5-2001

Making Use of the Most Expressive Jumping Emerging Patterns for Classification

Jinyan Li

Guozhu Dong
Wright State University - Main Campus, guozhu.dong@wright.edu

Kotagiri Ramamohanarao

Making Use of the Most Expressive Jumping Emerging Patterns for Classification

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

[1] Department of CSSE, The University of Melbourne, Parkville, Vic. 3052, Australia.
{jyli, rao}@cs.mu.oz.au
[2] Dept. of CSE, Wright State University, Dayton OH 45435, USA. gdong@cs.wright.edu

Abstract. Classification aims to discover a model from training data that can be used to predict the class of test instances. In this paper, we propose the use of *jumping emerging patterns* (JEPs) as the basis for a new classifier called the *JEP-Classifier*. Each JEP can capture some crucial difference between a pair of datasets. Then, aggregating all JEPs of large supports can produce more potent classification power. Procedurally, the JEP-Classifier learns the *pair-wise features* (sets of JEPs) contained in the training data, and uses the *collective impacts* contributed by the *most expressive* pair-wise features to determine the class labels of the test data. Using only the most expressive JEPs in the JEP-Classifier strengthens its resistance to noise in the training data, and reduces its complexity (as there are usually a very large number of JEPs). We use two algorithms for constructing the JEP-Classifier which are both scalable and efficient. These algorithms make use of the *border representation* to efficiently store and manipulate JEPs. We also present experimental results which show that the JEP-Classifier achieves much higher testing accuracies than the association-based classifier of [8], which was reported to outperform C4.5 in general.

1 Introduction

Classification is an important problem in the fields of data mining and machine learning. In general, classification aims to classify instances in a set of test data, based on knowledge learned from a set of training data. In this paper, we propose a new classifier, called the *JEP-Classifier*, which exploits the discriminating power of *jumping emerging patterns* (JEPs) [4]. A JEP is a special type of EP [3] (also a special type of *discriminant rule* [6]), defined as an itemset whose support increases *abruptly* from zero in one dataset, to non-zero in another dataset — the ratio of support-increase being ∞. The JEP-Classifier uses JEPs exclusively, and is distinct from the CAEP classifier [5] which mainly uses EPs with *finite* support-increase ratios.

The exclusive use of JEPs in the JEP-Classifier is motivated by our belief that JEPs represent knowledge which discriminates between different classes more strongly than any other type of EPs. Consider, for example, the Mushroom dataset taken from the UCI data repository [1]. The itemset $\{$ODOR = foul$\}$ is a JEP, whose support increases from 0% in the edible class to 55% in the poisonous class. If a test instance contains this particular EP, then we can claim with a very high degree of certainty that this instance belongs to the poisonous class, and not to the edible class. In contrast, other

kinds of EPs do not support such strong claims. Experimental results show that the JEP-Classifier indeed gives much higher prediction accuracy than previously published classifiers.

Example 1. This simplified example illustrates how JEPs are used in the JEP-Classifier. Consider two sets of training data, $\mathcal{D}_1$ and $\mathcal{D}_2$, such that all instances in $\mathcal{D}_1$ are of Class 1, and all instances in $\mathcal{D}_2$ are of Class 2. Let each instance be a subset of $\{a, b, c, d, e\}$ (see Table 1). **Question:** Which class should the test instance $\{a, b, c\}$ be classified as?

Table 1. Two simplified datasets containing 4 instances each.

$\mathcal{D}_1$				
a		c	d	e
a				
	b			e
	b	c	d	e

$\mathcal{D}_2$				
a	b			
		c		e
a	b	c	d	
			d	e

Answer: Class 2. **Rationale:** The test instance $\{a, b, c\}$ contains the JEP $\{a, b\}$ from $\mathcal{D}_1$ to $\mathcal{D}_2$, whose support in $\mathcal{D}_2$ is 50%. Furthermore, the remaining proper subsets of $\{a, b, c\}$ — namely, $\{a\}$, $\{b\}$, $\{c\}$, $\{a, c\}$, and $\{b, c\}$ — appear in both classes of data with the same frequencies. These facts give us a higher confidence that the test instance should be classified as Class 2.

In general, a test instance T may contain several JEPs, and these EPs can favour different classes. Consider again the datasets in Table 1, this time with the test instance $T = \{a, b, e\}$. The instance T contains the following JEPs:

- the subsets $\{b, e\}$ and $\{a, e\}$, in favour of Class 1 with supports in $\mathcal{D}_1$ of, respectively, 50% and 25%;
- the subset $\{a, b\}$ in favour of Class 2, with a support in $\mathcal{D}_2$ of 50%.

We let all three JEPs contribute an *impact* equal to its support in its favoured class — the final decision is reached using the *collective impact*, obtained as the sum of the impacts of the individual JEPs, and choosing the class with the largest collective impact as the class of the test instance. It follows that the instance $\{a, b, e\}$ should be classified as Class 1, since the collective impact in favour of Class 1 ($50\% + 25\% = 75\%$) is larger than that of Class 2 (50%). This aggregation of the supports of JEPs is at the core of the JEP-Classifier.

