Diverging Patterns: Discovering Significant Dissimilarities in Large Databases

Qian Wan and Aijun An

Technical Report CSE-2008-10

December 22, 2008

Department of Computer Science and Engineering

4700 Keele Street Toronto, Ontario M3J 1P3 Canada

Diverging Patterns: Discovering Significant Dissimilarities in Large Databases

Qian Wan* Aijun An†

Abstract

The problem of finding contrast patterns has recently attracted much attention. As a result, a number of promising methods have been proposed to capture significant differences or changes between two or more datasets. Such differences can be captured by emerging patterns and some other types of contrasts. In this paper, we present a framework for mining *diverging patterns*, a new type of contrast patterns whose frequency changes in different directions in two data sets, e.g., it changes from a relatively low to a relatively high value in one dataset, but from high to low in the other. In this framework, a measure called *diverging ratio* is used to discover diverging patterns. We use a two-dimensional vector to represent a pattern, and define the pattern's diverging ratio based on the angular difference between its vectors in two datasets. An algorithm is proposed to mine diverging patterns from a pair of datasets, which makes use of a standard frequent pattern mining algorithm to compute relevant vectors efficiently. We demonstrate the usefulness of our approach on some real-world datasets, showing that the method can reveal novel and interesting knowledge from large databases.

Keywords: Diverging patterns, Contrast patterns, Frequent patterns.

1 Introduction

The problem of finding contrast patterns has recently attracted much attention. Contrast patterns are those patterns that capture important and significant differences or changes between two or more sets of data in the same domain. A number of promising methods have been proposed to capture such differences, including emerging pattern mining [6, 2], contrast set mining [4], and some other methods for mining complex types of contrasts [11, 16, 12]. Contrast patterns are useful for identifying distinguishing characteristics of data and can also be used to build powerful classifiers [10, 7].

The support-confidence framework [1] is the most common measure used in mining interesting patterns, including contrast patterns, due to its anti-monotonicity that effectively simplifies the search lattice. For instance, in emerging pattern mining, the significance of differences is measured by the magnitude of the support-change-ratio (called growth rate [6]) of the patterns over one data set to another. The larger the support-change-ratio, the more different and important the patterns are. However, this is not always the case. Let's see the following examples.

Table 1: Simple gene expression data sets

	normal				*disease*			
	G_1	G_2	G_3	G_4	G_1	G_2	G_3	G_4
$t01$	1	0	1	0	1	1	1	0
$t02$	0	1	1	1	0	1	1	1
$t03$	1	1	1	0	0	1	1	0
$t04$	1	1	1	1	1	0	0	1
$t05$	1	1	0	1	1	1	1	1
$t06$	1	1	1	1	1	0	0	1
$t07$	0	1	0	1	1	1	1	1
$t08$	1	1	0	1	1	1	1	0
$t09$	1	1	0	0	1	0	1	1
$t10$	1	1	1	1	0	0	1	0
$t11$	0	1	1	0	0	1	1	1
$t12$	1	1	1	1	1	1	0	1
$t13$	1	1	0	0	1	1	1	1
$t14$	1	1	1	1	0	1	0	1
$t15$	1	0	0	1	1	1	1	1
$t16$	1	1	0	0	1	1	0	0
$t17$	0	1	1	0	1	1	1	0
$t18$	1	0	0	1	1	1	1	1
$t19$	0	1	1	0	1	0	1	1
$t20$	1	0	0	1	1	1	1	0

Example 1. Microarrays have already been extensively used in bioinformatics to address a wide variety of problems. One of the most attractive applications of microarray technology is the study of the differential gene expression in diseases. There are many genetic diseases that are the result of mutations in a gene of a set of genes. For example, microarray-based gene expression profiling can be used to identify disease genes by comparing gene expression in diseased and normal cells. Suppose we have a pair of simplified gene-time data sets, *normal* and *disease*, which record the expression levels of four genes $(G_1, ..., G_4)$ in normal and diseased tissues during important biological processes over a series of time-stamps $(t_{01}, ..., t_{20})$, as shown in Table 1. A row represents the expression levels of all genes measured at

*York University, Toronto, Canada.
†York University, Toronto, Canada.

one time-stamp, and a column the expression levels of a single gene. Each cell represents the expression level of a certain gene at a certain time-stamp, whose value is represented by 0 or 1. If a gene is up or down regulated at a specific time-stamp, the corresponding element takes a value of 1; otherwise, the value is 0.

Let us consider the gene set $\{G_1, G_2\}$. Because its frequency is exactly the same ($\frac{11}{20} = 0.55$) in both data sets, it cannot be considered as a valid emerging pattern, or any other contrast patterns based on the support-confidence framework. However, interesting differences of the gene set between these two data sets can be found after we take into account the time information in the data sets. We can easily see that before (and including) $t10$ in the *normal* data set, $\{G_1, G_2\}$ appears 7 times; but after $t10$, $\{G_1, G_2\}$ only occurs 4 times. That is to say that the frequency of $\{G_1, G_2\}$ in *normal* decreases significantly from 70% to 40% after $t10$. On the contrary, the frequency of $\{G_1, G_2\}$ in *disease* increases significantly from 40% to 70%.

Example 2. Consider two groups of patients, A and B, that have different demographic backgrounds. Both groups of patients have used a new drug for a chronic disease for a period of time, during which they have been tested for their blood level of *alkaline phosphatase (ALP)* that is an enzyme produced by liver (and other) cells. Elevated blood levels of this substance may indicate abnormal function of the liver. Suppose that for the observed period of time high ALP levels occur frequently in both groups of patients and their frequencies are roughly equal in the two groups. This indicates that the new drug has a side effect of increasing ALP levels in both groups of patients. However, a closer examination of the blood test results reveals that for group A the frequency of high ALP levels decreases significantly over the observed period of time, but the frequency for group B increases significantly in the same period of time. This indicates that group A patients can tolerate the side effect of the new drug better than group B patients in a long run although at the beginning of taking this drug they showed a more severe side effect than group B.

These observations assert that a pattern might have significantly different distributions between two data sets, even though it has a subtle frequency-difference in the data sets. This motivates our research in this paper to identify a new class of contrast patterns to capture their significant dissimilarities between the data sets, especially those patterns whose frequency changes in opposite directions, i.e., it changes from a relatively low to a relatively high value in one data set, but from high to low in the other.

The main contributions of this paper are as follows:

- We propose a framework for mining *diverging patterns*, a new type of contrast patterns whose frequency changes in different directions in a pair of data sets;
- We define a measure, called *diverging ratio*, to capture the frequency-change-difference of a pattern in two data sets. The measure uses a two-dimensional vector to represent a pattern in a data set and is defined based on the angular difference between its vectors in two data sets;
- An algorithm is proposed to mine diverging patterns from a pair of data sets. The algorithm makes use of a standard frequent pattern mining algorithm to compute relevant vectors efficiently;
- We perform extensive experiments on several real-world data sets to verify the effectiveness and usefulness of diverging patterns. Our results illustrate that mining diverging patterns is highly promising as a practical approach to discovering novel and interesting knowledge in real-world domains.

The rest of the paper is organized as follows. Section 2 presents basic concepts and notations. The formal definition of diverging patterns and an algorithm for mining the set of diverging patterns from a pair of data sets are given in Section 3 and Section 4, respectively. A discussion is given in Section 5. We provide extensive experimental evaluations of our techniques in Section 6. We review related work in Section 7 and conclude this paper in Section 8.

