The Space of Jumping Emerging Patterns and Its Incremental Maintenance

Jinyan Li

Kotagiri Ramamohanarao

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

The Space of Jumping Emerging Patterns and Its Incremental Maintenance Algorithms

Jinyan Li JYLI@CS.MU.OZ.AU
Kotagiri Ramamohanarao RAO@CS.MU.OZ.AU
Department of Computer Science and Software Engineering, The University of Melbourne, Parkville, Vic. 3010, Australia

Guozhu Dong GDONG@CS.WRIGHT.EDU
Department of Computer Science and Engineering, Wright State University, Dayton OH 45435, USA

Abstract

The concept of *jumping emerging patterns* (JEPs) has been proposed to describe those discriminating features which only occur in the positive training instances but do not occur in the negative class at all; JEPs have been used to construct classifiers which generally provide better accuracy than the state-of-the-art classifiers such as C4.5. The algorithms for maintaining the *space of jumping emerging patterns* (JEP space) are presented in this paper. We prove that JEP spaces satisfy the property of convexity. Therefore JEP spaces can be *concisely* represented by two bounds: consisting respectively of the *most general* elements and the *most specific* elements. In response to insertion of new training instances, a JEP space is modified by operating on its boundary elements and the boundary elements of the JEP spaces associated with the new instances. This strategy completely avoids the need to go back to the most initial step to build the new JEP space. In addition, our maintenance algorithms can well handle such other cases as deletion of instances, insertion of new attributes, and deletion of attributes.

1. Introduction

The problem of how to discover powerful distinguishable features from classes of data is an important research topic in the field of machine learning and the field of data mining. The concept of *jumping emerging patterns* (JEPs) (Dong & Li, 1999) has been proposed to describe those discriminating features which only occur in the positive training instances but do not occur in the negative class at all. The most frequently appearing JEPs have been used to build accurate classifiers (Dong et al, 1999; Li, Dong, & Ramamohanarao, in press). Their accuracy is generally better than the state-of-the-art classifiers such as C4.5 (Quinlan, 1993). In this paper, we first propose the concept of the *space of JEPs*, called JEP space, consisting of all JEPs with respect to a given set of positive and negative data. We prove that JEP spaces satisfy the property of convexity, which means that JEP spaces can be bounded and then they can be concisely represented by the boundary elements. Forming the concept of JEP space, we shift the perspective of looking at JEPs individually to the perspective of examining all JEPs as a whole. Furthermore, we can utilize its convexity to develop efficient maintenance algorithms to modify its boundary elements in response to changes to the data. This point is extremely crucial for practical applications because modifications to the previously processed data will be frequent and the new JEP space will be constantly needed.

In this paper, the algorithms for maintaining JEP spaces in response to *insertion of new instances*, *deletion of instances*, *insertion of new attributes*, and *deletion of attributes* are proposed. As a proportion of the JEPs previously discovered before the change in the relations still constitutes valid knowledge following the changes in the data set, the maintenance algorithms take advantage of nearly repeated computations on inputs that differ slightly from one another, computing new JEP spaces incrementally by making use of the previous JEP spaces rather than recalculating from scratch. Therefore, the maintenance procedure is a chain of operations on JEP spaces. The high efficiency of these algorithms mainly stems from the operations on the boundary elements of JEP spaces rather than enumerating and examining all individual JEPs. Consequently the maintenance algorithms will provide great computational savings and validate the scalability of the JEP-based classifiers.

Our JEP space is closely related to *version space* (Hirsh, 1994; Mitchell, 1977, 1982). Given a set of positive and a set of negative training instances, a version space is the set of all *generalizations* (or *item patterns*) that each match (or be contained in) every positive instance and no negative instance in the training set. In contrast, a JEP space is the set of all item patterns that each match (or be contained in) one or more (not necessarily every) positive instances and no negative instance in the set. Therefore, the consistency restrictions with the training data are significantly different between JEP spaces and version spaces. The different consistency restrictions result in fundamentally different algorithms for creating JEP spaces and version spaces. On the other hand, the similar aspect of JEP space and version space is that both of them are convex spaces (Gunter, Ngair, & Subramanian, 1997) and both of them can be concisely represented and efficiently maintained. Moreover, often a JEP space still contains many discriminating features where a version space may contain no elements.

Another work related to JEP space is the JEP-based classifiers, one of which called JEP classifier (Li, Dong, & Ramamohanarao, in press). JEP classifier is a learning method, which consists of two phases. In the first phase, all JEPs are discovered. In the second phase, the frequencies of JEPs are weighted to form classification scores when a test instance is given. So, the efficiency for the maintenance of JEP spaces is an important factor to make JEP classifier up to date by including the new information as soon as possible.

As mentioned before, the notion of emerging patterns is a previously proposed concept. For concise representation, emerging patterns (with some constraints) and a special type of them, JEPs, are represented by *multiple* borders in Dong and Li (1999) and Li, Dong, and Ramamohanarao (in press). However, we use only *one* border instead of multiple borders to represent all JEPs in this paper. Such a one-border-representation greatly enhances the expressiveness and succinctness of border representation mechanism.

The idea of decision trees has produced numerous classifiers (e.g., C4.5). The problem of how to efficiently *restructure* a decision tree when changes occur to the data has also been addressed previously by many people. Schlimmer and Fisher (1986) proposed ID4, an incremental algorithm for efficient maintenance of decision trees, and three *dimensions* which differentiate incremental and non-incremental tree induction systems. Utgoff, Berkman, and Clouse (1997) proposed ITI (incremental tree inducer), which makes extensive use of a tree transformation mechanism, for incrementally handle new data even some of which contain noise and missing values.