There can be a large (e.g., 10^8) number of JEPs in the dense and high-dimensional datasets of a typical classification problem. Obviously, the naive approach to discovering all JEPs and calculating their collective impacts is too time consuming. For the JEP-Classifier, we utilize two *border*-based algorithms [3, 4] to efficiently discover concise border representations of all JEPs from training dataset. The use of the border representation simplifies the identification of the *most expressive* JEPs. Intuitively, the most expressive JEPs are those JEPs with large support, which can be imagined as being at the "frontier" of the set of JEPs. Itemsets which are proper subsets of the boundary itemsets are not JEPs, while itemsets which are proper supersets of the boundary itemsets must have supports *not larger* than the largest support of the boundary itemsets. These boundary JEPs represent the essence of the discriminating knowledge in the training

dataset. The use of the most expressive JEPs strengthens the JEP-Classifier's resistance to noise in the training data, and can greatly reduce its overall complexity. Borders are formally defined in Section 3.

Example 1 above deals with a simple database containing only two classes of data. To handle the general cases where the database contains more classes, we introduce the concept of *pair-wise features*, which describes a collection of the discriminating knowledge of ordered pairs of classes of data. Using the same idea for dealing with two classes of data, the JEP-Classifier uses the collective impact contributed by the most expressive pair-wise features to predict the labels of more than two classes of data.

Our experimental results (detailed in Section 5) show that the JEP-Classifier can achieve much higher testing accuracy than previously published classifiers, such as the classifier proposed in [8], which generally outperforms C4.5, and the classifier in [5]. In summary, the JEP-Classifier has superior performance because:

1. Each individual JEP has sharp discriminating power, and
2. Identifying the most expressive JEPs and aggregating their discriminating power leads to very strong classifying ability.

Note that the JEP-Classifier can reach a 100% accuracy on any training data. However, unlike many classifiers, this does not lead to the usual overfitting problems, as JEPs can only occur when they are supported in the training dataset.

The remainder of this paper is organised as follows. In Section 2, we present an overall description of the JEP-Classifier (the learning phase and the classification procedure), and formally define its associated concepts. In Section 3, we present two algorithms for discovering the JEPs in a training dataset: one using a semi-naive approach, and the other using a border-based approach. These algorithms are complementary, each being useful for certain types of training data. In Section 4, we present a process for selecting the most expressive JEPs, which efficiently reduces the complexity of the JEP-Classifier. In Section 5, we show some experimental results using a number of databases from the UCI data repository [1]. In Section 6, we outline several previously published classifiers, and compare them to the JEP-Classifier. Finally, in Section 7, we offer some concluding remarks.

2 The JEP-Classifier

The framework discussed here assumes that the training database $\mathcal{D}$ is a normal relational table, consisting of N instances defined by m distinct attributes. An attribute may take categorical values (e.g., the attribute COLOUR) or numeric values (e.g., the attribute SALARY). There are q known classes, namely Class 1, $\cdots$, Class q; the N instances have been partitioned into q sets, $\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_q$, according to their classes.

To encode $\mathcal{D}$ as a binary database, the categorical attribute values are mapped to *items* using bijections. For example, the two categorical attribute values, namely *red* and *yellow*, of COLOR, are mapped to two items: (COLOR = *red*) and (COLOR = *yellow*). For a numeric attribute, its value range is first discretized into intervals, and then the intervals are mapped to items using an approach similar to that for categorical attributes. In this work, the values of numeric attributes in the training data are discretized into 10 intervals with the same length, using the so-called *equal-length-bin* method.

Let I denote the set of all items in the encoding. An *itemset* X is defined as a subset of I. The *support* of an itemset X over a dataset $\mathcal{D}'$ is the fraction of instances in $\mathcal{D}'$ that contain X, and is denoted $supp_{\mathcal{D}'}(X)$.

The most frequently used notion, JEPs, is defined as follows:

Definition 1. *The* JEPs *from $\mathcal{D}'$ to $\mathcal{D}''$, denoted* $\mathrm{JEP}(\mathcal{D}', \mathcal{D}'')$*, (or called the JEPs of $\mathcal{D}''$ over $\mathcal{D}'$, or simply the JEPs of $\mathcal{D}''$ if $\mathcal{D}'$ is understood), are the itemsets whose supports in $\mathcal{D}'$ are zero but in $\mathcal{D}''$ are non-zero.*

They are named *jumping* emerging patterns (JEPs), because the supports of JEPs grow sharply from zero in one dataset to non-zero in another dataset.

To handle the general case where the training dataset contains more than two classes, we introduce the concept of *pair-wise features*.

