Classification with Streaming Features: An Emerging Pattern Mining Approach*

KUI YU, Simon Fraser University

WEI DING, University of Massachusetts Boston

DAN A. SIMOVICI, University of Massachusetts Boston

HAO WANG, Hefei University of Technology

JIAN PEI, Simon Fraser University

XINDONG WU(✉), Hefei University of Technology and University of Vermont

Many data sets from real-world applications have very high-dimensional or increasing feature space. It is a new research problem to learn and maintain a classifier to deal with very high dimensionality or streaming features. In this paper, we adapt the well-known emerging-pattern based classification models and propose a semi-streaming approach. For streaming features, it is computationally expensive or even prohibitive to mine long emerging patterns, and it is non-trivial to integrate emerging pattern mining with feature selection. We present an online feature selection step, which is capable of selecting and maintaining a pool of effective features from a feature stream. Then in our offline step, separated from the online step, we periodically compute and update emerging patterns from the pool of selected features from the online step. We evaluate the effectiveness and efficiency of the proposed method using a series of benchmark data sets and a real-world case study on Mars crater detection. Our proposed method yields classification performance comparable to the state-of-art static classification methods. Mostly importantly, the proposed method is significantly faster and can efficiently handle data sets with streaming features.

Categories and Subject Descriptors: I.5.2 [**Computing Methodologies**]: Design Methodology

General Terms: Classifier Design and Evaluation, Feature Evaluation and Selection

Additional Key Words and Phrases: Emerging patterns, streaming features, feature selection, classification

*A preliminary version of this paper titled "Mining Emerging Patterns by Streaming Feature Selection" was published in the Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'12), pp. 60-68.

Author's addresses: Kui Yu and Jian Pei, School of Computing Science, Simon Fraser University, Burnaby, V5A 1S6, BC, Canada, email: {kuiy, jpei}@cs.sfu.ca; Wei Ding and Dan A. Simovici, Department of Computer Science, University of Massachusetts Boston, Boston, 02125, USA, email: {ding, dsim}@cs.umb.edu; Hao Wang, Department of Computer Science, Hefei University of Technology, Hefei, 230009, China, email: jsjxwangh@hfut.edu.cn; Xindong Wu(✉ corresponding author), Department of Computer Science, Hefei University of Technology, Hefei, 230009, China and Department of Computer Science, University of Vermont, Burlington, 05405, USA, email: xwu@cs.uvm.edu.

DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

In some real-world applications, we are facing data sets of not only very high dimensionality but also with features that keep arriving, that is, data sets with streaming features. For example, in order to monitor and analyze the environment in different areas, researchers may deploy a set of observation stations in those areas. Each station is treated as an object in the data collected. In other words, the number of data objects is fixed. However, the number of features in temporal domains keeps increasing, since new observation data is collected all the time. Even the data collection rate may not be very high, the dimensionality of the underlying data set can easily reach tens or hundreds of thousands after a while. As another example, in our research project in automatic detection of sub-kilometer craters in high resolution planetary images [Ding et al. 2011], a fixed number of craters are kept being monitored using the latest available high resolution images, which become available and updated over time. Since impact craters are among the most studied geomorphic features in the solar system and they yield information about the past and present geological processes and provide the only tool for measuring relative ages of observed geologic formations [Urbach and Stepinski 2009], it is invaluable to build and maintain robust classification models using the accumulated features in a streaming way extracted from the images available so far.

It is a novel and challenging problem to learn and maintain a classification model on such data with streaming features, where new features keep arriving. Although classification on data streams has been well studied in the data mining and machine learning literature [Aggarwal 2010; Dong et al. 2003], to the best of our knowledge, the existing emerging pattern based classification methods only focus on scenarios where new objects keep arriving and the features are pre-determined. Those methods are orthogonal to the scenarios studied in this paper.

To build and maintain effective classification models on data sets with streaming features, we have to address at least two major challenges. First, as the existing classification methods typically assume a pre-defined space of features, how to handle streaming features in an online manner is a new challenge. This challenge is highly related to feature selection, i.e., how to select and maintain a set of features from a stream of features. Second, how to build and maintain a classification model using the features incrementally selected from a stream of features is another new challenge. Ideally, the model construction and updating should be seamlessly integrated with online feature selection.

In order to tackle high dimensional data with streaming features, we adapt the well-known emerging pattern based classification methods [Dong and Li 1999; Novak et al. 2009]. Emerging patterns are well recognized effective in classifying high-dimensional data, since an emerging pattern can handle a sub-population in a subspace that deliberates a clear discriminative pattern [Dong et al. 1999; Duan et al. 2014]. However, it is still challenging to extend emerging patterns to classify data sets with streaming features. First, it is computationally expensive or even prohibitive to mine long emerging patterns. As features are accumulated over time, many emerging patterns may become long, partially due to the redundancy among features. Second, it is challenging to integrate mining emerging patterns and feature selection. To address the problem of learning classification models on data sets with streaming features, in this paper, we propose a "semi-streaming" approach. Specifically, to tackle the challenge of streaming features, we propose an online feature selection step, which is capable of selecting and maintaining a pool of effective features from a feature stream. Our feature selection step scans features one by one as they are available. Moreover, the proposed feature selection step is specially designed for emerging patterns. To tackle the challenge on classification model construction and maintenance, we propose an offline step. Periodically we can compute and update emerging patterns from the pool of selected features that are picked by the online step. Although the emerging pattern mining step cannot be made online in theory, this step can be conducted only periodically, and can be separated from the online step. In other words, when the classification model needs to be updated, the offline task of emerging pattern mining can take place.

Our "semi-streaming" approach is fundamentally different from applying emerging pattern mining straightforwardly on a data set with streaming features. The online feature selection step substantially reduces the dimensionality of the feature space under which the offline emerging pattern mining step has to handle. This becomes possible only when the feature selection step can handle streaming features in an online manner, and also can select features according to the requirements of emerging pattern mining. Due to the effective online feature selection customized for emerging pattern mining, the emerging patterns mined in the offline step tend to be short, since many redundant and correlated features are reduced before the patterns are mined. This practically facilitates emerging pattern mining dramatically.

Using a series of benchmark data sets, we evaluate the effectiveness and efficiency of the EPSF algorithm, and compare it with the baseline methods and the state-of-the-art methods. The empirical study clearly shows that our method not only achieves high accuracy, but also takes less CPU time than the existing classification methods. Most importantly, the EPSF algorithm can handle data sets with streaming features efficiently. In our real-world case study, we evaluate the EPSF algorithm with crater detection from planetary images, and the experimental results show that our method is not only highly comparable with the existing crater detection algorithms, but also produces a concise set of emerging patterns that are interpretable for domain scientists to understand the crater data.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 gives the preliminaries and Section 4 presents our approach. Section 5 reports our experimental re-sults. Finally, Section 6 concludes the paper.

2. RELATED WORK

Our work relates to emerging pattern mining and feature selection, and thus we will give a brief review of emerging patterns and feature selection in this section.

Dong and Li [Dong and Li 1999] first introduced emerging patterns (EPs) to represent strong contrasts between different classes of data. In addition, a Jump Emerging Pattern (JEP for short) is a special type of EPs whose support increases from zero in one class to non-zero in the other class [Li et al. 2001a]. Like other patterns composed of conjunctive combinations of feature-value pairs [Wang and Karypis 2005; Trépos et al. 2013; Sahoo et al. 2014], EPs can be easily understood and used directly in a wide range of applications [Song et al. 2014; Wang et al. 2013a], such as failure detection [Lo et al. 2009] and discovering knowledge in gene expression data [Fang et al. 2012].

Dong et al. [Dong et al. 1999] proposed the first EP classifier, called CAEP (Classification by Aggregating Emerging Patterns). Based on CAEP, Li et al. [Li et al. 2001a] proposed a JEP-classifier which is distinct from the CAEP classifier. The JEP-classifier uses JEPs exclusively because JEPs discriminate between different classes more strongly than any other type of EPs. Meanwhile, Li et al. [Li et al. 2000] also presented a lazy EP classifier based on an instance-based EP discovery, called DeEPs, to improve the efficiency and accuracy of CAEP and JEP-classifier. In addition, Fan and Ramamohanarao [Fan and Ramamohanarao 2006] proposed a robust EP-classifier, called SJEP-classifier, using a strong JEP. A strong JEP from class C_1 to class C_2 satisfies two conditions: (1) the support of itemset X is zero in C_1 but non-zero and satisfies a minimal support threshold in C_2, and (2) any proper subset of X does not satisfy condition (1). The SJEP-classifier integrates the CP-tree miner into the EP classifier, and uses much fewer JEPs than the JEP-classifier. The disadvantage of JEP and SJEP classifiers is that if a data set contains no or very few JEP and SJEP patterns, then it will cause the classification performance of JEP and SJEP classifiers significantly reduced.

The well-known bottleneck of EP classifiers is that they are computationally prohibitive to dealing with a data set with more than sixty dimensions without prior feature set reduction until the ZBDD (Zero-suppressed Binary Decision Diagrams) EP-miner was proposed [Loekito and Bailey 2006]. The ZBDD EP-miner can deal with a relative high-dimensional data set, but like previous EP mining approaches, it still suffers from an explosive number of EPs, even with a rather high support threshold. Accordingly, it is still a challenging research issue to build a robust EP classifier from high dimensionality.

Feature selection has been generally viewed as a problem of searching for a minimal subset of features in high-dimensional data that leads to the most accurate prediction model [Liu and Yu 2005]. There are two types of feature selection methods proposed in the literature, that is, batch methods and online methods.

A batch method has to access the entire feature set on the training data and performs a global search for the best feature at each round [Brown et al. 2012]. Contrast to the batch methods, online feature selection can be conducted in an online manner. Recently, Wang et al. [Wang et al. 2013b] proposed an OSF algorithm for online feature selection. The OFS algorithm assumes data instances keep arriving, and performs feature selection on each data instance as it is available. Different from OFS, the Fast-OSFS and alpha-investing algorithms were proposed to deal with the scenarios where features keep arriving but the number of data instances is fixed [Zhou et al. 2006; Wu et al. 2010; Wu et al. 2013]. Zhou et al. [Zhou et al. 2006] presented Alpha-investing which sequentially considers new features as the addition to a predictive model by modeling the candidate feature set as a dynamically generated stream. However, Alpha-investing requires prior information of the original feature set and never evaluates redundancy among the selected features as time goes. To fix the drawbacks, Wu et al. [Wu et al. 2010; Wu et al. 2013] presented the OSFS (Online Streaming Feature Selection) algorithm and its faster version, the Fast-OSFS algorithm.

In a recent study, Yu et al. [Yu et al. 2013] integrated local causal-structure learning into EP mining to help reduce high dimensionality. The study has shown that this integration can efficiently extract a minimum number of sets of strongly predictive patterns from high-dimensional data and get highly accurate EP classifiers. Different from the work of [Yu et al. 2013] that requires a complete set of features available beforehand, the proposed algorithm in this paper is capable of dealing with a high-dimensional data set with streaming features. Our new algorithm can mine emerging patterns from data sets of not only very high dimensionality but also with features kept arriving, that is, data sets with streaming features.

Table I. Summary on mathematical notations

Notation	Mathematical meanings
D	training data set
D_l, D_m	subsamples of D
F	a feature set from D
F_i	a single feature (attribute), $F_i \in F$
f_i	a discrete value of F_i
$Dom(.)$	$Dom(F_i)$ denotes all discrete values of F_i
S	a feature subset $S \subseteq F$
C	the class attribute
C_i	a class label, $C_i \in C$
I	a set of all items from F
X	an itemset, $X \in I$
$count_D(X)$	the number of instances in D that supports X
$\|.\|$	$\|D\|$ returns the number of instances in D
$support_D(X)$	the support value of X on D
$GR_{D_l \to D_m}(X)$	the growth rate of X from D_l to D_m
ρ	a threshold of $GR_{D_l \to D_m}(X)$
e	an emerging pattern
E_i	a set of emerging patterns
$Rateimp(e)$	the growth rate improvement of e
$P(.\|.)$	$P(C = C_j \| S = s)$ denotes the posterior probability of C_j conditioned on s
t	a time point
T	a test instance
$CMB(C)_t$	a Markov blanket of C selected at time t

3. NOTATIONS AND DEFINITIONS

3.1. Emerging Patterns

Consider a training data set D is defined upon a feature set F and the class attribute C. F contains N features, i.e., $F = \{F_1, F_2, \cdots, F_N\}$. For $\forall F_i \in F$ we assume that it is in a discrete domain $dom(F_i)$. Let I be the set of all items, i.e., $I = \{F_i = f_i | F_i \in F, f_i \in Dom(F_i)\}$, the class attribute $C = \{C_1, C_2, \cdots, C_K\}$ be a finite set of K distinct class labels, and X be an itemset and $X \subseteq I$. The data set D can be partitioned into $D_1, D_2, \cdots, D_K$, where D_j consists of instances with class label $C_j, j = 1, \cdots, K$. The mathematical notations used in this paper are summarized in Table I.

