This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2563424, IEEE Transactions on Knowledge and Data Engineering

Online Learning from Trapezoidal Data Streams

Qin Zhang, Peng Zhang, Guodong Long, Wei Ding, *Senior Member, IEEE,* Chengqi Zhang, *Senior Member, IEEE,* and Xindong Wu, *Fellow, IEEE*

Abstract—In this paper, we study a new problem of continuous learning from doubly-streaming data where both data volume and feature space increase over time. We refer to the doubly-streaming data as trapezoidal data streams and the corresponding learning problem as online learning from trapezoidal data streams. The problem is challenging because both data volume and data dimension increase over time, and existing online learning [1] [2], online feature selection [3], and streaming feature selection algorithms [4] [5] are inapplicable. We propose a new Online Learning with Streaming Features algorithm (OL$_{SF}$ for short) and its two variants that combine online learning [1] [2] and streaming feature selection [4] [5] to enable learning from trapezoidal data streams with infinite training instances and features. Specifically, when a new training instance carrying new features arrives, a classifier updates the existing features by following the passive-aggressive update rule [2] and updates the new features by following the structural risk minimization principle. Then, feature sparsity is introduced by using the projected truncation technique. We derive performance bounds of the OL$_{SF}$ algorithm and its variants. We also conduct experiments on real-world data sets to show the performance of the proposed algorithms.

Index Terms—Online Learning, Streaming Features, Sparsity, Trapezoidal Data Streams.

1 Introduction

Recently we have witnessed an increasing number of applications on doubly-streaming data where both data volume and data dimensions increase with time. For example, in graph node classification, both the number of graph nodes and the node features (e.g., the ego-network structure of a social network node) often change dynamically. In text classification and clustering, both the number of documents and text vocabulary increase over time, such as the *infinite vocabulary topic model* [6] to allow the addition, invention and increased prominence of new terms to be captured. Fig. 1 gives an example of doubly-streaming text data where both new documents and new text vocabulary arrive over time.

We refer to the above doubly-streaming data as *trapezoidal data streams* where data dynamically change in both volume and feature dimension. The problem of learning from trapezoidal data streams is much more difficult than existing data stream mining and online learning problems [7], [8]. The main challenge of learning from trapezoidal data streams is how to design highly dynamic classifiers that can learn from increasing training data with an expanding feature space. Obviously, existing online learning [1], [9], online feature selection [3] and streaming feature selection algorithms [5] cannot be directly used to handle the problem

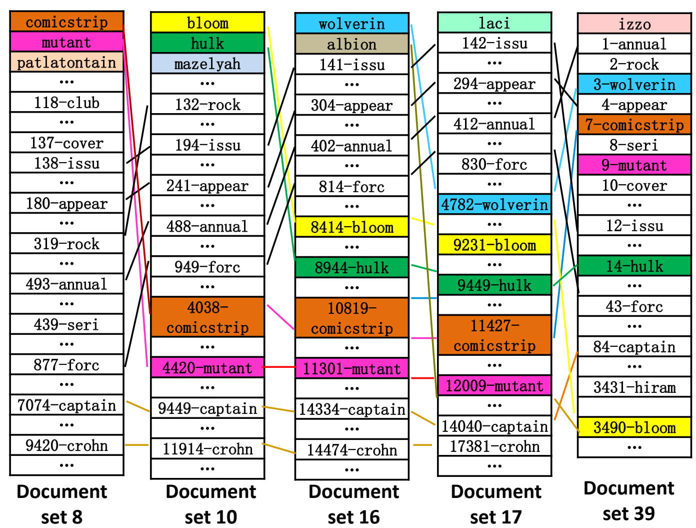


Figure 1. Each column is a document set. We observe document sets continuously arrive as a stream. In each column, words in colored boxes are new words introduced by document sets and the number associates with each word is the importance rank for classification. For example, in document set 16, the word "wolverin" in the blue box was first observed and then became one of the most important words for classification in Document set 39.

because they are not designed to deal with the simultaneous change of data volume and data dimension.

Online learning algorithms [1] were proposed to solve the problem where training instances arrive one by one but the feature space is fixed and known a prior before learning. The algorithms update classifiers using incoming instances and allow the sum of training loss gradually to be bounded [1]. To date, online learning algorithms, such as the Perceptron algorithm [10], the Passive Aggressive algorithm [2] and the Confidence-Weighted algorithm [11], are commonly used in data-driven optimizations, but cannot be directly used to handle a dynamic feature space.

Online feature selection algorithms [1], [3] were pro-

- *Qin Zhang, Peng Zhang, Guodong Long and Chengqi Zhang are with the Centre for Quantum Computation and Intelligent Systems, Faculty of Engineering and Information Technology, University of Technology, Sydney, Australia.*
  E-mail: Qin.Zhang@student.uts.edu.au; Peng.Zhang@uts.edu.au; Guodong.Long@uts.edu.au; Chengqi.Zhang@uts.edu.au
- *Wei Ding is with Department of Computer Science, University of Massachusetts, Boston, MA 02125-3393, USA.*
  E-mail: wei.ding@umb.edu
- *Xindong Wu is with the School of Computer Science and Information Engineering, Hefei University of Technology, China, and the Department of Computer Science, University of Vermont, Burlington, VT 05405, USA.*
  E-mail: xwu@cs.uvm.edu
- *Peng Zhang is the corresponding author.*

posed to perform feature selection in data streams where data arrive sequentially with a fixed feature space. Online feature selectors are only allowed to maintain a small number of active features for learning [3]. These algorithms use sparse strategies, such as feature truncation, to select representative features. Sparse online learning via truncated gradient [1] and the OFS algorithm [3] are typical algorithms. However, these algorithms cannot solve the trapezoidal data stream mining problem because they assume the feature space is fixed.

Online streaming feature selection algorithms [5] were proposed to select features in a dynamic feature space where features arrive continuously as streams. Each new feature is processed upon its arrival and the goal is to select a "best so far" set of features to train an efficient learning model. It, in some ways, can be seen as the dual problem of online learning [5]. Typical algorithms include the online streaming feature selection (OSFS) algorithm [4] and the fast-OSFS [5] algorithm. However, these algorithms consider only a fixed training set where the number of training instances is given in advance before learning.

In this paper, we propose a new Online Learning with Streaming Features (OL$_{SF}$) algorithm and its two variants OL$_{SF}$-I and OL$_{SF}$-II for mining trapezoidal data streams. OL$_{SF}$ and its variants combine online learning and streaming feature selection to continuously learn from trapezoidal data streams. Specifically, when new training instances carrying new features arrive, a classifier updates existing features by following the passive-aggressive update rule used in online learning and updates the new features by following the structural risk minimization principle. Then, feature sparsity is introduced by using feature projected truncation. Theoretical and empirical studies validate the performance of the proposed algorithms. The **contributions** of the paper are summarized as follows:

1) We study a new problem of learning from trapezoidal data streams where training data change in both data volume and feature space;
2) We propose a new learning algorithm OL$_{SF}$ and its two variants. OL$_{SF}$ combines the merits of online learning and streaming feature selection methods to learn from doubly-streaming data;
3) We theoretically analyze the performance bounds of the proposed algorithms;
4) We empirically validate the performance of the algorithms extensively on 14 real-world data sets.

The remainder of the paper is organized as follows: Section 2 surveys the related work. Section 3 introduces the setting of the learning problem. Section 4 discusses the proposed OL$_{SF}$ algorithm and its variants. Section 5 analyzes the performance bounds. Section 6 conducts experiments and Section 7 concludes the paper.

2 Related Work

Our work is closely related to online learning, online feature selection and online streaming feature selection.

Online learning represents an important family of efficient and scalable data mining and machine learning algorithms for massive data analysis [12] [13]. In general, online learning algorithms can be grouped into two categories, the first-order and second-order learning algorithms [12].

The first-order online learning algorithms exploit first order information during update. The Perceptron algorithm [10] [14] and Online Gradient Descent algorithm (OGD) [15] are two well-known first-order online learning methods. Moreover, a large number of first-order online learning algorithms have been proposed recently by following the criterion of maximum margin principle [3], such as the Passive Aggressive algorithms (PA) [2], Approximate Maximal Margin Classification algorithm (ALMA) [16], and the Relaxed Online Maximum Margin algorithms (ROMMA) [16].

The second-order online learning algorithms, which can better explore the underlying structure between features [12], have been explored recently. Most second-order learning algorithms assume that the weight vector follows a Gaussian distribution. The model parameters, including both the mean vector and the covariance matrix, are updated in the online learning process [12]. The Second-Order Perceptron (SOP) [17], Normal Herding method via Gaussian Herding (NHERD) [18], Confidence-Weighted (CW) learning, Soft Confidence Weighted algorithm(SCW) [11], online learning algorithms by Improved Ellipsoid (IELLIP) [19], and Adaptive Regularization of Weight Vectors (AROW) [20], New variant of Adaptive Regularization (NAROW) [21] are representative of the second-order online learning algorithms.

Feature selection is a widely used technique for reducing dimensionality. Feature selection aims to select a small subset of features minimizing redundancy and maximizing relevance to the class label in classification. Feature selection can be categorized into supervised [22] [23], unsupervised [24] [25] and semi-supervised [26] [27] algorithms.

Supervised feature selection can be categorized into the filter models, wrapper models and embedded models [28]. The filter models separate feature selection from classifier learning so that the bias of a learning algorithm does not interact with the bias of a feature selection algorithm. The Relief [29], Fisher score [30] and Information Gain based methods [31] [32] are the representative algorithms. The wrapper models use the predictive accuracy of a predetermined learning algorithm to determine the quality of selected features. The embedded methods [33] [34] [35] aim to integrate feature selection into model training. It achieves model fitting and feature selection simultaneously [36] [37]. The embedded methods are usually the fastest methods.

Unsupervised feature selection attempts to select features that preserve the original data similarity or manifold structures, and it is difficult to evaluate the relevance of features [38] [28]. Laplacian Score [39], spectral feature selection [40], and recently proposed $l_{2,1}$-norm regularized discriminative feature selection [41] are representatives of unsupervised feature selection. Semi-supervised feature selection is between the supervised methods and unsupervised methods. Under the assumption that labeled and unlabeled data are sampled from the same population generated by the target concept, semi-supervised feature selection uses both labeled and unlabeled data to estimate feature relevance [27].

Online feature selection [3] and *sparse online learning* [42] [1] aim to learn a sparse linear classifier from a sequence of high-dimensional training instances. Online feature selection combines feature selection with online learning and

Table 1
Symbols and Notations.

Symbol	Description
B	$B \in [0,1]$, proportion of selected features (projected feature space)
C	$C > 0$, tradeoff in the objective function of OL_{SF}-I and OL_{SF}-II
$d_t, t = 1, \ldots, T$	$d_t \le d_{t+1}$, dimension of instance x_t
$d_{w_t}, t = 1, \ldots, T$	dimension of classifier w_t
λ	$\lambda > 0$, regularization parameter
$l_t, t = 1, \ldots, T$	$l_t = l(w, (x_t, y_t))$, hinge loss on instance (x_t, y_t)
$l_t^*, t = 1, \ldots, T$	$l_t^* = l(\Pi_{x_t} u; (x_t, y_t))$, hinge loss on instance (x_t, y_t) based on the classifier $u \in \mathbb{R}^{d_t}$
L_T	$L_T = \sqrt{\sum_{t=1}^{T} l_t^2}$
M	the number of false predictions by OL_{SF}-I in Theorem 3
$\nabla_t, t = 1, \ldots, T-1$	$\nabla_t = \|w_t - \Pi_{w_t} u\|^2 - \|w_{t+1} - \Pi_{w_{t+1}} u\|^2$
R	upper bound for the $L1$-norm of $x_t, t = 1, \ldots, T$
T	$T \in N^+$, total number of instances
u	$u \in \mathbb{R}^{d_T}$, arbitrary vector in $\mathbb{R}^{d_T}$
U_T	$U_T = \sqrt{\sum_{t=1}^{T} (l_t^*)^2}$
$w_t, t = 1, \ldots, T$	$w_t \in \mathbb{R}^{d_{t-1}}, t = 2, \ldots, T, w_1 \in \mathbb{R}^{d_1}$, classifier built at round t
$\|w_t \cdot x_t\|, t = 1, \ldots, T$	confidence degree of x_t with respect to classifier w_t
$\tilde{w}_{t+1}, t = 1, \ldots, T-1$	$\tilde{w}_{t+1} = \Pi_{w_t} w_{t+1}$, vector of elements w_{t+1} projected to feature space w_t
$\hat{w}_{t+1}, t = 1, \ldots, T-1$	$\hat{w}_{t+1} = \Pi_{\neg w_t} w_{t+1}$, vector of elements w_{t+1} not projected to the feature space w_t
$\bar{w}_{t+1}, t = 1, \ldots, T-1$	intermediate variable of new classifier after the update operation
$\breve{w}_{t+1}, t = 1, \ldots, T-1$	intermediate variable of new classifier on the L_1 ball without truncation
$\Pi_{w_{t+1}/w_t} u, t = 1, \ldots, T-1$	vector of elements u projected to the feature space of w_{t+1} but not to w_t
$x_t, t = 1, \ldots, T$	$x_t \in \mathbb{R}^{d_t}$, input training instance at time t in d_t dimensions
$\tilde{x}_t, t = 2, \ldots, T$	$\tilde{x}_t = \Pi_{w_t} x_t, x_t \in \mathbb{R}^{d_t}, w_t \in \mathbb{R}^{d_{t-1}}, d_{t-1} \le d_t$, vector of elements x_t projected to the feature space of w_t
$\hat{x}_t, t = 2, \ldots, T$	$\hat{x}_t = \Pi_{\neg w_t} x_t, x_t \in \mathbb{R}^{d_t}, w_t \in \mathbb{R}^{d_{t-1}}, d_{t-1} \le d_t$, vector of elements x_t not projected to the feature space of w_t
$\{(x_t, y_t) \| t = 1, 2, \ldots, T\}$	sequence of input training data
ξ	slack variable
$y_t, t = 1, \ldots, T$	$y_t \in \{-1, +1\}$, real label of instance x_t
$\hat{y}_t, t = 1, \ldots, T$	$\hat{y}_t = sign(w_t \cdot \Pi_{w_t} x_t)$, predicted label of instance x_t
$\tau_t, t = 1, \ldots, T$	learning rate variable

