Machine Learning, 54, 99–124, 2004

DeEPs: A New Instance-Based Lazy Discovery and Classification System*

JINYAN LI jinyan@i2r.a-star.edu.sg
Institute for Infocomm Research, 21, Heng Mui Keng Terrace, Singapore 119613

GUOZHU DONG gdong@cs.wright.edu
Department of CSE, Wright State University, USA

KOTAGIRI RAMAMOHANARAO rao@cs.mu.oz.au
Department of CSSE, The University of Melbourne, AU

LIMSOON WONG limsoon@i2r.a-star.edu.sg
Institute for Infocomm Research, 21, Heng Mui Keng Terrace, Singapore 119613

Editor: Raul Valdes-Perez

Abstract. Distance is widely used in most lazy classification systems. Rather than using distance, we make use of the frequency of an instance's subsets of features and the frequency-change rate of the subsets among training classes to perform both knowledge discovery and classification. We name the system DeEPs. Whenever an instance is considered, DeEPs can efficiently discover those patterns contained in the instance which sharply differentiate the training classes from one to another. DeEPs can also predict a class label for the instance by compactly summarizing the frequencies of the discovered patterns based on a view to collectively maximize the discriminating power of the patterns. Many experimental results are used to evaluate the system, showing that the patterns are comprehensible and that DeEPs is accurate and scalable.

Keywords: instance-based, emerging patterns, borders, lazy learning, classification

1. Introduction

Lazy learning (Aha, 1997), as exemplified by k-nearest neighbor (k-NN) (Cover & Hart, 1967), is an extensively and thoroughly studied topic in the machine learning field (Aha, Kibler, & Albert, 1991; Dasarathy, 1991; Salzberg, 1991; Zhang, 1992; Langley & Iba 1993; Wettscherech, 1994; Datta & Kibler 1995, 1997; Wettschereck & Dietterich, 1995; Devroye, Gyorfi, & Lugosi, 1996; Domingos, 1996; Wilson & Martinez, 1997, 2002; Keung & Lam, 2000; Kubat & Cooperson, 2000). The intuition behind the k-NN classifier is that the class of a test instance is *most likely* to be the prevailing class among the k nearest

*Parts of the results presented in this paper appear in the Proceedings of the Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases, Lyon, France, 2000, and the Proceedings of the Fifth Pacific-Asia Conference On Knowledge Discovery and Data Mining, Hong Kong, 2001.

training instances (examples) of the test instance according to some distance measure. The most common characteristic of the k-NN classifiers is that they always locate "raw" training instances or their prototypes in the whole training set without any extraction of high level patterns or rules. Such a classification that is based solely on a distance measure is insufficient for sophisticated applications, like cell type classification and patients diagnosis (Li & Wong, 2002), where comprehensible knowledge patterns and rules are needed for explaining prediction outcome.

This paper considers lazy learning from a new direction. Instead of focusing on distance, we are interested in regular patterns contained in an instance which *frequently* occur in one class of training data but *less* frequently, perhaps even not occurring at all, in the other classes of training data. Note that we define here an instance as a set of attribute-value pairs. Those patterns satisfying the above frequency-change requirement are generally called *Emerging Patterns* (EPs) (Dong & Li, 1999; Li, 2001). We name our system DeEPs.[1] This new lazy learning and classification approach goes beyond distance to take into account pattern frequency and makes use of the change ratio of frequency.

The number of subsets of a test instance is exponential in terms of the number of attributes describing the instance. Suppose a training data set consists of two classes of instances, the naive enumeration of all subsets of a test instance, and the calculation and comparison of their frequency in the two classes is very expensive.

We attack this problem with an effective data reduction method. Whenever a new instance is being considered, the method uses that instance as a filter to remove irrelevant training values, making the original training data table sparse in terms of both dimension (number of attributes) and volume (number of instances). The reduced training instances are further compressed along the volume direction by selecting only those maximal ones. After this remarkable reduction, the discovery of our interesting patterns becomes much less expensive than operating on the whole original training data as valid candidate subsets of the considered instance have already been identified in the reduction process. This is a fundamental idea of our DeEPs system.

According to our previous experiments, using an *eager* learning approach to discover all EPs from training data is time consuming. Sometimes, it is impossible to complete the learning phase. As a significant data reduction can be achieved, we choose to use the proposed lazy learning approach.

Specifically, three main issues are investigated in this work:

1. *Efficient instance-based discovery*. Given an instance (either training or test), we discover the subsets of the instance which *most frequently* occur in one class of the training data, but do not occur in the other classes, namely with a frequency-change ratio of infinity. Being *most frequently* means the patterns are minimal (or most general) in the set-containment sense among all patterns having the infinity rate. As will be seen, such patterns are boundary elements of some pattern spaces, and they can be efficiently discovered by our border-based algorithms. By focusing on the most frequent EPs, the EPs which are proper supersets of the boundary EPs (and have lower frequencies) can be ignored. Then the complexity of our system is much simplified.
2. *Ranking the discovered patterns*. In some situations, the number of the discovered boundary patterns is large. To help assess the importance of the individual patterns, we rank

them according to some interestingness measurements such as the pattern's frequency, length, and frequency-change ratio.
3. *Aggregating the frequencies of the patterns to predict the class label for test instances.* The usefulness of the patterns is a primary motivation of this work. To show the usefulness, we apply our patterns to many classification problems. Basically, the classification is performed by comparing aggregated frequencies of the discovered patterns in different classes.

Good solutions to the problems in the first issue can quickly provide comparative and discriminating knowledge for us to understand a new instance. For example, to diagnose a new heart disease patient, medical doctors would like to know what physical and clinical features, of this patient, or their combinations could best match other patients and best differentiate this patient from healthy people. The patterns that the doctors would need were exactly those we discussed in the first issue.

Top-ranked patterns in different orderings of the patterns, as addressed in the second issue, can help us thoroughly understand an instance from different angles. Reliable and accurate predictions by a classification system is an important factor in evaluating the system. According to our experimental results on 40 data sets (Blake & Murphy, 1998), DeEPs is accurate. Its performance is comparable to and often better than classifiers such as k-NN and C5.0 (Quinlan, 1993). DeEPs can handle both continuous and discrete attributes and is scalable over the number of training instances. Due to the lazy-learning scheme, DeEPs is extremely useful for practical applications where the training data must be frequently updated.

The remainder of this paper is organized as follows: Section 2 outlines the basic ideas and essential features of the DeEPs approach. Section 3 presents detailed steps for instance-based pattern discovery, including an explanation on how the discovered EPs are concisely represented by borders, and why instance-based discovery is highly efficient. Section 4 describes how DeEPs can predict a class label of a test instance by using summarized frequencies of the discovered patterns. Sections 5 describes an example to illustrate the entire process of DeEPs. Section 6 describes the algorithms JEPPRODUCER and BORDER-DIFF which are used by DeEPs. Section 7 presents methods to rank the discovered patterns, as human users might want to examine ordered patterns according to some bias. Section 8 provides thorough experimental results on the accuracy, speed, and scalability of DeEPs. Section 9 combines the strength of pattern frequency and distance for instance-based classification. Section 10 discusses some interesting topics related to the current work. Section 11 concludes this paper with a brief summary.

2. Overview of DeEPs

Firstly, we again stress that this paper defines an instance as a set of attribute-value pairs with a cardinality equal to the number of attributes describing the underlying relational data. Next, we specify a definition of emerging patterns.[2]

Definition 2.1. Given two data sets $\mathcal{D}_1$ and $\mathcal{D}_2$, an emerging pattern is an *itemset* (a set of attribute-value pairs) whose frequency-change ratio either from $\mathcal{D}_1$ to $\mathcal{D}_2$ or from $\mathcal{D}_2$ to $\mathcal{D}_1$ is infinite.

Table 1. Weather conditions and Saturday morning activity.

Class $\mathcal{P}$ (suitable for activity)				Class $\mathcal{N}$ (not suitable)			
`outlook`	`temperature`	`humidity`	`windy`	`outlook`	`temperature`	`humidity`	`windy`
overcast	hot	high	false	sunny	hot	high	false
rain	mild	high	false	sunny	hot	high	true
rain	cool	normal	false	rain	cool	normal	true
overcast	cool	normal	true	sunny	mild	high	false
sunny	cool	normal	false	rain	mild	high	true
rain	mild	normal	false				
sunny	mild	normal	true				
overcast	mild	high	true				
overcast	hot	normal	false				

Note that a pattern X's frequency-change rate from $\mathcal{D}_1$ to $\mathcal{D}_2$ is $freq_2(X)$ (its frequency in $\mathcal{D}_2$) divided by $freq_1(X)$. It is infinite if $freq_2(X) > 0$, but $freq_1(X) = 0$.

Next we present an example (Li, Dong, & Ramamohanarao, 2000) to briefly illustrate the basic ideas of DeEPs and the data reduction techniques used in DeEPs.

Example 2.2. Table 1 (Quinlan, 1986) contains a training data set for predicting if the weather is good for some "Saturday morning" activity. The instances, each described by four attributes, are divided into two classes: class $\mathcal{P}$ and class $\mathcal{N}$.

