Efficient Mining of Contrast Patterns on Large Scale Imbalanced Real-Life Data

Jinjiu Li, Can Wang, Wei Wei, Mu Li, and Chunming Liu

Advanced Analytics Institute, University of Technology Sydney, Australia
{fJinjiu.Li,Can.Wang,Wei.Wei,Mu.Li,Chunming.Liug}@student.uts.edu.au

Abstract. Contrast pattern mining has been studied intensively for its strong discriminative capability. However, the state-of-the-art methods rarely consider the class imbalanced problem, which has been proved to be a big challenge in mining large scale data. This paper introduces a novel pattern, i.e. converging pattern, which refers to the itemsets whose supports contrast sharply from the minority class to the majority one. A novel algorithm, ConvergMiner, which adopts T*-tree and branch bound pruning strategies to mine converging patterns efficiently, is proposed. Substantial experiments in online banking fraud detection show that the ConvergMiner greatly outperforms the existing cost-sensitive classification methods in terms of predicative accuracy. In particular, the efficiency improves with the increase of data imbalance.

Keywords: Contrast Pattern, Class Imbalance, Fraud Detection.

1 Introduction

Contrast patterns [11] are the itemsets showing the discrimination between two classes in a data set, and they are very useful to detect anomalies. It has been widely studied in terms of emerging pattern [3], jumping emerging pattern [5], and contrast capturing method [11]. In addition, classifiers built by the contrast patterns, such as CAEP [4], are shown to outperform most of the existing methods (e.g. C4.5 [12], CMAR [8]) on accuracy. However, the existing contrast pattern mining methods do not consider the issue of class imbalance (i.e. the data distribution is extremely imbalanced among different classes). Such characteristics is commonly observed in the real-life applications.

Table 1 gives an example of four fraudulent patterns that appear in online banking with their False Positive Rates (FPR). There are 1,000,000 genuine transactions (Genuine for short) and 1,000 fraud transactions (Fraud for short) in the transaction set that involves 112 attributes. In order to catch frauds effectively, two criteria are proposed: $FPR \leq 0.8$ and the support in Genuine is no larger than 0.0001. We can see that only p_4 is qualified. In contrast, patterns p_1, p_2 and p_3 have rather high FPR and larger supports in the Genuine, so they can not be applied at an acceptable cost of the investigation fee spent on post-inspection.

J. Pei et al. (Eds.): PAKDD 2013, Part I, LNAI 7818, pp. 62–73, 2013.

Table 1. Patterns for Fraud Detection

Rules	$FPR = \|Genuine\|/\|Genuine + Fraud\|$	Frequency	
		Genuine (size=1M)	Fraud (size=1K)
$p_1 \rightarrow Fraud$	0.96	5000	200
$p_2 \rightarrow Fraud$	0.997	6000	20
$p_3 \rightarrow Fraud$	0.968	3000	100
$p_4 \rightarrow Fraud$	0.769	100	30

The fraudsters invariably try to disguise their behaviors maliciously and bypass the detection system, some typical features are then identified: (i) The itemsets that frequently occur in the Fraud are also usually frequent in the Genuine (such as p_1, p_2, p_3). (ii) The itemsets with a strong discriminative power generally are rarely seen ($support \leq 0.0001$) in Genuine (e.g. p_4). Therefore, there are three main challenges in mining contrast patterns to detect the Fraud. (1) A strict constraint on FPR demands an extremely small support in the Genuine (as shown in Table 1). Given a small support threshold, a huge number of itemsets will be generated, especially when plenty of attributes are involved. (2) Mining contrast patterns among the large scale data is computationally expensive. (3) Selecting the optimal pattern set for classification is also with a high computational complexity. It is common to see that the serious overlapping among patterns, on which the classifiers are built, negatively impact the overall performance in catching the Fraud.

However, the existing Emerging pattern miner MDB-LL_{border} [3] dose not consider the imbalance issue. By applying Max-Miner [1] to mine long patterns with a small support threshold in the Genuine, MDB-LL_{border} consumes a overwhelming memory and computational time due to challenges (1) and (2). Though jumping emerging patterns [5] can be quickly captured, they are rarely observed in the fraud detection scenario according to the feature (i). CAEP [4] suffers from the serious pattern overlapping confronting with challenge (3).

In this paper, we propose a novel approach to mine contrast patterns in the large scale imbalanced data by exploring the converging patterns, which effectively handle the above challenges. The main contributions are as follows.

- Define the converging patterns, a novel type of contrast patterns that significantly differentiate itemsets in the class imbalanced data; and propose an effective algorithm, called ConvergMiner, to mine the converging patterns.
- Introduce a series of branch bound pruning strategies to reduce the computational complexity; present a set of operators and theorems for border substraction to cut the computational time and memory cost; and design a new index structure T*-tree to support the fast checking of candidate patterns.
- Perform extensive experiments to evaluate the effectiveness of our approach and strategies against the state-of-art methods in terms of accuracy, convergence sensitivity and scalability.