Definition 2. *The* pair-wise features *in a dataset $\mathcal{D}$, whose instances are partitioned into q classes $\mathcal{D}_1, \cdots, \mathcal{D}_q$, consist of the following q groups of JEPs: those of $\mathcal{D}_1$ over $\cup_{j=2}^{q}\mathcal{D}_j$, those of $\mathcal{D}_2$ over $\cup_{j\neq 2}^{q}\mathcal{D}_j$, $\cdots$, and those of $\mathcal{D}_q$ over $\cup_{j=1}^{q-1}\mathcal{D}_j$.*

For example, if $q = 3$, then the pair-wise features in $\mathcal{D}$ consist of 3 groups of JEPs: those of $\mathcal{D}_1$ over $\mathcal{D}_2 \cup \mathcal{D}_3$, those of $\mathcal{D}_2$ over $\mathcal{D}_1 \cup \mathcal{D}_3$, and those of $\mathcal{D}_3$ over $\mathcal{D}_1 \cup \mathcal{D}_2$.

Example 2. The pair-wise features in $\mathcal{D}_1$ and $\mathcal{D}_2$ of Table 1 consist of the JEPs from $\mathcal{D}_1$ to $\mathcal{D}_2$, $\{a,b\}, \{a,b,c\}, \{a,b,d\}, \{a,b,c,d\}$, and the JEPs from $\mathcal{D}_2$ to $\mathcal{D}_1$, $\{a,e\}, \{b,e\}, \{a,c,e\}, \{a,d,e\}, \{b,c,e\}, \{b,d,e\}, \{c,d,e\}, \{a,c,d,e\}, \{b,c,d,e\}$.

Note that we *do not enumerate* all these JEPs individually in our algorithms. Instead, we use *borders* to represent them. Also, the border representation mechanism facilitates the simple selection of the most expressive JEPs. The concept of border was proposed in [3] to succinctly represent a large collection of sets. (It will be reviewed later in section 3.)

Continuing with the above example, the JEPs from $\mathcal{D}_1$ to $\mathcal{D}_2$ can be represented by the border of $<\{\{a,b\}\}, \{\{a,b,c,d\}\}>$. Its *left bound* is $\{\{a,b\}\}$, and its *right bound* is $\{\{a,b,c,d\}\}$; it represents all those sets that are supersets of some itemset in its left bound, and are subsets of some itemset in its right bound. Obviously, $\{a,b\}$, the itemset in the left bound, has the *largest support* among all itemsets covered by the border. Similarly, the JEPs from $\mathcal{D}_2$ to $\mathcal{D}_1$ can be represented by two borders: $<\{\{a,e\}, \{c,d,e\}\}, \{\{a,c,d,e\}\}>$ and $<\{\{b,e\}, \{c,d,e\}\}, \{\{b,c,d,e\}\}>$. (Details will be given in Section 4.) Therefore, the *most expressive* JEPs in $\mathcal{D}_1$ and $\mathcal{D}_2$ are those in the set of $\{\{a,b\}, \{a,e\}, \{b,e\}, \{c,d,e\}\}$, the *union* of the left bounds of the three borders above. Observe that it is much smaller than the set of all JEPs.

In JEP-Classifier, the most expressive JEPs play a central role. To classify a test instance T, we evaluate the collective impact of only the most expressive JEPs that are subsets of T.

Definition 3. *Given a pair of datasets $\mathcal{D}'$ and $\mathcal{D}''$ and a test instance T, the collective impact in favour of the class of $\mathcal{D}'$ contributed by the most expressive JEPs of $\mathcal{D}'$ and of $\mathcal{D}''$ is defined as*

$$\sum_{X \in \textit{MEJEP}(\mathcal{D}', \mathcal{D}'') \textit{ and } X \subseteq T} supp_{\mathcal{D}'}(X),$$

where MEJEP$(\mathcal{D}', \mathcal{D}'')$ is the union of the most expressive JEPs of $\mathcal{D}'$ over $\mathcal{D}''$ and the most expressive JEPs of $\mathcal{D}''$ over $\mathcal{D}'$. The collective impact in favour of the class of $\mathcal{D}''$ is defined similarly.

The *classification procedure* of JEP-Classifier for a given test instance is a simple process as follows. Given a test instance T, the q collective impacts respectively in favour of the q classes are first computed. Then, the JEP-Classifier determines the class label as the class where T obtains the largest collective impact. When a tie occurs (i.e., the collective impacts obtained are equal), we can use popularities to break the tie.

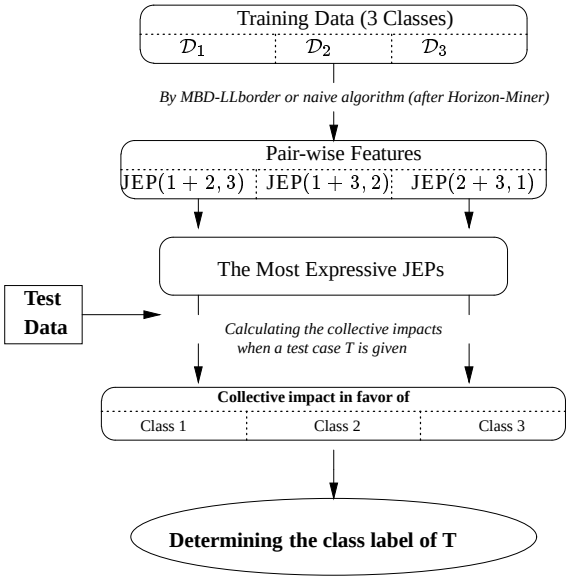

Fig. 1. JEP-Classifier working on a database with three classes of data.