Consider an instance $T = \{sunny, mild, high, true\}$.[3] How does DeEPs predict its class label? First of all, DeEPs calculates the frequency (in both classes) of the proper subsets of T. We organize the proper subsets of T and their frequencies into three groups:

1. those that only occur in Class $\mathcal{N}$ but not in Class $\mathcal{P}$:

Subset of T	Frequency in class $\mathcal{P}$ ($freq_{\mathcal{P}}$)	$freq_{\mathcal{N}}$ (%)
{sunny, high}	0	60
{sunny, mild, high}	0	20
{sunny, high, true}	0	20

2. those that only occur in Class $\mathcal{P}$ but not in Class $\mathcal{N}$:

Subset of T	$freq_{\mathcal{P}}$	$freq_{\mathcal{N}}$
{sunny, mild, true}	11%	0

3. those that occur in both classes:

Subset of T	$freq_{\mathcal{P}}$ (%)	$freq_{\mathcal{N}}$ (%)	Subset of T	$freq_{\mathcal{P}}$ (%)	$freq_{\mathcal{N}}$ (%)
∅	100	100	{mild, high}	22	40
{mild}	44	40	{sunny, true}	11	20
{sunny}	22	60	{high, true}	11	40
{high}	33	80	{mild, true}	11	20
{true}	33	60	{mild, high, true}	11	20
{sunny, mild}	11	20			

Intuitively, we have three main classification arguments:

1. The first group of subsets—which are indeed emerging patterns of Class $\mathcal{N}$ as they do not appear in Class $\mathcal{P}$—favors the prediction that T should be classified as Class $\mathcal{N}$.
2. However, the second group of subsets gives us a contrasting indication that T should be classified as Class $\mathcal{P}$, though this indication is not as strong as that of the first group.
3. The third group also strongly suggests that we should favor Class $\mathcal{N}$ as T's label, though the pattern {mild} contradicts this slightly.

Viewing these EPs in a collective manner, not separately, DeEPs decides that T's label is Class $\mathcal{N}$ since the "aggregation" of EPs occurring in Class $\mathcal{N}$ is much stronger than that in Class $\mathcal{P}$.

Normally, an instance may contain tens or hundreds attributes. To examine all the subsets and to discover relevant EPs contained in such instances by naive enumeration is too expensive. DeEPs is made efficient and scalable for high dimensional data by the following data reduction, concise pattern representation, and important pattern selection techniques.

- Training data sets are reduced firstly by removing those values that do not occur in the test instance being considered, resulting in sparse training instances. Secondly, the maximal ones are selected from the processed training instances,[4] leading to a significant reduction of the volume of the original training data. Using these ideas, the original training data tables can be transformed to be much sparser in both horizontal and vertical directions.
- Borders (Dong & Li, 1999; Li, Ramamohanarao, & Dong, 2000; Li, 2001), which are two-bound structures of the form $\langle \mathcal{L}, \mathcal{R} \rangle$, are used to succinctly represent all EPs contained in an instance without the heavy computation of enumerating all the subsets. Importantly, the borders are derived from the reduced training data sets. This concise pattern representation technique greatly saves the cost of DeEPs.
- Boundary EPs, namely those EPs covered by left bounds $\mathcal{L}$s, are considered in DeEPs' classification. This is because boundary EPs maximally differentiate EPs and non-EPs. The selection also significantly reduces the number of EPs that are used in the classification. For example, the number of boundary EPs used in the mushroom data set is around 81, which is much smaller than the millions of EPs contained in the training data.

Table 2. Reduced training data after removing values which are irrelevant to the instance {sunny, mild, high, true}. A "–" indicates that an item is discarded. There are only two maximal itemsets in the Reduced Class $\mathcal{P}$. They are {*sunny*, *mild*, *true*} and {*mild*, *high*, *true*}. And only 3 maximal itemsets are in the Reduced Class $\mathcal{N}$.

Reduced class $\mathcal{P}$				Reduced class $\mathcal{N}$			
`outlook`	`temperature`	`humidity`	`windy`	`outlook`	`temperature`	`humidity`	`windy`
–	–	high	–	sunny	–	high	–
–	mild	high	–	sunny	–	high	true
–	–	–	–	–	–	–	true
–	–	–	true	sunny	mild	high	–
sunny	–	–	–	–	mild	high	true
–	mild	–	–				
sunny	mild	–	true				
–	mild	high	true				
–	–	–	–				

Detailed discussions and illustrations of these ideas are presented in the subsequent sections. Table 2 illustrates the first stage of the sparsifying effect on both the volume and dimension of $\mathcal{D}_{\mathcal{P}}$ and $\mathcal{D}_{\mathcal{N}}$ after the removal of all values that do not occur in T. Observe that the transformed $\mathcal{D}_{\mathcal{P}}$ and $\mathcal{D}_{\mathcal{N}}$ are sparse, whereas the original $\mathcal{D}_{\mathcal{P}}$ and $\mathcal{D}_{\mathcal{N}}$ are *dense* since there is a value for each attribute of any instance. In Sections 3 and 6, we formally discuss how to select the maximal itemsets from the reduced training instances and how to utilize the reduced training data with the use of borders to gain more efficiency. That is the second stage of the sparsifying effect.

Since EPs usually have very low frequency, they are not suitable to be used individually for classification. We present the *compact summation* method to aggregate the discriminating power contributed by all selected EPs to form classification scores. Interestingly, the compact summation method avoids duplicate contribution of training instances.

3. Instance-based pattern discovery

We provide in this section detailed steps to discover important EPs contained in particular instances. Borders (Dong & Li, 1999; Li, Ramamohanarao, & Dong, 2000; Li, 2001) are used to concisely represent all EPs. The border representation and its associated algorithms

- can avoid enumerating the whole collection of the subsets of an instance in the discovery process, and
- can avoid the enumeration of all patterns in the output.

More importantly, the two bounds themselves are precisely the EPs we are *most* interested in.

Definition 3.1. A border, denoted $\langle \mathcal{L}, \mathcal{R} \rangle$, is defined as an ordered pair of two bounds $\mathcal{L}$ and $\mathcal{R}$ such that $\mathcal{L}$ and $\mathcal{R}$ are two anti-chains[5] satisfying

- $\forall X \in \mathcal{L}, \exists Y \in \mathcal{R}$ such that $X \subseteq Y$,
- $\forall Y \in \mathcal{R}, \exists X \in \mathcal{L}$ such that $Y \supseteq X$.

In this work, bounds are sets of itemsets. Semantically, the border $\langle \mathcal{L}, \mathcal{R} \rangle$ represents a collection which consists of patterns (itemsets) Z satisfying $X \subseteq Z \subseteq Y$ for any $X \in \mathcal{L}$ and any $Y \in \mathcal{R}$. This collection is denoted $[\mathcal{L}, \mathcal{R}]$ (Dong & Li, 1999; Li, 2001).

For example, the border $\langle\{\{a\}, \{b\}\}, \{\{a, b, c\}, \{b, c, d\}\}\rangle$ represents the collection $\{\{a\}, \{b\}, \{a, b\}, \{a, c\}, \{b, c\}, \{b, d\}, \{c, d\}, \{a, b, c\}, \{b, c, d\}\}$. We also say that the latter can be concisely represented by the former.

3.1. Three steps in the discovery

Assume a classification problem has a set $\mathcal{D}_p = \{P_1, \ldots, P_m\}$ of positive training instances, a set $\mathcal{D}_n = \{N_1, \ldots, N_n\}$ of negative training instances, and a set of test instances.

Consider a fixed test instance T. DeEPs uses the steps below to discover two borders representing two special sub-collections of the subsets of T; those subsets are required to satisfy the definition of emerging patterns with respect to $\mathcal{D}_p$ and $\mathcal{D}_n$.

1. Take the intersection of each training instance with T, namely $T \cap P_1, \ldots, T \cap P_m$ and $T \cap N_1, \ldots, T \cap N_n$. This operation is equivalent to the removal of irrelevant training values. We will discuss later how to conduct this intersection operation when continuous attributes are present.
2. Select the maximal itemsets from $\{T \cap P_1, \ldots, T \cap P_m\}$, and similarly from $\{T \cap N_1, \ldots, T \cap N_n\}$. Denote the former collection of maximal itemsets as $max\_\mathcal{R}_p$ and the latter as $max\_\mathcal{R}_n$.
3. Discover two EP borders: (i) Discover a border which represents those subsets of T which occur in $\mathcal{D}_p$ but not in $\mathcal{D}_n$, by taking the *border difference* operation $[\{\emptyset\}, max\_\mathcal{R}_p] - [\{\emptyset\}, max\_\mathcal{R}_n]$; (ii) On the other hand, to discover a border which represents those subsets of T which occur in $\mathcal{D}_n$ but not in $\mathcal{D}_p$ class, by similarly taking another border difference operation $[\{\emptyset\}, max\_\mathcal{R}_n] - [\{\emptyset\}, max\_\mathcal{R}_p]$.

Observe that, by intersecting the training data with T, all zero-frequency subsets of T are removed from the training data. By selecting the maximal itemsets, the completeness of all non-zero-frequency subsets of T is retained. By conducting the difference operation, the boundary EPs contained in T can be derived.

3.2. Important knowledge: The two borders

Step 3 above is used to efficiently discover the two borders. Observe that the compressed training data $\langle\{\emptyset\}, max\_\mathcal{R}_p\rangle$ or $\langle\{\emptyset\}, max\_\mathcal{R}_n\rangle$ can still represent large collections of sets, despite the previous tremendous reduction. To enumerate all itemsets covered by $\langle\{\emptyset\},$

$max\_\mathcal{R}_p\rangle$ or $\langle\{\emptyset\}, max\_\mathcal{R}_n\rangle$ and to conduct the difference operation naively on them is costly. DeEPs uses the efficient JEPPRODUCER algorithm (Li, Dong, & Ramamohanarao, 2000; Li, Ramamohanarao, & Dong, 2000; Li, 2001) to conduct the border difference operation

$$[\{\emptyset\}, max\_\mathcal{R}_p] - [\{\emptyset\}, max\_\mathcal{R}_n]$$

and

$$[\{\emptyset\}, max\_\mathcal{R}_n] - [\{\emptyset\}, max\_\mathcal{R}_p].$$

The algorithm only manipulates elements in the bounds $max\_\mathcal{R}_p$ and $max\_\mathcal{R}_n$ without the enumeration of $[\{\emptyset\}, max\_\mathcal{R}_p]$ and $[\{\emptyset\}, max\_\mathcal{R}_n]$. The algorithm produces a border

$$\langle ep\mathcal{L}_p, ep\mathcal{R}_p\rangle \quad \text{and, respectively,} \quad \langle ep\mathcal{L}_n, ep\mathcal{R}_n\rangle$$

as a succinct representation of all the EPs.