The remainder of this paper is organized as follows. Section 2 introduces the concept of JEP spaces and presents the convexity of JEP spaces. A concise representation structure, called *borders*, is also described in this section. Section 3 and 4 propose our efficient incremental maintenance algorithms. Section 5 uses experimental results to evaluate the efficiency of our algorithms. Section 6 concludes this paper.

2. JEP Spaces and Borders

The concept of *JEP spaces*, *borders*, and *horizontal borders* are frequently used throughout this paper. JEPs are used to capture the frequency change of some patterns between two data sets. Borders (Dong & Li, 1999) and a special type of them, horizontal borders, are efficient representation structures of large collections of sets. We first define some basic terminologies.

Relational data is described by *attributes*. Some attributes are assigned with nominal values, e.g., the attribute COLOR having nominal values of *red*, *yellow*, and *blue*. The other attributes are continuous attributes. For example, AGE can have continuous values ranging from 0 to 150. An *attribute-value pair* is defined as an *item*. So, COLOR-*red* is an item and AGE-$[0, 10)$ is also an item after the discretization of the age values. An *instance* is defined as a set of items. An instance is called *positive* if it is labeled with the positive class. Otherwise it is called *negative*. A set of instances is called a *data set*. An *item set* is also defined as a set of items, emphasizing some subset of an instance. We say item set I_1 is *more general* than item set I_2 if $I_1 \subset I_2$; it is also said that I_2 is *more specific* than I_1. Given an item set A, the percentage of the instances in data set $\mathcal{D}$ containing A is defined as the *support* of A in $\mathcal{D}$, denoted $supp_{\mathcal{D}}(A)$.

2.1 JEP Spaces

We are interested in a type of item sets, JEPs, whose supports change abruptly from one data set to another. Formally,

Definition 2.1 Given a set $\mathcal{D}_p$ of positive instances and a set $\mathcal{D}_n$ of negative instances, a *JEP* (with respect to $\mathcal{D}_p$ and $\mathcal{D}_n$) is defined as an item set which occurs in $\mathcal{D}_p$ but does not occur in $\mathcal{D}_n$. A *JEP space* is defined as the set of all JEPs with respect to $\mathcal{D}_p$ and $\mathcal{D}_n$.

Here, an item set is considered to *occur* in a data set if and only if one or more instances in this data set contain this item set.

Note that JEP space is significantly different from version space (Mitchell, 1982) because of different consistency restrictions of their elements with the training data. As mentioned in the introduction, each element in a version space must match (or be contained in) every positive instance and no negative instance (under the partial order of set-containment.) This condition is much stricter than that of JEPs. In practice, for example, the data set in UCI repository (Blake & Murphy, 1998) always produce empty version spaces rather than those discussed in Hirsh (1994) and Mitchell (1982) which contain large, even sometimes infinite, number of elements. With a weaker consistency restriction, JEP space becomes more useful in practice.

The size of JEP spaces can be large; for example, the JEP space of the mushroom data (Blake & Murphy, 1998) contains up to 10^8. To enumerate all the elements is time consuming. Interestingly, JEP spaces hold a nice property, called *convexity*, or *interval closure*. By exploiting this property, JEP spaces can be succinctly represented by the most general and the most specific elements in them.

Definition 2.2 (Gunter, Ngair, & Subramanian, 1997; Dong & Li, 1999) A collection $\mathcal{C}$ of sets is said to be a *convex space* if the conditions $X \subseteq Y \subseteq Z$ and $X, Z \in \mathcal{C}$ imply that $Y \in \mathcal{C}$.

If a collection is a convex space, we say it holds convexity or it is interval closed.

Example 2.3 All of the sets $\{1\}$, $\{1,2\}$, $\{1,3\}$, $\{1,4\}$, $\{1,2,3\}$, and $\{1,2,4\}$ form a convex space. The set $\mathcal{L}$ of all the most general elements in this space is $\{\{1\}\}$; the set $\mathcal{R}$ of all the most specific elements in this space is $\{\{1,2,3\},\{1,2,4\}\}$. All the other elements can be considered "between" $\mathcal{L}$ and $\mathcal{R}$. ∎

Theorem 2.4 Given a set $\mathcal{D}_p$ of positive instances and a set $\mathcal{D}_n$ of negative instances, the JEP space with respect to $\mathcal{D}_p$ and $\mathcal{D}_n$ is a convex space.

Proof: Suppose item sets X and Z satisfy (i) $X \subseteq Z$; (ii) X and Z are two JEPs. Then, for any item set Y satisfying $X \subseteq Y \subseteq Z$, Y is also a JEP. This is because

- X does not occur in $\mathcal{D}_n$. So, all of its supersets, which are more specific than X, cannot occur in $\mathcal{D}_n$. Therefore, Y cannot occur in $\mathcal{D}_n$.

- the support in $\mathcal{D}_p$ of item set Z is not zero. So, all subsets of Z, which are more general than Z, have a non-zero support in $\mathcal{D}_p$. Therefore, Y occurs in $\mathcal{D}_p$ indeed.

Consequently, the JEP space with respect to $\mathcal{D}_p$ and $\mathcal{D}_n$ holds convexity. ∎

Using this property, JEP spaces can be represented and bounded by two sets like the sets $\mathcal{L}$ and $\mathcal{R}$ in Example 2.3, which play the boundary role.

With the $\mathcal{L}$-and-$\mathcal{R}$ representation, all JEPs in a JEP space can be generated and recognized by examining its bounds. We next formalize the two boundary sets as the concept of *borders*.

