数理解析研究所講究録
第 1644 巻 2009 年 20-34

Classification by Ordering Data Samples

Kazuya Haraguchi* Seok-Hee Hong† Hiroshi Nagamochi‡

Abstract

Visualization plays an important role as an effective analysis tool for huge and complex data sets in many application domains such as financial market, computer networks, biology and sociology. However, in many cases, data sets are processed by existing analysis techniques (e.g., classification, clustering, PCA) before applying visualization. In this paper, we study visual analysis of classification problem, a significant research issue in machine learning and data mining community. The problem asks to construct a classifier from given set of positive and negative samples that predicts the classes of future samples with high accuracy. We first extract a bipartite graph structure from the sample set, which consists of a set of samples and a set of subsets of attributes. We then propose an algorithm that constructs a two-layered drawing of the bipartite graph, by permuting the nodes using an edge crossing minimization technique. The resulting drawing can act as a new classifier. Surprisingly, experimental results on bench mark data sets show that our new classifier is competitive with a well-known decision tree generator C4.5 in terms of prediction error. Furthermore, the ordering of samples from the resulting drawing enables us to derive new analysis and insight into data such as clustering.

1 Introduction

1.1 Background

We consider a mathematical learning problem called *classification*, which has been a significant research issue from classical statistics to modern research fields on learning theory (e.g., machine learning) and data analysis (e.g., data mining) [9, 18, 3, 8]. Indeed, major existing methodologies to this problem have a geometric, spatial flavor. The main aim of our research is to establish a new learning framework based on information visualization.

In classification, we are given a set $S$ of *samples*. Each sample is specified with values on n *attributes* and belongs to either *positive* $(+)$ or *negative* $(-)$ class. The aim of classification is to construct a function (called a *classifier*) from the sample domain to the class $\{+,-\}$ by using the given sample set $S$, so that the constructed classifier can predict the classes of future samples with high accuracy.

Many existing methodologies are based on geometric concepts and construct a hyperplane as classifier. Classical ones (e.g., *Fisher's linear discriminant* in 1930's and *perceptron* in 1960's [12])

*Department of Information Technology and Electronics, Faculty of Science and Engineering, Ishinomaki Senshu University, Japan (kazuyah@isenshu-u.ac.jp)

†School of Information Technologies, University of Sydney, Australia (shhong@it.usyd.edu.au)

‡Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, Japan (nag@amp.i.kyoto-u.ac.jp)

assume the sample domain to be the n-dimensional real space and decide $n+1$ coefficients $\{\omega_j\}_{j=0}^{n}$ of the hyperplane $f(x)$ in the form:

$$f(x) = \sum_{j=1}^{n} \omega_j x_j + \omega_0,$$

by which a sample $x = (x_1, \dots, x_n)$ is classified into the class determined by $\mathrm{sgn}\{f(x)\}$. Nowadays, *kernel methods* or *support vector machines* enable us to construct hyperplane classifier even from non-spatial data (e.g., protein sequence, webgraph) [16, 13, 17, 15].

Existing methods have made great success in many application areas and one can find various learning algorithms implemented on such softwares as WEKA [19]. When we use them, however, we often face with such difficulties as scaling, choice of distance measure, hardness of interpretation, many of which arise due to geometric concepts. Often it may not be easy to overcome these difficulties. Furthermore it is not easy for human to interact with the resulting classifier or to visualize the hidden structures implicitly learned by the methods. These current situations require us to develop a new framework of classification from another perspective.

1.2 Our Contribution

Visualization plays an important role as an effective analysis tool for huge and complex data sets in many application domains such as financial market, computer networks, biology and sociology. However, in many cases, data sets need to be processed by existing analysis techniques (e.g., classification, clustering, PCA) before they are displayed in a two or three dimensional space.

In this paper, we hypothesize that good visualization (e.g., visual objects with low visual complexity) itself can discover essential or hidden structure of data without relying on data analysis techniques, which can lead to novel learning technique. Based on our hypothesis, we construct a classifier using visualization and show the effectiveness of the classifier by empirical studies. The main contribution of this paper is to open new possibility of such new learning methodologies that can find and visualize essential information on data simultaneously.

We briefly outline how to construct our visual classifier from a set S of samples on a domain $\mathcal{D}_1 \times \cdots \times \mathcal{D}_n$ of n attributes. We first represent the relationship between S and the domain as a two-layered drawing of a bipartite graph, as shown in Figure 1, where each *value node* in the top level represents one of the values in n attributes, and each *sample node* in the bottom level representing one of the samples is joined by edges to the value nodes that specifies the sample.

We then try to reduce the number of edge crossings by changing the drawing (i.e., the ordering of nodes on each side) and by replacing two subdomains into their product as a new subdomain (increasing the number of value nodes) until a termination criteria is satisfied. Figure 2 shows an example of a two-layered drawing of a final bipartite graph, where a large number of positive (negative) samples form a cluster in the bottom level.

We can use the resulting drawing as a classifier as follows. Given a new sample, we determine its position in the bottom level as the average of the positions of the corresponding value nodes, and then judge it as positive (negative) if it falls among positive (negative) sample nodes. Performance of classifier is usually evaluated by error rate on test sample set, and surprisingly, our classifier is competitive with a well-known decision tree classifier C4.5 [11].

This paper is organized as follows. We give the formal definition of bipartite graphs, called "SF-graphs" and describe how to visualize the graphs in Section 2. We explain the crossing