The JEPPRODUCER (Li, Dong, & Ramamohanarao, 2000; Li, Ramamohanarao, & Dong, 2000; Li, 2001) will be briefly reviewed in Section 6.

3.3. Comprehensibility of the borders

Suppose $T = \{a, b, c, d, e, f, g, h\}$ and

$$\langle ep\mathcal{L}_p, ep\mathcal{R}_p\rangle = \langle\{\{a, b\}, \{c, d\}\}, \{\{a, b, c, d\}\}\rangle$$
$$\langle ep\mathcal{L}_n, ep\mathcal{R}_n\rangle = \langle\{\{e\}, \{f, g, h\}\}, \{\{c, d, e\}, \{e, f, g, h\}\}\rangle.$$

Taking a pattern, for example $\{a, b\}$, in the left bound of the border $\langle ep\mathcal{L}_p, ep\mathcal{R}_p\rangle$, we know that

- $\{a, b\}$ occurs most frequently in the positive data but never occur in the negative data.
- The pattern $\{a\}$ or $\{b\}$ may have larger frequency than their parent $\{a, b\}$ in the positive data, but $\{a\}$ or $\{b\}$ definitely occur in the negative data as well.

The left boundary elements in $\langle ep\mathcal{L}_n, ep\mathcal{R}_n\rangle$ can be understood in a symmetrical manner with respect to the positive and negative data.

The right boundary elements in the above borders are always the most specific subsets of T which occur in one class but not the other.

A special border $\langle\emptyset, \emptyset\rangle$ is used to represent an empty collection. We have occasionally seen the case of $ep\mathcal{L}_p = ep\mathcal{R}_p = \emptyset$ in our experiments. This happens only when two or more identical instances of a considered test instance exist in both positive and negative training data. Such a training set is obviously not clean.

Note that the patterns covered by $\langle ep\mathcal{L}_p, ep\mathcal{R}_p\rangle$ and $\langle ep\mathcal{L}_n, ep\mathcal{R}_n\rangle$ are significantly different from patterns covered by a version space (Mitchell, 1977, 1982; Hirsh, 1994) where each pattern must have a 100% frequency in the positive data. Our patterns relax this strict constraint by considering the most occurrence (not the full completeness). Conceptually, our patterns are very similar to patterns covered by a disjunctive version space (Sebag, 1996). However, no patterns like ours are explicitly derived in the work of Sebag (1996).

3.4. The efficiency of the data reduction process

With the intersection operation in step 1, the dimension of the training data is substantially reduced, as many values from the original training data do not occur in the test instance T. Secondly, with the maximal itemset selection step, the volume of the training data is also substantially reduced since itemsets $T \cap P_i$ are frequently contained in some other itemsets $T \cap P_j$. Then, $max\_\mathcal{R}_p$ can be viewed as a compressed $\mathcal{D}_p$, and $max\_\mathcal{R}_n$ a compressed $\mathcal{D}_n$.

We use the mushroom data set to demonstrate this point.[6] The original mushroom data has a volume of 3788 edible training instances, with 22 attributes per instance. The average number of items (or length) of the 3788 processed instances (by intersection with a test instance) is 11, and those processed instances are further compressed into 7 maximal itemsets. Thus there is a reduction in volume from 3788 to 7, and a reduction in dimension from 22 to 11. See Table 3 for more details. We can see this reduction effect from another example previously shown in Table 2 which is about weather conditions data.

3.5. Neighborhood-based intersection for discretizing continuous attributes

Suppose `attri_A` is a continuous attribute and its domain is [0, 1]. All `attri_A` values in the training instances can be normalized into the range of [0, 1] if its domain is not [0, 1]; and all testing values can be scaled down or up similarly. In this work, we use the formula $\frac{x-min}{max-min}$ to scale every value x of any attribute `attri_A`, where *max* and *min* are respectively the largest and smallest values of `attri_A` in the training data.

Given a training instance S, $T \cap S$ will contain the `attri_A` value of T, if the `attri_A` value of S is in the neighborhood

$$[x_1 - \alpha, x_1 + \alpha],$$

where x_1 is the normalized `attri_A` value for T. The parameter α is called the *neighborhood factor*, which can be used to adjust the length of the neighborhood. Usually, we set it as

Table 3. Data reduction in the mushroom data set after intersecting with an instance.

	Volume		
Mushroom data	Poisonous	Edible	Dimension
Original	3525	3788	22
After reduction	9	7	11

Table 4. The original training data are transformed into binary transaction data after removing values which are irrelevant to a test instance $T = \{\texttt{square}, 0.30, 0.25\}$. A chosen neighborhood of 0.30 is [0.28, 0.32] and a chosen neighborhood of 0.25 is [0.23, 0.27].

Original training data				Binary data		
Circle	0.31	0.24	$\rightarrow$	0	1	1
Square	0.80	0.70	$\rightarrow$	1	0	0
Diamond	0.48	0.12	$\rightarrow$	0	0	0

$T = \{\texttt{square}, 0.30, 0.25\}$.

0.12. We use Table 4 to demonstrate how DeEPs deals with both categorical and continuous attributes and how it transforms the intersected data into binary transaction data, when a test instance is given.

4. Classification by DeEPs

The discovered boundary EPs not only provide differentiating knowledge for us to understand the instance T, but can also be used to classify T if it is a test instance. In this section, we describe how to make use of the frequencies of the boundary EPs to predict a class label for a test instance.

Note that $\langle ep\mathcal{L}_p, ep\mathcal{R}_p\rangle$ and $\langle ep\mathcal{L}_n, ep\mathcal{R}_n\rangle$ represent two non-overlapping sub-collections of the whole subsets of the instance T. The patterns in the first sub-collection occur in the positive class only; particularly, the patterns in $ep\mathcal{L}_p$ occur in the positive class most frequently. Similarly, the patterns covered by $ep\mathcal{L}_n$ occur in the negative class only and most frequently. It is apparent that the patterns in the first sub-collection favor the prediction of T as positive class, but the patterns in the second sub-collection favor the prediction as negative class. As sometimes the two borders can represent a very large number of EPs (of the order of 10^6 in mushroom, waveform, ionosphere, and sonar data), we select important representatives (Li, Dong, & Ramamohanarao, 2001; Li, Ramamohanarao, & Dong, 2000) and collectively compare their frequencies to make a prediction.

We select the boundary EPs covered in $ep\mathcal{L}_p$ and $ep\mathcal{L}_n$ as representative EPs. The ideas of this selection include: (i) The left boundary elements always have larger frequency than their supersets; and (ii) the proper subsets of those left boundary elements are no longer EPs.

4.1. *Determining collective scores for classification*

After selecting the boundary EPs, DeEPs proceeds to calculate classification scores based on the frequencies of the EPs. The collective score of T for any specific class C (positive class, negative class, or other classes where a database contains over two classes) is calculated by aggregating the frequencies of the selected EPs in class C. The aggregation of the frequencies of the individual EPs is performed by the *compact summation* method.

Definition 4.1. The **compact summation** of the frequencies in $\mathcal{D}_C$ of a collection of EPs is defined as the percentage of instances in $\mathcal{D}_C$ that contain one or more of the EPs; this percentage is called the **compact score** of T for class C, that is,

$$compactScore(C) = \frac{count_{\mathcal{D}_C}(SEP)}{|\mathcal{D}_C|}$$

where *SEP* is the collection of EPs and $count_{\mathcal{D}_C}(SEP)$ is the number of instances in $\mathcal{D}_C$ that contain one or more EPs in *SEP*.

The purpose of this aggregation is to avoid counting duplicate contribution of training instances. Suppose the selected EPs are $\{e_1, \ldots, e_4\}$ and $\mathcal{D}_C = \{X_1, X_2, \ldots, X_n\}$, and assume X_1 contains e_1, e_2, and e_3. Then, the instance X_1 is counted three times if the frequency values of e_1, e_2, and e_3 are linearly added together. However, X_1 is only counted once by the method of compact summation.

DeEPs makes its final decision only after the compact scores, formed by compact summation, for all classes are calculated. DeEPs simply assigns to T the class where T obtains the largest score. A majority rule is used to break ties.

4.2. *Handling data containing more than two classes*

We briefly describe how DeEPs can be extended to handle more classes. Suppose a database containing 3 classes of training instances $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$. Similar to the discussion above, DeEPs discovers the borders of the EPs with respect to $\mathcal{D}'_1$ and $(\mathcal{D}'_2 \cup \mathcal{D}'_3)$, those with respect to $\mathcal{D}'_2$ and $(\mathcal{D}'_1 \cup \mathcal{D}'_3)$, and those with respect to $\mathcal{D}'_3$ and $(\mathcal{D}'_1 \cup \mathcal{D}'_2)$. Here $\mathcal{D}'_1$, $\mathcal{D}'_2$, and $\mathcal{D}'_3$ are respectively the reduced $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$ after their intersection with T. The compact scores for the three classes are then calculated based on three groups of the left boundary EPs. The classifier chooses the class where the largest compact score is obtained as T's class label.

5. An illustrating example

So far, we have described how DeEPs uses data reduction, border representation, and the EP selection idea to efficiently conduct a knowledge discovery and classification task. We continue with the weather conditions data set described in Example 2.2 to show the whole process that DeEPs takes. The borders involved in this example can be derived by the algorithms discussed in the next section. We emphasize that the work flow below is significantly different from the one discussed in Example 2.2 where DeEPs had taken a naive approach.

