the cluster evaluation and the Active Learning Module. As described in Section 3, we utilize the FCM to obtain our first set of cluster prototypes. The evaluation of the actual clustering can be based on several factors:

Cluster Validity Measures can give us information of the quality of the clustering [15]. We employ the within cluster variation and the between cluster variation as an

indicator. This descriptor can be useful for the initial selection of features. Naturally, the significance of this method decreases with the proceeding steps of labeling and adaptation of the cluster prototypes.

Visual Cluster Inspection allows the user to make a judgment of the clustering quality. Instead of presenting the numerical features, we select the corresponding image of the data tuple that is closest to the cluster prototype. We display the images with the highest membership to the actual cluster and the samples at the boundary between two clusters if they are in different classes. This approach is obviously prone to mistakes due to wrong human perception and should therefore be used only as an overview technique.

Evaluation in Adaptive Active Learning is performed as described in Section 3.3 where we judge the new classification based on the previously labelled examples. This method is the most suitable method to evaluate the quality of the classification. It also allows for the possibility to show the progress of the classification, so that the user can decide whether he wants to continue or not.

The classification of new images is obtained by classifying each individual cell within the given image. Each cell is assigned to a cluster and its corresponding class. The proportion of the distribution of the different classes is the decisive factor for classifying the whole image. If a clear majority decision can be made, the image is not considered further. Borderline cases with equal distributions of classes are sorted into a special container to be assessed manually by the biological expert. It becomes apparent that this approach allows for a rather high fault tolerance, as a human will have no objections to label a few images by hand rather than to risk a misclassification.

5. Experimental Results

As is the nature of the active learning problem, we do not have a large labeled dataset available to test the performance of our scheme. Therefore, we have created several artificial test sets to evaluate our classifier.

The first test set demonstrates the mode of action of our new active clustering scheme and is shown in Figure 1. It is a 3-dimensional artificial test set consisting of 4036 samples. The class label is indicated as the brightness. This dataset shows a typical problem where one class is underrepresented and the decision boundaries of the unsupervised clustering are not optimal because of the bias of the data. Figure 4 shows the class boundaries of the cluster prototypes and the decision boundaries. The optimum of 10 steps has been performed, selecting $N = 5$ examples

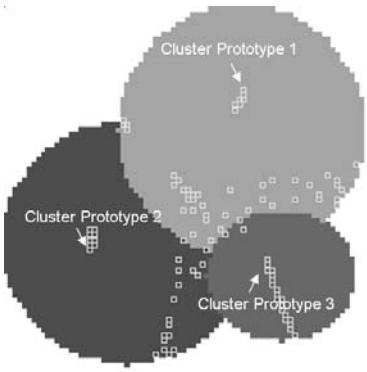


Figure 4. Movement of cluster prototypes over time and set of additionally labeled examples

in each iteration. As can clearly be seen, the active clustering scheme improves the positions of the cluster prototypes significantly to reduce the classification error. On the other hand, too many steps decrease the performance. An overview of the number of steps and the misclassification rate is shown in Table 1. As we can see, the bias of the clas-

# steps	Misclassification rate
0	16.13 %
5	11.89%
8	8.94%
9	8.57%
10	8.00%
11	8.45%
12	8.82%
15	10.16%
25	25.54%

Table 1. Number of steps vs. Misclassification Rate

sifier can be reduced and the decision boundaries between overlapping classes in the feature space can be optimized.

In our second test set, we added some noise to the clusters to test how distortion of class labels at the border influences the moving of the cluster prototypes. The effect on this dataset is shown in Figure 5. With the increasing noise at the border between clusters, the misclassification rate based on the initial unsupervised clustering naturally increases, too. With 10 steps and $N = 5$ labeled examples on the borders, we improved the misclassification rate from 17.95 % to 9.97 %. We increased the noise at the borders (see Figure 6) with the result that the misclassification rate improved from 27.77 % to 17.96 % . In this case, the initial clustering had two cluster prototypes that belonged

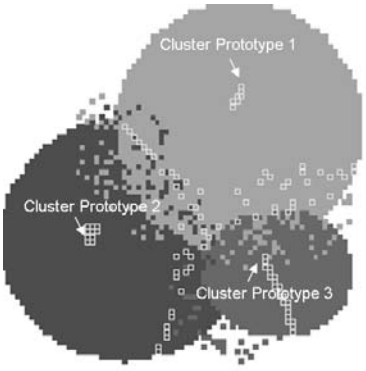


Figure 5. Movement of cluster prototypes over time and set of additionally labeled examples with noise

to the same class. This could be neutralized by requerying the labels for the cluster prototypes in each step of our adaptive active clustering procedure. This seems also useful to explore new classes in the dataset that have not yet been found if not enough clusters have been used. We observed this phenomenon in the Ionosphere-dataset from the UCI Repository [5], too. Having used four clusters to classify the data, only one class has been found from the initial fuzzy c-means clustering. The additional queries allowed to find the second class and due to this we were able to shift the classification accuracy. However, this is not the major goal of our algorithm.

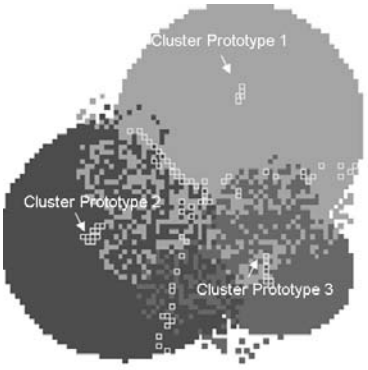


Figure 6. Movement of cluster prototypes over time and set of additionally labeled examples with more noise

6. Related Work

There have been a number of approaches to perform partial supervision in the clustering process. In the aforementioned works from [1] and [7], the objective function of the fuzzy c-means algorithm is extended by a cost factor for violating pairwise constraints. In the work of [14], labeled patterns are incorporated in the objective function of the Fuzzy ISODATA algorithm. All these approaches take a set of labeled patterns or constraints as input before the clustering process is started. These samples are selected randomly.

In [9], a very similar approach to our own work has been proposed that selects the points to query based on the Voronoi diagram that is induced by the reference vectors. The datapoints to query are selected from the set of Voronoi vertices with different strategies. However, our approach differs from all others in the way that the data is preclustered before supervision enhances the classification accuracy and the queries can be obtained in a fast and simple way.

7. Conlusions and Future Work

In this work, we have addressed the problem of classifying a large dataset when only a few labeled examples can be provided by the user. We have shown that the fuzzy c-means algorithm is well applicable for stable initial clustering and that it has the advantage that data points on the border can easily be detected by scanning through their memberships to the cluster prototypes. Based on the labels of the selected examples at the borders between clusters and the labeled cluster prototypes, we have proposed to move the cluster prototypes, similar to the Learning Vector Quantization (LVQ) method. We have shown that the misclassification rate can be improved, especially when the class distributions are skewed. We have discussed an application in the mining of cell assay images, where the data often inherits the aforementioned properties.

Future work needs to be done to optimize the number N of queries that are posed during the active clustering process. It would be desirable to pose just as many queries as necessary. Another important point are wrong classifications given by the user. Examples that contradict each other in terms of the model by their given labels could be requeried to be able to filter out wrong human classifications.

References

[1] S. Basu, A. Banerjee, and R. J. Mooney. Active semi-supervision for pairwise constrained clustering. *Proceedings of the SIAM International Conference on Data Mining (SDM-2004)*, 2004.

