Towards Mining Trapezoidal Data Streams

Qin Zhang*, Peng Zhang*, Guodong Long *, Wei Ding†, Chengqi Zhang*, and Xindong Wu‡
* QCIS, University of Technology, Sydney NSW 2007, Australia
Email: Qin.Zhang@student.uts.edu.au, Peng.Zhang@uts.edu.au,
Guodong.Long@uts.edu.au, Chengqi.Zhang@uts.edu.au
†College of Science and Mathematics, University of Massachusetts Boston, MA 02125-3393
Email: wei.ding@umb.edu
‡Dept. of Computer Science, University of Vermont Burlington, VT05405.
Email: xwu@cs.uvm.edu

***Abstract*—In this paper we study a new problem of learning from doubly-streaming data where both data volume and feature space increase by time. We refer to the problem as mining trapezoidal data streams. The problem is challenging because both data volume and feature space are increasing. Existing online learning, online feature selection and streaming feature selection algorithms are inapplicable. To this end, we propose a new Sparse Trapezoidal Streaming Data mining algorithm (STSD) and its two variants which combine online learning and online feature selection algorithms to enable learning trapezoidal data streams with infinite training instances and features. Specifically, when new training instances carrying new features arrive, the classifier updates the 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 projected truncation techniques. In this way, STSD can auto-learn from trapezoidal data streams. Experiments on UCI data sets show the performance of the proposed algorithms.**

I. Introduction

Recent years have witnessed an increasing number of applications on doubly-streaming data where both data volume and data dimension increase with time. For example, in text clustering, both the number of documents and the text vocabulary may increase with time, and a recent work proposed an *infinite vocabulary topic model* [21] which allows the addition, invention and increased prominence of new terms to be captured. In graph node classification, both the number of graph nodes and the node features (e.g., the ego-network structure of a node) may change by 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 obviously much more difficult than existing data stream mining and online learning problems. The main challenge of learning from trapezoidal data streams is how to design highly dynamic classifiers that can learn from increasing training data with expanding feature space. Obviously, existing online learning [7], online feature selection [15] and streaming feature selection algorithms [17] cannot be used to handle our problem directly.

Online learning algorithms [8] are proposed to solve the problem where training instances arrive one by one but the feature space is static and known a prior before learning. The algorithms update classifiers using incoming instances and allow the sum of training loss gradually to be bounded [8]. To date, online learning algorithms, such as the Perceptron algorithm [10], the Passive Aggressive algorithm [2] and the Confidence-Weighted algorithm [3], are commonly used in data-driven optimization problems. However, these algorithms cannot be directly used to handle dynamic feature space.

Online feature selection algorithms [8], [15] were proposed 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 [15]. These algorithms use sparse strategies, such as feature truncation, to select representative features. Sparse online learning via truncated gradient [8] and the OFS algorithm [15] are typical algorithms. However, these algorithms cannot solve the trapezoidal data stream mining problem because they assume feature space is fixed.

Online streaming feature selection algorithms [17] 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 model. It, in some ways, can be seen as the dual problem of online learning [17]. Typical algorithms include the online streaming feature selection (OSFS) algorithm [16] and the fast-OSFS [17] algorithms. However, these algorithms consider only a static training set where the number of training instances is given in advance before learning. Therefore, they cannot be used in our problem.

To sum up, existing online learning, online feature selection and streaming feature selection algorithms cannot be used to learn from trapezoidal data streams. In this paper, we propose a new Sparse Trapezoidal Streaming Data (STSD) algorithm and its two variants STSD-I and STSD-II for mining trapezoidal data streams. STSD and its variants combine online learning and online feature selection to continuously learn from trapezoidal data streams. Specifically, when new training instances carrying new features arrive, the 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 the feature projected truncation techniques. We conduct empirical tests on UCI data sets to show their performance.