The Support and Growth Rate (GR for short) of an itemset X, and an emerging pattern from D_l to $D_m (l, m = 1, \cdots, K, and\ l \neq m)$, are defined as follows.

Definition 3.1 (*Support*).

$$support_D(X) = count_D(X)/|D| \tag{1}$$

where $count_D(X)$ is the number of instances in D containing X and $|D|$ is the number of instances in D. ■

Definition 3.2 (*GR: Growth Rate*). [Dong and Li 1999]

$$GR_{D_l \to D_m}(X) = support_{D_m}(X)/support_{D_l}(X) \tag{2}$$

If $support_{D_m}(X) = 0$ and $support_{D_l}(X) = 0$, then $GR_{D_l \to D_m}(X) = 0$; if $support_{D_m}(X) \neq 0$ but $support_{D_l}(X) = 0$, then $GR_{D_l \to D_m}(X) = \infty$. ■

Definition 3.3 (*EP: Emerging Pattern*). [Dong and Li 1999] Given a threshold $\rho > 1$, an EP from D_l to D_m is an itemset X where $GR_{D_l \to D_m}(X) \geq \rho$. ■

An emerging pattern e from D_l to D_m is also called an EP of D_m. If $GR(e) = \infty$, e is called a Jumping EP (JEP). The goal of EP mining is to extract EP set E_i for class C_i

which consists of EPs from $\{D - D_i\}$ to D_i, given a minimum support threshold and a minimum growth rate threshold.

A positive growth rate improvement threshold is introduced to ensure a concise and representative set of EPs which are not subsumed by one another and consist of items that are strong contributors to their predictive power. The growth rate improvement can also help to reduce the search space by eliminating EPs that are redundant. Here is the definition of growth rate improvement.

Definition 3.4 (*Growth Rate Improvement*). [Zhang et al. 2000a] Given an EP e, the growth rate improvement of e, $Rateimp(e)$, is defined as the minimum difference between its growth rate and the growth rates of all of its subsets,

$$Rateimp(e) = \min(\forall e' \subset e, GR(e) - GR(e')). \tag{3}$$

■

3.2. Feature Relevance in Feature Selection

In feature selection, a feature space F in general can be divided into three disjoint groups, namely, strongly relevant, weakly relevant, and irrelevant features [Koller and Sahami 1995]. The goal of feature selection is to select a subset from F without performance degradation on prediction models. In the following definitions, $P(C = C_j|S = s)$ is the posterior probability of class C_j given a set of values of s of a subset S.

Definition 3.5 (*Conditional Independence*). Two distinct features $F_i \in F$ and $F_k \in F$ are conditionally independent on a feature subset $S \subseteq F - \{F_i \cup F_k\}$, iff there exists an assignment of values f_i and f_k, s.t.

$$P(F_i = f_i|F_k = f_k, S = s) = P(F_i = f_i|S = s). \tag{4}$$

■

Definition 3.6 (*Strong Relevance*). A feature F_i is strongly relevant to the class attribute C, iff there exists an assignment of values f_i, C_j, and s for which $P(S = s, F_i = f_i) > 0$,

$$\forall S \subseteq F - \{F_i\} \; s.t. \; P(C = C_j|S = s, F_i = f_i) \neq P(C = C_j|S = s). \tag{5}$$

■

Definition 3.7 (*Weak Relevance*). A feature F_i is weakly relevant to the class attribute C, iff it is not strongly relevant, and $\forall f_i, C_j$, and s for which $P(S = s) > 0$,

$$\exists S \subset F - \{F_i\} \; s.t. \; P(C = C_j|S = s) \neq P(C = C_j|S = s, F_i = f_i). \tag{6}$$

■

Definition 3.8 (*Irrelevance*). A feature F_i is irrelevant to the class attribute C, iff it is neither strongly nor weakly relevant, and there exists an assignment of values f_i, C_j, and s for which $P(S = s, F_i = f_i) > 0$,

$$\forall S \subseteq F - \{F_i\} \; s.t. \; P(C = C_j|S = s, F_i = f_i) = P(C = C_j|S = s). \tag{7}$$

■

Definition 3.9 (*Markov Blanket*). [Koller and Sahami 1995] The Markov blanket of feature F_i, denoted as $M_i \subset F - \{F_i\}$ makes all other features independent of F_i given M_i, that is,

$$\forall F_k \in F - (M_i \cup \{F_i\}) \; s.t. \; P(F_i|M_i, F_k) = P(F_i|M_i). \tag{8}$$

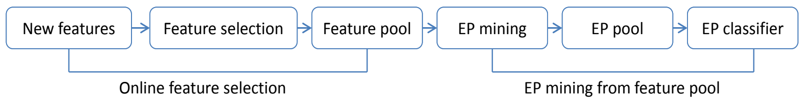


Fig. 1. The framework of a semi-streaming approach

■

With Markov blankets, weakly relevant features can be divided into redundant features and non-redundant features [Yu and Liu 2004].

Definition 3.10 (*Redundant Feature*). A feature is redundant hence should be removed from F, iff it is weakly relevant and has a Markov blanket within F. ■

4. MINING EMERGING PATTERN WITH STREAMING FEATURES

It is computationally expensive to evaluate the complete item combinations for a high-dimensional data set. To mitigate this problem, we propose to effectively prune the feature space before EP mining. Recently, Wu et al. [Wu et al. 2013] proposed streaming feature selection to deal with data with streaming features. In this model, the number of data instances is fixed, while features keep arriving and each feature is evaluated upon its arrival. Compared to traditional feature selection, the strength of streaming feature selection is that the number of features is no longer required to be fixed in advance. Feature selection is streamlined and conducted online to be able to deal with streaming features.

Although the emerging pattern mining cannot be made online in theory, in this paper, we propose a "semi-streaming" approach to bridge streaming feature selection and EP mining to learn and maintain an EP classification model on data with streaming features.

Definition 4.1 (*Semi-streaming approach*). A semi-streaming approach includes an online feature selection step customized for an offline step to periodically mine emerging patterns from the features that are picked by the online step. ■

In the semi-streaming approach, the online feature selection step is to select and maintain a pool of effective features from streaming features by scanning features one by one as they are available, while an offline step is to construct and update an EP classifier by periodically mining emerging patterns from the pool of selected features that are picked by the online step. Definition 4.1 indicates that although the emerging pattern mining cannot be fully online, the offline step only needs to be conducted periodically. When the EP classifier needs to be updated, the offline task of emerging pattern mining will take place.

The framework of the semi-streaming approach is illustrated in Figure 1. At the first stage, we present an online feature selection step, which is capable of selecting and maintaining a pool of effective features from a feature stream. Our feature selection step processes features one by one as they are available. At the second stage, we propose an offline step. Periodically we can compute and update emerging patterns from the pool of selected features that are picked by the online step. There are two key research problems to be addressed:

(1) How to build an influential feature candidate pool to be used for EP mining as features are available over time;

(2) How to build an EP pool to be used for classification by extracting EPs from this influential feature pool.

4.1. Online Building an Influential Feature Pool

In Figure 1, the feature pool shall keep the features which are only useful for producing predictive EPs, and it may be updated over time as features are available one by one. To customize the online feature selection step for efficient emerging pattern mining, we must be able to evaluate the degree of feature relevance with the discriminative power of EPs. We have theoretically proved the association of causal relevance in causal Bayesian networks and EP discriminability in EP mining [Yu et al. 2013]. Here we provide theoretically an analysis on the relationships between feature relevance (irrelevant features, strongly relevant features, and redundant features) and EP discriminability (non-EPs, strongly predictive EPs, and redundant EPs) in the following propositions.

As discussed in Definitions 3.2 and 3.3, the discriminability of an EP is determined by its support value and growth rate. Proposition 4.2 establishes the relations between non-EPs and irrelevant features.

PROPOSITION 4.2. *For* $\forall F_i \in F$*,* $\forall f_i \in dom(F_i)$*, and* $\forall C_j \in dom(C)$*,* $GR_{\{D-D_j\}\to D_j}(F_i = f_i) = 1$ *holds iff* F_i *is irrelevant to* C*.*

PROOF. Assume a data set D has two classes: positive class C_p and negative class $C_n, C = \{C_p, C_n\}$. D_p represents C_p class data, D_n represents C_n class data, $sup_{D_p}(F_i = f_i)$ is the support value of the itemset $\{F_i = f_i\}$ in D_p, and $sup_{D_n}(F_i = f_i)$ is its support value in D_n. Then $GR(F_i = f_i)$ from D_n to D_p is calculated as follows.

$$\begin{aligned} GR_{D_n\to D_p}(F_i = f_i) &= \frac{sup_{D_p}(F_i=f_i)}{sup_{D_n}(F_i=f_i)} \\ &= \frac{P(F_i=f_i|C=C_p)}{P(F_i=f_i|C=C_n)} \\ &= \frac{P(F_i=f_i,C=C_p)}{P(C=C_p)} \Big/ \frac{P(F_i=f_i,C=C_n)}{P(C=C_n)} \\ &= \frac{P(C=C_p|F_i=f_i)P(F_i=f_i)}{P(C=C_p)} \Big/ \frac{P(C=C_n|F_i=f_i)P(F_i=f_i)}{P(C=C_n)} \\ &= \frac{P(C=C_p|F_i=f_i)}{P(C=C_n|F_i=f_i)} \bullet \frac{P(C=C_n)}{P(C=C_p)} \end{aligned}$$

If $GR_{D_n\to D_p}(F_i = f_i) = 1$, then the follows holds.

$$\frac{P(C = C_p)}{P(C = C_n)} = \frac{P(C = C_p|F_i = f_i)}{P(C = C_n|F_i = f_i)}$$

As $P(C = C_p) + P(C = C_n) = 1$ and $P(C = C_p|F_i = f_i) + P(C = C_n|F_i = f_i) = 1$, we get

$$P(C = C_p|F_i = f_i) = P(C = C_p)$$

(with $\frac{a}{b} = \frac{c}{d}$ equivalent to $\frac{a}{b+a} = \frac{c}{d+c}$), as well as $P(C = C_n|F_i = f_i) = P(C = C_n)$.

According to Definition 3.8, for any assignments $f_i \in dom(F_i)$ and $C_j \in dom(C)$ to F and C, $P(C = C_j|F_i = f_i) = P(C = C_j)$ holds, therefore F_i is irrelevant to C. Similarly, from D_p to D_n, if $GR_{D_p\to D_n}(F_i = f_i) = 1$, we can also prove that F_i is irrelevant to C.

On the other hand, if F_i is irrelevant to C, we get

$$\begin{aligned} GR_{D_n\to D_p}(F_i = f_i) &= \frac{P(C=C_p|F_i=f_i)}{P(C=C_n|F_i=f_i)} \bullet \frac{P(C=C_n)}{P(C=C_p)} \\ &= \frac{P(C=C_p|F_i=f_i)}{P(C=C_n|F_i=f_i)} \bullet \frac{1-P(C=C_p)}{P(C=C_p)} \\ &= 1 \end{aligned}$$

Thus, Proposition 4.2 is proven. □

According to Definition 3.4, for an EP e, if we can find an $e' \subset e$ to make $Rateimp(e) \leq 0$, then e is a redundant EP, and might be replaced by a subset within e. Thus, avoiding generation of those redundant EPs in advance will improve search efficiency. Proposition 4.3 explains the relationship between feature redundancy and EP redundancy.

PROPOSITION 4.3. ***For*** $\exists F_i \in F$, $\exists S \subset F - F_i, \forall f_i \in dom(F_i), \forall s \subset \bigcup_{k=1}^{|S|} dom(S_k)$, ***and*** $\forall C_j \in dom(C), GR_{\{D-D_j\} \to D_j}(F_i = f_i, S = s) = GR_{\{D-D_j\} \to D_j}(S = s)$ ***holds, iff*** F_i ***is redundant to*** C ***conditioning on the subset*** S.

