An Algorithm to Efficiently Generate an Approximation of a Theory Set[*]

Jasmine A. Malinao and Henry N. Adorna
Department of Computer Science
(Algorithms and Complexity)
Velasquez Ave., UPDiliman
Quezon City 1101
{jamalinao, ha}@dcs.upd.edu.ph

ABSTRACT

A theory set essentially consists of patterns that summarizes the observed inherent behavior of each cluster of observation in a data set. Furthermore, it is used in evaluating the cluster membership of new sets of data. Due to the large amount of computational resources needed to generate a theory set of a given data set, we propose an algorithm to approximate this set efficiently.

The Approximate Crisp Theory Set Formation (ACTSF) algorithm is proposed to efficiently generate an approximation of theory set of a given k-tagged linearly inseparable data set. It is based on the methodologies developed in the Logical Analysis of Data (LAD) introduced by Peter Hammer. Since ACTSF is an approximation algorithm, it is inevitable to have an increased number of unclassified observations. To compensate for this, we extend ACTSF to include a discriminant function. This paper shows the improvement in the classification power of the theory sets as this discriminant function is injected in the proposed methodology.

We show the effectiveness of this algorithm by testing it on a data set from the UCI Repository of Machine Learning Databases, University of California. In particular, we use the famous multivariate Iris data set created by Fisher in the 1988. Then, we compare the average performance of our algorithm against other known algorithms in the literature to measure its competency. ACTSF performed better than almost all of them.

[*]This is partially supported by a grant from ERDT project entitled "Information Visualization via Data Signatures".

1. INTRODUCTION

In the 1980s, Peter Hammer, a Romanian mathematician (who died in 2006) developed a data analysis tool called Logical Analysis of Data (LAD). The main distinguishing factor of LAD from other frequently used data analysis tools, e.g. Neural Networks, Support Vector Machines, Decision Trees, Bayesian-based classification rules, is that "instead of just asking the question whether a new observation is positive or negative, it tries to approximate the subspace of R^n containing the positive and that containing the negative observations"[1].

Thus, logical explanations are directly derivable from the clusters. In this case, explaining the behavior of these groupings is relatively easy and well understood and may provide new and insightful information of the data set. Apparently, the terms *positive* and *negative* connote data sets having at most two clusters.

It had undergone series of developments to suit the needs of data analysis. From concepts ranging from Boolean Logic to pattern generation of certain data spaces, LAD has found numerous practical and medical applications over the years [1, 3, 4].

In particular, it has been applied to explain medical data sets with tags such as {poor prognosis, good prognosis} for breast cancer cases, {high risk, low risk} to mortality by a certain disease [1]. In most cases, these data clusters are crisp in nature, i.e. a clear boundary exists between them, each of which contains exclusively positive or negative observations.

Following the concepts inherent in LAD, we develop the Approximate Crisp Theory Set (ACTSF) algorithm to be able to deal with classes which are linearly inseparable from one another. Furthermore, we would also deal with data sets with k number of tags, where $k \geq 2$.

2. DEFINITIONS

The Data Set

DEFINITION 1. *Let β be a data space, that is could either be nominal or numeric.*

A data set S^{n+1} is a set of observations or $(n+1)$-tuple $X_i = (x_{i1}, x_{i2}, \ldots, x_{in}, t)$, where $x_{ij}, t \in \beta$, where each x_{ij} as the attribute value in the jth dimension of X_i, and t is the cluster to which X_i belongs.

We call t as the tag associated to X_i.

Note that for convenience, we would also use the notation $x_j, j \in \{1, 2, ..., n\}$ to simply refer to the j^{th} dimension of the data set.

We note that tags can either be elementary or nonelementary. By elementary, we mean that no other tag composes it and nonelementary are combinations of elementary tags. If we let T denote the set of all distinct tags, then in our case we set $|T| = k \geq 2$.

Let $m \in T$. We identify a cluster of observations P_m in the data set whose tag is m, for every m. We would often times use the term *k-tagged* data set to mean a data set whose tag set T has k elements.

It is always the case that $|P_m| \geq 1$, and no two clusters contain the same observation, i.e. $\bigcap_{m \in T} P_m = \emptyset$. If we take the union of all these clusters in S, we have

$$S^{n+1} = \bigcup_{m \in T} P_m \subseteq \beta^{n+1}.$$

This suggests that clusters of observation in S^{n+1} partition the data set S^{n+1}.

Linear Separability

Let r and s be distinct tags in T. Let P_r and P_s be the clusters in S^{n+1} corresponding to these tags.

Let the observation $X_g = (x_{g_1}, x_{g_2}, ..., x_{g_n}, r) \in P_r$ and the observation $Y_h = (y_{h_1}, y_{h_2}, ..., y_{h_n}, s) \in P_s$.

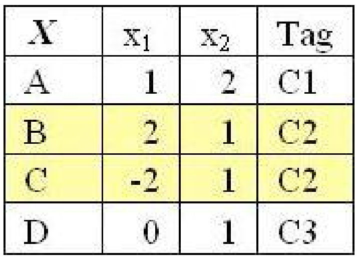
We say that P_r and P_s are *linearly separable* if and only if there exists a set of coefficients $Q = \{q_0, q_1, ..., q_n\}, q_j \in \Re$, such that

$$\sum_{j=1}^{n} q_j x_{g_j} \geq q_0, \quad \forall X_g \in P_r, \quad \text{and}$$

$$\sum_{j=1}^{n} q_j y_{h_j} \geq q_0, \quad \forall Y_h \in P_s.$$

If no such solution exists, then P_r and P_s are *linearly inseparable.*

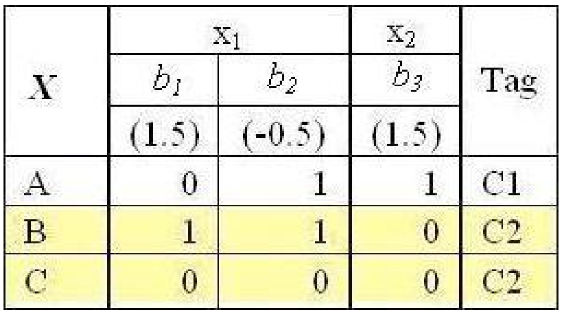
Suppose we are given the data set S^4 with the tag set $T = \{C1, C2, C3\}$ as shown in Figure 1, we see that we can express observation $A = (a_1, a_2) \in P_{C1}$ as $a_j = (\frac{5}{4})(b_j) + (\frac{3}{4})(c_j)$, where $B = (b_1, b_2)$, and $C = (c_1, c_2) \in P_{C2}$. Thus P_{C1} and P_{C2} are linearly inseparable.



X	x1	x2	Tag
A	1	2	C1
B	2	1	C2
C	-2	1	C2
D	0	1	C3

Figure 1: Sample Data set S^{n+1}

However, by introducing a process introduced in [3] known as *data binarization* to these two clusters, we can easily obtain separation between them. This process converts the numeric- and nominal-valued attributes into 0*s* and 1*s*. The binarization of the data set in Figure 1 is illustrated in Figure 2.



X	x1		x2	Tag
	b_1	b_2	b_3	
	(1.5)	(-0.5)	(1.5)	
A	0	1	1	C1
B	1	1	0	C2
C	0	0	0	C2

Figure 2: ${S_{bin}}^{n'+1}$ of Sample Data set

Linear separability is observed when our set of coefficients in the linear inequalities is $Q = \{0, 1, -1, 0\}$.

This binarization process apparently becomes helpful in the process of discriminating sets of observations. It breaks down the raw information of the data set into finer details. Notice that a dimension in the original data set S^{n+1} is mapped into one or more binary dimensions b_i forming ${S_{bin}}^{n'+1}$, $n' >> n$. Each b_i is associated to a parameter called a *cut point*[3]. We see in Figure 2 these cut points as real values indicated below each b_i. They may however take nominal values depending on the type of dimensions at hand. These cut points are used to transform numeric or nominal values into 0s and 1s by conducting attribute-cut point tests. We provide a detailed discussion on this transformation on the subsequent sections.

2.1 Pattern Space

To enable us to describe the behavior of each cluster in the data set S^{n+1}, we need to generate a set of patterns for each of them.

Let a literal be a dimension ${x_j}^{\alpha}$, $1 \leq j \leq n, \alpha \in \{0, 1\}$. We may interchangeably use the notations x_j and $\overline{x_j}$ to denote the literals ${x_j}^1$ and ${x_j}^0$, respectively.

Let $LogicOperators = \{IsEqual(a, b), IsNotEqual(a, b),$

$IsGreaterThan(a,b), IsLessThan(a,b),$
$IsGreaterThanEqual(a,b), IsLessThanEqual(a,b)\}$. The elements of this set contain relational operations to test the relationship of the first parameter a to the second parameter b with the logical operations $=, \neq, >, <, \geq, \leq$, respectively. It returns 1 (i.e. TRUE) when a and b conforms with the specified boolean operation, 0 (i.e. FALSE) otherwise.

We associate to the literal ${x_j}^{\alpha}$ the tuple $L = (V, OP)$, where we have the set of cut points $V = \{v | v \in \beta\}$ and $OP = \{op | op \in \mathit{LogicOperators}\}$, $|V| \leq 2$. Each element $v \in V$ associates to an element $op \in OP$ such that $|V| = |OP|$.

A term Tr is a conjunction of literals, i.e.

$$Tr = \bigwedge_{j \in \{1,2,\ldots,n\}} {x_j}^{\alpha_j}.$$

The number of literals in the term Tr is the *degree* of Tr.

DEFINITION 2. *Let an observation $Y_i = (y_{i1}, y_{i2}, \ldots, y_{in}, m) \in P_m (\subset S^{n+1})$. Let a term $Tr = \bigwedge_{j \in \{1,2,\ldots,n\}} {x_j}^{\alpha_j}$ associated to $L_j = (V_j, OP_j)$, where $|V_j| \leq 2, \alpha_j \in \{0,1\}$. Let the function $NOTXOR(a,b)$ be the usual boolean operation, i.e. $NOTXOR(a,b) = 1$ if $a = b$, 0 otherwise.*

Let $r \in T \backslash m$.