2.2 Using Borders to Represent JEP Spaces

A border is a structure, consisting of two bounds. A simple example might be $<\{\{a\},\{b\}\},\{\{a,b,c\},\{b,d\}\}>$, which represent all those sets which are supersets of $\{a\}$ or $\{b\}$ and subsets of $\{a,b,c\}$ or $\{b,d\}$. Formally,

Definition 2.5 (Dong & Li, 1999) An ordered pair $<\mathcal{L},\mathcal{R}>$ is called a *border*, $\mathcal{L}$ the *left bound* of this border and $\mathcal{R}$ the *right bound*, if (i) each one of $\mathcal{L}$ and $\mathcal{R}$ is an antichain — a collection of sets in which any two elements X and Y satisfy $X \not\subseteq Y$ and $Y \not\subseteq X$, (ii) 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 $\mathcal{L}$. The collection of sets *represented* by a border $<\mathcal{L},\mathcal{R}>$ consists of those item sets which are supersets of some element in $\mathcal{L}$ but subsets of some element in $\mathcal{R}$. This collection is denoted $[\mathcal{L},\mathcal{R}] = \{Y \mid \exists X \in \mathcal{L}, \exists Z \in \mathcal{R}$ such that $X \subseteq Y \subseteq Z\}$.

Note the difference and similarity between the two notations of $<\mathcal{L},\mathcal{R}>$ and $[\mathcal{L},\mathcal{R}]$. A border $<\mathcal{L},\mathcal{R}>$ is a syntactic object consisting of the two bounds $\mathcal{L}$ and $\mathcal{R}$, and its semantics is $[\mathcal{L},\mathcal{R}]$ consisting of the interval sets bounded by the sets in $\mathcal{L}$ from below and by the sets in $\mathcal{R}$ from above.

There is a one-to-one correspondence between borders and convex spaces.

Proposition 2.6 Each convex space $\mathcal{C}$ has a unique border $<\mathcal{L},\mathcal{R}>$, where $\mathcal{L}$ is the collection of the most general sets in $\mathcal{C}$ and $\mathcal{R}$ is the collection of the most specific sets in $\mathcal{C}$.

In summary, it can be seen that

- Given a border $<\mathcal{L},\mathcal{R}>$, then its corresponding collection $[\mathcal{L},\mathcal{R}]$ is a convex space.

- Given a convex space, then it can be represented by a unique border.

Example 2.7 (**Horizontal space**). Given a data set $\mathcal{D}$, all *non-zero support* item sets Xs, namely, $supp_{\mathcal{D}}(X) \neq 0$, construct a convex space. This is mainly due to the fact that any subset of a non-zero support item set has a non-zero support. This convex space is specially called *horizontal space*. Horizontal spaces can be used to *exclude* those item sets Ys which do not occur in $\mathcal{D}$, namely $supp_{\mathcal{D}}(Y) = 0$. As horizontal space is a convex space, then it can be represented by a border. This border is specially called *horizontal border*. The left bound of this border is $\{\emptyset\}$ and the right bound is the set $\mathcal{R}$ of all most specific non-zero support item sets. The right bound $\mathcal{R}$ can be imagined as a horizontal line which separates all non-zero support item sets from those zero support item sets. The most specific non-zero support item sets can be simply identified in $\mathcal{D}$, viewing each instance as an item set. ∎

Differences between JEP space and horizontal space are obvious. A JEP space is associated with two data sets $\mathcal{D}_p$ and $\mathcal{D}_n$, while a horizontal space is associated with one data set. The sharp difference between two data sets can be described by JEP space, but the support trend of item sets within a data set can be described by horizontal space.

Next, horizontal spaces are very useful for rewriting and computing JEP spaces.

Proposition 2.8 Given a set $\mathcal{D}_p$ of positive instances and a set $\mathcal{D}_n$ of negative instances, then the JEP space with respect to $\mathcal{D}_p$ and $\mathcal{D}_n$ is

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

where, $[\{\emptyset\}, \mathcal{R}_p]$ is the horizontal space of $\mathcal{D}_p$ and $[\{\emptyset\}, \mathcal{R}_n]$ is the horizontal space of $\mathcal{D}_n$.

Proof: By definition, all elements of a JEP space must occur in the positive data set but not in the negative data set. So, subtracting all non-zero support item sets in $\mathcal{D}_n$ from all non-zero support item sets in $\mathcal{D}_p$ produces all the JEPs. ∎

Therefore, it can be seen that a JEP space can be represented by two horizontal borders. Based on this idea, the border representation of a JEP space can be efficiently derived by border-based algorithms proposed in (Dong & Li, 1999), which will be reviewed later. This idea also lays down a foundation for maintaining JEP spaces efficiently. Throughout this paper, when we say there is a *given* JEP space, then it means that the border $<\mathcal{L}, \mathcal{R}>$ of the JEP space is known.

3. Maintenance Algorithms for JEP Spaces

Efficient algorithms for maintaining JEP spaces should at least avoid totally going back to the most initial stage to construct a new space when some change occurs to the original training data sets $\mathcal{D}_p$ and $\mathcal{D}_n$. Given a JEP space based on one set of training data and assuming a set Δ_p of new positive instances are inserted, our maintenance algorithms show that the new JEP space, in which all JEPs consistent with all the previously processed instances and plus the new instances, is the *union* of the previous JEP space and a JEP space associated with Δ_p. Similarly, when a set Δ_n of new negative instances are inserted, the new JEP space is the *intersection* of the previous JEP space and some JEP space associated with Δ_n. Therefore, the maintenance procedure is a "chain" of intersection or union of old JEP spaces and some JEP spaces created by new data. More importantly, the border of the resulting new JEP space can be derived by manipulating the borders of the two provided JEP spaces rather than selecting the most general and the most specific elements from the intersection or union of the two provided JEP spaces. Next, we discuss the maintenance algorithms in length.