PROOF. $GR(F_i = f_i, S = s)$ from D_n to D_p is calculated as follows.

$$\begin{aligned} GR_{D_n \to D_p}(F_i = f_i, S = s) &= \frac{sup_{D_p}(F_i{=}f_i,S{=}s)}{sup_{D_n}(F_i{=}f_i,S{=}s)} \\ &= \frac{P(F_i{=}f_i,S{=}s|C{=}C_p)}{P(F_i{=}f_i,S{=}s|C{=}C_n)} \\ &= \frac{P(F_i{=}f_i,S{=}s,C{=}C_p)}{P(C{=}C_p)} / \frac{P(F_i{=}f_i,S{=}s,C{=}C_n)}{P(C{=}C_n)} \\ &= \frac{P(C{=}C_p|F_i{=}f_i,S{=}s)P(F_i{=}f_i,S{=}s)}{P(C{=}C_p)} / \frac{P(C{=}C_n|F_i{=}f_i,S{=}s)P(F_i{=}f_i,S{=}s)}{P(C{=}C_n)} \\ &= \frac{P(C{=}C_p|F_i{=}f_i,S{=}s)}{P(C{=}C_n|F_i{=}f_i,S{=}s)} \bullet \frac{P(C{=}C_n)}{P(C{=}C_p)} \end{aligned}$$

From D_n to D_p, $GR(S = s) = P(S = s|C = C_p)/P(S = s|C = C_n)$

$$\begin{aligned} \frac{P(C{=}C_p|F_i{=}f_i,S{=}s)}{P(C{=}C_n|F_i{=}f_i,S{=}s)} \bullet \frac{P(C{=}C_n)}{P(C{=}C_p)} &= \frac{P(S{=}s|C{=}C_p)}{(P(S{=}s|C{=}C_n)} \\ \frac{P(C{=}C_p|F_i{=}f_i,S{=}s)}{P(C{=}C_n|F_i{=}f_i,S{=}s)} &= \frac{P(S{=}s|C{=}C_p)P(C{=}C_p)}{(P(S{=}s|C{=}C_n)P(C{=}C_n)} \\ \frac{P(C{=}C_p|F_i{=}f_i,S{=}s)}{P(C{=}C_n|F_i{=}f_i,S{=}s)} &= \frac{P(C{=}C_p|S{=}s)}{P(C{=}C_n|S{=}s)} \end{aligned}$$

Using the same reasoning in proving Proposition 4.2, we can get two equations $P(C = C_p|F_i = f_i, S = s) = P(C = C_p|S = s)$ and $P(C = C_n|F_i = f_i, S = s) = P(C = C_n|S = s)$. By Definitions 3.9 and 3.10, we can find a subset $S \subset F$ as a Markov blanket of F_i, and for any assignments $f_i \in dom(F_i), s \subseteq dom(S)$ and $C_j \in dom(C)$ to F_i, S and C, $P(C = C_j|F_i = f_i, S = s) = P(C = C_j|S = s)$ holds, and so F_i is redundant to C given S.

On the other hand, if F_i is redundant to C, from D_n to D_p, then the follows holds.

$$\begin{aligned} GR_{D_n \to D_p}(F_i = f_i, S = s) &= \frac{P(C{=}C_p|F_i{=}f_i,S{=}s)}{P(C{=}C_n|F_i{=}f_i,S{=}s)} \bullet \frac{P(C{=}C_n)}{P(C{=}C_p)} \\ &= \frac{1-P(C{=}C_n|F_i{=}f_i,S{=}s)}{P(C{=}C_n|F_i{=}f_i,S{=}s)} \bullet \frac{P(C{=}C_n)}{P(C{=}C_p)} \\ &= \frac{1-P(C{=}C_n|S{=}s)}{P(C{=}C_n|S{=}s)} \bullet \frac{P(C{=}C_n)}{P(C{=}C_p)} \\ &= \frac{P(C{=}C_p|S{=}s)}{P(C{=}C_n|S{=}s)} \bullet \frac{P(C{=}C_n)}{P(C{=}C_p)} \\ &= \frac{P(S{=}s|P(C{=}C_p)}{P(S{=}s|C{=}C_n)} \\ &= GR_{D_n \to D_p}(S = s) \end{aligned}$$

Thus, Proposition 4.3 is proven. □

Proposition 4.3 shows that if F_i is redundant to C conditoned on a subset S, then an itemset $\forall f_i \in dom(F_i)$ together with an itemset $\forall s \subset \bigcup_{k=1}^{|S|} dom(S_k)$ contains the same predictive information as the itemset $\forall s \subset \bigcup_{k=1}^{|S|} dom(S_k)$.

With Propositions 4.2 and 4.3, as features are processed in a sequential scan, we can online build an influential feature pool and discarding irrelevant and redundant features to avoid generating non-EPs or redundant EPs in the EP mining process later on.

Once we understand the feature relevance with the discriminative power of EPs, the next step is to understand how to online build this influential feature pool, and then how to adjust the feature pool once a new feature is added into the pool. To build an influential feature pool, we need to online assess whether a new feature is irrelevant or not. If so, it is discarded. If not, we use Proposition 4.4 proposed by Wu et al. [Wu et al. 2013] to handle this newly arrived feature.

PROPOSITION 4.4. *A current Markov blanket of C at time t is denoted as $CMB(C)_t$. Assume a new feature F_i at time $t+1$ is weakly relevant to C, if $\exists S \subseteq CMB(C)_t$ such that $P(C|F_i, S) = P(C|S)$, then F_i can be discarded.*

After F_i is added into $CMB(C)$, we must check whether any existing features in the feature pool become redundant. We shall use Proposition 4.5 proposed by Wu et al. [Wu et al. 2013] to online update the current feature pool, that is, determining which of the existing features in the current feature pool can be removed as F_i is added.

PROPOSITION 4.5. *With $CMB(C)_t$ at time t, a new feature F_i arrives at time $t+1$, and there does not exist any $MB(F_i)$ within $CMB(C)_t$. If $\exists Y \in CMB(C)_t$ and $\exists S \subseteq \{CMB(C)_t \cup F_i\} - \{Y\}$ s.t. $P(C|Y,S) = P(C|S)$, then Y can be removed from $CMB(C)_t$.*

4.2. Building an EP pool

The EP pool stores the candidate EPs which are mined from the feature pool. To make the EP pool correspond to the changes of the feature pool, we divide the construction of the EP pool into two steps. One step is to online build 1-itemset EP pool. This 1-itemset EP pool should be updated correspondingly as the feature pool is updated. The other step is offline but periodically mines all EPs from the 1-itemset EP pool to construct an EP classifier.

4.2.1. Online building a 1-itemset EP pool

(1) **Online building 1-itemset EP pool**. As a new feature F_i arrives, we first assesses whether it is irrelevant; and if so, it is discarded. Otherwise, we evaluate whether it is redundant to C by Proposition 4.4; and if so, it is also discarded. If not, it is added to the feature pool $CMB(C)$. And then, the EPSF algorithm (discussed in details in Section 4.4) converts feature F_i into a set of itemsets I_{F_i} and has a mapping between I_{F_i} and F_i, named map_form. This mapping can guarantee that itemsets contain items mapped from the same feature, and their supersets should be pruned. With I_{F_i} and the mapping, EPSF divides the training data by class, mines EPs for each class and stores the EPs in a candidate EP pool named CEP.

(2) **Online updating 1-itemset EP pool**. Due to $F_i's$ inclusion, EPSF updates the feature pool $CMB(C)$ by removing redundant features according to Proposition 4.5. If feature Y is removed from $CMB(C)$, we online update CEP and map_form by removing EPs in CEP generated from Y and the mapping between itemsets I_y and Y in map_form, respectively.
To update $CMB(C)$, EPSF checks all subsets within $CMB(C)$ to re-examine the redundancy of each feature in $CMB(C)$. To improve this updating efficiency, we only validate the redundancy of each originally existing feature in $CMB(C)$ by testing the subsets created by the inclusion of the new feature F_i.

4.2.2. Periodically mining all EPs from the 1-itemset EP pool. With the current 1-itemset EP pool, we propose an offline step to periodically mine all EPs, for an EP classifier construction and maintenance. At this step, we can periodically compute and update emerging patterns from the 1-itemset EP pool that is picked by the online step. Although the emerging pattern mining step cannot be made online in theory, this step can be conducted only periodically, and can be separated from the online step. In other words, when the classification model needs to be updated, an offline task of emerging pattern mining can take place.

4.3. A Score function for EP classifiers

When applying EPs to classification, we get all the EPs of each class C_i in a training set. With the EPs for K classes, we derive K scores for a test instance T, one score per class, by feeding the EPs of each class into a scoring function. In this paper, we use the score function based on information theory proposed by [Zhang et al. 2000b] for classifying unlabeled instances, since this function is simpler and more efficient than the score function proposed by [Dong et al. 1999] by avoiding computing the base score for each class. Zhang et al. [Zhang et al. 2000b] defined the score function of a test instance T by the following Eq.(9).

$$L(T|C_i) = -\sum_{k=1}^{|E_i|} \log_2 P(X_k|C_i), X_k \in E_i \ and \ X_k \in T \tag{9}$$

where $|E_i|$ is the number of emerging patterns in the EP set E_i, and X_k is an emerging pattern in E_i. The test instance T is assigned class label C_i when $L(T|C_i)$ is the minimum. Given an itemset X, $P(X_i)$ is approximately computed by Eq.(10).

$$P(X|C_i) = (|X \cap C_i| + 2|X|/|D|)/(|C_i| + 2) \tag{10}$$

where $|X \cap C_i|$ is the number of training instances belonging to class C_i and containing X, $|X|$ is the total number of training instances containing X, $|D|$ is the total number of training instances, and $|C_i|$ is the number of training instances for class C_i.

In addition, to ensure that we can always find a partition for an instance, all single-item itemsets of each class whether they satisfy the given thresholds or not are taken into account when Eq.(9) is used to classify a test instance.

4.4. The EPSF Algorithm

To integrate online feature selection and EP mining, we propose the algorithm EPSF, mining Emerging Patterns with Streaming Features, as shown in Algorithm 1.

EPSF online builds two pools: a feature pool and a 1-itemset EP pool, and periodically computes and updates emerging patterns from the 1-itemset EP pool for an EP classification model construction and maintenance. As a new feature arrives, if it is added into the feature pool, EPSF transforms it into a set of itemsets, and online mines 1-itemset EPs which are then added into the 1-itemset EP pool. As the dimensions are processed one by one, in order to quickly respond to this change, EPSF only online mines 1-itemset EPs for each feature available, and updates the current 1-itemset EP pool correspondingly with the change of the feature pool. Then EPSF periodically computes and updates emerging patterns from the 1-itemset EP pool.

As for the EPSF algorithm in its early version (we call it Pre-EPSF) [Yu et al. 2012], the Pre-EPSF algorithm needs to check all subsets within $CMB(C)$ to re-examine the redundancy of each feature in $CMB(C)$ due to a new feature $F_i's$ inclusion. In Step 28 of Algorithm 1, the EPSF algorithm in the current version validates the redundancy

ALGORITHM 1: The EPSF Algorithm

```
1  Initialize the minimum support threshold $\alpha$, growth rate threshold $\rho$, and CMB(C)={};
2  repeat
3      Input a new feature $X$;
4      /*Discard irrelevant features*/;
5      if $P(C|X) = P(C)$ then
6          Discard $X$ and goto step 41;
7      end
8      /*Remove redundant features*/;
9      if $\exists S \subset CMB(C)\ s.t.\ P(C|X,S) = P(C|S)$ then
10         Go to step 41;
11     end
12     /*Add $X$ into the current feature pool $CMB(C)$*/;
13     $CMB(C) = CMB(C) \cup \{X\}$;
14     /*Convert feature $X$ into a set of itemsets*/;
15     $I_X = convert(X), I_X \in Dom(X)$;
16     /*Map between $I_X$ and $X$*/;
17     $map\_form$=mapping($X$,$I_X$);
18     for $i = 1 : |C|$ do
19         /*$|C|$ denotes the number of classes*/;
20         /*Mine 1-itemset EPs for each class with the thresholds $\alpha$ and $\rho$*/;
21         $EP_i$=mineEP($I_X$, $\alpha$, $\rho$);
22         /*Add $EP_i$ to the current EP pool $CEP$*/;
23         $CEP = CEP \cup EP_i$;
24     end
25     /*Update $CMB(C)$*/;
26     for each feature $Y$ within $CMB(C)$ excluding $X$ do
27         /*Find $S \subset CMB(C)$ containing $X$*/;
28         if $\exists S \subset CMB(C)\ s.t.\ P(C|Y,S) = P(C|S)$ then
29             $CMB(C) = CMB(C) - Y$;
30             /*Update $CEP$ by removing EPs generated from feature $Y$*/;
31             for each $y \in I_y$ do
32                 if $y \in CEP$ then
33                     $CEP = CEP - y$;
34                 end
35             end
36             /*Update $map\_form$*/;
37             $map\_form = map\_form(I_y)$;
38         end
39     end
40     Periodically mine all EPs from $CEP$ with $map\_form$;
41 until No more features are available;
42 Classify unlabeled instances by the mined EPs using Eq.(9);
```

of each originally existing feature in $CMB(C)$ by only checking the subsets created by the inclusion of the new feature F_i at each time point. By avoiding checking all subsets within $CMB(C)$ at each time point, the revised EPSF significantly reduces the number of subsets that need to be checked, and thus improves the updating efficiency by avoiding performing some unnecessary calculation.