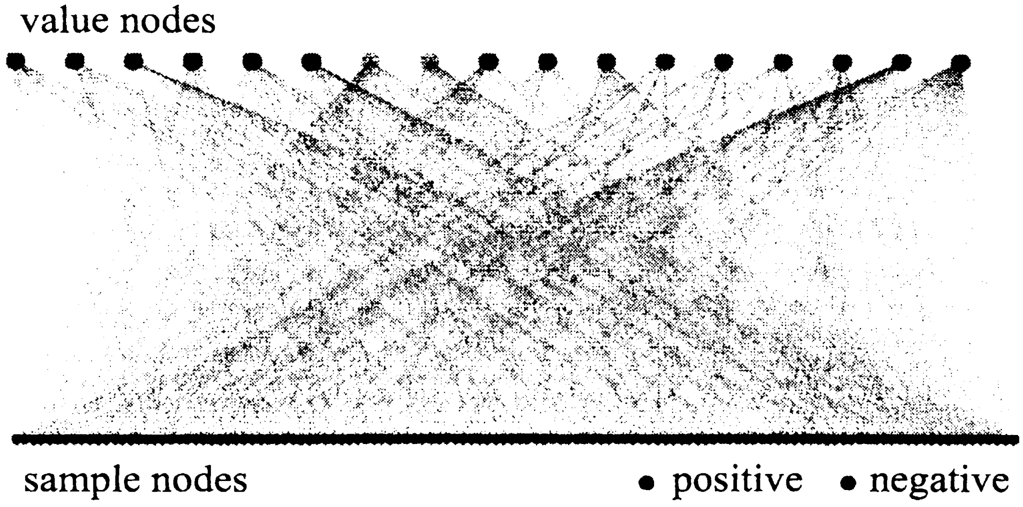


Figure 1: The initial SF-graph from MONKS-1

minimization algorithm for combining subdomains and permuting orderings of nodes in SF-graphs in Section 3. In Section 4, we show how to use visualized SF-graphs as a new classifier. We present some empirical studies in Section 5 and make concluding remarks in Section 6.

2 Preliminary

2.1 Samples and Features over Attributes

In the subsequent discussion, we assume that a set $S = \{s_1, \ldots, s_m\}$ of positive/negative samples over $n$ attributes is given. Let $\mathcal{D}_j$ denote the domain of attribute $j$, i.e., the set of all values taken as a value of attribute $j$, where we assume that each $\mathcal{D}_j$ is a finite unordered set of at least two discrete values. Each sample s_i is specified by an $n$-dimensional vector $s_i = (s_{i,1}, s_{i,2}, \ldots, s_{i,n})$ such that $s_{i,j} \in \mathcal{D}_j$ for each attribute $j \in \{1, 2, \ldots, n\}$. Table 1 shows a set of six samples over 4 attributes.

To handle subdomains of the entire domain, we define "features" and "feature sets." A *feature* is a nonempty subset $F \subseteq \{1, 2, \ldots, n\}$ of the $n$ attributes, and the domain $\mathcal{D}_F$ of a feature $F = \{j_1, \ldots, j_q\}$ is defined as the set of all combinations of values taken by attributes in $F$, i.e., $\mathcal{D}_F = \mathcal{D}_{j_1} \times \cdots \times \mathcal{D}_{j_q}$. The restriction $s|_F$ of a sample $s_i = (s_{i,1}, s_{i,2}, \ldots, s_{i,n})$ to feature $F$ is defined by $(s_{i,j_1}, s_{i,j_2}, \ldots, s_{i,j_q}) \in \mathcal{D}_F$. A value $v \in \mathcal{D}_F$ is called *covered* if there is a sample $s_i \in S$ with $s_i|_F = v$, and a feature $F$ is called *covered* if all values $v \in \mathcal{D}_F$ are covered. A *feature set* $\mathcal{F} = \{F_1, \ldots, F_p\}$ is a set of disjoint features, i.e., $F_k \cap F_{k'} = \emptyset$ for $k \neq k'$. The set of all values in the domains of features $F_1, \ldots, F_p$ is denoted by $\mathcal{D}_{\mathcal{F}} = \mathcal{D}_{F_1} \cup \mathcal{D}_{F_2} \cup \cdots \cup \mathcal{D}_{F_p}$.

2.2 SF-graphs

Given a feature set $\mathcal{F} = \{F_1, \ldots, F_p\}$, we represent the relationship between the sample set $S$ and the set $\mathcal{D}_{\mathcal{F}}$ of values by *sample-feature graph* (*SF-graph*), which is a bipartite graph $G_{\mathcal{F}} = (\mathcal{D}_{\mathcal{F}}, S, E_{\mathcal{F}})$, defined as follows.

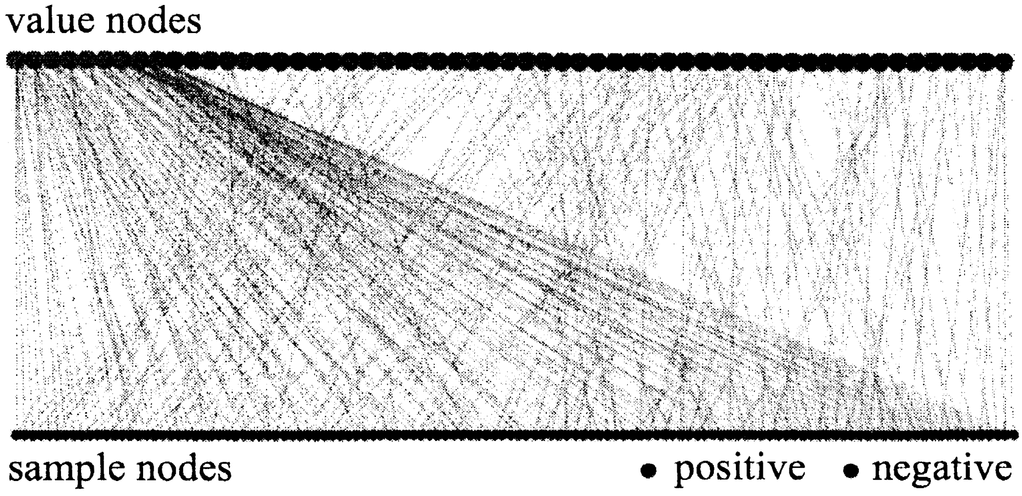


Figure 2: A final SF-graph for MONKS-1

Table 1: A sample set $S = \{s_1, \ldots, s_6\}$ with $\mathcal{D}_1 = \{0, 1\}$, $\mathcal{D}_2 = \{0, 1, 2\}$, $\mathcal{D}_3 = \{\mathrm{T}, \mathrm{F}\}$ and $\mathcal{D}_4 = \{\mathrm{Y}, \mathrm{N}\}$

	Class	Att. 1	Att. 2	Att. 3	Att. 4
s_1	+	0	2	T	Y
s_2	+	1	1	F	N
s_3	+	0	0	T	Y
s_4	−	1	0	F	N
s_5	−	1	1	T	Y
s_6	−	0	2	F	N

- Each value $v \in \mathcal{D}_{\mathcal{F}}$ is represented by a node, called a *value node* in the first node set, and each sample $s_i \in S$ is represented by a node, called a *sample node* in the second node set, where we use the same notation for nodes for simplicity.
- A value node v and a sample node s_i is joined by an edge $(v, s_i) \in \mathcal{D}_{\mathcal{F}} \times S$ if and only if $s_i|_{F_k} = v$ for some feature $F_k \in \mathcal{F}$. Thus the edge set is given by

$$E_{\mathcal{F}} = \{(v, s_i) \in \mathcal{D}_{\mathcal{F}} \times S \mid s_i|_{F_k} = v \text{ for some } F_k \in \mathcal{F}\}.$$

If $\mathcal{F}$ is a singleton $\mathcal{F} = \{F_1\}$, we write $G_{\mathcal{F}}$ by G_{F_1} for convenience.