The rest of the paper is organized as follows: Section 2 reviews the related work. Section 3 defines the problems and terminologies. Section 4 presents the

approach of pattern border operation. Section 5 introduces two refining strategies, T*tree and border splitting. Section 6 proposes the algorithm of converging pattern mining. Section 7 proposes the pattern selection and scoring methods for prediction. Section 8 shows the evaluation. We conclude in Section 9.

2 Related Work

Several approaches have been proposed to mine the contrast patterns [11]. Emerging Patterns (EPs), firstly introduced in [3], uses growth rate to measure the support contrast of an itemset. Disjunctive emerging pattern [11] is a variant of EPs to discover the contrast patterns in a high dimensional data set. The work in [5] extends the concept of EPs, and introduces the Jumping Emerging Patterns (JEPs) whose supports increase abruptly from zero in one data set to non-zero in another one. According to [5,4], classifiers built by the EPs or JEPs outperform most methods (e.g. C4.5 [12], CMAR [8]) on accuracy.

All the above methods are proposed for the class balanced data. However, more researchers pay attention to the class imbalanced problems [13,10], which widely exit in the real world and greatly challenge the classic data mining algorithms. Tremendous research efforts have been made on the class-imbalanced data, for instance, Cost-Sensitive Neural Network [10], Ad-Cost [13], Cost-SVM [9], etc. None of them addresses the mining of contrast patterns in the class imbalanced data set, while we focus on solving this problem.

3 Problem Statement

Let $I = \{i_1, i_2, ..., i_m\}$ be a set of items, an itemset X is a subset of I. A transaction T is an itemset X whose number of elements is fixed according to the number of attributes in the data set. A data set D is a set of T. Suppose there are two classes denoted as C_t (the target class, e.g. Fraud) and C_b (the background class, e.g. Genuine), and two data sets D_t and D_b in D correspond to the respective classes C_t and C_b. Transactions in D_b are labeled as C_b, and

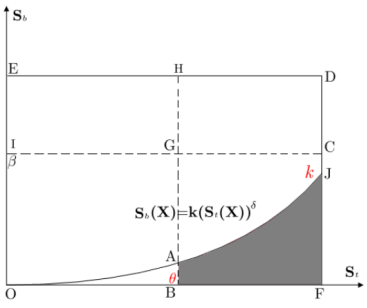

Fig. 1. An example of converging patterns

those in D_t are marked as C_t. $S_b(X)$ and $S_t(X)$ denote the supports of itemset X in D_b and D_t, respectively.

Definition 1. ***Converging Patterns*** *(CPs for short) are composed of the itemsets X that satisfy the following condition: $CPs = \{X | S_b(X) \leq k(S_t(X))^{\delta}, S_t(X) \geq \theta\}$, and the* ***Contrast Rate*** *of CPs is defined as $F(X) = (S_t(X))^{\delta}/S_b(X)$, where $k > 0$ is the contrast coefficient, $\delta > 0$ is the converging exponent, and $\theta > 0$ is the threshold of the minimal support in D_t.*

As the above definition indicates, CPs are controlled by k and δ. The larger the δ, the higher the contrast rate of the generated converging patterns. For example, in Fig. 1, itemsets are projected onto the support plane where the vertical axis stands for $S_b(X)$ and the horizontal axis denotes $S_t(X)$. According to Definition 1, the itemsets in the shadow region $\mathcal{I}_{ABFJ}$ compose CPs, denoted as $\{\mathcal{I}_{ABFJ}\}$. It is also derived by:

$$CPs = \{\mathcal{I}_{ABFJ}\} = \{\mathcal{I}_{BFDH}\} - \{\mathcal{I}_{GCDH}\} - \{\mathcal{I}_{AJCG}\}, \tag{1}$$

where $\mathcal{I}_{\square}$ represents all the itemsets in region $\square$.

The itemsets in $\mathcal{I}_{BFDH}$ can be obtained by the Max-Miner [1] in D_t, i.e. $\{\mathcal{I}_{BFDH}\} = \{X | S_t(X) \geq \theta\}$. For the second term in Equation (1), we have $\{\mathcal{I}_{GCDH}\} = \{\mathcal{I}_{BFDH}\} \cap \{\mathcal{I}_{ICDE}\}$. The mining of itemsets in $\mathcal{I}_{ICDE}$ is similar to that in $\mathcal{I}_{BFDH}$. Let β=max(k, π), where π is a proper support threshold for Max-Miner to output long patterns successfully in D_b. Unlike the algorithm in [3], which applies a strict growth rate β to mine EPs and does not work in imbalanced data as mentioned in Section 1. Then the itemsets in $\mathcal{I}_{ICDE}$ are obtained as $\{\mathcal{I}_{ICDE}\} = \{X | S_b(X) \geq \beta\}$. Thus, $\{\mathcal{I}_{GCDH}\} = \{X | S_t(X) \geq \theta\} \cap \{X | S_b(X) \geq \beta\}$. Therefore, the main task of finding the itemsets in $\mathcal{I}_{BFCG}$ becomes the effective filtering of all the itemsets in $\mathcal{I}_{AJCG}$, i.e., the last term in Equation (1).