[2] J. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York, 1981.
[3] D. Cohn, R. Caruana, and A. McCallum. Semi-supervised clustering with user feedback, 2003.
[4] D. Cohn, Z. Ghahramani, and M. Jordan. Active learning with statistical models. *Advances in Neural Information Processing Systems*, 7:705–712, 1995.
[5] C. B. D.J. Newman, S. Hettich and C. Merz. UCI repository of machine learning databases, 1998.
[6] B. Gabrys and L. Petrakieva. Combining labelled and unlabelled data in the design of pattern classification systems. *International Journal of Approximate Reasoning*, 2004.
[7] N. Grira, M. Crucianu, and N. Boujemaa. Active semi-supervised clustering for image database categorization. *Content-Based Multimedia Indexing*, 2005.
[8] R. Haralick, K. Shanmugam, and I. Dinstein. Textural features for image classification. *IEEE Transactions on systems, man and cybernetics*, 1973.
[9] M. Hasenjäger and H. Ritter. Active learning with local models. *Neural Processing Letters*, 7:107–117, 1998.
[10] K. Huang and R. F. Murphy. Automated classification of subcellular patterns in multicell images without segmentation into single cells. *IEEE Intl Symp Biomed Imaging (ISBI)*, pages 1139–1142, 2004.
[11] T. R. Jones, P. Golland, and A. Carpenter. Voronoi-based segmentation on manifolds, 2005.
[12] T. Kohonen. *Self-Organizing Maps*. Springer Verlag, Heidelberg, 1995.
[13] H. Nguyen and A. Smeulders. Active learning using pre-clustering. *ICML*, 2004.
[14] W. Pedrycz and J. Waletzky. Fuzzy clustering with partial supervision. *IEEE Transactions on systems, man and cybernetics Part B: Cybernetics*, 27.
[15] M. Windham. Cluster validity for fuzzy clustering algorithms. *Fuzzy Sets and Systems*, 5:177–185, 1981.

An Adaptive Intrusion Detection System using a Data Mining Approach

Sujaa Rani Mohan, E.K. Park, Yijie Han
University of Missouri, Kansas City
{srmhv7 | ekpark | hanyij }@umkc.edu

Abstract

Weak data dependencies in large databases coupled with poorly written web based applications are a major cause for malicious transactions. The problem of security becomes especially acute when access roles are changed among users. Also the poorly maintained data base caches are a cause for added security leaks. We propose an adaptive Intrusion detection system to keep track of the varying data dependencies as and when the definitions for various access roles are changed. We use an association rule based approach to track all relevant data dependency rule sets for different access roles using a hierarchical structure. We then identify malicious transactions from the transaction logs in the database using the data dependency rule sets. These rule sets are continuously updated and stored in a repository. Our approach is shown to reduce data access bottlenecks, and ensures minimal manual intervention for maintaining a secure database.

1. Introduction.

Securing important data from malicious users has been a long time concern for many both in the industry as well as in research. Nowadays with web applications used to access large databases over a network the need for Intrusion Detection has become a dire necessity. When a Database is first designed, it is designed and architected based on initial requirements obtained from the users of the proposed database. There are few security leaks and the web application is well written for the predicted database transactions. Usually a Database system so designed will not be expected to be very susceptible to intrusion. It is a well known fact that no software can be made completely bug free. Loop holes are usually over looked due to poor testing or oversight as part of the database designer. Also the database may require some re-definitions of the database access roles based on the changes in the user's tasks. New tables and views may have to be added or old ones removed which causes changes in the data dependencies among tables. Such changes are generally invoked to make the database more feasible and this sometimes drastically affects the security level. A once secure database now becomes a perfect haven for malicious attacks. This is the core problem that we are trying to solve in our paper.

A database which is a part of a network or a host is usually monitored by the database administrator. He defines the various access roles for the users. These users hence have restricted access. With a number of users accessing the database with usually common queries there is a high bottleneck that arises. To prevent slow access speeds, the result sets of some queries are cached in a database cache. This reduces the access time but also opens the door for malicious unauthorized accesses. Usually large enterprises have a lot of sensitive data and hence in most cases such caches are poorly used. Malicious activity also arises when access roles are changed or the permissions for a user are changed. Another way to intrude into the database is by performing an unauthorized sequence of transactions. For example, a delete operation on a data item cannot occur without reading the item first.

Intrusion Detection Systems(IDS) have been developed to identify any unauthorized attempts or successful attacks on any type of monitored data or resources available as part of a network or host system. Most IDSs detect such malicious activity either at the transaction level or at the operating system(OS) level [1]. It is also shown that transaction level attacks take care of most OS level attacks [2], [3]. But there are many attacks which occur internal to the network such as by a user with lesser privileges accessing data that requires more access rights. Such attacks can be identified by analyzing the transaction logs. A transaction log contains all the transactions made on a database. More on Transaction logs are explained in later sections. By analyzing these logs most malicious activity can be identified. In a typical database accessed over a network there may be as many as one million transactions a day and any kind of computational analysis will prove to be costly and tedious. There have been a number of approaches to reduce the

time using different approaches, the most recent effective strategy being data mining [4].

Data mining is the analysis of data to establish relationships and identify hidden patterns of data which otherwise would go unnoticed. Even so existing approaches require analysis of millions of records. Our approach reduces the time to determine the sensitive data patterns from changing data access roles in the database, thereby identify any malicious activity and allow secure database caching at the network level.

Usually the database itself will not have security restrictions on individual data items. Access roles define the read/write/execute rights for each table in the database. Users generally query the database using transactions (or bulk transactions) through a web API. These transactions limit the number of valid queries that are allowed on a database. This is very common in web applications which use a large database over the internet. It is very easy to break into the database by writing malicious code to run illegal transactions and executing them on the database if the web application is not written carefully. In [5] the authors propose an effective means to locate and repair the damage on-the-fly for web based data intensive applications with reasonable (database) performance penalty.

In our approach we propose an adaptive IDS which defines a set of data dependency rule sets based on changing access roles which are maintained in a repository to identify such malicious transactions. The rest of the paper is organized as follows: Section 2 briefly discusses the current approaches and related work in this area of research. Section 3 outlines our concepts and assumptions used in our approach. We describe the various phases of our adaptive IDS in Section 4. We present an analysis of our approach to two well known database oriented web applications in Section 5. A brief conclusion and a discussion on our future work of applying an adaptive IDS to distributed databases using a multi agent framework is given in Section 6.

2. Current and related work.

Many researchers have dwelled into the field of database intrusion detection in databases using data mining. In [4], the author talks about a framework for continuously adapting the intrusion detection system for a computer environment as it is upgraded. The paper shows a number of data mining approaches to solve this problem and greatly discusses the results. Intrusion Detection has been approached using data mining by many researchers like [6]. In [5] , a multiphase damage confinement approach to ensure no damage is spread across the database after the detection is done. [7]'s paper uses a data dependency miner to identify correlations between data items and defines read and write sets for each data item. These rules are mined by scanning the database logs but it does not take into consideration the fact that the data dependency rules do not hold good for different access roles. We show in this paper that by applying the data dependency miner for transactions of each access role the, data dependencies will be more reliable. Also we show how the database transaction cache which caches frequently queried resultsets can be used more efficiently, by effectively preventing malicious accesses.

In [7], the authors argue that malicious writes are the major security threats and simple approach is given to identify malicious writes using data dependencies. The paper does not identify illegal reads. Our approach identifies all types of illegal accesses. Data mining is required to identify hidden data dependencies which might cause security threats and using triggers and stored procedures, we can only prevent expected loop holes. A possible solution is using dynamic stored procedures to maintain changing database data dependencies for intrusion detection but this an inflexible approach especially for a database which is accessed using a web based application.

Our approach deals with illegal transactions submitted to the DBMS via some mechanism using a user id with lower access rights using a web based application. Also it will prevent those intruders who bypassed the access control mechanism as the data dependency rule set will still track the valid sequence of reads and writes required for each transaction.

A change in the database models is inevitable and hence the security of the DB is at stake if the IDS model does not adapt itself to these changes. Also, adaptive database models have been shown to detect malicious transactions more effectively. In [4], the authors present a number of data mining approaches and their effectiveness for detecting malicious transactions.

3. Concepts and Assumptions

3.1. Database Schema, Access Roles and privileges.

A Database schema is a set of objects owned by a user. A user automatically has all object privileges for schema objects contained in his or her schema. A user can grant any object privilege on any schema object he or she owns to any other user or role. A privilege can be granted explicitly. For example, the privilege to insert records into table X can be explicitly granted to the user A. Alternatively a privilege can be granted to an access role which is a named group of privileges, and then the role can be granted to one or more users. For example, the privilege to insert records into a STUDENT table can be granted to the role named ADVISOR, which in turn can be granted to the users AdvisorB and ProfA.

In a relational Database such access roles are granted privileges in the form of a hierarchy with higher level access roles inheriting all privileges of lower level access roles. Our model works on this assumption. A sample hierarchical pattern is shown in Figure1

A← B ←C,D
Figure 1

Each Access Role type is assigned a weight which represents its level in the hierarchy. For example from Fig.1, C and D will be at level 1, B at level 2 , and A at level 3. In our paper we define each type of access roles with a set of read/write permissions for each table or view.

