Instance-Based Classification by Emerging Patterns

Jinyan Li[1], Guozhu Dong[2], and Kotagiri Ramamohanarao[1]

[1] Dept. of CSSE, The University of Melbourne, Vic. 3010, Australia.
{jyli, rao}@cs.mu.oz.au
[2] Dept. of CSE, Wright State University, Dayton OH 45435, USA. gdong@cs.wright.edu

Abstract. Emerging patterns (EPs), namely itemsets whose supports change significantly from one class to another, capture discriminating features that sharply contrast instances between the classes. Recently, EP-based classifiers have been proposed, which first mine as many EPs as possible (called eager-learning) from the training data and then aggregate the discriminating power of the mined EPs for classifying new instances. We propose here a new, instance-based classifier using EPs, called DeEPs, to achieve much better accuracy and efficiency than the previously proposed EP-based classifiers. High accuracy is achieved because the instance-based approach enables DeEPs to pinpoint all EPs relevant to a test instance, some of which are missed by the eager-learning approaches. High efficiency is obtained using a series of data reduction and concise data-representation techniques. Experiments show that DeEPs' decision time is linearly scalable over the number of training instances and nearly linearly over the number of attributes. Experiments on 40 datasets also show that DeEPs is superior to other classifiers on accuracy.

1 Introduction

The problem of classification has been studied extensively, using eager-learning approaches or lazy instance-based approaches, in machine learning, pattern recognition, and recently also in the data mining community. In this paper we introduce *DeEPs*, a new instance-based classifier which makes *De*cisions through *E*merging *P*atterns. The notion of emerging patterns (EPs) was proposed in [5] and is defined as multivariate features (i.e., *itemsets*) whose *supports* (or frequencies) change significantly from one class to another. Because of sharp changes in support, EPs have strong discriminating power. Two eager-learning classifiers based on the concept of EPs, CAEP [6] and the JEP-Classifier [9], have been proposed and developed, using the novel idea of aggregating the discriminating power of pre-mined EPs for classification. The newly proposed DeEPs classifier has considerable advantages on accuracy, speed, and dimensional scalability over CAEP and the JEP-Classifier, because of its efficient new ways to select sharp and relevant EPs, its new ways to aggregate the discriminating power of individual EPs, and most importantly the use of instance-based approach which creates a remarkable reduction on both the volume (i.e., the number of instances) and the dimension (i.e., the number of attributes) of the training data. Another advantage is that DeEPs can handle new training data without the need to re-train the classifier which is, however, commonly required by the eager-learning based classifiers. This feature is extremely useful for practical applications where the training data must be frequently updated. DeEPs

D.A. Zighed, J. Komorowski, and J. Zytkow (Eds.): PKDD 2000, LNAI 1910, pp. 191–200, 2000.

can handle both numeric and discrete attributes and DeEPs is nicely scalable over the number of training instances.

Given two classes of data $\mathcal{D}_1$ and $\mathcal{D}_2$ and a test instance T, the basic idea of DeEPs is to discover those subsets of T which are emerging patterns between $\mathcal{D}_1$ and $\mathcal{D}_2$, and then use the supports of the discovered EPs for prediction. The test instance T may not contain any pre-mined EPs if the eager-learning approaches are used; in contrast, using the instance-based approach, DeEPs will be able to efficiently pinpoint all relevant EPs for classifying T. The basic idea of DeEPs is made practical and scalable to high dimension data because we use a series of data reduction and concise data represensation techniques. We use the following example to illustrate the ideas behind DeEPs.

Example 1. Table 1, taken from [14], contains a training set, for predicting whether 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 $\mathcal{N}$.