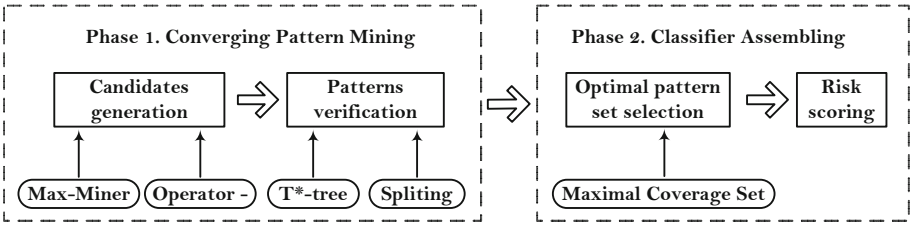

Fig. 2. Framework of building the anomaly detector powered by CPs, where oval boxes display the supporting techniques for each stage

Accordingly, the mining of CPs is fulfilled by two phases (as shown in Fig. 2): candidates generation and pattern verification (in phase 1), and predicative classifier building (in phase 2). Section 4 introduces the method to generate the candidate itemsets, section 5 provides the strategies to eliminate the redundancy, section 7 presents the techniques to select the optimal pattern set and calculate the prediction score.

4 Candidates Generation

As specified in Section 3, CPs is identified by Equation (1). In order to perform the substraction of itemsets quickly, we propose the Pattern Border to collect the itemsets and provide several theorems to support the retrieve of the candidate CPs by algebraic operations rather than traversing all elements in borders.

4.1 Pattern Border

Definition 2. *Given two itemsets $\mathcal{L}$ and $\mathcal{R}$ ($\mathcal{L} \subseteq \mathcal{R}$), the ordered interval $[\mathcal{L}, \mathcal{R}]$ forms a **Pattern Border**, composed of a set of patterns. The collection of itemsets in $[\mathcal{L}, \mathcal{R}]$ is defined as: $[\mathcal{L}, \mathcal{R}] = \{Y | \exists X \subseteq \mathcal{R}, \mathcal{L} \subseteq Y \subseteq X\}$.*

Example 1. *Border $[a, abcd]$ contains 8 patterns: $a, ab, ac, ad, abc, abd, acd, abcd$.*

Pattern border is an important concept in CPs mining. A proper adjustment on the pattern border greatly avoids the full iteration of every candidate itemset in $\mathcal{I}_{AJCG}$, and dramatically speeds up the search of CPs. Since the smallest itemset in $[\mathcal{L}, \mathcal{R}]$ is $\mathcal{L}$ and $\mathcal{R}$ is the largest one, an itemset $X \in [\mathcal{L}, \mathcal{R}]$ then satisfies the following conditions:

$$\max_{\mathcal{L} \subseteq X \subseteq \mathcal{R}} (S_t(X)) \leq S_t(\mathcal{L}), \min_{\mathcal{L} \subseteq X \subseteq \mathcal{R}} (S_t(X)) \leq S_t(\mathcal{R}) \tag{2}$$

Definition 3. *The **Upper Bound** $F^u([\mathcal{L}, \mathcal{R}])$ and **Lower Bound** $F^l([\mathcal{L}, \mathcal{R}])$ of the contrast rate $F(X)$, where $X \in [\mathcal{L}, \mathcal{R}]$, are defined as follows:*

$$F^l([\mathcal{L}, \mathcal{R}]) \leq F(X) \leq F^u([\mathcal{L}, \mathcal{R}]),\ \mathcal{L} \subseteq \mathrm{X} \subseteq \mathcal{R} \tag{3}$$

$$F^l([\mathcal{L}, \mathcal{R}]) = \frac{(S_t(\mathcal{R}))^\delta}{S_b(\mathcal{L})},\ F^u([\mathcal{L}, \mathcal{R}]) = \frac{(S_t(\mathcal{L}))^\delta}{S_b(\mathcal{R})}. \tag{4}$$

In Section 5.2, $F^u(X)$ and $F^l(X)$ are applied to prune the searching space.

4.2 Subtraction of Pattern Borders

As shown in *Example* 1, pattern border is a simple and efficient way to collect patterns. Border substraction is frequently performed to generate the candidate itemsets in $\mathcal{I}_{BFCG}$, according to Equation (1). The most straightforward method is to enumerate each element in the border and eliminate the redundant elements. But, it is rather costly in both computational time and memory when the border is large. Thus, we implement the substraction of two borders only based on two operators $\times$ and $-$, rather than exploring the borders directly.

Let itemset $X \subseteq I$, $I_X = \{X' | X' \subseteq X\}$, and $I_X^+ = \{X' | X' \subseteq X, X' \neq \emptyset\}$, several definitions and theorems are proposed to support the border subtraction on the two sets of itemsets: S_1 and S_2.