Figure 1 depicts how the JEP-Classifier is built from the training data, and how it is then used to classify testing data, for the case when a database contains three classes of data. In this figure, JEP$(1+2,3)$ represents the JEPs from $\mathcal{D}_1 \cup \mathcal{D}_2$ to $\mathcal{D}_3$, and similarly for JEP$(1+3,2)$ and JEP$(2+3,1)$. The HORIZON-MINER [4] and MBD-LLBORDER [3] algorithms, used to extract the pair-wise features from the training dataset, are outlined in Section 3. Determining the most expressive JEPs is discussed in Section 4.

3 Discovering the Pair-wise Features

As the pair-wise features in $\mathcal{D}$ are defined as the JEPs over q pairs of datasets, we only need to consider how to discover the JEPs over one pair of datasets. Without loss of generality, suppose dataset $\mathcal{D}$ consists of only two classes of data $\mathcal{D}_1$ and $\mathcal{D}_2$, then the pair-wise features in $\mathcal{D}$ are the JEPs from $\mathcal{D}_1$ to $\mathcal{D}_2$ and the JEPs from $\mathcal{D}_2$ to $\mathcal{D}_1$. Now, we consider how to discover the JEPs from $\mathcal{D}_1$ to $\mathcal{D}_2$.

The most naive way to find the JEPs from $\mathcal{D}_1$ to $\mathcal{D}_2$ is to check the frequencies, in $\mathcal{D}_1$ and $\mathcal{D}_2$, of all itemsets. This is clearly too expensive to be feasible. The problem of efficiently mining JEPs from dense and high-dimensional datasets is well-solved in [3][4]. The high efficiency of these algorithms is a consequence of their novel use of borders [3]. In the following subsections we present two approaches to discovering JEPs. The first approach is a semi-naive algorithm which makes limited use of borders, while the second approach uses an efficient border-based algorithm called MBD-LLBORDER [3].

3.1 Borders, Horizontal Borders, and HORIZON-MINER

A *border* is a structure used to succinctly represent certain large collections of sets.

Definition 4. *[3]. A border is an ordered pair $<\mathcal{L}, \mathcal{R}>$ such that each of $\mathcal{L}$ and $\mathcal{R}$ is an antichain collection of sets, each element of $\mathcal{L}$ is a subset of some element in $\mathcal{R}$, and each element of $\mathcal{R}$ is a superset of some element in $\mathcal{L}$; $\mathcal{L}$ is the* left bound *of the border, and $\mathcal{R}$ is its* right bound.

The collection of sets represented *by $<\mathcal{L}, \mathcal{R}>$ (also called the* set interval *of $<\mathcal{L}, \mathcal{R}>$) is*

$$[\mathcal{L}, \mathcal{R}] = \{Y \mid \exists X \in \mathcal{L},\ \exists Z \in \mathcal{R} \ \text{ such that } \ X \subseteq Y \subseteq Z\}.$$

We say that $[\mathcal{L}, \mathcal{R}]$ has *$<\mathcal{L}, \mathcal{R}>$ as its border, and that each $X \in [\mathcal{L}, \mathcal{R}]$ is covered by $<\mathcal{L}, \mathcal{R}>$.*

Example 3. The set interval of $<\{\{a, b\}\}, \{\{a, b, c, d, e\}, \{a, b, d, e, f\}\}>$ consists of twelve itemsets, namely all sets that are supersets of $\{a, b\}$ and that are subsets of either $\{a, b, c, d, e\}$ or $\{a, b, d, e, f\}$.

Definition 5. *The* horizontal border *of a dataset is the border $<\{\emptyset\}, \mathcal{R}>$ that represents all* non-zero support *itemsets in the dataset.*

Example 4. The horizontal border of $\mathcal{D}_1$ in Table 1 is $<\{\emptyset\}, \{\{a, c, d, e\}, \{b, c, d, e\}\}>$.

The simple HORIZON-MINER algorithm [4] was proposed to discover the horizontal border of a dataset. The basic idea of this algorithm is to select the maximum itemsets from all instances in $\mathcal{D}$ (an itemset is maximal in the collection $\mathcal{C}$ of itemsets if it has no proper superset in $\mathcal{C}$). HORIZON-MINER is very efficient as it requires only one scan through the dataset.

3.2 The Semi-naive Approach to Discovering JEPs

The semi-naive algorithm for discovering the JEPs from $\mathcal{D}_1$ to $\mathcal{D}_2$ consists of the following two steps: (i) Use HORIZON-MINER to discover the horizontal border of $\mathcal{D}_2$; (ii) Scan $\mathcal{D}_1$ to check the supports of all itemsets covered by the horizontal border of $\mathcal{D}_2$; the JEPs are those itemsets with zero support in $\mathcal{D}_1$. The pruned SE-tree [3] can be used in this process to irredundantly and completely enumerate the itemsets represented by the horizontal border.