resolves the feature selection in an online fashion by developing online classifiers that involve only a small and fixed number of features for classification. OFS and OFS_P [3] are the representative algorithms proposed recently.

Online streaming feature selection algorithms have been studied recently [43] [5] where features arrive one by one and training instances are available before the training process starts. The number of training instances remains fixed through the process [4]. The goal is to select a subset of features and train an appropriate model at each time step given the features observed so far.

Compared with the above learning methods, the problem studied in this paper is more challenging because of the doubly streaming data scenario. Existing online learning, online feature selection and online streaming feature selection algorithms are incapable of learning from trapezoidal data streams.

3 Problem Setting

We consider the binary classification problem on trapezoidal data streams. Let $\{(x_t, y_t) | t = 1, \ldots, T\}$ be a sequence of input training data. Each $x_t \in \mathbb{R}^{d_t}$ is a d_t dimension vector where $d_{t-1} \le d_t$ and class label $y_t \in \{-1, +1\}$ for all t. At each round, the classifier uses information on the current instance to predict its label to be either $+1$ or -1. After the prediction is made, the true label of the instance is revealed and the algorithm suffers an instantaneous loss which reflects the degree of infelicity of the prediction [2]. At the end of each round, the algorithm uses the newly obtained instance-label pair to improve its prediction rule for the rounds to come.

We restrict the discussion to a linear classifier based on a vector of weights w which is the common setting in online learning. The magnitude $|w \cdot x|$ is interpreted as the degree of confidence in the prediction. $w_t \in \mathbb{R}^{d_{t-1}}$ denotes the classifier, i.e., the vector we aim to solve in the algorithm at round t. w_t has the same dimension of the instance x_{t-1}, and has either the same or less dimension as the current instance x_t, for all $t = 2, \ldots, T$, and w_1 is initialized with the same dimension of x_1. For the loss function, we choose the hinge loss. Specifically, $l(w, (x_t, y_t)) = \max\{0, 1 - y_t(w \cdot x_t)\}$, where w and x_t are in the same dimension. In our study, the ultimate dimension d_T is very large, so we also introduce feature selection into our learning algorithm. Table 1 demonstrates the symbols and notations used in the paper.

4 Online Learning with Trapezoidal Data Streams

In this section we present the Online Learning with Streaming Features algorithm (OL_{SF}) and its two variants for mining trapezoidal data streams. There are two challenges to be addressed by the algorithms. The first challenge is to update the classifier with an augmenting feature space. The classifier update strategy is able to learn from new features. We build the update strategy based on the margin-maximum principle. The second challenge is to build a feature selection method to achieve a sparse but efficient model. As the dimension increases over time, it is essential to use feature selection to prune redundant features. We use a truncation strategy to obtain sparsity. Also, in order to improve the truncation, a projection step is introduced before the truncation.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2563424, IEEE Transactions on Knowledge and Data Engineering

The pseudo-codes for the OL$_{SF}$ algorithm and its two variants are given in Algorithms 1, 2 and 3 (OL$_{SF}$-I and OL$_{SF}$-II are different to OL$_{SF}$ in parameter τ_t during updates). The vector w_1 is initialized to a zero vector with dimension d_1, i.e., $w_1 = (0, \ldots, 0) \in \mathbb{R}^{d_1}$ for all the three algorithms, where d_1 is the dimension of the first instance for each algorithm. Then, online learning is divided into the update step and the sparsity step.

Algorithm 1. The OL$_{SF}$ algorithm and its two variants OL$_{SF}$-I and OL$_{SF}$-II

Input:
- $C > 0$: the tradeoff parameter of OL$_{SF}$-I and OL$_{SF}$-II
- $\lambda > 0$: the regularization parameter
- $B \in (0, 1]$: the proportion of selected features

Initialize:
- $w_1 = (0, \ldots, 0) \in \mathbb{R}^{d_1}$

For $t = 1, 2, \ldots$ **do**
  receive instance: $x_t \in \mathbb{R}^{d_t}$
  predict: $\hat{y}_t = sign(w_t \cdot \Pi_{w_t} x_t)$
  receive correct label: $y_t \in \{+1, -1\}$
  suffer loss: $l_t = max\{0, 1 - y_t(w_t \cdot \Pi_{w_t} x_t)\}$
  update step:
  - set parameter :
    $\tau_t = Parameter\_Set(x_t, l_t, C)$
    (See Algorithm 2)
  - update w_t to $\bar{w}_{t+1}$:
    $\bar{w}_{t+1} = [w_t + \tau_t y_t \Pi_{w_t} x_t, \tau_t y_t \Pi_{\neg w_t} x_t]$

  sparsity step:
  - project $\bar{w}_{t+1}$ to a L_1 ball:
    $\breve{w}_{t+1} = \min\{1, \frac{\lambda}{\|\bar{w}_{t+1}\|_1}\} \bar{w}_{t+1}$
  - truncate $\breve{w}_{t+1}$ to w_{t+1}:
    $w_{t+1} = Truncate(\breve{w}_{t+1}, B)$
    (See Algorithm 3)

end for

Algorithm 2. $\tau_t = Parameter\_Set(x_t, l_t, C)$

if OL$_{SF}$:
  $\tau_t = \frac{l_t}{\|x_t\|^2}$
else if OL$_{SF}$-I:
  $\tau_t = min\{C, \frac{l_t}{\|x_t\|^2}\}$
else if OL$_{SF}$-II:
  $\tau_t = \frac{l_t}{\|x_t\|^2 + \frac{1}{2C}}$
end if

Algorithm 3. $w = Truncate(\breve{w}, B)$

$\breve{w} \in \mathbb{R}^{d_{\breve{w}}}$
if $\|\breve{w}\|_0 \geq B \cdot d_{\breve{w}}$ **then**
  $w = \breve{w}^B$
  $\breve{w}^B$ is $\breve{w}$, and remain $\max\{1, floor(B \cdot d_{\breve{w}})\}$ largest elements; set others to zero, where $floor\{x\}$ is the largest integer smaller then x.
else
  $w = \breve{w}$
end if

The update strategy

The three algorithms are different in their update strategy. We first focus on the update strategy of the basic algorithm. At round t, with the classifier $w_t \in \mathbb{R}^{d_{t-1}}$, the new classifier $w_{t+1} = [\tilde{w}_{t+1}, \hat{w}_{t+1}] \in \mathbb{R}^{d_t}$ is obtained as the solution to the constrained optimization problem in Eq.(2), where $\tilde{w} = \Pi_{w_t} w_{t+1} \in \mathbb{R}^{d_{t-1}}$ represents a projection of the feature space from dimension d_t to dimension d_{t-1}, it is a vector consisting of elements of w_{t+1} which are in the same feature space of w_t, and $\hat{w} = \Pi_{\neg w_t} w_{t+1} \in \mathbb{R}^{d_t - d_{t_1}}$ denotes the vector consisting of elements of w_{t+1} which are not in the feature space of w_t,

$$
\begin{aligned}
w_{t+1} &= [\tilde{w}_{t+1}, \hat{w}_{t+1}] \\
&= \underset{\substack{w = [\tilde{w}, \hat{w}] : \\ l_t = 0}}{\operatorname{argmin}} \; \frac{1}{2}\|\tilde{w} - w_t\|^2 + \frac{1}{2}\|\hat{w}\|^2
\end{aligned}
\tag{1}
$$

where $l_t = l(w, (x_t, y_t))$ is the loss at round t , which can be written as,

$$
l_t = l(w, (x_t, y_t)) = \max\{0, 1 - y_t(\tilde{w} \cdot \tilde{x}_t) - y_t(\hat{w} \cdot \hat{x}_t)\}. \tag{2}
$$

Note that the definition of $\tilde{x}_t = \Pi_{\tilde{w}} x_t$ and $\hat{x}_t = \Pi_{\hat{w}} x_t$ are similar to the ones of $\tilde{w}$ and $\hat{w}$ respectively.

In the above constrained optimization problem, if the existing classifier w_t predicts the right label with the current instance x_t, i.e., $l_t = \max\{0, 1 - y_t(w_t \cdot \tilde{x}_t)\} = 0$, then we can easily know that the optimal solution is $\tilde{w} = w_t, \hat{w} = (0, \ldots, 0)$, that is, $w_{t+1} = [w_t, 0, \ldots, 0]$.

On the other hand, if the existing classifier makes a wrong prediction, the algorithm forces the updated classifier to satisfy the constraint in Eq. (1). At the same time, it also forces $\tilde{w}_{t+1}$ close to w_t in order to inherit information and let $\hat{w}_{t+1}$ be small to minimize structural risk and avoid overfitting. The solution to Eq. (1) has a simple closed form,

$$
w_{t+1} = [w_t + \tau_t y_t \tilde{x}_t, \tau_t y_t \hat{x}_t], \;\; where\ \tau_t = l_t / \|x_t\|^2 \tag{3}
$$

We now discuss the derivation of the update strategy.

- In case that the dimension of the new classifier does not change, i.e., $d_t = d_{t-1}$, the problem degenerates to an online learning problem where $\hat{w}_{t+1}$ disappears and $w_{t+1} = \tilde{w}_{t+1}$.
- In case that $d_t > d_{t-1}$ and $l_t = 0$, the optimal solution is $\tilde{w}_{t+1} = w_t$ and $\hat{w}_{t+1} = (0, \cdots, 0)$.
- In case that $d_t > d_{t-1}$ and $l_t > 0$, we solve Eq. (1) to obtain the solution.

To solve Eq.(1), we use the Lagrangian function and the Karush-Khun-Tucker conditions [44] on Eq.(2) and obtain

$$
\begin{aligned}
L(w, \tau) &= \frac{1}{2}\|\tilde{w} - w_t\|^2 + \frac{1}{2}\|\hat{w}\|^2 \\
&\quad + \tau(1 - y_t(\tilde{w} \cdot \tilde{x}_t) - y_t(\hat{w} \cdot \hat{x})), \\
\tilde{w} &= w_t + \tau y_t \tilde{x}_t; \quad \hat{w} = \tau y_t \hat{x}_t
\end{aligned}
\tag{4}
$$

where τ is a Lagrange multiplier. Plugging the last two equations into the first one, taking the derivative of $L(\tau)$ with respect to τ and setting it to zero, we can obtain

$$
\begin{aligned}
L(\tau) &= -\frac{1}{2}\tau^2\|\tilde{x}_t\|^2 - \frac{1}{2}\tau^2\|\hat{x}\|^2 + \tau - \tau y_t(w_t \cdot \tilde{x}) \\
\tau_t &= \frac{1 - y_t(w_t \cdot \tilde{x}_t)}{\|\tilde{x}\|^2 + \|\hat{x}_t\|^2} = \frac{l_t}{\|x_t\|^2}
\end{aligned}
\tag{5}
$$

So, the update strategy is $w_{t+1} = [w_t + \tau_t y_t \tilde{x}_t, \tau_t y_t \hat{x}_t]$, where $\tau_t = l_t / \|x_t\|^2$. In addition, this update rule is also applied when $l_t = 0$. So we can take it as a general update rule.