We say that Tr is a **pattern** *of P_m true to Y_i if and only if*

- *$NOTXOR(\alpha_j, op_{j_g}(y_{ij}, v_{j_g})) = 1$, and* (2.1)
- *for all $Z_e = (z_{e1}, z_{e2}, \ldots, z_{en}, r) \in S^{n+1} \backslash P_m$, $NOTXOR(\alpha_j, op_{j_g}(z_{ej}, v_{j_g})) = 0$,* (2.2)

where $v_{j_g} \in V_j$, and $op_{j_g} \in OP_j, j \in \{1,2,\ldots,n\}$

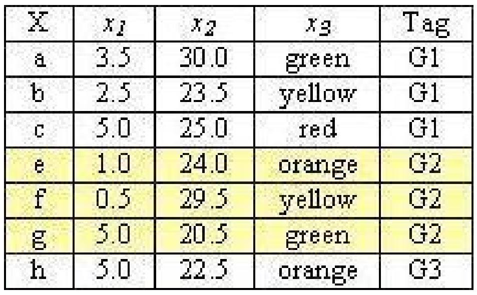
If Tr is a pattern of P_m for the observation Y_i, we say that Tr *covers* Y_i, and conversely, Y_i is being covered by Tr.

This definition implies patterns are not self-contradictory, i.e. a literal x_j and its negation $\overline{x_j}$ cannot appear simultaneously in a given pattern. However, a literal (or its negation) can appear at least once but will be associated to different values of the tuple L_j.

Suppose we are given the following 3-tagged data set S^4 in Figure 3 with $n = 3, T = \{G1, G2, G3\}$.

Let the term $Tr = \overline{x_1}$. We associate to the literal x_1 the tuple $L_1 = (V_1, Op_1)$ where $V_1 = \{2.0\}$ and $Op_1 = \{IsGreaterThanEqual(a,b)\}$. This association tells us to test whether the negated (or complemented) result (note that $\alpha_1 = 0$) of evaluating if the value of an observation in its x_1 dimension is greater than or equal to 2.0.

More precisely, we express $Tr = \overline{(x_1 \geq 2.0)}$. For the above data set S^4, we see that Tr covers $e, f \in P_{G2}$. Furthermore, Tr does not cover any observation in the union of P_{G1} and P_{G2}. These tests satisfy conditions (2.1) and (2.2) in Definition 2, respectively. Thus, we say that Tr is a pattern for the cluster P_{G2}.



X	x_1	x_2	x_3	Tag
a	3.5	30.0	green	G1
b	2.5	23.5	yellow	G1
c	5.0	25.0	red	G1
e	1.0	24.0	orange	G2
f	0.5	29.5	yellow	G2
g	5.0	20.5	green	G2
h	5.0	22.5	orange	G3

Figure 3: Data set S^4

We also see the patterns $Tr_1 = (3.0 \leq x_1 < 3.75)$ and $Tr_2 = (23.0 \leq x_2 < 23.75)$ for some observations in P_{G1}. Additionally, the term $Tr_3 = (22.0 \leq x_2 < 22.75)$ is a pattern for P_{G3}. All of these patterns have degrees equal to 1.

Theory Set

DEFINITION 3. *A theory τ_{P_m} is a disjunction of patterns collectively covering all observations of a given cluster $P_m (\subset S^{n+1})$. Formally,*

$$\tau_{P_m} = \bigvee_{1 \leq i \leq |P_m|} Tr_i$$

$$\forall X \in P_m, \exists Tr_i \text{ covering } X.$$

Using constraint (2.2) of Definition 2, it follows that

$$\bigcap_{\forall m : m \in T} \tau_{P_m} = \emptyset.$$

The set of all theories generated for the data set S^{n+1} is called a *theory set.*

LAD creates a theory set such that each of the observation in the training set is described by at least one pattern in the set. This assures us that each theory of a cluster has a characteristic of being complete - i.e. it is capable of describing every observation in the training set collectively. Furthermore, we use the patterns of the theory set to evaluate the cluster membership of new sets of data.

2.2 Data Binarization

To be able to generate pattern sets with more descriptive power, LAD introduces the concept of data binarization [3]. This process converts the numeric- and nominal-valued attributes into 0*s* and 1*s*. We transform the data set S^{n+1} into its binarized form ${S_{bin}}^{n'+1}$. We follow the steps given below to do this transformation.

1. Let x_i be a dimension of $S^{n+1}, 1 \leq i \leq n$. Let $P_m \subset S^{n+1}, 2 \leq m \leq k$.

Let T be the set of tags in the data set S^{n+1}.

Let $T(k)$ be the k^{th} element in T.

Let c_k be the number of observations tagged to $T(k)$, $k = 1, 2, \ldots, |T|$.

Let the number of *level cut points*[3] be denoted as CP_Lvl. Level cut points are the cut points associated to literals in patterns having a tuple $L = (V, OP)$ where $|V| = 1$. Then,

$$CP_{Lvl} = \sum_{k=1}^{|T|} c_k \sum_{k'=k+1}^{|T|} c_{k'}.$$

Let the number of *interval cut points*[3] be denoted as CP_{Int}. Interval cut points are the cut points associated to literals in patterns having a tuple $L = (V, OP)$ where $|V| = 2$. They are used only for numeric-valued dimensions. Then,

$$CP_{Int} = \binom{CP_{Lvl}}{2}.$$

We transform a dimension x_i into e $b_i s$, where

$$e \leq CP_{Lvl} + CP_{Int}.$$

We let these $b_i s$ be the dimensions of ${S_{bin}}^{n'+1}$, where $n' \leq n * e$.

2. The transformation in (1) converts an attribute value $v \in \beta$ in X_i to a value $v' \in \{0, 1\}$.

   Let $Vals_{x_i} = \{{d_1}^{x_{im}}, {d_2}^{x_{im}}, ..., {d_f}^{x_{im}}\}, f \leq |S^{n+1}|$, $m \in T$ be the distinct values of the dimension x_i of S^{n+1}.

   Let $u = {d_a}^{x_{ir}}$ and $w = {d_b}^{x_{is}}$, $a \neq b$, $r \in T$, and $s \in T \backslash r$. Compute for u and w's level cut point $C_{u,w}$ as

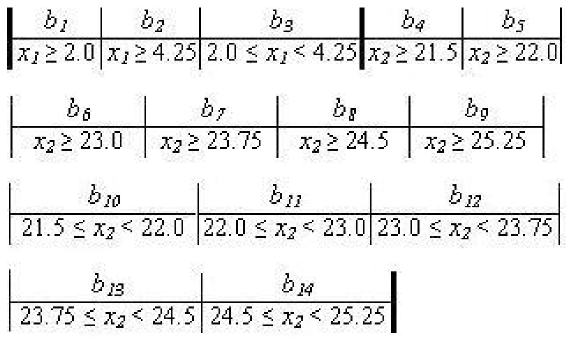
   $$C_{u,w} = \frac{u + w}{2}.$$

   For all continuous-valued dimensions, $v' = 1$ if $v' \geq v$, $v' = 0$ otherwise. On the other hand, for nominal-valued dimensions we simply take the cut points equal to the attribute values themselves. $v' = 1$ if the attribute value is equal to the level cut point, 0 otherwise.

   After computing all level cut points, we generate all interval cut points, say CP_1 and CP_2 by taking 2 level cutpoints and associating them to a binary dimension b_i with a tuple $L = (V, OP)$ where $V = (CP_1, CP_2)$, $OP = (<, \leq)$. If $CP_1 < v \leq CP_2$, then $v' = 1$, else $v' = 0$.

   Suppose we choose the following cut points reflected in Figure 4 and use them to binarize our original data set in Figure 3.

   Next we obtain the corresponding binarized data set ${S_{bin}}^{n'+1}$ as shown in Figure 5.



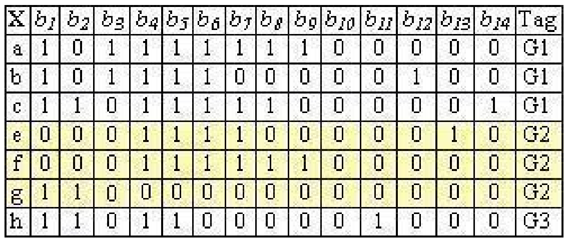
b_1	b_2	b_3	b_4	b_5
$x_1 \geq 2.0$	$x_1 \geq 4.25$	$2.0 \leq x_1 < 4.25$	$x_2 \geq 21.5$	$x_2 \geq 22.0$

b_6	b_7	b_8	b_9
$x_2 \geq 23.0$	$x_2 \geq 23.75$	$x_2 \geq 24.5$	$x_2 \geq 25.25$

b_{10}	b_{11}	b_{12}
$21.5 \leq x_2 < 22.0$	$22.0 \leq x_2 < 23.0$	$23.0 \leq x_2 < 23.75$

b_{13}	b_{14}
$23.75 \leq x_2 < 24.5$	$24.5 \leq x_2 < 25.25$

Figure 4: Chosen Cut Points Data set S^{n+1}



X	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}	b_{11}	b_{12}	b_{13}	b_{14}	Tag
a	1	0	1	1	1	1	1	1	1	0	0	0	0	0	G1
b	1	0	1	1	1	1	0	0	0	0	0	1	0	0	G1
c	1	1	0	1	1	1	1	1	0	0	0	0	0	1	G1
e	0	0	0	1	1	1	1	0	0	0	0	0	1	0	G2
f	0	0	0	1	1	1	1	1	1	0	0	0	0	0	G2
g	1	1	0	0	0	0	0	0	0	0	0	0	0	0	G2
h	1	1	0	1	1	0	0	0	0	0	1	0	0	0	G3

Figure 5: Binarized Data set ${S_{bin}}^{n'+1}$

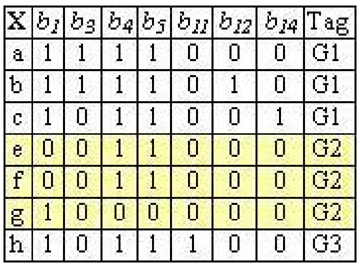
2.3 Support Sets

The explosion of the number of dimensions from data set S^{n+1} to ${S_{bin}}^{n'+1}$ becomes a hindrance to the efficient generation of optimal theory set for each cluster. Furthermore, we can expect that there exists a number of binary dimensions in ${S_{bin}}^{n'+1}$ which may not be necessary in distinguishing one cluster from another. Thus LAD develops the use of support sets[3]. A support set is a set obtained by eliminating a number of binary dimensions from ${S_{bin}}^{n'+1}$ so as to obtain a *contradiction-free* data set, i.e. there does not exist an observation in more than one cluster in the data set ${S_{bin}}^{n'+1}$.