Figure 3 shows SF-graph $G_{\mathcal{F}} = (\mathcal{D}_{\mathcal{F}}, S, E_{\mathcal{F}})$ of the sample set S in Table 1 for feature set $\mathcal{F} = \{\{1\}, \{2, 3\}, \{4\}\}$. The associated value vector is shown above each value node. Observe that SF-graph shows the membership of samples to the subdomains determined by features in $\mathcal{F}$.

2.3 Visualization by Two-Layered Drawings

We visualize SF-graph $G_{\mathcal{F}}$ with a two-layered drawing, which is defined by a pair of orderings on the two node sets in $G_{\mathcal{F}}$. We denote an *ordering* on a sample set S by a bijection $\sigma : S \rightarrow \{1, \ldots, |S|\}$.

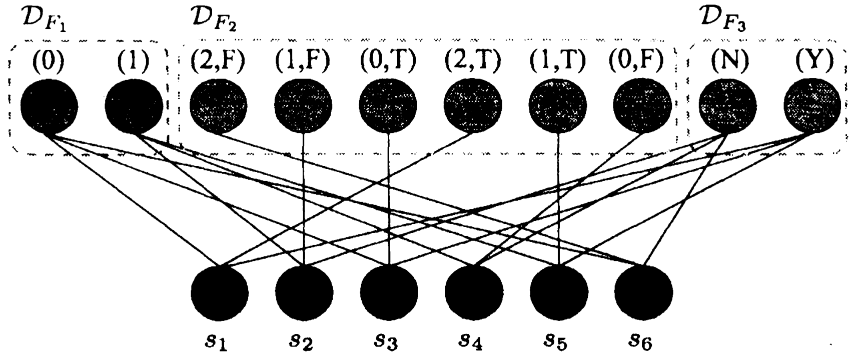


Figure 3: SF-graph $G_{\mathcal{F}} = (\mathcal{D}_{\mathcal{F}}, S, E_{\mathcal{F}})$ of the sample set S in Table 1 constructed for $\mathcal{F} = \{\{1\}, \{2,3\}, \{4\}\}$

For a set $\mathcal{F}$ of features, we denote an ordering on the set $\mathcal{D}_{\mathcal{F}}$ of value nodes by a bijection $\pi_{\mathcal{D}_{\mathcal{F}}} : \mathcal{D}_{\mathcal{F}} \rightarrow \{1, \ldots, |\mathcal{D}_{\mathcal{F}}|\}$. In the rest of the paper, we abbreviate $\pi_{\mathcal{D}_{\mathcal{F}}}$ into $\pi_{\mathcal{F}}$ ($\pi_{\mathcal{D}_{\mathcal{F}}}$ with $\mathcal{F} = \{F_k\}$ into π_k) for convenience if no confusion arises. In the two-layered drawing, the value nodes in $\mathcal{D}_{\mathcal{F}}$ are placed in the top level according to the order $\pi_{\mathcal{F}}$, while the sample nodes in S are placed in the bottom level according to the order σ. Let $\chi(G_{\mathcal{F}}, \pi_{\mathcal{F}}, \sigma)$ denote the number of edge crossings in the drawing $(\pi_{\mathcal{F}}, \sigma)$ of $G_{\mathcal{F}}$.

In order to find a "good" visualization of the given sample set S, we need to find a feature set $\mathcal{F} = \{F_1, \ldots, F_p\}$ and its two-layered drawing $(\pi_{\mathcal{F}}, \sigma)$ such that

$\chi(G_{\mathcal{F}}, \pi_{\mathcal{F}}, \sigma)$ is minimized
subject to
$F_1 \cup \cdots \cup F_p = \{1, 2, \ldots, n\}$,
each feature $F_k \in \mathcal{F}$ is covered.

The motivation of this formulation is as follows. There are many graph drawing studies claiming that edge crossings has the greatest impact on *readability* among various criteria (e.g., bends, symmetry) [10, 6]. In terms of SF-graph, crossing minimization may provide us good information on the true locations of samples in their domain which we cannot usually see.

We claim that the optimal ordering on sample nodes reflects their true locations. This claim is supported by the following observation: Let us take a feature set $\mathcal{F}$ having only one feature. We show how crossing minimization works for such $G_{\mathcal{F}}$ in Figure 4. We observe that samples having the same value (i.e., those belong to the same subdomain) get gathered together in a chain of samples, which we call a *sample chain*, as the result of crossing minimization.

Note that merging two features $F_k, F_{k'} \in \mathcal{F}$ into a new one $F_{k''} = F_k \cup F_{k'}$ gives a two-layered drawing with a smaller number of edge crossings. For an extreme example, the feature set $\mathcal{F} = \{F_1 = \{1, \ldots, n\}\}$ admits a two-layered drawing $(\pi_{\mathcal{F}}, \sigma)$ with no edge crossings, although F_1 is not covered in general (in many data sets, it holds that $S \subsetneq \mathcal{D}_1 \times \cdots \times \mathcal{D}_n$). However, we are not interested in *uncovered* features. Discarding such ones, the number of value nodes from a feature of our interest is bounded by m, while the domain of a feature, direct product of attribute domains, can be exponentially large unless restricted. Since we have no information on uncovered values, continuing to generate uncovered features may easily result in overfitting to the given sample set.