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				

Now, given the test instance *T*={*sunny, mild, high, true*}, which class label should it take? Basically, DeEPs calculates the supports (in both classes) of the proper subsets of T in its first step. The proper subsets of T and their supports are organized as the following three groups:

1. those that only occur in Class $\mathcal{N}$ but not in Class $\mathcal{P}$, namely, {sunny, high}, {sunny, mild, high}, and {sunny, high, true}; their supports in Class $\mathcal{N}$ are 60%, 20%, and 20% respectively.
2. those that only occur in Class $\mathcal{P}$ but not in Class $\mathcal{N}$, namely, {sunny, mild, true}; its support in Class $\mathcal{P}$ is 11%.
3. those that occur in both classes, namely, $\emptyset$, {mild}, {sunny}, {high}, {true}, {sunny, mild}, {mild, high}, {sunny, true}, {high, true}, {mild, true}, and {mild, high, true}. Except for the patterns $\emptyset$ and {mild}, all these subsets have larger supports in Class $\mathcal{N}$ than in Class $\mathcal{P}$.

Obviously, the first group of subsets — which are indeed EPs of Class $\mathcal{N}$ because they do not appear in Class $\mathcal{P}$ at all — favors the prediction that T should be classified as Class $\mathcal{N}$. However, the second group of subsets gives us a contrasting indication that T should be classified as Class $\mathcal{P}$, although this indication is not as strong as that of the first group. The third group also strongly suggests that we should favor Class $\mathcal{N}$

as T's label, although the pattern $\{\text{mild}\}$ contradicts this mildly. Using these EPs in a *collective* way, not separately, DeEPs would decide 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 $\mathcal{P}$.

In practice, an instance may contain many (e.g., 100 or more) attributes. To examine all subsets and discover the relevant EPs contained in such instances by naive enumeration is too expensive (e.g., checking 2^{100} or more sets). We make DeEPs efficient and scalable to high dimensional data by the following data reduction and concise data representation techniques.

- We reduce the training datasets firstly by removing those items that do not occur in the test instance and then by selecting the *maximal* ones from the processed training instances. (Set X is *maximal* in collection $\mathcal{S}$ if there are no proper supersets of X in $\mathcal{S}$.) This data reduction process makes the training data sparser in both horizontal and vertical directions.
- We use *border* [5], a two-bound structure like $<\mathcal{L}, \mathcal{R}>$, to succinctly represent all EPs contained in a test instance. Importantly, we use efficient border-based algorithms to derive EP borders from the reduced training datasets.
- We select *boundary* EPs, those in $\mathcal{L}$ (typically small in number, e.g., 81 in mushroom), for DeEPs' decision making. These selected EPs are "good" representatives of all EPs occurring in a test instance. This selection method also significantly reduces the number of EPs that are used for classification.

Detailed discussions of these points will be given in the next three sections. Table 2 illustrates the first stage of the sparsifying effect on both the volume and dimension of $\mathcal{D}_{\mathcal{P}}$ and $\mathcal{D}_{\mathcal{N}}$, by removing all items 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 every attribute of any instance. Sections 3 and 4 will discuss formally how to select the maximal ones of the reduced training instances and how to utilize the reduced training data to gain more efficiency with the use of borders.

Table 2. Reduced training data after removing items irrelevant to the instance $\{$sunny, mild, high, true$\}$. A "*" indicates that an item is discarded.

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				
*	*	*	*				

Since EPs usually have very low supports, they are not suitable to be used individually for classification. We will use the *compact summation* method to aggregate the discriminating power contributed by all selected EPs to form classification scores.

Importantly, the *compact summation* method avoids duplicate contribution of training instances.

For continuous attributes, we introduce a new method, called *neighborhood-based intersection*. This allows DeEPs to determine which continuous attribute values are relevant to a given test instance, without the need to pre-discretize data.

The remainder of this paper is organized as follows. Section 2 reviews the notions of EPs and borders. Section 3 formally introduces the DeEPs classifier. Section 4 presents border-based algorithms to implement the main ideas of DeEPs. Section 5 presents the experimental results of DeEPs on 40 datasets, taken from the UCI Repository on Machine Learning [2]. The experiments demonstrate the scalability and high accuracy of DeEPs. Section 6 compares our DeEPs classifier with other classification models. Section 7 concludes this paper.