1. Border representations of the EPs.

   EPs in $\mathcal{D}_\mathcal{N}$: $\texttt{border}_n = \langle\{\{s, high\}\}, \{\{s, m, high\}, \{s, high, t\}\}\rangle$;

   EPs in $\mathcal{D}_\mathcal{P}$: $\texttt{border}_p = \langle\{\{s, m, t\}\}, \{\{s, m, t\}\}\rangle$.

   Observe that the two borders correspond to the first two subset groups in Example 2.2. Here, s, m, t are short for *sunny*, *mild*, and *true* respectively.

2. Selection of the boundary EPs. DeEPs selects the left bound of $\texttt{border}_n$, i.e., $\{\{s, \textit{high}\}\}$, and the left bound of $\texttt{border}_p$, i.e., $\{\{s, m, t\}\}$ for class $\mathcal{N}$ and class $\mathcal{P}$ respectively in classification.
3. Calculation of scores by compaction summation and classification.

$$\textit{compactScore}(\textit{Class}\mathcal{N}) = \frac{3}{5} = 0.6$$
$$\textit{compactScore}(\textit{Class}\mathcal{P}) = \frac{1}{9} = 0.11$$

Finally, DeEPs assigns class $\mathcal{N}$ as T's label because the score based on the collection of the selected EPs of $\mathcal{D}_{\mathcal{N}}$ is larger than that of $\mathcal{D}_{\mathcal{P}}$. This is the same prediction as made by the naive DeEPs in Section 2.

6. Border-based algorithms for DeEPs

Recall that the first step of the discovery phase of DeEPs is the intersection of $\mathcal{D}_p$ with T and the intersection of $\mathcal{D}_n$. Following that, two reduced training data sets are obtained. We denote them as $\mathcal{D}'_p$ and $\mathcal{D}'_n$. Subsequently, DeEPs finds the maximal elements from $\mathcal{D}'_p$ and from $\mathcal{D}'_n$. Then the border difference operation, conducted by our backbone algorithm JEPPRODUCER (Li, Dong, & Ramamohanarao, 2001; Li, Ramamohanarao, & Dong, 2000; Li, 2001), is used to derive the two borders $\langle ep\mathcal{L}_p, ep\mathcal{R}_p\rangle$ and $\langle ep\mathcal{L}_n, ep\mathcal{R}_n\rangle$.

Next, we describe how JEPPRODUCER is used to conduct the border difference, for example, $[\{\emptyset\}, max\_\mathcal{R}_p] - [\{\emptyset\}, max\_\mathcal{R}_n]$. Denote $max\_\mathcal{R}_p = \{A_1, A_2, \ldots, A_{k_1}\}$ and $max\_\mathcal{R}_n = \{B_1, B_2, \ldots, B_{k_2}\}$. A pseudo code of the algorithm is:

```
JEPPRODUCER($\langle\{\emptyset\}, \{A_1, \ldots, A_{k_1}\}\rangle$, $\langle\{\emptyset\}, \{B_1, \ldots, B_{k_2}\}\rangle$)
  ;; return $\langle\mathcal{L}, \mathcal{R}\rangle$ such that $[\mathcal{L}, \mathcal{R}] = [\{\emptyset\}, \{A_1, \ldots, A_{k_1}\}] - [\{\emptyset\}, \{B_1, \ldots, B_{k_2}\}]$
  1) $\mathcal{L} \leftarrow \{\}$; $\mathcal{R} \leftarrow \{\}$;
  2) for $j$ from 1 to $k_1$ do
  3)   if some $B_{k_i}$ is a superset of $A_j$ then continue;
  4)   border = BORDER-DIFF($\langle\{\emptyset\}, \{A_j\}\rangle$, $\langle\{\emptyset\}, \{B_1, \ldots, B_{k_2}\}\rangle$);
  5)   $\mathcal{R} = \mathcal{R} \cup$ the right bound of border;
  6)   $\mathcal{L} = \mathcal{L} \cup$ the left bound of border;
  7) return $\langle\mathcal{L}, \mathcal{R}\rangle$;
```

The subroutine BORDER-DIFF (Dong & Li, 1999; Li, 2001) is called multiple times in the algorithm. A pseudo code of BORDER-DIFF is given as follows:

```
BORDER-DIFF($\langle\{\emptyset\}, \{U\}\rangle$, $\langle\{\emptyset\}, \{S_1, S_2, \ldots, S_k\}\rangle$)
;; return border of $[\{\emptyset\}, \{U\}] - [\{\emptyset\}, \{S_1, S_2, \ldots, S_k\}]$
1) initialize $\mathcal{L}$ to $\{\{x\} \mid x \in U - S_1\}$;
2) for $i = 2$ to $k$ do
3)   $\mathcal{L} \leftarrow \{X \cup \{x\} \mid X \in \mathcal{L}, x \in U - S_i\}$;
4)   remove all $Y$ in $\mathcal{L}$ that are not minimal;
5) return $\langle\mathcal{L}, \{U\}\rangle$;
```

As both BORDER-DIFF and JEPPRODUCER operate on the boundary elements of borders, the algorithms are very efficient according to our extensive experimental results. For the proof of the algorithms and more detailed description of the experimental results, the readers are referred to Dong and Li (1999), Li, Ramamohanarao, and Dong (2000), and Li (2001).

7. Ranking the discovered patterns

Our DeEPs system has two aspects of roles: One is to provide EPs represented by borders, the other is to predict a class label for previously unseen instances. Those boundary EPs in $ep\mathcal{L}_p$ and $ep\mathcal{L}_n$ have been shown to be very important in DeEPs. Note that the size of $ep\mathcal{L}_p$ or $ep\mathcal{L}_n$ can be large. Especially when mining a large database, the number of patterns contained in $ep\mathcal{L}_p$ and $ep\mathcal{L}_n$ can easily exceed the capabilities of a human user to go through the bounds and identify the most important patterns. Simply listing the patterns is not sufficient to examine the importance of the individual patterns, and manually examining a large number of patterns may miss important ones.

This section presents two ranking methods to order the patterns contained in $ep\mathcal{L}_p$ or $ep\mathcal{L}_n$.

1. **Frequency-based ranking**. Frequency reflects the occurrence of a pattern in a data set. Usually, the larger the frequency of an EP is, the more important the EP is. We rank the patterns in $ep\mathcal{L}_p$ (and $ep\mathcal{L}_n$) in a descending order with respect to their frequency. By examining the top-ranked ones in this order, human users would know which subsets of a considered instance have a sharply changing frequency from one class of training data to the other training data.
2. **Length-based ranking**. The length of a pattern shows the similarity between the pattern and the raw training instances. With more items matched, the pattern becomes more similar to a training instance. Prior to presenting the patterns to human users, the patterns can be sorted into a descending order with respect to the patterns' length. Therefore, the users can easily get the longest pattern whose frequency maximally differs in the two classes of training data.

Another interesting problem is to rank EPs with finite frequency-change rate. Recall that all patterns in $ep\mathcal{L}_p$ and $ep\mathcal{L}_n$ have the same frequency-change rate of the infinity. However, the frequency-change rate of the proper subsets of boundary EPs is finite and various. A descending order ranked according to frequency-change rate can provide users those patterns whose frequency changes with the largest *finite* degree.

Sometimes, patterns with finite changing degree are more interesting than those with the infinite degree. Suppose a pattern X have $freq_p(X) = 5\%$ and $freq_n(X) = 0\%$, another pattern Y have $freq_p(Y) = 50\%$ and $freq_n(Y) = 0.1\%$. So, the pattern X has a frequency-change rate of the infinity while the pattern Y has a rate of 500. Obviously, the pattern Y is more interesting than X as the former has a much larger coverage in the positive data than the latter, and both of them have no or almost no occurrence in the negative data.

Ranking discovered patterns is an intensively studied topic in data mining, the readers are referred to Klemettinen et al. (1994), Silberschatz and Tuzhilin (1996), Dong and Li (1998), Padmanabhan and Tuzhilin (1998), Bayardo and Agrawal (1999), Sahar (1999), and

Hilderman and Hamilton (2001) for other subjective and objective measurements originated in information theory, statistics, ecology, and economics.

8. Performance evaluation: Accuracy, speed, and scalability

We report in this section the performance of our method in comparison to the performance of k-NN and C5.0. We used 40 data sets, taken from the UCI Machine Learning Repository (Blake & Murphy, 1998), for experimental evaluation. For each dataset-algorithm combination, the test accuracies were measured by a ten-fold stratified cross validation (CV-10). Each of the exclusive ten-fold test instances were randomly selected from the original data sets. The same splits of the data were used for all the three classification algorithms. These experiments were carried out on a 500 MHz PentiumIII PC, with 512 M bytes of RAM.

8.1. Accuracy

DeEPs is first compared with k-nearest neighbor. In this work, k is set as 3. Empirically, other values for k were not reported to produce better accuracy results. Then, DeEPs is compared with C5.0 [Release 1.12], a commercial version of C4.5 (Quinlan, 1993). The experimental results are summarized in Table 5.

The data sets are listed in Column 1 of Table 5, and their properties in Column 2. Columns 3 and 4 show the CV-10 average accuracy of DeEPs, when the neighborhood factor α is fixed as 0.12 for all data sets, and respectively when different α is selected for the data sets. (This will be explained in Section 8.2). Note that for data sets such as chess, flare, splice, mushroom, voting, soybean-l, t-t-t, and zoo which do not contain any continuous attributes, DeEPs does not require an α. The accuracies of k-nearest neighbor and C5.0 are listed respectively in Columns 5 and 6. Columns 7 and 8 respectively show the average time used by DeEPs and k-nearest neighbor to test one instance.