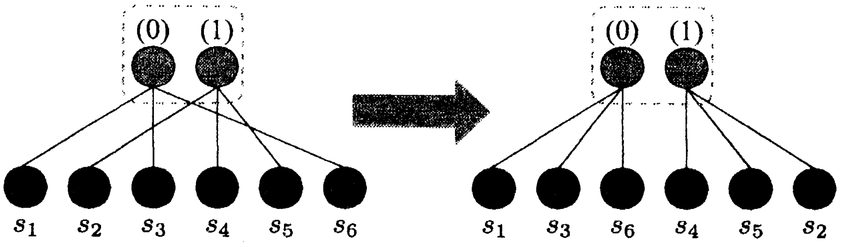

(0)
(1)
s_1 s_2 s_3 s_4 s_5 s_6
(0)
(1)
s_1 s_3 s_6 s_4 s_5 s_2

Figure 4: Crossing minimization on SF-graph with one feature

Hence we can expect that the above good visualization identifies similar samples and collects them as sample chains over a well-structured set of subdomains.

2.4 Crossing Minimization

We now review the studies on crossing minimization in two-layered drawings. *Two-sided crossing minimization problem* (2-CM) asks to decide both $\pi_{\mathcal{F}}$ and σ that minimize $\chi(G_{\mathcal{F}}, \pi_{\mathcal{F}}, \sigma)$. However, 2-CM is NP-hard even if the ordering on one side is fixed [4, 2, 7]. This restricted version of 2-CM is called *one-sided crossing minimization problem* (1-CM). We design a heuristic procedure of crossing minimization on SF-graph in the next section, where 1-CM with a fixed ordering on $\mathcal{D}_{\mathcal{F}}$ forms its bases. Thus, we can formalize 1-CM as follows.

Problem 1-CM$(G_{\mathcal{F}}, \pi_{\mathcal{F}})$

Input: A bipartite graph $G_{\mathcal{F}} = (\mathcal{D}_{\mathcal{F}}, S, E_{\mathcal{F}})$ and an ordering $\pi_{\mathcal{F}}$ on $\mathcal{D}_{\mathcal{F}}$.

Output: An ordering σ on S that minimizes $\chi(G_{\mathcal{F}}, \pi_{\mathcal{F}}, \sigma)$.

3 Algorithm to Grow up SF-graph

In this section, we describe our algorithm PERMUTE-AND-MERGE for finding a "good" visualization of a given sample set S. It consists of two subroutines, one for combining two features into a new one, and the other for permuting value (sample) nodes to reduce edge crossings. Note that replacing two features F_k and $F_{k'}$ with their union $F = F_k \cup F_{k'}$ increases the number of value nodes and decreases the number of edges in SF-graph, since $|\mathcal{D}_F| = |\mathcal{D}_{F_k}| \times |\mathcal{D}_{F_{k'}}|$ is larger than $|\mathcal{D}_{F_k}| + |\mathcal{D}_{F_{k'}}|$ and two edges (s_i, a) and (s_i, b) with $a \in \mathcal{D}_{F_k}$ and $b \in \mathcal{D}_{F_{k'}}$ are replaced with a single edge $(s, (a, b))$, $(a, b) \in \mathcal{D}_F$. Thus the algorithm grows up the value node side of SF-graph by combining features until no two features F_k and $F_{k'}$ generate a new covered feature $F = F_k \cup F_{k'}$.

Algorithm PERMUTE-AND-MERGE is described in Algorithm 1. The algorithm starts from the set $\mathcal{F}$ of n singleton features (line 2) and arbitrary orderings on value and sample nodes (line 3). Then it iterates computing node orderings by crossing minimization (using procedure PERMUTE in line 5) and merging an appropriate pair of features into one (using function MERGE in line 6); if no pair results in a covered feature, then the algorithm terminates. The next two sections describe the two subroutines, respectively.

Algorithm 1 PERMUTE-AND-MERGE

```
 1: procedure
 2:     $\mathcal{F} \leftarrow \{\{j\} \mid j \in \{1,\ldots,n\}\}$          ▷ $G_{\mathcal{F}}$ is constructed
 3:     $\pi_{\mathcal{F}}, \sigma \leftarrow$ arbitrary orderings on $\mathcal{D}_{\mathcal{F}}, S$
 4:     repeat
 5:         PERMUTE
 6:     until MERGE outputs no
 7:     output $G_{\mathcal{F}}$ and $(\pi_{\mathcal{F}}, \sigma)$
 8: end procedure
```

Algorithm 2 PERMUTE-BY-LS

```
 9: procedure
10:     repeat
11:         $\pi_{\mathcal{F}}^{\text{init}} \leftarrow \pi_{\mathcal{F}}$, $\sigma^{\text{init}} \leftarrow \sigma$
12:         for all $\pi'_{\mathcal{F}} \in N(\pi_{\mathcal{F}})$ do
13:             $\sigma' \leftarrow$1-CM$(G_{\mathcal{F}}, \pi'_{\mathcal{F}})$
14:             if $\chi(G_{\mathcal{F}}, \pi'_{\mathcal{F}}, \sigma') < \chi(G_{\mathcal{F}}, \pi_{\mathcal{F}}, \sigma)$ then
15:                 $\pi_{\mathcal{F}} \leftarrow \pi'_{\mathcal{F}}$, $\sigma \leftarrow \sigma'$
16:                 break the for-loop
17:             end if
18:         end for
19:     until $\pi_{\mathcal{F}} \equiv \pi_{\mathcal{F}}^{\text{init}}$ and $\sigma \equiv \sigma^{\text{init}}$
20: end procedure
```

3.1 Procedure PERMUTE

The subroutine PERMUTE can be any algorithm for crossing minimization. Here we propose PERMUTE-BY-LS in Algorithm 2, a heuristic method based on *local search*. Starting from an ordering $\pi_{\mathcal{F}}$, the local search seeks for a better ordering $\pi'_{\mathcal{F}}$ in its *neighborhood*. The neighborhood of $\pi_{\mathcal{F}}$, denoted by $N(\pi_{\mathcal{F}})$, is a set of orderings which can be obtained by a *slight* change of $\pi_{\mathcal{F}}$. A neighbor ordering $\pi'_{\mathcal{F}} \in N(\pi_{\mathcal{F}})$ is evaluated by edge crossings $\chi(G_{\mathcal{F}}, \pi'_{\mathcal{F}}, \sigma')$ where σ' is obtained by solving 1-CM$(G_{\mathcal{F}}, \pi'_{\mathcal{F}})$ (lines 13 and 14). If the edge crossings is smaller than the previous one, then the procedure adopts $\pi'_{\mathcal{F}}$ and σ' (line 15) and continues to search the neighborhood of the new $\pi_{\mathcal{F}}$. Otherwise, the procedure terminates.