2 Preliminaries and Notations

Mining frequent patterns is one of the fundamental operations in data mining applications for extracting interesting patterns from databases (e.g. association rules, correlations, sequences, episodes, classifiers, clusters). In this section, we briefly review the basic concepts of frequent pattern mining. Table 2 summarizes the notations that will be used throughout this paper and their meanings.

Let $\mathcal{I} = \{i_1, i_2, \ldots, i_m\}$ be a set of m distinct *items*. A subset $X \subseteq \mathcal{I}$ is called an *itemset* or a *pattern*. A k-itemset is an itemset that contains k items. In this paper, we use AB to represent pattern $\{A, B\}$, where $A \in \mathcal{I}$ and $B \in \mathcal{I}$, for simplicity. A transaction over $\mathcal{I}$ is a couple $\mathcal{T} = (tid, I)$ where tid is the transaction identifier (or time-stamp) and $I \subseteq \mathcal{I}$ is an itemset. A transaction $\mathcal{T} = \langle tid\ ,\ I \rangle$ is said to support an itemset $X \subseteq \mathcal{I}$, if and only if $X \subseteq I$. A transaction database $\mathcal{D}$ over $\mathcal{I}$ is a set of transactions over $\mathcal{I}$.

The *cover* of an itemset X in $\mathcal{D}$, denoted as $cov(X, \mathcal{D})$, consists of the set of transactions in $\mathcal{D}$ that support

Table 2: Notations used in this paper

$\mathcal{D}$	a database of transactions
$\|\|\mathcal{S}\|\|$	the cardinality of set $\mathcal{S}$
$cov(X, \mathcal{D})$	the *cover* of a pattern X in $\mathcal{D}$
$sup(X, \mathcal{D})$	the *support* of a pattern X in $\mathcal{D}$
$\rho(\mathcal{T}, \mathcal{D})$	The *position* of a transaction $\mathcal{T}$ in $\mathcal{D}$
$\delta(\mathcal{T}, \mathcal{D})$	the *central distance* of a transaction $\mathcal{T}$ in $\mathcal{D}$
$\omega^-(X, \mathcal{D})$	the *negative weight* of a pattern X in $\mathcal{D}$
$\omega^+(X, \mathcal{D})$	the *positive weight* of a pattern X in $\mathcal{D}$
$\overrightarrow{\omega}(X, \mathcal{D})$	the *weight* of a pattern X in $\mathcal{D}$
$div(X)$	the *diverging ratio* of a pattern X
$sup^i(X, \mathcal{D})$	the i^{th} *support* of a pattern X in $\mathcal{D}$

X.

$$cov(X, \mathcal{D}) := \{\langle tid, I\rangle \mid \langle tid, I\rangle \in \mathcal{D}, X \subseteq I\}.$$

An itemset X in a transaction database $\mathcal{D}$ has a *support*, denoted as $sup(X, \mathcal{D})$, which is the ratio of transactions in $\mathcal{D}$ containing X. That is, $sup(X, \mathcal{D}) = \frac{\|cov(X,\mathcal{D})\|}{||\mathcal{D}||}$, where $\|\mathcal{S}\|$ is the cardinality of set $\mathcal{S}$. Given a transaction database $\mathcal{D}$ and a user specified minimum support threshold *min_sup*, an itemset X is called a frequent itemset or frequent pattern in $\mathcal{D}$ if $sup(X, \mathcal{D}) \geq min\_sup$. The problem of mining frequent patterns is to find the complete set of frequent patterns in a transaction database with respect to a given support threshold.

3 Diverging Pattern

In this section we define several concepts relevant to our proposed framework.

3.1 Definition of weight.

DEFINITION 3.1. *Assuming that the transactions in a transaction database $\mathcal{D}$ are ordered by their transaction identifiers (or time-stamps), the* position *of a transaction $\mathcal{T}$ in $\mathcal{D}$, denoted as $\rho(\mathcal{T}, \mathcal{D})$, is the number of transactions whose transaction identifier (or time-stamp) is less than or equal to that of $\mathcal{T}$. Thus, $1 \leq \rho(\mathcal{T}, \mathcal{D}) \leq \|\mathcal{D}\|$.*

DEFINITION 3.2. *The* central distance *of a transaction $\mathcal{T}$ in $\mathcal{D}$, denoted as $\delta(\mathcal{T}, \mathcal{D})$, is the distance from the transaction to the central position of $\mathcal{D}$:*

$$\delta(\mathcal{T}, \mathcal{D}) := [\rho(\mathcal{T}, \mathcal{D}) - \frac{\|\mathcal{D}\| + 1}{2}], \tag{3.1}$$

where $[x]$ is a round *function of a real number x in which a positive x is converted to the smallest integer that is greater than x but a negative x is converted to the biggest integer that is smaller than x.*[1]

[1]The reason we use a round function is to keep the unit (which is 1) of distance from a transaction to the central position to be the same for the data sets with either even or odd number of transactions.

The central distance of a transaction $\mathcal{T}$ is negative if the position of $\mathcal{T}$ is before the central position of the database; and it is positive if the position is after the central position. For example, the central position of the *normal* or *disease* database is between transactions $t10$ and $t11$. Thus, transaction $t10$ is the first transaction before the central position. According to the above definition, its central distance is -1. Similarly, the central position of transaction $t12$ is +2.

DEFINITION 3.3. *The* negative weight *of a pattern X in $\mathcal{D}$, denoted as $\omega^-(X, \mathcal{D})$, is the ratio of the sum of all negative central distances of transactions in $cov(X, \mathcal{D})$ to the number of transactions in $\mathcal{D}$:*

$$\omega^-(X, \mathcal{D}) := \frac{\sum_{\mathcal{T}\in cov(X,\mathcal{D})}\{\delta(\mathcal{T}, \mathcal{D}) < 0\}}{\|\mathcal{D}\|}. \tag{3.2}$$

DEFINITION 3.4. *The* positive weight *of a pattern X in $\mathcal{D}$, denoted as $\omega^+(X, \mathcal{D})$, is the ratio of the sum of all positive central distances of transactions in $cov(X, \mathcal{D})$ to the number of transactions in $\mathcal{D}$:*

$$\omega^-(X, \mathcal{D}) := \frac{\sum_{\mathcal{T}\in cov(X,\mathcal{D})}\{\delta(\mathcal{T}, \mathcal{D}) > 0\}}{\|\mathcal{D}\|}. \tag{3.3}$$

DEFINITION 3.5. *The* weight *of a pattern X in $\mathcal{D}$, denoted as $\overrightarrow{\omega}(X, \mathcal{D})$, is defined as a two-dimensional vector of $\omega^-(X, \mathcal{D})$ and $\omega^+(X, \mathcal{D})$:*

$$\overrightarrow{\omega}(X, \mathcal{D}) :=< \omega^-(X, \mathcal{D}),\ \omega^+(X, \mathcal{D}) > . \tag{3.4}$$

Intuitively, the weight of a pattern X in a database $\mathcal{D}$ provides us with some information about the distribution of X in $\mathcal{D}$ by considering the distances of the transactions where X occurs to the central position of $\mathcal{D}$. It can capture the general frequency-changing direction of patter X in database $\mathcal{D}$. [2] For example, $\overrightarrow{\omega}(G_1G_2,\ normal) =< -1.6,\ +0.75 >$ and $\overrightarrow{\omega}(G_1G_2,\ disease) =< -1.0,\ +2.05 >$ indicate that the frequency of pattern G_1G_2 is generally decreasing in the *normal* database, but it is increasing in the *disease* database.