Definition 4. *The* ***Operator*** $\times$ *for* S_1 *and* S_2 *is defined as:*

$$S_1 \times S_2 = \{X_1 \cup X_2 | X_1 \in S_1, X_2 \in S_2\} \tag{5}$$

The ***Operator*** $-$ *for* S_1 *and* S_2 *is defined as:*

$$S_1 - S_2 = \{X' | X' \in S_1, X' \notin S_2\} \tag{6}$$

In Particular, $I_{X_1} \times I_{X_2} = \{X'_1 \cup X'_2 | X'_1 \in I_{X_1}, X'_2 \in I_{X_2}\} = \{X'_1 \cup X'_2 | X'_1 \subseteq X_1, X'_2 \subseteq X_2\} = I_{X_1 \cup X_2}$, and $I_{X_1} - I_{X_2} = \{X' | X' \subseteq X_1, X' \not\subseteq X_2\}$. The operator $\times$ is used for splitting a big border, e.g. $[\emptyset, \{a,b,c,d\}] = I_{\{a,b\}} \times I_{\{c,d\}}$. The operator $-$ is adopted during the subtraction of two collections of itemsets, e.g. $[\emptyset, \{a,b,c,d\}] - [\emptyset, \{a,b\}] = I_{\{a,b\}} \times I^+_{\{c,d\}}$. It will be frequently executed when generating the candidate itemsets in $\mathcal{I}_{BFCG}$. We then easily get the following properties and theorems for the operators:

$$\textit{Distributive law} : (I_{X_1} \cup I_{X_2}) - I_{X_3} = (I_{X_1} - I_{X_3}) \cup (I_{X_2} - I_{X_3}) \tag{7}$$

$$\textit{Commutative law} : I_{X_1} \times I_{X_2} = I_{X_2} \times I_{X_1} \tag{8}$$

$$\textit{Associative law} : (I_{X_1} \times I_{X_2}) \times I_{X_3} = I_{X_1} \times (I_{X_2} \times I_{X_3}) \tag{9}$$

$$\textit{Allocation law} : I_{X_1} \times (I_{X_2} \cup I_{X_3}) = (I_{X_1} \times I_{X_2}) \cup (I_{X_1} \times I_{X_3}) \tag{10}$$

Theorem 1. $I_{X_1 \cup X_2} = (I^+_{X_1} \times I^+_{X_2}) \cup I^+_{X_1} \cup I^+_{X_2} \cup I_\emptyset$

Proof. $I_{X_1} \times I_{X_2} = (I^+_{X_1} \cup \emptyset) \times (I^+_{X_2} \cup \emptyset) = (I^+_{X_1} \times I^+_{X_2}) \cup I^+_{X_1} \cup I^+_{X_2} \cup I_\emptyset$

Theorem 1 decomposes $I_{X_1 \cup X_2}$ into smaller parts easily to be processed.

Theorem 2. *If* $X_1 \cap X_3 = \emptyset$*, then* $(I^+_{X_1} \times I_{X_2}) - I_{X_3} = I^+_{X_1} \times I_{X_2}$

Proof. For $\forall X \in (I^+_{X_1} \times I_{X_2})$, we have $X \cap X_1 \neq \emptyset$. Then $(I^+_{X_1} \times I_{X_2}) \cap I_{X_3} = \emptyset$ as $X_1 \cap X_3 = \emptyset$

Below, we illustrate the use of these two operators for the border substraction.The above techniques are applied to Algorithm 1 (line 4-7) in Section 6.

Example 2. *The border subtraction* $[\emptyset, abcdefgh] - [\emptyset, ab] - [\emptyset, bc]$ *is obtained by the following calculation:* $(I_{abcdefgh} - I_{ab}) - I_{bc} = (I_{ab} \times I^+_{cdefgh}) - I_{bc} = I_{ab} \times ((I_c \times I^+_{defgh}) \cup I^+_c) - I_{bc} = (I_{abc} \times I^+_{defgh}) \cup (I^+_a \times I^+_c \times I_b)$*. By iterating* $I^+_{X_2}$*, we get* $I_{X_1} \times I^+_{X_2} = I_{X_1} \times \sum_{i=1, x_i \in X_2}^{|X_2|} (I^+_{x_i} \times \prod_{j>i}^{|X_2|} I_{x_j})$*. The set* I^+_{defgh} *further splits into sub-borders:* $I^+_{defgh} = (I^+_d \times I_{efgh}) \cup (I^+_e \times I_{fgh}) \cup (I^+_f \times I_{gh}) \cup (I^+_g \times I_h) \cup I^+_h$*. Therefore, we get* $[\emptyset, abcdefgh] - [\emptyset, ab] - [\emptyset, bc] = [d, abcdefgh] \cup [e, abcefgh] \cup [f, abcfgh] \cup [g, abcgh] \cup [h, abch] \cup [ac, abc]$.

5 Pattern Verification

The qualification of itemsets in $\mathcal{I}_{AJCG}$ is verified in a Branch-and-Bound manner. Given a border $[\mathcal{L}, \mathcal{R}]$, $F^u(X)$ and $F^l(X)$ are estimated by Equation (4), and are used to check the qualification of $[\mathcal{L}, \mathcal{R}]$. Instead of scanning the database repeatedly, we propose the T*-tree to calculate $S_b(\mathcal{L})$, $S_t(\mathcal{L})$, $S_b(\mathcal{R})$ and $S_t(\mathcal{R})$, which are used to compute $F^u(X)$ and $F^l(X)$ by scanning database only once. Then, the strategies are presented to split $[\mathcal{L}, \mathcal{R}]$ for checking the sub borders.