The followings are detailed settings of PERMUTE-BY-LS.

Restriction on Value Node Ordering. In order to reduce the search space, we restrict ourselves to such $\pi_{\mathcal{F}}$ where value nodes from the same feature are ordered consecutively, as in Figure 3. This restriction enables us to represent $\pi_{\mathcal{F}}$ as a concatenation of p orderings $\pi_1, \ldots, \pi_p$ on $\mathcal{D}_{F_1}, \ldots, \mathcal{D}_{F_p}$, where the concatenation is given by an ordering $\pi^* : \mathcal{F} = \{F_1, \ldots, F_p\} \to \{1, \ldots, p\}$. Thus $\pi_{\mathcal{F}}(v)$ for $v \in \mathcal{D}_{F_k}$ is given by:

$$\pi_{\mathcal{F}}(v) = \sum_{k':\ \pi^*(F_{k'}) < \pi^*(F_k)} |\mathcal{D}_{F_{k'}}| + \pi_k(v).$$

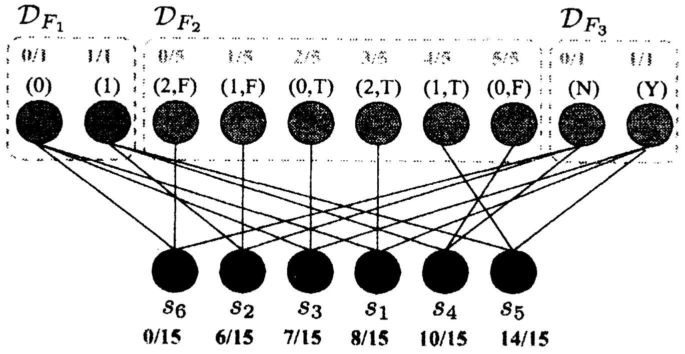


Figure 5: SF-graph ordered by the barycenter heuristic

It is easy to show that the total number of edge crossings is given by:

$$\begin{aligned}\chi(G_{\mathcal{F}}, \pi_{\mathcal{F}}, \sigma) = &\sum_{k=1}^{p} \chi(G_{F_k}, \pi_k, \sigma) \\ &+ p(p-1)|S|(|S|-1)/2, \end{aligned} \qquad (1)$$

where the second term of the right hand is a constant for a fixed p. The above equality (1) means that the number of edge crossings is invariant with a choice of an ordering π^* on $\mathcal{F}$. In what follows, we concentrate on searching $\pi_1, \ldots, \pi_p$ during the procedure, assuming that π^* is chosen arbitrarily.

Neighborhood. For neighborhood $N(\pi_{\mathcal{F}})$, we take the set of all orderings on $\mathcal{D}_{\mathcal{F}}$ which can be obtained by swapping $\pi_k(v)$ and $\pi_k(v')$ for any $v, v' \in \mathcal{D}_{F_k}$ $(k = 1, \ldots, p)$. In line 12, a neighbor $\pi'_{\mathcal{F}} \in N(\pi_{\mathcal{F}})$ is chosen at random.

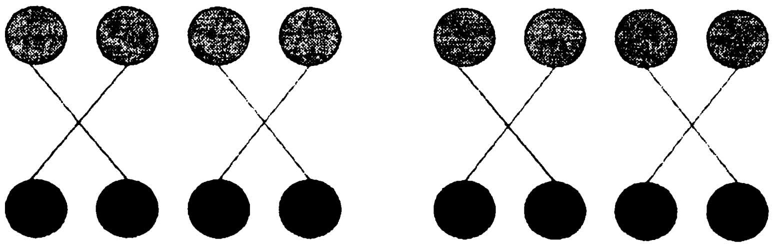
Solving 1-CM. We use the *barycenter heuristic* in solving 1-CM$(G_{\mathcal{F}}, \pi'_{\mathcal{F}})$ to decide σ' in line 13 [14, 7]. The barycenter heuristic permutes the set S of sample nodes in the increasing order of *barycenter* values: Let us take any subset $\mathcal{D}_{F_k} \subseteq \mathcal{D}_{\mathcal{F}}$ $(k = 1, \ldots, p)$. For a value node $v \in \mathcal{D}_{F_k}$ and an ordering π'_k on $\mathcal{D}_{F_k}$, we define the *normalized index* $I(v, \pi'_k)$ as follows:

$$I(v, \pi'_k) = \frac{\pi'_k(v) - 1}{|\mathcal{D}_{F_k}| - 1}. \qquad (2)$$

The barycenter $\beta(s, \pi'_{\mathcal{F}})$ of a sample node $s \in S$ (w.r.t. an ordering $\pi'_{\mathcal{F}}$ on $\mathcal{D}_{\mathcal{F}}$ including $\pi'_1, \ldots, \pi'_p$) is defined by

$$\beta(s, \pi'_{\mathcal{F}}) = \frac{1}{p} \sum_{k=1}^{p} I(s|_{F_k}, \pi'_k). \qquad (3)$$

Figure 5 shows the result of the barycenter heuristic on the SF-graph of Figure 3, along with normalized indices of value nodes and barycenter values of sample nodes. We note that $I(v, \pi'_k) = \pi'_k(v)$ is usually used in the literature to number value nodes. However, by (2), all features have equal influence in deciding barycenter since $I(v, \pi'_k) \in [0, 1]$, which improved the learning performance in our preliminary experiments.



$\chi = 2$, $\chi_{\rm col} = 2$ $\chi = 2$, $\chi_{\rm col} = 0$

Figure 6: Illustration of χ and $\chi_{\rm col}$

Evaluation of Orderings. In line 14, it is natural to use the number of edge crossings χ in evaluation of orderings $(\pi_{\mathcal{F}}, \sigma)$, but we can introduce various evaluating functions there. For example, assume coloring each edge $(v, s_i) \in E$ according to the class of sample $s_i \in S$. Then we can define *colored edge crossings* $\chi_{\rm col}(G_{\mathcal{F}}, \pi_{\mathcal{F}}, \sigma)$ to be the number of crossings between different colored edges. In the two drawings in Figure 6, $\chi_{\rm col}$ prefers the right one while χ rates both situations similarly. By using $\chi_{\rm col}$ as an evaluator, we can expect samples belonging to the same class to form clusters.