3.2 Definition of diverging ratio. We are interested in finding patterns whose frequency changes in different directions in a pair of databases. For this purpose, we need to measure the difference between the

[2]Note that the weight of a pattern as defined above does not capture the detailed form of the pattern's distribution in the database, e.g., whether the distribution is normal or periodical, but rather the general frequency-changing direction.

weight vector of a pattern in one database and its weight vector in the other.

A simple way to measure the difference between two vectors ($\overrightarrow{\omega}(X, \mathcal{D}_1)$ and $\overrightarrow{\omega}(X, \mathcal{D}_2)$) is to treat each vector as a point in a 2-dimensional space and use the Euclidean distance between the two points to describe the difference between the two vectors, which is computed as

$$\sqrt{(\omega^-(X,\mathcal{D}_2) - \omega^-(X,\mathcal{D}_1))^2 + (\omega^+(X,\mathcal{D}_2) - \omega^+(X,\mathcal{D}_1))^2}.$$

However, the use of the Euclidean distance for our purpose has the following limitation. The Euclidean distance only shows the difference in the value magnitude of the two vectors. If the sizes of the two databases are quite different or if the pattern is much more frequent in one database than it is in the other, the absolute values of the pattern's positive and negative weights in one database can be much larger than the ones in the other database. In this situation, the Euclidean distance between the two vectors can be large no matter whether the frequency of the pattern changes differently or not in the two databases. This situation is illustrated in Figure 1, in which three weight vectors, $\overrightarrow{\omega}(X, \mathcal{D}_1)$, $\overrightarrow{\omega}(X, \mathcal{D}_2)$ and $\overrightarrow{\omega}(X, \mathcal{D}_3)$, are depicted. Obviously, the difference between vectors $\overrightarrow{\omega}(X, \mathcal{D}_1)$ and $\overrightarrow{\omega}(X, \mathcal{D}_3)$ is due to that fact that $\overrightarrow{\omega}(X, \mathcal{D}_1)$ has larger negative and positive weights than $\overrightarrow{\omega}(X, \mathcal{D}_3)$ does, but the ratio between the positive and negative weights in $\overrightarrow{\omega}(X, \mathcal{D}_1)$ is the same as the one in $\overrightarrow{\omega}(X, \mathcal{D}_3)$, indicating X's frequency change is the same in $\mathcal{D}_1$ and $\mathcal{D}_3$. On the other hand, it can be easily seen that the difference between $\overrightarrow{\omega}(X, \mathcal{D}_1)$ and $\overrightarrow{\omega}(X, \mathcal{D}_2)$ is due to the different directions of changes in their positive and negative weights. However, the Euclidean distance between $\overrightarrow{\omega}(X, \mathcal{D}_1)$ and $\overrightarrow{\omega}(X, \mathcal{D}_3)$ are about the same as the distance between $\overrightarrow{\omega}(X, \mathcal{D}_1)$ and $\overrightarrow{\omega}(X, \mathcal{D}_2)$.

Another popular measure of similarity between two vectors is the cosine of the angle θ between two vectors, which is defined as follow:

$$cos(\theta) = \frac{\overrightarrow{\omega}(X,\mathcal{D}_1) \bullet \overrightarrow{\omega}(X,\mathcal{D}_2)}{|\overrightarrow{\omega}(X,\mathcal{D}_1)| \times |\overrightarrow{\omega}(X,\mathcal{D}_2)|},$$

where $u \bullet v$ is the *dot product* of two vectors u and v, and $|v|$ is the *magnitude* of the vector v. Different from the Euclidean distance, the cosine measure is independent of magnitude differences, and it treats both vectors as unit vectors by normalizing them with the magnitudes of the vectors. Thus, it can be used to provide a good measure of divergence between the two weight vectors. According to Definitions 3.3 and 3.4, the angle between the weight vectors of a pattern in any pair of data sets is in the range $[0, \pi/2]$. Thus, the cosine measure ranges from 1 ($cos(0)$=1) for vectors pointing in the same direction to 0 for orthogonal vectors ($cos(\pi/2)$=0). Based on the cosine measure, we define a measure of distance between two weight vectors as follows.

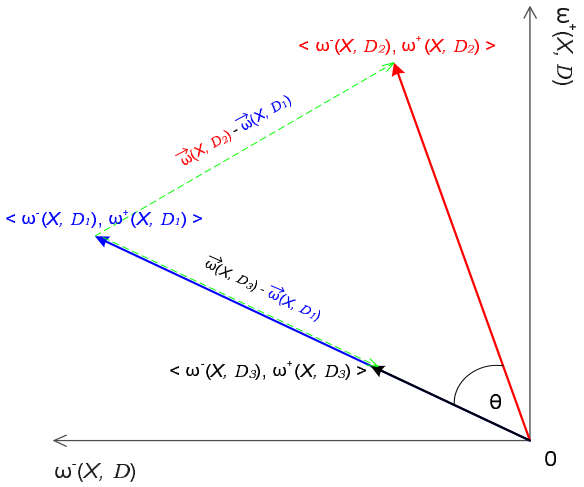


Figure 1: The pattern weight plane

DEFINITION 3.6. *The* diverging ratio *of a pattern X between a pair of data sets $\mathcal{D}_1$ and $\mathcal{D}_2$, denoted as $div(X)$, is defined as:*[3]

$$(3.5) \quad div(X) := 1 - \frac{\overrightarrow{\omega}(X,\mathcal{D}_1) \bullet \overrightarrow{\omega}(X,\mathcal{D}_2)}{|\overrightarrow{\omega}(X,\mathcal{D}_1)| \times |\overrightarrow{\omega}(X,\mathcal{D}_2)|}.$$

where $u \bullet v$ is the dot product *of two vectors u and v, and $|v|$ is the* magnitude *of the vector v.*

For example, the diverging ratio of pattern G_1G_2 between the *normal* and *disease* data sets is $div(G_1G_2)$ = 22.16% ($\theta \approx 38.9^o$). It's easy to see that the diverging ratio is between 0 and 1. The higher the diverging ratio, the bigger the angle between $\overrightarrow{\omega}(X, \mathcal{D}_1)$ and $\overrightarrow{\omega}(X, \mathcal{D}_2)$, and thus the greater the difference of the pattern between the two data sets.

3.3 Definition of diverging pattern.

In this paper, we are interested in finding patterns whose diverging ratio is large, which is defined as follows.

DEFINITION 3.7. *A pattern X is a* Diverging Pattern *(*DP*) with respect to a pair of transaction databases $\mathcal{D}_1$ and $\mathcal{D}_2$, if the following conditions hold:*

(1) $sup(X, \mathcal{D}_1) \geq t_s$ and $sup(X, \mathcal{D}_2) \geq t_s$;

(2) $div(X) \geq t_d$;

where t_s and t_d are called pattern support threshold *and* diverging ratio threshold*, respectively.*

[3]We omit $\mathcal{D}_1$ and $\mathcal{D}_2$ whenever it is clear from the context.