5.1 T*-tree Index

Transaction Tree (T*-tree) extends the classic spatial index, i.e. R-tree [2], to obtain new properties which significantly accelerate the calculation of support. In a R-tree, an object is represented by its Minimal Bounding Box (MBB) [2], which is the minimum bounding rectangle surrounding the object. R-tree is built on all the MBBs recursively. In order to efficiently get the support of an itemset, we introduce a novel index T*-tree, in which all the transactions are treated as spatial objects wrapped by their MBBs. The query on T*-tree is to check how many transactions match a given pattern. Accordingly, the patterns to be queried are also mapped to MBBs.

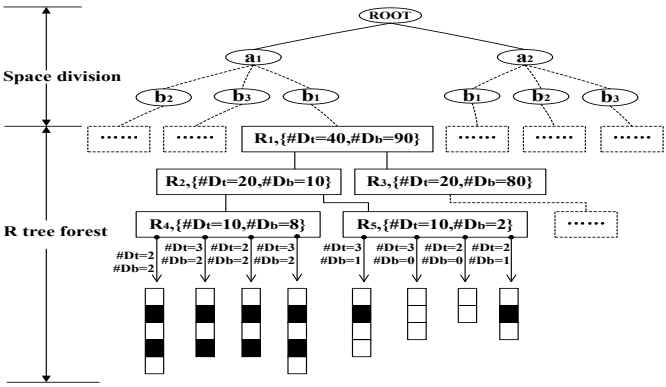

Fig. 3. An example of T*-tree, where R_i stands for MBB of an internal node

The main challenge of using T*-tree in a high dimensional data set is that the overlapping among nodes increases when more objects are inserted. Serious overlapping affects the query efficiency severely. Therefore, based on the data distribution, the data space is partitioned into multiple subspaces to reduce the overlapping of nodes as much as possible. The space partition follows a two-tier tree structure, as showed in Fig. 3. In this way, the computational time is dramatically cut by taking the following advantages:

1) A T*-tree consists of two layers: trunk and branch. All the attributes are sorted in a descending order on the gain ratio [6]. Top 10% (an empirical value, for instance) attributes are chosen as the trunk nodes, and the rest are the branch nodes. The top one attribute is the first level of trunk and so forth. With the growth of trunk levels, the data space is divided into smaller subspaces. After that, the branches are built in a similar process as R-tree.
2) Each node stores two values (i.e. $\sharp D_t$ and $\sharp D_b$, called *local supports*), which record the number of transactions in D_t and D_b, respectively.
3) The leaf nodes store the transaction chains that record the numbers of transactions covered by the same MBBs in D_t and D_b respectively. Multiple transactions can be stored in one transaction chain, especially when the transaction volume is huge in D_b.

5.2 Splitting Strategies

Given $[\mathcal{L}, \mathcal{R}]$, if $F^l([\mathcal{L}, \mathcal{R}]) \leq 1/k \leq F^u([\mathcal{L}, \mathcal{R}])$ (k is the contrast coefficient in Definition 1), then $[\mathcal{L}, \mathcal{R}]$ needs to be split into smaller borders for further validation. The splitting strategies below are used to speed up the search process, output CPs and remove unqualified borders quickly.

Strategy 1. If $F^u([\mathcal{L}, \mathcal{R}]) < 1/k$, then $F(X) \leq F^u([\mathcal{L}, \mathcal{R}]) < 1/k$. As a result, no itemset in $[\mathcal{L}, \mathcal{R}]$ is qualified to be chosen. So $[\mathcal{L}, \mathcal{R}]$ is safely removed.

Strategy 2. If $F^l([\mathcal{L}, \mathcal{R}]) \geq 1/k$, then $F(X) \geq F^l([\mathcal{L}, \mathcal{R}]) \geq 1/k$. Consequently, all the itemsets in $[\mathcal{L}, \mathcal{R}]$ must be selected. There are two directions to split $[\mathcal{L}, \mathcal{R}]$: for each single item $i \in \{\mathcal{R}\} - \{\mathcal{L}\}$, we extend $\mathcal{L}$ to the sub-border $[\mathcal{L} \cup \{i\}, \mathcal{R}]$, denoted as $\mathcal{L}^+$; or narrow down $\mathcal{R}$ to form the sub-border $[\mathcal{L}, \mathcal{R}/\{i\}]$, denoted as $\mathcal{R}^-$.