2 Preliminaries: EPs and Borders

In our discussion, the basic elements are the *items*. Relational data in the form of vectors (or tuples) are first translated as follows. For each relational attribute A with a continuous domain and a given interval $[l, h)$, "$A \in [l, h)$" is an item. If A is a discrete attribute and a is in the domain of A, then "$A = a$" is an item. Vectors or tuples are now represented as sets of items. An *instance* is a set of items, and a *dataset* $\mathcal{D}$ is a set of instances. A set X of items is also called an *itemset*. We say an instance S *contains* an itemset X, if $X \subseteq S$. The *support* of an itemset X in a dataset $\mathcal{D}$, $supp_{\mathcal{D}}(X)$, is $\frac{count_{\mathcal{D}}(X)}{|\mathcal{D}|}$, where $count_{\mathcal{D}}(X)$ is the number of instances in $\mathcal{D}$ containing X.

The notion of *emerging patterns* (EPs) [5] and a special type, *jumping emerging patterns* (JEPs), were proposed to capture differences between classes.

Definition 1. *[5] (***EP** *and* **JEP***) Given a real number $\rho > 1$ and two datasets $\mathcal{D}_1$ and $\mathcal{D}_2$, an itemset X is called an* ρ-emerging pattern *(EP) from $\mathcal{D}_1$ to $\mathcal{D}_2$ if the support ratio $\frac{supp_{\mathcal{D}_2}(X)}{supp_{\mathcal{D}_1}(X)} \geq \rho$. (Define $\frac{0}{0} = 0$ and $\frac{\neq 0}{0} = \infty$.) Specially, a* jumping *EP (JEP) of $\mathcal{D}_2$ is such an EP which occurs in $\mathcal{D}_2$ (or, whose support in $\mathcal{D}_2$ is* non-zero*) but does not occur in $\mathcal{D}_1$ (or, whose support in $\mathcal{D}_1$ is zero).*

We have already seen three JEPs of $\mathcal{D}_{\mathcal{N}}$ and one JEP of $\mathcal{D}_{\mathcal{P}}$ in Example 1.

Informally, a border is a concise structure used to describe a large collection of sets. For example, the border $<\{\{1\}\}, \{\{1,2\}, \{1,2,3,4,5,6\}\}>$ is bounded by its *left bound* $\{\{1\}\}$ and its *right bound* $\{\{1,2\}, \{1,2,3,4,5,6\}\}$. This border represents the collection of all sets which are supersets of $\{\{1\}\}$ and are subsets of either $\{1,2\}$ or $\{1,2,3,4,5,6\}$. This collection of sets is denoted $[\{\{1\}\}, \{\{1,2\}, \{1,2,3,4,5,6\}\}]$. More details about the definitions of borders can be found in [5].

3 The DeEPs Classifier

Our DeEPs classifier needs three main steps to determine the class of a test instance: (i) Discovering border representations of EPs; (ii) Selecting the more discriminating EPs; (iii) Determining collective scores based on the selected EPs for classification.

We present algorithms for each of the three steps in the subsequent subsections.We begin by presenting algorithms to handle datasets with only two classes and then generalize DeEPs in Section 3.4 to handle datasets with more classes.

3.1 Discovering Border Representations of EPs

This step aims to learn discriminating knowledge from training data and represent them concisely, by first reducing the data and then discovering all JEPs (and optionally other EPs). Assume we are given a classification problem, having a set $\mathcal{D}_p = \{P_1, \cdots, P_m\}$ of positive training instances, a set $\mathcal{D}_n = \{N_1, \cdots, N_n\}$ of negative training instances, and a set of test instances.

For each test instance T, the DeEPs classifier uses the three procedures below to discover border representations of the EPs from the training data.