The contributions of this paper are summarized as follows:

1) We study a new problem of learning from trapezoidal data streams where training data doubly change in both data volume and feature space;
2) We propose a new algorithm STSD and its two variants. They combine the merits of online learning and online feature selection to learn from trapezoidal data streams;
3) We empirically validate the performance of the algorithms on UCI data sets.

The remainder of this paper is organized as follows: Section 2 surveys related work. Section 3 introduces the problem in detail. Section 4 discusses the proposed STSD algorithm and its variants. Section 5 conducts experiments and Section 6 concludes the paper.

II. Related Work

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

Online learning represents an important family of efficient and scalable data mining and machine learning algorithms for massive data analysis. In general, online learning algorithms can be grouped into two categories, first-order and second-order learning algorithms [6].

The first-order online learning algorithms exploit first order information during update. The Perceptron algorithm [10] and Online Gradient Descent algorithm (OGD) [24] 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 [15], such as the PA algorithm [2], ALMA algorithm [4], and ROMMA algorithm [4].

The second-order online learning algorithms, which can better explore the underlying structure between features [6] 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 [6]. The CW [3], and IELLIP [19], algorithms 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. According to training set is labeled or not, feature selection can be categorized into supervised [12], unsupervised [9] and semi-supervised feature selection [22] algorithms.

Supervised feature selection can be categorized into filter models, wrapper models and embedded models [14]. 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. Relief [11], Fisher scoreand Information Gain based methods [18] are the representative algorithms. Wrapper models use the predictive accuracy of a predetermined learning algorithm to determine the quality of selected features. Embedded methods aim to integrate feature selection into the model training process. It achieves model fitting and feature selection simultaneously [13]. Embedded methods are usually faster than the former two 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 [14]. Laplacian Score [5], spectral feature selection [23], and the recently proposed $l_{2,1}$-norm regularized discriminative feature selection [20] 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 makes use of both labeled and unlabeled data to estimate feature relevance [22].

Online feature selection [15] and *sparse online learning* [8] 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 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$_P$ [15] are the representative algorithms proposed recently.

Online streaming feature selection algorithms have been studied recently [17] 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 [16]. 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 mining trapezoidal data streams.

III. Problem Setting

We consider the binary classification problem on trapezoidal data streams where both data volume and feature space increase simultaneously. 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 $d_{t-1} \leq d_t$ and $y_t \in \{-1, +1\}$ for all t. On each round, the classifier uses information on a 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. 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 [2].

We restrict the discussion to a linear classifier which is based on a vector of weights w. 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 weight vector 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 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 introduce feature selection into our mining algorithm. Formally, in each trial t, instead of using all features for classification, we require the classifier $w_t \in \mathbb{R}^{d_{w_t}}$ to have at most a proportion of B nonzero elements, i.e.,

$$\|w_t\|_0 \leq B \cdot d_{w_t}, \tag{1}$$

where $B \in [0, 1]$ is a predefined parameter that controls the proportion of features used in the algorithm.

We refer to this problem as the problem of learning from trapezoidal streaming data. The ultimate goal is to design an effective algorithm for trapezoidal streaming data which is highly accurate.

IV. Sparse Trapezoidal Streaming Data Algorithms

In this section we present the proposed Sparse Trapezoidal Streaming Data learning algorithm (STSD) and its two variants. There are two challenges with 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 sparsity. As the dimension increases with 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 project step is introduced before the truncation.

Algorithm 1. The STSD algorithm and its two variants STSD-I and STSD-II

```
Input:
  • $C > 0$: tradeoff parameter
  • $\lambda > 0$: 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:
      $\check{w}_{t+1} = \min\{1, \frac{\lambda}{\|\bar{w}_{t+1}\|_1}\}\bar{w}_{t+1}$
    • truncate $\check{w}_{t+1}$ to $w_{t+1}$:
      $w_{t+1} = Truncate(\check{w}_{t+1}, B)$
          (See Algorithm 3)
end for
```

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

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

The pseudo-codes for the STSD algorithm and its two variants are given in Algorithms 1, 2 and 3 respectively (STSD-I and STSD-II are different to STSD in parameter τ_t during updates). The vector w_1 is initialized to zero 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 each algorithm receives. Then, online learning is divided into the update step and the sparsity step.