Strategy 3. If $F^u([\mathcal{L}, \mathcal{R}]) - 1/k \leq 1/k - F^l([\mathcal{L}, \mathcal{R}])$, it is quicker to locate the sub-borders that can be removed immediately with *Strategy 1* by narrowing their upper bound. The splitting must follow the direction in which F^u of sub-borders decreases as fast as possible. If $\max\limits_{i \in \{\mathcal{R}\} - \{\mathcal{L}\}} \{F^u([\mathcal{L}, \mathcal{R}]) - F^u([\mathcal{L} \cup \{i\}, \mathcal{R}])\} \geq \max\limits_{i \in \{\mathcal{R}\} - \{\mathcal{L}\}} \{F^u([\mathcal{L}, \mathcal{R}]) - F^u([\mathcal{L}, \mathcal{R}/\{i\}])\}$, the direction is $\mathcal{L}^+$; otherwise, $\mathcal{R}^-$.

Strategy 4. If $F^u([\mathcal{L}, \mathcal{R}]) - 1/k \geq 1/k - F^l([\mathcal{L}, \mathcal{R}])$, it is more efficient to qualify the sub-borders which can be delivered directly with *Strategy 2* by increasing their lower bound. The splitting must follow the direction in which the F^l of sub-borders increases as quickly as possible. If $\max\limits_{i \in \{\mathcal{R}\} - \{\mathcal{L}\}} \{F^l([\mathcal{L} \cup \{i\}, \mathcal{R}]) - F^l([\mathcal{L}, \mathcal{R}])\} \geq \max\limits_{i \in \{\mathcal{R}\} - \{\mathcal{L}\}} \{F^l([\mathcal{L}, \mathcal{R}/\{i\}]) - F^l([\mathcal{L}, \mathcal{R}])\}$, the direction is $\mathcal{L}^+$; otherwise, $\mathcal{R}^-$.

6 Algorithm of Converging Pattern Mining

Here, a novel algorithm ConvergMiner is proposed to mine CPs efficiently. Algorithm 1 presents the main process of mining CPs, and function *CheckContrast* is the key function to validate the qualification of candidate itemsets. There are four steps in the main process: **Step 1**. Build the T*-tree on D_t and D_b (Line 1). **Step 2**. Employ the Max-Miner to extract the pattern borders located at $\mathcal{I}_{ICDE}$ and $\mathcal{I}_{BFDH}$ (Line 2-3). **Step 3**. Perform the border substraction to obtain the pattern borders locate at $\mathcal{I}_{BFCG}$ (Line 4-7). **Step 4**. Search CPs in $\mathcal{I}_{BFCG}$ by a Divide-and-Conquer procedure (Line 8-9).

7 Scoring for Classification

There are two critical issues to be solved during the construction of an accurate CPs-based classifier: pattern selection and risk scoring. We adopt an effective measure, Maximal Coverage Gain (MCG) [7], to select the globally optimal pattern set. Once the optimal pattern set $\widehat{P}$ has been obtained, a base score for

Algorithm 1. ConvergMiner

Data: $D_t, D_b, k, \delta, \theta, \beta, len$
Result: Converging patterns
$H \leftarrow \emptyset$, Build T^*-tree on D_t and D_b
$S^{'} = \{[\emptyset, X] | S_t(X) \geq \theta\}$ /* Get itemsets from $\mathcal{I}_{BFDH}$ */
$S^{''} = \{[\emptyset, X] | S_b(X) \geq \beta\}$ /* Get itemsets from $\mathcal{I}_{ICDE}$ */
for *each* E *in* $S^{'}$ **do**
  for *each* U *in* $S^{''}$ **do**
    $E \leftarrow E - U$ /* Obtain candidates by Opertaor $-$ introduced in Section 4.2 */
  $H \leftarrow H \cup E$
for *each* $[\mathcal{L}, \mathcal{R}]$ *in* H **do**
  $CheckContrast([\mathcal{L}, \mathcal{R}], T^*tree, k, \delta)$ /* Search CPs in $[\mathcal{L}, \mathcal{R}]$ */
Function CheckContrast($[\mathcal{L}, \mathcal{R}], T^*tree, k, \delta$)
Query T*-tree to collect $S_t(\mathcal{L}), S_t(\mathcal{R}), S_b(\mathcal{L}), S_b(\mathcal{R})$
if $S_b(\mathcal{L})$=0 *or* $F^l([\mathcal{L}, \mathcal{R}]) \geq \frac{1}{k}$ **then** // Based on strategy 2
  Output $[\mathcal{L}, \mathcal{R}]$ as CPs group
else if $S_b(\mathcal{R}) > 0$ *and* $F^u([\mathcal{L}, \mathcal{R}]) < \frac{1}{k}$ **then** // Based on strategy 1
  Remove $[\mathcal{L},\mathcal{R}]$
else if Direction$([\mathcal{L},\mathcal{R}],k,\delta)=\mathcal{R}^-$ **then** // Based on strategies 3 and 4
  for *each* $i \in \{\{\mathcal{R}\} - \{\mathcal{L}\}\}$ **do**
    CheckContrast$([\mathcal{L}$,R/{i}],k,δ)
else // Based on strategies 3 and 4
  for *each* $i \in \{\{\mathcal{R}\} - \{\mathcal{L}\}\}$ **do**
    if $\mathcal{L} \cup \{i\} \subsetneq$H **then**
      CheckContrast$([\mathcal{L} \cup$ {i}$,\mathcal{R}],k,\delta$)

each rule $r : p \rightarrow C_t$ in $\widehat{P}$ is calculated by Equation (11). The base score of p_i represents its impact in classifiers.