1. **Intersecting the training data with T: $T \cap P_1$, $\cdots$, $T \cap P_m$ and $T \cap N_1$, $\cdots$, $T \cap N_n$.** We will discuss how to conduct intersection operation, using *neighborhood-based intersection* method, when continuous attributes are present.
2. **Selecting the maximal itemsets from $T \cap P_1$, $\cdots$, $T \cap P_m$, and similarly from $T \cap N_1$, $\cdots$, $T \cap N_n$.** Denote the former collection of maximal itemsets as $\mathcal{R}_p$ and the latter as $\mathcal{R}_n$.

3.a **Discovery of jumping emerging patterns**. Mining those subsets of T which occur in $\mathcal{D}_p$ but not in $\mathcal{D}_n$, i.e., *all* the JEPs in $\mathcal{D}_p$ (positive class), by taking *border difference* operation $[\{\emptyset\}, \mathcal{R}_p] - [\{\emptyset\}, \mathcal{R}_n]$. On the other hand, mining those subsets of T which occur in $\mathcal{D}_n$ but not in $\mathcal{D}_p$, i.e., *all* the JEPs in $\mathcal{D}_n$ (negative class), by similarly taking *border difference* operation $[\{\emptyset\}, \mathcal{R}_n] - [\{\emptyset\}, \mathcal{R}_p]$.

3.b **Discovery of common emerging patterns**. Mining those subsets of T which occur in both $\mathcal{D}_p$ and $\mathcal{D}_n$, namely, $commonT = [\{\emptyset\}, \mathcal{R}_p] \cap [\{\emptyset\}, \mathcal{R}_n]$, and then selecting those itemsets whose supports change significantly from $\mathcal{D}_p$ to $\mathcal{D}_n$ or from $\mathcal{D}_n$ to $\mathcal{D}_p$. This step is optional, and is omitted if decision speed is important.

Using intersection and maximal itemsets to reduce volume and dimension of training data. First, with the intersection operation in step 1, the dimension of the training data is substantially reduced, because many values of the original training data do not occur in the test instance T. Second, 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, $\mathcal{R}_p$ can be viewed as a compressed $\mathcal{D}_p$, and $\mathcal{R}_n$ a compressed $\mathcal{D}_n$. We use the mushroom dataset to demonstrate this point. 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 these processed instances are further compressed into 7 maximal itemsets. Thus we have achieved a reduction from 3788 to 7 in volume and from 22 to 11 in dimension in the mushroom data. Table 2 of Section 1 also briefly illustrated this 2-directional sparsifying effect. Therefore, this data reduction mechanism narrows our search space. This compression effect is made possible by the instance-based learning approach and it lays a foundation for the high accuracy and high efficiency of DeEPs.

Using border algorithms to efficiently discover EPs. Step 3.a is used to efficiently discover the JEP border representations. Note that $<\{\emptyset\}, \mathcal{R}_p>$ or $<\{\emptyset\}, \mathcal{R}_n>$ can still represent large collections of sets, despite the tremendous effect of reduction as discussed above. To enumerate all itemsets covered by these borders is costly. The *border difference* operation avoids the expensive enumeration, by manipulating the boundary

elements $\mathcal{R}_p$ and $\mathcal{R}_n$, to output $<jump\mathcal{L}_p, jump\mathcal{R}_p>$ or $<jump\mathcal{L}_n, jump\mathcal{R}_n>$ as a succinct border representation of the discriminating features (JEPs) of T in the positive class or in the negative class. The border difference operation itself is reviewed later in Section 4. Similarly, by manipulating the boundary elements in $\mathcal{R}_p$ and $\mathcal{R}_n$, we can represent $commonT$ by the border $<\{\emptyset\}, common\mathcal{R}>$. The above three borders are border representations of the EPs that DeEPs needs.

Neighborhood-based intersection of continuous attributes. For datasets containing continuous attributes, we need a new way to intersect two instances. We introduce a method called *neighborhood-based intersection*. It helps select the most relevant information from the training instances for classifying the test instance T, and avoids pre-discretizing the training instances. Suppose `attri_A` is a continuous attribute and its domain is $[0, 1]$. (We can normalize all `attri_A` values in the training instances to the range of $[0, 1]$ if its domain is not $[0, 1]$.) Given a test instance T and 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 $[a_1 - \alpha\%, a_1 + \alpha\%]$, where a_1 is the normalized `attri_A` value for T. The parameter α is called *neighborhood factor*, which can be used to adjust the length of the neighborhood.