There are several reasons that we only consider frequent itemsets in both data sets, as shown in condition (1). First, the inclusion of infrequent itemsets would much enlarge the size of mining results, which has been deemed unfavorable to the end users. Second, infrequent itemsets are likely to be random noise patterns. Including them may lead to misleading patterns. Third, in finding a diverging pattern, we are more interested in a pattern's different frequency-change directions than its actual frequency-difference between two data sets. Finally, another reason is that the weight vector of every frequent pattern can be obtained efficiently during the frequent pattern mining process, which will be discussed in more detail in the next section.

In practice, t_s should be set to a low value for real data sets, as experienced in frequent pattern mining. Intuitively, t_d can be set to 0.3 to guarantee that the frequency of a discovered pattern changes in different directions in the two databases since $cos(\pi/4) = 0.707$.[4] However, a user may want to lower the t_d value a bit to find more patterns whose frequency changes either in the same direction but the extents of their frequency changes are significantly different in the two databases, or in different directions but the extents of their frequency changes in the two databases are less different from each other than what they would be if t_d was set to 0.3. More discussion on the setting of t_d will be given in Section 6.2.

4 Mining Diverging Patterns

In this section, we present an algorithm, called *DP-mine*, for mining the set of diverging patterns from a pair of data sets. Since a diverging pattern is a frequent pattern in both of the data sets, it is natural to make use of existing well-developed frequent pattern mining algorithms such as Apriori and *FP-growth* in finding diverging patterns. However, since the diverging ratio does not satisfy the anti-monotone property (that is, it does NOT hold that if a pattern satisfies a diverging ratio threshold t_d, all of its subsets also satisfy t_d), it is difficult (if not impossible) to incorporate the diverging ratio constraint (i.e., Condition (2) in Definition 3.7) into the frequent pattern mining process. Nevertheless, it is still possible to make use of a frequent pattern mining algorithm to calculate the positive and negative weights of a pattern at the same time of mining frequent patterns. After the weights of frequent patterns are calculated, the remaining process of finding diverging patterns becomes trivial. Next, we first present the general structure of the *DP-mine* algorithm and then describe how to incorporate the weight calculation into two most popular frequent pattern mining algorithms.

[4]This is because if $div(X) > 0.3$, the angle between the two weight vectors of X in the two databases is greater than 45^o, which guarantees that the absolute value of the negative weight of X is less than its positive weight in one database but it is greater than its positive weight in the other database.

4.1 General structure of *DP-mine*.

The general steps of the *DP-mine* algorithm is presented below.

Algorithm: ***DP-mine*** (***Mine Diverging Patterns*** *from a pair of data sets*)

Input: A pair of data sets ($\mathcal{D}_1$ and $\mathcal{D}_2$), pattern support threshold (t_s) and diverging ratio threshold (t_d)

Output: The set of diverging patterns ($\mathcal{S}_{DP}$)

1: Extract two sets of frequent patterns, $\mathcal{F}_1$ and $\mathcal{F}_2$, from $\mathcal{D}_1$ and $\mathcal{D}_2$, respectively, and their weights using a frequent pattern generation algorithm with $min\_sup = t_s$.
2: $\mathcal{S} \leftarrow \mathcal{F}_1 \bigcap \mathcal{F}_2$
3: $\mathcal{S}_{DP} \leftarrow \emptyset$
4: **for all** $P \in \mathcal{S}$ **do**
5: Compute $div(P)$;
6: **if** $div(P) \geq t_d$ **then**
7: $\mathcal{S}_{DP} \leftarrow \mathcal{S}_{DP} \bigcup \{P\}$
8: **end if**
9: **end for**
10: **return** $\mathcal{S}_{DP}$

There are two major phases in this algorithm. During the first phase (Step 1), all frequent itemsets in the two data sets along with their supports and weights are derived using a standard frequent pattern generation algorithm, such as *Apriori* or *FP-growth*, with t_s as the minimum support threshold. In the second phase (starting from Step 2 to the end), the algorithm finds all the diverging patterns between the data sets based on the set of frequent itemsets.

In Step 2, patterns that are frequent in both $\mathcal{D}_1$ and $\mathcal{D}_2$ are collected in set $\mathcal{S}$. From step 4 to 9, the algorithm computes the diverging ratio of each pattern in $\mathcal{S}$ based on the pattern's weights obtained in step 1. If the pattern's diverging ratio is greater than or equal to t_d, then the pattern is a diverging pattern and is added into the set $\mathcal{S}_{DP}$, which is returned in the last step of the algorithm.

4.2 Computation of positive and negative weights.

Apriori [1] and *FP-growth* [8] are two most popular algorithms for mining frequent patterns. Below we describe how weight computation can be integrated into either of the two algorithms in Step (1) of the *DP-mine* algorithm.

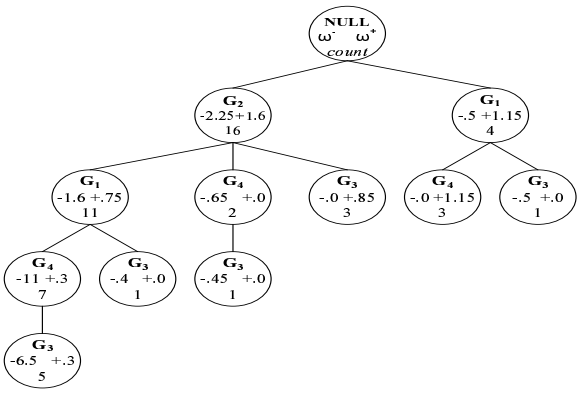


Figure 2: FP-tree with negative and positive weights

Apriori [1] and its variants conduct a bottom-up level-wise search over the itemset lattice. Candidate itemsets with $k+1$ items are only generated from frequent itemsets with k items. In each level, all candidate itemsets are tested for frequency by scanning the database. During the database scan, we can update the positive or negative weight of a candidate at the same time when the count of the candidate needs to be updated, i.e., when the candidate is matched with a transaction, by calculating the central distance of the transaction based on the position of the transaction in the database. Thus, the positive and negative weights of all the frequent itemsets can be obtained.

The *FP-growth* [8] algorithm first constructs a (compressed) FP-tree, which maps each transaction into a path of the tree, while keeping only the frequent items. Each tree node is associated with the count of transactions that contain the items on the path from the root to the node. To incorporate weight computation into the mining process, we also associate each node with a negative weight field and a positive weight field. During the FP-tree building process, when processing a transaction, if the count value of a node is incremented by 1, we also add the central distance of the transaction to the negative weight field of the node if the distance is less than zero; otherwise, it is added to the positive weight field of the node. Thus, after the tree is built, the negative weight field of a node records the sum of the negative central distances of all the transactions represented by the path from the root to the node, and the positive weight field records the sum of positive central distances of the corresponding transactions.

Figure 2 illustrates the modified FP-tree for the *normal* data set with $min\_sup = 25\%$. When building this tree, the frequent items are sorted in frequency-descending order: G_2, G_1, G_4 and G_3. Each node in the tree records two extra values: negative and positive weights. During the recursive mining process of *FP-growth*, whenever there is a need to compute the count of a pattern in a conditional pattern base, the negative and positive weights of the pattern are computed in the same way. For instance, there are only two paths in the tree that contain both G_3 and G_4: G_2-G_1-G_4-G_3 and G_2-G_4-G_3. Therefore, the count value of G_3G_4 is the sum of the count values of the two G_3 nodes on the two paths, i.e., $1+5=6$. Similarly, the negative (positive) weight of G_3G_4 is the sum of negative (positive) weight values of the same nodes, i.e., $\omega^-(G_3G_4,\ normal) = (-0.65) + (-0.45) = -1.1$ and $\omega^+(G_3G_4,\ normal) = (+0.3) + (+0.0) = +0.3$.