Using the Balloon data set from UCI Repository of Machine Learning Databases [Blake and Merz 1998], we give an illustrating example to explain the EPSF algorithm. In Table II, the Balloon data set includes 4 features (*color*, *size*, *act* and *age*) and one class attribute (*inflated*) with 20 samples. Assuming the input

Table II. The Balloon data set

color	size	act	age	inflated
yellow	small	stretch	adult	True
yellow	small	stretch	child	True
yellow	small	dip	adult	True
yellow	large	stretch	adult	True
yellow	large	stretch	child	True
yellow	large	dip	adult	True
purple	small	stretch	adult	True
purple	small	stretch	child	True
purple	small	dip	adult	True
purple	large	stretch	adult	True
purple	large	stretch	child	True
purple	large	dip	adult	True
yellow	small	dip	child	False
yellow	small	dip	child	False
yellow	large	dip	child	False
yellow	large	dip	child	False
purple	small	dip	child	False
purple	small	dip	child	False
purple	large	dip	child	False
purple	large	dip	child	False

Table III. CEP after adding *act* for class F

Candidate EP	Support(class T)	Support(class F)	$GR_{T\rightarrow F}(e)$
{act=dip}	0.33	1	3

Table IV. CEP after adding *act* for class T

Candidate EP	Support(class F)	Support(class T)	$GR_{F\rightarrow T}(e)$
{act=stretch}	0	0.67	∞

order of features is *color*, *size*, *act* and *age*, and the G^2 test [Spirtes et al. 2000] is employed to compute conditional independence defined in Definition 3.5 in Section 3.2 to determine feature relevance and feature redundancy, the EPSF algorithm is traced as follows.

(1) As feature *color* arrives, at Step 6 in Algorithm 1, *color* is discarded as an irrelevant feature. And then EPSF processes the next feature *size* directly. Since feature *size* is also independent to the class attribute *inflated*, *size* is also discarded, and will never be considered again.

(2) As feature *act* is available, at Step 5 in Algorithm 1, *act* is regarded as a relevant feature. And then Step 9 checks whether *act* is a redundant feature given the current feature pool $CMB(C)$. If so, *act* will be discarded and EPSF will consider a next feature available; if not, *act* will be added to $CMB(C)$. Since the current feature pool $CMB(C)$ is empty, *act* is added to $CMB(C)$ at Step 13 and $CMB(C) = \{act\}$. At Steps 14 to 17, feature *act* is converted into a set of itemsets, that is, $\{act = dip\}$ and $\{act = stretch\}$. From Steps 18 to 24, EPSF mines 1-itemset EPs of *act* from two classes and stores those 1-itemsets into the current EP pool, CEP, as shown in Tables III and IV, using the minimum support threshold 0.2 and the growth rate threshold $\rho > 1$. Since the current feature pool $CMB(C)$ only contains *act*, so Steps 25 to 39 are not implemented and EPSF directly processes the next feature *age*.

(3) As feature *age* comes, *age* is considered a relevant feature at Step 5. At Step 9, given the current feature pool $CMB(C) = \{act\}$, $P(inflated, age|act) \neq P(inflated, age)$. Accordingly *age* is added to $CMB(C)$ at Step 13, and $CMB(C) =$

Table V. CEP after adding *age* for class F

Candidate EP	Support(class T)	Support(class F)	$GR_{T\to F}(e)$
{act=dip}	0.33	1	3
{age=child}	0.33	1	3

Table VI. CEP after adding *age* for class T

Candidate EP	Support(class F)	Support(class T)	$GR_{F\to T}(e)$
{act=stretch}	0	0.67	∞
{age=adult}	0	0.70	∞

$\{act, age\}$. At Steps 14 to 17, *act* is converted into a set of itemsets, i.e., $\{age = child\}$ and $\{age = adult\}$. From Steps 18 to 24, the current EP pool CEP is updated as shown in Tables V and VI.

(4) Due to $age's$ addition to $CMB(C)$, Steps 25 to 39 further check whether *act* is a redundant feature. If so, *act* will be removed from $CMB(C)$, and its corresponding 1-itemsets in CEP also will be removed.

(5) With the current EP pool CEP, EPSF periodically mines all EPs by employing a level-wise, candidate generation-and-test approach to mine EPs (we use the ConsEPMiner algorithm [Zhang et al. 2000a]), then uses them to classify test instances later on.

In summary, compared to the CE-EP algorithm and other existing EP algorithms, we are the first group to mine EPs from data with streaming features. With an effective online feature selection customized for emerging pattern mining, the EPSF algorithm is designed for data sets with streaming features, as it does not need to store the whole data in the memory to mine EPs. This practically facilitates emerging pattern mining dramatically.

Moreover, EPSF can online mine EPs from the features available so far and can consume new features in an online manner as they become available. Accordingly, the EPSF algorithm allows more expensive calculation, including feature redundancy checking (step 9), emerging pattern mining (steps 14-24), $CMB(C)$ and CEP updating (steps 25-39) all to be online conducted within the current $CMB(C)$, which is usually much smaller than the whole feature space.

5. EXPERIMENT RESULTS

5.1. Experiment Setup

In order to thoroughly evaluate the proposed EPSF algorithm, 16 data sets (in Table VII) are selected including four from the UCI machine learning repository (the first four) [Blake and Merz 1998], four very high-dimensional biomedical data sets (*hiva*, *ovarian-cancer*, *lymphoma*, and *breast-cancer*), four NIPS 2003 feature selection challenge data sets (*madelon*, *arcene*, *dorothea*, and *dexter*), and four frequently studied public microarray data sets (the last four).

Our comparative study has the following systematical design, using 10-fold cross-validation for all the experiments unless specified.

(1) Comparing EPSF with the state-of-the-art EP classifiers, CE-EP [Yu et al. 2013], the EPSF algorithm in our previous KDD conference version (we call it Pre-EPSF) [Yu et al. 2012], and the IG-EP classifier, the EP classifier with the information gain feature selection method. (We don't compare EPSF with CAEP [Dong

Table VII. 16 Data sets used in our comparative study
(#: number of features, SIZE: number of instances)

ID	Dataset	#	SIZE	ID	Dataset	#	SIZE
1	kr-vs-kp	36	3,196	9	dexter	20,000	300
2	spectf	44	267	10	breast-cancer	17,816	286
3	promoters	57	106	11	arcene	10,000	100
4	infant	86	5,337	12	dorothea	100,000	800
5	madelon	500	2,000	13	colon	2,000	62
6	hiva	1,617	4,229	14	leukemia	7,129	72
7	ovarian-cancer	2,190	216	15	lung-cancer	12,533	181
8	lymphoma	7,399	227	16	prostate	6,033	102

et al. 1999], CBA [Liu et al. 1998], CMAR [Li et al. 2001b] and CPAR [Yin and Han 2003] since they fail to deal with high dimensionality in the scale of thousands or more.)

(2) Comparing the prediction accuracy of EPSF with that of the state-of-the-art non-associative classifiers, including Naïve Bayes (NB), KNN, Decision Tree J48, SVM, Bagging and AdaBoost using their implementation provided by the Weka tool [Hall et al. 2009].

(3) Comparing the prediction accuracy of EPSF with that of NB, KNN, J48, SVM, Bagging and AdaBoost classifiers with the add-on information gain feature selection method in Weka.

(4) Analyzing the statistical qualities of the EPSF algorithm against the rivals mentioned above using the kappa statistic [Cohen 1960], the Friedman test [Friedman 1940], and the Nemenyi test [Demšar 2006].

We simulate the steaming feature setting using benchmark data sets to evaluate EPSF and Pre-EPSF, by assuming that the dimensions on a benchmark training data set are available one by one at a time and each dimension is processed upon its arrival. To discretize continuous features, we use the discretization method in the Causal Explorer Toolkit proposed by Aliferis et al. [Aliferis et al. 2003]. In the experiments, we set the growth rate to 20 for EPSF and CE-EP. To test the impact of the minimum support threshold, we set seven minimum supports for EPSF, Pre-EPSF, and CE-EP, including 0.005, 0.01, 0.05, 0.1, 0.2, 0.3, and 0.4, respectively. The experiments were performed on a Window 7 DELL workstation with an Intel Xeon 2.93 GHz processor and 12.0GB RAM.

5.2. Comparison of EPSF and Pre-EPSF

Table VIII gives the results of computer running time of EPSF against Pre-EPSF (the EPSF algorithm in our previous version [Yu et al. 2012]). Comparing to Pre-EPSF, for validating the redundancy of each originally existing feature in the current feature subset, EPSF in this paper avoids checking all subsets within the current feature subset at each round by only testing the subsets created by the inclusion of a new feature at each time point. Thus, in comparison with Pre-EPSF, EPSF in this paper significantly improves the updating efficiency, as shown in Table VIII. The best results are highlighted in bold face.

Moreover, since EPSF needs fewer statistical tests to determine the redundancy of a feature in the current feature subset than Pre-EPSF (hence less unintended estimation errors are introduced), Table IX shows that EPSF gets higher prediction accuracy on some data sets than Pre-EPSF, especially on high-dimensional data sets with small

Table VIII. Running time (in seconds):
EPSF and Pre-EPSF

Dataset	EPSF	Pre-EPSF
infant	**26**	41
kr-vs-kp	**24**	43
promoters	17	**16**
spectf	**17**	**17**
madelon	**20**	23
hiva	**33**	163
ovarian-cancer	**20**	68
lymphoma	**20**	44
dexter	**31**	387
arcene	**19**	30
breast-cancer	**101**	958
dorothea	**146**	440
colon	**17**	18
leukemia	**19**	22
lung-cancer	**30**	117
prostate	**20**	27

Table IX. Prediction accuracy (%):
EPSF and Pre-EPSF

Dataset	EPSF	Pre-EPSF
infant	**91.44**	91.35
kr-vs-kp	92.39	**92.42**
promoters	**72.00**	71.00
spectf	**86.92**	**86.92**
madelon	59.80	**61.20**
hiva	90.71	**95.17**
ovarian-cancer	92.38	**93.81**
lymphoma	**80.91**	76.82
dexter	**90.67**	89.67
arcene	**84.44**	80.00
breast-cancer	**95.19**	92.59
dorothea	**95.06**	94.94
colon	**95.00**	91.67
leukemia	**100**	**100**
lung-cancer	**99.44**	98.89
prostate	**98.00**	95.00

Table X. Kappa statistic and its corresponding Kappa agreement

Kappa statistic	< 0	0.01-0.20	0.21-0.40	0.61-0.80	0.81-0.99
Kappa Agreement	less than chance agreement	slight agreement	moderate agreement	substantial agreement	almost perfect agreement

samples, such as four NIPS 2003 feature selection challenge data sets and four frequently studied public microarray data sets. In Table IX, we select the best prediction accuracy under the seven minimum supports as the results for our comparative study.

Finally, Figure 2 reports the Kappa statistics of EPSF and Pre-EPSF. In Figure 2, the x-axis denotes all of the 16 data sets corresponding to Table VII. The kappa statistic is a measure of consistency amongst different raters, taking into account the agreement occurring by chance [Cohen 1960]. The kappa statistic is standardized to lie on a -1 to 1 scale, where 1 is perfect agreement, 0 is exactly what would be expected by chance, and negative values indicate agreement less than chance, and the other values of Kappa statistics and their corresponding Kappa agreements are shown in Table X [Landis and Koch 1977].