The update strategy

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

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

The three algorithms are different in their update strategy. We first focus on the update strategy of the basic algorithm. At round t, when 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}, and $\hat{w} = \Pi_{\neg w_t} w_{t+1} \in \mathbb{R}^{d_t - d_{t_1}}$ denotes new features that are in w_{t+1} but not in w_t,

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

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

$$l_t = l(w, (x_t, y_t)) = \max\{0, 1 - y_t(\tilde{w} \cdot \tilde{x}_t) - y_t(\hat{w} \cdot \hat{x}_t)\}. \tag{3}$$

Note that $\tilde{x}_t$ and $\hat{x}_t$ are similar to $\tilde{w}$ 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. (2). 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. (2) has a simple closed form,

$$w_{t+1} = [w_t + \tau_t y_t \tilde{x}_t, \tau_t y_t \hat{x}_t] \quad where \ \tau_t = l_t / \|x_t\|^2 \tag{4}$$

We now discuss the derivation of the update strategy.

- In a case where 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 a case where $d_t > d_{t-1}$ and $l_t = 0$, then the optimal solution is $\tilde{w}_{t+1} = w_t$ and $\hat{w}_{t+1} = (0, \cdots, 0)$.
- In a case where $d_t > d_{t-1}$ and $l_t > 0$, then we solve Eq. (2) to obtain the solution.

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

$$\begin{aligned} L(w,\tau) = \frac{1}{2}\|\tilde{w} - w_t\|^2 + \frac{1}{2}\|\hat{w}\|^2 \\ + \tau(1 - y_t(\tilde{w} \cdot \tilde{x}_t) - y_t(\hat{w} \cdot \hat{x})) \\ \tilde{w} = w_t + \tau y_t \tilde{x}_t; \quad \hat{w} = \tau y_t \hat{x}_t \end{aligned} \tag{5}$$

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

$$\begin{aligned} L(\tau) &= -\frac{1}{2}\tau^2\|\tilde{x}_t\|^2 - \frac{1}{2}\tau^2\|\hat{x}\|^2 + \tau - \tau y_t(w_t \cdot \tilde{x}) \\ \tau_t &= \frac{1 - y_t(w_t \cdot \tilde{x}_t)}{\|\tilde{x}\|^2 + \|\hat{x}_t\|^2} = \frac{l_t}{\|x_t\|^2} \end{aligned} \tag{6}$$

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. (2), we can see that the update strategy of the STSD 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 STSD algorithm which use the soft-margin technique by introducing a slack variable ξ into the optimization problem. The first one is abbreviated as STSD-I. Its objective function scales linearly with ξ, namely,

$$w_{t+1} = \underset{\substack{w = [\tilde{w}, \hat{w}]: \\ l_t \leq \xi; \xi \geq 0}}{\operatorname{argmin}} \quad \frac{1}{2}\|\tilde{w} - w_t\|^2 + \frac{1}{2}\|\hat{w}\|^2 + C\xi \tag{7}$$

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

$$w_{t+1} = \underset{\substack{w = [\tilde{w}, \hat{w}]: \\ l_t \leq \xi}}{\operatorname{argmin}} \quad \frac{1}{2}\|\tilde{w} - w_t\|^2 + \frac{1}{2}\|\hat{w}\|^2 + C\xi^2 \tag{8}$$

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 STSD-I and STSD-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

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

The update strategies of STSD-I and STSD-II are similar to the STSD 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. As the dimension changes over time, but if only a fixed number of features are used in learning, the results are not always satisfactory.