4.3 Complexity analysis. The first phase of the *DP-mine* algorithm is dominated by the complexity of finding frequent patterns in the pair of data sets, with extra space to store two weights in each node of the FP-tree if the *FP-growth* algorithm is used. Steps 4-9 can be implemented in $O(nlogn)$, assuming the patterns in $\mathcal{F}_1$ and $\mathcal{F}_2$ are ordered lexicographically and $n = max(\|\mathcal{F}_1\|, \|\mathcal{F}_2\|)$. Therefore, the time complexity of the *DP-mine* algorithm is $F + O(nlogn + n)$, where F is the complexity of the frequent pattern mining algorithm used in the first phase and n is usually much smaller than the number of transactions in either of the data sets. As can be seen, if an efficient algorithm (such as *FP-growth*) is used for frequent pattern mining, *DP-mine* is also efficient since its efficiency is dominated by the frequent pattern mining process.[5] This is the benefit of using existing well-developed frequent pattern algorithms in the design of *DP-mine*.

5 Discussion

We have defined a framework for finding diverging patterns from a pair of databases. This framework is related to but different from *emerging patterns*, first introduced in [6]. An Emerging Pattern (EP) is defined as an itemset X satisfying $growthrate(X) = \frac{sup(X,D_2)}{sup(X,D_1)} \geq g$, where D_1, D_2 are two different data sets and $g > 1$ is called the growth rate threshold.

In our framework, we defined the weight vector of a pattern in a database based on the distances of the pattern's occurrences to the central position of the database. It might appear that diverging patterns defined based on the weight vectors can be found by splitting each of the two databases in half at the central

[5] FP-growth may not be efficient when the database is sparse [13]. In such a case, other efficient frequent pattern mining algorithms, such as H-mine [13], can be used. Incorporating weight computation into these algorithms is possible but beyond the topic of this paper.

position and then finding patterns that are emerging in opposite directions in the two databases. However, this is not true. Let us look at the following example.

Assume that there are two databases (A and B), each with 100 transactions. Suppose that in database A a pattern X appears in the first 8 transactions (whose tids range from 1 to 8) and in the four transactions close to the central position in the second half of the database, whose tids are 55, 60, 65 and 70 respectively. According to Definition 3.4, the weight of X in database A is $< -3.72,\ 0.50 >$. Now let us consider database B. Suppose that in database B pattern X appears in 16 transactions right before the central position, whose tids are 35, 36, ..., 50 respectively, and in the 8 transactions near the end of the database, whose tids are 86, 88, 90, 92, 94, 96, 98 and 100 respectively. Thus, the weight of X in database B is $< -1.36,\ 3.44 >$. Obviously, pattern X changes in different directions in A and B. In database A, X is frequent at the beginning and in the middle, but it is infrequent at other places including the end. But in database B, X is infrequent at the beginning, but frequent in the middle and at the end. This difference can be well captured by our framework because the diverging ratio of X in A and B is 0.5113 according to Definition 3.5, which is quite high.

However, this difference cannot be captured by splitting each database in half at the central position and finding the patterns that are emerging in opposite directions in the two databases. This is because in database A the frequency of X changes from 8 to 4 with respect to the central position and in database B it changes from 16 to 8, resulting in the same growth rate in both databases. This example indicates that simply applying emerging pattern mining to each of the databases as described above cannot find all the diverging patterns.

This example also illustrates that although the weights used in our framework are defined based on the distances of the pattern to the central position of a database, the framework can still find the significant frequency-changing difference of a pattern in two databases even when the pattern's frequency change with respect to the central position is the same in the two databases.

6 Experimental Evaluation

In this section we present our empirical results to demonstrate the effectiveness and usefulness of diverging patterns and the scalability of *DP-mine*.

6.1 Real-world data sets. We conducted experiments on several real-world data sets with different characteristics, as shown in Table 3. The first three data sets[6] were contributed by Blue Martini Software as the KDD Cup 2000 data [17]. *BMS-WebView-1* and *BMS-WebView-2* contain several months worth of click-stream data from two e-commerce web sites. *BMS-POS* contains several years worth of point-of-sale data from a large electronics retailer. Data in *MSweb*[7] was obtained from UCI Machine Learning Repository. It records the use of www.microsoft.com by 38000 anonymous, randomly-selected users. For each user, the data lists all the areas of the web site that the user visited in one week. *Retail*[8] is taken from the Frequent Itemset Mining Implementations Repository, it contains the (anonymized) retail market basket data from an anonymous Belgian retail store [5]. The *Livelink* data set was first used in [9] to discover interesting association rules from *Livelink*[9] web log data. The log files contain *Livelink* access data for a period of two months. For each data set except *Livelink*, we randomly select a split point between the 30th and 70th percentiles of the data to partition the data set into two disjoined subsets. For example, *BMS-WebView-1* is partitioned into A_1 and A_2, where A_1 contains the first 18,451 transactions of *BMS-WebView-1* and A_2 contains the last 41,151 transactions. For the *Livelink* data set, two subsets L_1 and L_2 are click streams from two groups of *Livelink* users. Column 3 of Table 3 shows the sizes of subsets for each data set.

Table 3: Characteristics of data sets

data set	# of items	# of trans	# of *FP*s	# of joined FPs	# of *DP*s
A_1	497	18451	906	555	50
A_2		41151	854		
B_1	3340	44760	9235	1322	24
B_2		32752	4794		
C_1	1657	223957	27791	24414	8818
C_2		291640	27957		
M_1	294	19096	1731	1554	1
M_2		13615	1837		
R_1	16470	36251	3032	1882	62
R_2		51911	2803		
L_1	38679	16041	29794	12101	11888
L_2		14545	28130		

$t_s = 0.2\%$ $t_d = 0.3$

A = *BMS-WebView-1* M = *MSweb*

B = *BMS-WebView-2* R = *Retail*

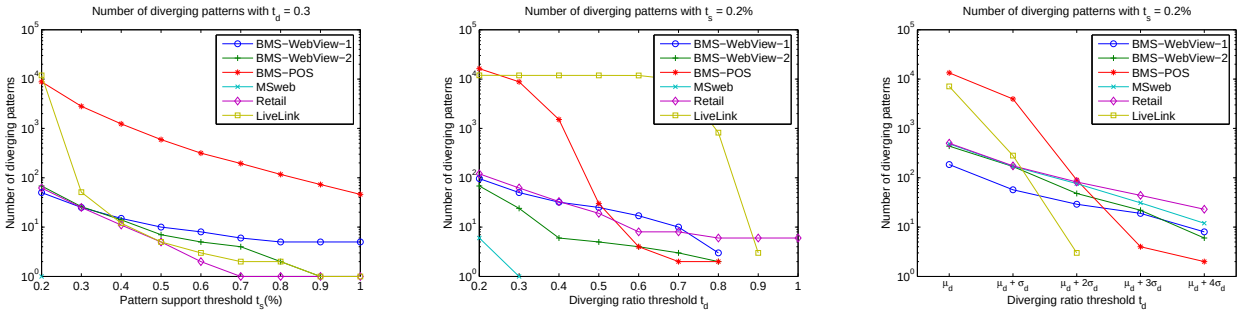
C = *BMS-POS* L = *Livelink*

[6] http://www.ecn.purdue.edu/KDDCUP/data/

[7] http://kdd.ics.uci.edu/databases/msweb/msweb.html

[8] http://fimi.cs.helsinki.fi/data/retail.dat

[9] *Livelink* is a web-based enterprise content management product of Open Text Corporation.