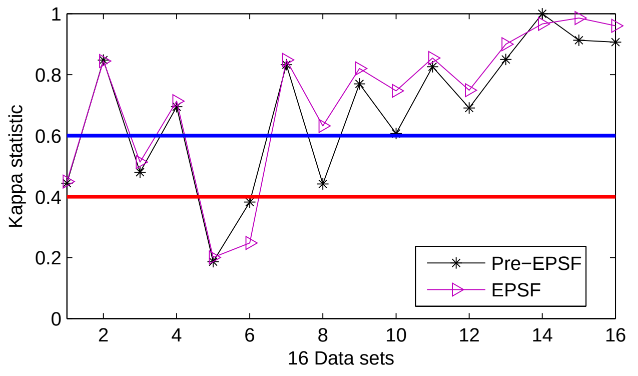


Fig. 2. Kappa statistics of Pre-EPSF and EPSF on 16 data sets

From Figure 2, EPSF is better than Pre-EPSF using Kappa statistics, and then we can conclude that EPSF is more reliable than Pre-EPSF. The explanation is that fewer statistical tests of EPSF than those of Pre-EPSF make EPSF have more statistical power than Pre-EPSF. We can see that both EPSF and Pre-EPSF have only two kappa statistics (the *madelon* and *hive* data sets) that are lower than 0.4 (under the red line in Figure 2), since the *madelon* data set is a synthetic data set including many redundant and noise features and *hive* is a very class-imbalanced data set (the proportion of positive class is only 3.52%). Meanwhile both of them have 11 kappa statistics that are higher than 0.6 (above the blue line in Figure 2). Pre-EPSF is a reliable emerging pattern classifier, while the improved EPSF is more reliable and more efficient.

5.3. Comparison of EPSF with CE-EP and IG-EP

5.3.1. Comparison of Prediction Accuracy. Table XI reports detailed results in terms of prediction accuracy (the percentage of the correctly classified test instances which are previously unseen) of EPSF, CE-EP, and IG-EP on the 16 benchmark data sets. As for EPSF, CE-EP, and IG-EP, we select the best prediction accuracy under the seven minimum supports as the results for our comparative study. The best result is highlighted in bold face for each data set.

To further investigate the classification results, we conduct paired t-tests at a 95% significance level and summarize the win/tie/loss counts of EPSF against CE-EP and IG-EP. For example, as shown in the last row of Table XI, against CE-EP, EPSF wins five times, ties eight times and loses three times on the 16 data sets. And EPSF is always superior to or tie with IG-EP.

5.3.2. Comparison of the Numbers of Patterns and Running Time. Figures 3 and 4 compare the numbers of patterns mined by EPSF against CE-EP and IG-EP. We report the average numbers of mined patterns over all seven minimum support thresholds. In Fig.3, the x-axis denotes all of the 16 data sets corresponding to Table VII. From Fig. 3, we can see that EPSF is very competitive with CE-EP on the number of mined patterns, while IG-EP selects more patterns than EPSF and CE-EP, as shown in Figure 4. These results illustrate that both EPSF and CE-EP can select a small set of strongly predictive EPs from a very high-dimensional data set. Furthermore, we can see that even with very high dimensionality, the numbers of patterns selected by both EPSF and

Table XI. Prediction accuracy (%): EPSF, CE-EP, and IG-EP

Dataset	EPSF	CE-EP	IG-EP
infant	91.44	**94.92**	91.61
kr-vs-kp	**92.39**	92.23	87.58
promoters	72.00	72.00	**75.00**
spectf	**86.92**	83.85	83.08
madelon	59.80	59.00	**60.80**
hiva	90.71	**93.70**	93.36
ovarian-cancer	92.38	**92.86**	83.33
lymphoma	**80.91**	77.73	78.18
dexter	**90.67**	88.33	79.33
arcene	84.44	**86.67**	68.89
breast-cancer	**95.19**	92.22	90.74
dorothea	**95.06**	**95.06**	93.92
colon	**95.00**	**95.00**	88.33
leukemia	**100**	**100**	**100**
lung-cancer	**99.44**	**99.44**	98.89
prostate	**98.00**	94.00	94.00
win/tie/loss	/	5/8/3	10/3/3

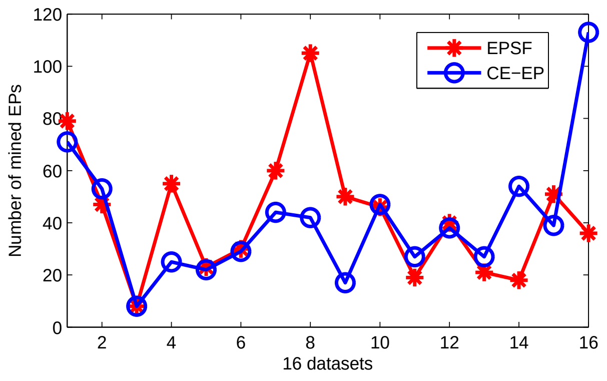


Fig. 3. Numbers of mined EPs: EPSF against CE-EP

CE-EP do not change much in comparison with those on the first four low-dimensional data sets in Table VII.

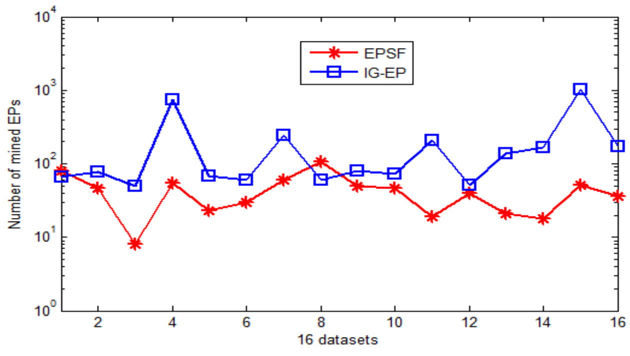


Fig. 4. Numbers of mined EPs: EPSF against IG-EP

Table XII. Kappa statistics of EPSF, CE-EP, and IG-EP

Dataset	EPSF	CE-EP	IG-EP
infant	0.4493	**0.7875**	0.4774
kr-vs-kp	**0.8448**	**0.8448**	0.7508
promoters	0.5133	0.5133	**0.5400**
spectf	**0.7131**	0.6476	0.6306
madelon	0.2007	0.1866	**0.2191**
hiva	0.2477	**0.3497**	0.2311
ovarian-cancer	0.8485	**0.8621**	0.7077
lymphoma	**0.6318**	0.5818	0.5864
dexter	**0.8200**	0.7742	0.5983
arcene	0.7467	**0.7800**	0.4867
breast-cancer	**0.8548**	0.8197	0.7888
dorothea	**0.7488**	0.7362	0.6381
colon	0.9000	**0.9167**	0.7833
leukemia	0.9667	**1.0000**	**1.0000**
lung-cancer	**0.9857**	**0.9857**	0.9714
prostate	**0.9600**	0.8733	0.8800

The running time (in seconds) of EPSF, CE-EP, and IG-EP contains all learning time, including importing data sets, and 10-fold cross validation learning and testing. Figure 5 reports the average running time over all seven minimum support thresholds. In Figure 5, we can see that EPSF is the fastest algorithm while IG-EP is the slowest one.

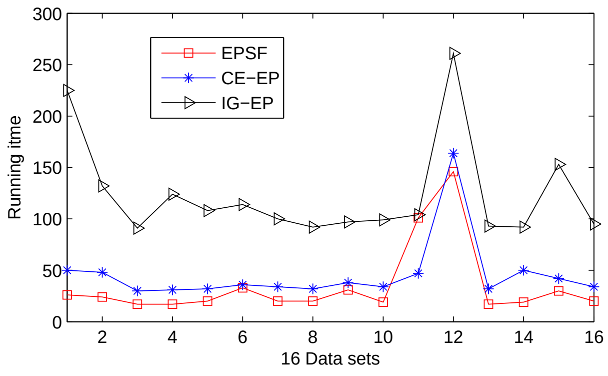


Fig. 5. Running time (in seconds): EPSF, CE-EP, and IG-EP

5.3.3. Analysis of the Statistical Qualities. To further analyze EPSF, CE-EP, and IG-EP, we compare them by the kappa statistic and the Nemenyi test. To calculate the kappa statistic, we set the support threshold to 0.2 and the growth rate threshold to 20 for EPSF, CE-EP, and IG-EP. Table XII shows the kappa statistics of EPSF, CE-EP, and IG-EP. We observe that with the kappa statistics, the three classifiers have no significant difference according to Table X.

Accordingly, we further use the Friedman test [Friedman 1940] and the Nemenyi test [Demšar 2006] to assess whether the performance of our algorithm EPSF is comparable to that of CE-EP and IG-EP in prediction accuracy. With the Friedman test at 95% significance level, under the null-hypothesis, which states that whether the performance of EPSF and that of CE-EP and IG-EP have no significant difference in

Table XIII. Comparison of prediction accuracy (%): EPSF, NB, IG-NB, KNN, IG-KNN, J48, and IG-J48

Dataset	EPSF	NB	IG-NB	KNN	IG-KNN	J48	IG-J48
infant	91.44	91.91	92.69	94.92	95.05	95.39	**95.43**
kr-vs-kp	92.39	83.92	85.89	96.46	96.37	**99.31**	96.75
promoters	72.00	**74.53**	71.70	62.26	65.09	63.21	70.75
spectf	86.92	86.63	83.90	84.27	86.52	86.14	**87.27**
madelon	59.80	59.20	**62.30**	53.55	61.95	57.50	62.20
hiva	90.71	87.06	94.16	96.50	96.38	96.39	**96.62**
lymphoma	**80.91**	68.28	80.61	63.88	73.57	71.81	70.93
breast-cancer	**95.19**	93.01	89.96	86.36	90.55	80.77	85.66
ovarian-cancer	**92.38**	70.83	84.26	85.19	88.89	91.67	89.35
dorothea	**95.06**	90.25	93.63	90.63	93.38	89.38	93.13
arcene	**84.44**	63.00	73.00	80.00	79.00	62.00	77.00
dexter	90.67	**93.33**	78.33	63.67	86.00	82.67	87.00
colon	**95.00**	79.03	91.94	83.87	93.55	82.26	90.32
leukemia	**100**	93.06	**100**	97.22	**100**	93.06	95.83
lung-cancer	**99.44**	98.34	98.90	98.34	98.34	90.61	96.69
prostate	**98.00**	69.61	94.12	93.14	94.12	88.24	93.14

prediction accuracy, the null-hypothesis is rejected. We get the average ranks for EPSF, CE-EP, and IG-EP as 2.3125, 2.1563, and 1.5313, respectively.

Then we proceed with the Nemenyi test as a post-hoc test to deal with this situation. With the Nemenyi test, the performance of the two classifiers is significantly different if the corresponding average ranks differ by at least the critical difference (how to calculate the average ranks and the critical difference, please see Section 3.2.2 of [Demšar 2006]). With the Nemenyi test, the critical difference we get is up to 0.8275.

Thus, with the critical difference and the average ranks calculated above, we conclude that the performance of EPSF and that of CE-EP have no significant difference in prediction accuracy, but is significantly better than that of IG-EP.

5.4. Comparison of EPSF against the Non-Associative Classifiers

Tables XIII to XIV give the empirical results in terms of prediction accuracy of EPSF, six non-associative classifiers, and the same six classifiers with the add-on information gain feature selection method on the 16 benchmark data sets. We denote the six classifiers with information gain feature selection as IG-NB, IG-KNN, IG-J48, IG-SVM, IG-Bagging, and IG-AdaBoost, respectively. Table XV reports the running time of EPSF, NB, KNN, J48, SVM, Bagging, and AdaBoost. EPSF is very competitive with these six non-associative classifiers. But on data sets with very high dimensionality or large sample sizes, such as the *madelon*, *hiva*, and *dorothea* data sets, the running time of most of non-associative classifiers is more than that of EPSF.

To investigate the classification results of prediction accuracy, we conduct paired t-tests at a 95% significance level and summarize the win/tie/loss counts of EPSF against the other rivals in Table XVI. In Table XVI, we can see that EPSF is superior to NB, KNN, J48, Bagging and AdaBoost and their variants with the information gain feature selection method, and also very competitive with SVM and IG-SVM. We have observed in the experiments that the integration of streaming feature selection into EP mining can avoid generating non-EPs and redundant EPs. This enables EPSF not only to handle high-dimensional data sets such as the last 12 data sets in Table VII, but also to produce very promising prediction accuracy.