In our study, we introduce a parameter to control the proportion of the features. 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 $d_{t-1} \leq d_t$. After the update operation, a projection and a truncation are introduced to prune redundant features based on the parameter B. Namely, we require the learner only retain at most a portion of B nonzero elements of $w_t \in \mathbb{R}^{d_{w_t}}$, i.e. $\|w_t\|_0 \leq B \cdot d_{w_t}$. Specifically, if the resulting classifier w_t has more than a portion of B nonzero elements, we will simply keep the portion of B elements in w_t with the largest absolute weights, as demonstrated in Algorithm 3. In this way, at most a portion 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. 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 [15]. Specifically, the projection technique is,

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

where λ is the a regularization parameter which is always positive.

V. Experiments

In this section, we describe our experiments to evaluate the performance of the proposed STSD algorithm and its two variants. We first evaluate the predictive performance of the three proposed algorithms, and analyse the relationship between classification accuracy, feature fraction parameter B, and the tradeoff parameter C on several benchmark data sets. Then, we compare our approach with three benchmark algorithms. Also, we test the performance with an application on real-world website classification. The source codes are available online https://github.com/BlindReview/onlineLearning

A. Experiment I: Performance tests of the STSD algorithm and its variants

We present empirical results of the three algorithms on several benchmark data sets from the UCI repository.

1) Testbed on UCI Data and Experimental Setup: We test the performance of the proposed algorithms on a number of publicly available benchmarking data sets.All the data sets can be downloaded from the UCI machine learning repository. Table I provides details of the data sets.

To compare fairly, all algorithms use the same experimental settings. We set the parameter $B = 0.5$, i.e., the portion of selected features is 50% . We set the tradeoff parameter C to be 0.1 and the radius parameter λ to be 30. For the special scenario of trapezoidal data streams, we assume that the first 10% of instances can access the first 10% of features, and the next 10% of instances can access the first 20% of features, and the same increase increment with the rest. After this, all experiments are conducted 20 times, each with a random permutation of a data set. All the experiment results are reported by an average over 20 runs.

Table I. THE UCI DATA SETS USED IN THE EXPERIMENTS

Dataset	$\sharp$ Samples	$\sharp$ Dimensions
magic04	19020	10
german	1000	24
svmguide3	1234	21
splice	3175	60
spambase	4601	57
a8a	32561	123

2) Evaluation of Predictive Performance: Table II summarizes the online predictive performance of the compared algorithms with a fixed fraction of selected features (50 percent of all features) on six data sets. Several observations can be drawn from the results. First, we found that among the three algorithms, STSD is the most rigid update strategy that has the highest error rate in five data sets except “svmguide3”. This shows that the gentle update strategy with a slack variable achieves better classifications by avoiding noise in the data sets. The reason for the failure of STSD-I in “svmguide3” is the unsuitable setting of the parameter C (We just set a constant number to the parameter C for all three algorithms and all six data sets without choosing the best one for each case). In fact, the difference is very small. Second, we found that in the six data sets, STSD-I and STSD-II achieve the best performance in three data sets. This shows that the two algorithms, using a gentle update strategy have similar performance.

Table II. EVALUATION OF THE AVERAGE NUMBER OF ERRORS BY STSD AND ITS VARIANTS ON THE SIX DATA SETS

Algorithm	magic04	svmguide3	german	splice	spambase	a8a
STSD	8051.3 ± 49.0	396.7 ± 15.8	415.9 ± 15.6	1314.6 ± 30.3	1132.1 ± 29.7	12673.5 ± 75.9
STSD-I	**6732.3±73.3**	359.1 ± 42.9	**366.9±8.8**	1243.7 ± 13.6	**1004.5±25.6**	**11204.7±713.1**
STSD-II	6924.5 ± 39.6	**357.5±26.9**	**366.9±12.8**	**1238.8±16.8**	1013.2±26.1	11317.2±233.1

Fig. 1 shows the performance of the three algorithms under different C values. From the results, we can see that there is always a parameter C to force STSD-I and STSD-II to have less errors than STSD. The larger C is, the closer the STSD-I algorithm to the STSD algorithm, because the parameter τ_t in STSD-I is the smaller one in C or the parameter τ_t in the STSD algorithm. When C is very large, STSD-I is reduced to STSD. Fig 1 demonstrates this finding.