From Eq. (1), we can see that the update strategy of the OL_{SF} algorithm is rigorous because the new classifier needs to predict the current instance correctly. This may make the model sensitive to noise, especially label noise [2]. In order to avoid this drawback, we give two general updated variants of the OL_{SF} algorithm which use the soft-margin technique by introducing a slack variable ξ into the optimization problem. The first one is abbreviated as OL_{SF}-I. Its objective function scales linearly with ξ, namely,

$$w_{t+1} = \underset{\substack{w=[\tilde{w},\hat{w}]:\\ l_t\le\xi;\xi\ge 0}}{\operatorname{argmin}} \quad \frac{1}{2}\|\tilde{w}-w_t\|^2+\frac{1}{2}\|\hat{w}\|^2+C\xi \tag{6}$$

The second one, OL_{SF}-II, is the same as OL_{SF}-I except that its objective function scales quadratically with the slack variable ξ, i.e.,

$$w_{t+1} = \underset{\substack{w=[\tilde{w},\hat{w}]:\\ l_t\le\xi}}{\operatorname{argmin}} \quad \frac{1}{2}\|\tilde{w}-w_t\|^2+\frac{1}{2}\|\hat{w}\|^2+C\xi^2 \tag{7}$$

In these two optimization problems, parameter C is a positive number which is a tradeoff between rigidness and slackness. A larger value of C implies a more rigid update step.

The update strategy of OL_{SF}-I and OL_{SF}-II also shares the simple closed form $w_{t+1}=[w_t+\tau_t y_T\tilde{x}_t, \tau y_t\hat{x}_t]$, where

$$\tau_t=\min\{C,\frac{l_t}{\|x_t\|^2}\}\ (I)\ \ or\ \ \tau_t=\frac{l_t}{\|x_t\|^2+\frac{1}{2C}}\ (II).$$

The update strategies of OL_{SF}-I and OL_{SF}-II are similar to the OL_{SF} algorithm, so we omit their details due to space constraints.

The sparsity strategy

In many applications, the dimension of training instances increases rapidly and we need to select a relatively small number of features.

In our study, we introduce a parameter to control the proportion of the features used. For example, in each trial t, the learner presents a classifier $w_t\in\mathbb{R}^{d_{t-1}}$ to classify instance $x_t\in\mathbb{R}^{d_t}$ where $d_{t-1}\le d_t$. After the update operation, a projection and a truncation are introduced to prune redundant features based on the parameter B, which locates in $[0,1]$. Namely, we require the learner only retain at most a proportion of B nonzero elements of $w_t\in\mathbb{R}^{d_{w_t}}$, i.e. $\|w_t\|_0\le B\cdot d_{w_t}$. Specifically, if the resulting classifier w_t has more than a proportion of B nonzero elements, we will simply keep the proportion of B elements in w_t with the largest absolute weights, as demonstrated in Algorithm 3. In this way, at most a proportion of B features are used in the model and sparsity is introduced.

We introduce a projection step because one single truncation step does not work well. Although the truncation selects the B largest elements, this does not guarantee the numerical values of the unselected attributes are sufficiently small and may potentially lead to poor performance [3]. When projecting a vector to an L_1 ball, most of its numerical values are concentrated to its largest elements, and then removing the smallest elements will result in a small change to the original vector. Specifically, the projection is,

$$\check{w}_{t+1}=\min\{1,\frac{\lambda}{\|\bar{w}_{t+1}\|_1}\}\bar{w}_{t+1}, \tag{8}$$

where λ is the a positive regularization parameter.

5 Theoretical Analysis

In this section, we derive performance bounds of the OL_{SF} algorithm and its two variants OL_{SF}-I and OL_{SF}-II. There are four theorems and one lemma in this section. The first theorem discusses the upper bound of the cumulative squared hinge loss of OL_{SF} when data are linearly separable, and the second derives the bound when data are linearly inseparable. The third and the fourth theorems relate to the upper bounds of the OL_{SF}-I and OL_{SF}-II algorithms respectively.

If instance x_t is falsely predicted, then $y_t(w_t\cdot\Pi_{w_t}x_t)<0$, and the loss function $l_t>1$. So the cumulative squared hinge loss $\sum_t l_t^2$ is an upper bound of the number of false predictions [2]. Therefore, the loss bound will be the upper bound of the total number of false predictions and the cumulative squared hinge loss. Our bounds essentially prove that our algorithms cannot do much worse than the best fixed prediction, which is chosen in hindsight for any sequence of instances.

For clarity, we use two abbreviations throughout the paper. We denote by l_t the instantaneous loss suffered by our algorithm at round t. In addition, we denote by l_t^* the loss of an off-line predictor at round t. Formally, let $u\in\mathbb{R}^{d_T}$ be an arbitrary vector in $\mathbb{R}^{d_T}$, we define l_t and l_t^* as follows,

$$l_t=l(w_t;(\Pi_{w_t}x_t,y_t))\quad and\quad l_t^*=l(\Pi_{x_t}u;(x_t,y_t)) \tag{9}$$

Then, we have Lemma 1 as follows.

Lemma 1. Let $(x_1,y_1),\ldots,(x_T,y_T)$ be a sequence of training instances, where $x_t\in\mathbb{R}^{d_t}, d_{t-1}\le d_t$ and $y_t\in\{+1,-1\}$ for all t. Let the learning rate $\tau_t\in\left\{\frac{l_t}{\|x_t\|^2}, min\{C,\frac{l_t}{\|x_t\|^2}\}, \frac{l_t}{\|x_t\|^2+\frac{1}{2C}}\right\}$, as given in Algorithm 2. Then, the following bound holds for any $u\in\mathbb{R}^{d_T}$, $\sum_{t=1}^T\tau_t(2l_t-\tau_t\|x_t\|^2-2l_t^*)\le\|u\|^2$

proof 1. Define ∇_t to be $\|w_t-\Pi_{w_t}u\|^2-\|w_{t+1}-\Pi_{w_{t+1}}u\|^2$. We prove the lemma by summing up all ∇_t over t in $1,\ldots,T$ and bounding this sum. Note that $\sum_t\nabla_t$ is a telescopic sum which collapses to

$$\begin{aligned}\sum_{t=1}^{T-1}\nabla_t&=\sum_{t=1}^{T-1}(\|w_t-\Pi_{w_t}u\|^2-\|w_{t+1}-\Pi_{w_{t+1}}u\|^2)\\&=\|w_1-\Pi_{w_1}u\|^2-\|w_T-\Pi_{w_T}u\|^2,\end{aligned} \tag{10}$$

where w_1 is initialized as a zero vector, and $\|w_T-\Pi_{w_T}u\|^2\ge 0$ always holds. Thus, we can upper bound the right-hand side of the above equation by $\|\Pi_{w_1}u\|^2$,

$$\sum_{t=1}^{T-1}\nabla_t\le\|\Pi_{w_1}u\|^2. \tag{11}$$

We now turn to bound every single ∇_t. If the minimum margin requirement is not violated on round t , i.e. l_t =0, then $\tau_t=0$ and hence $\nabla_t\le 0$. Now we only focus on rounds on which $l_t > 0$. With the update strategy $\bar{w}_{t+1} = [w_t + \tau_t y_t \Pi_{w_t} x_t, \tau_t y_t \Pi_{\neg w_t} x_t]$ where $\Pi_{\neg w_t} x_t$ is a vector consisting of elements in x which are not in the same feature space of w_t. And in light of the fact that $\check{w}_{t+1} \leq \bar{w}_{t+1}$ and $w_{t+1} \leq \check{w}_{t+1}$ we have

$$\begin{aligned}\nabla_t =&\|w_t - \Pi_{w_t} u\|^2 - \|w_{t+1} - \Pi_{w_{t+1}} u\|^2 \\ \geq&\|w_t - \Pi_{w_t} u\|^2 - \|w_t + \tau_t y_t \Pi_{w_t} x_t - \Pi_{w_t} u\|^2 \\ &- \|\tau_t y_t \Pi_{\neg w_t} x_t - \Pi_{w_{t+1}\ w_t} u\|^2 \\ =&-2\tau_t y_t \Pi_{w_t} x_t (w_t - \Pi_{w_t} u) - \tau_t^2 \|\Pi_{w_t} x_t\|^2 \\ &- \|\tau_t y_t \Pi_{\neg w_t} x_t - \Pi_{w_{t+1}/w_t} u\|^2\end{aligned} \tag{12}$$

From $l_t = 1 - y_t(w_t \cdot \Pi_{w_t} x_t)$ and $l_t^* \geq 1 - y_t(\Pi_{x_t} u \cdot x_t)$, we have $y_t(w_t \cdot \Pi_{w_t} x_t) = 1 - l_t$ and $y_t(\Pi_{w_t} u \cdot \Pi_{w_t} x_t) + y_T(\Pi_{w_{t+1}/w_t} u \cdot \Pi_{w_{t+1}/w_t} x_t) \geq 1 - l_t^*$. Using these two facts in Eq. (12) gives,

$$\begin{aligned}\nabla_t \geq& 2\tau_t(y_t \Pi_{w_t} x_t \Pi_{w_t} u + y_t \Pi_{w_{t+1}/w_t} x_t \Pi_{w_{t+1}/w_t} u \\ &- y_t \Pi_{w_t} x_t w_t) - \tau_t^2 \|x_t\|^2 - \|\Pi_{w_{t+1}/w_t} u\|^2 \\ \geq& \tau_t(2l_t - 2l_t^* - \tau_t \|x_t\|^2) - \|\Pi_{w_{t+1}/w_t} u\|^2.\end{aligned} \tag{13}$$

Summing up ∇_t over all t and comparing the lower bound of Eq. (13) with the upper bound in Eq.(11), we can obtain

$$\sum_{t=1}^{T} \tau_t(2l_t - 2l_t^* - \tau_t \|x_t\|^2) \leq \|\Pi_{w_1} u\|^2 + \sum_{t=1}^{T-1} \|\Pi_{w_{t+1}/w_t} u\|^2.$$

The lemma is proved.

Below we first prove a loss bound for the OL$_{SF}$ algorithm in the linearly separable case. We assume that there is a classifier $u \in \mathbb{R}^{d_T}$ such that $y_t(\Pi_{x_t} u \cdot x_t) > 0$ for all $t \in \{1, \ldots, T\}$. Without loss of generality, we assume that classifier u is scaled such that $y_t(\Pi_{x_t} u \cdot x_t) \geq 1$. The loss of u is zero on all T instances in the sequence. Then, we have the following bound of the cumulative squared loss of OL$_{SF}$.

Theorem 1. Let $(x_1, y_1), \ldots, (x_T, y_T)$ be a sequence of instances where $x_t \in \mathbb{R}^{d_t}$, $d_{t-1} \leq d_t$, $y_t \in \{+1, -1\}$ and $\|x_t\| \leq R$ for all t. Assume that there exists a classifier u such that $l_t^* = 0$ for all t. Then, the cumulative squared loss of OL$_{SF}$ on the sequence is bounded by

$$\sum_{t=1}^{T} l_t^2 \leq \|u\|^2 R^2.$$

proof 2. Since $l_t^* = 0$ for all t , Lemma 1 implies that,

$$\sum_{t=1}^{T} \tau_t(2l_t - \tau_t \|x_t\|^2) \leq \|u\|^2. \tag{14}$$

According to the definition $\tau_t = \frac{l_t}{\|x_t\|^2}$, we have

$$\sum_{t=1}^{T} \frac{l_t^2}{\|x_t\|^2} \leq \|u\|^2$$

and

$$\sum_{t=1}^{T} l_t^2 \leq \|u\|^2 R^2.$$

Hence, the theorem is proved.

The following theorems generalize the linearly separable case. We consider that the classifier u cannot perfectly separate the training data. In addition, we assume that the input sequence is normalized so that $\|x_t\|^2 = 1$. Then, we have the following bounds of the cumulative squared loss of the OL$_{SF}$ algorithm.