(a) Number of DPs with different t_s.

(b) Number of DPs with different t_d.

(c) Number of DPs with $t_d = \mu_d + n \cdot \sigma_d$.

Figure 3: Number of Diverging Patterns with different t_s and t_d.

6.2 Number of discovered patterns. The last three columns of Table 3 list the number of frequent patterns in each subset, the number of patterns that are freqent in both subsets, and the number of diverging patterns generated from each pair of data sets, with $t_s = 0.2\%$ and $t_d = 0.3$, respectively. As can be seen, when the pattern support threshold is low, a huge number of frequent patterns can be generated from these data sets. In contrast, the number of diverging patterns discovered by our approach is generally much smaller except in the *Livelink* data set. Note that the number of diverging patterns discovered from a pair of data sets depends on the underlying distributions of patterns in the two data sets. The great number of diverging patterns in the *Livelink* data set indicates that the viewing behaviors of the two groups of users are actually quite different even though they both frequently view a large set of information objects in common. Our approach enables us to identify those frequent patterns that are actually different between the two data sets but cannot be distinguished by frequent pattern mining.

Figure 3(a) shows the numbers of diverging patterns discovered in these data sets, with $t_d = 0.3$ and t_s varying from 0.1% to 1.0%. Figure 3(b) illustrates the numbers of diverging patterns, with $t_s = 0.2\%$ and t_d varying from 0.1 to 1.0. As expected, at any fixed diverging ratio threshold (or pattern support threshold), the number of generated diverging patterns decreases with the increase of the pattern support threshold (or diverging ratio threshold). We also used $\mu_d + n \cdot \sigma_d$ ($0 \leq n \leq 4$) as the diverging ratio threshold, where μ_d and σ_d are the the means and standard deviations of the diverging ratios in the joint set of frequent patterns generated from the two data sets, respectively. Figure 3(c) shows the numbers of diverging patterns discovered in each data set, using such a threshold.

In practice, analysis may start with a high diverging ratio threshold (for example, $t_d = \mu_d + 2\sigma_d$) at very small support threshold, to effectively extract strong diverging patterns; and then gradually reduce the diverging ratio threshold (to $\mu_d + \sigma_d$ or $t_d = \mu_d$) to obtain more diverging patterns that may be interesting to the end users.

6.3 Visualization and significance test. In this subsection, we exploit the empirical distribution function (or empirical c.d.f) used in the Kolmogorov-Smirnov test [14] (also called the K-S test[10]) in statistics to graphically illustrate the distribution difference of a diverging pattern between a pair of data sets. First, from each data set, we collect a sample of random variables for a diverging pattern X, called the i^{th} *support* of X which is defined below.

DEFINITION 6.1. *The* i^{th} support *of a pattern* X *in* $\mathcal{D}$, *denoted as* $sup^i(X, \mathcal{D})$, *is the probability of* X *occurring in the set of transactions from the beginning of* $\mathcal{D}$ *to the* i^{th} *occurrence of* X *in* $\mathcal{D}$.

Let $sup^1(X, \mathcal{D})$, $sup^2(X, \mathcal{D})$, ..., $sup^n(X, \mathcal{D})$ ($n = \|cov(X, \mathcal{D})\|$) be the set of the i^{th} supports ($1 \leq i \leq n$) of a pattern X in $\mathcal{D}$. The empirical distribution function $F_n(x)$ is a function of x, which equals the decimal fraction of the number of $sup^i(X, \mathcal{D})$ that are less than or equal to x for each x, $-\infty < x < +\infty$, *i.e.*,

$$F_n(x) = \frac{1}{n} \sum_{i=1}^{n} I_{\{sup^i(X,\mathcal{D}) \leq x\}}, \tag{6.6}$$

[10] The K-S test is a goodness of fit test used to determine whether two underlying one-dimensional probability distributions differ, or whether an underlying probability distribution differs from a hypothesized distribution, in either case based on finite samples.

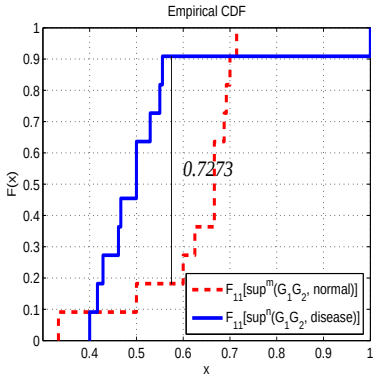


Figure 4: K-S test for gene set $\{G_1, G_2\}$.

where $I_{\{x_i \leq x\}}$ is the indicator function that equals 1 if $x_i \leq x$ and 0 otherwise.

One of the advantages of empirical distribution function is that it leads to a graphical presentation of the data. Figure 4 plots the empirical distribution functions of $sup^m(G_1G_2, normal)$ and $sup^n(G_1G_2, disease)$, where $1 \leq m, n \leq 11$. You can see that, for most of the x values, the distribution of $sup^m(G_1G_2, normal)$ that is strictly less than x is clearly less than the distribution of $sup^n(G_1G_2, disease)$ that is less than x. That is, by-and-in-large the $sup^m(G_1G_2, normal)$ values are larger than the $sup^n(G_1G_2, disease)$ values for the same empirical distribution. Moreover, in this case, the maximum deviation occurs near $x = 0.56$, where the empirical c.d.f of $sup^m(G_1G_2, normal)$ that is less than 0.56 is around 0.1818 (2 out of the 11 values), and the empirical c.d.f of $sup^n(G_1G_2, disease)$ that is less than 0.56 is 0.9091 (10 out of the 11 values). Thus the maximum difference between $sup^m(G_1G_2, normal)$ and $sup^n(G_1G_2, disease)$ in empirical c.d.f is 0.7273. According to the K-S test, the two distributions are significantly different at the 0.3% significance level.

Table 4 describes some discovered diverging patterns (one for each dataset) in terms of their support values (column 2 and 3), weights (column 4 and 5), and diverging ratio (column 6). We illustrate their empirical distribution functions in Figure 5. According to the K-S test, the two distributions illustrated in each sub-figure are significantly different with p-values less than 0.5%.

We have done the K-S test on all the diverging patterns discovered from the above six real-world databases with $t_s = 0.2\%$ and $t_d = 0.3$. The results show that the distributions of all the discovered patterns in the two corresponding datasets are significantly different with p-values less than 1%.