3.2. Transaction.

A transaction consists of a sequence of reads and writes of different data items from different tables in the database. A transaction Tk can be denoted as < o1(d1), o2(d2), o3(d1) > where di where i = 1….n is the different data items in the database, oi is an operation of the data and belongs to the set of reads and writes <r,w>. The data sequence for a transaction can be shown as Dk = {d1,d2}. Such a transaction will exist for the read or write operation for every data item in the database. i.e to perform a read or write operation there maybe some other read or write operation required on other data items prior to this read/write. This is called data dependency between data items and such validations are usually not made in by the database. The database only checks for foreign key, and primary key dependencies. Our data mining analyzer analyses all transactions' read/write sequences and formulates data dependency rule sets that are valid for different access roles. From these rule sets we show in the following sections how malicious transactions can be identified.

3.3. Transaction logs.

Each transaction requested by a user is logged in the transaction log table. In our approach a log entry consists of the following fields:

i. Transaction ID
ii. Ischange (has the record changed as a result of the transaction)
iii. isDelete (has the record been deleted)
iv. isRestore (is this entry a restore point if the record is lost)
v. which data items have been changed
vi. isMalicious (is this transaction malicious)
vii. SecurityDegree (the minimum access role level required for the user to initiate such a transaction)
viii. UserId (who initiated this transaction)
ix. AccessRole (what access roles does this user have on the database)

In our approach, these logs show the malicious transactions and the degree of maliciousness.

4. Our Approach.

The Apriori algorithm has become a well known standard for identifying patterns using association rules [8]. Its main disadvantage is that, if a pattern of length n is needed, then n passes are needed through the items. This can become a large overhead for our current application. [9] describes an efficient approach to perform incremental Mining of Frequent Sequence Patterns in Web logs. The approach we have used in this paper is a variation of the Apriori algorithm [10] which identifies frequent rule sets using a pattern repository in linear time [11]. The main advantage of this approach is the ease of updating the rule set and scaling. New frequent rule sets added to the repository can be used immediately.

However, our problem requires rule sets which identify all relevant patterns and not only the frequent ones. The rule set should be complete and all dependencies that may cause a security threat need to be identified. Also our problem requires an algorithm that can quickly adjust the rule sets rather than completely redefining them depending on changing items and item sets. We use a pattern repository similar to [11] to keep track of valid data dependencies. Also we modify the Apriori algorithm to reflect all relevant data dependencies.

Our approach will help build frequent rule sets by analyzing all data dependencies for different user access role types to identify malicious activity from database transaction logs.

Our approach can be subdivided into the following phases.

4.1. Phase 1.

a) Initial Database Scan: This phase involves identifying the different tables, views, data items, primary key and foreign key constraints in the database.
b) Identifying Access Roles and their hierarchy: Based on the read/write/execute rights on different views and tables that each access role has the access roles are classified in a hierarchy with the access role having all rights (for example the database administrator) being at the top of the hierarchy. For example, Administrator ← Professor ← Student.
c) Let T be a universal Transaction set containing all read/write sequences for each transaction tx where x = 1….m , the total number of all transactions for the database based on all access role definitions. Let AR with the set of all access roles ranging from AR1 to ARn where ARn is the access role with all rights on the database.
d) Ordering Data items for rule set formulation: The ordering can be performed by performing a count on the number of times a data item occurs in all transactions in

the universal transaction set T and then numbering the data items by the descending order of their counts. Also the primary key and foreign key constraints of the data items in a view or table should be taken into care while ordering as this ordering affects the order in which data items are selected while formulating the data dependency rule sets for the various access roles. This order will affect the formulation of rule sets involving data items that have cross dependencies between tables. This phase can only be partially automated and highly depends on the way the database is defined. For larger databases care should be taken not to assign cyclic dependencies between tables (for example, a write on Item x depends on a write in Item y and vice versa).

4.2. Phase 2.

a) We follow a reverse hierarchical order for rule set formulation. i.e. the rule sets for the access role with the least permission (lowest level in the hierarchy) are identified first followed by one of its siblings at the same level in the hierarchy as its access role definition will be different. Once all the rule sets for the lowest level in the hierarchy are identified, the rule sets for an access role in the next higher level are formulated and so on. All rule sets formulated for an access role are directly inherited by its parent and these rule sets are no longer reformulated.

b) For each access role chosen as per the reverse hierarchical order,

i. Let TAi{} be a subset of T containing the read/write operational sequences for all transactions for the access role ARi. Let each transaction be denoted as tz where z=1 to the number of transactions in TAi. Each transaction is a sequence of reads and writes, ordered from left to right in the order in which they need to occur. Let DAi be the set which maintains the counts for each unique aggregating rule set. The Access role ARi has operation-item set Ai{} which is initially empty.

ii. First pass: Determine all rule sets of length 1 (Length 1 means a single operation which may be a read or write on a data item). This is done by scanning the first operation in each transaction in TAi{} from left to right. Assign a count to the number of times each sequence of operation occurs in all TAi{} and store that count in DAi. These rule sets are now added to Ai{}.

Pass 2 to Pass MaxLengthTransaction of TAi{} or until bigger rule sets cannot be formed (Pass j): All Transactions which do not contain a rule set in Ai{} are removed from TAi{}. Determine all rule sets of length j. Repeat the same procedure as in Pass 1 and assign the count of the number of times the operation o(di) occurs with the existing rule set and store it in DAi.

By setting a support level we can remove infrequent rule sets from being formed at each pass by tracking the DAi counts for each aggregating rule set, but it must be noted that the purpose of this approach is to keep track of all relevant rule sets as opposed to frequent rule sets since the main aim is to identify all malicious transactions. A sample result set aggregation for a few passes for transactions starting with o(d1) is shown in Table1.

Table1. Sample data dependency rule set formulation.

Pass	Aggregating rule sets	DAi counts Cardinality of TAi = 10
1	o(d1)	8
2	o(d1) o(d2)	8
3	o(d1) o(d2) o(d3)	4
3	o(d1) o(d2) o(d4)	3
3	o(d1) o(d2) o(d5)	1
4	o(d1) o(d2) o(d3) o(d4)	4

Figure 2 shows the algorithm for the data dependency rule set formulation. Since the total number of iterations for both the FOR loops are less than the total number of transactions in the database, the algorithm runs with a predictable run time.

c) The rule sets that are in each access role's Ai{} are added to the rule set repository called the Data Dependency Repository (DDR). The DDR is kept up to date with all data dependency rule sets and heavily used to identify all malicious transactions and effectively secure the database cache as is explained in the following phases.

Figure 2 Algorithm for Rule Set Formulation

4.3. Phase 3

a) Identifying malicious Database transaction from Transaction logs: For every transaction a log entry is made into the Transaction logs. The IDS will identify the rule set for the transaction from the repository which will also give the minimum hierarchy level required to initiate this transaction. This hierarchy level shows the degree of maliciousness (whether a level 3 user is trying to access a database using level 5 access privileges).

b) Intrusion Trends: A performance check on the logs can identify trends in malicious transactions and by tracing the transactions marked malicious weak data dependencies which may cause these security leaks can be easily determined.

c) Securing database cache: Database caches which cache the resultsets from frequent queries to reduce bottle necks and database pool accesses, can now use the malicious degree assigned to the transaction to determine whether to cache the resultset or not. By not caching highly malicious resultset we can help prevent security leaks due to malicious accesses to the database cache.

```
1        i←0
2        For each ARi
3            Set Ai ← {}
4            Set TAi ←{ set of all transactions tz in T for
ARi}
5                Set DAi ←{ set of all operations on
all data items in TAi}
6            Set ruleSetLength ← 1
7                While(rule sets aggregate)
8                j ← 0
9                For each tz in TAi
10                   add each unique sequence to Ai{}
11                   DAi[j] ← number of times a unique
operational (read/write) sequence of length ruleSetLength
occurs in TAi's transactions. The transactions are
scanned from left to right in that order only.
12               j← j + 1
13           ruleSetLength ← ruleSetLength + 1
```

Figure 2. Algorithm for data dependency rule set formulation.