The semi-naive algorithm is fast on small databases. However, on large databases, a huge number of itemsets with non-zero support make the semi-naive algorithm too slow to be practical. With this in mind, in the next subsection, we present a method which is more efficient when dealing with large databases.

3.3 Border-based Algorithm to Discover JEPs

In general, MBD-LLBORDER [3] finds those itemsets whose supports in $\mathcal{D}_2$ are $\geq$ some support threshold θ but whose support in $\mathcal{D}_1$ are less than some support threshold δ for a pair of dataset $\mathcal{D}_1$ and $\mathcal{D}_2$. Specially, this algorithm produces exactly all those itemsets whose supports are nonzero in $\mathcal{D}_2$ but whose supports are zero in $\mathcal{D}_1$, namely the JEPs from $\mathcal{D}_1$ to $\mathcal{D}_2$. In this case, MBD-LLBORDER takes the horizontal border from $\mathcal{D}_1$ and the horizontal border from $\mathcal{D}_2$ as inputs. Importantly, this algorithm does not output all JEPs individually. Instead, MBD-LLBORDER outputs a family of borders in the form of $<\mathcal{L}_i, \mathcal{R}_i>$, $i = 1, \cdots, k$, to concisely represent all JEPs.

Unlike the semi-naive algorithm, which must scan the dataset $\mathcal{D}_1$ to discover the JEPs, MBD-LLBORDER works by manipulating the horizontal borders of the datasets $\mathcal{D}_1$ and $\mathcal{D}_2$. As a result, the MBD-LLBORDER algorithm scales well to large databases. This is confirmed by the experimental results in Section 5. The MBD-LLBORDER algorithm for discovering JEPs is described in detail in the Appendix.

4 Selecting the Most Expressive JEPs

We have given two algorithms to discover the pair-wise features from the training data $\mathcal{D}$: the semi-naive algorithm is useful when $\mathcal{D}$ is small, while the MBD-LLBORDER algorithm is useful when $\mathcal{D}$ is large. As seen in the past section, the MBD-LLBORDER algorithm outputs the JEPs represented by borders. These borders can represent very large collections of itemsets. However, only those itemsets with large support contribute significantly to the collective impact used to classify a test instance. By using only the most expressive JEPs in the JEP-Classifier, we can greatly reduce its complexity, and strengthen its resistance to noise in the training data.

Consider $\text{JEP}(\mathcal{D}_1, \mathcal{D}_2) \cup \text{JEP}(\mathcal{D}_2, \mathcal{D}_1)$, the pair-wise features in $\mathcal{D}$. Observe that $\text{JEP}(\mathcal{D}_1, \mathcal{D}_2)$ is represented by a family of borders of the form $<\mathcal{L}_i, \mathcal{R}_i>$, $i = 1, \cdots, k$, where the $\mathcal{R}_i$ are singleton sets (see the pseudo-code for MBD-LLBORDER in the Appendix). We believe that the itemsets in the left bounds, $\mathcal{L}_i$, are the most expressive JEPs in the dataset. The reasons behind this selection include:

- By definition, the itemsets in the left bound of a border have the largest supports of all the itemsets covered by that border because the supersets of an itemset X have smaller supports than that of X. Then, the most expressive JEPs cover more instances (at least equal) of the training dataset than the other JEPs.
- Any proper subset of the most expressive JEPs is not a JEP any more.

It follows that we can select the most expressive JEPs of $\text{JEP}(\mathcal{D}_1, \mathcal{D}_2)$ by taking the union of the left bounds of the borders produced by MBD-LLBORDER. This union is called the LEFT-UNION of $\text{JEP}(\mathcal{D}_1, \mathcal{D}_2)$. So, LEFT-UNION $= \cup\mathcal{L}_i$. Similarly, we can select the most expressive JEPs of $\text{JEP}(\mathcal{D}_2, \mathcal{D}_1)$. Combining the two LEFT-UNION, the most expressive pair-wise features in $\mathcal{D}$ are then constructed.

Algorithmically, finding LEFT-UNION can be done very efficiently. If the MBD-LLBORDER algorithm is used, then we simply use the left bounds of the borders it produces. In practice, this can be done by replacing the last line of the pseudo code of the MBD-LLBORDER algorithm in the Appendix with

return the union of the left bounds of all borders in EPBORDERS.

If the semi-naive algorithm is used, then LEFT-UNION can be updated as each new JEP is discovered.

Example 5. To illustrate several points discussed in this subsection, consider $\mathcal{D}_1$ and $\mathcal{D}_2$ from Table 1. The horizontal border of $\mathcal{D}_1$ is $<\{\emptyset\}, \{acde, bcde\}>$[1], and that of $\mathcal{D}_2$ is $<\{\emptyset\}, \{ce, de, abcd\}>$. The JEPs from $\mathcal{D}_2$ to $\mathcal{D}_1$ are represented by two borders $<\mathcal{L}_i, \mathcal{R}_i>$, $i = 1, 2$, namely $<\{ae, cde\}, \{acde\}>$ and $<\{be, cde\}, \{bcde\}>$. (The readers can use MBD-LLBORDER in the Appendix to derive these borders.)