Analogously with χ in (1), $\chi_{\rm col}(G_{\mathcal{F}}, \pi_{\mathcal{F}}, \sigma)$ can be represented as the summation of $\chi_{\rm col}(G_{F_k}, \pi_k, \sigma)$'s and a constant. Taking the quality of feature into account, we define *weighted colored edge crossings* $\chi^*_{\rm col}(G_{\mathcal{F}}, \pi_{\mathcal{F}}, \sigma)$ by

$$\chi^*_{\rm col}(G_{\mathcal{F}}, \pi_{\mathcal{F}}, \sigma) = \sum_{k=1}^{p} \frac{\chi_{\rm col}(G_{F_k}, \pi_k, \sigma)}{\rho(F_k)} \tag{4}$$

for an *impurity function* $\rho(F_k)$ on $\mathcal{D}_{F_k}$ which indicates the quality of feature F_k. We use the following for our experiments:

$$\rho(F_k) = \sum_{v \in \mathcal{D}_{F_k}} \deg^+(v) \deg^-(v),$$

where $\deg^+(v)$ and $\deg^-(v)$ denote the numbers of positive and negative sample nodes $s_i \in S$ with $s_i|_{F_k} = v$, respectively. The $\rho(F_k)$ becomes small (resp., large) if F_k divides the sample set S into pure (resp., impure) subsets in the sense of class distribution and thus may (resp., may not) have nice information on classification. Then $\chi^*_{\rm col}$ in (4) must be much influenced by $\chi_{\rm col}$ of good features.

3.2 Function Merge

The subroutine Merge in Algorithm 1 can be any function that outputs either *yes* or *no* according to whether there exists an appropriate pair of features to be merged. It should also merge such a pair into a new feature if *yes*.

We present our algorithm Merge-by-Priority in Algorithm 3. The algorithm first constructs a priority queue Q for all pairs of features in $\mathcal{F}$ (line 22). The prioritization of feature pairs determines the ordering according to which they are tested for merge, and we will mention the details below.

Algorithm 3 MERGE-BY-PRIORITY

```
21: function
22:     $Q \leftarrow$ a priority queue for all pairs of features in $\mathcal{F}$
23:     while $Q$ is not empty do
24:         $\{F_k, F_{k'}\} \leftarrow$ dequeue($Q$)
25:         $F \leftarrow F_k \cup F_{k'}$
26:         if $F$ is covered then
27:             $\mathcal{F} \leftarrow (\mathcal{F} \setminus \{F_k, F_{k'}\}) \cup \{F\}$
28:             reconstruct $G_{\mathcal{F}}$ for the new $\mathcal{F}$
29:             initialize $\pi_{\mathcal{F}}$
30:             $\sigma \leftarrow$1-CM($G_{\mathcal{F}}, \pi_{\mathcal{F}}$)
31:             return yes
32:         end if
33:     end while
34:     return no
35: end function
```

The algorithm removes the feature pair $\{F_k, F_{k'}\}$ from the head of Q (operation **dequeue** in line 24) and tests whether it can be merged under our criteria. If Q becomes empty as the result of repetition of removal, then the algorithm returns *no*, indicating that there is no appropriate feature pair to be merged.

We test in line 26 whether the new F merged from F_k and $F_{k'}$ is covered, i.e., all generated value nodes in F are connected to sample nodes in the reconstructed SF-graph. For example, starting from $\mathcal{F} = \{\{1\}, \{2\}, \{3\}, \{4\}\}$, the pair of features $\{2\}$ and $\{3\}$ passes the test since all resulting value nodes are connected, as shown in Figures 3 or 5.

If the feature pair $\{F_k, F_{k'}\}$ passes the test, then the algorithm merges F_k and $F_{k'}$ into a new feature $F = F_k \cup F_{k'}$, reconstructs SF-graph, initializes $\pi_{\mathcal{F}}$ and σ and returns *yes* (lines 27 to 31). Now let us describe the details of the algorithm.

Prioritization of Feature Pairs. Merge of feature pair has great influence on the quality of the final drawing. Hence we should be careful in choosing the priority measure which decides the ordering for test. In our experiments, we give higher priority to such a pair $\{F_k, F_{k'}\}$ that has a smaller value in the following summation:

$$\frac{\chi_{\mathrm{col}}(G_{F_k}, \pi_k, \sigma)}{\rho(F_k)} + \frac{\chi_{\mathrm{col}}(G_{F_{k'}}, \pi_{k'}, \sigma)}{\rho(F_{k'})}, \qquad (5)$$

which comes from the definition of weighted colored edge crossings (4). We expect the above evaluator to refine features following the sample ordering well (which have small χ_{col}) or to remove impure features (which have large ρ).

Initialization of Value Node Ordering. In line 29, for the new $\mathcal{D}_{\mathcal{F}}$, we need to decide $\pi_{\mathcal{F}}$ so that it serves as a good initial solution in the next local search. Let us recall that $\mathcal{F}$ has $p-1$ features and that $\pi_{\mathcal{F}}$ is determined by orderings on $p-1$ value node subsets.

We now describe how to choose π_F on the value node set $\mathcal{D}_F$ arising from the new feature $F = F_k \cup F_{k'}$. As we have π_k on $\mathcal{D}_{F_k}$ and $\pi_{k'}$ on $\mathcal{D}_{F_{k'}}$ which have been parts of the local optimum

Table 2: Summary of data sets from UCI Repository

Data	$\|S\|$	n
MONKS-1	124	6
MONKS-2	169	6
MONKS-3	122	6
BCW	699	9 (22.6)
GLASS	214	9 (15.9)
HABER	306	3 (34.4)
HEART	270	13 (23.7)
HEPA	155	19 (19.5)
IONO	200	34 (42.5)

found in the last local search, we decide π_F by using lexicographic ordering based on them: we define $\pi_F(a) < \pi_F(b)$ for two values $a, b \in \mathcal{D}_F$ if and only if the following holds:

$$\begin{aligned}&(\pi_k(a|_{F_k}) < \pi_k(b|_{F_k})) \text{ or}\\&(\pi_k(a|_{F_k}) = \pi_k(b|_{F_k}) \text{ and } \pi_{k'}(a|_{F_{k'}}) < \pi_{k'}(b|_{F_{k'}})).\end{aligned}$$

Finally, we keep the orderings for the rest $p-2$ value node subsets which have not been changed by merge.