By inspection, we see that we can eliminate the dimensions $\{b_2, b_6, b_7, b_8, b_9, b_{10}, b_{13}\}$ so as to obtain a contradiction - free set from the data set in Figure 5. Notice that we have $SUP = \{b_1, b_3, b_4, b_5, b_{11}, b_{12}, b_{14}\}$ so that every observation $X_a \in P_m$ has a Hamming distance of at least 1 from all other observations $Y_b \in {S_{bin}}^{n'+1} \backslash P_m$. We define the Hamming distance for two observations as the number of positions wherein their components have different values. For example, the observations $a \in P_{G1}$, and $g \in P_{G2}$ differs in the components corresponding to the dimensions b_5, b_6, b_8, b_{10}, b_{11}, and b_{12}, thus their Hamming distance is equal to 6.

In other words, using this set assures us of at least one dimension which differentiates an observation $X_a \in P_m$ from all other clusters where X_a is not a member thereof. The choice of this set induces the disjointness (crispness) of the theories across all clusters in the data set. This set is a support set of ${S_{bin}}^{n'+1}$. We can actually obtain another support set be adding the dimension x_{13} to SUP, i.e. $SUP = \{b_1, b_3, b_4, b_5, b_{11}, b_{12}, b_{13}, b_{14}\}$. Notice that if we are to eliminate the dimensions in this solution (as well as the component values they represent) from ${S_{bin}}^{n'+1}$, the vectors $\{a, f\}$ in the remaining data set would be the same, thus, contradiction arises.

Suppose we take $SUP = \{b_1, b_3, b_4, b_5, b_{11}, b_{12}, b_{14}\}$ from our table in Figure 5, we form ${S_{sup}}^{|SUP|+1}$ with the following vectors as shown in Figure 6.



X	b_1	b_3	b_4	b_5	b_{11}	b_{12}	b_{14}	Tag
a	1	1	1	1	0	0	0	G1
b	1	1	1	1	0	1	0	G1
c	1	0	1	1	0	0	1	G1
e	0	0	1	1	0	0	0	G2
f	0	0	1	1	0	0	0	G2
g	1	0	0	0	0	0	0	G2
h	1	0	1	1	1	0	0	G3

Figure 6: ${S_{sup}}^{|SUP|+1}$

We see that we can generate a theory $\tau_{P_{G1}} = b_1 b_3 b_4 \vee \overline{b_{11}} b_{14}$.

If we are to associate back the literals of the patterns in $\tau_{P_{G1}}$, we find that the patterns covering the observations in the cluster P_{G1} would be

$$\{(x_1 \geq 2.0)(2.0 \leq x_1 < 4.25)(x_2 \geq 21.5),$$

$$\overline{(22.0 \leq x_2 < 23.0)}(24.5 \leq x_2 < 25.25)\}.$$

2.4 Pareto-optimality of Patterns

Generating exhaustively for patterns to comprise the theory set of a data set requires a lot of computational resources. This is still true even when we have used the support set in finding these patterns. There exists an exponential number of them. At times, some of these patterns may not be necessary when they can be represented by other patterns within their own theory.

The concept of finding Pareto-optimal patterns is discussed in [6]. In the same paper, a criteria is set to find these special sets of patterns categorized into three namely:

- Strong Patterns
- Spanned Patterns
- Strong Spanned Patterns

We discuss how we eliminate some patterns in a given cluster's theory by the criteria set in [6].

2.4.1 *Strong Patterns*

Two criteria introduced in [6] relates to this type of pattern namely (a) Simplicity Preference, and (b) Evidential Preference. We discuss each one of them in the following subsections.

Simplicity Preference

DEFINITION 4. *A pattern Tr_1 is simplicity-wise preferred against another pattern Tr_2 (expressed as $Tr_1 \succeq_\sigma Tr_2$) if and only if the literals of Tr_1 (denoted by $Lit(Tr_1)$) is contained in the literals of Tr_2(resp. $Lit(Tr_2)$), i.e. $Lit(Tr_1) \subseteq Lit(Tr_2)$.*

The pattern Tr_1 is *prime* if there does not exist a pattern Tr_j with which $Lit(Tr_1) \supset Lit(Tr_j)$. Naturally, prime patterns are simplicity-preferred to all other patterns in a given theory of a cluster.

Using the data set in Figure 6, we see that the patterns $Tr_1 = b_1 b_3$ is simplicity-prefered against $Tr_2 = b_1 b_3 b_4 \overline{b_{11}}$. Furthermore, we see that Tr_1 also a prime pattern.

Evidential Preference

DEFINITION 5. *A pattern Tr_1 is evidential-wise preferred to another pattern Tr_2 (expressed as $Tr_1 \succeq_\epsilon Tr_2$)) if and only if the covered observations of Tr_2 (denoted as $Cov(Tr_2)$) also exists in the coverage of Tr_1 (resp. $Cov(Tr_1)$), i.e. $Cov(Tr_1) \supseteq Cov(Tr_2)$.*

A pattern Tr_1 is called *strong* if and only if there does not exist a pattern Tr_j with which $Tr_j \succeq_\epsilon Tr_1$.

In Figure 6 for instance, if $Tr_1 = \overline{b_3 b_{11} b_{14}}$, then $Cov(Tr_1) = \{e, f, g\}$. On the other hand, if $Tr_2 = \overline{b_1} b_4$, then $Cov(Tr_2) = \{e, f\}$. Thus, $Tr_1 \succeq_\epsilon Tr_2$.

Generally, note that prime patterns are strong patterns, i.e.

$$Tr_1 \succeq_\sigma Tr_2 \Longrightarrow Tr_1 \succeq_\epsilon Tr_2.$$

2.4.2 *Spanned Patterns*

When we increase the degree of a given pattern, i.e. the number of literals composing it, the more restrictions we put on the behavior describing the cluster where this pattern is a member of its theory. This provides us of a finer characterization of the cluster with respect to other clusters in the data set. We discuss these type of pattern known as Spanned Patterns introduced in [6].

Selectivity Preference Let Tr be a term. A $SPAN(Tr) = \{Sp = \{X_1, X_2, ...\} | Sp \subset \{0,1\}^{|SUP|}$ for which Tr covers $X_i, \forall i\}$.

DEFINITION 6. *A pattern Tr_1 is selective-wise preferred to a pattern Tr_2 (expressed as $Tr_1 \succeq_\Sigma Tr_2$) if and only if $SPAN(Tr_1) \subseteq SPAN(Tr_2)$.*

We see that the set of minterm patterns, i.e. patterns of degree SUP, are selective-wise against all other patterns in the theory of a cluster. However, these may not be useful in describing the cluster since it is specific to only one observation in it. One may find this set of patterns helpful in the detection of exceptions or outliers in the cluster.

Note that if $Tr_1 \succeq_\sigma Tr_2 \Longrightarrow Tr_2 \succeq_\Sigma Tr_1$. Selectivity-wise preferred patterns have been used in combination of other types of patterns in the literatures of LAD [6] to create another type of patterns to achieve better characterization of a given cluster.

To generate the minterm patterns of a cluster in our data set ${S_{sup}}^{|SUP|+1}$, the process is straightforward. Note that since this data set is contradiction-free, we do not create a minterm which covers an observation found in more that 1 cluster. Thus, we still generate a crisp theory set.

We say that the minterm theory τ_{P_m} is the disjunction of terms Tr_i of degree $|SUP|$ covering the i^{th} observation of the cluster P_m of the data set ${S_{sup}}^{|SUP|+1}$.

We use the selectivity preference in conjunction with the evidential preference to characterize spanned patterns. Notice that in the succeeding discussions, we discuss the conjunctions and refinements done by the methodologies of LAD to the set of preferences to produce other types of patterns.

Definition 7. *Given preferences ϵ (evidential) and Σ (selectivity), a pattern Tr_1 is preferred to a pattern Tr_2 with respect to the intersection $\epsilon \wedge \Sigma$ (expressed as $Tr_1 \succeq_{\epsilon\wedge\Sigma} Tr_2$) if and only if $Tr_1 \succeq_{\epsilon} Tr_2$ and $Tr_1 \succeq_{\Sigma} Tr_2$. The patterns which are Pareto-optimal with respect to $\epsilon \wedge \Sigma$ are called spanned patterns.*

For our table in Figure 6, let the patterns (for some observations in P_{G1}) $Tr_1 = b_1 b_4 b_5 \overline{b_{11}}$ and $Tr_2 = b_1 b_4$. Here, $Cov(Tr_1) = \{a, b, c\} = Cov(T_2)$ and $Lit(Tr_1) \supset Lit(Tr_2)$. Thus, $Tr_1 \succeq_{\epsilon\wedge\Sigma} Tr_2$. Futhermore, there does not exist a pattern Tr_3 (covering an observation in P_{G1}) with which $Tr_3 \succeq_{\epsilon\wedge\Sigma} Tr_1$.