Theorem 2. Let $(x_1, y_1), \ldots, (x_T, y_T)$ be a sequence of instances where $x_t \in \mathbb{R}^{d_t}$, $d_{t-1} \leq d_t$, $y_t \in \{+1, -1\}$ and $\|x_t\|^2 = 1$ for all t. Then, for any vector $u \in \mathbb{R}^{d_T}$, the cumulative squared loss of OL$_{SF}$ on the sequence is bounded by

$$\sum_{t=1}^{T} l_t^2 \leq \left(\|u\| + 2\sqrt{\sum_{t=1}^{T} (l_t^*)^2} \right)^2. \tag{15}$$

proof 3. Since $\|x_t\|^2 = 1$, τ_t and l_t are equal, according to Lemma 1, we have $\sum_{t=1}^{T} l_t^2 \leq \|u\|^2 + \sum_{t+1}^{T} 2l_t \cdot l_t^*$. Denote

$$L_T = \sqrt{\sum_{t=1}^{T} l_t^2} \ \ and \ \ U_T = \sqrt{\sum_{t=1}^{T} (l_t^*)^2}.$$

By using the Cauchy-Schwartz inequality to bound the right-hand side of Eq.(15), we obtain

$$L_T^2 \leq \|u\|^2 + 2L_T U_T.$$

Therefore, to obtain an upper bound of L_T, we need to find the largest solution of $L_T^2 - 2U_T L_T - \|u\|^2 = 0$, i.e.,

$$U_T + \sqrt{U_T^2 + \|u\|^2}.$$

Using the fact that $\sqrt{\alpha + \beta} \leq \sqrt{\alpha} + \sqrt{\beta}$, we have

$$L_T \leq \|u\| + 2U_T.$$

Furthermore, we can obtain

$$\sum_{t=1}^{T} l_t^2 \leq \left(\|u\| + 2\sqrt{\sum_{t=1}^{T} (l_t^*)^2} \right)^2$$

and the theorem is proved.

Next we derive the bound for OL$_{SF}$-I. The following theorem provides an error rate bound of OL$_{SF}$-I based on the total number of falsely predicted instances that $y_t \neq sign(w_t \cdot \Pi_{w_t} x_t)$.

Theorem 3. Let $(x_1, y_1), \ldots, (x_T, y_T)$ be a sequence of instances, where $x_t \in \mathbb{R}^{d_t}$, $d_{t-1} \leq d_t$, $y_t \in \{+1, -1\}$ and $\|x_t\|^2 \leq R^2$ for all t. For any vector $u \in \mathbb{R}^{d_T}$, the number of false predictions by OL$_{SF}$-I is bounded by,

$$\max\{R^2, \frac{1}{C}\} \left(\|u\|^2 + 2C \sum_{t=1}^{T} l_t^* \right),$$

where C is the parameter in OL$_{SF}$-I.

proof 4. If OL$_{SF}$-I outputs a false prediction at round t, then $y_t(w_t \cdot \Pi_{w_t} x_t) \leq 0$, so $l_t \geq 1$. Under the assumption $\|x_t\|^2 \leq R^2$ and the definition $\tau_t = min\{l_t/\|x_t\|^2, C\}$, for the error occurring at round t, we have

$$\min\{\frac{1}{R^2}, C\} M \leq \sum_{t=1}^{T} \tau_t l_t,$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2563424, IEEE Transactions on Knowledge and Data Engineering

where M is the number of false predictions by OL$_{SF}$-I. Based on the definition of τ_t , we know that $\tau_t l_t^* \leq C l_t^*$ and $\tau_t \|x_t\|^2 \leq l_t$. Plugging these two inequalities into Lemma 1 gives the result,

$$\sum_{t=1}^{T} \tau_t l_t \leq \|u\|^2 + 2C \sum_{t=1}^{T} l_t^*.$$

Combining the above two inequations, we obtain that

$$\min\{1/R^2, C\} M \leq \|u\|^2 + 2C \sum_{t=1}^{T} l_t^*.$$

The theorem is proved by multiplying both sides of the above inequation with $\max\{R^2, 1/C\}$.

Now, we turn to the bound analysis for OL$_{SF}$-II.

Theorem 4. Let $(x_1, y_1), \ldots, (x_T, y_T)$ be a sequence of instances where $x_t \in \mathbb{R}^{d_t}$, $d_{t-1} \leq d_t$, $y_t \in \{+1, -1\}$ and $\|x_t\|^2 \leq R$ for all t . Then, for any classifier (vector) $u \in \mathbb{R}^{d_T}$, the cumulative squared loss of OL$_{SF}$-II is bounded by,

$$\sum_{t=1}^{T} l_t^2 \leq \left(R^2 + \frac{1}{2C}\right)\left(\|u\|^2 + 2C \sum_{t=1}^{T} (l_t^*)^2\right).$$

proof 5. Lemma 1 states that

$$\|u\|^2 \geq \sum_{t=1}^{T} (2\tau_t l_t - \tau^2 \|x_t\|^2 - 2\tau_t l_t^*).$$

Define $\alpha = 1/\sqrt{2C}$, and by subtracting the non-negative term $(\alpha\tau_t - l_t^*/\alpha)^2$ from each result on the right-hand side of the above inequality, we can obtain

$$\begin{aligned} \|u\|^2 &\geq \sum_{t=1}^{T} (2\tau_t l_t - \tau^2 \|x_t\|^3 - 2\tau_t l_t^* - (\alpha\tau_t - l_t^*/\alpha)^2) \\ &= \sum_{t=1}^{T} (2\tau_t l_t - \tau^2 (\|x_t\|^2 + \alpha^2) - (l_t^*)^2/\alpha^2). \end{aligned} \tag{16}$$

Plugging in the definition of α, and using the definition $\tau_t = l_t/(\|x_t\|^2 + 1/(2C))$, we can obtain the following lower bound,

$$\|u\|^2 \geq \sum_{t=1}^{T} \left(\frac{l_t^2}{\|x_t\|^2 + \frac{1}{2C}} - 2C(l_t^*)^2\right).$$

Replacing $\|x_t\|^2$ with its upper bound of R^2 and rearranging the terms gives the desired bound.

6 EXPERIMENTS

In this section, we empirically evaluate the performance of OL$_{SF}$ and its two variants OL$_{SF}$-I and OL$_{SF}$-II[1].

The experiments are conducted from four aspects. Firstly, we evaluate the performance of the proposed three algorithms with respect to classification accuracy, projected feature space B, and tradeoff C in Section 6.1. Secondly, we evaluate the update strategy and the sparse strategy used in the three algorithms by comparing with three benchmark methods in Section 6.2. Thirdly, we compare the proposed algorithms with the state-of-the-art online feature selection algorithms in Section 6.3. Finally, we test the applications of the proposed algorithms on two real-world trapezoidal data streams in Section 6.4.

Experimental Setup. We test on 12 UCI data sets and two real-world large-scale streams as listed in Table 2.

To simulate trapezoidal streams, we split the data sets into 10 chunks, where each chunk carries only 10% instances and a variant number of features. For example, the first data chunk carries the first 10% instances with the first 10% features. The second data chunk carries the second 10% instances with another 10% features (in total 20% features).

We measure the performance in terms of the average prediction accuracy. The experiments are repeated 20 times with a random permutation on the data sets. The results are reported by an average over the 20 repeats.

We set λ to be 30, C from 10^{-4} to 10^4 with a step of 10^1, and B from 0 to 1. The parameters are chosen with cross validation.

Table 2
The data sets used in the experiments

Dataset	♯ instances	♯ Dimensions
wpbc	198	34
ionosphere	351	35
wdbc	569	31
isolet	600	618
wbc	699	10
german	1,000	24
svmguide3	1,234	21
splice	3,175	60
HAPT	3,266	562
spambase	4,601	57
magic04	19,020	10
a8a	32,561	123
rcv1	697,641	47,236
URL	2,396,130	3,231,961

6.1 Experiment I: Comparisons between OL$_{SF}$ and its two variants

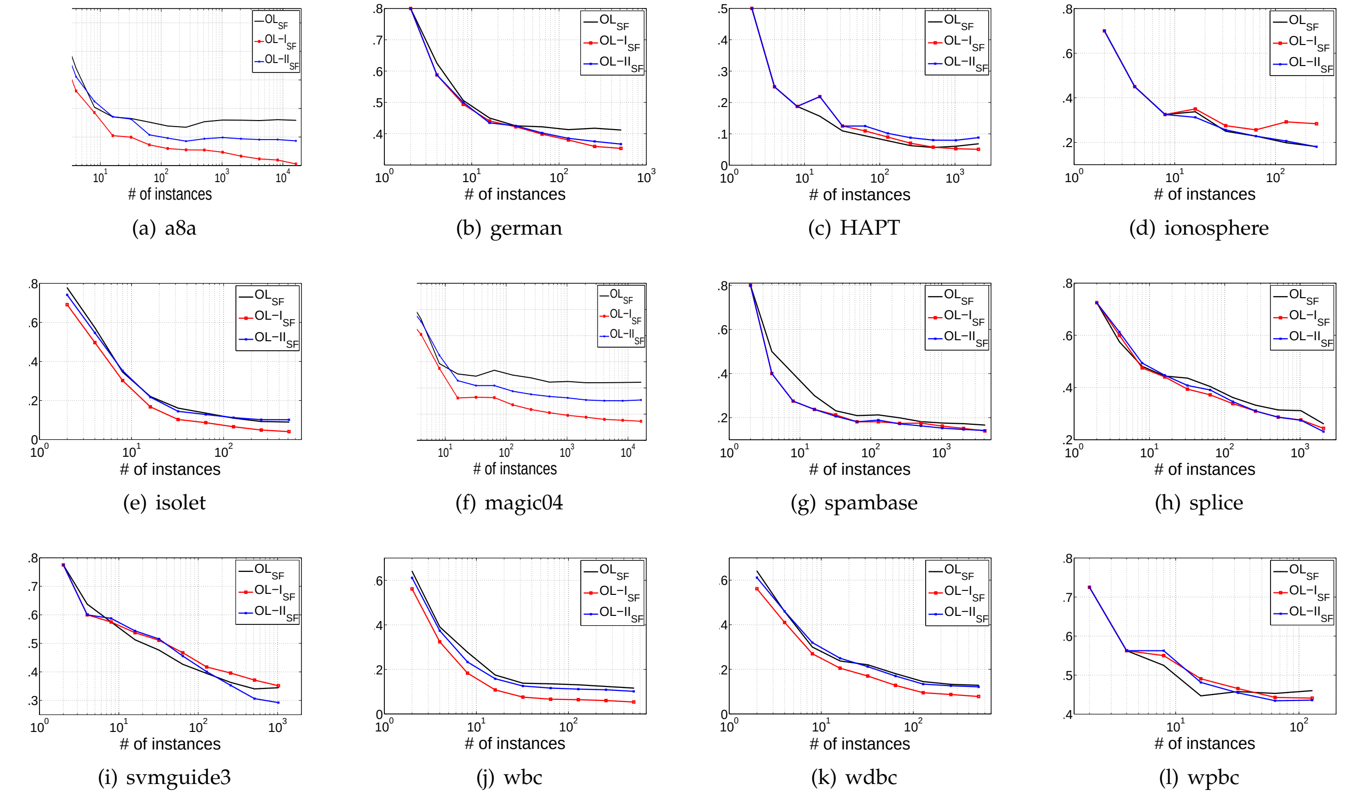
In this part, we present the empirical results of the three algorithms on the 12 UCI benchmark data sets.

Table 3 summarizes the performance of the three algorithms on a projected feature space. We can observe that OL$_{SF}$-I performs the best on six data sets, a8a, german, HAPT, magic04, spambase, wpbc, OL$_{SF}$-II performs the best on the remaining six date sets, ionosphere, isolet, splice, svmguide3, wbc, wdbc. Among the three algorithms, OL$_{SF}$ performs the worst on all the 12 data sets. This is because the 12 UCI data sets contain noise, OL$_{SF}$ which relies on a strict update strategy overfits the noise and thus performs the worst. In contrast, OL$_{SF}$-I and OL$_{SF}$-II using a "soft" update strategy can avoid overfitting. Furthermore, we can see that OL$_{SF}$-I scales well on large data sets, while OL$_{SF}$-II performs the best on small data sets. This is because OLSF-I scales linearly with the slack variable.