We next discuss the case when the neighborhood factor is fixed as 0.12 for all the data sets. For the mushroom data set, DeEPs, k-NN, and C5.0 all can achieve 100% testing accuracy. For the remaining data sets, we highlight some interesting points.

1. DeEPs versus k-NN.
   - Both DeEPs and k-NN perform equally accurately on soybean-small (100%) and on iris (96%).
   - DeEPs wins on 26 data sets; k-NN wins on 11. that of k-NN.
   - The speed of DeEPs is about 1.5 times slower than that of k-NN. The main reason is that DeEPs needs to conduct the border operations.
2. DeEPs versus C5.0.
   - DeEPs wins on 25 data sets; C5.0 wins on 14.
   - DeEPs is slower than C5.0. However, DeEPs takes an instance-based learning strategy.
3. DeEPs, k-NN, and C5.0.
   - DeEPs wins on 20 data sets; k-NN wins on 7; C5.0 wins on 14. (DeEPs and k-NN reach a draw on soybean-s and iris.)

Table 5. Accuracy of DeEPs in comparison to those of k-nearest neighbor and C5.0.

		DeEPs					
Data sets	inst, attri classes	$\alpha = 0.12$	Dynamical α	k-NN $k = 3$	C5.0	Time (s.) DeEPs	Time (s.) k-NN
Australia	690, 14, 2	84.78	88.41* (.05)	66.69	**85.94**	0.054	0.036
Breast-w	699, 10, 2	96.42	96.42 (.12)	**96.85**	95.43	0.055	0.036
Census	30162, 16, 2	**85.93**	85.93* (.12)	75.12	85.80	2.081	1.441
Chess	3196, 36, 2	97.81	97.81	96.75	**99.45**	0.472	0.145
Cleve	303, 13, 2	**81.17**	84.21* (.15)	62.64	77.16	0.032	0.019
Diabete	768, 8, 2	**76.82**	76.82* (.12)	69.14	73.03	0.051	0.039
Flare	1066, 10, 2	**83.50**	83.50	81.62	82.74	0.028	0.016
German	1000, 20, 2	**74.40**	74.40 (.12)	63.1	71.3	0.207	0.061
Heart	270, 13, 2	**81.11**	82.22 (.15)	64.07	77.06	0.025	0.013
Hepatitis	155, 19, 2	**81.18**	82.52 (.11)	70.29	74.70	0.018	0.011
Letter-r	20000, 16, 26	93.60	93.60* (.12)	**95.58**	88.06	3.267	1.730
Lymph	148, 18, 4	**75.42**	75.42 (.10)	74.79	74.86	0.019	0.010
Pima	768, 8, 2	**76.82**	77.08* (.14)	69.14	73.03	0.051	0.038
Satimage	6435, 36, 6	88.47	88.47* (.12)	**91.11**	86.74	2.821	1.259
Segment	2310, 19, 7	94.98	95.97* (.05)	95.58	**97.28**	0.382	0.365
Shuttle-s	5800, 9, 7	97.02	99.62* (.01)	99.54	**99.65**	0.438	0.295
Splice	3175, 60, 3	69.71	69.71	70.03	**94.20**	0.893	0.248
Vehicle	846, 18, 4	70.95	74.56* (.15)	65.25	**73.68**	0.134	0.089
Voting	433, 16, 2	95.17	95.17	92.42	**97.00**	0.025	0.012
Waveform	5000, 21, 3	**84.36**	84.36* (.12)	80.86	76.5	2.522	0.654
Yeast	1484, 8, 10	**59.78**	60.24* (.10)	54.39	56.14	0.096	0.075
Anneal	998, 38, 6	**94.41**	95.01 (.06)	89.70	93.59	0.122	0.084
Auto	205, 25, 7	67.65	72.68 (.035)	40.86	**83.18**	0.045	0.032
crx	690, 15, 2	**84.18**	88.11* (.035)	66.64	83.91	0.055	0.038
Glass	214, 9, 7	58.49	67.39 (.10)	67.70	**70.01**	0.021	0.017
Horse	368, 28, 2	84.21	85.31* (.035)	66.31	**84.81**	0.052	0.024
Hypo	3163, 25, 2	97.19	98.26 (.05)	98.26	**99.32**	0.275	0.186
Ionosph	351, 34, 2	86.23	91.24 (.05)	83.96	**91.92**	0.147	0.100
Iris	150, 4, 3	**96.00**	96.67* (.10)	**96.00**	94.00	0.007	0.006
Labor	57, 16, 2	87.67	87.67* (.10)	**93.00**	83.99	0.009	0.008
Mushroom	8124, 22, 2	**100.0**	100.0	**100.0**	**100.0**	0.436	0.257
Nursery	12960, 8, 5	**99.04**	99.04	98.37	97.06	0.290	0.212
Pendigits	10992, 16, 10	98.21	98.44 (.18)	**99.35**	96.67	1.912	0.981
Sick	4744, 29, 2	94.03	96.63 (.05)	93.00	**98.78**	0.284	0.189
Sonar	208, 60, 2	**84.16**	86.97* (.11)	82.69	70.20	0.193	0.114
Soybean-s	47, 34, 4	**100.0**	100.0* (.10)	**100.0**	98.00	0.022	0.017
Soybean-l	683, 35, 19	90.08	90.08	91.52	**92.96**	0.072	0.051
t-t-t	958, 9, 2	**99.06**	99.06	98.65	86.01	0.032	0.013
Wine	178, 13, 3	**95.58**	96.08* (.11)	72.94	93.35	0.028	0.019
Zoo	101, 16, 7	**97.19**	97.19	93.93	91.26	0.007	0.005

An important conclusion we can reach here is that DeEPs is an accurate instance-based lazy classifier. Its accuracy is generally comparable to, and is often better than, C5.0 and k-nearest neighbor. However, the speed of DeEPs needs improvement in order to match other classifiers.

The boundary EPs and the central idea of compactly summarizing those EPs are perhaps the most important factors contributing to the high accuracy of DeEPs. Usually, C5.0 produces a set of divide-and-conquer rules as small as possible. Thus, the tree may miss some small sub-contexts[7] of the training data. So, if a test instance happens to match those small sub-contexts. C5.0 would undertake a hard decision. However, DeEPs can efficiently discover rules for regulating those small sub-contexts though the frequency of the rule patterns is not outstanding. This is a reason why DeEPs can be better than C5.0 in some applications.

8.2. Higher accuracy by DeEPs with different α

The neighborhood factor can be fixed for all data sets. However, we observed that different neighborhood factors α can cause accuracy variance, although slight, on testing data. When α is too large, it may happen that originally different instances from different classes can be transformed into the same binary instance; consequently, the inherent discriminating features among these instances disappear. When α is too small, nearly identical attribute values may be considered different, thus useful discrimination information in the training data may be overlooked.

The effect of the neighborhood factor on testing accuracies is studied in Li, Ramamohanarao, and Dong (2001). Experiments have been conducted when α varied from 0.02, 0.05, 0.08, 0.10, 0.15, to 0.20. The corresponding accuracies on the australian and german data sets are plotted in figure 1. Note that the two curves in this figure are shaped in a different manner. The accuracy in the left curve peaks when $\alpha = 0.05$, and then goes down with increasing α values. However, the right curve starts with a decline till $\alpha = 0.1$, and then climbs to its summit at $\alpha = 0.2$. These facts strongly indicate that data sets have different properties. They motivated the idea of selecting an "optimal" neighborhood for a given data sets by using a guidance role played by a partial training data. This idea (Li, Ramamohanarao, & Dong, 2001) is developed in Section 9.

The accuracy results by DeEPs with different α are shown in the column 4 of Table 5.

How to determine a proper value of α for different data sets or even different continuous attributes within the same data set is still an interesting problem. We will study this problem in the future work.

8.3. Comparison with other classifiers

DeEPs is also compared with another four classifiers: CBA (Liu, Hsu, & Ma, 1998), LB (Meretakis & Wuthrich, 1999), the Naive Bayesian classifier (NB) (Duda & Hart, 1973; Langley, Iba, & Thompson, 1992), and TAN (Friedman, Geiger, & Goldszmidt, 1997). We did not implement their algorithms. We cite the accuracy results of these classifiers from the literature work. So, their data set partitioning methods may be different. As a result, this comparison is not as strict as the above discussion when DeEPs is compared with k-NN and

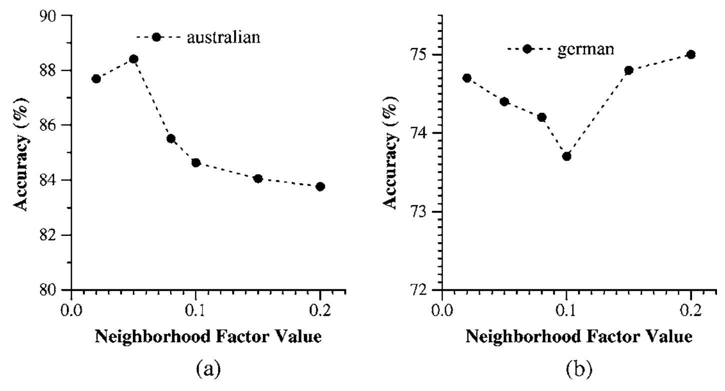

Figure 1. Different neighborhoods of a test instance can produce different accuracies by DeEPs. Partial training data can be used to select proper neighborhoods of test instances to improve performance of the classifier. (a) Accuracy varies in the range of [84.20%, 88.50%] on the australian data set. (b) Accuracy varies in the range of [73.70%, 75.00%] on the german data set.

C5.0. However, the comparison is also meaningful as all the data were randomly shuffled and all the instances were tested once.