B. Experiment II: Comparisons with other algorithms

We compare our algorithms with the three benchmark algorithms. Due to the good performance of STSD-I, we use STSD-I as the representative of our three algorithms, and we compare the performance of the STSD-I algorithm with the three benchmark algorithms.

The first benchmark is **STSDI-all** which is the same as STSD-I except that not only a portion of features is selected but all features are used. The second one is **STSDI-rand** which has the same update strategy but with randomly selected features. The third is **STSDI-per** which has the same sparsity strategy in selecting features but uses the Perceptron update strategy, and the Perceptron update strategy is $w_{t+1} = [w_t + y_t x_t, y_t x_t]$ [10]. We again use the UCI data sets listed in Table I. The parameter settings and experimental settings are the same as in Experiments I.

Table III gives the average number of test errors by the four algorithms on the six data sets. First, we can observe that the more features, the number of errors by STSD-I is lower on average. We also observe that when $B = 0.8$, the performance of STSD-I is better than STSDI-all which can access all the features in most data sets. Although on some data sets, STSDI-all has a better performance than the STSD-I algorithm, the difference is not significant. When we use a small portion of features, we can see that STSD-I has a better performance. Second, the STSDI-rand algorithm randomly chooses a fixed portion of features and has the worst performance on all data sets. This further indicates that the sparsity strategy we introduced can improve the performance of the classifier significantly. Third, the STSDI-per algorithm which uses the Perceptron update strategy has higher error rates than the STSD-I algorithm on most data sets, demonstrating that the update strategy of our algorithms is better.

VI. CONCLUSIONS

This paper investigated a new problem of mining trapezoidal data streams where both data volume and feature space increase by time. We presented new sparse trapezoidal streaming data mining algorithms as the solution. We also examined the empirical performance on UCI data sets. The encouraging results have shown that the proposed algorithms are effective for mining trapezoidal streaming data, and more efficient and scalable than batch-based algorithms.

Future work includes extending the proposed algorithms to real-world applications such as big dynamic network mining, and studying the multiclass classification and regression problems on trapezoidal data streams.

REFERENCES

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

[2] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *Journal of Machine Learning*, 7:6551–585, 2006.

[3] K. Crammer, M. Dredze, and A. Kulesza. Multi-class confidence weighted algorithms. *In EMNLP*, pages 496–504, 2009.

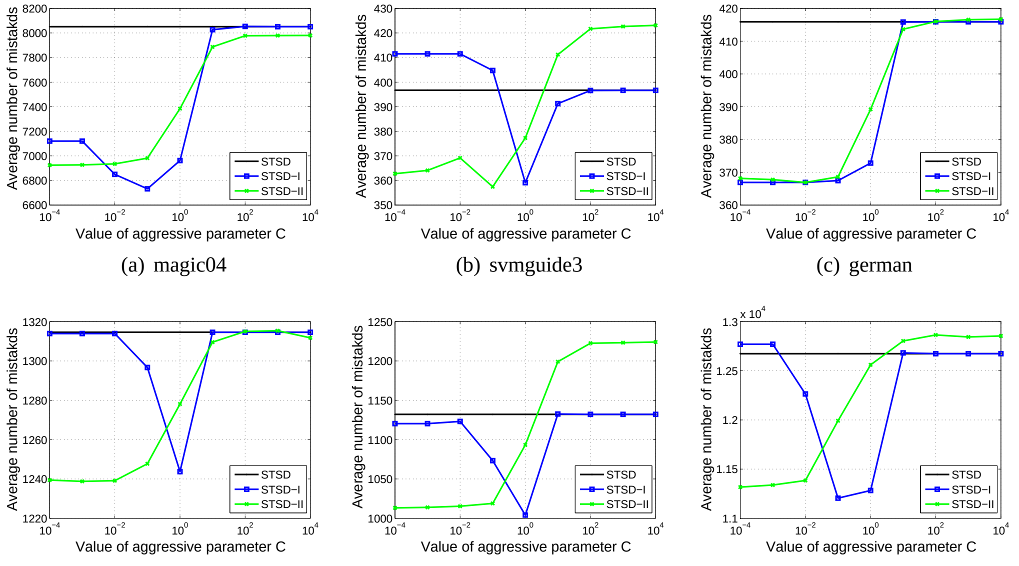
[4] C. Gentile. A new approximate maximal margin classifiction algorithm. *Journal of Machine Learning Research*, 2:213–242, 2001.