4 Classifying with Visualized SF-graphs

In this section, we describe how to use a two-layered drawing $(\pi_{\mathcal{F}}, \sigma)$ of a final SF-graph $G_{\mathcal{F}}$ as a new visual classifier. Let us denote a test sample by $t \in \mathcal{D}_1 \times \cdots \times \mathcal{D}_n$. We classify $t$ according to its nearest neighbor in barycenter: we examine the value nodes to which sample node $t$ should be connected, i.e., the value node $v \in \mathcal{D}_{F_k}$ with $t|_{F_k} = v$ for each feature F_k $(k = 1, \ldots, p)$, and compute its barycenter $\beta(t, \pi_{\mathcal{F}})$ by (3). Finally, we classify $t$ into the class of the nearest sample in terms of barycenter.

For example, let us take the SF-graph in Figure 5 again. In this case, a test sample $t = (1, 2, \mathrm{F}, \mathrm{Y})$ has barycenter 10/15 and is classified into negative class since negative sample s_4 is the nearest.

5 Empirical Studies

In this section, we present empirical studies on data sets from UCI Repository of Machine Learning [1]. First we give summary of the data sets, and then describe experimental results.

5.1 Data Sets

Table 2 shows the summary of the used data sets, where we will explain the decimals in the rightmost column later. MONKS-1 to 3 are *artificial* data sets in the sense that the oracle function from the sample domain to the class set is available. In these data sets, each attribute $j$ has 2 to

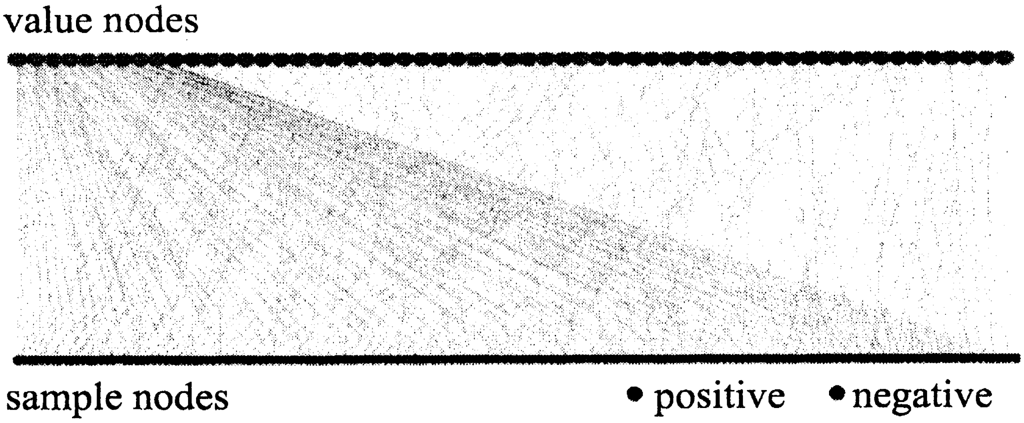


Figure 7: SF-graph obtained from MONKS-2

4 categorical values as its domain $\mathcal{D}_j$. For example, a sample $s_i = (s_{i,1}, \ldots, s_{i,6})$ in MONKS-1 is positive if and only if $(s_{i,1} = s_{i,2})$ or $(s_{i,5} = \text{“1”})$.

The others are *real* data sets, where the oracle is not available. For example, BCW (abbreviation of *breast-cancer-wisconsin*) consists of 699 samples (patients), where 241 samples are positive (malignant) and the rest 458 samples are negative (benign). Each sample has integral values for 9 attributes (e.g., clump thickness, uniformity of cell size).

We evaluate a classifier by its prediction error rate on future samples. In the literature, prediction error rate is usually estimated by constructing a classifier from a *training set* of samples and then evaluating its error rate on a *test set*. For artificial data sets, we take the entire data set as the training set and consider all possible samples as the test set by generating them. The test set contains 432 samples in all for each MONKS data set.

For real data sets, we estimate prediction error rate by the average of error rates observed in 5 trials of 10-fold cross validation, a well-known methodology to divide the data set into training and test sets. In these data sets, some attributes take continuous numerical values which cannot be treated in our formulation. Thus we generate binarization rules (e.g., a rule may map a continuous value to 1 if it is larger than a computed threshold, and to 0 otherwise) from a training set by the algorithm proposed in [5], and then construct a classifier from the binarized data set. A decimal in Table 2 indicates average on the number of generated binary attributes.

5.2 SF-graphs

Figures 2 and 7 show the output SF-graphs for MONKS-1 and 2 respectively. The sample chain extracted from the output SF-graph for BCW is shown in Figure 8. In all figures, we see that samples of the same class form clusters. We show how SF-graph provides information in interactive data analysis.

MONKS-1. Among samples remote from boundaries of clusters, we find those belonging to the class to a strong degree: a sample in MONKS-1 is positive if and only if it satisfies *either* condition stated in the last subsection. Let us regard those satisfying *both* conditions as *strongly* positive samples. The training set contains 8 strongly positive samples, and we observed that they are

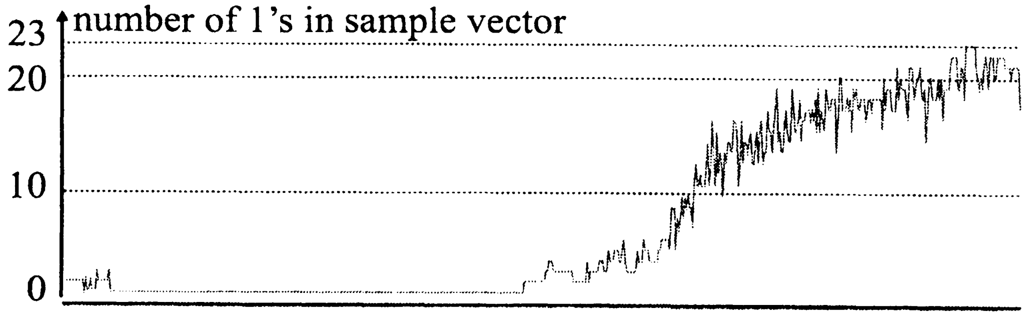


Figure 8: Sample chain from BCW and the number of 1's in sample vector

Table 3: Prediction error rates (%)

Data	SFC	C4.5
MONKS-1	16.20	0.00
MONKS-2	25.60	29.60
MONKS-3	19.36	0.00
BCW	5.12 ± 0.53	5.16 ± 0.64
GLASS	9.86 ± 0.94	7.74 ± 1.63
HABER	38.30 ± 1.69	25.50 ± 1.16
HEART	26.07 ± 1.11	21.31 ± 1.60
HEPA	23.54 ± 2.03	20.89 ± 1.52
IONO	20.20 ± 1.50	14.40 ± 1.08

actually gathered in the left end of the sample chain.