3.1 Insertion of New Positive Instances

Suppose $\mathcal{D}_p$ and $\mathcal{D}_n$ are the old data sets of positive and negative instances respectively, and suppose Δ_p is the set of newly inserted positive instances. Let their horizontal borders be respectively $<\{\emptyset\}, \mathcal{R}_p>$, $<\{\emptyset\}, \mathcal{R}_n>$, and $<\{\emptyset\}, \mathcal{R}_p^{\Delta}>$. Then the JEP space with respect to $(\mathcal{D}_p + \Delta_p)$ and $\mathcal{D}_n$ is precisely the following set

$$\begin{aligned} & ([\{\emptyset\}, \mathcal{R}_p] \cup [\{\emptyset\}, \mathcal{R}_p^{\Delta}]) - [\{\emptyset\}, \mathcal{R}_n] \\ = \; & ([\{\emptyset\}, \mathcal{R}_p] - [\{\emptyset\}, \mathcal{R}_n]) \cup ([\{\emptyset\}, \mathcal{R}_p^{\Delta}] - [\{\emptyset\}, \mathcal{R}_n]) \end{aligned}$$

Here, the first term is exactly the JEP space with respect to $\mathcal{D}_p$ and $\mathcal{D}_n$, whose border is known explicitly. The second term is exactly the JEP space with respect to Δ_p and $\mathcal{D}_n$, which is brought by the insertion of a set of new positive instances. As the union of the two JEP spaces is a JEP space as well, the border of the resulting JEP space can be efficiently computed by manipulating the borders of the operand JEP spaces. Therefore, the maintenance in response to the insertion of new positive instances consists of two steps:

1. Discovering the border of the JEP space with respect to Δ_p and $\mathcal{D}_n$;
2. Taking a union operation on two borders. (The union operation is introduced shortly.)

As will be seen in Section 5, our experiments show that this maintenance algorithm is much more efficient than the naive approach of recomputing from scratch.

3.2 Insertion of New Negative Instances

Suppose the horizontal border of the set Δ_n of new negative instances is $<\{\emptyset\}, \mathcal{R}_n^{\Delta}>$, following the notations discussed in the above subsection, then the JEP space with respect to $\mathcal{D}_p$ and $\mathcal{D}_n + \Delta_n$ is precisely the following set

$$\begin{aligned} & [\{\emptyset\}, \mathcal{R}_p] - ([\{\emptyset\}, \mathcal{R}_n] \cup [\{\emptyset\}, \mathcal{R}_n^{\Delta}]) \\ = \; & ([\{\emptyset\}, \mathcal{R}_p] - [\{\emptyset\}, \mathcal{R}_n]) \cap ([\{\emptyset\}, \mathcal{R}_p] - [\{\emptyset\}, \mathcal{R}_n^{\Delta}]) \end{aligned}$$

Once again the first term signifies a JEP space already known. The second term generates the JEP space with respect to $\mathcal{D}_p$ and Δ_n.

With a new specification of how the intersection of two JEP spaces is taken, the maintenance algorithm in response to the insertion of new negative instances is similar to the case where in response to the insertion of new positive instances. The two steps are as follows.

1. Discovering the border of the JEP space with respect to $\mathcal{D}_p$ and Δ_n;
2. Taking an intersection operation on two borders.

It can be seen again that significant computational savings will be achieved in our maintenance algorithm.

3.3 Border Operations: Difference, Union, and Intersection

There are three border operations involved in the maintenance algorithm as discussed above. These operations include *border union* of two JEP spaces, *border intersection* of two JEP spaces, and *border difference* of two horizontal spaces. Note that border difference of two horizontal spaces is used to derive the border of a JEP space, e.g., the border of $[\{\emptyset\}, \mathcal{R}_p] - [\{\emptyset\}, \mathcal{R}_n]$, of $[\{\emptyset\}, \mathcal{R}_p] - [\{\emptyset\}, \mathcal{R}_n^{\Delta}]$, and of $[\{\emptyset\}, \mathcal{R}_p^{\Delta}] - [\{\emptyset\}, \mathcal{R}_n]$. The most important characteristics of border operations is that the outcome of boundary element operations can still be used to represent convex spaces. Border union of two JEP spaces and border intersection of two JEP spaces are new in this paper, while the operation of border difference is fundamentally similar to MBD-LLBORDER (Dong & Li, 1999) with a slight difference in output.

3.3.1 Border Union of Two JEP Spaces

The problem of how to take border union operation is described as follows:

- **Given:**
  1. Two sets $\mathcal{D}_{p1}$ and $\mathcal{D}_{p2}$ of positive instances and one set $\mathcal{D}_n$ of negative instances.
  2. The border $<\mathcal{L}_1, \mathcal{R}_1>$ of the JEP space with respect to $\mathcal{D}_{p1}$ and $\mathcal{D}_n$.
  3. The border $<\mathcal{L}_2, \mathcal{R}_2>$ of the JEP space with respect to $\mathcal{D}_{p2}$ and $\mathcal{D}_n$.
- **Determine:** The border of $[\mathcal{L}_1, \mathcal{R}_1] \cup [\mathcal{L}_2, \mathcal{R}_2]$.
- **Algorithm:**
  1. $\mathcal{R}_3 \leftarrow$ the set of the most specific elements in $\mathcal{R}_1 \cup \mathcal{R}_2$;
  2. $\mathcal{L}_3 \leftarrow \mathcal{L}_1 \cup \mathcal{L}_2$;
  3. $<\mathcal{L}_3, \mathcal{R}_3>$ is the border of $[\mathcal{L}_1, \mathcal{R}_1] \cup [\mathcal{L}_2, \mathcal{R}_2]$.