2.4.3 Strong Spanned Patterns

Definition 8. *Given preferences ϵ and Σ, a pattern Tr_1 is preferred to a pattern Tr_2 with respect to the lexicographic refinement $\epsilon|\Sigma$ (expressed as $Tr_1 \succeq_{\epsilon|\Sigma} Tr_2$) if and only if $Tr_1 \succ_{\epsilon} Tr_2$ or ($Tr_1 \approx_{\epsilon} Tr_2$ and $Tr_1 \succeq_{\Sigma} Tr_2$). The patterns which are Pareto-optimal with respect to $\epsilon|\Sigma$ are called strong spanned patterns.*

Note that for any preference χ, if a pattern $Tr_1 \succeq_{\chi} Tr_2$ and $Tr_2 \succeq_{\chi} Tr_1$, then $Tr1 \approx_{\chi} Tr_2$. On the other hand, if $Tr_1 \succeq_{\chi} Tr_2$ and $Tr_2 \not\succeq_{\chi} Tr_1$, then $Tr_1 \succ_{\epsilon} Tr_2$.

The definition above tells us that a pattern is strong spanned if and only if it is both strong and spanned.

3. THE ACTSF ALGORITHM

3.1 Support Set Generation

We first describe the input to the problem of finding the support set for the binarized data set ${S_{bin}}^{n'+1}$. Then, we present a greedy algorithm geared to provide a solution to this problem for k-tagged data sets.

1. To generate a contradiction free data set, intuitively, we decide to include a dimension b_j of the data set ${S_{bin}}^{n'+1}$ if there exists an attribute value in this dimension which possess a unique behavior describing the cluster where it belongs with respect to others belonging to another cluster. Our original data set S^{n+1} has been transformed into a binary data set ${S_{bin}}^{n'+1}$ where $n' >> n$, so the possibility of finding these dimensions becomes larger. However, if no such dimension exists, we iteratively attempt to find another dimension b_h that, together with our previous choices, forms a contradiction-free set. This problem is particularly hard since there are $2^{n'}$ possible solutions to it.

However, [3] has reduced this to the well known NP-complete set cover problem. Revising the original method of LAD to accomodate k-tagged data sets, $k \geq 2$, we transform in polynomial time the original input $S^{n'+1}$ to a data set $COMP^{n'+1}$ whose dimensions $D_j, j \in \{1, 2, ..., n'\}$ is associated to every binary dimension b_j of ${S_{bin}}^{n'+1}$. We call each D_j an *indicator dimension* (adopted from [3]) of b_j. Every D_j "indicates" whether b_j is to be included in the support set (i.e. $D_j = 1$), or not (i.e. $D_j = 0$).

Input: ${S_{bin}}^{n'+1}$. Let b_j be the binary dimensions of ${S_{bin}}^{n'+1}$. Let $P_m (\subset {S_{bin}}^{n'+1}), 2 \leq m \leq k$.

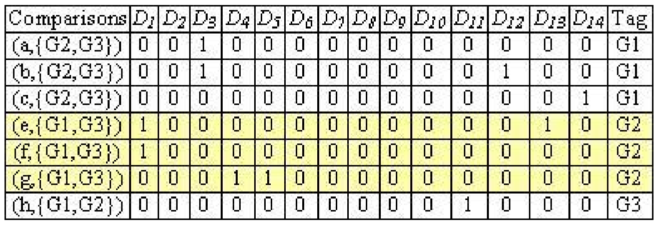
Goal: Generate a binary data set $COMP^{n'+1}$ with indicator dimensions $D_j, j \in \{1, 2, ..., n'\}$

Let the data set $COMP^{n'+1}$ contain the observations $X'_i = (D_{i1}, D_{i2}, ..., D_{in'}, m)$, $m \in T$.

For each observation $Y_a = (y_{a1}, y_{a2}, ..., y_{an'}, r) \in P_r$ $(\subset {S_{bin}}^{n'+1})$, and for all vectors $Z_b = (z_{b1}, z_{b2}, ..., z_{bn'}, s) \in {S_{bin}}^{n'+1} \backslash P_r$, where $s \in T \backslash r$, let D_{ij} be

$$D_{ij} = \begin{cases} 1, & \text{if } y_{aj} \neq z_{bj}, \\ 0, & \text{otherwise.} \end{cases} \qquad (3.1.1.\text{a})$$

For the example indicated in Figure 5, ${S_{bin}}^{n'+1}$ into the data set in Figure 7.



Comparisons	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}	D_{11}	D_{12}	D_{13}	D_{14}	Tag
(a,{G2,G3})	0	0	1	0	0	0	0	0	0	0	0	0	0	0	G1
(b,{G2,G3})	0	0	1	0	0	0	0	0	0	0	0	1	0	0	G1
(c,{G2,G3})	0	0	0	0	0	0	0	0	0	0	0	0	0	1	G1
(e,{G1,G3})	1	0	0	0	0	0	0	0	0	0	0	0	1	0	G2
(f,{G1,G3})	1	0	0	0	0	0	0	0	0	0	0	0	0	0	G2
(g,{G1,G3})	0	0	0	1	1	0	0	0	0	0	0	0	0	0	G2
(h,{G1,G2})	0	0	0	0	0	0	0	0	0	0	1	0	0	0	G3

Figure 7: Data set $COMP^{n'+1}$ of indicator dimensions D_j

To be able to solve the set cover problem, we adapt the following Integer Program (IP) from [3].

$$\text{minimize } \sum_{j=1}^{n'} D_j$$
$$\text{s.t. } \sum_{j=1}^{n'} D_{ij} \geq 1, \forall i$$

Then, the set of dimensions D_j satisfying the IP will be a solution SUP_{Inds} to the set cover problem. Note that by solving this IP, we generate the so-called *irredundant* support set. A support set is called irredundant if no proper subset of it is a support set[3]. The value of the right-hand side of the constraints tells us the number of dimensions used to discriminate the clusters (and theory sets) of the data set. With the IP above, we generate the least number of these dimensions from the original data set. The problem here lies in the fact that users may want to know more about the data, thus it would be natural to increase the right hand side of the IP. If we are to do this, the more cluster-specific our patterns would be for our theory set, the better we can discriminate the behavior among all clusters.

2. The greedy algorithm presented below is built on the idea of iteratively finding a new set of binary indicator dimensions which can form a bigger support set by starting with the irredundant support set. Implicitly, we try to increase the right hand side of the constraint in every iteration. The reason for this is to increase the degrees of the patterns which will be used to discriminate the clusters from among themselves. Implicitly, we attempt to lessen the number of false positives when we evaluate new sets of untagged data. In finding a new binary dimension to be added to the support set, we added an evaluation function which measures the degree of importance of this dimension against other dimensions. The dimension with the highest contribution in discriminating the clusters is the one added to the support set. This assures us that we choose the set of the most significant binary dimensions in the data set. We take advantage of the transformation we did in (*3.1.1.*a) to accelerate the pace of generating these dimensions. Apparently, the ACTSF algorithm helps us in finding high-degree patterns in lesser computational resources.

Note that we do not include infeasible constraints in the process of finding the support set of the data set, i.e. those X_i' whose components are equal to 0.

Greedy Algorithm:

Input: A binary data set $COMP^{n'+1}$ with indicator dimensions $D_j, j \in \{1, 2, ..., n'\}$. Let the observations $X'_i = (D_{i1}, D_{i2}, ..., D_{in'}, m)$, $m \in T$, $i = 1, 2, \dots, |COMP^{n'+1}|$.

Output: Generate a support set for ${S_{bin}}^{n'+1}$.

```
1 Set $FC = |COMP^{n'+1} \backslash X'_i|$, $X'_i \in COMP^{n'+1}$,
    $X'_i = (0, 0, \dots, 0)$, $i \in \{1, 2, \dots, |COMP^{n'+1}|\}$.

2 Set $SUP_{Inds} = \emptyset$.
3 Set $SUP = \emptyset$.
4 Set $SUP_{temp} = \emptyset$.

5 For all $D_j \notin SUP_{Inds} \cup SUP_{temp}$ do
      Set $F_j = \sum_{i=1}^{|COMP^{n'+1}|} D_{ij}$.

6 For all $D_j \notin SUP_{Inds} \cup SUP_{temp}$ do
      Set $CovConstr(D_j) = \{i | D_{ij} = 1, i \in \{1, 2, \dots, |COMP^{n'+1}|\}\}$.
      Set $V_{D_j} = CovConstr(D_j) \backslash (CovConstr(SUP_{temp}) \cap CovConstr(D_j))$,
          $D_j \notin SUP_{Inds} \cup SUP_{temp}$.

7 Set $V^{max} = MAX_{D_j \notin SUP_{Inds} \cup SUP_{temp}} V_{D_j}$.
8 Set ${D_j}^{max} = D_j$ where $V_{D_j} = V^{max}$.
9 If $V^{max} > 0$ then
      Set $SUP_{temp} = SUP_{temp} \cup {D_j}^{max}$.
      If $|\bigcup_{D_j \in SUP_{temp}} D_j| = FC$ then
          Set $SUP_{Inds} = SUP_{Inds} \cup SUP_{temp}$.
      Go to Step 4.
10 Else
      Set $SUP = \{b_j | \exists D_j \in SUP_{Inds}\}$.
      Output $SUP$.
      Stop.
```

Note that at the end of the greedy algorithm SUP will contain some binary dimensions $b_j, j \in \{1, 2, ..., n'\}$ and SUP_{Inds} will contain the indicator dimensions D_j corresponding to the binary dimensions in SUP.

For our example in Figure 7, $SUP_{Inds} = \{D_1, D_3, D_4, D_{11}, D_{14}\}$ with the right hand side of the constraints equal to 1 only. Thus $SUP = \{b_1, b_3, b_4, b_{11}, b_{14}\}$.