4.4. Phase 4.

Updating the DDR: In this phase all changes in the access role definitions are analyzed. (A mobile agent can be used to collect the changes from the database(s)). Not all rule sets need to be changed. Only that access role whose permission changed and its parents/grandparents need to have their rule sets updated as the other access roles will not be affected. It must also be noted that any change in a user's access role will automatically come into effect and will not affect the intrusion detection system as the rule sets will not change. Re-running the rule set formulation passes for every access role change is not necessary as generally all changes are made during database upgrades periodically by the database administrator during general maintenance. Hence re-running the rule set formulation will not decrease the performance of the database.

5. Analysis.

In this section we show how we have analyzed our approach by applying it to two well known scenarios thereby showing the effectiveness of the approach.

Scenario 1: Consider a typical Student-Course-Professor database. A sample definition for its access roles is shown in Table 2. The hierarchy can be represented as follows: Admin ← Professor ← Advisor, Student. This means that all rule sets that apply to the Advisor and/or Student roles will apply to the Professor role and all rights of the Professor, Advisor and Student are inherited by the Admin.

Table 2. Access role definition for scenario 1

	Course View	Student Info	Course Info	Professor Availability
Student	r	r	r	
Professor	r/w	r/w	r/w	r/w
Advisor	r	r/w	r	r/w
Admin	r/w	r/w	r/w	r/w

Scenario 2:

We simulated a typical Employee-Payroll Accountant-Employer database. A sample access role definition is shown in Table 3. The hierarchy can be shown as Admin ← Employer, Payroll Accountant, Employer.

Table 3. Access role definition for scenario 2

	Employee Personal View	Employees Payroll table	HR View
Payroll Accountant	r	r/w	r
Employer	r	r	r/w
Employee	r/w		r
Admin	r/w	r/w	r/w

Notice that in Scenario 2 all access role types except for the admin are at the same level. There are not many levels of hierarchy here and hence the algorithm took longer to generate all dependencies. In a typical system, the time taken to generate the initial rule sets will be the longest. Updating the rule sets in the repository will not affect the performance of the database by itself. It must be noted that the robustness of the result sets in identifying malicious transactions with changing access role definitions is the main factor determining the efficiency of this approach. For both scenarios: We used auto-generated log files of 10,000 random transactions from a list of valid and invalid transactions. Once Phase 2 is completed, a list of rule sets depicting data dependency patterns is stored in the repository and intrusion detection begins.

We then tested the rule sets with a list of 1000 random transactions consisting of both valid and invalid transactions. Every transaction is marked malicious by identifying the security degree for that transaction (The degree specifies the minimum user access role required to initiate that transaction) based on the access role the pattern adheres to. We changed the access role definition for Professors and the patterns were updated. The database slowed down significantly when the rule sets were reformulated every time an access role was redefined. We changed our setup so that all access role changes were collected over a period of time and the rule sets were

updated only when a significant change in access roles has occurred (such that the support level for the rule sets is below the minimum or when its efficiency in identifying malicious transaction falls below the required level). In this case the reformulation can be planned and made to occur concurrently with database back ups. The transition down time for the databases IDS can thus be kept at a minimum when the rule set generator is rerun during off peak times or when the database is shut down for back up. On contrary all updates/deletes of users, user permissions did not affect the database performance and can be done on the fly as the rule sets are not affected by these changes.

The support level for including a rule set pattern was set to 25% which means the pattern must be seen in at least 25% of the total number of transactions considered. By varying the support level the security level changed accordingly. Care should be taken to set the support level as it directly affects the robustness of the rule sets in identifying malicious transactions.

The processing time for the rule sets is an initial cost. The updates do not interfere with normal database operations and the database is made more intrusion safe with negligible down time. Also the DDR helps in keeping the security leaks caused by poor database cache maintenance in check. Our data dependency rule set algorithm has been shown reduce computation time but since the analysis of the adaptive IDS was made on simulated values, our proposed approach has only been subjected to a preliminary testing. An actual deployment of the approach is currently being performed using intelligent agents and is discussed in the next section.

6. Conclusion and Future work

Our approach gives an efficient approach to deal with intrusion detection in large databases. Our adaptive IDS approach significantly reduces the computational time for identifying and maintaining valid rule sets using hierarchical access roles and pattern repositories. It significantly reduces the databases vulnerability to malicious transactions and weak data dependencies as a result of varying access role definitions. It provides an ability to detect malicious activity when it occurs within existing or past users of the database without slowing the database transactional activity.

Our approach can be deployed onto a real time database system in many ways. We are currently working on deploying our approach using light weight intelligent agents. Using light weight agents allows one to add more functionality in the future without affecting the performance of existing protocols.

Below are some of the agents that may be required to perform the various tasks in our approach:

a) An agent to scan and classify the database schema and access roles. It would take the role of a pre-processor during the initial set up and later to pre-process the transaction logs.

b) A mobile agent to identify changes in the access role definitions.

c) An agent to run the rule set formulation and update the Pattern repository. This agent is a static agent triggered by the mobile agent which identifies the access role definition itself.

d) A Database monitor agent is used to monitor the database caches.

e) All malicious transactions of all degrees can be monitored using a intrusion detection monitor agent.

These agents can be set to run every midnight (or when database activity is low) so that changes in the rule set will not decrease the IDS performance.

Our approach can also be scaled to distributed databases and mobile agents can be used to identify data dependency patterns across databases similarly. These agents are also light weight agents and we are currently working on the performance results for such an approach.

7. References.

[1] Peng Liu Jiwu Jing, Pramote Luenam, Ying Wang Lunquan Li, Supawadee Ingsriswang, "The Design and Implementation of a Self-Healing Database System", School of Info Sciences and Technology Department of Information Systems, Pennsylvania State University UMBC, University Park, PA 16802 Baltimore, MD 21250.

[2 J. McDermott and D. Goldschlag, "Towards a model of storage jamming", *Proceedings of the IEEE Computer Security Foundations Workshop*, Kenmare, Ireland, June 1996, pp. 176-185.

[3] Pramote Luenam, Peng Liu, "ODAM: An On-the-fly Damage Assessment and Repair System for Commercial Database Applications", Dept. of Info. Systems, UMBC Baltimore, MD 21250.

[4] W. Lee, SJ Stolfo, KW Mok, "Data mining approaches for intrusion detection", *Proceedings of the 7th USENIX Security Symposium*, 1998.

[5] P. Liu and S. Jajodia, "Multi-phase damage confinement in database systems for intrusion tolerance", *Proceedings of the 14th IEEE Computer Security Foundations Workshop*, June 2001, pp. 191 – 205.

[6] Ashoka Savasere, Edward Omiecinski, Shamkant B. Navathe, "An Efficient Algorithm for Mining Association Rules in Large Databases", *Proceedings of the 21st International Conference on Very Large Data Bases*, San Francisco, CA, USA, pp. 432 – 444, 1995.

[7] Yi Hu and Brajendra Prasad, "A Data Mining approach for Database Intrusion Detection", *ACM Symposium on Applied Computing*, 2004, x(y): 711 – 716.

[8] Rakesh Agrawal, Andreas Arning, Toni Bollinger, Manish Mehta, John Shafer, Srikant Ramakrishnan, "The Quest Data Mining System", *Proc. of the 2nd Int'l ACM Conference on Knowledge Discovery in Databases and Data Mining*, Portland, Oregon, August 1996, pp. 244-249.

[9] Maged El-Sayed, Carolina Ruiz, and Elke A. Rundensteiner, "FS-Miner: Efficient and Incremental Mining of Frequent Sequence Patterns in Web logs", *Proc. of the ACM WIDM'04*, Washington, DC, November 2004, pp. 12-13.

[10] Rakesh Agrawal, Srikant Ramakrishnan, "Fast Algorithms for Mining Association Rules", *Proc. of the 20th Int'l ACM Conference on Very Large Databases*, Santiago, Chile, September 1994, pp. 487-499.

[11] Richard Relue and Xindong Wu, "Rule generation with the pattern repository", *Proc. of the IEEE International Conference on Artificial Intelligence Systems*, September 2002, pp. 186 – 191.