The algorithm is correct because: (i) the most specific elements in $\mathcal{R}_1 \cup \mathcal{R}_2$ are exactly the most specific JEPs with respect to $(\mathcal{D}_{p1} \cup \mathcal{D}_{p2})$ and $\mathcal{D}_n$; (ii) the most general JEPs with respect to $\mathcal{D}_{p_i}$ and $\mathcal{D}_n$ $(i = 1, 2)$ remain the most general JEPs with respect to $(\mathcal{D}_{p1} \cup \mathcal{D}_{p2})$ and $\mathcal{D}_n$. The second point means that there is no element X in $\mathcal{L}_1$ and no element Y in $\mathcal{L}_2$ such that $X \subset Y$ or $X \supset Y$. This point is proved as follows. Suppose there exist $X \in \mathcal{L}_1$ and $Y \in \mathcal{L}_2$ satisfying $X \subset Y$. Then all proper subsets of Y, definitely including X, must occur in $\mathcal{D}_n$ (because Y is a most general JEP with respect to $\mathcal{D}_{p2}$ and $\mathcal{D}_n$); this is *contradictory* with the fact that X does not occur in $\mathcal{D}_n$ (because X is a JEP with respect to $\mathcal{D}_{p1}$ and $\mathcal{D}_n$).

We note that the algorithm above for the border union of two JEP spaces is correct, due to the constraint that the set of negative instances of the two input borders are the same. However, this algorithm does not work for the union of arbitrary general borders.

3.3.2 Border Intersection of Two JEP Spaces

Similarly as discussed above, the problem of how to take border intersection operation is described as follows:

- **Given:**
  1. One set $\mathcal{D}_p$ of positive instances and two sets $\mathcal{D}_{n1}$ and $\mathcal{D}_{n2}$ of negative instances.
  2. The border $<\mathcal{L}_1, \mathcal{R}_1>$ of the JEP space with respect to $\mathcal{D}_p$ and $\mathcal{D}_{n1}$.
  3. The border $<\mathcal{L}_2, \mathcal{R}_2>$ of the JEP space with respect to $\mathcal{D}_p$ and $\mathcal{D}_{n2}$.
- **Determine:** The border of $[\mathcal{L}_1, \mathcal{R}_1] \cap [\mathcal{L}_2, \mathcal{R}_2]$.
- **Algorithm:**
  1. $\mathcal{R}_3 \leftarrow \mathcal{R}_1 \cap \mathcal{R}_2$;
  2. $\mathcal{L}'_3 \leftarrow \{A \cup B \mid A \in \mathcal{L}_1, B \in \mathcal{L}_2\}$;
  3. $\mathcal{L}_3 \leftarrow$ the set of the most general elements in $\mathcal{L}'_3$;
  4. remove those elements C in $\mathcal{L}_3$ such that no elements in $\mathcal{R}_3$ contain C;
  5. $<\mathcal{L}_3, \mathcal{R}_3>$ is the border of $[\mathcal{L}_1, \mathcal{R}_1] \cap [\mathcal{L}_2, \mathcal{R}_2]$.

The algorithm is correct because: (i) the most specific JEPs with respect to $\mathcal{D}_p$ and $\mathcal{D}_{n1} \cup \mathcal{D}_{n2}$ are exactly the elements in $\mathcal{R}_1 \cap \mathcal{R}_2$; (ii) the set $\{A \cup B \mid A \in \mathcal{L}_1, B \in \mathcal{L}_2\}$ is a candidate set of the most general JEPs with respect to $\mathcal{D}_p$ and $\mathcal{D}_{n1} \cup \mathcal{D}_{n2}$. According to the definition of border, the non-most general elements and those elements which are not contained in any elements in $\mathcal{R}_3$ must be removed from this candidate set.

Note that a more general case of border intersection operation was proposed in (Hirsh, 1994; Gunter, Ngair, & Subramanian, 1997).

3.3.3 Border Difference

The operation of border difference is mainly used to discover the border of a JEP space when a set $\mathcal{D}_p$ of positive instances and a set $\mathcal{D}_n$ of negative instances are given, namely to discover the border of the difference of two horizontal spaces. Suppose the horizontal border of $\mathcal{D}_p$ is $<\{\emptyset\}, \{A_1, A_2, \cdots, A_{k_1}\}>$ and the horizontal border of $\mathcal{D}_n$ is $<\{\emptyset\}, \{B_1, B_2, \cdots, B_{k_1}\}>$, a pseudo code of the algorithm to discover the border of the JEP space associated with $\mathcal{D}_p$ and $\mathcal{D}_n$ is as follows.

```
DIFF($<\{\emptyset\}, \{A_1, \cdots, A_{k_1}\}>, <\{\emptyset\}, \{B_1, \cdots, B_{k_2}\}>$)
    ;;  return $<\mathcal{L}, \mathcal{R}>$ such that $[\mathcal{L}, \mathcal{R}] =$
        $[\{\emptyset\}, \{A_1, \cdots, A_{k_1}\}] - [\{\emptyset\}, \{B_1, \cdots, B_{k_2}\}]$
    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($<\{\emptyset\}, \{A_j\}>$,
              $<\{\emptyset\}, \{B_1, \cdots, B_{k_2}\}>$);
    5)    $\mathcal{R} = \mathcal{R} \cup$ right bound of border;
    6)    $\mathcal{L} = \mathcal{L} \cup$ left bound of border;
    7)  return $<\mathcal{L}, \mathcal{R}>$;
```

The correctness of this algorithm is obvious according to the proof for border union of two JEP spaces. The subroutine BORDER-DIFF is detailed in (Dong & Li, 1999). Its code is optimized in this paper.