Fig. 2 shows the error rate with respect to the streaming iterations on the 12 data sets. Similar to the above results, we can observe that both OL$_{SF}$-I and OL$_{SF}$-II consistently outperform OL$_{SF}$. In addition, the performance gain of OLSF-I and OLSF-II raises with a large probability when

1. The Matlab source codes are available online at *https://github.com/BlindReview/onlineLearning.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2563424, IEEE Transactions on Knowledge and Data Engineering



(a) a8a (b) german (c) HAPT (d) ionosphere

(e) isolet (f) magic04 (g) spambase (h) splice

(i) svmguide3 (j) wbc (k) wdbc (l) wpbc

Figure 2. Comparison among the proposed three algorithms OL$_{SF}$, OL$_{SF}$-I and OL$_{SF}$-II on the 12 UCI data sets. We can observe that OL$_{SF}$-I and OL$_{SF}$-II outperform OL$_{SF}$ because their "soft" update strategies can avoid overfitting to noise.

Table 3
The average number of prediction errors on the 12 UCI data sets

Algorithms	a8a	german	HAPT
OL$_{SF}$	12673.5 $\pm$ 75.9	415.9 $\pm$ 15.6	257.0 $\pm$ 8.5
OL$_{SF}$-I	**11204.7$\pm$713.1**	**366.9$\pm$8.8**	**167.0 $\pm$ 7.1**
OL$_{SF}$-II	11317.2$\pm$233.1	366.9$\pm$12.8	191.0 $\pm$ 9.9
Algorithms	ionosphere	isolet	magic04
OL$_{SF}$	55.0 $\pm$ 2.8	23.5 $\pm$4.9	8051.3 $\pm$ 49.0
OL$_{SF}$-I	55.0 $\pm$2.8	21.5$\pm$2.1	**6732.3$\pm$73.3**
OL$_{SF}$-II	**50.5 $\pm$ 6.4**	**18.0**$\pm$4.2	6924.5 $\pm$ 39.6
Algorithms	spambase	splice	svmguide3
OL$_{SF}$	1132.1 $\pm$ 29.7	1314.6 $\pm$ 30.3	396.7 $\pm$ 15.8
OL$_{SF}$-I	**1004.5$\pm$25.6**	1243.7 $\pm$ 13.6	359.1 $\pm$ 42.9
OL$_{SF}$-II	1013.2$\pm$26.1	**1238.8$\pm$16.8**	**357.5$\pm$26.9**
Algorithm	wbc	wdbc	wpbc
OL$_{SF}$	37.5 $\pm$ 0.7	43.5 $\pm$ 3.5	88.5 $\pm$ 2.1
OL$_{SF}$-I	35.5 $\pm$ 0.7	39.5 $\pm$ 0.7	**82.0 $\pm$ 8.5**
OL$_{SF}$-II	**34.0 $\pm$ 4.2**	**38.5** $\pm$ **4.9**	83.0 $\pm$ 1.4

new training instances arrive. This observation validates that OL$_{SF}$-I and OL$_{SF}$-II, by using slack variants to obtain soft update, can avoid overfitting to noise.

Fig. 3 shows the performance of the three algorithms under different projected feature space B. We can observe that OL$_{SF}$-I and OL$_{SF}$-II often outperform OL$_{SF}$. The results show the robustness of OL$_{SF}$-I and OL$_{SF}$-II under different subspace defined by B.

Fig. 4 shows the performance of the three algorithms under different tradeoff C. From the results, we can observe that varying the parameter C can alter the error rate of OL$_{SF}$-I and OL$_{SF}$-II. The larger of C, the closer of OL$_{SF}$-I to OL$_{SF}$. This is because the parameter τ_t in OL$_{SF}$-I is smaller than both parameter C and τ_t in OL$_{SF}$. When C is very large, OL$_{SF}$-I degenerates to OL$_{SF}$.

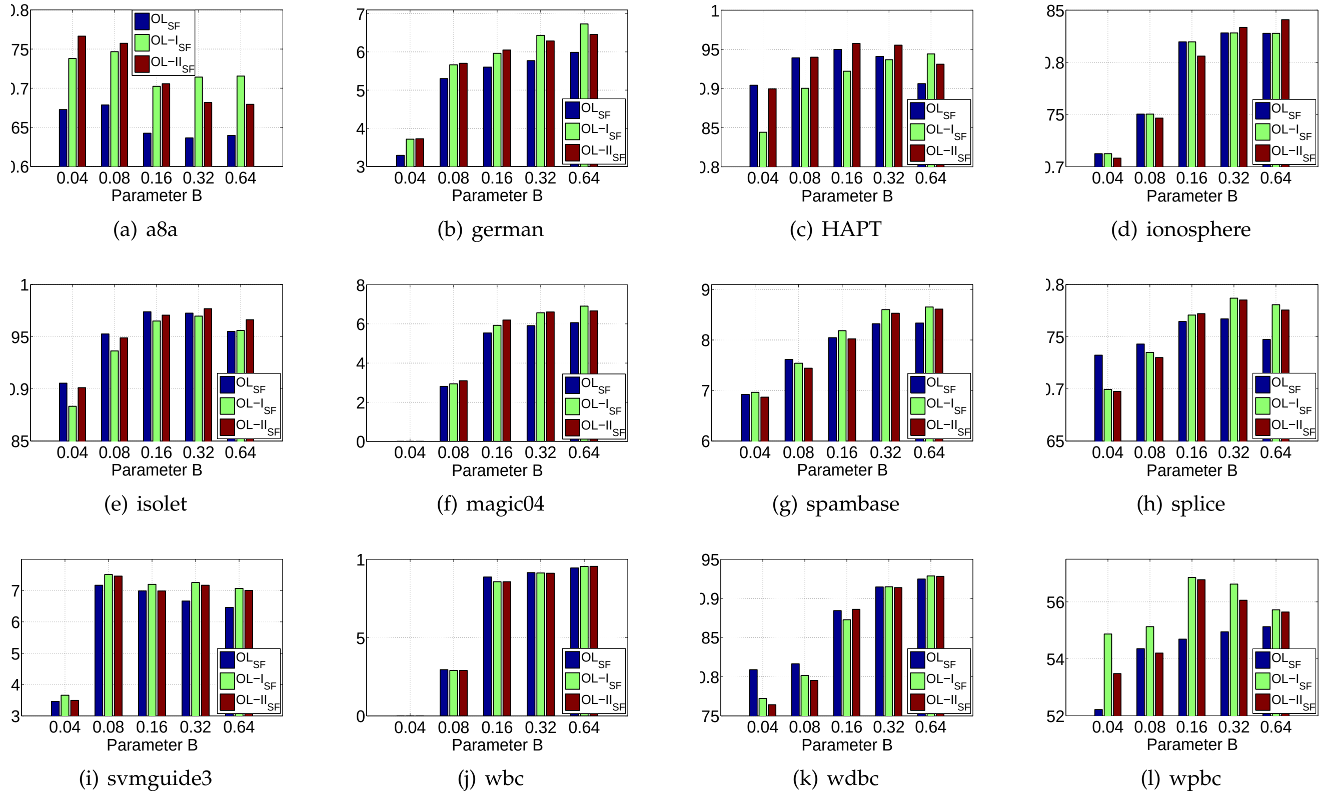
6.2 Experiment II: Comparisons with benchmarks

We compare the proposed algorithms with three benchmark methods. According to the similar performance of OL$_{SF}$-I and OL$_{SF}$-II, we use OL$_{SF}$-I as the representative algorithm in this part.

Now we introduce the three benchmark methods. The first algorithm is **OLI$_{SF}$-all**. Different from OL$_{SF}$-I that only uses a small projected feature space for learning, OLI$_{SF}$-all uses all features for learning. The second algorithm is **OLI$_{SF}$-rand** which uses randomly selected features for learning. The third algorithm is **OLI$_{SF}$-per** which uses the Perceptron update strategy for learning, i.e., $w_{t+1} = [w_t + y_t x_t, y_t x_t]$ [10]. We still use the 12 UCI data sets for our evaluation. The parameter settings are the same as in Experiments I.

Table 4 lists the average number of error predictions of the four algorithms on the 12 UCI data sets with different values of the parameter B. First, we can observe that OL$_{SF}$-I obtains the best results on 10 data sets out of 12. It even beats the OLI$_{SF}$-all algorithm which uses all the features for learning. Compared to the other two algorithms OLI$_{SF}$-rand and OLI$_{SF}$-per, OL$_{SF}$-I outperforms them under different B. The OL$_{SF}$I-rand algorithm randomly chooses a fixed proportion of features which receives the worst performance on all the 12 data sets. The OL$_{SF}$I-per algorithm which uses the Perceptron update strategy has higher error

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2563424, IEEE Transactions on Knowledge and Data Engineering



(a) a8a (b) german (c) HAPT (d) ionosphere

(e) isolet (f) magic04 (g) spambase (h) splice

(i) svmguide3 (j) wbc (k) wdbc (l) wpbc

Figure 3. Performance comparison with respect to the projected feature space B. The results show that OL$_{SF}$-I and OL$_{SF}$-II are robust algorithms under different projected subspaces B.

Table 4
The average number of error predictions on the 12 UCI data sets with respect to parameter B.

	Algorithms	a8a	german	HAPT	magic04	ionosphere	isolet
$B = 0.04$	OL$_{SF}$-I	87.2 ± 8.8	56.8 ± 21.3	19020.0 ± 0.0	313.2 ± 55.2	788.4 ± 38.2	628.6 ± 42.5
	OL$_{SF}$I-rand	133.0 ± 4.5	190.2 ± 10.0	19020.0 ± 0.0	1318.6 ± 31.8	1041.0 ± 9.3	791.2 ± 11.1
	OL$_{SF}$I-per	141.9 ± 6.3	92.5 ± 40.6	19020.0 ± 0.0	737.0 ± 171.8	1034.7 ± 29.0	868.3 ± 18.2
$B = 0.16$	OL$_{SF}$-I	83.2 ± 6.9	**<u>15.7 ± 3.9</u>**	7379.2 ± 98.2	164.1 ± 14.8	**<u>348.3 ± 49.4</u>**	402.2 ± 39.7
	OL$_{SF}$I-rand	112.5 ± 6.4	144.4 ± 8.3	13107.3 ± 53.1	1158.1 ± 27.9	720.3 ± 13.3	582.4 ± 11.3
	OL$_{SF}$I-per	141.2 ± 5.8	35.4 ± 12.4	13143.5 ± 52.7	441.0 ± 51.1	828.9 ± 51.2	711.9 ± 17.1
$B = 0.64$	OL$_{SF}$-I	82.6 ± 3.7	25.6 ± 2.9	**<u>5882.2 ± 105.3</u>**	182.4 ± 41.9	364.4 ± 11.0	**<u>326.7 ± 7.0</u>**
	OL$_{SF}$I-rand	91.3 ± 4.8	69.7 ± 5.6	8361.8 ± 60.0	779.6 ± 15.2	570.2 ± 20.3	456.0 ± 18.9
	OL$_{SF}$I-per	83.4 ± 3.5	32.5 ± 3.1	6864.1 ± 49.5	420.6 ± 17.7	368.3 ± 51.7	365.0 ± 9.8
$B = 1.00$	OL$_{SF}$I-all	**<u>79.3 ± 3.3</u>**	16.7 ± 1.8	6634.3 ± 35.2	**<u>157.0 ± 8.9</u>**	360.9 ± 7.2	344.1 ± 7.1

	Algorithms	spambase	splice	svmguide3	wbc	wdbc	wpbc
$B = 0.04$	OL$_{SF}$-I	108.7 ± 38.4	683.0 ± 0.0	100.9 ± 12.4	850.2 ± 69.9	1397.2 ± 176.1	7488.4 ± 93.7
	OL$_{SF}$I-rand	375.8 ± 7.3	683.0 ± 0.0	234.7 ± 6.3	2026.5 ± 26.5	2808.7 ± 28.1	18249.4 ± 59.3
	OL$_{SF}$I-per	469.4 ± 14.1	683.0 ± 0.0	225.3 ± 8.6	2221.1 ± 43.5	2980.9 ± 80.9	20265.0 ± 285.0
$B = 0.16$	OL$_{SF}$-I	65.8 ± 13.1	77.2 ± 15.7	63.3 ± 8.7	**<u>728.1 ± 20.7</u>**	**<u>835.3 ± 109.0</u>**	**<u>8680.3 ± 316.9</u>**
	OL$_{SF}$I-rand	240.5 ± 10.4	405.5 ± 7.8	186.6 ± 6.4	1532.0 ± 30.4	1935.5 ± 38.0	16087.6 ± 102.8
	OL$_{SF}$I-per	327.0 ± 14.1	529.6 ± 8.5	220.5 ± 7.0	1308.8 ± 34.5	1248.8 ± 96.3	9659.6 ± 1444.9
$B = 0.64$	OL$_{SF}$-I	**<u>40.6 ± 4.3</u>**	**<u>31.3 ± 3.0</u>**	**<u>54.8 ± 3.1</u>**	683.7 ± 14.2	571.1 ± 15.3	9265.4 ± 115.4
	OL$_{SF}$I-rand	87.4 ± 7.0	79.2 ± 7.7	115.9 ± 8.5	1464.2 ± 33.4	1601.9 ± 23.4	15341.5 ± 71.0
	OL$_{SF}$I-per	63.4 ± 3.8	85.9 ± 6.3	60.0 ± 4.7	1244.8 ± 25.3	1003.7 ± 26.9	11325.6 ± 127.5
$B = 1.00$	OL$_{SF}$I-all	57.5 ± 3.7	82.9 ± 6.4	55.3 ± 2.7	1236.1 ± 29.5	983.6 ± 21.7	10243.0 ± 109.8