The border $<\{ae, cde\}, \{acde\}>$ consists of the JEPs $\{ae, ace, ade, cde, acde\}$, while the border $<\{be, cde\}, \{bcde\}>$ consists of the JEPs $\{be, bce, bde, cde, bcde\}$. Note that the JEPs in the left bounds have the largest supports.

The LEFT-UNION of JEP$(\mathcal{D}_2, \mathcal{D}_1)$ is the union of the left bounds of the above two borders, namely $\{ae, cde\} \cup \{be, cde\} = \{ae, be, cde\}$.

5 Experimental Results

In this section we present the results of our experiments, where we run the JEP-Classifier on 30 databases (some contain up to 10 classes, some have up to 30162 instances, some have up to 60 attributes) taken from the UCI Repository of Machine Learning Databases [1]. These experiments were carried out on a 500MHz PentiumIII PC with 512M bytes of RAM. The accuracy was obtained using the methodology of ten-fold cross-validation [10] (but one fold was tested in census-income).

The experiment's pre-processes are: (i) download original datasets, say $\mathcal{D}$, from the UCI website; (ii) partition $\mathcal{D}$ into class datasets $\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_q$; (iii) randomly shuffle $\mathcal{D}_i, i = 1, \cdots, q$; (iv) for each $\mathcal{D}_i$, choose the first 10% instances as the testing data and the remaining 90% as the training data. Repeatedly, choose the second 10% as the testing data, and so forth; (v) if there exist continuous attributes, discretize them by our *equal-length-bin* method in the training datasets first, and then map the intervals to the testing data. This step is used to convert the original training and testing data into the standard binary transactional data. (These executable codes are available from the authors on request.) After pre-processing, we followed the steps illustrated in Figure 1 to get the results. Alternatively, MLC++ technique [7] was also used to discretize continuous attributes in the glass, ionosphere, pima, sonar, and vehicle datasets. These testing accuracies are reported in Table 2. The main disadvantage of MLC++ technique is that it sometimes, for example in the liver dataset, produces many different instances with different labels into identical instances.

Table 2 summarizes the results. In this table, the first column lists the name of each database, followed by the numbers of instances, attributes, and classes in Column 2. The third column presents the error rate of the JEP-Classifier, calculated as the percentage of test instances incorrectly predicted. Similarly, columns 4 and 5 give the error rate of, respectively, the CBA classifier in [8] and C4.5. (These results are the *best results* taken from Table 1 in [8]; a dash indicates that we were not able to find previous reported

[1] For readability, we use $acde$ as shorthand for the set $\{a, c, d, e\}$.

Table 2. Accuracy Comparison.

Datasets	#inst, attri, class	JEP-Cla.	CBA	C4.5rules	# JEPs	#CARs
anneal*	998, 38, 5	4.4	**1.9**	5.2	5059	65081
australian*	690, 14, 2	13.66	**13.2**	13.5	9806	46564
breast-w*	699, 10, 2	**3.73**	3.9	3.9	2190	399
census	**30162**, 16, 2	**12.7**	–	–	68053	–
cleve*	303, 13, 2	**15.81**	16.7	18.2	8633	1634
crx*	690, 15, 2	**14.06**	14.1	15.1	9880	4717
diabete*	768, 8, 2	**23.31**	24.7	25.8	4581	162
german*	1000, 20, 2	**24.8**	25.2	27.7	32510	69277
glass*	214, 9, 7	**17.4**	27.4	27.5	127	291
heart*	270, 13, 2	**17.41**	18.5	18.9	7596	624
hepatitis*	155, 19, 2	17.40	**15.1**	19.4	5645	2275
horse*	368, 28, 2	16.8	17.9	**16.3**	22425	7846
hypo*	3163, 25, 2	2.69	1.6	**0.8**	1903	493
ionosphere*	351, 34, 2	**6.9**	7.9	8.0	8170	10055
iris*	150, 4, 3	**2.67**	7.1	4.7	161	23
labor*	57, 16, 2	**8.67**	17.0	20.7	1400	313
liver	345, 6, 2	**27.23**	–	32.6	1269	–
lymph*	148, 18, 4	28.4	**18.9**	21.0	5652	2965
mushroom	**8124**, 22, 2	**0.0**	–	–	2985	–
nursery	**12960**, 8, 5	**1.04**	–	–	1331	–
pima*	768, 8, 2	**20.4**	26.9	24.5	54	2977
sick*	4744, 29, 2	2.33	2.7	**1.5**	2789	627
sonar*	208, 60, 2	**14.1**	21.7	27.8	13050	1693
soybean	47, 34, 4	**0.00**	–	8.3	1928	–
tic-tac-toe*	958, 9, 2	1.0	**0.0**	0.6	2926	1378
vehicle*	846, 18, 4	27.9	31.2	**27.4**	19461	5704
vote1*	433, 16, 2	8.53	6.4	**4.8**	5783	–
wine*	178, 13, 3	**6.11**	8.4	7.3	5531	1494
yeast*	1484, 8, 10	**33.72**	44.9	44.3	2055	–
zoo*	101, 16, 7	**4.3**	5.4	7.8	624	686

results). Column 6 gives the number of the most expressive JEPs used by the JEP-Classifier. The last column gives the number of CARs used in CBA.