4. Handling Other Maintenance Problems

In addition to the insertion of new instances, many other changes such as *insertion of new items*, *deletion of items*, and *deletion of instances* may happen to a given set of instances. The algorithms for maintaining JEP spaces in response to these cases are presented here.

4.1 Insertion of a New Item e

Following the insertion of a new item e into the database we denote the new states of the original data sets $\mathcal{D}_p$ and $\mathcal{D}_n$ as $new\mathcal{D}_p$ and $new\mathcal{D}_n$. Because of the insertion of the item e, the non-zero support item sets in $new\mathcal{D}_p$ can be denoted $[\{\emptyset\}, \mathcal{R}_p] \cup [\{\{e\}\}, \mathcal{R}'_p]$, where $[\{\emptyset\}, \mathcal{R}_p]$ is the horizontal space of $\mathcal{D}_p$, $<\{\{e\}\}, \mathcal{R}'_p>$ is the border of the collection of the item sets in $new\mathcal{D}_p$ containing e. Similarly, the non-zero support item sets in $new\mathcal{D}_n$ can be denoted $[\{\emptyset\}, \mathcal{R}_n] \cup [\{\{e\}\}, \mathcal{R}'_n]$, where $[\{\emptyset\}, \mathcal{R}_n]$ is the horizontal space of $\mathcal{D}_n$, $<\{\{e\}\}, \mathcal{R}'_n>$ is the border of the collection of the item sets in $new\mathcal{D}_n$ containing e. Therefore, the JEP space with respect to $new\mathcal{D}_p$ and $new\mathcal{D}_n$ is the following set

$$\begin{aligned} & ([\{\emptyset\}, \mathcal{R}_p] \cup [\{\{e\}\}, \mathcal{R}'_p]) - ([\{\emptyset\}, \mathcal{R}_n] \cup [\{\{e\}\}, \mathcal{R}'_n]) \\ = \; & ([\{\emptyset\}, \mathcal{R}_p] - [\{\{e\}\}, \mathcal{R}'_n] - [\{\emptyset\}, \mathcal{R}_n]) \cup \\ & ([\{\{e\}\}, \mathcal{R}'_p] - [\{\emptyset\}, \mathcal{R}_n] - [\{\{e\}\}, \mathcal{R}'_n]) \\ = \; & ([\{\emptyset\}, \mathcal{R}_p] - [\{\emptyset\}, \mathcal{R}_n]) \cup ([\{\{e\}\}, \mathcal{R}'_p] - [\{\{e\}\}, \mathcal{R}'_n]) \end{aligned}$$

Obviously, the maintenance algorithm in response to the insertion of a new item consists of the following two steps:

1. Take border difference operation to discover the border of $[\{\{e\}\}, \mathcal{R}'_p] - [\{\{e\}\}, \mathcal{R}'_n]$;
2. Take border union operation to discover the border of $([\{\emptyset\}, \mathcal{R}_p] - [\{\emptyset\}, \mathcal{R}_n]) \cup ([\{\{e\}\}, \mathcal{R}'_p] - [\{\{e\}\}, \mathcal{R}'_n])$.

4.2 Deletion of an Item e

Following the removal of an item e from the database, the state of the original data sets $\mathcal{D}_p$ and $\mathcal{D}_n$ are denoted as $new\mathcal{D}_p$ and $new\mathcal{D}_n$. Two borders $<\{\{e\}\}, \mathcal{R}'_p>$ and $<\{\{e\}\}, \mathcal{R}'_n>$ are used respectively to represent those item sets that are lost in $\mathcal{D}_p$ and in $\mathcal{D}_n$ due to the removal. Therefore, the JEP space with respect to $new\mathcal{D}_p$ and $new\mathcal{D}_n$ is the following set

$$\begin{aligned} & ([\{\emptyset\}, \mathcal{R}_p] - [\{\{e\}\}, \mathcal{R}'_p]) - ([\{\emptyset\}, \mathcal{R}_n] - [\{\{e\}\}, \mathcal{R}'_n]) \\ = \; & ([\{\emptyset\}, \mathcal{R}_p] - [\{\{e\}\}, \mathcal{R}'_p] - [\{\emptyset\}, \mathcal{R}_n]) \cup \\ & (([\{\emptyset\}, \mathcal{R}_p] - [\{\{e\}\}, \mathcal{R}'_p]) \cap [\{\{e\}\}, \mathcal{R}'_n]) \\ = \; & [\{\emptyset\}, \mathcal{R}_p] - [\{\emptyset\}, \mathcal{R}_n] - [\{\{e\}\}, \mathcal{R}'_p] \end{aligned}$$

This highlights the fact that any previously discovered JEPs must be removed if they contain the item e. Procedurely, this can be done efficiently, assuming the border of $[\{\emptyset\}, \mathcal{R}_p] - [\{\emptyset\}, \mathcal{R}_n]$ is $<\mathcal{L}_1, \mathcal{R}_1>$, by

- Removing those item sets in $\mathcal{L}_1$ which contain e;
- Removing item e from those item sets in $\mathcal{R}_1$ which contain e;
- $\mathcal{R}_1 \leftarrow$ the set of the most specific elements in $\mathcal{R}_1$.

Therefore, the current $<\mathcal{L}_1, \mathcal{R}_1>$ is the border of the JEP space with respect to $new\mathcal{D}_p$ and $new\mathcal{D}_n$.