We observed that different neighborhood factors α can cause accuracy variance, although slight, on the test data. When α is too large, it may happen that many originally different instances from different classes can be transformed into an identical binary instance; consequently, the inherent discriminating features among these instances disappear. When α is too small, nearly identical attribute values may be considered different, and thus useful discrimination information in the training data might be missed. Briefly, we select a suitable α for each dataset by using part of training data as a guide; the details are given in [10]. This is also a research topic in our future work.

3.2 Selecting the More Discriminating EPs

We observed that the number of JEPs (or optional EP candidates) that occur in a test instance is usually large (e.g., of the order of 10^6 in mushroom, waveform, ionosphere, and sonar data). It is expensive to aggregate the contributions of all those EPs. To solve this problem, we select more discriminating EPs by taking advantage of border representation of EPs: We select the most *general* JEPs among all JEPs, namely those in $jump\mathcal{L}_p$ and $jump\mathcal{L}_n$, as the necessary EPs, and select itemsets in the right bound of $commonT$, namely $common\mathcal{R}$, as optional EPs. By the most *general* JEPs, we mean that their proper subsets are not JEPs any more. Partial reasons of this selection are given in [9]. Whether the optional EPs are used depends on how much time a user is willing to spend in classifying instances. With less time, we only select those EPs that must be included; with more time, we can consider the optional EPs for inclusion.

3.3 Determining Collective Scores for Classification

We determine the collective score of T for any specific class C by aggregating the supports of the selected EPs in class C, using our *compact summation* method.

Definition 2. *The compact summation of the supports in $\mathcal{D}_C$ of a collection of selected EPs is defined as the percentage of instances in $\mathcal{D}_C$ that contain one or more of the selected EPs; this percentage is called the* compact score *of T for C, that is,*

$compactScore(C) = \frac{count_{\mathcal{D}_C}(SEP)}{|\mathcal{D}_C|}$, *where* SEP *is the collection of selected 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 main purpose of this aggregation is to avoid counting duplicate contribution of training instances.

DeEPs makes the final decision only when the compact scores, formed by compact summation, for all different classes are available. Then the DeEPs classifier will simply assign to T the class for which T's score is largest. We use majority rule to break ties.

3.4 Handling Datasets Containing More than Two Classes

We have already discussed how DeEPs is applied to the problems with two classes. In the following, the DeEPs classifier is generalized to handle more classes. For example, given a test instance T and a training database containing 3 classes of data $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$, we only need to discover the border representation of the JEPs (and optionally EPs) from $(\mathcal{D}_2 \cup \mathcal{D}_3)$ to $\mathcal{D}_1$, those from $(\mathcal{D}_1 \cup \mathcal{D}_3)$ to $\mathcal{D}_2$, and those from $(\mathcal{D}_1 \cup \mathcal{D}_2)$ to $\mathcal{D}_3$. The compact scores for these three classes can be calculated based on three groups of the boundary EPs. We choose the class in which the biggest compact score is obtained as T's class label.

4 Algorithms for DeEPs

We need three algorithms, maxSelector, intersecOperation, and diffOperation [11], for DeEPs to find the desired EPs. maxSelector is used to select the maximal ones from a collection of sets. intersecOperation and diffOperation are used to conduct *border intersection* and *border difference* respectively; they are needed to discover border representations of $commonT$ and of JEPs. As the latter two algorithms only manipulate the bounds of borders to handle huge collections, they are highly efficient and scalable in practice.