These results raise several points of interest.

1. Our JEP-Classifier performed perfectly (100% or above 98.5% testing accuracy) on some databases (nursery, mushroom, tic-tac-toe, soybean).
2. Among the 25 databases marked with * (indicating results of both CBA and C4.5 are available) in table 2, the JEP-Classifier outperforms both C4.5 and CBA on 15 datasets; CBA wins on 5; and C4.5 wins on 5 (in terms of the testing accuracies).
3. For the databases (with bold font), they have much larger data sizes than the remaining databases. The JEP-Classifier performs well on those datasets.
4. For unbalanced datasets (having unbalanced numbers of instances for each class), the JEP-Classifier performs well. For example, nursery dataset contains 5 classes and have respectively 4320, 2, 328, 4266, and 4044 instances in each class. Interest-

ingly, we observed that the testing accuracy by the JEP-Classifier was consistently around 100% for each class. For CBA, its support threshold was set as 1%. In this case, CBA would mis-classify all instances of class 2. The reason is that CBA cannot find the association rules in class 2.

Our experiments also indicate that the JEP-Classifier is fast and highly efficient.

- Building the classifiers took approximately 0.3 hours on average for the 30 cases considered here.
- For databases with a small number of items, such as the iris, labor, liver, soybean, and zoo databases, the JEP-Classifier completed both the learning and testing phases within a few seconds. For databases with a large number of items, such as the mushroom, sonar, german, nursery, and ionosphere databases, both phases required from one to two hours.
- In dense databases, the border representation reduced the total numbers of JEPs (by a factor of up to 10^8 or more) down to a relatively small number of border itemsets (approximately 10^3).

We also conducted experiments to investigate how the number of data instances affects the scalability of the JEP-Classifier. We selected 50%, 75%, and 90% of data instances from each original database to form three new databases. The JEP-Classifier was then applied to the three new databases. The resulting run-times shows a linear dependence on the number of data instances when the number of attributes is fixed.

6 Related Work

Extensive research on the problem of classification has produced a range of different types of classification algorithms, including nearest neighbor methods, decision tree induction, error back propagation, reinforcement learning, and rule learning. Most classifiers previously published, especially those based on classification trees (e.g., C4.5 [9], CART [2]), arrive at a classification decision by making a sequence of micro decisions, where each micro decision is concerned with one attribute only. Our JEP-Classifier, together with the CAEP classifier [5] and the CBA classifier [8], adopts a new approach by testing groups of attributes in each micro decision. While CBA uses one group at a time, CAEP and the JEP-Classifier use the aggregation of many groups of attributes. Furthermore, CBA uses association rules as the basic knowledge of its classifier, CAEP uses emerging patterns (mostly with finite growth rates), and the JEP-Classifier uses jumping emerging patterns.

While CAEP has some common merits with the JEP-Classifier, it differs from the JEP-Classifier in several ways:

1. **Basic idea**. The JEP-Classifier utilizes the JEPs of large supports (the most discriminating and expressive knowledge) to maximize its collective classification power when making decisions. CAEP uses the collective classifying power of EPs with finite growth rates, and possibly some JEPs, in making decisions.
2. **Learning phase**. In the JEP-Classifier, the most expressive JEPs are discovered by simply taking the union of the left bounds of the borders derived by the MBD-LLBORDER algorithm (specialised for discovering JEPs). In the CAEP classifier, the candidate EPs must be enumerated individually after the MBD-LLBORDER algorithm in order to determine their supports and growth rates.

3. **Classification procedure**. The JEP-Classifier's decision is based on the collective impact contributed by the most expressive pair-wise features, while CAEP's decision is based on the normalized ratio-support scores.
4. **Predicting accuracy**. The JEP-Classifier outperforms the CAEP classifier in large and high dimension databases such as mushroom, ionosphere, and sonar. For small datasets such as heart, breast-w, hepatitis, and wine databases, the CAEP classifier reaches higher accuracies than the JEP-Classifier does. On 13 datasets where results are available for CAEP, the JEP-Classifier outperforms CAEP on 9 datasets.

While our comparison to CAEP is still preliminary, we believe that CAEP and JEP-Classifiers are complementary. More investigation is needed to fully understand the advantages offered by each technique.

7 Concluding Remarks

In this paper, we have presented an important application of JEPs to the problem of classification. Using the border representation and border-based algorithms, the most expressive pair-wise features were efficiently discovered in the learning phase. The collective impact contributed by these pair-wise features were then used to classify test instances. The experimental results have shown that the JEP-Classifier generally achieves a higher predictive accuracy than previously published classifiers, including the classifier in [8], and C4.5. This high accuracy results from the strong discriminating power of an individual JEP over a fraction of the data instances and the collective discriminating power by all the most expressive JEPs. Furthermore, our experimental results show that the JEP-Classifier scales well to large datasets.