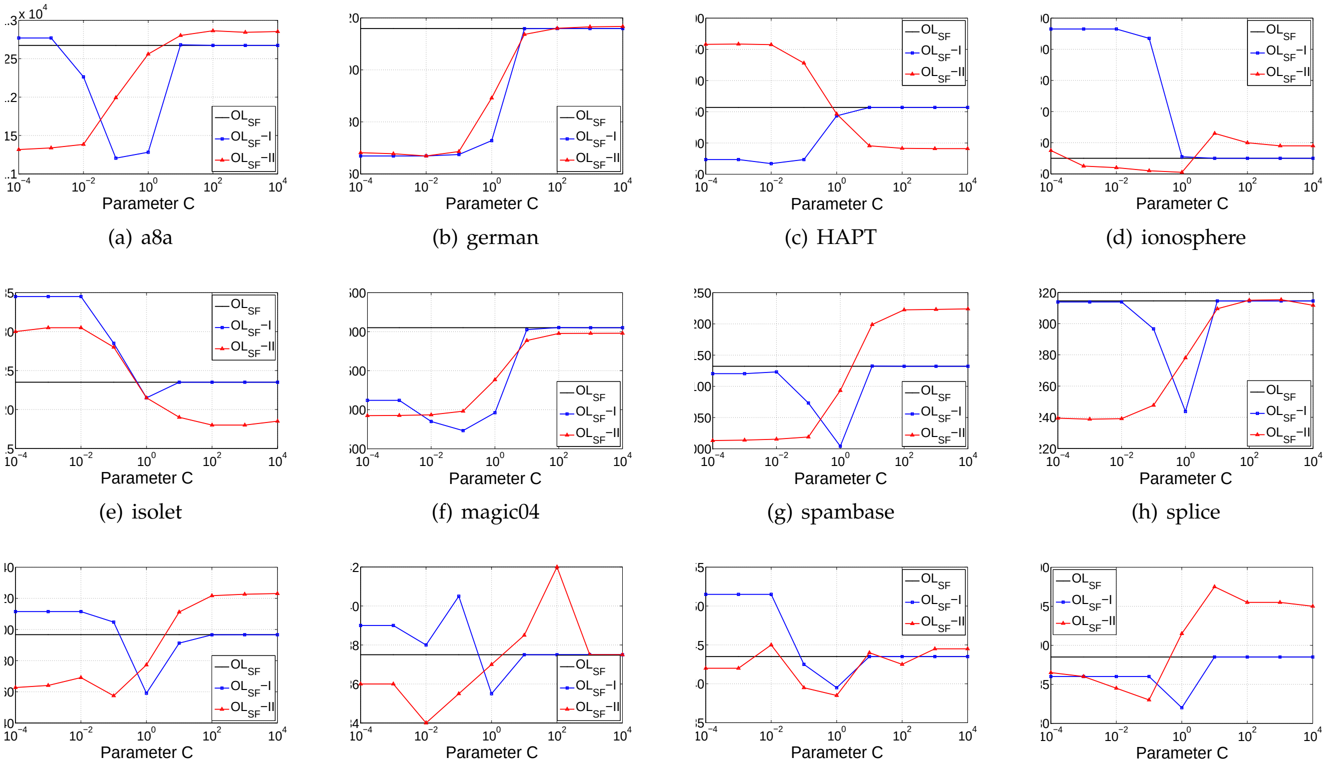
rates than OL$_{SF}$-I on all the 12 data sets, which shows that our update is better than the Perceptron update.

To sum up, the results show that the sparsity strategy in OL$_{SF}$-I can significantly improve the performance and our update strategy outperforms the Perceptron update strategy.

Fig. 5 shows the results of the online average error rates during the online learning on the 12 data sets. We can observe that the error rate of the algorithms decreases rapidly and becomes stable. OL$_{SF}$-I obtains the best results on all the data sets, which OLI$_{SF}$-rand obtains the worst results. The observation validate the results in Table 4.

To further examine the performance of these four algorithms, Fig. 6 shows the performance of the four algorithms with respect to different feature sets. The OL$_{SF}$-I algorithm outperforms the other three benchmark algorithms under the same feature sets. In particular, OL$_{SF}$-I significantly

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2563424, IEEE Transactions on Knowledge and Data Engineering

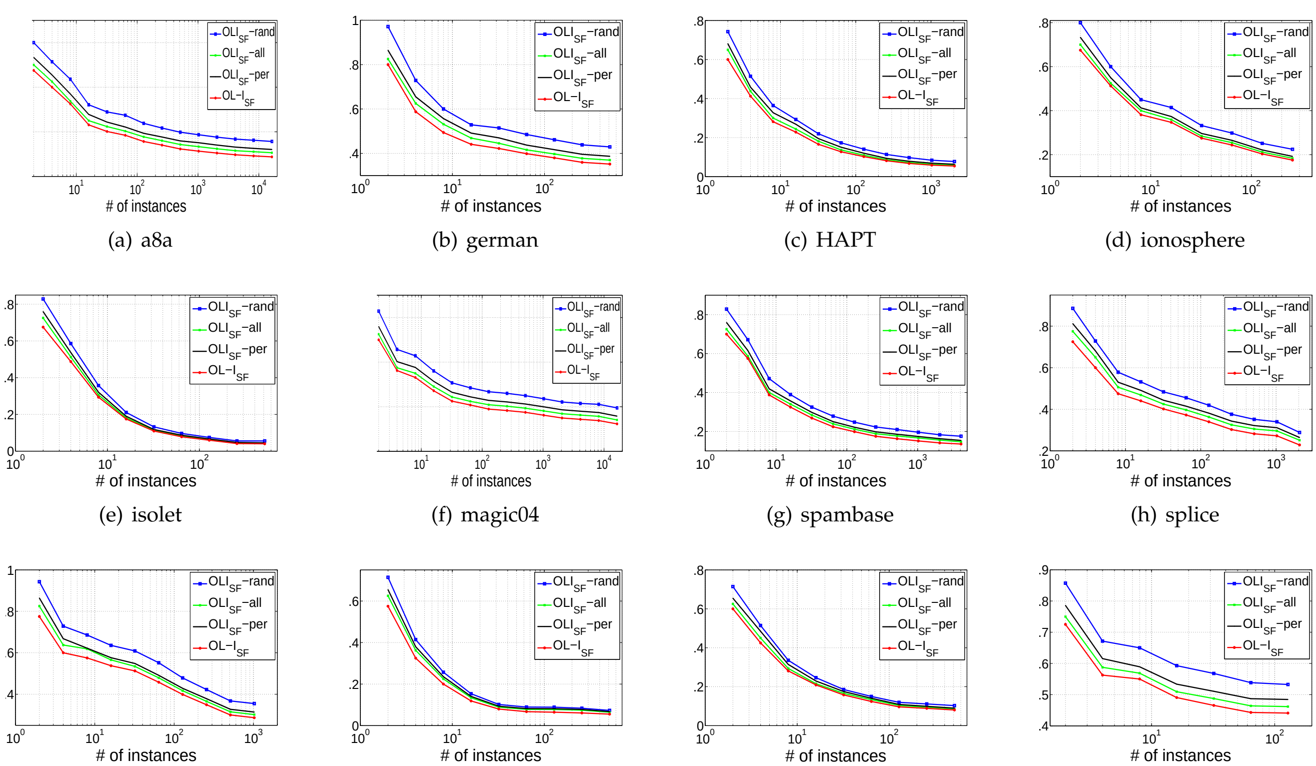


(a) a8a (b) german (c) HAPT (d) ionosphere

(e) isolet (f) magic04 (g) spambase (h) splice

(i) svmguide3 (j) wbc (k) wdbc (l) wpbc

Figure 4. The average number of error predictions with respect to parameter C. We can choose the best parameter for the algorithms on the 12 UCI data sets.



(a) a8a (b) german (c) HAPT (d) ionosphere

(e) isolet (f) magic04 (g) spambase (h) splice

(i) svmguide3 (j) wbc (k) wdbc (l) wpbc

Figure 5. Comparison of the four algorithms under online learning setting. We can observe that OL$_{SF}$-I obtains the best results on all the 12 data sets because its sparsity strategy can significantly improve the performance and outperforms the Perceptron update strategy.

outperforms the others when the subspace is very sparse, i.e., the parameter B is very small. The results show that the OL$_{SF}$-I algorithm can gain better sparsity and OL$_{SF}$-I performs well under a sparse feature space. This encouraging result verifies the efficacy of the proposed algorithms. Compared to OL$_{SF}$I-all that uses all features for learning, OL$_{SF}$-I achieves better results with sparser feature available.

6.3 Experiment III: Comparisons with the state-of-the-art online feature selection algorithms

In this section, we compare the proposed OL$_{SF}$-I and OL$_{SF}$-II algorithms with the Online Feature Selection algorithms (OFS for short) proposed by J. Wang et al. [3] and its variant OFS$_P$, i.e., OFS with partial feature sets.

The OFS algorithm can access all the features for training and efficiently identify a fixed number of relevant features for prediction by using a gradient-based online learning update strategy and an l_2-norm projected truncation approach. OFS_P assumes only a partial number of features can be selected based on a Bernoulli distribution and then used for learning. The original codes of OFS and OFS$_P$ can be obtained online *http://OFS.stevenhoi.org/*.

In this part, we set the parameter $B = 0.1$, i.e., we use 10% features for learning at each round t. The tradeoff parameter C ranges from 10^{-4} to 10^4. OL$_{SF}$-I and OL$_{SF}$-II use 50% of the features for learning before 10% training instances are observed. Then, the algorithm continuously observes additional 10% features at each new data chunk.

Table 5 and Fig. 7 show the average number of error predictions of the four algorithms. We can observe that OL$_{SF}$-I obtains the lowest error rate on the six data sets. Moreover, OL$_{SF}$-I significantly outperforms both OFS and OFS$_P$. When comparing OL$_{SF}$-II with OFS$_P$ and OFS, we can observe that OL$_{SF}$-II performs better on the six data sets than OFS$_P$. OL$_{SF}$-II also outperforms OFS on four data sets. This is because OL$_{SF}$-I and OL$_{SF}$-II have better update strategies than OFS and OFS$_P$ by adding a flexible learning rate τ_t. We can also observe that OL$_{SF}$-I and OL$_{SF}$-II are more stable because their standard deviations are significantly lower than OFS and OFS$_P$.

Table 5
Comparison with respect to the average number of error predictions.

Algorithms	a8a	german	magic04
OFS	9424.4 ± 2545.8	432.8 ± 13.6	6023.4 ± 1342.3
OFS$_P$	16931.0 ± 164.6	589.3 ± 33.9	10274.2 ± 172.1
OL$_{SF}$-I	**9322.7 ± 41.1**	**318.5 ± 7.3**	**5858.4 ± 29.6**
OL$_{SF}$-II	10709.3 ± 56.0	348.7 ± 11.6	5917.9 ± 55.9
Algorithms	spambase	splice	svmguide3
OFS	913.1 ± 157.8	735.4 ± 68.3	400.9 ± 66.8
OFS$_P$	1954.2 ± 78.7	1418.1 ± 70.5	701.5 ± 42.5
OL$_{SF}$-I	**616.6 ± 12.2**	**725.5 ± 18.8**	**374.2 ± 10.8**
OL$_{SF}$-II	690.7 ± 14.0	748.7 ± 16.0	382.1 ± 11.4

Furthermore, we compare the online prediction performance in Fig. 8. We can observe that the error rate varies at each iteration, where the curves of OL$_{SF}$-I and OL$_{SF}$-II descend much faster than those of OFS and OFS$_P$ and eventually become stable with better results.