$$Base(p_i) = S_t(p_i) * F(p_i)/(1 + F(p_i)) \tag{11}$$

In the prediction phase, the score of u is calculated as follows:

$$\mathbb{S}(u) = \sum_{p_i \in \widehat{P}, u \prec p_i} Base(p_i) / \sum_{p_i \in P} Base(p_i) \tag{12}$$

Intuitively, given two transactions u_1 and u_2, if $\mathbb{S}(u_1) > \mathbb{S}(u_2)$, then the probability of $u_1 \in C_t$ is larger than that of $u_2 \in C_t$. Consequently, transaction u with a larger $\mathbb{S}(u)$ is more likely to be classified into C_t.

8 Experiment and Evaluation

Two real-life data sets are used: DS_1, the online banking transaction data from a major Australian bank. It contains 1,000,000 Genuine (D_b) and 1,000 Fraud (D_t), and the number of attributes involved is 112; DS_2, the social welfare payment claim data from Australia. It has 120,000 Genuine (D_b) and 2,120 Fraud (D_t) with 85 attributes. Below, ConvergMiner is compared with the state-of-the-art classification algorithms on these two class imbalanced data sets in terms of accuracy (i.e. Ada-Cost [13], Cost-NN [10], Cost-SVM [9] and CAEP [4]), efficiency (i.e. MDB-LL$_{border}$), and effectiveness of pruning strategies.

8.1 Accuracy

We compare the performance of the classifier powered by CPs with Ada-Cost [13], Cost-NN [10], Cost-SVM [9] and CAEP [4], from the perspective of False Positive Rate (FPR) and Detection Rate (DR, or True Positive Rate), which are defined as:

$$FPR = \frac{\text{False Alerts Number}}{\text{Genuine Instances Number}},\ DR = \frac{\text{Frauds Caught}}{\text{Total Frauds Number}} \tag{13}$$

In fact, the smaller the FPR and the larger the DR, the better the classifier. Fig. 4 shows that when 60% Fraud are caught, the FPRs of CPs, CAEP, Cost-NN, Ada-Cost and Cost-SVM are 6‰, 9‰, 12‰, 18‰ and 30‰, respectively. However, with the increase of FPR, all the classifiers achieve a higher DR. When $FPR = 11‰$, CPs catches 82% Fraud, but CAEP only catches 72%, Cost-NN catches 55%, Ada-cost catches 40%, and Cost-SVM catches 38%. The accuracy test on the data set 2 reveals the similar performance for the above methods (as shown in Fig. 5). Overall, we observe that at the same level of FPR, CPs achieves much higher DR than other methods; with the same DR, CPs obtains much lower FPR. In Fig. 4, when $FPR = 5\%$, CPs outperforms CAEP by 65%, Cost-NN by 80%, Ada-Cost by 90%, and Cost-SVM by 250%. In Fig. 5, for $DR = 60\%$, CPs outperforms CAEP by $FPR = 15‰$, which is only 60% of that of CAEP ($FPR = 24‰$). So, CPs outperforms benchmark methods in accuracy.

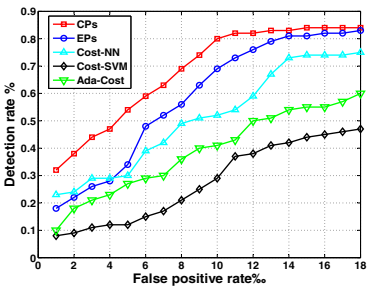

Fig. 4. ROC on DS_1

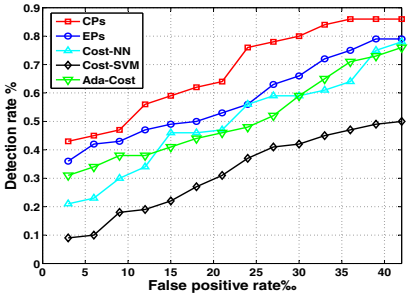

Fig. 5. ROC on DS_2

8.2 Efficiency

We compare the computation time consumed by ConvergMiner and MDB-LL$_{border}$ to evaluate their efficiency on DS_1. For both algorithms, we choose the same level of converging exponent δ=1. The contrast rate in ConvergMiner and MDB-LL$_{border}$ are $1/k$ and the growth rate, respectively. As shown in Fig. 6, with the increase of $1/k$, ConvergMiner gains more benefit from the splitting strategies, by which a huge number of sub-borders are eliminated more easily.

As a result, the number of borders to be split for the further checking decreases dramatically. In addition, MDB-LL$_{border}$ takes more time to process a huge number of the sub-border iterations due to the extremely low support in D_b. For instance, when the contrast rate is 1000, ConvergMiner takes only around 5% of the computation time consumed by MDB-LL$_{border}$. When $1/k$ is larger than 1000, MDB-LL$_{border}$ does not succeed in identifying the itemsets with the support less than 0.0005, but ConvergMiner still works stably. The reason is that ConvergMiner assigns a looser value rather than a fixed value to β (in D_b), and leaves the redundancy to the next step: bound checking and further validation. In summary, the results show that ConvergMiner significantly outperforms MDB-LL$_{border}$ on the imbalanced data set in terms of the efficiency.