Structured Multinomial Models of Traffic Flow

Juan K. Lin
Department of Statistics
Rutgers University
Piscataway, NJ 08854
jklin@stat.rutgers.edu

Abstract

We investigate various latent variable models of traffic flow. More specifically, we present various structured multinomial mixture models for analyzing source-destination traffic. A "highway" model is formulated which posits highway entrance and exit hubs, and highway traffic between the entrances and exits. The model's structure is based on an analogy with car traffic from a local origin in one city to a destination in another city. The model's parameters correspond to an onramp traffic distribution from sources to highway entrances, the highway traffic distribution between entrances and exits, and an offramp traffic distribution from highway exits to final destinations. The highway traffic model extracts community structure based on source-destination traffic information, but in addition captures the aggregate "highway" traffic between the communities. This important distinction extends the highway traffic analysis beyond clustering and allows it to extract out underlying backbone traffic structure from traffic data. For comparison, we also describe a "hub" traffic model with no highways which has a latent variable structure that has been well studied in the past.

1. INTRODUCTION

Traffic engineering and network design have been extensively studied in the engineering communities. The investigations cover how best to route traffic based on an existing connectivity graph, and optimizing connectivity paths to best fit the traffic. Source-destination traffic matrix estimation has been addressed from a statistical perspective in e.g.[1][2]. Here we present a probabilistic "highway" traffic model of source-destination traffic. Our goal in the analysis is to model both the underlying community structure and the aggregate traffic between communities. Viewing the source-destination traffic matrix as a weighted graph, we seek to discover both tightly connected regions in the graph, and an underlying "highway" backbone structure in the graph.

Our analysis extends latent variable models which have appeared under the names Latent Class Models [3], Aggregate Markov Models [4]-[6], Non-negative Matrix Factorization (NMF)[7], and probabilistic LSA (pLSA)[8]. Many of the recent applications of these models have been in the fields of natural language processing and information retrieval. These latent variable models when applied to source-destination traffic data translate into a "hub" traffic model with only onramp and offramp traffic to latent hubs. The highway traffic latent variable model contains both highway entrance and exit hubs, and highway traffic between them. This allows the model to find both tightly interconnected communities, and the traffic flow between them. In addition to the analysis of source-destination traffic data, the highway traffic model is applicable to the analysis of random walk traffic on a source-destination connectivity graph. In related work, spectral clustering based on finding communities which minimize transitions between different communities has received considerable attention in image segmentation[9][10].

An outline of this paper is as follows. First we describe the highway traffic model and it's relation to a hub traffic model. Section 3 presents comparative analysis of the highway and hub traffic models for the analysis of traffic on an autonomous system connectivity graph and computer skills graph. Section 4 presents a symmetric hub traffic model. The paper concludes with a discussion of some properties of the highway traffic model.

2. Highway and Hub Traffic Models

Consider traffic flow data consisting of n_{ij} counts of traffic from source $X = i$ to destination $X' = j$. We assume that all sources are destinations, and destinations sources. Discrete latent variables H and H' are introduced which characterize the underlying entrance hubs and exit hubs on the highway. We assume that all entrances are exits, and

vice versa. Our model of traffic flow consists of onramp traffic from sources to highway entrances, highway traffic from entrances to exits, and offramp traffic from highway exits to destinations. The model assigns a probability of going from source i to destination j of:

$$p(i,j) = \sum_{k,l} \alpha_{ik}\beta_{kl}\gamma_{jl},$$

where $\alpha_{ik} = P(X = i|H = k)$, $\beta_{kl} = P(H = k, H' = l)$, and $\gamma_{jl} = P(X' = j|H' = l)$. In words, α_{ik} is the fraction of traffic at entrance k from source i, β_{kl} is the probability of going from entrance k to exit l on the highway, and γ_{jl} is the fraction of traffic at exit l that proceed to destination j. The double sum in the expression is over all highway entrances and exits. Note that the traffic model is probabilistic, and in general allows for more than one highway route from source to destination. We further impose a constraint equating the onramp and offramp traffic distributions:

$$\gamma_{jl} = \alpha_{jl}.$$

Thus the fraction of traffic at exit l which continue to destination j is equal to the fraction of traffic at entrance l which originate from j. The model parameters are specified by $\alpha(x|h) = P(x|h)$ and $\beta(h,h') = P(h,h')$, which specify respectively the onramp/offramp traffic distribution, and highway traffic between the entrances and exits. Let the total amount of observed traffic be $N = \sum_{i,j} n_{ij}$, and let $\tilde{p}_{ij} = n_{ij}/N$ be the observed empirical joint distribution $\tilde{p}(x = i, x' = j)$. The log-likelihood function is given by

$$\mathcal{L} = N \sum_{x,x'} \tilde{p}(x,x') \log[\sum_{h,h'} \alpha(x|h)\beta(h,h')\alpha(x'|h')].$$

Maximizing the likelihood of the observed source-destination traffic counts is equivalent to minimizing the following Kullback-Leibler divergence:

$$\mathcal{D}(\tilde{p}(x,x') \parallel \sum_{h,h'} \alpha(x|h)\beta(h,h')\alpha(x'|h')).$$

The EM algorithm gives the following update equations
E-step

$$q(h,h'|x,x') = \frac{p(x,x',h,h')}{\sum_{hh'} p(x,x',h,h')}$$

where $p(x,x',h,h') = \alpha(x|h)\beta(h,h')\alpha(x'|h')$.
M-step

$$\alpha(x|h) = \frac{\tilde{p}(X = x, H = h) + \tilde{p}(X' = x, H' = h)}{\tilde{p}(H = h) + \tilde{p}(H' = h)}.$$

$$\beta(h,h') = \tilde{p}_{hh'},$$

where $\tilde{p}_{xh}$, $\tilde{p}_{x'h'}$, $\tilde{p}_h$, $\tilde{p}_{h'}$, and $\tilde{p}_{hh'}$ are the corresponding marginals of $\tilde{p}_{xx'}q(h,h'|x,x')$.

Representing the model parameters α and β as matrices, the highway traffic model seeks an approximation of the empirical traffic distribution $\tilde{p}$ by minimizing

$$\mathcal{D}(\tilde{p} \parallel \alpha\beta\alpha^t).$$

In comparison, a traffic model with the same structure as pLSA/NMF [4][7][8] seeks to minimize

$$\mathcal{D}(\tilde{p} \parallel AB).$$

The traffic interpretation of this model, which will be referred to as the "hub" traffic model, consists of an onramp distribution to the hubs from the sources, the hub distributions, and the offramp distributions from hubs to destinations. The highway model assumes more structure in the traffic data, and is a constrained version of the hub model. In particular, a highway model can always be represented as a hub model by equating corresponding terms in $(\alpha\beta)(\alpha^t) = (A)(B)$, effectively folding in the highway traffic between entrances and exits into the onramp traffic distribution specified by A. This comes at the cost of reduced sparseness of the onramp traffic distribution, and an increase in complexity of the hub model. Without equating onramp to offramp traffic in the highway model, the highway traffic has extra degrees of freedom since we can always write $\alpha\beta\gamma = (\alpha\beta)(I)(\gamma)$. Here the onramp traffic incorporates the highway traffic, and now there is no cross-traffic between entrances and exits. By equating onramp to offramp traffic, these degrees of freedom are removed in the highway traffic term β.

The highway and hub traffic models differ in complexity, sparseness and structure. In Section 3.1, the highway and hub traffic models will be compared using the Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) scores, as well as predictive test-set log-likelihoods. In Section 3.2, we demonstrate the extraction of a highway backbone structure in a random walk traffic matrix.

3. Numerical Experiments

3.1 Synthetic graph analysis

We start with a simple example analysis which elucidates the highway traffic model's ability to find communities and their interrelations. A simple synthetic graph consisting traffic between 12 nodes is depicted on the left in Figure 1. Directed edges correspond to one traffic count in the given direction, whereas undirected edges represent a traffic count in both directions. The empirical joint source-destination distribution for the graph has exact decompositions according to both the highway and hub traffic models, with zero KL-divergences. Thus, the comparison here

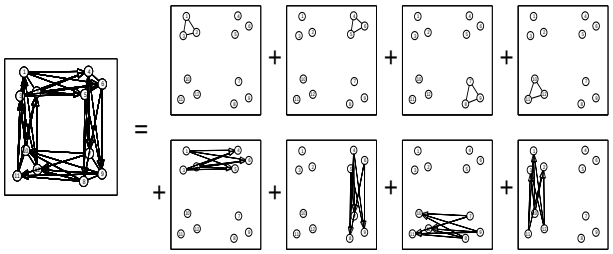


Figure 1. Synthetic graph decomposition based on the highway traffic model. The decomposition consist of four subgraphs of tightly knit communities and four subgraphs of relations between the communities.