Table 6 lists the results. Columns 4, 5, 6, and 7 show the accuracy results of CBA, LB, NB, and TAN respectively; these results, for the first 21 data sets, were copied exactly from Table 1 of Meretakis and Wuthrich (1999). For the remaining data sets, we selected, for CBA and C4.5, the *best* result from Table 1 of Liu, Hsu, and Ma (1998). A dash indicates that we were not able to find a previously reported result and “N/A” means the classifier is not applicable to the data sets.

We draw some interesting points as follows:

- Among the first 21 data sets where results of the other four classifiers are available, DeEPs achieves the best accuracy on 11 data sets. CBA, LB, NB, and TAN achieve the best accuracy on 1, 5, 4 and 3 data sets respectively. It can be seen that DeEPs in general outperforms the other classifiers.
- For the 37 data sets where the results for CBA are available, DeEPs achieves the best accuracy on 22 data sets. CBA achieves the best accuracy on 15 data sets. In addition, DeEPs can reach the 100% accuracy on mushroom, 99.04% on nursery, and 98.21% on pendigits. (The accuracies of CBA for these three data sets were not available.)
- DeEPs can reach very high accuracy on large data sets such as letter (93.60%), pendigits (98.21%), satimage (88.47%), waveform (84.36%), mushroom (100%), nursery (99.04%), sonar (84.16%), shuttle-small (97.02%), and hypo (97.19%).

8.4. Speed and scalability

The primary metric for evaluating classifier performance is classification accuracy. We have already shown DeEPs is an accurate classifier and it is generally superior to the other classifiers. In this section, we discuss the decision speed by DeEPs and then demonstrate the noticeable scalability of DeEPs over the number of training instances.

Table 6. Accuracy of DeEPs in comparison to CBA, LB, NB, and TAN.

	DeEPs					
Data sets	$\alpha = .12$	Dynamical α	CBA	LB	NB	TAN
Australian	84.78	88.41* (.05)	85.51	**85.65**	**85.65**	85.22
Breast-w	96.42	96.42 (.12)	95.28	96.86	**97.00**	N/A
Census-inc	**85.93**	85.93* (.12)	85.67	85.11	84.12	N/A
Chess	97.81	97.81	**98.12**	90.24	87.15	92.12
Cleve	81.17	84.21* (.15)	77.24	82.19	**82.78**	N/A
Diabete	**76.82**	76.82* (.12)	72.9	76.69	75.13	76.56
Flare	**83.50**	83.50	83.11	81.52	79.46	82.64
German	74.40	74.40 (.12)	73.2	**74.8**	74.1	72.7
Heart	81.11	82.22 (.15)	81.87	82.22	82.22	**83.33**
Hepatitis	81.18	82.52 (.11)	80.20	**84.5**	83.92	N/A
Letter	**93.60**	93.60* (.12)	51.76	76.4	74.94	85.7
Lymph	75.42	75.42 (.10)	77.33	**84.57**	81.86	83.76
Pima	**76.82**	77.08* (.14)	73.1	75.77	75.9	75.77
Satimage	**88.47**	88.47* (.12)	84.85	83.9	81.8	87.2
Segment	**94.98**	95.97* (.05)	93.51	94.16	91.82	93.51
Shuttle-s	97.02	99.62* (.01)	99.48	99.38	98.7	**99.64**
Splice	69.71	69.71	70.03	**94.64**	**94.64**	**94.63**
Vehicle	**70.95**	74.56* (.15)	68.78	68.8	61.12	70.92
Voting	**95.17**	95.17	93.54	94.72	90.34	93.32
Waveform	**84.36**	84.36* (.12)	75.34	79.43	78.51	79.13
Yeast	**59.78**	60.24* (.10)	55.1	58.16	58.05	57.21
Anneal	94.41	95.01 (.06)	**98.1**	–	–	–
Automobile	67.65	72.68 (.035)	**79.00**	–	–	–
crx	84.18	88.11* (.035)	**85.9**	–	–	–
Glass	58.49	67.39 (.10)	**72.6**	–	–	–
Horse	**84.21**	85.31* (.035)	82.1	–	–	–
Hypo	97.19	98.26 (.05)	**98.4**	–	–	–
Ionosphere	86.23	91.24 (.05)	**92.1**	–	–	–
Iris	**96.00**	96.67* (.10)	92.9	–	–	–
Labor	**87.67**	87.67* (.10)	83.00	–	–	–
Mushroom	100.0	100.0	–	–	–	–
Nursery	**99.04**	99.04	–	–	–	–
Pendigits	**98.21**	98.44 (.18)	–	–	–	–
Sick	94.03	96.63 (.05)	**97.3**	–	–	–
Sonar	**84.16**	86.97* (.11)	78.3	–	–	–
Soybean-small	**100.0**	100.0* (.10)	98.00	–	–	–
Soybean-large	90.08	90.08	**92.23**	–	–	–
tic-tac-toe	99.06	99.06	**100.0**	–	–	–
Wine	**95.58**	96.08* (.11)	91.6	–	–	–
Zoo	**97.19**	97.19	94.6	–	–	–

Overall, DeEPs is a relatively slow classifier, especially compared with C5.0. However, decision time per instance by DeEPs is typically a small fraction of a second. For only 5 of the 40 data sets (census-inc, letter, satimage, pendigits, and waveform) decision time per instance exceeds 1 second. All these five data sets have a very large volume of training instances, high dimensions, or both. Clearly, this speed is adequate for most applications such as credit-worthiness check, disease diagnosis, etc.

As mentioned earlier, a primary advantage of DeEPs is that it can handle new training instances without the need to re-train the classifier. For eager learning classifiers, the re-training after modifications to the previous training instances can take a considerable amount of time.

To examine the scalability of DeEPs on the number of training instances, we randomly selected 90% of the original data set as the training data set and 10% as the testing data set. Then, we selected 20%, 40%, 60%, and 80% of the training data set to form 4 new training data sets, all with the same number of attributes. All the new data sets (including the testing data sets) have a class distribution similar to the original one. Figure 2(a) shows the linear scalability of the average decision time per instance of DeEPs when the number of training instances increases in tic-tac-toe, mushroom, sonar, and census-inc. This scalability is a consequence of the instance intersection operation and the maximum itemset selection in the learning phase of DeEPs as discussed in Section 3.

Suppose T is a test instance and P_{new} is a new training data. Then $T \cap P_{\text{new}}$ is mostly likely contained in some previously produced maximum itemset $T \cap P_i$. Therefore, the number of maximum itemsets would change slightly (may become smaller). To a large extent, when the number of data instances increases, the additional cost experienced by DeEPs is proportional to the computation for the intersection operation needed with *new* training data.

The next experiment studies the performance of DeEPs as the number of attributes increases. We randomly selected 90% of the original data sets as the training data and

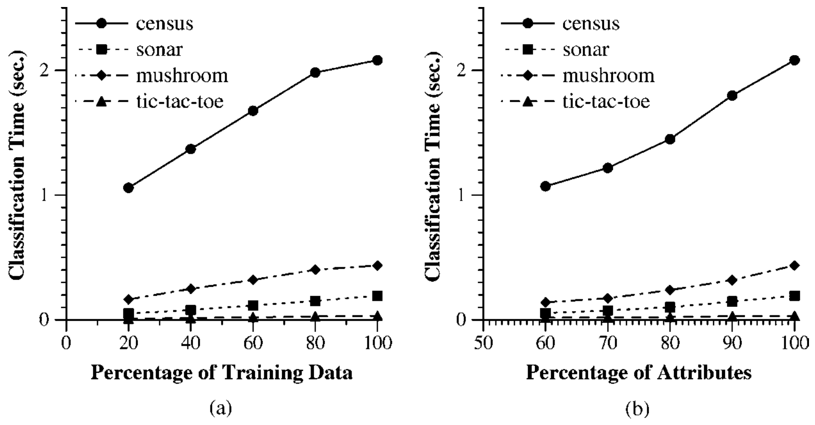

Figure 2. (a) Scalability on numbers of training instances. (b) Scalability on numbers of attributes.

10% as the testing data, both of which have the same class distribution as the original data set. We fixed the number of instances in the training and testing data sets but varied the number of attributes of both training and testing data from 60%, 70%, 80%, 90%, to 100%. Figure 2(b) shows the scalability of DeEPs when the number of attributes increases. These experiments indicate that DeEPs' decision time scales approximately linearly over the number of attributes.

Theoretically, the worst computational complexity of DeEPs is exponential with regard to the number of attributes describing the relation. Fortunately, this does not always occur in real applications as shown in our reported experiments. However, in our recent experiments on gene expression profiles, the running time by DeEPs did not appear to be as expected. This may be caused by complex interactions among genes.

9. Combining the strength of pattern frequency and distance

Both instance-based classifiers, DeEPs and the k-NN classifier (Cover & Hart, 1967), are closely related. Next, we investigate how to combine the strength of pattern frequency and distance to solve classification problems. We refer to this classification method as DeEPsNN (Li, Ramamohanarao, & Dong, 2001).

Suppose an instance T is to be classified. The basic idea of DeEPsNN is (see figure 3):

(a) If a chosen neighborhood of T covers some training instances, 3-NN is applied to classify T. (On special situations where only two or one instance is covered, 1-NN is applied.)
(b) Otherwise, when the neighborhood does not contain any training instances, DeEPs is applied.

9.1. *Selecting a proper neighborhood*

DeEPsNN will need to choose a neighborhood parameter. We first define the notion of a neighborhood.