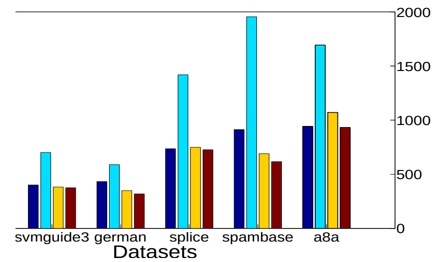


Figure 7. Comparison with respect to the average number of error predictions. We can observe that OL$_{SF}$-I and OL$_{SF}$-II performs better than OFS and OFS$_P$ by adding a flexible learning rate τ_t.

Table 6
Comparison with respect to the average number of error predictions ($B = 0.001$).

Algorithms	rcv1	URL
OL$_{SF}$-I	239582.0±1104.2	**599352.0±8888.1**
OL$_{SF}$I-all	**235280.8±1459.4**	607019.6±8051.6
OL$_{SF}$I-rand	482310.1±443.5	1520743.8±12546.3
OL$_{SF}$I-per	329572.6±1113.5	602546.8±9063.3

6.4 Experiment IV: Applications to real-world trapezoidal data streams

In this part, we evaluate the performance of the proposed algorithms on two real-world data streams. The data sets can be downloaded online [45].

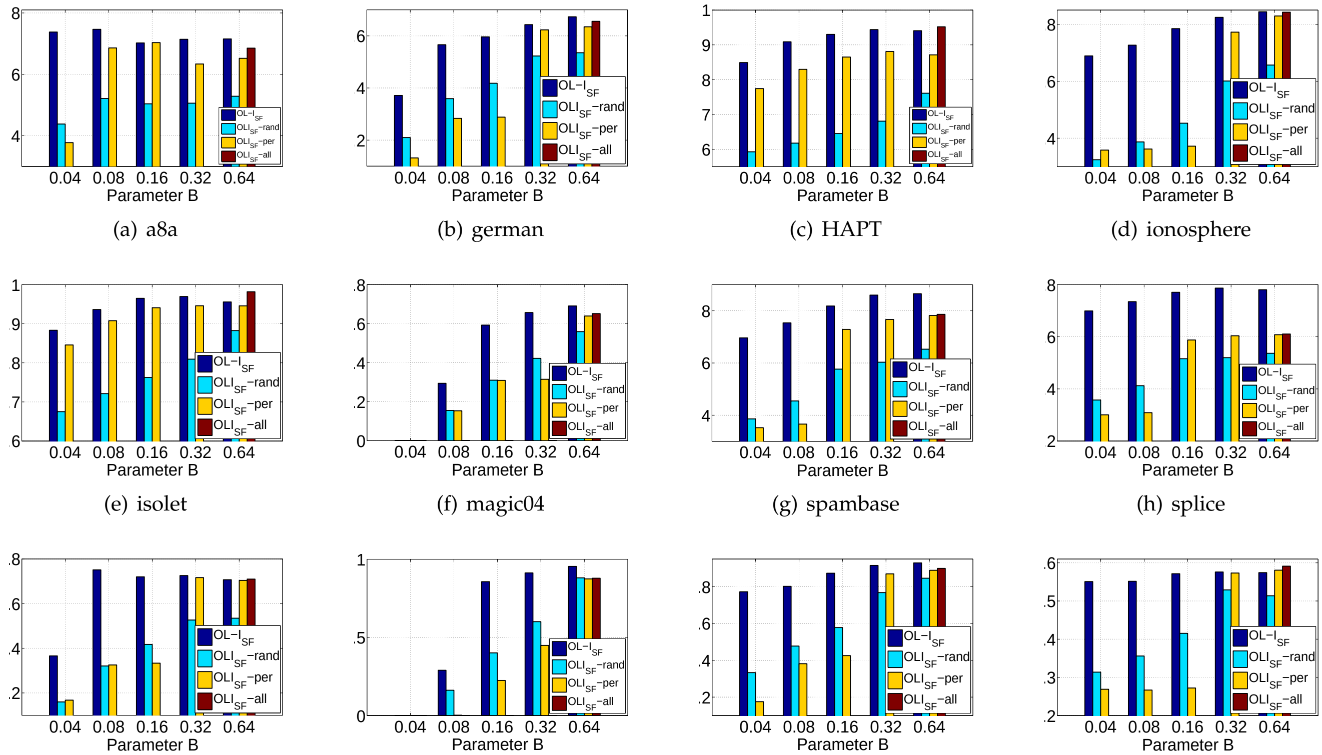
The task of the URL dataset [46] is to detect malicious URLs from Webpage streams using lexical and host-based features of URLs. In the task, URLs arrive continuously as streams, where each URL carries lexical and host-based features that we have never seen before. The purpose is to continuously learn a URL classifier that can identify malicious Webpages from normal ones. Thus, the learning problem can be formulated as online learning from trapezoidal data streams. The task of rcv1 text classification is to categorize the JMLR articles into different groups. Because new articles are published continuously with new research topics, the problem can be also defined as online learning from trapezoidal data streams.

Table 6 shows the experimental results of the average number of error predictions of the four algorithms. We set the parameter $B = 0.001$. The tradeoff parameter $C = 0.1$. From the results, we can observe that OLSF-I that uses only 0.1% features performs similarly to OLSF-all that uses all the features on rcv1 dataset. Fig. 9 shows the performance of the algorithms with respect to the number of training instances when $B = 0.01$, i.e., using 1% features to learn. We can observe that OL$_{SF}$-I, OL$_{SF}$-all, OL$_{SF}$-per converge fast when the number of training instances increases. Moreover, OL$_{SF}$-I performs better than the other three algorithms and converges to the lowest error rates.

6.5 Discussions

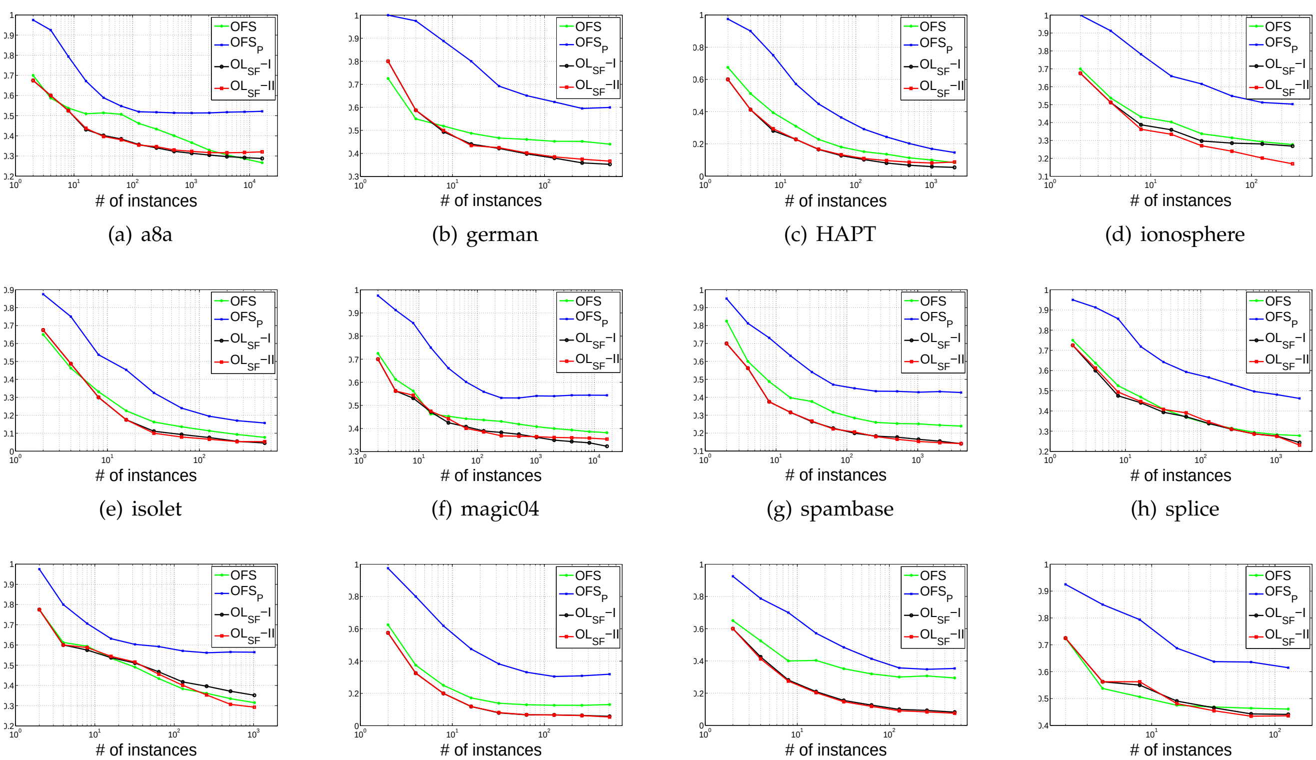
Multi-class classification. There are two methods *One vs Rest* and *One vs One* [47] that can extend the proposed algorithms to multi-class classification by converting the problem to be multiple binary classification problems [48]. For a c-class problem in *One vs One*, it often requires to build

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2563424, IEEE Transactions on Knowledge and Data Engineering



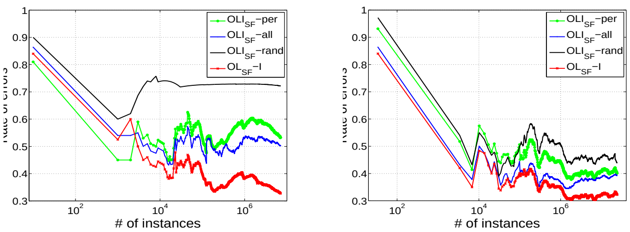
(a) a8a (b) german (c) HAPT (d) ionosphere
(e) isolet (f) magic04 (g) spambase (h) splice
(i) svmguide3 (j) wbc (k) wdbc (l) wpbc

Figure 6. Online classification accuracy with respect to the parameter B. We can observe that OL$_{SF}$-I performs the best especially when the feature space is sparse, i.e., B is very small.



(a) a8a (b) german (c) HAPT (d) ionosphere
(e) isolet (f) magic04 (g) spambase (h) splice
(i) svmguide3 (j) wbc (k) wdbc (l) wpbc

Figure 8. Comparison with respect to online prediction. We can observe that the curves of OL$_{SF}$-I and OL$_{SF}$-II descend much faster than those of OFS and OFS$_P$ and eventually become stable with lower error rates.



(a) rcv1 text classification (b) URL data set

Figure 9. Performance on real trapezoidal data streams($B = 0.01$).

$c(c-1)/2$ binary classifiers. From the model formulation perspective, we can directly extend the vector-based models to matrix-based models.

Semi-supervised classification. In many applications, labels are provided only for a few data points [49] [50]. Here, pseudo-labels can be used to enlarge a labeled training set. Specifically, we can use the classifiers trained from labeled examples to predict class labels (pseudo-labels) of unlabeled examples. Then, a semi-supervised learner can be built from both labeled and pseudo-labeled examples.

7 CONCLUSIONS

In this paper we studied a new problem of online learning from trapezoidal data streams where both data volume and feature space increase by time. We proposed a new Online Learning with Streaming Features algorithm (OL$_{SF}$) and its two variants OL$_{SF}$-I and OL$_{SF}$-II as the solution. Theoretical and empirical analysis have shown the performance of the proposed algorithms.

ACKNOWLEDGEMENTS

This work was supported by Australia ARC Discovery Project (DP140102206), 973 Program of China (2013CB329604), Australia ARC Linkage project (LP150100671), PCSIRT of the Ministry of Education of China (IRT13059), NVIDIA Foundation Compute the Cure, and NSFC of China (61370025 and 61229301).

REFERENCES

[1] J. Langford, L. Li, and T. Zhang, "Sparse online learning via truncated gradient," *J. Machine Learning Research*, vol. 10, pp. 777–801, 2009.

[2] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithms," *J. Machine Learning Research*, vol. 7, pp. 6551–6585, 2006.

[3] J. Wang, P. Zhao, S. C. Hoi, and J. Wan, "Online feature selection and its applications," *IEEE Trans. Knowl. and Data Eng.*, vol. 26(3), pp. 698–710, 2014.

[4] X. Wu, K. Yu, W. Ding, H. Wang, and X. Zhu, "Online feature selection with streaming features," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35(5), pp. 1178–1192, 2013.