To be able to prove that the output of this algorithm, i.e. SUP is indeed a support set of the original data set ${S_{bin}}^{n'+1}$, we construct the data set ${S_{sup}}^{|SUP|+1}$ whose dimensions $b_j \in SUP$ and prove that it is contradiction-free. The values for this data set would correspond to the column values in every chosen b_j from ${S_{bin}}^{n'+1}$. Note that with the elimination of the infeasible constraints in the evaluation of SUP, we also do not include their corresponding vectors (observations) in the construction of ${S_{sup}}^{|SUP|+1}$.

Let ${S_{sup}}^{|SUP|+1}$ be the data set whose observations $X_i = (b_{ij}, m), b_j \in SUP$, $m \in T$. Note that we do not include in this data set those vectors in ${S_{bin}}^{n'+1}$ corresponds to a zero vector in $COMP^{n'+1}$.

Let $y^{gP_m} = \{D_i | D_i \in SUP_{Inds}$ in the g^{th} constraint corresponding to the comparison of $(X, \{Y\})$, $X \in P_m, Y \in {S_{bin}}^{n'+1} \backslash P_m$. Note that $|y^{gP_m}| \geq 1$.

Let $y^{P_m} = \bigcup_{g=1}^{|P_m|} y^{gP_m}$.

Using the transformation 3.1.1.a, we see that $\nexists D_j \in y^{gP_r}$ and $D_j \in y^{hP_s}$, $\forall g, h$, and $s \in T \backslash r$. It follows that

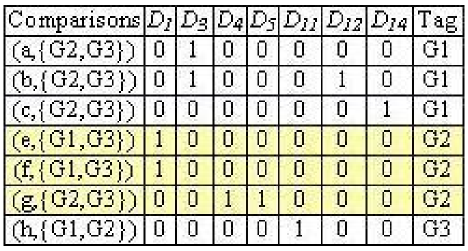
$$\bigcap_{i \in \{1, 2, \dots, |T|\}} y^{P_i} = \emptyset.$$

Therefore, $\exists D_j \in y^{P_r}$ for which the j^{th} component of the vector $X \in P_r$ is not equal to the j^{th} component of the vector $Y \in P_s$, where $P_r, P_s \in {S_{sup}}^{|SUP|+1}$. Hence the data set ${S_{sup}}^{|SUP|+1}$ is contradiction-free.

3.2 Pattern Generation in ACTSF

Let us recall the construction of y^{gP_m} in Section 3.1. We use the data set ${S_{sup}}^{|SUP|+1}$ in Figure 6. (Note that the data set is constructed from the choice of $SUP = \{b_1, b_3, b_4, b_5, b_{11}, b_{12}, b_{14}\}$).

We extract from Figure 7 the dimensions which correspond to the elements of SUP as shown in Figure 8. Let this data set be ${COMP_{SI}}^{|SUP_{Inds}|+1}$.



Comparisons	D_1	D_3	D_4	D_5	D_{11}	D_{12}	D_{14}	Tag
(a,{G2,G3})	0	1	0	0	0	0	0	G1
(b,{G2,G3})	0	1	0	0	0	1	0	G1
(c,{G2,G3})	0	0	0	0	0	0	1	G1
(e,{G1,G3})	1	0	0	0	0	0	0	G2
(f,{G1,G3})	1	0	0	0	0	0	0	G2
(g,{G2,G3})	0	0	1	1	0	0	0	G2
(h,{G1,G2})	0	0	0	0	1	0	0	G3

Figure 8: $COMP_{SI}{}^{|SUP_{Inds}|+1}$

DEFINITION 9. *A primality dimension set* $y^{g_{P_m}}$ *is the maximum set of indicator dimensions* D_i, $i \in \{1, 2, ..., |SUP_{Inds}|\}$ *of* $COMP_{SI}{}^{|SUP_{Inds}|+1}$ *whose vector* $X'_g = (d_{g_1}, d_{g_2}, ..., d_{g_{|SUP_{Inds}|}}, m)$, $m \in T$ *have* $d_{g_j} = 1$.

Below is the listing of $y^{g_{P_{G1}}}, g \in \{1, 2..., |P_{G1}|\}$.

- $y^{1_{P_{G1}}} = \{D_3\}$ *(5.1)*
- $y^{2_{P_{G1}}} = \{D_3, D_{12}\}$
- $y^{3_{P_{G1}}} = \{D_{14}\}$

Let $y_j{}^{P_m} = \{y^{g_{P_m}} | \forall g, h : y^{g_{P_m}} \cap y^{h_{P_m}} \neq \emptyset\}$, $j \in \{1, 2, ..., |P_m|\}$.

Let $LS_j = \bigcap_{h=1}^{|y_j{}^{P_m}|} s_h$, where $s_h \subseteq y_j{}^{P_m}$, and cluster $P_m \subset COMP_{SI}{}^{|SUP_{Inds}|+1}$. Let $LS'_j = \{b_i | \forall i\ \exists D_i \in LS_j\}$.

For our example (5.1), we see that a common value exists in $y^{g_{P_{G1}}}, g \in \{1, 2\}$ such that $y_1{}^{P_{G1}} =\{\ \{D_3\}, \{D_3, D_{12}\}\}$. Then, $LS_1 = \{D_3\}$ and $LS'_1 = \{b_3\}$. Furthermore, we also form $y_2{}^{P_{G1}} = \{D_{14}\}$. Here, $LS_2 = \{D_{14}\}$ and $LS'_2 = \{b_{14}\}$.

OBSERVATION 1. *Let a pattern* $Tr = \wedge b_i{}^{\alpha_i}$, $\forall i\ \exists D_i \in |SUP_{Inds}|$. *Tr is strong, if there exists* $j \in \{1, 2, ..., |P_m|\}$ *where* $Lit(Tr) = LS'_j$.

Below is the pseudocode to generate strong patterns for a cluster $P_m \subset S_{bin}^{n'+1}$ based on the construction of ACTSF given above.

Pseudocode.
Input: Data set $COMP_{SI}{}^{|SUP_{Inds}|+1}$ of indicator dimensions D_j derived from the binary data set $S_{bin}{}^{n'+1}$.
Output: The set of strong patterns for $S_{bin}{}^{n'+1}$.

```
Set $g = 1$.
For each $X_i = (D_{ij}, m) \in P_m, \forall j : \exists D_j \in SUP_{Inds}$
    For $j = 1$ to $|SUP_{Inds}|$
        If $D_{ij} = 1$ then
            Set $y^{g_{P_m}} = y^{g_{P_m}} \cup D_j$.
    Increment $g$.

For $g = 1$ to $|P_m|$
    Set $y^{P_m} = y^{P_m} \cup y^{g_{P_m}}$.

Set $q = 1$.
For $c = 1$ to $g$
    Set $y_q{}^{P_m} = y^{c_{P_m}}$.
    For $d = 2$ to $g$
        If $y_q{}^{P_m} \cap\ y^{d_{P_m}} \neq \emptyset\ AND$
            $y^{d_{P_m}} \in y^{P_m}$ then
            Set $y_q{}^{P_m} = y_q{}^{P_m} \cup y^{d_{P_m}}$.
            Set $y^{P_m} = y^{P_m} \backslash y^{d_{P_m}}$.
    Increment $q$.
    If $y^{c+1_{P_m}} \notin\ y^{P_m}$ then
        Increment $c$.

For $e = 1$ to $q$
    For each $S \in y_e{}^{P_m}$
        Set $LS_e = LS_e \cap S$.

For $e = 1$ to $q$
    For each $D_j \in LS_e$
        Set $LS'_j = LS'_j \cup b_j$.

For $e = 1$ to $q$
    For each $b_j \in LS'_e$
        If $D_{ij} = 1$ then
            Set $Tr_e{}^{P_m} = Tr_e{}^{P_m} \wedge b_j$.
        Else
            Set $Tr_e{}^{P_m} = Tr_e{}^{P_m} \wedge \overline{b_j}$.
```

At the end of the pseudocode, we generate the strong patterns $Tr_e, e \geq 1$ for P_m.

Suppose we have take the pattern $Tr = b_3$ for the cluster P_{G1} from Example 5.1, since the $Lit(Tr) = LS'_1$, therefore Tr is strong.

Next, we characterize spanned patterns in the context of our ACTSF algorithm.

DEFINITION 10. *A spanning dimension set* $z^{h_{P_m}}$, $h \leq 2^{|SUP_{Inds}|} - 1$, *is the maximal set of indicator dimensions* $D_i, i \in \{1, 2, ..., |SUP_{Inds}|\}$ *of* $COMP_{SI}{}^{|SUP_{Inds}|+1}$ *whose vector components* D_{ij}, $j \in \{1, 2, ..., |P_m|\}$ *are equal for some* $\pi_c \subseteq P_m$.

From Figure 8, we see that

- $z^{1_{P_{G1}}} = \{D_1, D_3, D_4, D_5, D_{11}, D_{14}\}$ *(5.2)*
  - accounting observations $\{a, b\}$
- $z^{2_{P_{G1}}} = \{D_1, D_4, D_5, D_{11}, D_{12}\}$
  - accounting observations $\{a, c\}$
- $z^{3_{P_{G1}}} = \{D_1, D_4, D_5, D_{11}, D_{14}\}$
  - accounting observations $\{b, c\}$
- $z^{4_{P_{G1}}} = \{D_1, D_4, D_5, D_{11}\}$
  - accounting observations $\{a, b, c\}$
- $z^{5_{P_{G1}}} = \{D_1, D_2, D_3, D_4, D_5, D_{11}, D_{14}\}$
  - accounting observation $\{a\}$

- ${z^6}_{P_{G1}} = \{D_1, D_2, D_3, D_4, D_5, D_{11}, D_{14}\}$
  - accounting observation $\{b\}$
- ${z^7}_{P_{G1}} = \{D_1, D_2, D_3, D_4, D_5, D_{11}, D_{14}\}$
  - accounting observation $\{c\}$

Let $X_i = (b_{i1}, b_{i2}, ..., b_{i|SUP|+1}, m) \in {S_{sup}}^{|SUP|+1}$.