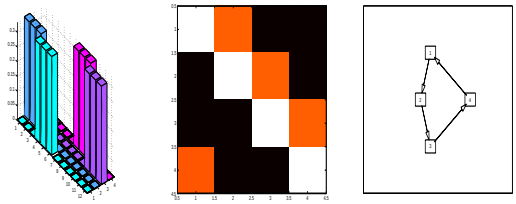


Figure 2. The highway model's on-ramp/offramp distribution (left), highway traffic β (center), and highway traffic visualization (right).

is in terms of the structure each model extracts from the data. For the highway model, the EM algorithm described above is run for 100 iterations starting from random initializations for $\alpha(x|h)$ and $\beta(h,h')$. The algorithm often finds the exact decomposition as shown in the figure. The exact decomposition of the graph consists of $k = 4$ fully connected communities consisting of 3 nodes each, given by $\alpha(x|h = i)\beta(h = i, h' = i)\alpha(x'|h' = i)$. These are depicted in the top four subgraphs on the right in Figure 1. In addition, the relations between the communities, as given by $\alpha(x|h = i)\beta(h = i, h' = j)\alpha(x'|h' = j)$, is depicted on in the bottom four subgraphs.

In Figure 2, the onramp/offramp distribution parameter α, and the highway traffic parameter β are displayed. In addition, a binary representation of the graph's highway backbone structure is visualized by thresholding β. For comparison, we fit a hub traffic model to the data. The corresponding graph decomposition if shown in Figure 3. The hub model all traffic within communities together with all outbound traffic from that community. The hub model essentially incorporated the highway traffic distribution into the offramp distribution.

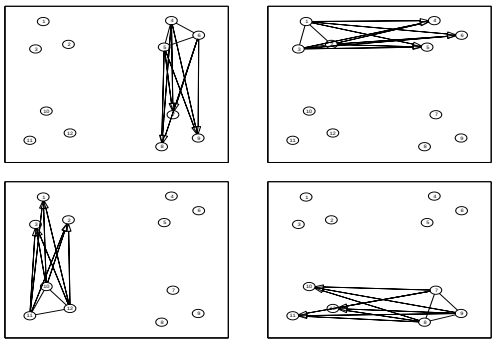


Figure 3. Synthetic graph decomposition based on the hub traffic model.

The highway model's analysis of this simple traffic graph successfully captures the tightly knit communities and their interrelations. In addition, the highway traffic matrix $\beta(h,h')$ describes the highway backbone traffic structure in the data.

3.2. Autonomous system connectivity graph

We analyzed simulated internet traffic data based on an undirected connectivity graph between Autonomous Systems (AS). The connectivity graph consists of AS paths in BGP routing tables collected by the server *route-views.oregon-ix.net*. This data is the basis of the power-law analysis in [11], and is publicly available at *http://topology.eecs.umich.edu/data.html*. After trimming out nodes with no edges, we are left with an undirected binary AS connectivity graph with 13233 interconnected AS nodes.

We compared the highway traffic model to the hub traffic model normalizing for the complexity differences of the two models. With k latent hub states in the hub model, and n sources/destinations, the hub model has $[2k(n-1)+k-1]$ parameters. In contrast, the highway model with the same number k or entrances/exits contains only $[k(n-1)+k^2-1]$ parameters. We compared the two models using the Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC), and predictive test-set log-likelihoods. The simulated traffic data was constructed as follows. For the training set, we performed 100000 *single* random walk steps on the connectivity graph. The 100000 source nodes were sampled in accordance with the stationary distribution of the random walk on the connectivity graph. Traffic from source nodes are assumed to follow each of the edge paths with equal probability. Since multinomial mixture models can be prone to over-fitting problems, we added a single pseudo-count traffic for each edge in the connectivity graph. If traffic from a source to a destination is not observed in

the test set, but appears in the training set, the traffic models may assign zero probability to the test set likelihood. Early stopping will effectively stop parameter updates if an update assigns zero probability to a traffic path that appears in the test set. An additional inverse annealing (heating) is used in [8] to smooth multinomial parameters and prevent sparseness. For the test set, 20000 single random walk steps were simulated.

In Table 1 the AIC and BIC scores for the highway and hub models are tabulated for a number of different k values. For each model and each k, 10 EM runs with random parameter initializations are performed. Scores for the best respective runs are reported in the table. The highway traffic model has significantly better (lower) AIC and BIC scores than the hub traffic model.

values $\times 10^6$	k=26	k=51	k=100	k=197
Highway AIC	**4.90**	**5.41**	**6.59**	**9.07**
Hub AIC	5.56	6.72	9.16	14.15
Highway BIC	**8.34**	**12.2**	**19.9**	**35.0**
Hub BIC	12.4	20.2	35.5	66.1

Table 1. AIC and BIC scores for the highway and hub models.

We also compared predictive test-set log-likelihoods for a highway model and hub model with comparable degrees of freedom. Comparing the $k = 51$ highway model with 677432 parameters with the $k = 26$ hub model with 688089 parameters, the best test set log-likelihoods ($\times 10^5$) were -2.64 and -2.74 for the highway and hub models respectively. Comparing the $k = 100$ highway model with 1333199 parameters with the $k = 51$ hub model with 1349714 parameters, the best test set log-likelihood($\times 10^5$) were -2.55 and -2.63 respectively. Finally, the $k = 197$ highway model with 2645512 parameters and the $k = 100$ hub model with 2646499 parameters had best test set log-likelihoods($\times 10^5$) of -2.46 and -2.54 respectively. In all three comparisons, the highway model had slightly fewer parameters, but significantly higher (less negative) predictive test set log-likelihoods.

3.3. Random walk traffic on computer skills graph

Aside from complexity and sparseness considerations, the highway model extracts underlying backbone traffic structure which clustering models like the hub model does not. We analyzed a smaller, more easily interpretable data set to try to find communities and the relationships between the communities. A computer jobs description data set, provided courtesy of Prof. Richard Martin and the IT consulting firm Comrise was analyzed. The raw data consists of a collection of computer job descriptions, each of which contain a subset of 159 computer skills the hiring manager considered important for the job. The most frequently occurring skills keywords in the job descriptions are "unix", "pc(ibm)", "windows95", "windowsnt", "c" and "oracle". Entries along the diagonal of the co-occurrence matrix contain the number of times each skill occurred over all the job descriptions. The elements of this matrix is interpreted as a the amount of (co-occurrent) traffic between pairs of job skills. This interpretation is equivalent to the normalization used in the random walk view of segmentation [9]. From the co-occurrent traffic information on the computer skills graph, we seek to extract out underlying computer skill topic communities, and the underlying backbone connectivity structure between the topic communities.

A visualization of the computer skills traffic graph is shown in Figure 4(a) using the *GraphViz* [12] spring model graph layout program from *AT&T Labs-Research*. Only edge connections with average transition probability greater than $.085$ are shown. Even though the graph is not very large with 159 nodes, the visualization is not easily readable, and only provides vague clues to relationships between various skills.

From the job skills co-occurrence table the observed empirical joint distribution $\tilde{p}_{xx'}$ is constructed. The EM algorithm is used to find the maximum likelihood estimators for the conditional $\alpha(x|h)$ and the joint $\beta(h, h')$.

Since the onramp and offramp traffic distributions are equal in the highway model, we will simply refer to the offramp traffic. The offramp traffic distribution from a few exits are tabulated in Table 2 This specifies the fraction of traffic at the specified exit which flow to each destination node. The destination computer skill with the largest traffic fraction is used as the label for the exit. The five top skills, ranked in descending order of their conditional probabilities are shown for each exit. From Table 2, we see a **UNIX** skills community containing *Unix*, *C* and *C++*, and a **SUNOS** operating systems community containing *SunOS*, *solaris* and *sunsparc*, and an **HP** cluster with *HP*, *HP-UX*. The model also identified skills groups affiliated with Microsoft, containing skills *PC(IBM)*, *Windows95*, *MSoffice*, *MSproject* and *dos*, and a *Java* group (not tabulated) containing *javascript*, *perl*, *cgi* and *html*.