MONKS-2. We see that similar samples get closer in the sample chain: roughly speaking, we see two large clusters of positive samples in Figure 7. A sample s_i in MONKS-2 is positive if and only if s_i has nominal "1" for exactly two attributes. We observed that the right cluster contains all positive samples having nominal "1" for the 1-st attribute, and that the left cluster contains more than 70% of positive samples (20 out of 28) having nominal "1" for the 5-th attribute.

BCW. Figure 8 shows the number of 1's in binarized sample vector by green line besides the obtained sample chain. Based on our research experience, we have observed that samples are more likely to be positive with many 1's although we do not know the oracle function of BCW exactly. We can view the phenomena in the figure, where degree of class membership is also reflected in the sample chain.

5.3 Classification

We construct an SF-graph based classifier (SFC for short) as described in Section 4. We present prediction error rates in Table 3, where we compare our SFC with C4.5 [11], a well-known decision tree generator. Standard deviation is shown for real data sets, which was derived from 5 trials of 10-fold cross validation. C4.5 constructs decision trees within 1 second in all cases while SFC takes from 1 (for GLASS) to 120 seconds (for IONO), which is not too expensive. We observed that computation time of SFC is proportional to the number of attributes which decides the size of neighborhood in the local search. All experiments are conducted on our PC carrying 2.83GHz CPU and 4GB main memory.

A typical decision tree is regarded as disjunction of if-then rules, each of which is represented by conjunction of conditions on *one* attribute value. We can realize small decision trees easily that represent the oracle functions of MONKS-1 (see Section 5.1) and 3, and thus it is reasonable to say that C4.5 is successful for the two data sets. On the other hand, it must be not be so easy to represent the oracle of MONKS-2 (see Section 5.2) by small tree, and for such a data set, SFC is superior to C4.5.

Let us discuss our results for real data sets. Our SFC is worse than C4.5 for almost all the data sets but is competitive with BCW. Thus SFC can attain sufficient classifiers for suitable data sets. We need to further examine what data type is suitable for SFC, but for such data sets, SFC surely serves as an excellent classifier.

6 Concluding Remarks

Our results are significant particularly in the following sense: for machine learning research, our classifier is quite different from existing approaches because it is constructed based on combinatorial formulation (bipartite graph and crossing minimization) rather than geometric concepts. We assert that decision tree is also one of geometric classifiers since a typical algorithm constructs a decision tree by partitioning the sample domain with hyperplanes, even though it is represented by tree on the surface. Thus our work may give a new direction to studies of learning problems. For information visualization, on the other hand, we could show possibility of such a new scheme that enables human to gain knowledge using visualization.

It seems possible for us to improve the performance of our SFC classifier further. For example, we may divide a value node by the current definition into a pair of positive and negative value nodes, where positive (resp., negative) sample nodes are connected to the former (resp., latter) ones. This setting enlarges the search space, but we think that it should be effective for our purpose. Our future work also includes finding good application areas of our methodology to gain some more insights from the feedback.

This work is supported by Grant-in-Aid for Young Scientists (Start-up, 20800045) from Japan Society for the Promotion of Science (JSPS).

References

[1] A. Asuncion and D.J. Newman. *UCI Machine Learning Repository.* University of California, Irvine, School of Information and Computer Sciences, 2007. `http://www.ics.uci.edu/ mlearn/MLRepository.html`.

[2] P. Eades and N. C. Wormald. Edge crossings in drawings of bipartite graphs. *Algorithmica*, Vol. 11, pp. 379–403, 1994.

[3] J. H. Friedman. Recent advances in predictive (machine) learning. *Journal of Classification*, Vol. 23, pp. 175–197, 2006.

[4] M. R. Garey and D. S. Johnson. Crossing number is NP-complete. *SIAM Journal on Algebraic and Discrete Methods*, Vol. 4, pp. 312–316, 1983.

[5] K. Haraguchi and H. Nagamochi. Extension of ICF classifiers to real world data sets. In *IEA/AIE*, Vol. 4570 of *Lecture Notes in Computer Science*, pp. 776–785, Kyoto, Japan, 2007. Springer.

[6] W. Huang, S. Hong, and P. Eades. Layout effects on sociogram perception. In *Proceedings of Graph Drawing 2005 (GD2005)*, pp. 262–273, Limerick, Ireland, 2006.

[7] M. Jünger and P. Mutzel. 2-layer straightline crossing minimization: Performance of exact and heuristic algorithms. *Journal of Graph Algorithms and Applications*, Vol. 1, No. 1, pp. 1–25, 1997.

[8] Jon R. Kettenring. The practice of cluster analysis. *Journal of Classification*, Vol. 23, pp. 3–30, 2006.

[9] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell. *Machine Learning: An Artificial Intelligence Approach.* Morgan Kaufmann, 1983.

[10] H. Purchase. Which aesthetic has the greatest effect on human understanding? In *Proceedings of the 5th International Conference on Graph Drawing (GD'97)*, pp. 248–261, Rome, Italy, 1997.

[11] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.

[12] B. D. Ripley. *Pattern Recognition and Neural Networks.* Cambridge University Press, 1996.

[13] B. Schölkopf and A. J. Smola. *Learning with Kernels.* MIT Press, 2002.

[14] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-11, No. 2, pp. 109–125, 1981.

[15] T. Tamura and T. Akutsu. Subcellular location prediction of proteins using support vector machines with alignment of block sequences utilizing amino acid composition. *BMC Bioinformatics*, Vol. 8, No. 466, November 2007.

[16] V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer, 2nd edition, 1999.

[17] T. Washio and H. Motoda. State of the art of graph-based data mining. *ACM SIGKDD Explorations Newsletter*, Vol. 5, No. 1, pp. 59–68, July 2003.

[18] S. M. Weiss and C. A. Kulikowski. *Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems.* Morgan Kaufmann, 1991.

[19] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, San Francisco, 2nd edition, 2005. `http://www.cs.waikato.ac.nz/ml/weka/`.