Let $TrElems^{h_{P_m}} = \{{b_i}^{\alpha_i} | \forall i, \exists D_i \in z^{h_{P_m}}, \alpha_i = b_{ij}\}$.

In example 5.2,

- $TrElems^{1_{P_{G1}}} = \{b_1, b_3, b_4, b_5, \overline{b_{11}}, \overline{b_{14}}\}$
- $TrElems^{2_{P_{G1}}} = \{b_1, b_4, b_5, \overline{b_{11}}, \overline{b_{12}}\}$
- $TrElems^{3_{P_{G1}}} = \{b_1, b_4, b_5, \overline{b_{11}}, \overline{b_{14}}\}$
- $TrElems^{4_{P_{G1}}} = \{b_1, b_4, b_5, \overline{b_{11}}\}$
- $TrElems^{5_{P_{G1}}} = \{b_1, b_3, b_4, b_5, \overline{b_{11}}, \overline{b_{12}}, \overline{b_{14}}\}$
- $TrElems^{6_{P_{G1}}} = \{b_1, b_3, b_4, b_5, \overline{b_{11}}, b_{12}, \overline{b_{14}}\}$
- $TrElems^{7_{P_{G1}}} = \{b_1, \overline{b_3}, b_4, b_5, \overline{b_{11}}, \overline{b_{12}}, b_{14}\}$

OBSERVATION 2. *The set of patterns*

$$ {Tr_h}^{P_m} = \bigwedge_{j=1}^{|TrElems^{h_{P_m}}|} b_j, $$

where $b_j \in TrElems^{h_{P_m}}$ *is spanned.*

Below is the pseudocode to generate spanned patterns for a cluster $P_m \subset S_{bin}^{n'+1}$ based on the construction of ACTSF given above.

Pseudocode.
Input: Data set ${COMP_{SI}}^{|SUP_{Inds}|+1}$ of indicator dimensions D_j derived from the binary data set ${S_{bin}}^{n'+1}$.
Output: The set of spanned patterns for ${S_{bin}}^{n'+1}$.

```
Set $D = \{D_j |\ D_j \in SUP_{Inds}\}$.
Set $Pow(D)$ to be the power set of $D$.
For $c = |D|$ to 1
    For each $R \in Pow(D)\ AND\ |R| = c$
        Set $Q = \{b_j | \exists D_j \in R\}$.
        Set $\overline{Q} = \{\overline{b_j} | \exists b_j \in Q\}$.
        Set $Q' = Q \cup \overline{Q}$
        Set $TRS_c$ be the set of patterns of $P_m$ of
            degree $c$ with literals from $Q'$.
        If $Cov(Tr_1) = Cov(Tr_2)$, where $Tr_1 \in TRS_c$,
            $Tr_2 \in TRS_d$, $c < d$ then
            Set $TRS_c = TRS_c \backslash Tr_1$.

For $c = |D|$ to 1
    For each $S \in TRS_c$
        For $j \in SUP_{Inds}$
            If $b_j \in Lit(S)\ OR\ \overline{b_j} \in Lit(S)$ then
                Set $z^{h_{P_m}} = z^{h_{P_m}} \cup D_j$.
        Increment $h$.

For $e = 1$ to $h$
    For each $D_j \in z^{e_{P_m}}$
        If $D_{ij} = 1$ in the accounted observation(s) of
            $z^{e_{P_m}}$ then
            Set $TrElems^{e_{P_m}} = TrElems^{e_{P_m}} \cup b_j$.
        Else
            Set $TrElems^{e_{P_m}} = TrElems^{e_{P_m}} \cup \overline{b_j}$.

For $e = 1$ to $h$
    For each $b_j \in TrElems^{e_{P_m}}$
        If $D_{ij} = 1$ then
            Set ${Tr_e}^{P_m} = {Tr_e}^{P_m} \wedge b_j$.
        Else
            Set ${Tr_e}^{P_m} = {Tr_e}^{P_m} \wedge \overline{b_j}$.
```

At the end of the pseudocode, we generate the spanned patterns ${Tr_e}^{P_m}, e \geq 1$ for P_m.

Finally, we character strong spanned patterns in the context of our ACTSF algorithm. We use the construction of $TrElems^{h_{P_m}}$ to arrive at this set of patterns.

Let ${w_j}^{P_m} = \{TrElems^{g_{P_m}} | \forall g, h : TrElems^{g_{P_m}} \cap TrElems^{h_{P_m}} \neq \emptyset\}$, $j \leq 2^{|SUP_{Inds}|} - 1\}$.

Let ${LSP_j}^{P_m} = \bigcap_{h=1}^{|{w_j}^{P_m}|} s_h$, where $s_h \subseteq {w_j}^{P_m}$.

Using the above example, we have ${w_1}^{P_{G1}}$ as the intersection of all $TrElems^{h_{P_m}}, \forall h$. Therefore the value of ${LSP_1}^{P_{G1}} = \{b_1, b_4, b_5, \overline{b_{11}}\}$.

OBSERVATION 3. *The set of patterns*

$$ {Tr_h}^{P_m} = \bigwedge_{j=1}^{|{LSP_j}^{P_m}|} b_j, $$

$\forall b_j \in {LSP_j}^{P_m}$ *is strong spanned.*

For the data set in Figure 6, we realize that $b_1 b_4 b_5 \overline{b_{11}}$, the pattern generated using ${LSP_1}^{P_{G1}}$, is strong spanned. Note that it covers every observation in P_{G1}.

Below is the pseudocode to generate strong spanned patterns for a cluster $P_m \subset S_{bin}^{n'+1}$ based on the construction of ACTSF given above.

Pseudocode.
Input: $TrElems^{h_{P_m}}$.
Output: The set of strong spanned patterns for ${S_{bin}}^{n'+1}$.

```
/*We use the value of $h$ in the previous pseudocode.*/
For $e = 1$ to $h$
    Set $w^{P_m} = w^{P_m} \cup TrElems^{e_{P_m}}$.
Set $q = 1$.
For $c = 1$ to $h$
    Set ${w_q}^{P_m} = TrElems^{c_{P_m}}$.
    For $d = 2$ to $h$
        If ${w_q}^{P_m} \cap TrElems^{d_{P_m}} \neq \emptyset$ AND
            $TrElems^{d_{P_m}} \in w^{P_m}$ then
            Set ${w_q}^{P_m} = {w_q}^{P_m} \cup TrElems^{d_{P_m}}$.
            Set $w^{P_m} = w^{P_m} \backslash TrElems^{d_{P_m}}$.
    Increment $q$.
    If $TrElems^{c+1_{P_m}} \notin w^{P_m}$ then
        Increment $c$.
For $e = 1$ to $q$
    For each $S \in {w_e}^{P_m}$
        Set $LSP_e = LSP_e \cap S$.

For $e = 1$ to $q$
    For each $b \in LSP_e$
        Set ${Tr_e}^{P_m} = {Tr_e}^{P_m} \wedge b$.
```

At the end of the pseudocode, we generate the strong spanned patterns ${Tr_e}^{P_m}, e \geq 1$ for P_m.

3.3 Extensions to ACTSF

With the construction of the data set of indicator dimensions $COMP^{n'+1}$, the mapping done in 3.1.1.a. sometimes introduces vectors of zeros. This happens when there does not exist a dimension in the data set ${S_{bin}}^{n'+1}$ with which an observation $X \in P_m$ differs from other observations $Y \in {S_{bin}}^{n'+1} \backslash P_m$. We have mentioned in previous sections of infeasible constraints - i.e. an $X'_i = (0) \in COMP^{n'+1}$. Since we cannot possibly add at least one indicator dimension in the i^{th} constraint, we have disregarded these sets of zero vectors from the computation of the support set $|SUP|$. Because of this, the i^{th} vector of ${S_{sup}}^{|SUP|}$ may possibly induce a contradiction to our ${S_{sup}}^{|SUP|}$, i.e. the same vector would exist in more than 1 cluster. Thus, it would be natural that we strike them out when we build our sets of patterns for each cluster in the data set ${S_{sup}}^{|SUP|}$.

We refer back to the definition of theory set defined in LAD. Note that every observation in a cluster of a given space (i.e. the training set) is covered by at least one pattern in the theory that cluster. Since ACTSF strikes out the observations with a zero vector in the data set ${S_{sup}}^{|SUP|+1}$, we have, in effect, striked out the pattern covering these observations. This is the rationale why the proposed algorithm was refered to as an approximation.

It is inevitable then that we increase the number of unclassified observations. However, we may see that it is more advantageous for us to not conclusively put membership to a given observation rather than take the opportunity of erroneously classifying it to another cluster. To do so, we increase the number of false positives for new sets of observations.

To be able to put membership on these unclassified sets of data, we extend ACTSF to include a discriminant function. When the behavior of an observation do not fall explicitly in one of the clusters due to the inherent inseparability of the observations in the data set, this function became very helpful in detecting membership for it. This discriminant function computes for the distance of a given untagged observation $X = (x_1, x_2, ..., x_n)$ with respect to the theory of a given cluster.

Let $\tau_m = \bigvee Tr_i$, where $m \in T, i \geq 1$, and Tr_i is a pattern for the cluster m.

Let Var_{Tr_i} be the number of dimensions considered in the pattern Tr_i.

Let ${V_{x_j}}^{Tr_i} = \{v | v$ is a cutpoint appearing in appearing in the pattern Tr_i associated to the dimension $x_j\}$.

Let the distance between an observation X and a theory τ_m be defined as

$$\Delta(X, \tau_m) = \overset{MIN}{Tr_i \in \tau_m} \left\{ \frac{\sum_{j=1}^{|Var_{Tr_i}|} MIN\{|x_j - v_i|, v_i \in Var_{Tr_i}, \forall i\}}{|Var_{Tr_i}| * \sum_{i=1}^{|\tau_m|} |Cov(Tr_i)|} \right\}$$

It can be seen that we give preference to patterns possessing higher degrees by dividing the distances with $|Var_{Tr_i}|$. Furthermore, it can be seen that the number of covered observations of a given theory is a major factor to the determination of how we classify the data. The larger the number of evidences for the justification of a behavior (i.e. a pattern) the set of observations in the cluster, the more likely we are to group new observations to that cluster.