Table XIV. Comparison of prediction accuracy (%):
EPSF, SVM, IG-SVM, Bagging, IG-Bagging, AdaBoost, and IG-AdaBoost

Dataset	EPSF	SVM	IG-SVM	Bagging	IG-Bagging	AdaBoost	IG-AdaBoost
infant	91.44	95.45	95.48	**95.65**	95.51	95.43	95.43
kr-vs-kp	92.39	95.06	94.02	**99.22**	96.25	93.84	93.84
promoters	72.00	**79.25**	70.75	66.98	72.64	66.04	72.64
spectf	86.92	88.02	89.89	**90.37**	87.64	84.27	85.39
madelon	59.80	56.45	**62.75**	62.20	62.45	60.50	60.7
hiva	90.71	94.70	96.26	**96.76**	96.57	96.48	96.48
lymphoma	**80.91**	77.53	79.30	68.28	78.85	62.56	68.72
breast-cancer	**95.19**	92.31	90.21	84.97	88.81	84.61	87.41
ovarian-cancer	92.38	**93.52**	91.67	88.89	87.96	91.67	89.35
dorothea	**95.06**	92.00	94.00	94.13	93.75	93.75	93.75
arcene	**84.44**	81.00	74.00	72.00	78.00	71.00	79.00
dexter	90.67	**91.33**	86.33	89.33	88.67	83.33	85.00
colon	**95.00**	85.48	88.71	85.48	85.48	85.48	91.94
leukemia	**100**	98.61	**100**	94.44	97.22	**100**	**100**
lung-cancer	99.44	**100**	**100**	93.92	99.45	96.69	99.45
prostate	**98.00**	94.12	94.12	92.16	95.10	92.16	94.12

Table XV. Comparison of running time (in seconds):
EPSF, NB, KNN, J48, SVM, Bagging, and AdaBoost

Dataset	EPSF	NB	KNN	J48	SVM	Bagging	AdaBoost
infant	26	5	5	10	59	20	8
kr-vs-kp	24	5	5	5	10	5	5
promoters	17	5	5	5	5	5	5
spectf	17	5	5	5	5	5	5
madelon	20	1	21	14	770	50	21
hiva	33	5	95	185	269	430	43
lymphoma	20	6	6	6	6	15	6
breast-cancer	101	10	10	21	33	51	26
ovarian-cancer	20	5	5	5	5	5	5
dorothea	146	70	975	675	670	1425	625
arcene	19	1	1	1	5	14	3
dexter	31	5	5	52	27	94	17
colon	17	5	5	5	5	5	5
leukemia	19	5	6	6	10	10	10
lung-cancer	30	5	5	5	10	15	10
prostate	20	5	5	5	10	10	10

Table XVI. Win/tie/loss counts of EPSF vs. the other 12 non-associative classifiers
(pairwise t-test at 95% significance level)

	NB	KNN	J48	SVM	Bagging	AdaBoost
EPSF	11/3/2	13/0/3	11/2/3	8/2/6	10/2/4	10/3/3
	IG-NB	IG-KNN	IG-J48	IG-SVM	IG-Bagging	IG-AdaBoost
EPSF	9/4/3	10/2/4	11/1/4	8/3/5	9/3/4	9/4/3

To further analyze prediction accuraces of these classifiers, we use the Friedman test and the Nemenyi test to assess their performance. With the Friedman test at 95% significance level for EPSF, NB, KNN, J48, SVM, Bagging and AdaBoost, the null-hypothesis is also rejected. The average ranks of EPSF, NB, KNN, J48, SVM, Bagging and AdaBoost are 5.4063, 3.0313, 3.1875, 2.75, 5.375, 4.5, and 3.75, respectively. After the Nemenyi test, the critical difference is up to 2.252. Therefore, we can conclude that the performance of EPSF is significantly better than that of NB and J48, but is highly comparable to that of KNN, SVM, Bagging and AdaBoost.

Finally, with the Friedman test at 95% significance level for EPSF, IG-NB, IG-KNN, IG-J48, IG-SVM, IG-Bagging and IG-AdaBoost, the null-hypothesis cannot be rejected.

Table XVII. Comparison of kappa statistics:
EPSF against 6 associative classifiers with information gain feature selection method

Dataset	EPSF	IG-NB	IG-KNN	IG-J48	IG-SVM	IG-Bagging	IG-AdaBoost
infant	0.4493	0.4548	0.4328	0.5019	0.4851	0.5012	**0.5174**
kr-vs-kp	0.8448	0.7165	0.9273	**0.9348**	0.8800	0.9248	0.8762
promoters	**0.5133**	0.4340	0.3019	0.4151	0.4151	0.4528	0.4528
spectf	**0.7131**	0.5528	0.5933	0.5886	0.6845	0.5797	0.5314
madelon	0.2007	0.2460	0.2390	0.2440	**0.2550**	0.2490	0.2140
hiva	**0.2477**	0.2327	0.0663	0.0996	0.0077	0.0731	0.0596
lymphoma	**0.6318**	0.6124	0.4717	0.4185	0.5859	0.5571	0.3743
breast-cancer	**0.8548**	0.7515	0.7668	0.6281	0.7470	0.7035	0.6853
ovarian-cancer	**0.8485**	0.6862	0.7765	0.7837	0.8305	0.7552	0.7827
dorothea	**0.7488**	0.6181	0.5425	0.5502	0.6100	0.5884	0.6091
arcene	**0.7467**	0.4664	0.5728	0.5298	0.4698	0.5557	0.5770
dexter	**0.8200**	0.5567	0.7200	0.7400	0.7267	0.7733	0.7000
colon	**0.9000**	0.8256	0.8561	0.7842	0.7456	0.6729	0.8220
leukemia	0.9667	**1.0000**	**1.0000**	0.9089	**1.0000**	0.9376	**1.0000**
lung-cancer	0.9857	0.9620	0.9438	0.8861	**1.0000**	0.9808	0.9808
prostate	**0.9600**	0.8825	0.8825	0.8627	0.8824	0.9020	0.8823

Thus, the performance of EPSF has no significant difference from that of the six non-associative classifiers using the information gain feature selection method.

To further analyze the statistical qualities of EPSF, we compare EPSF with IG-NB, IG-KNN, IG-J48, IG-SVM, IG-Bagging, and IG-AdaBoost by the kappa statistics and the Nemenyi test. Since the prediction accuracy of IG-NB, IG-KNN, IG-J48, IG-SVM, IG-Bagging and IG-AdaBoost is better than NB, KNN, J48, SVM, Bagging, and AdaBoost, we do not give the kappa statistics of NB, KNN, J48, SVM, Bagging, and AdaBoost. We can see that EPSF gets higher kappa statistics than the other six classifiers in Table XVII, especially on the class-imbalance data sets, such as *hiva* and *dorothea*, or the data sets with high dimensionality but small sample sizes, such as *lymphoma* and *prostate*. A possible explanation is that the emerging patterns of each class are correctly mined by EPSF from the corresponding class data and represent strong contrasts between different classes of data.

Why is the prediction accuracy of EPSF not better than that of IG-EP and 12 non-associative classifiers on the low-dimensional data sets, such as *infant*, *kr-vs-kp*, *promoters*, *spectf*, *madelon*, and *hiva*, while it is better than that of these algorithms on the remaining high-dimensional data sets? The explanation is that a high-dimensional data set would have a better chance of including excessive irrelevant or redundant features than a low-dimensional data set. Those excessive irrelevant or redundant features might significantly reduce performance of predictive models. Thus, on a high-dimensional data set, the adverse impact of irrelevant or redundant features on predictive models is more significant than that on low-dimensional data sets. Our empirical results reveal that EPSF can deal with irrelevant or redundant features in high-dimensional data sets much better than the other rivals.

In summary, we can conclude that on data sets with streaming features, the performance of EPSF is very competitive with that of CE-EP and is better than that of IG-EP, which both need to obtain a complete set of features in advance. Furthermore, in comparison with the six non-associative classifiers, and the six classifiers with the information gain feature selection method, the prediction accuracy of EPSF is also very competitive with that of these 12 non-associative classifiers.

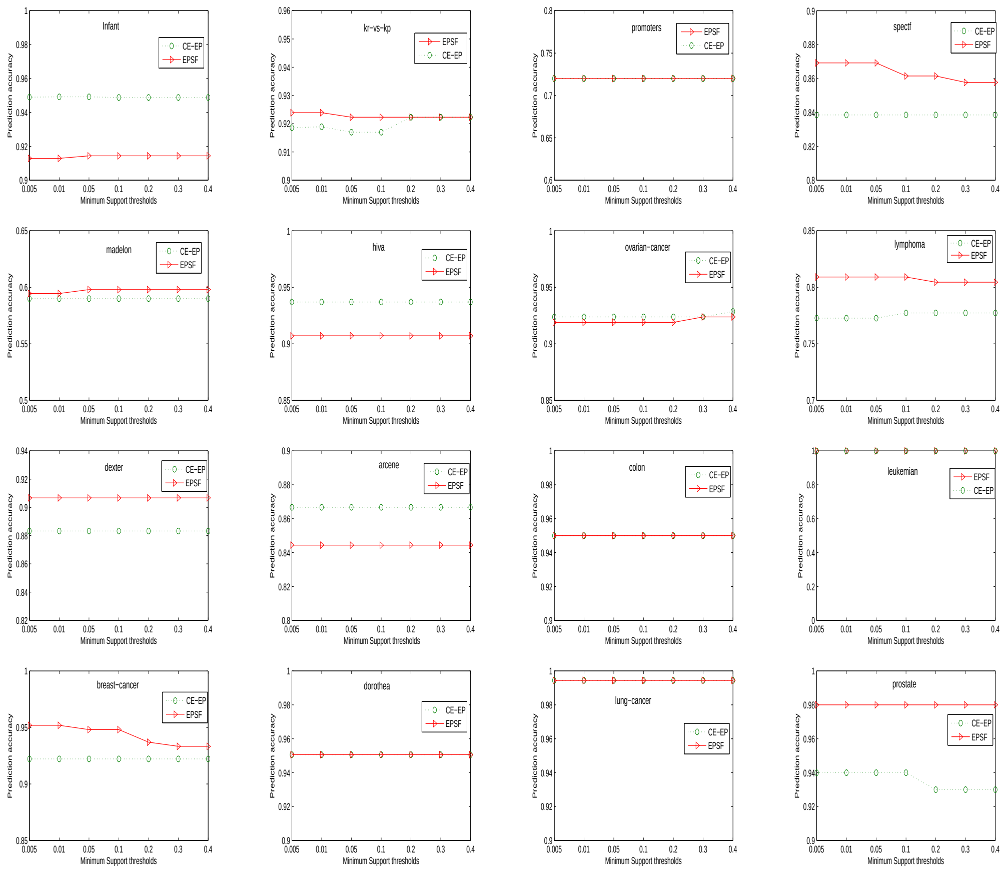


Fig. 6. Sensitivity analysis of support thresholds on prediction accuracy

5.5. Analysis of Prediction Accuracy on Support Thresholds

Figure 6 shows the prediction accuracy of EPSF and CE-EP under the seven support thresholds, 0.005, 0.01, 0.05, 0.1, 0.2, 0.3, and 0.4. We can see that in prediction accuracy, for all 16 data sets, EPSF is insensitive to the different support thresholds, even for those high-dimensional data sets. Furthermore, EPSF is not only more insensitive, but also always achieves higher accuracy under all the seven support thresholds than CE-EP.

5.6. Analysis of Prediction Accuracy on Growth-rate Thresholds

To further explore the performance of EPSF and CE-EP, we conduct an analysis on prediction accuracy of EPSF and CE-EP under seven minimum growth rate thresholds, as shown in Figures 7 and 8, where GR stands for Growth Rate thresholds and the minimum support threshold is fixed at 0.1. In Figures 7 and 8, the x-axis denotes all of the 16 data sets corresponding to Table VII. From Figures 7 and 8, we can see that CE-EP and EPSF are not sensitive to the minimum growth rate thresholds at all.

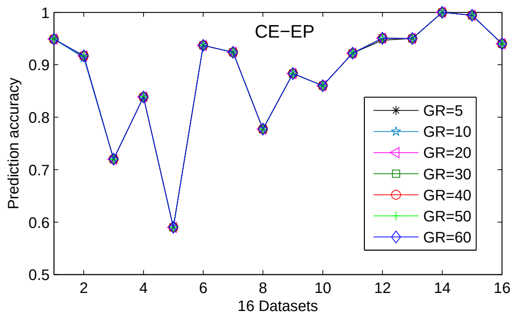


Fig. 7. The effect of growth rate thresholds on CE-EP

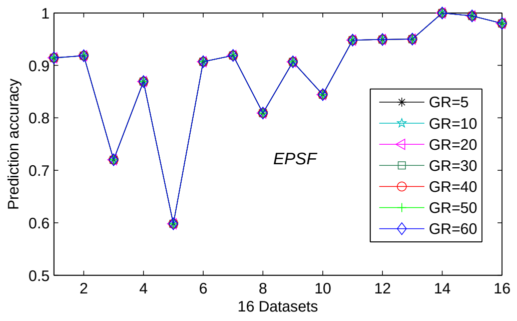


Fig. 8. The effect of growth rate thresholds on EPSF

5.7. Effect of the Input Orders of Features

Since streaming features are processed one by one as they are available, we conduct an analysis of prediction accuracy on the input (or scan) order of features, against CE-EP and SVM as the rival algorithms. We generate a number of trials in which each trial represents a random input order of features. We apply EPSF to each randomized trial and report the results in Figure 9, where the x-axis represents each of the randomized trials and the y-axis represents the prediction accuracy from the corresponding trial. The results in Figure 9 confirm that varying the input order of features does slightly impact on the prediction accuracy, however, the results demonstrate that EPSF has relatively stable performance.