4.3 Deletion of Instances

Suppose a set Δ_p of positive instances are removed from the original positive data set $\mathcal{D}_p$ and the original negative data set $\mathcal{D}_n$ remains unchanged, then the JEP space with respect to the original $\mathcal{D}_p$ and $\mathcal{D}_n$ will be reduced by taking away all elements occurring in Δ_p. Procedurely, this can be done efficiently, assuming the border of the original JEP space is $<\mathcal{L}_1, \mathcal{R}_1>$, by

- Discovering the horizontal border of $(\mathcal{D}_p - \Delta_p)$ and denote this border as $<\{\emptyset\}, \mathcal{R}'_p>$;
- Removing those elements C in $\mathcal{L}_1$ such that no elements in $\mathcal{R}'_p$ contain C.

Therefore, $<\mathcal{L}_1, \mathcal{R}'_p>$ is the border of the JEP space with respect to $(\mathcal{D}_p - \Delta_p)$ and $\mathcal{D}_n$.

The problem of how to efficiently maintain a JEP space when a set Δ_n of negative instances are removed is as yet unsolved. A naive maintenance method is to take a border difference operation to discover the border of the JEP space with respect to $\mathcal{D}_p$ and $(\mathcal{D}_n - \Delta_n)$.

Table 1. Details about four data sets.

DATA SETS	#INSTANCES	#ATTRI	#ITEMS
MUSHROOM	4208(+), 3916 (-)	22	125
PIMA	268(+), 500 (-)	8	17
TIC-TAC-TOE	626(+), 332 (-)	9	27
NURSERY	4320(+), 8640(-)	8	27

5. Experimental Results

We choose four data sets in UCI repository (Blake & Murphy, 1998) to experimentally examine the maintenance algorithms, especially their efficiency. These data sets are mushroom, pima, tic-tac-toe, and nursery. More details can be seen in Table 1.
Note that the continuous attributes in the pima data set are discretized by MLC++ techniques (Kohavi et al, 1994). An interesting thing is that this discretization method compressed some different instance points into one point in the pima data set. The original nursery data set has five classes. Here, we consider class `not_recom` as positive class and the remaining four classes all as negative.

We would like to point out that the 10-fold average testing accuracies achieved by JEP classifier for the mushroom, pima, tic-tac-toe, and nursery data sets are very high, which are 100%, 79.6%, 99.06%, and 98.96% respectively, with comparison respectively to 99.8%, 75.5%, and 98.6% achieved by C4. 5. The accuracy of nursery by C4.5 was not available. If version spaces were used for classification, the version spaces for the four data sets would contain no elements. Then, the classification problem would fall into a real dilemma.

We first examine the efficiency of the maintenance algorithms in response to insertion of new *positive* instances. The experimental steps are as follows.

1. Divide data set $\mathcal{D}$ (mushroom, pima, or tic-tac-toe) into $\mathcal{D}_p^{whole}$ and $\mathcal{D}_n^{whole}$ containing all positive and negative instances respectively;

2. Partition $\mathcal{D}_p^{whole}$ into $k (k = 20, 10, \text{or } 5)$ number of *parts*, denoted $\mathcal{D}_p^1, \mathcal{D}_p^2, \cdots, \mathcal{D}_p^k$;

3. Using border difference operation, discover the border $\mathcal{B}_1$ of the JEP space with respect to $\mathcal{D}_p$ and $\mathcal{D}_n^{whole}$, where $\mathcal{D}_p$ consists of any $k-1$ parts in $\mathcal{D}_p^{whole}$. View the derived border $\mathcal{B}_1$ as the *old* border from the previously processed data;

4. Take the remaining part in $\mathcal{D}_p^{whole}$ as Δ_p, a set of new positive instances, and then discover the border $\mathcal{B}_2$ of the JEP space with respect to Δ_p and $\mathcal{D}_n^{whole}$;

5. Take border union operation on $\mathcal{B}_1$ and $\mathcal{B}_2$ to discover the border $\mathcal{B}_3$ of the JEP space with respect to $\mathcal{D}_p^{whole}$ and $\mathcal{D}_n^{whole}$.

Table 2. Time comparison in mushroom data set.

CASES IN MUSHROOM	MAINTAINING TIME			NAIVE TIME (SEC.)
	$k=20$	$k=10$	$k=5$	
INSERTING Δ_p	141.8	281.4	596.1	3360.2
INSERTING Δ_n	243.1	280.7	300.2	3360.2

Table 3. Time comparison in pima data set.

CASES IN PIMA	MAINTAINING TIME (SEC.)			NAIVE TIME (SEC.)
	$k=20$	$k=10$	$k=5$	
INSERTING Δ_p	0.05	0.11	0.31	1.24
INSERTING Δ_n	0.20	0.23	0.25	1.24

For a fixed k, the time required by the maintenance algorithms is summed over the time spent on steps 4 and 5. The time is averaged over k cases of insertions where each case of insertion adds one part $\mathcal{D}_p^i$ as new instances to the old instances, $\mathcal{D}_p^{whole} - \mathcal{D}_p^i$. For comparison to a naive recomputation method, the time to discover directly, using border difference operation, the border $\mathcal{B}_3$ of the JEP space with respect to $\mathcal{D}_p^{whole}$ and $\mathcal{D}_n^{whole}$ is also required.

Secondly, we examine the efficiency of the maintenance algorithm in response to insertion of new *negative* instances. The experimental procedure is actually a dual 5-step procedure to the one as discussed above, by substituting p with n, n with p, *positive* with *negative*, *negative* with *positive*, and *border union* with *border intersection*.

The CPU time of the experiments are shown in Table 2, Table 3, and Table 4 when varying k and data sets. These experiments were carried out on a 500Mhz PentiumIII (running Linux) with 512M bytes of RAM.