[5] X. Wu, K. Yu, H. Wang, and W. Ding, "Online streaming feature selection," *Proc. 27th Int'l Conf. Machine Learning*, pp. 1159–1166, 2010.

[6] K. Zhai and J. Boyd-Graber, "Online latent dirichlet allocation with infinite vocabulary," *Proc. 30th Int'l Conf. Machine Learning*, vol. 28, 2013.

[7] P. Zhang, C. Zhou, P. Wang, B. J. Gao, X. Zhu, and L. Guo, "E-tree: An efficient indexing structure for ensemble models on data streams," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 27, no. 2, pp. 461–474, 2015.

[8] P. Zhang, B. J. Gao, P. Liu, Y. Shi, and L. Guo, "A framework for application-driven classification of data streams," *Neurocomputing*, vol. 92, pp. 170–182, 2012.

[9] J. Kivinen and M. K. Warmuth, "Exponentiated gradient versus gradient descent for linear predictors," *Information and Computation*, vol. 132(1), pp. 1–63, 1997.

[10] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," *Psychological Rev.*, vol. 65, pp. 386–407, 1958.

[11] K. Crammer, M. Dredze, and A. Kulesza, "Multi-class confidence weighted algorithms," *In Empirical Methods in Natural Language Proc.*, pp. 496–504, 2009.

[12] S. C. Hoi, J. Wang, P. Zhao, and J. Wan, "Libol: A library for online learning algorithms," *Nanyang Technological Univ.*, 2012.

[13] Z. Peng, Z. Chuan, W. Peng, G. B. J., Z. Xingquan, and G. Li, "E-tree: An efficient indexing structure for ensemble models on data streams," *IEEE Transactions on Knowledge Data Engineering*, vol. 27, no. 2, pp. 461–474, 2015.

[14] Y. Freund and R. Schapire, "Large margin classification using the perceptron algorithm," *J. Machine Learning Research*, vol. 37(3), pp. 277–296, 1999.

[15] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," *Proc. 20th Int'l Conf. Machine Learning*, pp. 928–936, 2003.

[16] C. Gentile, "A new approximate maximal margin classification algorithm," *J. Machine Learning Research*, vol. 2, pp. 213–242, 2001.

[17] N. Cesa-Bianchi, A. Conconi, and C. Gentile, "A second-order perceptron algorithm." *SIAM J. Comput.*, vol. 34(3), pp. 640–668, 2005.

[18] K. Crammer and D. D. Lee, "Learning via gaussian herding," *in Proc. Adv. Neural Inf. Process. Syst.*, pp. 451–459, 2010.

[19] L. Yang, R. Jin, and J. Ye., "Online learning by ellipsoid method," *Proc. 26th Int'l Conf. Machine Learning*, p. 145, 2009.

[20] K. Crammer, A. Kulesza, and M. Dredze, "Adaptive regularization of weight vectors," *in Proc. Adv. Neural Inf. Process. Syst.*, pp. 414–422, 2009.

[21] F. Orabona and K. Crammer, "New adaptive algorithms for online classification," *in Proc. Adv. Neural Inf. Process. Syst.*, pp. 1840–1848, 2010.

[22] J. Weston, A. Elisseff, B. Schoelkopf, and M. Tipping, "Use of the zero norm with linear models and kernel methods," *J. Machine Learning Research*, vol. 3, pp. 1439–1461, 2003.

[23] L. Song, A. Smola, A. Fretton, K. Borgwardt, and J. Bedo, "Supervised feature selection via dependence estimation," *Proc. 24th Int'l Conf. Machine Learning*, 2007.

[24] J. Dy and C. Brodley, "Feature selection for unsupervised learning," *J. Machine Learning Research*, vol. 5, pp. 845–889, 2004.

[25] P. Mitra, C. A. Murthy, and S. Pal, "Unsupervised feature selection using feature similarity," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, pp. 301–312, 2002.

[26] Z. Xu, R. Jin, J. Ye, M. Lyu, and I. King, "Discriminative semi-supervised feature selection via manifold regularization," *Proc. 20th Int'l Joint Conf. Artificial Intelligence*, 2009.

[27] Z. Zhao and H. Liu, "Semi-supervised feature selection via spectral analysis," *Proc. SIAM Int'L Conf. Data Mining*, 2007.

[28] J. Tang, S. Alelyani, and H. Liu, "Feature selection for classification: A review," *Data Classification: Algorithms and Applicaions Boca Raton, FL USA: CRC Press*, 2014.

[29] M. R. Sikonja and I. Kononenko, "Theoretical and empirical analysis of relief and relieff," *J. Machine Learning Research*, vol. 53, pp. 23–69, 2003.

[30] R. Duda, P. Hart, and D. Stork, "Pattern classification," *Wiley-Interscience*, 2012.

[31] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information: criteria of max-dependency, max-relevance and min-redundancy," *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 1226–1238, 2005.

[32] Z. Xu, R. Jin, J. Ye, M. Lyu, and I. King, "Non-monotonic feature selection," *Proc. 26th Int'l Conf. Machine Learning*, p. 144, 2009.

[33] J. Bi, K. Bennett, M. Embrechts, C. Breneman, and M. Song, "Dimensionality reduction via sparse support vector machines," *J. Machine Learning Research*, vol. 3, pp. 1229–1243, 2003.

[34] N. V. A.B. Chan and G. Lanckriet, "Direct convex relaxations of sparse svm," *Proc. 24th Int'l Conf. Machine Learning*, pp. 145–153, 2007.

[35] Y. Zhou, R. Jin, and S. Hoi, "Exclusive lasso for multi-task feature seleciton," *J. Machine Learning Research*, vol. 9, pp. 988–995, 2010.

[36] J. R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[37] N. C. Talbot, G. C. Cawley, and M. Girolami, "Sparse multimonial logistic regression via bayesian l1 regularisation," *in Proc. Adv. Neural Inf. Process. Syst.*, 2006.

[38] J. Dy and C. Brodley, "Feature subset selection and order identification for unsupervised learning," *Proc. 17th Int'l Conf. Machine Learning*, pp. 247–254, 2000.

[39] X. He, D. Cai, and P. Niyogi, "Laplacian score for feature selection," *in Proc. Adv. Neural Inf. Process. Syst.*, 2005.

[40] Z. Zhao and H. Liu, "Spectral feature selection for supervised and unsupervised learning," *Proc. 24th Int'l Conf. Machine Learning*, pp. 1151–1157, 2007.

[41] Y. Yang, H. Shen, Z. Ma, Z. Huang, and X. Zhou, "$l_{2,1}$-norm regularized discriminative feature selection for unsupervised learning," *Proc. 22th Int'l Joint Conf. Artificial Intelligence*, pp. 1589–1594, 2011.

[42] J. Duchi and Y. Singer, "Efficient online and batch learning using forward backward splitting," *J. Machine Learning Research*, vol. 10, pp. 2899–2934, 2009.

[43] K. Glocer, D. Eads, and J. Theiler, "Online feature selection for pixel classification," *Proc. 22th Int'l Conf. Machine Learning*, pp. 249–256, 2005.

[44] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[45] C.-C. Chang and C.-J. Lin, "Libsvm : a library for support vector machines," *ACM Trans. on Intell. Syst. and Tech.*, vol. 2(27), pp. 1–27, 2011.

[46] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Identifying suspicious urls: An application of large-scale online learning," *Proc. 26th Int'l Conf. Machine Learning*, 2009.

[47] J. A. Sez, M. Galar, J. Luengo, and F. Herrera, "Analyzing the presence of noise in multi-class problems: alleviating its influence with the one-vs-one decomposition," *Knowledge and Information Systems*, vol. 38, no. 1, pp. 179–206, 2014.

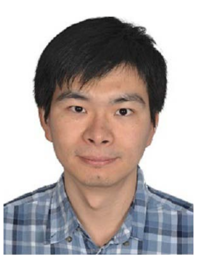
[48] V. Boln-Canedo, N. Snchez-Maroo, and A. Alonso-Betanzos, "Feature selection and classification in multiple class datasets: An application to kdd cup 99 dataset," *Expert Systems with Applications*, vol. 38, no. 5, pp. 5947–5957, 2011.

[49] I. Triguero, S. Garca, and F. Herrera, "Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study," *Knowledge and Information Systems*, vol. 42, no. 2, pp. 245–284, 2015.

[50] G. Yu, G. Zhang, Z. Zhang, Z. Yu, and L. Deng, "Semi-supervised classification based on subspace sparse representation," *Knowledge and Information Systems*, vol. 43, no. 1, pp. 81–101, 2015.

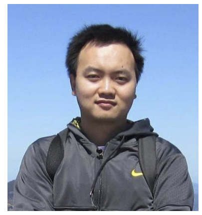
Qin Zhang is currently a PhD student in the Centre for Quantum Computation and Intelligent Systems(QCIS), University of Technology, Sydney (UTS) Australia. She received her Master degree in July 2014 from the University of Chinese Academy of Sciences, China. Her main research interests include data mining and online learning.

She served as a reviewer (sub-reviewer) in KDD-15, ICDM-15, IJCAI-15, AAAI-15 and NIPS-15.



Peng Zhang received his PhD degree in July 2009 from University of the Chinese Academy of Sciences. Since then, Dr. Zhang has worked with one national engineering laboratory at the Chinese Academy of Sciences and two universities in the USA. In January 2014, he joined the QCIS research center, University of Technology Sydney (UTS), as a Lecturer.

Dr. Zhang is Associate Editor of two Springer journals, Journal of Big Data, and Annals of Data Science. To date, he has published more than 100 research papers in data mining, including IEEE TKDE, KDD, ICDM, SDM, IJCAI, AAAI, CIKM, WWW and PAKDD. He serves as a PC member (reviewer) in IEEE TKDE, ACM TKDD, KDD, ICDM, IJCAI, AAAI, NIPS, etc. He won the Best Paper Award at ICCS-14 (ERA Rank A) held in Queensland, Australia.



Guodong Long received his Ph.D. degree from University of Technology, Sydney (UTS), Australia, in 2014. He is a researcher associate at the Centre for Quantum Computation and Intelligent Systems (QCIS), University of Technology, Sydney (UTS), Australia. His research focuses on data mining and social network analysis.

Wei Ding received her Ph.D. degree in Computer Science from the University of Houston, USA in 2008. She is an Associate Professor of Computer Science in the University of Massachusetts Boston. Her research interests include data mining, machine learning, artificial intelligence, computational semantics, and with applications to astronomy, geosciences, and environmental sciences. She has published more than 100 referred research papers, 1 book, and has 2 patents. She is an Associate Editor of Knowledge and Information Systems (KAIS) and an editorial board member of the Journal of Information System Education (JISE), the Journal of Big Data, and the Social Network Analysis and Mining Journal. She is an IEEE senior member and an ACM senior member.

Chengqi Zhang received the Ph.D. degree from the University of Queensland, Brisbane, Australia, in 1991 and the D.Sc. degree (higher doctorate) from Deakin University, Geelong, Australia, in 2002. Since December 2001, he has been a Professor of Information Technology with the University of Technology, Sydney (UTS), Sydney, Australia, where he has been the Director of the UTS Priority Investment Research Centre for Quantum Computation and Intelligent Systems since April 2008. Since November 2005, he has been the Chairman of the Australian Computer Society National Committee for Artificial Intelligence. He has published more than 200 research papers, including several in first-class international journals, such as the *Artificial Intelligence*, *IEEE*, and *ACM Transactions*. He has published six monographs and edited 16 books, and has attracted 11 Australian Research Council grants. His research interests mainly focus on data mining and its applications. He has been serving as an Associate Editor for three international journals, including IEEE *Transactions on Knowledge and Data Engineering* (2005-2008); and he served as General Chair, PC Chair, or Organising Chair for five international Conferences including ICDM 2010 and WI/IAT 2008. He is also General Co-Chair of KDD 2015 in Sydney and the Local Arrangements Chair of IJCAI-2017 in Melbourne. He is a Fellow of the Australian Computer Society and a Senior Member of the IEEE.

Xindong Wu received his PhD degree in artificial intelligence from the University of Edinburgh, United Kingdom. He is a Yangtze River scholar in the School of Computer Science and Information Engineering, Hefei University of Technology, China, and a professor of Computer Science at the University of Vermont. His research interests include data mining and knowledge-based systems. Prof. Wu is the Steering Committee chair of the IEEE International Conference on Data Mining, and the editor-in-chair of Knowledge and Information Systems. He is a fellow of the IEEE and AAAS.