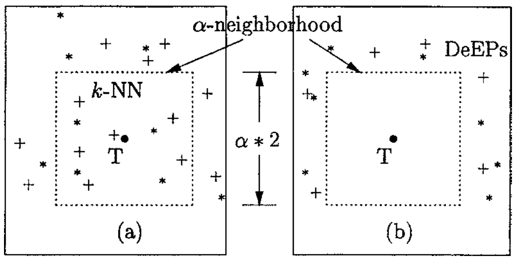

Figure 3. The algorithm deals with two cases when a test instance T is required to classify. The signs "*" and "+" represent instances from two different classes.

Definition 9.1. An α-neighborhood of $T = \{a_1, a_2, \ldots, a_n\}$ consists of the following set of neighbors:

$\{X = \{x_1, x_2, \ldots, x_n\} \mid$ for all $i (1 \leq i \leq n)$, $|x_i - a_i| \leq \alpha$ if a_i is continuous, and $x_i = a_i$ if a_i is categorical$\}$.

To select a proper neighborhood, three initial values for α: 0.05, 0.10, and 0.20 are set. For each such α, a random 10% of the training data are chosen and they are viewed as testing" instances; an accuracy, by using steps (a) and (b) above to classify this special collection of testing" instances, is then obtained. Then, the value of α, by which the highest accuracy is obtained, is applied to the real test instances. Observe that this heuristic uses part of the training data to tune the parameters of DeEPsNN.

9.2. *Performance of DeEPsNN*

Table 7 provides the experimental results on 30 data sets (not including those pure categorical attributes data sets as they are independent of the neighborhood). In this table, columns 4, 5, and 6 show the test accuracies achieved by k-NN, C5.0, and DeEPsNN respectively. Along the vertical direction, Table 7 is organized into three groups according to the performance differences between DeEPsNN and C5.0: significant differences ($\geq$2.00%) are in the top and bottom groups, while small differences ($<$2.00%) in the middle.

It can be seen that the accuracy of DeEPsNN is much higher than 3-NN and also significantly superior to (on average over 30 data sets, 2.18% higher than) C5.0. We found that partial training data guidance can play a key role in selecting a suitable neighborhood for a test instance, and hence can determine when the system should use k-NN or DeEPs for prediction.

10. Discussion

Since the introduction of the notion of emerging patterns in 1998 (Dong & Li, 1999), two eager-learning EP-based classifiers, CAEP (Dong et al., 1999) and JEP-C (Li, Dong, & Ramamohanarao, 2001) have been proposed. In the learning phase, both systems discover all emerging patterns for some pre-defined frequency threshold or frequency-change rate threshold. In the classifying phase, they both aggregate the frequencies of the discovered patterns to construct classification bias.

By adopting eager learning approaches, CAEP and JEP-C produced large numbers of EPs which may never be used for classification. The cost of mining such large number of surplus EPs was often so expensive that they were not able to complete the training phase for large and high- dimensional data sets such as letter and satimage.

Meanwhile, CAEP and JEP-C missed those EPs whose frequency is lower than the threshold of the pre-mined EPs. So, a test instance may not contain any of the pre-mined EPs. The current DeEPs system has considerable advantages in accuracy, overall speed, and dimensional scalability over CAEP and JEP-C, due to its efficient new ways of selecting sharp and relevant EPs, its new ways of aggregating the discriminating power

Table 7. Accuracy comparison among DeEPsNN, C5.0, and k-NN.

	Numbers of attributes		Accuracy (%)			Difference
Data sets	cont.	categ.	3-NN	C5.0	DeEPsNN	DeEPsNN vs. C5.0
Australian	6	8	66.69	85.94	88.41	+2.47
Cleve	5	8	62.64	77.16	83.18	+6.02
crx	6	9	66.64	83.91	86.37	+2.46
German	7	13	63.1	71.3	74.40	+3.10
Heart	6	7	64.07	77.06	81.11	+4.05
Hepatitis	6	13	70.29	74.70	82.56	+7.86
Iris	4	0	96.00	94.00	96.00	+2.00
Letter-recog.	16	0	95.58	88.06	95.51	+7.45
Labor-neg	8	8	93.00	83.99	91.67	+7.68
Lym	3	15	74.79	74.86	84.10	+9.24
Pendigits	16	0	99.35	96.67	98.81	+2.14
Satimage	36	0	91.11	86.74	90.82	+4.08
Sonar	60	0	82.69	70.20	85.13	+14.93
Soybean-s	35	0	100.0	98.00	100.0	+2.00
Waveform	21	0	80.86	76.5	83.78	+7.28
Wine	13	0	72.94	93.35	95.55	+2.20
Anneal	6	32	89.70	93.59	95.11	+1.52
Breast-w	10	0	96.85	95.43	96.28	+0.85
Diabetes	8	0	69.14	73.03	73.17	+0.14
Horse-colic	7	15	66.31	84.81	85.05	+0.24
Hypothyroid	7	18	98.26	99.32	98.17	−1.15
Ionosphere	34	0	83.96	91.92	91.08	−0.84
Pima	8	0	69.14	73.03	73.17	+0.14
Segment	19	0	95.58	97.28	96.62	−0.66
Shuttle-s	9	0	99.54	99.65	99.74	+0.09
Yeast	8	0	54.39	56.14	54.62	−1.52
Auto	15	10	40.86	83.18	74.04	−9.14
Glass	9	0	67.70	70.01	67.98	−2.03
Sick	7	22	93.00	98.78	96.55	−2.23
Vehicle	18	0	65.25	73.68	68.71	−4.97
Average			78.98	84.07	86.25	+2.18

of individual EPs, and most importantly its use of an instance-based approach which creates a remarkable reduction in both the volume and the dimension of the training data. All of these factors contribute to the high accuracy of DeEPs: Among 13 data sets where we conducted tests on both DeEPs and CAEP, DeEPs outperformed CAEP in 8;

among 27 data sets where we conducted tests on both DeEPs and JEP-C, DeEPs outperformed JEP-C in 14. The newly proposed neighborhood-based intersection of continuous attributes and the method of compact summation also contribute to the high accuracy of DeEPs.

As a future work, we are interested in efficiently mining other type of EPs contained in a test instance T. One possibility is to discover those subsets of T which occur in both $\mathcal{D}_p$ and $\mathcal{D}_n$, denoted $commonT = [\{\emptyset\}, max\_\mathcal{R}_p] \cap [\{\emptyset\}, max\_\mathcal{R}_n]$. Then select some of those subsets whose frequency changes *significantly* from $\mathcal{D}_p$ to $\mathcal{D}_n$ or from $\mathcal{D}_n$ to $\mathcal{D}_p$.

Our discussion in Section 7 shows that some EPs with finite frequency-change rate may be more useful than those with infinite rate. Therefore, by adding those EPs with large frequency growth rate into DeEPs, its accuracy could be further improved.

For discretizing continuous attributes, we have introduced a new method called *neighborhood-based intersection*. It allows DeEPs to flexibly determine which continuous attribute values are relevant to a considered instance, without the need to pre-discretize data.

For eager learning based classifiers such as CBA (Liu, Hsu, & Ma, 1998), LB (Meretakis & Wuthrich, 1999), CAEP (Dong et al., 1999), and JEP-C (Li, Dong, & Ramamohanarao, 2001), they must discretize continuous attributes before any algorithms are applied to induce useful patterns or rules. Currently there exist several widely-used methods to discretize continuous attributes (Holte, 1993; Fayyad & Irani, 1993; Kohavi et al., 1994; Quinlan, 1996). The problem of discretizing continuous attributes is still being extensively investigated in machine learning and data mining. We need to emphasize that pre-discretization algorithms can apparently be used in DeEPs.

11. Conclusion

Instead of looking for raw" training instances by k-NN, we have introduced in this paper a new knowledge discovery and classification system which targets regular patterns with large frequency-change rates among training data. The proposed DeEPs system also takes an instance-based learning strategy, efficiently discovering the most discriminating knowledge patterns contained in every considered instance. The discovered patterns are explicit, representative, expressive, and easily comprehensible. They, especially those top-ranked patterns, can be used to help users understand and analyse an instance. They have been demonstrated to be useful in classification: their aggregated frequencies play a crucial role in the high accuracy of the DeEPs system.

We have described two border-based algorithms, JEPPRODUCER and BORDER-DIFF, for discovering the boundary EPs. Together with other data reduction methods, JEPPRODUCER and BORDER-DIFF help the DeEPs system run efficiently as they avoid the naive enumeration of large collections of itemsets.

We have proposed a neighborhood-based intersection method to cope with continuous attributes. We have proposed compact summation to effectively aggregate the frequency of the EPs. Our experimental results on 40 data sets have shown that the accuracy achieved by DeEPs is comparable to and often better than k-nearest neighbor and C5.0. We also compared DeEPs with other classifiers such as CBA, LB, NB and TAN. The results have confirmed the effectiveness of our novel ideas regarding the pattern selection criteria and the compact summation.

Our experimental results have also shown that DeEPs is scalable over the number of training instances and nearly scalable over the number of attributes in large data sets. However, the DeEPs classifier is still slow especially compared with k-NN and C5.0. Our future research will be focused on issues relating to its speed improvement.

Acknowledgment

We thank the three anonymous reviewers for their critical and constructive comments about this paper. We also thank Raul Valdes-Perez, the associate editor, for his kind help in improving the quality of our paper. The first author would like to thank the University of Melbourne for providing a Melbourne Research Scholarship over the author's PhD candidature as most of the current work was done during that period.

Notes

1. DeEPs is short for **De**cision-making by **E**merging **P**attern**s** (Li, Dong, & Ramamohanarao, 2000).
2. The original definition (Dong & Li, 1999) includes patterns with finite growth rates.
3. The attribute names of an instance are always omitted if no confusion is caused.
4. Note that the set X is **maximal** in collection $\mathcal{S}$ if there are no proper supersets of X in $\mathcal{S}$.
5. A collection of itemsets is called an anti-chain if any two elements X and Y of this collection satisfy $X \not\subseteq Y$ and $Y \not\subseteq X$.
6. All the data sets used in the following text are obtained from Blake and Murphy (1998), we do not cite it again.
7. A sub-context can be described as a small cluster of a class which is contained and overwhelmed by the main cluster of another class. In a decision tree, a leaf node which contains mixed instances can be referred to as a main cluster containing a different sub-context.