The membership of X would be the cluster to which $\tau_m, m \in T$ returns the minimum value of $\Delta(X, \tau_m)$.

4. EMPIRICAL TESTING

The data set used for testing the effectiveness of ACTSF was taken from the [5]. It is the famous multivariate Iris data set created by Fisher in 1988. A number of publications have already cited this data set and created models to characterize the clusters in the data set. It is composed of 4 real-valued dimensions namely: (a) sepal length, (b) sepal width, (c) petal length, and (d) petal width all expressed in unit centimeters. It contains 50 observations in each of the 3 classes namely: (1) Iris-setosa, (2) Iris-versicolor, and (3) Iris-virginica, the first one of which is linearly separable from the others, and the last two are linearly inseparable from one another.

We apply 10 10-fold cross validation to get the average performance of the proposed algorithm. This is done by partitioning the data set in 10 equally-sized groups of randomly-selected observations. We construct a training set from the union of 9 groups, and the remaining cluster will be the test set. We then develop a theory set using the training set and measure its performance on the test set. For the succeeding experiments, we choose 1 group from the previous training set to be our new test set. The previous test set will be merged in the new training set. We do this 10 times (folds).

Using ACTSF only (without the discriminant function), we an classified the observations of the test sets correctly 87% of the time. Misclassification reaches 7% and 11% were left unclassified. We already expected that there would be a higher rate of unclassification (as compared to the misclassification) since we had left out of consideration some of the zero vectors in $COMP^{n'+1}$. This average performances are detailed out in Figure 9.

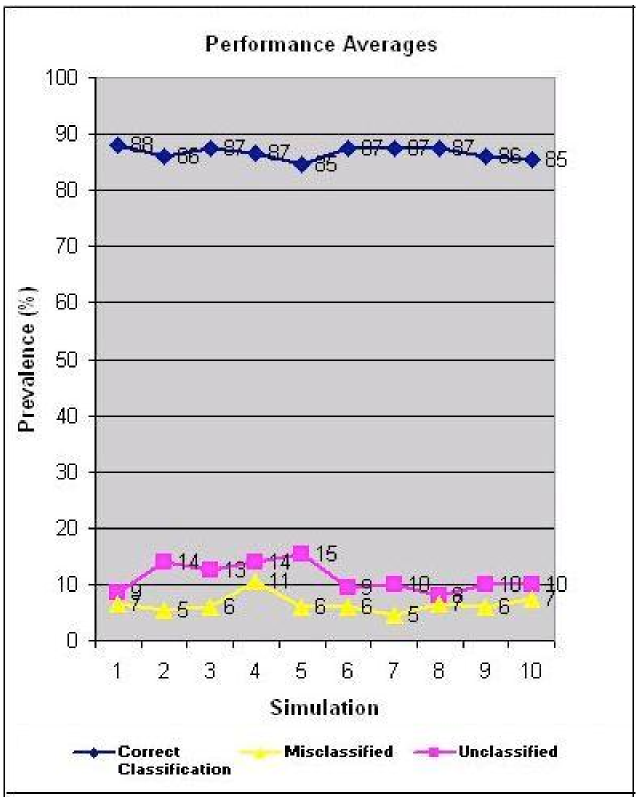


Figure 9: ACTSF Average Performance on Test Set

We achieved a slightly better performance of the algorithm on the training sets. Correct classification rates was 88%. Since we modeled our theory set on the training sets, there were no misclassified observations. Furthermore, unclassified observations reached to 12%. These results are shown in Figure 10.

With the incorporation of $\Delta(X, \tau_m)$ to ACTSF, the classification rate significantly increased to 96% while misclassification rate remained at 7% on the test sets. There were no unclassified observations observed. The average performances are detailed out in Figure 11

For the training sets, correct classification improved to 98%. Still there were no unclassified observations. However, due to the high level of similiarity of the Iris-versicolor and Iris-virginica observations, the discriminant function $\Delta(X, \tau_m)$ failed to correctly classify 2% of the observations on the average. The results are shown in Figure 12.

We present a summary of the results from these literatures and compare it with our algorithm in Table 1.

4.1 Inclusion of Zero Vectors in Finding the Theory Set

For experimental purposes, we report the effect of the inclusion of the vectors in ${S_{sup}}^{|SUP|+1}$ whose vectors in $COMP^{n'+1}$ contain 0 components. Since these vectors were in effect

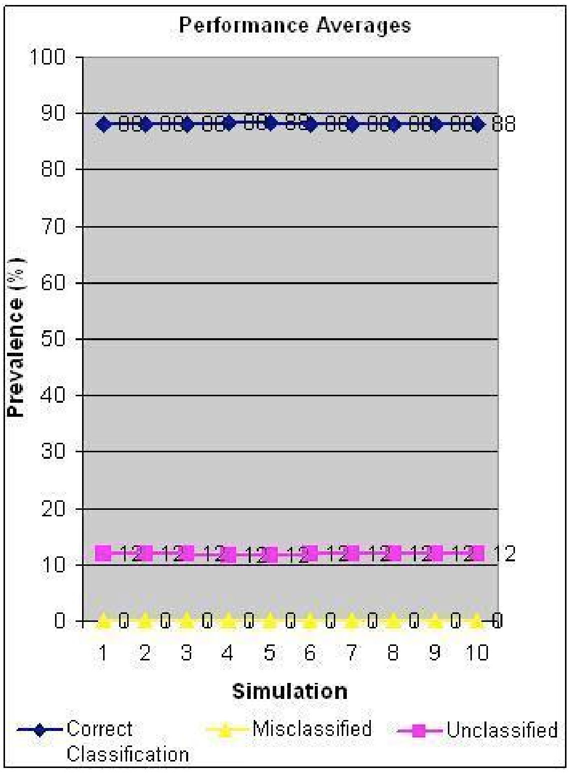


Figure 10: Average Performances of ACTSF on Training Sets

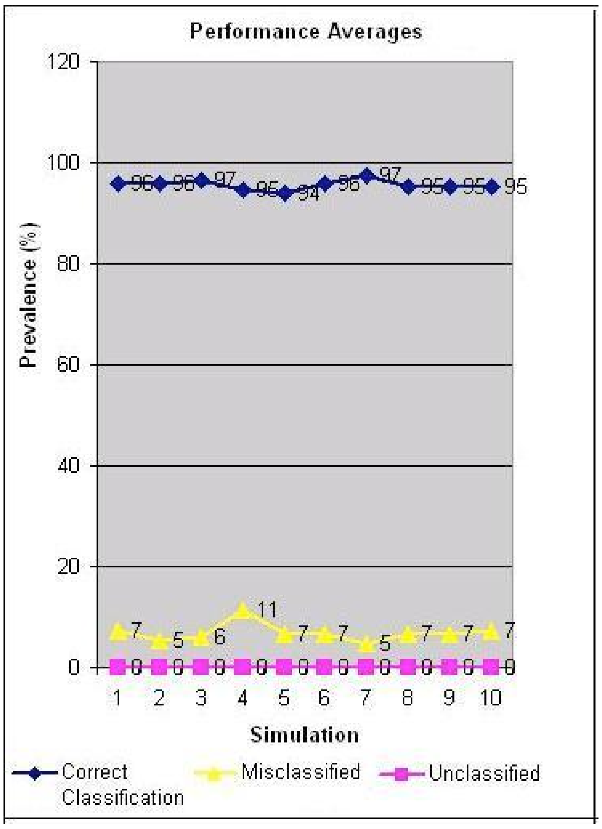


Figure 11: ACTSF with $\Delta(X, \tau_m)$) Average Performance on Test Sets

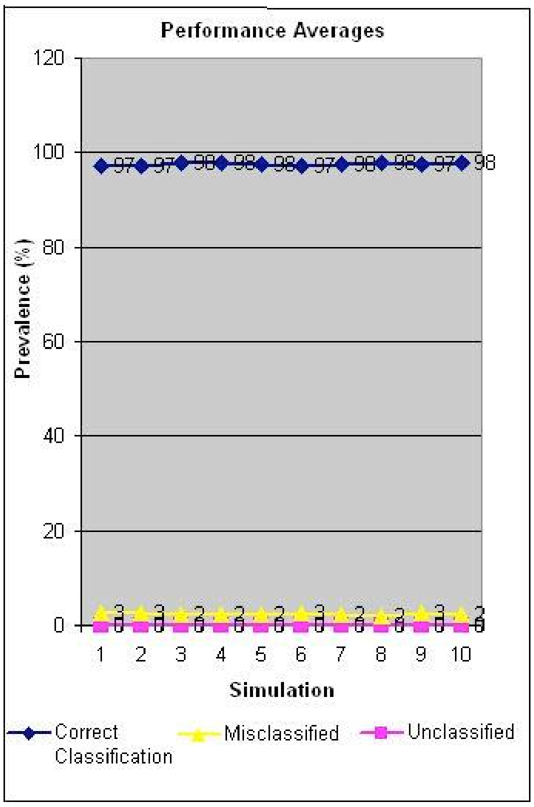


Figure 12: ACTSF with $\Delta(X, \tau_m)$) Average Performance on Training Sets

Algorithm	Performance % [ref]
ACTSF	96
ItemRSFit	83.33 [9]
UChoo	92 [8]
C4.5	90 [8]
PCA-Spikey	70 [10]
SMLP	96 [11]
MLP2LN	97.3 [12]

Table 1: Performance of Different Algorithms on the Iris data set

unaccounted for in the computation of the support set of ${S_{bin}}^{n'+1}$, we can expect that contradiction arises in other clusters with respect to some of these vectors. With this, it is not surprising that our rate of misclassification shot up to 13% in the test sets. A decrease with an average of 1% was reported on the rate of unclassified observations. However, an improvement in the classification rate was seen at 93%. It is noticeable that along with the increase of the percentage of correctly classified observations, the percentage of misclassification did so too.