In Table 2, Table 3, and Table 4, the first column shows the cases for handling insertion of new positive instances or insertion of new negative instances. The middle three columns show the time spent by our incremental maintenance algorithms. The fifth column is the time spent by the naive recomputation method. Two important observations are: **(i)** For handling insertion of Δ_p, the time spent by incremental maintenance algorithms is approximately $\frac{1}{k}*$ `naive time`. So, our algorithms are highly efficient, especially when a small number of new positive instances are inserted (e.g., $k = 20$). **(ii)** For handling insertion of Δ_n, the efficiency of the incremental algorithms varies among the three data sets. The efficiency is very high in the mushroom and pima data sets. However, it is not obvious in tic-tac-toe data set. The efficiency mainly depends on how large portion of the elements in the previously discovered left bound ($\mathcal{L}_1$) contain the elements of the left bound ($\mathcal{L}_2$)

Table 4. Time comparison in tic-tac-toe data set.

CASES IN TIC-TAC-TOE	MAINTAINING TIME (SEC.)			NAIVE TIME (SEC.)
	$k=20$	$k=10$	$k=5$	
INSERTING Δ_p	0.85	1.74	3.58	18.28
INSERTING Δ_n	13.99	17.20	17.50	18.28

Table 5. Bound size change in mushroom.

NEW INSTANCES	SIZE OF $\mathcal{B}_1$	SIZE OF $\mathcal{B}_2$	SIZE OF $\mathcal{B}_3$
POSITIVE	$\|\mathcal{L}_1\| = 1606$ $\|\mathcal{R}_1\| = 3787$	$\|\mathcal{L}_2\| = 704$ $\|\mathcal{R}_2\| = 421$	$\|\mathcal{L}_3\| = 1606$ $\|\mathcal{R}_3\| = 4208$
NEGATIVE	$\|\mathcal{L}_1\| = 1602$ $\|\mathcal{R}_1\| = 4208$	$\|\mathcal{L}_2\| = 462$ $\|\mathcal{R}_2\| = 4208$	$\|\mathcal{L}_3\| = 1606$ $\|\mathcal{R}_3\| = 4208$

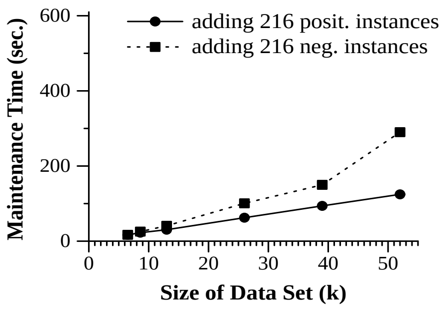

Figure 1. Dependency of maintenance time on the data set size.

associated with the incremental data.

We also carried experiments to see the trend of the bound sizes from the old border $\mathcal{B}_1$ to the current border $\mathcal{B}_3$ via the incremental border $\mathcal{B}_2$. Table 5 shows the sizes of the bounds when handling insertion of new positive instances and handling insertion of new negative instances in the mushroom data, where $k = 10$ and Δ_p (or Δ_n) is the first part of $\mathcal{D}_p^{whole}$ (or $\mathcal{D}_n^{whole}$).

Finally, we run our programs to see the trend of maintenance time when the size of the training data varies. We varied the size of the nursery data set from 6480 to 8640, 12960, 25920, 38880, and 51840 and fixed $|\Delta_p|$ as 216, and $|\Delta_n|$ as 216. (Note that the data set size was increased from 12960 to 25920, 38880, and 51840 by adding some instances constructed by modifying values of the original 12960 instances.) For fixed size of new positive instances, Figure 1 shows the linear scalability of the maintenance algorithms when the training data size increases. Also, the naive method, as expected, is approximately $\frac{DataSetSize}{216}$ times slower than the incremental method. The incremental maintenance time for new negative data, although not linearly scalable, is on average only $\frac{1}{15}$ of the time needed by the naive scheme.

6. Conclusion

This paper has introduced the concept of JEP space and presented efficient algorithms for incrementally maintaining JEP spaces in response to a wide range of change to the previously processed data. The experimental results have confirmed the efficiency and the correctness of the algorithms. As JEPs can be used to build accurate classifiers, the presented algorithms and results are important to the problem of classification because modification to the old data happens frequently in practice. Experimental results of the algorithms proposed in Section 4 in relation to adding/deleting attributes will be reported elsewhere.

References

Blake, C. L., & Murphy, P. M. (1998). 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.

Dong, G., & Li, J. (1999). Efficient mining of emerging patterns: discovering trends and differences. *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). Classification by aggregating emerging patterns. *Proceedings of the Second International Conference on Discovery Science* (pp. 30–42). Tokyo, Japan: Springer-Verlag.

Gunter, C. A., Ngair, T.-H., & Subramanian, D. (1997). The common order-theoretic structure of version spaces and ATMS's. *Artificial Intelligence, 95*, 357–407.

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

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

Li, J., Dong, G., & Ramamohanarao, K. (in press). Making use of the most expressive jumping emerging patterns for classification. *Proceedings of the Fourth Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Kyoto, Japan: Springer-Verlag.

Mitchell, T. M. (1977). Version spaces: a candidate elimination approach to rule learning. *Proceedings of the Fifth International Joint Conference on Artificial Intelligence* (pp. 305–310). Cambridge, MA: AAAI Press.

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

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

Schlimmer, J.C., & Fisher, D. (1986). A case study of incremental concept induction. *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 496–501). Philadelphia, PA: AAAI Press.

Utgoff, P.E., Berkman, N.C., & Clouse, J.A. (1997). Decision tree induction based on efficient tree restructuring. *Machine Learning, 29*, 5–44.