In addition to the communities of related computer skills, the model also extracts out the relationships between the communities. In Figure 4(b), we used *GraphViz* [12] to visualize the underlying highway traffic between entrance and exit hubs as defined by $\beta(h, h')$. This backbone traffic structure in the source-destination traffic data is visualized by thresholding $\beta(h, h')$ into a binary adjacency matrix. We emphasize that this is only for visualization purposes; the model contains more information than is visualized. From the highway traffic graph, we see tightly coupled highway

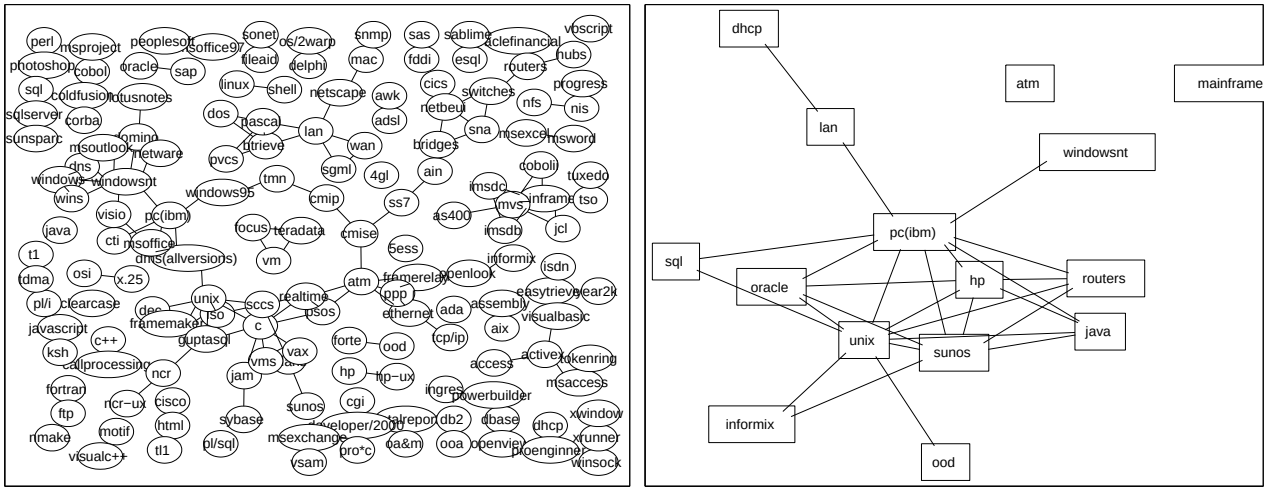


Figure 4. (a) Graph layout of the computer skills traffic graph using *GraphViz's* spring model layout algorithm. (b) Highway traffic visualization - each node in this graph is a highway entrance or exit hub, and corresponds to a computer skills community. Onramp and offramp distributions to sources and destinations are tabulated in Table 2.

Unix	.156	SunOS	.174	HP	.181	pc/ibm	.214
c	.154	solari	.169	hp-ux	.146	win95	.184
c++	.123	tuxedo	.016	tcpip	.076	msoff	.146
syb	.050	sunspa	.009	nis	.008	mspro	.050
jam	.004	oa&m	.007	nfs	.007	dos	.027

Table 2. Onramp/offramp traffic distribution for highway traffic model. The skills with highest traffic fraction to/from the latent states are listed in the first row, and used to label the clusters in Figure 4(b). Each column represents an entrance/exit hub. Fractions of traffic to/from each skill is listed next to the skill name.

traffic between the **Unix, SunOS, HP** communities, as well as the **Java** and **SunOS** communities. The highway traffic model successfully finds computer skills communities as well as the relationships between the communities.

4. Symmetric Hub Traffic Model

The highway traffic model assumes that the traffic is generated from an underlying highway traffic distribution, onramp traffic distributions from sources to highway entrances, and an identically distributed offramp distribution from highway exits to destinations. In contrast, the hub traffic model only has onramp and offramp distributions, and no analog of highway traffic. As discussed in Section 2, the hub model contains the highway model as a special case, where the composition of the onramp and highway traffic is subsumed into a single onramp traffic distribution for the hub model. Using the hub model to describe traffic data comes at the complexity cost of roughly double (for $n >> k$) the number of parameters as the highway model. An even more restrictive traffic model can be defined by equating onramp and offramp traffic distributions in the hub model. A model with this structure was first investigated in [14]. This "symmetric hub" model can be obtained from the highway model by constraining $\beta(h, h')$ to be diagonal.

We assume the traffic flow in the empirical joint distribution $\tilde{p}(x, x')$ is symmetric with respect to x and x'. This implies that the transition matrix $\tilde{p}(x'|x)$ is consistent with a reversible random walk. Instead of minimizing the Kullback-Leibler divergence for the highway traffic model:

$$\mathcal{D}(\tilde{p}(x, x') \parallel \sum_{h,h'} \alpha(x|h)\beta(h, h')\alpha(x'|h')),$$

the symmetric hub model minimizes

$$\mathcal{D}(\tilde{p}(x, x') \parallel \sum_{h,h'} \alpha(x|h)\beta(h)\alpha(x'|h')).$$

The EM algorithm for this model results in the iterations:

E-step

$$p(h|x,x') = \frac{\alpha(x|h)\alpha(x'|h)\beta(h)}{\sum_h \alpha(x|h)\alpha(x'|h)\beta(h)}$$

M-step

$$\beta(h) = \sum_{x',x} p(h|x,x')\tilde{p}(x',x),$$

$$\alpha(x|h) = \sum_{x'} \frac{p(h|x,x')\tilde{p}(x',x)}{\beta(h)}.$$

This symmetric hub model is a constrained version of both the hub model, and the highway model as follows. First, the symmetric hub model can be seen as a hub model with identically distributed onramp and offramp distributions. Second, the symmetric hub model is a highway model with the constraint of no traffic between the entrance/exit hubs.

This model does not directly capture relationships between communities since it does not directly model traffic between hub communities. However, it can describe traffic between hubs after *two* time steps of the empirical source-destination transition. An inter-hub traffic can be specified by combining the offramp traffic from hubs to destinations during the first time step, with the onramp traffic from destinations back to the hubs during the second time step. This is computed as follows:

$$p(h,h') = \sum_x \alpha(x|h)\beta(h)\alpha(x|h').$$

This *2-step* inter-hub traffic layout for the computer skills data is shown in Figure 5, with the onramp/offramp distributions tabulated in Tables 3 and 4. The skills topic communities successfully combine unix groups, windows groups and programming language groups, while the inter-hub traffic successfully represents the relationships between topic groups.

SunOS	.172	HP	.168	C++	.147
unix	.145	hp-ux	.139	C	.067
solaris	.142	tcp/ip	.080	ood	.060
tuxedo	.016	nis	.009	nmake	.031
sunsparc	.010	nfs	.008	dec	.007

Table 3. Onramp/offramp traffic distribution for highway traffic model the hub traffic model in Figure 5.

5. Highway traffic model properties

The source-destination traffic data analyzed in this paper either directly translated into symmetrical empirical joint distributions (computer skills), or were simulated from a reversible random walk (AS traffic). The highway traffic model on the other hand can in general describe non-symmetric traffic data. Looking at the model in more detail, the equating of the onramp with the offramp traffic distribution in the model results in the following conditional distribution within each community:

$$p(x,x'|h=h') = \alpha(x|h)\alpha(x'|h).$$

This is the highway model's predicted probability of transiting from source x to highway entrance h, and immediately exiting to destination x'. This conditional distribution matrix has rank 1 and satisfies the detailed balance condition. Thus, within each community, the random walk traffic is symmetric, and one can show that $\pi_h(x) = \alpha(x|k)$ is simply the stationary distribution of the random walk within each community. Even though the random walk within each community is reversible, the highway model can model non-reversible traffic depending on the highway

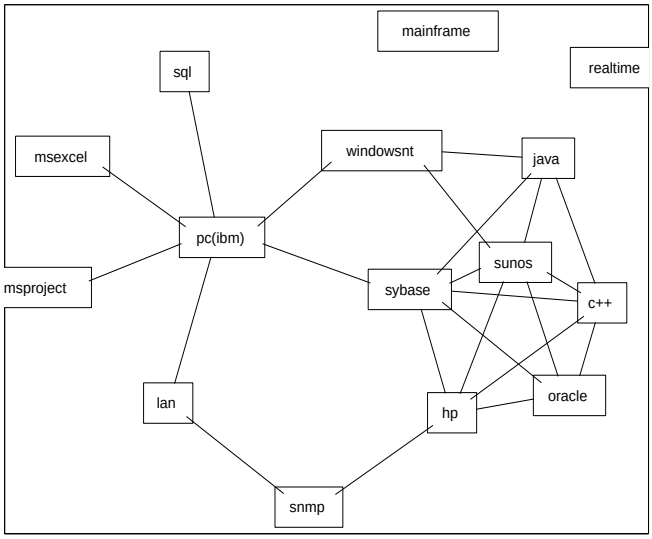


Figure 5. Symmetric hub model inter-hub traffic graph.