As future work, we plan to pursue several directions. (i) In this paper, collective impact is measured by the sum of the supports of the most expressive JEPs. As alternatives, we are considering other aggregates, such as the squared sum, and adaptive methods, such as neural networks. (ii) In this paper, JEPs are represented by borders. In the worst case, the number of the JEPs in the left bound of a border can reach $C^N_{N/2}$, where N is the number of attributes in the dataset. We are considering the discovery and use of only some of the itemsets in the left bound, to avoid this worst-case complexity. (iii) In discovering JEPs using the MBD-LLBORDER algorithm, there are multiple uses of the BORDER-DIFF sub-routine, dealing with different borders. By parallelizing these multiple calls, we can make the learning phase of the JEP-Classifier even faster and more scalable.

Acknowledgement

We are grateful to Richard Webber for his help. We thank Bing Liu and Jiawei Han for useful discussions. We also thank the anonymous referees for helpful suggestions.

References

1. Blake, C. L., Murphy, P. M.: UCI Repository of machine learning database. [http://www.cs.uci.edu/ mlearn/mlrepository.html]. Irvine, CA: University of California, Department of Information and Computer Science (1998)

2. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth International Group (1984)
3. Dong, G., Li, J.: Efficient mining of emerging patterns: Discovering trends and differences. Proceedings of ACM SIGKDD'99 International Conference on Knowledge Discovery & Data Mining, San Diego, USA, (1999) 43–52
4. Dong, G., Li, J., Zhang, X.: Discovering jumping emerging patterns and experiments on real datasets. Proceedings of the 9th International Database Conference (IDC'99), Hong Kong, (1999) 155–168
5. Dong, G., Zhang, X., Wong, L., Li, J.: CAEP: Classification by aggregating emerging patterns. DS-99: Second International Conference on Discovery Science, Tokyo, Japan, (1999)
6. Han, J., Fu, Y.: Exploration of the power of attribute-oriented induction in data mining. In: Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R. (eds): Advances in Knowledge Discovery and Data Mining. AAAI/MIT Press (1996) 399–421
7. Kohavi, R. , John, G. , Long, R. , Manley, D. , Pfleger, K.: MLC++: a machine learning library in C++. Tools with artificial intelligence, (1994) 740–743
8. Liu, B., Hsu, W., Ma, Y.: Integrating classification and association rule mining. Proceedings of the 4th International Conference on Knowledge Discovery in Databases and Data Mining, KDD'98 New York, USA (1998)
9. Quinlan, J. R.: C4. 5: program for machine learning. Morgan Kaufmann (1992)
10. Salzberg, S. L.: On comparing classifiers: pitfalls to avoid and a recommended approach. Data Mining and Knowledge Discovery **1** (1997) 317 – 327

Appendix: MBD-LLBORDER for Discovering JEPs

Suppose the horizontal border of $\mathcal{D}_1$ is $<\{\emptyset\}, \{C_1, \cdots, C_m\}>$ and the horizontal border of $\mathcal{D}_2$ is $<\{\emptyset\}, \{D_1, \cdots, D_n\}>$. MBD-LLBORDER finds the JEPs from $\mathcal{D}_1$ to $\mathcal{D}_2$ as follows.

MBD-LLBORDER(horizontalBorder($\mathcal{D}_1$), horizontalBorder($\mathcal{D}_2$))
;; *return all JEPs from $\mathcal{D}_1$ to $\mathcal{D}_2$ by multiple calls of* BORDER-DIFF
EPBORDERS$\leftarrow \{\}$;
for j from 1 to n **do**
  if some C_i is a superset of D_j then **continue**;
  $\{C'_1, \cdots, C'_m\} \leftarrow \{C_1 \cap D_j, \cdots, C_m \cap D_j\}$;
  RIGHTBOUND $\leftarrow$ all maximal itemsets in $\{C'_1, \cdots, C'_m\}$;
  add BORDER-DIFF($<\{\emptyset\}, D_j>, <\{\emptyset\}, $RIGHTBOUND$>$) into EPBORDERS;
return EPBORDERS;

BORDER-DIFF($<\{\emptyset\}, \{U\}>$, $<\{\emptyset\}, \{S_1, S_2, \cdots, S_k\}>$)
;; *return the border of* $[\{\emptyset\}, \{U\}] - [\{\emptyset\}, \{S_1, S_2, \cdots, S_k\}]$
initialize $\mathcal{L}$ to $\{\{x\} \mid x \in U - S_1\}$;
for $i = 2$ to k **do**
  $\mathcal{L} \leftarrow \{X \cup \{x\} \mid X \in \mathcal{L}, x \in U - S_i\}$;
  remove all itemsets Y in $\mathcal{L}$ that are not minimal;
return $<\mathcal{L}, \{U\}>$;

Note that given a collection $\mathcal{C}$ of sets, the *minimal* sets are those ones whose proper subsets are not in $\mathcal{C}$. For correctness and variations of BORDER-DIFF, the readers are referred to [3].