The diffOperation algorithm [11] is essential to the efficiency of DeEPs. It is used to discover the border representation of the JEPs of T in each class. Given two borders $<\{\emptyset\}, \mathcal{R}_1>$ and $<\{\emptyset\}, \mathcal{R}_2>$, the *border difference* operation, implemented by diffOperation, is used to derive the border of $[\{\emptyset\}, \mathcal{R}_1] - [\{\emptyset\}, \mathcal{R}_2]$.

More details about these three algorithms can be found in [11,5] and in our technical report [10].

5 Performance Evaluation: Accuracy, Speed, and Scalability

We now present the experimental results to demonstrate the accuracy, speed, and scalability of DeEPs. We have run DeEPs on 40 datasets taken from the UCI Machine Learning Repository [2]. The accuracy results were obtained using the methodology of ten-fold cross-validation (CV-10).These experiments were carried out on a 500MHz PentiumIII PC, with 512M bytes of RAM.

5.1 Pre-processes

The experiment's pre-processes include: (i) download original datasets, say $\mathcal{D}$, from the sources; (ii) partition $\mathcal{D}$ into class datasets $\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_q$, where q is the number of classes in $\mathcal{D}$; (iii) randomly shuffle each $\mathcal{D}_i, i = 1, \cdots, q$, using the function

`random_shuffle()` in Standard Template Library [4]; (iv) for each $\mathcal{D}_i$, do CV-10 partition; (v) if there exist continuous attributes, scale all the values in the training datasets of each attribute into the values in the range of $[0, 1]$, and then use the same parameters to scale the values in the testing datasets. This step is used to prepare for *neighborhood-based intersection of continuous attributes* and to prepare for the conversion of training datasets into binary ones. In this paper, we use the formula $\frac{x-min}{max-min}$ to scale every value x of the attribute `attri_A`, where max and min are respectively the biggest and the smallest values of `attri_A` in the *training* data.

5.2 Accuracy, Speed, and Scalability

We compare DeEPs with five other state-of-the-art classifiers: C4.5 [15], Naive Bayes (NB) [7], TAN [8], CBA [12], and LB [13].

Table 3reports results of the experiments. Column 1 lists the name of the datasets; column 2 lists the numbers of instances, attributes, and classes; columns 3 and 4 present the average accuracy of DeEPs, when the neighborhood factor α is fixed as 12 for all datasets, and respectively when α is dynamically selected within each dataset (as explained in [10]). Note that for the datasets such as chess, flare, nursery, splice, mushroom, voting, soybean-l, t-t-t, and zoo which do not contain any continuous attributes, DeEPs does not need α. Columns 5, 6, 7, 8, and 9 give the accuracies of CBA, C4.5, LB, NB, and TAN respectively; these results are exactly copied from Table 1 in [13] for the first 21 datasets. For the remaining datasets, we select for CBA and C4.5 the *best* result from Table 1 of [12]. A dash indicates that we were not able to find previous reported results and "N/A" means the classifier is not applicable to the datasets. The last column gives the average time used by DeEPs to test one instance.

The results for DeEPs were obtained by selecting only the left bounds of JEPs, without selecting any optional EPs. Our experiments also show that selecting optional EPs can increase accuracy but degrade speed.

We now discuss the case when the neighborhood factor is fixed as 12 for all datasets. We highlight some interesting points as follows:

- Among the first 21 datasets where results of the other five classifiers are available, DeEPs achieves the best accuracy (the numbers in bold font) on 9 datasets. C4.5, LB, NB, and TAN achieve the best accuracy on 3, 5, 4 and 3 datasets respectively. It can be seen that DeEPs in general outperforms the other classifiers.
- For the remaining 16 datasets where results of CBA and C4.5 are available, DeEPs achieves the best accuracy on 7 datasets. CBA and C4.5 achieve the best accuracy on 6 and 3 datasets respectively. In addition, DeEPs can reach 100% accuracy on mushroom, 99.04% on nursery, and 98.21% on pendigits.

Discussions on speed and scalability of DeEPs can be found in our technical report [10] and omitted here for space reasons.