References

Aha, D. W. (1997). *Lazy Learning*. Dordrecht, Netherlands: Kluwer Academic Publishers.

Aha, D. W., Kibler, D., & Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning, 6*, 37–66.

Bayardo, R. J., & Agrawal, R. (1999). Mining the most interesting rules. In S. Chaudhuri, & D. Madigan (Eds.), *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 145–154). San Diego, CA: ACM Press.

Blake, C., & Murphy, P. (1998). The UCI machine learning repository. [http://www.cs.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.

Cover, T. M., & Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory, 13*, 21–27.

Dasarathy, B. (1991). *Nearest Neighbor Pattern Classification Techniques*. Los Alamitos, CA: IEEE Computer Society Press.

Datta, P., & Kibler, D. (1995). Learning prototypical concept description. In A. Prieditis, & S. J. Russell (Eds.), *Machine Learning: Proceedings of the Twelfth International Conference* (pp. 158–166). San Francisco, CA: Morgan Kaufmann Publishers.

Datta, P., & Kibler, D. (1997). Symbolic nearest mean classifier. In D. H. Fisher (Ed.), *Machine Learning: Proceedings of the Fourteenth International Conference* (pp. 75–82). San Francisco, CA. Morgan Kaufmann.

Devroye, L., Gyorfi, L., & Lugosi, G. (1996). *A Probabilistic Theory of Pattern Recognition*. New York: Springer-Verlag.

Domingos, P. (1996). Unifying instance-based and rule-based induction. *Machine Learning, 24:2*, 141–168.

Dong, G., & Li, J. (1998). Interestingness of discovered association rules in terms of neighborhood-based unexpectedness. In X. Wu, K. Ramamohanarao, & K. B. Korb (Eds.), *Proceedings of the Second Pacific-Asia Conference on Knowledge Discovery and Data Mining, Melbourne* (pp. 72–86). Springer-Verlag.

Dong, G., & Li, J. (1999). Efficient mining of emerging patterns: Discovering trends and differences. In S. Chaudhuri, & D. Madigan (Eds.), *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 43–52). San Diego, CA: ACM Press.

Dong, G., Zhang, X., Wong, L., & Li, J. (1999). CAEP: Classification by aggregating emerging patterns. In S. Arikawa, & K. Furukawa (Eds.), *Proceedings of the Second International Conference on Discovery Science, Tokyo, Japan* (pp. 30–42). Springer-Verlag.

Duda, R., & Hart, P. (1973). *Pattern Classification and Scene Analysis*. New York: John Wiley & Sons.

Fayyad, U., & Irani, K. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. In R. Bajcsy (Ed.), *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence* (pp. 1022–1029). Morgan Kaufmann.

Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning, 29*, 131–163.

Hilderman, R. J., & Hamilton, H. J. (2001). Evaluation of interestingness measures for ranking discovered knowledge. In D. W.-L. Cheung, G. J. Williams, & Q. Li (Eds.), *Proceedings of the Fifth Pacific-Asia Conference on Knowledge Discovery and Data Mining* (pp. 247–259). Hong Kong, China: Springer-Verlag.

Hirsh, H. (1994). Generalizing version spaces. *Machine Learning, 17*, 5–46.

Holte, R. C. (1993). Very simple classification rules perform well on most commonly used data sets. *Machine Learning, 11*, 63–90.

Keung, C.-K., & Lam, W. (2000). Prototype generation based on instance filtering and averaging. In T. Terano, H. Liu, & A. L. P. Chen (Eds.), *Proceedings of the Fourth Pacific-Asia Conference on Knowledge Discovery and Data Mining* (pp. 142–152). Berlin Heidelberg: Springer-Verlag.

Klemettinen, M., Mannila, H., Ronkainen, P., Toivonen, H., & Verkamo, A. (1994). Finding interesting rules from large sets of discovered association rules. In *Proceedings of the 3rd International Conference on Information and Knowledge Management* (pp. 401–408). Gaithersburg, Maryland: ACM Press.

Kohavi, R., John, G., Long, R., Manley, D., & Pfleger, K. (1994). MLC++: A machine learning library in C++. In *Tools with artificial intelligence* (pp. 740–743).

Kubat, M., & Cooperson, M. (2000). Voting nearest-neighbor subclassifiers. In *Machine Learning: Proceedings of the Seventeenth International Conference* (pp. 503–510). Morgan Kaufmann.

Langley, P., Iba, W., & Thompson, K. (1992). An analysis of Bayesian classifier. In W. R. Swartout (Ed.), *Proceedings of the Tenth National Conference on Artificial Intelligence* (pp. 223–228). AAAI Press.

Langley, P., & Iba, W. (1993). Average-case analysis of a nearest neighbor algorithm. In R. Bajcsy (Ed.), *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence* (pp. 889 –894). Chambery, France.

Li, J. (2001). Mining emerging patterns to construct accurate and efficient classifiers. Ph.D. Thesis, Department of Computer Science and Software Engineering, The University of Melbourne, Australia.

Li, J., Dong, G., & Ramamohanarao, K. (2000). Instance-based classification by emerging patterns. In D. A. Zighed, H. J. Komorowski, & J. M. Zytkow (Eds.), *Proceedings of the Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases* (pp. 191–200). Lyon, France: Springer-Verlag.

Li, J., Dong, G., & Ramamohanarao, K. (2001). Making use of the most expressive jumping emerging patterns for classification. *Knowledge and Information Systems: An International Journal, 3*, 131–145.

Li, J., Ramamohanarao, K., & Dong, G. (2000). The space of jumping emerging patterns and its incremental maintenance algorithms. In *Proceedings of the Seventeenth International Conference on Machine Learning, Stanford, CA, USA* (pp. 551–558). San Francisco: Morgan Kaufmann.

Li, J., Ramamohanarao, K., & Dong, G. (2001). Combining the strength of pattern frequency and distance for classification. In D. W.-L. Cheung, G. J. Williams, & Q. Li (Eds.), *The Fifth Pacific-Asia Conference on Knowledge Discovery and Data Mining* (pp. 455–466). Hong Kong.

Li, J., & Wong, L. (2002). Identifying good diagnostic gene groups from gene expression profiles using the concept of emerging patterns. *Bioinformatics, 18:5*, 725–734.

Liu, B., Hsu, W., & Ma, Y. (1998). Integrating classification and association rule mining. In R. Agrawal, P. E. Stolorz, & G. Piatetsky-Shapiro (Eds.), *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining* (pp. 80–86). New York, USA: AAAI Press.

Meretakis, D., & Wuthrich, B. (1999). Extending naive bayes classifiers using long itemsets. In S. Chaudhuri, & D. Madigan (Eds.), *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 165–174). San Diego, CA: ACM Press.

Mitchell, T. (1977). Version spaces: A candidate elimination approach to rule learning. In R. Reddy (Ed.), *Proceedings of the Fifth International Joint Conference on Artificial Intelligence* (pp. 305–310). Cambridge, MA.

Mitchell, T. (1982). Generalization as search. *Artificial Intelligence, 18*, 203–226.

Padmanabhan, B., & Tuzhilin, A. (1998). A belief-driven method for discovering unexpected patterns. In R. Agrawal, P. E. Stolorz, & G. Piatetsky-Shapiro (Eds.), *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining* (pp. 94–100). New York, NY: AAAI Press.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning, 1*, 81–106.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann.

Quinlan, J. R. (1996). Improved use of continuous attributes in C4.5. *Journal of Artificial Intelligence Research, 4*, 77–90.

Sahar, S. (1999). Interestingness via what is not interesting. In S. Chaudhuri, & D. Madigan (Eds.), *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 332–336). San Diego, CA: ACM Press.

Salzberg, S. (1991). A nearest hyperrectangle learning method. *Machine Learning, 6*, 251–276.

Sebag, M. (1996). Delaying the choice of bias: A disjunctive version space approach. In *Machine Learning: Proceedings of the Thirteenth International Conference* (pp. 444–452). Morgan Kaufmann.

Silberschatz, A., & Tuzhilin, A. (1996). What makes patterns interesting in knowledge discovery systems. *IEEE Transactions on Knowledge and Data Engineering, 8:6*, 970–974.

Wettscherech, D. (1994). A hybrid nearest-neighbor and nearest-hyperrectangle algorithm. In F. Bergadano, & L. D. Raedt (Eds.), *Proceedings of the Seventh European Conference on Machine Learning* (pp. 323–335). Springer-Verlag.

Wettschereck, D., & Dietterich, T. G. (1995). An experimental comparison of the nearest-neighbor and nearest-hyperrectangle algorithms. *Machine Learning, 19:1*, 5–27.

Wilson, D. R., & Martinez, T. R. (1997). Instance pruning techniques. In D. H. Fisher (Ed.), *Machine Learning: Proceedings of the Fourteenth International Conference* (pp. 403–411). Morgan Kaufmann.

Wilson, D., & Martinez, T. R. (2000). Reduction techniques for instance-based learning algorithms. *Machine Learning, 38:3*, 257–286.

Zhang, J. (1992). Selecting typical instances in instance-based learning algorithms. In *Machine Learning: Proceedings of the Ninth International Conference* (pp. 470–479). Morgan Kaufmann.

Received July 23, 2001
Revised March 14, 2002
Accepted May 27, 2002
Final manuscript May 31, 2002