PC(IBM)	.235	windowsnt	.147	msexcel	.101
win95	.197	dos	.068	msword	.074
msoffi ce	.123	visualc++	.060	outlook	.011
lot-notes	.014	visualbasic	.031	visio	.011
mac	.002	vbscript	.007	isdn	.009

Table 4. Continuation of onramp/offramp traffic distribution for Figure 5.

traffic distribution $\beta(h, h')$.

If the highway traffic $\beta(h, h')$ between communities is symmetric with respect to source community (entrance) and destination community (exit), thereby satisfying the detailed balance condition, then the highway model describes symmetric source-destination traffic. One can verify that if the empirical traffic distribution is symmetric, and the highway traffic distribution $\beta(h, h')$ is initialized symmetric, then it will remain symmetric under all subsequent updates under the EM algorithm. Thus reversible source-destination traffic will be modeled with a reversible highway traffic model.

The traffic model approximates the empirical traffic flow in a maximum likelihood or minimum KL-divergence sense. For example, an approximation of the source traffic distribution can be obtained as follows. Let the source distribution of the highway traffic model be $\pi(x) = \sum_{h,h'} \alpha(x|h)\beta(h, h')$. Using Pinsker's inequality [13] we can bound the total variation distance between the empirical source distribution $\tilde{p}(x)$ and the source distribution of the highway model $\pi(x)$

$$\begin{aligned} & \sum_{x,x'} |\tilde{p}(x) - \pi(x)| \\ \leq \quad & \sqrt{2\mathcal{D}(\tilde{p}(x) \parallel \pi(x))} \\ \leq \quad & \sqrt{2\mathcal{D}(\tilde{p}(x, x') \parallel \sum_{h,h'} \alpha(x|h)\beta(h, h')\alpha(x'|h'))}. \end{aligned}$$

Similarly, the highway model can approximate any empirical traffic flow from a source set of nodes to a destination set, with the KL-divergence providing a bound on the approximation error.

6. Discussion

The symmetric hub model, highway model, and hub model are constructed with various structures and associated complexities in the multinomial mixture. This is analogous to controlling the covariance structure in Gaussian distributions, from spherical Gaussian, to Graphical Gaussian models and Factor Analysis, to the full Gaussian with arbitrary covariance structure.

We compared the highway and hub traffic models using the Akaike Information Criterion and also test set log-likelihood for the ASP data set. The suitability of each traffic model clearly depend on the underlying structure of the empirical source-destination traffic. Consider for example, source-destination *car* traffic. One could conceivably build a road system based on the hub traffic model, with onramps from origins to k underlying hubs, and offramps to the destinations. This could capture a highway traffic model distribution with k cities, and highway traffic between them. However, the added complexity of the hub model comes at the significant cost of building non-sparse onramps or offramps. In the extreme limit, one could build roads between all origins and destinations with empirical traffic counts. The benefit of the highway model is in the aggregation of traffic flow along an underlying highway infrastructure. An important consideration in comparing the models should be sparseness of the resulting traffic model. There will in general be domain specific sparseness related cost functions to consider.

Hub models with the same probabilistic structure as pLSA/NMF have been applied in the information retrieval setting to decompose document-word matrices [7][8] and document-citation matrices [15]. In those settings, pLSA does not provide a probabilistic generative model, and is not able to cleanly assign predictive probabilities to new documents. Latent Dirichlet Allocation [16] improves on pLSA by providing a proper generative model. In the source-destination traffic setting we consider, the sources/destinations constitute a fixed set, and the traffic models properly defines probabilities for new traffic between the sources and destinations. The traffic models are properly defined probabilistic models of source destination traffic. Over-fitting however, can be a problem. Specifically, if traffic from a source to a destination is not observed in the test set, but appears in the training set, the traffic models may assign zero probability to the test set likelihood. One can use smoothing or incorporate priors over the multinomial parameters.

In summary, the highway model extracts out communities and relational information in the form of highway traffic between the communities. It is related to spectral clustering algorithms where the interest is in finding communities of nodes with minimal traffic between the communities [9][10]. The highway traffic model extends the framework of minimizing traffic flow between communities and provides a low rank highway based approximation to the empirical source-destination traffic. In the relational data research field, models have been investigated in the context of binary link detection [17], binary relational modeling [18], and in a supervised learning context for link prediction. [19]. We are pursuing extensions of the highway traffic model to address the selection of the number of highway entrances/exits, as well as traffic models with highways and freeways. The highway model can also be extended from an unsupervised to a semi-supervised setting with some observations of highway and onramp/offramp traffic counts.

7. Acknowledgments

This work was supported by the National Science Foundation (NSF Grant DMS-0312275).

8. References

[1] Vardi, Y. (1996) Network Tomography: Estimating Source-Destination Traffic Intensities from Link Data. *Journal of the American Statistical Association*, Vol.91, No. 433, pp365-377.

[2] Cao, J., Davis, D., Wiel, S.V. and Yu, B. (2000) Time-Varying Network Tomography. *Journal of the American Statistical Association*, Vol.95, No. 452, pp.1063-1075.

[3] Everitt, B. (1984). An Introduction to Latent Variable Models. London: Chapman & Hall.

[4] Saul, L. and Pereira, F. (1997) Aggregate and mixed-order Markov models for statistical language processing. In *Proceedings of the second conference on empirical methods in natural language processing*, 81-89.

[5] Brown,P., Della Pietra, V., deSouza,P., Lai. J. (1992). Class-based n-gram models of natural language. Computational Linguistic, 18:467-479, 1992.

[6] Pereira, F., Tishby, N., and Lee, L. (1993) Distributional clustering of English words. In *Proceedings of the ACL*, pp183-190, 1993.

[7] Lee, D. and Seung, S. (1999) Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(675), 788-791.

[8] Hofmann, T. (2001) Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42, 177-196.

[9] Meila, M. and Shi, J. (2001) A random walks view of spectral segmentation. in *Proc. International Workshop on AI and Statistics (AISTATS), 2001*

[10] Ng, A., Jordan, M, and Weiss, Y. (2001) On spectral clustering: analysis and an algorithm. In *NIPS 14* 2001.

[11] Qien, C., Chang, H., Govindan, R., Jamin, S., Shenker, S. and Willinger, W. (1992) The Origin of Power Laws in Internet Topologies Revisited, *Proc. of IEEE Infocom, 2002*

[12] GraphViz, AT&T Labs-Research.
URL: *http://www.research.att.com/sw/tools/graphviz*

[13] Pinsker, M. (1960) Information and information stability of random variables and processes. Moskow: Izv. Akad. Nauk, 1960.

[14] Lin, J. (2003) Reduced-Rank Approximations of Transition Matrices, in C. M. Bishop and B. J. Frey (eds), Proceedings of AI-Statistics'2003, Jan 3-6, 2003, Key West, FL.

[15] Cohn, D. and Hofmann, T. (2001) The Missing Link A Probabilistic Model of Document Content and Hypertext Connectivity. In Advances in Neural Information Processing Systems 13

[16] Blei, D., Ng, A. and Jordan, M. (2003) Latent Dirichlet Allocation, The Journal of Machine Learning Research, vol 3, pp.993-1022

[17]Schneider, J., Kubica, J., Moore, A. and Yang, Y. (2002) Stochastic link and group detection. In *IJCAI* 2002.

[18] Kemp, C., Griffiths, T. and Tenenbaum, J. (2004) Discovering latent classes in relational data. AI Memo 2004-019, M.I.T. AI Laboratory.

[19] Taskar, B.,Wong, M.F., Abbeel, P. and Koller, D. (2003) Link Prediction in Relational Data. Neural Information Processing Systems Conference (NIPS03), Vancouver, Canada, December 2003.

Published by
Department of Mathematics and Computing Science

Saint Mary's University

Halifax, Nova Scotia, Canada

Technical Report Number: 2005-09 November, 2005
ISBN 0-9738918-5-8