5.8. Mining EPs with Features Kept Arriving

When the features keep arriving, EPSF provides a solution to this problem by processing features one by one and stopping this process using the EPs seen so far with a user-specified criterion. CE-EP cannot deal with this situation since it needs to access all features in advance to identify the causes and effects of the class attribute. We evaluate this performance of EPSF in Fig.10. For four gene data sets, we select the first 2/3 data instances as the training instances and the remaining for testing; for the *breast-cancer* data set, we select the first 200 data instances as the training instances and the remaining for testing. With respect to the *dorothea* data set, we use its original training and testing data sets. SVM and AdaBoost are used as baselines on the training and testing sets with a complete set of features. With streaming features,

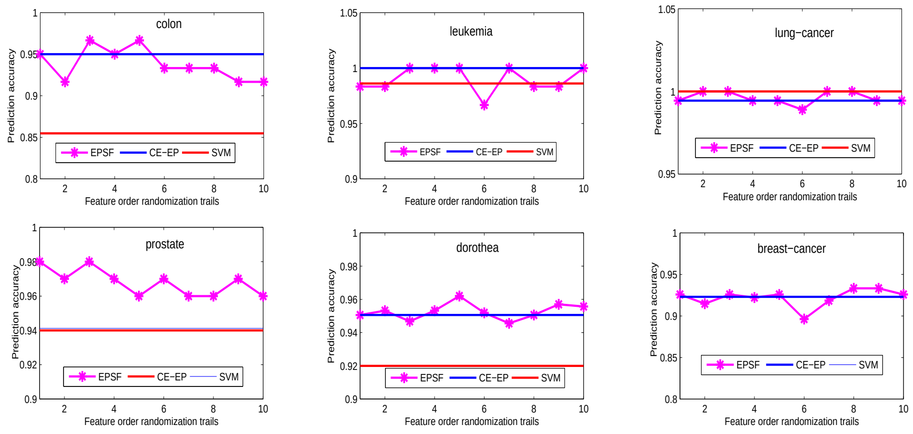


Fig. 9. Effect of the input order of features

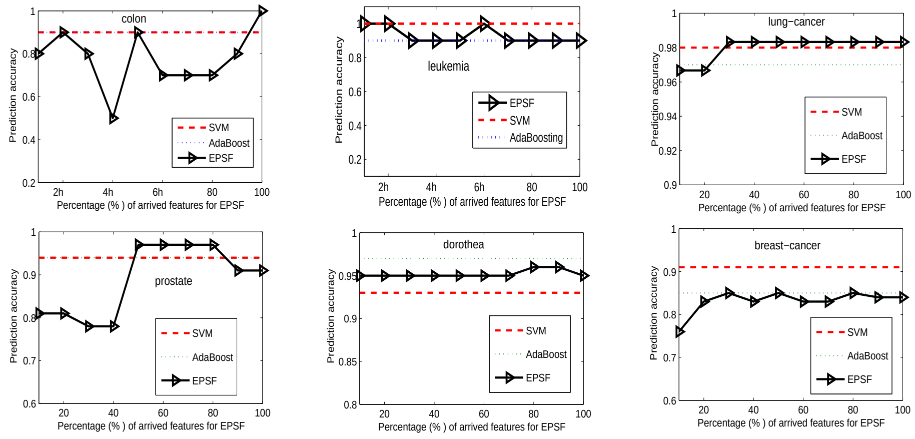


Fig. 10. Comparative performance of EPSF with features keeping arriving

EPSF mines EPs on the training samples as the features are available one by one and evaluates the current EPs on the testing samples.

On the *colon* data set, when the percentage of features available is up to 20% or 50%, the prediction accuracy of EPSF is the same as SVM. And when features are all available, the accuracy of EPSF is up to 100%, and is better than SVM. For the remaining data sets, EPSF can be also up to the accuracy of SVM or AdaBoost without exhaustive search over an entire feature set. This demonstrates that EPSF provides an effective and efficient solution to the EP mining problem when it is impossible to get a complete set of features in advance and must be consumed features in an online manner.

5.9. A Case Study on Automatic Impact Crater Detection

In addition to the validation on the publicly available benchmark data sets, we also apply our new approach to automatic impact crater detection in real planetary images. Impact craters, the structures formed by the collisions of meteoroids on planetary sur-

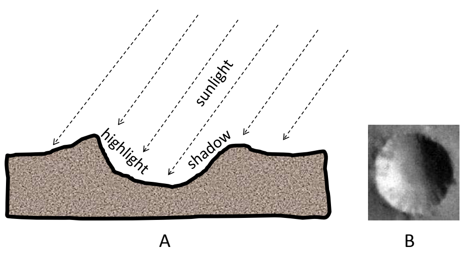


Fig. 11. (A) An illustration explaining why an image of a sub-kilometer crater consists of crescent-like highlight and shadow regions. (B) An image of an actual 1 km crater showing the highlight and shadow regions [Ding et al. 2011].

Table XVIII. Summary of crater datasets

	#samples (crater candidates)	#features
West region	6,708	1,089
Central region	2,935	1,089
East region	2,026	1,089

faces, are among the most studied geomorphic features in the solar system because they yield information about past and present geological processes and provide the only tool for measuring relative ages of planetary surfaces, i.e., heavily cratered surfaces are relatively older than less cratered surfaces [Urbach and Stepinski 2009; Ding et al. 2011].

In this case study, the EPSF algorithm is plugged into the crater detection framework designed by Ding et al. [Ding et al. 2011]. The calculation contains three steps: (1) identifying crater candidates; (2) extracting image texture features; and (3) detecting craters using supervised learning algorithms.

Crater candidates are the regions of an image that may potentially contain craters. A key insight to identifying crater candidates is that a crater can be recognized as a pair of crescent-like highlight and shadow regions in an image, as shown in Figure 11. Those highlight and shadow regions are matched so that each pair will be used to construct crater candidates, that is, the locations where craters are likely to reside. The experiments in crater detection are evaluated on Mars because it is at the center of NASA exploration efforts. A portion of the High Resolution Stereo Camera (HRSC) nadir panchromatic image h0905 is selected, taken by the Mars Express spacecraft, to serve as the test set [Ding et al. 2011]. The selected image has a resolution of 12.5 meters/pixel and a size of 3,000 by 4,500 pixels ($37{,}500\times56{,}250m^2$). The image represents a significant challenge to automatic crater detection algorithms because it covers a terrain that has spatially variable morphology and because its contrast is rather poor (mostly noticeable when the image is inspected at a small spatial scale).

As the image arrives, it is divided into three sections denoted as the west region, the central region, and the east region (see Figure 12) for the test sets summarized in Table XVIII. The central region is characterized by surface morphology that is distinct from the rest of the image. The west and east regions have similar morphology but the west region is much more heavily cratered than the east region. 1,089 image texture features are constructed. The training set consists of 204 true craters and 292 non-crater examples selected randomly from crater candidates located in the northern half of the east region.

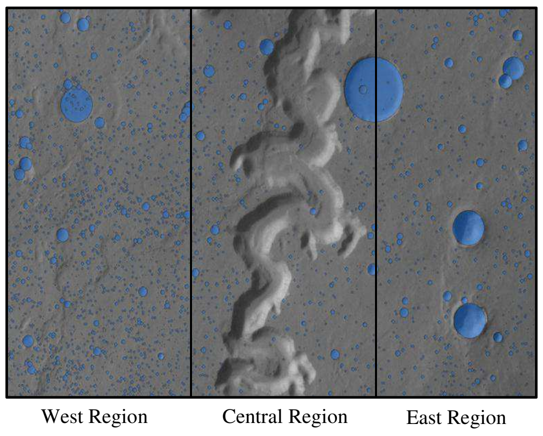
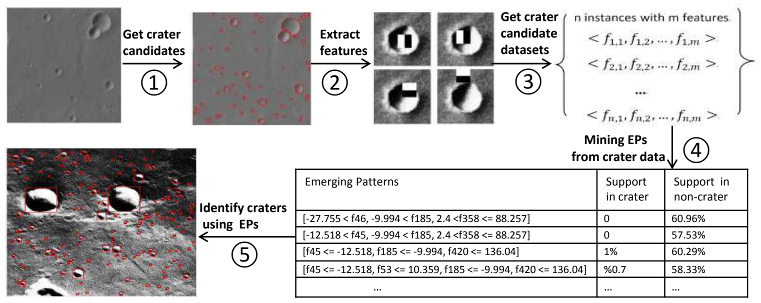


Fig. 12. Impact craters in a 37,500×56,250m^2 test image from Mars [Ding et al. 2011].


Get crater candidates
①
Extract features
②
Get crater candidate datasets
③
n instances with m features
$< f_{1,1}, f_{1,2}, \cdots, f_{1,m} >$
$< f_{2,1}, f_{2,2}, \cdots, f_{2,m} >$
...
$< f_{n,1}, f_{n,2}, \cdots, f_{n,m} >$
Mining EPs from crater data
④
Identify craters using EPs
⑤

Emerging Patterns	Support in crater	Support in non-crater
[-27.755 < f46, -9.994 < f185, 2.4 <f358 <= 88.257]	0	60.96%
[-12.518 < f45, -9.994 < f185, 2.4 <f358 <= 88.257]	0	57.53%
[f45 <= -12.518, f185 <= -9.994, f420 <= 136.04]	1%	60.29%
[f45 <= -12.518, f53 <= 10.359, f185 <= -9.994, f420 <= 136.04]	%0.7	58.33%
...	...	...

Fig. 13. Emerging patterns for crater detection

5.9.1. Emerging patterns for Crater Detection. With the crate datasets summarized above, the framework for counting craters by emerging patterns is shown in Fig.13. In steps 4 to 5 of Figure 13, we apply the EPSF and CE-EP algorithms to high-dimensional crater data for counting craters. Tables XIX and XX show the top 5 EPs mined from the crater training data sets for the crater class and non-crater class using the EPSF algorithm, respectively. In Tables XIX and XX, f45 denotes the 45th feature in the training crate data set while [-138.46, -12.52] represents the value of feature f45. In Table XIX, the supports of the EPs in the crater class are much larger than in the non-crater class. An instance containing one of those EPs will favor the crater class. In Table XX, the first three EPs are the jumping EPs of the non-crater class, and denote the instances containing those EPs as the non-crater class. Accordingly, we conclude that the EPs mined by the EPSF algorithm are high-quality patterns and possess the most discriminative power. They are the best candidates to be used to construct a highly accurate classifier and also can produce an understandable classifier for crater data.