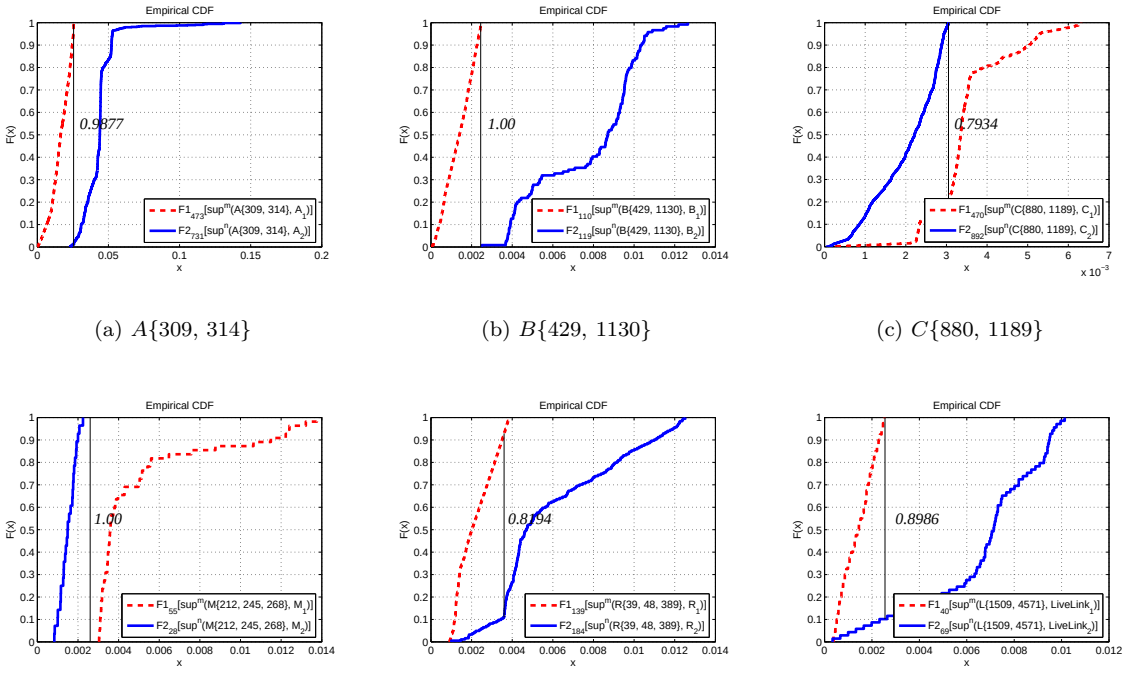
6.4 Usefulness of the discovered patterns. The two *Livelink* data sets (L_1 and L_2) are the click streams from two groups of *Livelink* users for the same period of time. The diverging pattern shown in Table 4 for the *Livelink* data set is $\{1509, 4571\}$, where 1509 represents "Advanced Search Options" and 4571 represents "Advanced Search Using Categories and Attributes". From Table 4, this pattern is more frequent in L_2 than in L_1, which seems to indicate that L_2 users like "Advanced Search Using Categories and Attributes" better than L_1 users. However, the frequency of the pattern changes differently in the two data sets. In L_1 its frequency increases as shown in its weight vector $\langle -1.02, +15.11 \rangle$, but in L_2 it decreases dramatically as shown in $\langle -12.24, +0.85 \rangle$. This means that the first group of users considers "Advanced Search using Categories and Attributes" useful and uses it more and more often, but the second group of users may consider this search option not as helpful as they thought and thus uses it less often than before. This discovery reveals that the "categories and attributes" designed in the advanced search tool is more suitable for the first group of users but not for the second group. Such a discovery can help the system designers improve their designs.

The other data sets used in our experiments are anonymized (i.e., their items and time stamps are renamed to protect privacy). Thus, we cannot reveal the true meanings of the diverging patterns discovered from them. To further show the usefulness of a diverging pattern, we assume that R_1 is a transaction data set for a supermarket for January and R_2 is for February. The frequencies that the customers bought the products {39, 48, 389} together are about the same in the two months as shown in Table 4, which means that the customers liked to buy these three products together in both months. However, the two weight vectors $< -5.07, +49.43 >$ and $< -40.02, +15.09 >$ indicate that the customers bought the three products together more frequently in late January and early February. The frequency of this product combination has significantly decreased in February. Such a trend cannot be discovered by applying only frequent pattern mining over the two data sets.

6.5 Scalability of *DP-mine* algorithm. Figure 6 illustrates the execution time of the proposed *DP-mine* algorithm on these data sets with different support thresholds between 0.2% and 1.0%. To test the scalability of *DP-mine* against the number of transactions, we generated several subsets of the biggest data set *BMS-POS*. The number of transactions in these subsets ranges from 100K to 500K. Figure 7 shows that *DP-mine* has linear scalability against the number of transactions in BMS-POS with $t_d = 0.2$ and different t_s values. Similar results are obtained from other data

Table 4: Selected diverging patterns

Pattern (X)	$sup(X, \mathcal{D}_1)$ (%)	$sup(X, \mathcal{D}_2)$ (%)	$\overrightarrow{\omega}(X, \mathcal{D}_1)$	$\overrightarrow{\omega}(X, \mathcal{D}_2)$	$div(X)$ (%)	Figure
$A\{309, 314\}$	2.56	1.78	$< -10.6, +110.12 >$	$< -204.49, +7.07 >$	86.96	5(a)
$B\{429, 1130\}$	0.25	0.36	$< -0.55, +38.81 >$	$< -29.87, +8.53 >$	71.20	5(b)
$C\{880, 1189\}$	0.21	0.31	$< -96.49, +24.02 >$	$< -26.75, +182.51 >$	62.03	5(c)
$M\{212, 245, 268\}$	0.29	0.21	$< -9.37, +4.89 >$	$< -1.55, +6.22 >$	33.61	5(d)
$R\{39, 48, 389\}$	0.38	0.35	$< -5.07, +49.43 >$	$< -40.02, +15.09 >$	55.36	5(e)
$L\{1509, 4571\}$	0.25	0.47	$< -1.02, +15.11 >$	$< -12.24, +0.85 >$	86.40	5(f)



(a) $A\{309, 314\}$ (b) $B\{429, 1130\}$ (c) $C\{880, 1189\}$

(d) $M\{212, 245, 268\}$ (e) $R\{39, 48, 389\}$ (f) $L\{1509, 4571\}$

Figure 5: Visualization of selected diverging patterns

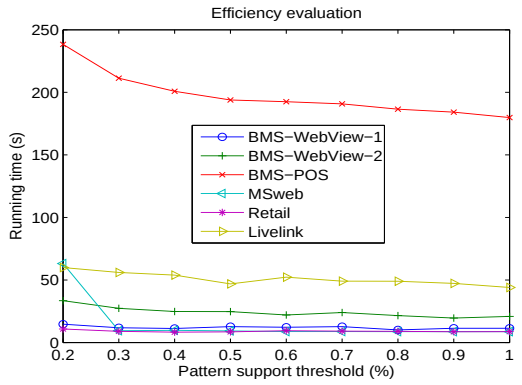


Figure 6: Runtime with different t_s

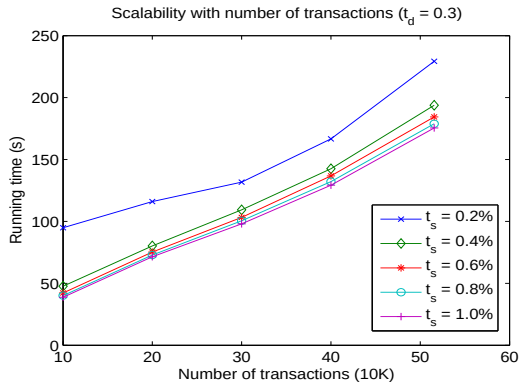


Figure 7: Scalability against number of transactions on BMS-POS

sets. We use *FP-growth* to find frequent patterns and their weights in the first phase of *DP-mine*. All the experiments are performed on a double-processor server, which has 2 Intel Xeon 2.4G CPU and 2G main memory, running on Linux with kernel version 2.6.