We see that the inclusion of the aforementioned vectors would only increase the chances of incorrectly marking untagged observations at the expense of trying to correctly classify most of them in the data space.

Note that we compare this performance with ACTSF only, thus we did not apply the discriminant function in determining membership of the observations. The details of this performance is shown in Figure 13.

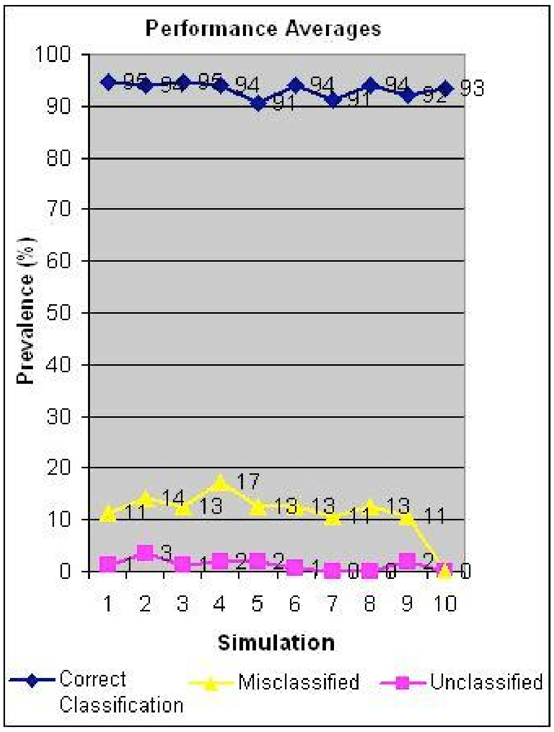


Figure 13: ACTSF Average Performance on Test Sets with inclusion of 0-vectors

5. LIMITS OF APPLICABILITY OF ACTSF

We recall our definition for the linear separability of two sets taken from [3] discussed in Section 2.

The problem of finding linear separability between two sets is equivalent to the disjointness of their corresponding convex hulls [3]. We adapt the definition in [6] for the concept of the convex hull of a binarized data set.

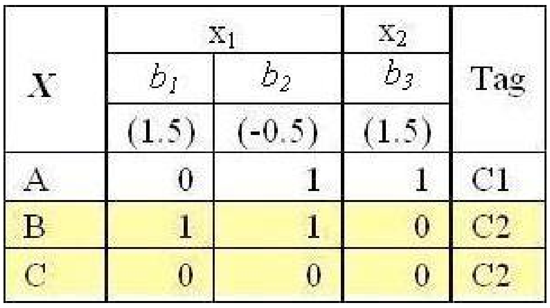
DEFINITION 11. *Let I be the set of indices j for which the j^{th} component of all vectors $X_i = (b_{i1}, b_{i2}, \ldots b_{in'}, m)$*

$\in P_m (\subset {S_{bin}}^{n'+1})$ *have a common value, say* $b_{ij} \in \{0,1\}$. *Then the convex hull of* P_m*, denoted as*

$$[P_m] = \bigwedge_{j \in I} {b_j}^{\alpha_j},$$

where $\alpha_j = b_{ij}$.

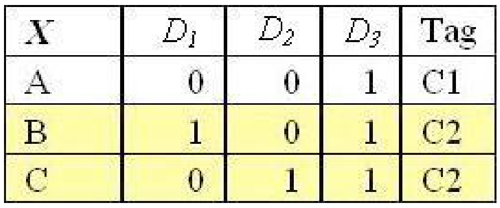
We recall the first data set in its binarized form which we had presented in Section 2. To avoid clutter, we show this data set in Figure 14.



X	x1		x2	Tag
	b_1	b_2	b_3	
	(1.5)	(-0.5)	(1.5)	
A	0	1	1	C1
B	1	1	0	C2
C	0	0	0	C2

Figure 14: Binarized Sample Data Set

Shown in Figure 15 is the data set of indicator dimensions $COMP^{n'+1}$ built from the data set in Figure 14.



X	D_1	D_2	D_3	Tag
A	0	0	1	C1
B	1	0	1	C2
C	0	1	1	C2

Figure 15: $COMP^{n'+1}$

The convex hull of P_{C1} and P_{C2} are $[P_{C1}] = \overline{b_1}b_2b_3$ and $[P_{C2}] = \overline{b_3}$. The same process applies to P_{C1} and P_{C3}, etc.

Observe that this process is actually equivalent to the construction of $z^{h_{P_m}}$ in ACTSF algorithm. Therefore, ACTSF can only construct a theory set (which contains strong spanned patterns) for the data set S^{n+1} if $|(\bigcup_{m \in T} [P_m])| \geq |T|$, where $P_m \subset {S_{bin}}^{n+1}$.

6. CONCLUSION

Our overall performance fairs comparably well from the results of previous literatures [8, 9, 10, 11, 12]. From these literatures, only [12] reported a maximum (best) of 97.3% overall accuracy during its cross validation for the Iris data set. Although [12] has reported of a higher performance percentage, it was seen that only 4 rules were found. These rules all had a degree of 1, thus this performance is not unexpected. Drawbacks on these types of low-degree patterns have been cited in previous sections.

Furthermore, we conclude that the inclusion of the aforementioned vectors would only increase the chances of incorrectly marking untagged observations at the expense of trying to correctly classify most of them in the data space. The use of the discriminant function to significantly improve the performance of ACTSF proved to be a better option compared to this.

By focusing only on dimensions where there exists an observation having a unique behavior in building our indicator dimensions, we narrowed down our search space for theory set formation. In addition to this, we have directly used these indicator dimensions to generate Pareto-optimal patterns for the theory set of the data set S^{n+1}. Overall, we therefore have achieved efficiency in describing the inherent characteristics of each cluster in the data set.

Acknowledgements

The authors would like to thank the people from the Algorithms and Complexity Lab of the Department of Computer Science, UPDiliman for the support they provided.

Ms. Malinao is a scholarship awardee of the Department of Science and Technology - Science Education Institute (DOST-SEI) through the Accelerated Human Resource Development Program.

7. REFERENCES

[1] P. Hammer, T. Bonates. Logical Analysis Of Data: From Combinatorial Optimization to Medical Applications. *RUTCOR Research Report*, RRR 10-2005, February 2005.

[2] E. Boros, P. Hammer, T. Ibaraki, A. Kogan, E. Mayoraz, I. Muchnik. An Implementation of Logical Analysis of Data. *RUTCOR Research Report*, RRR 22-96, July 1996.

[3] E. Boros, P. Hammer, T. Ibaraki, A. Kogan. Logical Analysis of Numerical Data. *RUTCOR Research Report*, RRR 04-97, February 1997.

[4] G. Alexe, S. Alexe, D. Axelrod, T. Bonates, I. Lozina, M. Reiss, P. Hammer. Breast cancer prognosis by combinatorial analysis of gene expression data. Research article, 19 July 2006.

[5] R. Fisher, M. Marshall. UCI Repository of Machine Learning Databases: Machine readable data repository, Department of Computer Science, University of California, Irvine, 1988.

[6] P. Hammer, A. Kogan, B. Simeone, S. Szedmak. Pareto-optimal Patterns in Logical Analysis Of Data. *RUTCOR Research Report*, RRR 7-2001, January 2001.

[7] P. Tan, M. Steinbach, and V. Kumar. Introduction to Data Mining. Addison Wesley 2006.

[8] D. Seo, C. Song, W. Lee. A Classifier Capable of Handling New Attributes. Proceedings of the 2007 IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2007).

[9] J. Li. Rough Set Based Rule Evaluations and Their Applications. PhD Thesis. University of Waterloo, Canada. 2007.

[10] L. Kornienko, D. Albrecht, and D. Dowe. A Preliminary MML Linear Classifier using Principal Components for Multiple Classes. 2005.

[11] W. Duch, K. Grabczewski. Searching for Optimal MLP (Multilayer Perceptron). Nicholas Copernicus University, Poland. 1999.

[12] W. Duch, R. Adamczak, and K. Grabczewski, Extraction of logical rules from backpropagation networks. Neural Processing Letters 7: 1-9, 1998.

Appendix

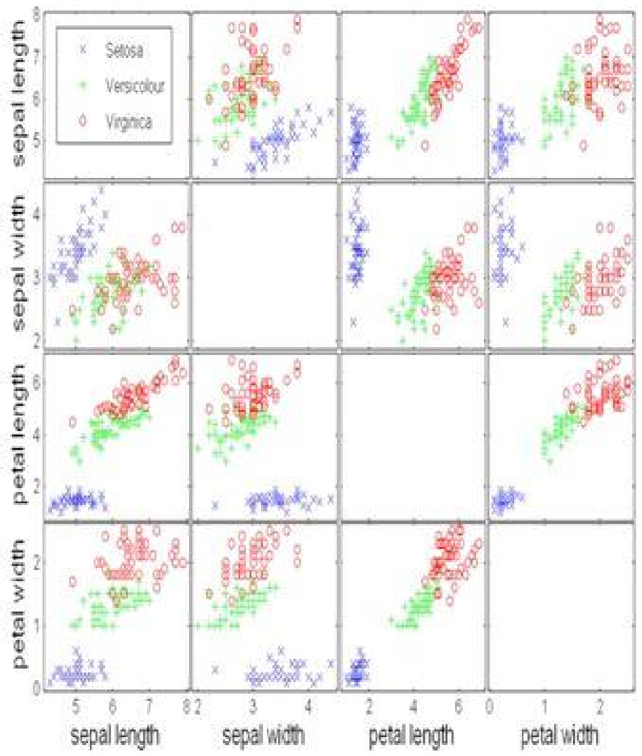


Figure 16: The Iris Data Set Visualization taken from *Introduction to Data Mining* by P. Tan et al.