5.9.2. Comparison with Existing Crater Detection Methods. In this section, we compare EPSF with the state-of-the-art crater detection algorithms, CE-EP and Nave Boost [Ding et al. 2011]. The best results are bold-faced in Table XXI. Table XXI shows

Table XIX. Top 5 emerging patterns for craters

ID	Emerging patterns for craters	Support in non-craters	Support in craters	Growth rate
1	{f45∈[-138.46, -12.52], f46∈[-165.18, -27.76], f185∈[-102.58, -9.99], f420∈[-39.82,136.04]}	0.34%	59.31%	173.1961
2	{f45∈[-138.46, -12.52], f46∈[-165.18, -27.76], f68∈[-140.53, 2.97], f420∈[-39.82,136.04]}	0.68%	62.75%	91.6078
3	{f45∈[-138.46, -12.52], f46∈[-165.18, -27.76], f358∈[-113.51, 2.4], f420∈[-39.82,136.04]}	0.68%	62.25%	90.8922
4	{f45∈[-138.46, -12.52], f53∈[-155.16, 10.36], f185∈[-102.58, -9.99], f420∈[-39.82,136.04]}	0.68%	58.33%	85.1667
5	{f45∈[-138.46, -12.52], f185∈[-102.58, -9.99], f420∈[-39.82,136.04]}	1%	60.29%	58.6863

Table XX. Top 5 emerging patterns for non-craters

ID	Emerging patterns for non-craters	Support in crater	Support in non-crater	Growth rate
1	{f46∈[-27.76,144.03], f185∈[-9.99,120.97], f358∈[2.4, 88.26]}	0	60.96%	∞
2	{f45∈[-12.52,158.41], f185∈[-9.99,120.97], f358∈[2.4, 88.26]}	0	57.53%	∞
3	{f68∈[2.97, 140.72], f185∈[-9.99,120.97], f358∈[2.4, 88.26]}	0	56.16%	∞
4	{[f53∈[10.36, 154.01], f358∈[2.4, 88.26]}	9.8%	61.30%	62.55
5	{f45∈[-12.52,158.41], f53∈[10.359,154.01], f185∈[-9.99,120.97], f420∈[136.04,276.73]}	1.47%	66.10%	44.95

Table XXI. The prediction accuracy on three regions

	West region	Central region	East region
EPSF	0.7847	**0.7959**	**0.7784**
CE-EP	**0.7852**	0.7802	0.7739
Nave Boost	0.7661	0.7888	0.7749

that, except for the west region, EPSF outperforms the CE-EP and Nave Boost algorithms.

5.9.3. Comparison with the Other Methods. In this section, we compare the prediction accuracy of EPSF with that of the classifiers, KNN and SVM, with some state-of-the-art feature selection algorithms, OSFS [Wu et al. 2013], HITON-PC [Aliferis et al. 2010], LARS [Efron et al. 2004], and FCBF [Yu and Liu 2004]. The best results are bold-faced in Tables XXII to XXIII. From Tables XXII to XXIII, we can see that EPSF produces higher prediction accuracy than the other four algorithms in the central region and gets very competitive results with the other rivals in the remaining regions. Moreover, the classifier constructed with the mined EPs can help us understand the crated data.

Table XXII. The prediction accuracy on three regions (KNN)

	West region	Central region	East region
EPSF	**0.7847**	**0.7959**	0.7784
OSFS	0.7809	0.7874	**0.7828**
HITON-PC	0.7749	0.7792	0.7813
LARS	0.7740	0.7881	0.7799
FCBF	0.7821	0.7833	**0.7828**
All features(1089)	0.7303	0.7499	0.7710

Table XXIII. The prediction accuracy on three regions (SVM)

	West region	Central region	East region
EPSF	0.7847	**0.7959**	0.7784
OSFS	**0.7856**	0.7874	0.7730
HITON-PC	0.7815	0.7877	0.7710
LARS	0.7840	0.7888	**0.7794**
FCBF	0.7826	0.7923	**0.7794**
All features(1089)	0.7683	0.7710	0.7754

6. CONCLUSIONS

In this paper, to learn and maintain a classification model on data with streaming features, we have adapted the well-known emerging pattern based classification methods and proposed a semi-streaming approach. This new approach is fundamentally different from applying emerging pattern mining straightforwardly on a data set with streaming features. With streaming feature selection, our approach online builds two pools: a feature pool and a 1-itemset EP pool, and periodically computes and updates emerging patterns from the 1-itemset EP pool for classification model construction and maintenance. The streaming feature selection step substantially reduces the dimensionality of the feature space under which the offline emerging pattern mining step has to operate. Due to the effective streaming feature selection customized for emerging pattern mining, the emerging patterns mined in the offline step tend to be short, and this practically facilitates emerging pattern mining dramatically. Comprehensive experimental results on benchmark data sets and a real-world case study on automatic impact crater detection have demonstrated the effectiveness and efficiency of our approach.

ACKNOWLEDGMENTS

We thank the handling editor and the anonymous reviewers for their invaluable and constructive comments, which helped to improve the paper substantially. This work is partly supported by a PIMS Post-Doctoral Fellowship Award of the Pacific Institute for the Mathematical Sciences, Canada, the Program for Changjiang Scholars and Innovative Research Team in University (PCSIRT) of the Ministry of Education, China (under grant IRT13059), the National 973 Program of China (under grant 2013CB329604), the National Natural Science Foundation of China (under grant 61229301 and 61305064), an NSERC Discovery grant and a BCIC NRAS Team Project.

All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies

REFERENCES

Charu C Aggarwal. 2010. Data streams: An overview and scientific applications. In *Scientific Data Mining and Knowledge Discovery*. Springer, 377–397.

Constantin F Aliferis, Alexander Statnikov, Ioannis Tsamardinos, Subramani Mani, and Xenofon D Koutsoukos. 2010. Local causal and markov blanket induction for causal discovery and feature selection for classification part i: Algorithms and empirical evaluation. *The Journal of Machine Learning Research* 11 (2010), 171–234.

Constantin F Aliferis, Ioannis Tsamardinos, Alexander R Statnikov, and Laura E Brown. 2003. Causal Explorer: A Causal Probabilistic Network Learning Toolkit for Biomedical Discovery.. In *METMBS*, Vol. 3. 371–376.

CL Blake and Christopher J Merz. 1998. UCI Repository of machine learning databases [http://www. ics. uci. edu/~ mlearn/MLRepository. html]. Irvine, CA: University of California. *Department of Information and Computer Science* 55 (1998).

Gavin Brown, Adam Pocock, Ming-Jie Zhao, and Mikel Luján. 2012. Conditional likelihood maximisation: a unifying framework for information theoretic feature selection. *The Journal of Machine Learning Research* 13, 1 (2012), 27–66.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement* 20, 1 (1960), 37–46.

Janez Demšar. 2006. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research* 7 (2006), 1–30.

Wei Ding, Tomasz F Stepinski, Yang Mu, Lourenco Bandeira, Ricardo Ricardo, Youxi Wu, Zhenyu Lu, Tianyu Cao, and Xindong Wu. 2011. Subkilometer crater discovery with boosting and transfer learning. *ACM Transactions on Intelligent Systems and Technology (TIST)* 2, 4 (2011), 39.

Guozhu Dong, Jiawei Han, Laks VS Lakshmanan, Jian Pei, Haixun Wang, and Philip S Yu. 2003. Online mining of changes from data streams: Research problems and preliminary results. In *Proceedings of the 2003 ACM SIGMOD Workshop on Management and Processing of Data Streams*.

Guozhu Dong and Jinyan Li. 1999. Efficient mining of emerging patterns: Discovering trends and differences. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 43–52.

Guozhu Dong, Xiuzhen Zhang, Limsoon Wong, and Jinyan Li. 1999. CAEP: Classification by aggregating emerging patterns. In *Discovery Science*. Springer, 30–42.

Lei Duan, Guanting Tang, Jian Pei, James Bailey, Guozhu Dong, Akiko Campbell, and Changjie Tang. 2014. Mining Contrast Subspaces. In *Advances in Knowledge Discovery and Data Mining*. Springer, 249–260.

Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, and others. 2004. Least angle regression. *The Annals of statistics* 32, 2 (2004), 407–499.

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

Gang Fang, Gaurav Pandey, Wen Wang, Manish Gupta, Michael Steinbach, and Vipin Kumar. 2012. Mining low-support discriminative patterns from dense and high-dimensional data. *Knowledge and Data Engineering, IEEE Transactions on* 24, 2 (2012), 279–294.

Milton Friedman. 1940. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics* 11, 1 (1940), 86–92.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. 2009. The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter* 11, 1 (2009), 10–18.

Daphne Koller and Mehran Sahami. 1995. Toward optimal feature selection. In *Proceedings of the 13th International Conference on Machine Learning*. 284–292.

J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *biometrics* (1977), 159–174.

Jinyan Li, Guozhu Dong, and Kotagiri Ramamohanarao. 2000. Instance-based classification by emerging patterns. In *Principles of Data Mining and Knowledge Discovery*. Springer, 191–200.

Jinyan Li, Guozhu Dong, and Kotagiri Ramamohanarao. 2001a. Making use of the most expressive jumping emerging patterns for classification. *Knowledge and Information systems* 3, 2 (2001), 131–145.

Wenmin Li, Jiawei Han, and Jian Pei. 2001b. CMAR: Accurate and efficient classification based on multiple class-association rules. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*. IEEE, 369–376.

Bing Liu, Wynne Hsu, and Yiming Ma. 1998. Integrating classification and association rule mining. In *Proceedings of the 4th ACM SIGKDD international conference on Knowledge discovery and data mining*. 80–86.

Huan Liu and Lei Yu. 2005. Toward integrating feature selection algorithms for classification and clustering. *Knowledge and Data Engineering, IEEE Transactions on* 17, 4 (2005), 491–502.

David Lo, Hong Cheng, Jiawei Han, Siau-Cheng Khoo, and Chengnian Sun. 2009. Classification of software behaviors for failure detection: a discriminative pattern mining approach. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 557–566.

Elsa Loekito and James Bailey. 2006. Fast mining of high dimensional expressive contrast patterns using zero-suppressed binary decision diagrams. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 307–316.

Petra Kralj Novak, Nada Lavrač, and Geoffrey I Webb. 2009. Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining. *The Journal of Machine Learning Research* 10 (2009), 377–403.

Jayakrushna Sahoo, Ashok Kumar Das, and A Goswami. 2014. An effective association rule mining scheme using a new generic basis. *Knowledge and Information Systems* (2014), 1–30.

Yuanfeng Song, Wilfred Ng, Kenneth Wai-Ting Leung, and Qiong Fang. 2014. SFP-Rank: significant frequent pattern analysis for effective ranking. *Knowledge and Information Systems* (2014), 1–25.

Peter Spirtes, Clark N Glymour, and Richard Scheines. 2000. *Causation, prediction, and search*. Vol. 81. MIT press.

Ronan Trépos, Ansaf Salleb-Aouissi, Marie-Odile Cordier, Véronique Masson, and Chantal Gascuel-Odoux. 2013. Building actions from classification rules. *Knowledge and information systems* 34, 2 (2013), 267–298.

Erik R Urbach and Tomasz F Stepinski. 2009. Automatic detection of sub-km craters in high resolution planetary images. *Planetary and Space Science* 57, 7 (2009), 880–887.

Jianyong Wang and George Karypis. 2005. HARMONY: Efficiently mining the best rules for classification. In *SDM*, Vol. 5. SIAM, 205–216.

Jialei Wang, Peilin Zhao, S Hoi, and Rong Jin. 2013b. Online feature selection and its applications. *Knowledge and Data Engineering, IEEE Transactions on* 26, 3 (2013), 698–710.

Xiaofeng Wang, Gang Li, Guang Jiang, and Zhongzhi Shi. 2013a. Semantic trajectory-based event detection and event pattern mining. *Knowledge and information systems* 37, 2 (2013), 305–329.

Xindong Wu, Kui Yu, Wei Ding, Hao Wang, and Xingquan Zhu. 2013. Online feature selection with streaming features. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 35, 5 (2013), 1178–1192.

Xindong Wu, Kui Yu, Hao Wang, and Wei Ding. 2010. Online streaming feature selection. In *Proceedings of the 27th international conference on machine learning (ICML-10)*. 1159–1166.

Xiaoxin Yin and Jiawei Han. 2003. CPAR: Classification based on Predictive Association Rules.. In *SDM*, Vol. 3. SIAM, 369–376.

Kui Yu, Wei Ding, Dan A Simovici, and Xindong Wu. 2012. Mining emerging patterns by streaming feature selection. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 60–68.

Kui Yu, Wei Ding, Hao Wang, and Xindong Wu. 2013. Bridging Causal Relevance and Pattern Discriminability: Mining Emerging Patterns from High-Dimensional Data. *Knowledge and Data Engineering, IEEE Transactions on* 25, 12 (2013), 2721–2739.

Lei Yu and Huan Liu. 2004. Efficient feature selection via analysis of relevance and redundancy. *The Journal of Machine Learning Research* 5 (2004), 1205–1224.

Xiuzhen Zhang, Guozhu Dong, and others. 2000b. Information-based classification by aggregating emerging patterns. In *Intelligent Data Engineering and Automated LearningIDEAL 2000. Data Mining, Financial Engineering, and Intelligent Agents*. Springer, 48–53.

Xiuzhen Zhang, Guozu Dong, and Ramamohanarao Kotagiri. 2000a. Exploring constraints to efficiently mine emerging patterns from large high-dimensional datasets. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 310–314.

Jing Zhou, Dean P Foster, Robert A Stine, and Lyle H Ungar. 2006. Streamwise feature selection. *The Journal of Machine Learning Research* 7 (2006), 1861–1885.