7 Related Work

Discovery of useful distinguishing features between data sets is an important objective in data mining. The concept of Emerging Patterns was first introduced in [6] to capture changes or differences between data sets. An Emerging Pattern (EP) is defined as an itemset X satisfying $growthrate(X) = \frac{sup(X,D_2)}{sup(X,D_1)} \geq g$, where D_1, D_2 are two different data sets and $g > 1$ is called the growth rate threshold. Several variants of emerging patterns have been introduced, with Jumping Emerging Patterns (JEPs) being the most important one [10]. JEPs are emerging patterns with infinity growth rate. Since a JEP can only be found in one distinct class in the database, it is a good indicator of that class. Other variants of emerging patterns include strong emerging patterns (which are emerging patterns with all subsets being emerging patterns) and the Most Expressive JEPs (which are the minimal JEPs) [10]. Different from diverging patterns, emerging patterns do not consider how frequency of a pattern changes within a dataset and how this change differs from the change in another dataset.

In [15], a new type of contrast patterns, called transitional patterns, was proposed to represent patterns whose frequencies increase/decrease dramatically at some time points of a transaction database. The concept of significant milestones for a transitional pattern was also introduced, which are the time points at which the frequency of the pattern changes most significantly. Different from diverging patterns, transitional patterns were defined on and discovered from a single database.

The problem of contrast-set mining was introduced in [3, 4], and the STUCCO algorithm was proposed to efficiently search through the space of contrast-sets. Contrast-sets are defined as conjunctions of attributes-value pairs that differ meaningfully in their probabilities across several groups. They can be used to identify differences between groups. Follow-up work in [16] discovered that existing commercial rule-finding system, Magnum Opus, can successfully perform the contrast-set task. The authors concluded that contrast-set mining is a special case of the more general rule-discovery task.

None of the above work addresses the problem of finding patterns whose frequency changes in different directions in two contrast data sets, which is the topic of this paper.

8 Conclusions

In this paper, we define a new type of contrast patterns, called *diverging patterns*, to represent the patterns whose frequency changes in different directions in two contrast data sets. We use a two-dimensional vector to represent each pattern, and define the pattern's diverging ratio based on the angular difference between its vectors in two data sets. Furthermore, an algorithm called *DP-mine* is developed, which makes use of a standard frequent pattern mining algorithm to compute relevant vectors efficiently. Finally, we present experimental results on six real-world data sets, showing that *DP-mine* can effectively and efficiently reveal new and useful knowledge from large databases.

There are several future topics of research that we are currently considering. First, we are applying diverging pattern mining to a medical data set that contains blood test results for a large number of patients in a five-year period. Our initial results are encouraging, showing interesting and potentially useful divering pat-

terns. We are currently validating the results. Due to proprietary reasons, we cannot release the initial results we obtained from this medical data set at this time, but may report them in the future. We also plan to apply diverging pattern mining to other real-world domains. For example, in stock market analysis, traders can make better transaction decisions by identifying situations of divergence, where the price of a stock and a set of relevant indicators, such as the money flow index (MFI), are moving in opposite directions. Second, in this paper we focused on proposing the new framework for defining and finding diverging patterns. The DP-mine algorithm proposed in this paper makes use of existing frequent pattern mining algorithms as the first step in finding diverging patterns. To further improve the efficiency of the algorithm, we will investigate whether it is possible to mine diverging patterns without finding all the frequent patterns in the two databases. Third, we are interested in applying the present studies to other data mining problems, such as sequential mining, stream mining, and graph mining. Finally, the weight vector defined in this paper captures the general direction of a pattern's frequency change by considering the distances of the pattern's occurrences to the central position of the database. Obviously, more types of frequency changes exist in real data sets. One way to extend this framework is to automatically find a reference point in each of the data sets, which is a point where a pattern changes its frequency most significantly, and use the reference point (instead of the center position) to calculate the positive and negative weights of the pattern in a data set. We will also study the possibility of representing a pattern's distribution using N-dimensional ($N > 2$) vectors so that more types of divergence can be represented and discovered.

References

[1] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD Intl. Conf. on Management of data*, pages 207–216, 1993.

[2] J. Bailey, T. Manoukian, and K. Ramamohanarao. Fast algorithms for mining emerging patterns. In *Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 39–50, 2002.

[3] S. D. Bay and M. J. Pazzani. Detecting change in categorical data: mining contrast sets. In *Proceedings of the fifth ACM SIGKDD Intl. Conf. on Knowledge discovery and data mining*, pages 302–306, 1999.

[4] S. D. Bay and M. J. Pazzani. Detecting group differences: Mining contrast sets. *Data Min. Knowl. Discov.*, 5(3):213–246, 2001.

[5] T. Brijs, G. Swinnen, K. Vanhoof, and G. Wets. Using association rules for product assortment decisions: a case study. In *Proceedings of the fifth ACM SIGKDD Intl. Conf. on Knowledge discovery and data mining*, pages 254–260, 1999.

[6] G. Dong and J. Li. Efficient mining of emerging patterns: discovering trends and differences. In *Proceedings of the fifth ACM SIGKDD Intl. Conf. on Knowledge discovery and data mining*, pages 43–52, 1999.

[7] H. Fan and K. Ramamohanarao. Fast discovery and the generalization of strong jumping emerging patterns for building compact and accurate classifiers. *IEEE Transactions on Knowledge and Data Engineering*, 18(6):721–737, 2006.

[8] J. Han, J. Pei, Y. Yin, and R. Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, 8(1):53–87, 2004.

[9] X. Huang, A. An, N. Cercone, and G. Promhouse. Discovery of interesting association rules from livelink web log data. In *Proceedings of the 2002 IEEE Intl. Conf. on Data Mining (ICDM'02)*, 2002.

[10] J. Li, G. Dong, and K. Ramamohanarao. Making use of the most expressive jumping emerging patterns for classification. In *Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Current Issues and New Applications*, pages 220–232, 2000.

[11] B. Liu, W. Hsu, and Y. Ma. Discovering the set of fundamental rule changes. In *Proceedings of the seventh ACM SIGKDD Intl. Conf. on Knowledge discovery and data mining*, pages 335–340, 2001.

[12] E. Loekito and J. Bailey. Fast mining of high dimensional expressive contrast patterns using zero-suppressed binary decision diagrams. In *Proceedings of the 12th ACM SIGKDD Intl. Conf. on Knowledge discovery and data mining*, Philadelphia, USA, 2006.

[13] Jian Pei, Jia-Wei Han, Hong-Jun Lu, Shojiro Nishio, Shi-Wei Tang, and Dong-Qing Yang. H-mine: Hyper-structure mining of frequent patterns in large databases. In *ICDM '01: Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 441–448, Washington, DC, USA, 2001. IEEE Computer Society. ISBN 0-7695-1119-8.

[14] A. Stuart, K. Ord, and S. Arnold. *Kendall's Advanced Theory of Statistics*. 1999.

[15] Q. Wan and A. An. Transitional patterns and their significant milestones. In *Proceedings of the 7th IEEE Intl. Conf. on Data Mining*, Omaha, NE, USA, 2007.

[16] G. I. Webb, S. Butler, and D. Newlands. On detecting differences between groups. In *Proceedings of the ninth ACM SIGKDD Intl. Conf. on Knowledge discovery and data mining*, pages 256–265, 2003.

[17] Z. Zheng, R. Kohavi, and L. Mason. Real world performance of association rule algorithms. In F. Provost and R. Srikant, editors, *Proceedings of the Seventh ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, pages 401–406, 2001.