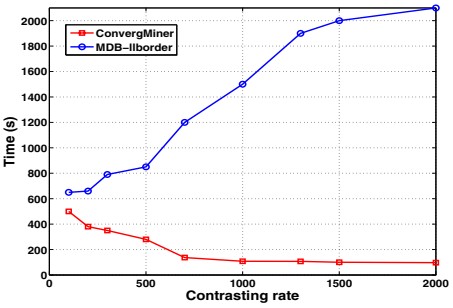

Fig. 6. Efficiency against contrast

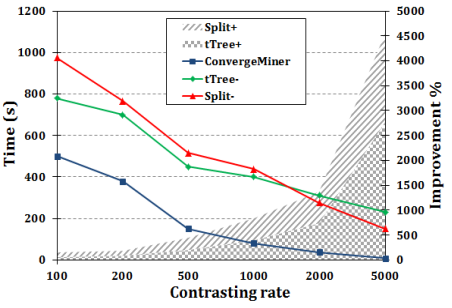

Fig. 7. Effectiveness on strategies

8.3 Effectiveness of the Pruning Strategies

Two benchmark algorithms, i.e. Split- and tTree-, are designed to evaluate the effectiveness of our pruning strategies on DS_1. Split- is a variant of ConvergMiner when the splitting strategies are replaced by a random one at line 17 in Function CheckContrast. tTree- is a variant of ConvergMiner by removing the local supports in T*-tree. Other components remain the same as ConvergMiner. Fig 7 displays the computational time of the three algorithms and the corresponding improvements gained from our pruning strategies under different contrast rates. ConvergMiner obtains the improvement of 50% (upon tTree-) and 90% (upon Split-), when the contrast rate is 100 due to the T*-tree and splitting strategies . With the increase of contrast rate, the improvement rate grows rapidly and reaches 2700% (upon tTree-) and 1700% (upon Split-) when the contrast rate is 5000. Thus, the proposed splitting strategies and T*-tree are effective in enhancing the computational efficiency.

9 Conclusion

In this paper, we introduce a novel type of patterns, i.e. converging patterns (CPs), on the extremely imbalanced data; and propose an efficient algorithm

ConvergMiner equipped with effective splitting strategies and a fast tree index to mine CPs. The experiments show that our algorithm greatly outperforms the state-of-the-art techniques on two real-life large scale imbalanced data sets in terms of accuracy and efficiency. In addition, ConvergMiner exhibits a strong capacity on the scalability and gains more advantages with a larger contrast rate. In future, we are going to mine CPs on the sequential data for online banking.

Acknowledgments. This work is sponsored by the Australian Research Council Discovery grant (DP1096218) and Linkage grant (LP100200774).

References

1. Bayardo, R.J.: Effciently mining long patterns from databases. In: SIGMOD 1998, pp. 85–93 (1998)
2. Beckmann, N., Kriegel, H., Schneider, R., Seeger, B.: The r*-tree: An efficient and robust access method for points and rectangles. In: SIGMOD 1990, pp. 322–331 (1990)
3. Dong, G., Li, J.: Efficient mining of emerging patterns: discovering trends and differences. In: SIGKDD 1999, pp. 43–52 (1999)
4. Dong, G., Zhang, X., Wong, L., Li, J.: CAEP: Classification by aggregating emerging patterns. In: Arikawa, S., Nakata, I. (eds.) DS 1999. LNCS (LNAI), vol. 1721, pp. 30–42. Springer, Heidelberg (1999)
5. Fan, H., Ramamohanarao, K.: Fast discovery and the generalization of strong jumping emerging patterns for building compact and accurate classifiers. TKDE 18(6), 721–737 (2006)
6. Kent, J.: Information gain and a general measure of correlation. Biometrika 70(1), 163–173 (1983)
7. Li, J., Wang, C., Cao, L., Yu, P.S.: Large efficient selection of globally optimal rules on large imbalanced data based on rule coverage relationship analysis. In: SDM 2013 (accepted, 2013)
8. Li, W., Han, J., Pei, J.: CMAR: Accurate and efficient classification based on multiple class-association rules. In: ICDM 2001, pp. 369–376 (2001)
9. Li, Y., Kwok, J., Zhou, Z.: Cost-sensitive semi-supervised support vector machine. In: AI 2010, pp. 500–505 (2010)
10. Liu, X., Zhou, Z.: Training cost-sensitive neural networks with methods addressing the class imbalance problem. TKDE 18(1), 63–77 (2006)
11. Loekito, E., Bailey, J.: Fast mining of high dimensional expressive contrast patterns using binary decision diagrams. In: SIGKDD 2006, pp. 307–316 (2006)
12. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, Los Altos (1993)
13. Sun, Y., Kamel, M.: Cost-sensitive boosting for classification of imbalanced data. Pattern Recognition 40(12), 3358–3378 (2007)