6 Related Work

CAEP [6] and the JEP-Classifier [9] are two relatives to DeEPs. The former two are eager-learning based approaches, but DeEPs is instance-based. For more comparison, the readers are refered to [6,9,10].

Table 3. Accuracy Comparison.

Datasets	#inst, attri, class	DeEPs		CBA	C4.5	LB	NB	TAN	time (sec.)
		$\alpha = 12$	*dynamical* α						
australian	690, 14, 2	84.78	88.41* (5)	85.51	84.28	**85.65**	**85.65**	85.22	0.054
breast-w	699, 10, 2	96.42	96.42 (12)	95.28	95.42	96.86	**97.00**	N/A	0.055
census-inc	**30162**, 16, 2	**85.93**	85.93* (12)	85.67	85.4	85.11	84.12	N/A	2.081
chess	3196, 36, 2	97.81	97.81	98.12	**99.5**	90.24	87.15	92.12	0.472
cleve	303, 13, 2	81.17	84.21* (15)	77.24	72.19	82.19	**82.78**	N/A	0.032
diabete	768, 8, 2	**76.82**	76.82* (12)	72.9	71.73	76.69	75.13	76.56	0.051
flare	1066,10,2	**83.50**	83.50	83.11	81.16	81.52	79.46	82.64	0.028
german	1000, 20, 2	74.40	74.40 (12)	73.2	71.7	**74.8**	74.1	72.7	0.207
heart	270, 13, 2	81.11	82.22 (15)	81.87	76.69	82.22	82.22	**83.33**	0.025
hepatitis	155, 19, 2	81.18	82.52 (11)	80.20	80.00	**84.5**	83.92	N/A	0.018
letter	**20000, 16, 26**	**93.60**	93.60* (12)	51.76	77.7	76.4	74.94	85.7	3.267
lymph	148, 18, 4	75.42	75.42 (10)	77.33	78.39	**84.57**	81.86	83.76	0.019
pima	768, 8, 2	**76.82**	77.08* (14)	73.1	72.5	75.77	75.9	75.77	0.051
satimage	**6435, 36, 6**	**88.47**	88.47* (12)	84.85	85.2	83.9	81.8	87.2	2.821
segment	2310, 19, 7	94.98	95.97* (5)	93.51	**95.8**	94.16	91.82	93.51	0.382
shuttle-small	5800, 9, 7	97.02	99.62* (1)	99.48	99.5	99.38	98.7	**99.64**	0.438
splice	3175, 60, 3	69.71	69.71	70.03	93.3	**94.64**	**94.64**	**94.63**	0.893
vehicle	846, 18, 4	**70.95**	74.56* (15)	68.78	69.82	68.8	61.12	70.92	0.134
voting	433, 16, 2	95.17	95.17	93.54	**95.66**	94.72	90.34	93.32	0.025
waveform	**5000, 21, 3**	**84.36**	84.36* (12)	75.34	70.4	79.43	78.51	79.13	2.522
yeast	1484, 8, 10	**59.78**	60.24* (10)	55.1	55.73	58.16	58.05	57.21	0.096
anneal	998, 38, 6	94.41	95.01 (6)	**98.1**	94.8	–	–	–	0.122
automobile	205, 25, 7	67.65	72.68 (3.5)	79.00	**80.1**	–	–	–	0.045
crx	690, 15, 2	84.18	88.11* (3.5)	**85.9**	84.9	–	–	–	0.055
glass	214, 9, 7	58.49	67.39 (10)	**72.6**	72.5	–	–	–	0.021
horse	368, 28, 2	**84.21**	85.31* (3.5)	82.1	83.7	–	–	–	0.052
hypo	3163, 25, 2	97.19	98.26 (5)	98.4	**99.2**	–	–	–	0.275
ionosphere	351, 34, 2	86.23	91.24 (5)	**92.1**	92.00	–	–	–	0.147
iris	150, 4, 3	**96.00**	96.67* (10)	92.9	95.3	–	–	–	0.007
labor	57, 16, 2	**87.67**	87.67* (10)	83.00	79.3	–	–	–	0.009
mushroom	**8124**, 22, 2	**100.0**	100.0	–	–	–	–	–	0.436
nursery	**12960**, 8, 5	**99.04**	99.04	–	–	–	–	–	0.290
pendigits	**10992, 16, 10**	**98.21**	98.44 (18)	–	–	–	–	–	1.912
sick	4744, 29, 2	94.03	96.63 (5)	97.3	**98.5**	–	–	–	0.284
sonar	208, 60, 2	**84.16**	86.97* (11)	78.3	72.2	–	–	–	0.193
soybean-small	47, 34, 4	**100.0**	100.0* (10)	98.00	98.00	–	–	–	0.022
soybean-large	683, 35, 19	90.08	90.08	**92.23**	92.1	–	–	–	0.072
tic-tac-toe	958, 9, 2	99.06	99.06	**100.0**	99.4	–	–	–	0.032
wine	178, 13, 3	**95.58**	96.08* (11)	91.6	92.7	–	–	–	0.028
zoo	101, 16, 7	**97.19**	97.19	94.6	92.2	–	–	–	0.007

Both being instance-based, DeEPs and the k-nearest-neighbor (k-NNR) classifier [3] are closely related. Their fundamental differences are: k-NNR uses the distances of the k nearest neighbors of a test instance T to determine the class of T; however, DeEPs uses the support change of some selected subsets of T.

7 Conclusions

In this paper, we have proposed a new classifier, DeEPs. The DeEPs classifier uses the instance-based, lazy approach to the mining of discriminating knowledge in the form of EPs. This strategy ensures that good representative EPs that are present in a new instance can be efficiently found and can be effectively used for classifying the new instance. Our experimental results have shown the classification accuracy achieved by DeEPs is very high. Our experimental results also have shown DeEPs' decision is quick and DeEPs is scalable over the number of training instances and nearly scalable over the number of attributes in large datasets.

References

1. D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.
2. C.L. Blake and P.M. Murphy. UCI Repository of machine learning database. [http://www.cs.uci.edu/ mlearn/mlrepository.html]. 1998.
3. T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21–27, 1967.
4. H. M. Deitel and P. J. Deitel. *C++ how to program, second edition.* Prentice Hall, Upper Saddle River, New Jersey, USA, 1998.
5. G. Dong and J. Li. Efficient mining of emerging patterns: Discovering trends and differences. In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining, San Diego, USA* (SIGKDD'99), pages 43–52, 1999.
6. G. Dong, X. Zhang, L. Wong, and J. Li. CAEP: Classification by aggregating emerging patterns. In *Proceedings of the Second International Conference on Discovery Science, Tokyo, Japan*, pages 30-42, 1999.
7. R. Duda and P. Hart. *Pattern Classification and Scene Analysis.* New York: John Wiley & Sons, 1973.
8. N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. In *Machine Learning*, 29: 131–163, 1997.
9. J. Li, G. Dong, and K. Ramamohanarao. Making use of the most expressive jumping emerging patterns for classification. In *Proceedings of the Fourth Pacific-Asia Conference on Knowledge Discovery and Data Mining, Kyoto, Japan*, pages 220–232, 2000.
10. J. Li, G. Dong, and K. Ramamohanarao. DeEPs: Instance-based classification by emerging patterns. Technical Report, Dept of CSSE, University of Melbourne, 2000.
11. J. Li, K. Ramamohanarao, and G. Dong. The space of jumping emerging patterns and its incremental maintenance algorithms. In *Proceedings of the Seventeenth International Conference on Machine Learning, Stanford, CA* June, 2000.
12. B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Proceedings of the Fourth International Conference on Knowledge Discovery in Databases and Data Mining, KDD'98 New York, USA*, 1998.
13. D. Meretakis and B. Wuthrich. Extending naive bayes classifiers using long itemsets. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego*, pages 165–174, 1999.
14. J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.
15. J.R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann, 1993.