[5] X. He, D. Cai, and P. Niyogi. Laplacian score for feature selection. *In NIPS ’05*, 2005.

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

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

[8] J. Langford, L. Li, and T. Zhang. Sparse online learning via truncated gradient. *Journal of Machine Learning Research*, 10:777–801, 2009.



(a) magic04 (b) svmguide3 (c) german

(d) splice (e) spambase (f) a8a

Figure 1. The average number of errors w.r.t. parameter C.

Table III. AVERAGE TEST ERRORS MADE BY THE ALGORITHMS ON THE SIX DATA SETS W.R.T. SELECTED FEATURES B

Algorithm		magic04	svmguide3	german	splice	spambase	a8a
STSD-I	B=0.2	13367.3±406.9	814.1±36.3	702.7±20.1	1282.3±38.1	1176.1±73.7	12279.1±3073.7
STSDI-all		**6634.3±35.2**	**360.9±7.2**	**344.1±7.1**	**1236.1±29.5**	**983.6±21.7**	**10242.6±109.8**
STSDI-rand		14101.8±53.3	919.3±11.3	739.3±11.2	1559.1±22.0	1930.7±38.1	15525.0±79.0
STSDI-per		13143.5±52.7	828.4±40.1	703.7±20.5	1300.6±41.5	1215.8±82.5	10047.7±1520.2
STSD-I	B=0.5	6732.3±73.3	**359.1±42.9**	366.9±8.8	1243.7±27.0	1004.1±25.6	11204.7±713.1
STSDI-all		**6634.3±35.2**	360.9±7.2	**344.1±7.1**	**1236.1±29.5**	**983.6±21.7**	**10242.6±109.8**
STSDI-rand		8014.3±45.9	563.5±20.9	464.3±16.2	1519.9±35.6	1699.9±25.2	15766.6±93.8
STSDI-per		6921.1±46.4	362.6±52.3	368.7±16.4	1239.2±26.7	1014.1±28.2	11303.6±308.0
STSD-I	B=0.8	**6633.7±37.2**	**350.3±7.2**	**336.4±8.0**	**1235.2±27.1**	**980.8±20.6**	10814.6±160.6
STSDI-all		6634.3±35.2	360.9±7.2	344.1±7.1	1236.1±29.5	983.6±21.7	**10242.6±109.8**
STSDI-rand		7785.2±34.2	490.8±19.8	402.8±11.9	1485.2±23.7	1490.7±19.2	14665.6±68.9
STSDI-per		6828.1±44.4	359.6±7.2	354.2±7.7	1238.5±26.6	991.0±21.0	11319.9±91.1

[9] P. Mitra, C. A. Murthy, and S. Pal. Unsupervised feature selection using feature similarity. *In PAMI*, 24:301–312, 2002.

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

[11] M. R. Sikonja and I. Kononenko. Theoretical and empirical analysis of relief and relieff. *Machine Learning*, 53:23–69, 2003.

[12] L. Song, A. Smola, A. Fretton, K. Borgwardt, and J. Bedo. Supervised feature selection via dependence estimation. *In ICML*, 2007.

[13] N. C. Talbot, G. C. Cawley, and M. Girolami. Sparse multimonial logistic regression via bayesian l1 regularisation. *In Neural Information Processing Systerms*, 2006.

[14] 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.

[15] J. Wang, P. Zhao, S. C. Hoi, and J. Wan. Online feature selection and its applications. *In TKDE*, 26(3):698–710, 2014.

[16] X. Wu, K. Yu, W. Ding, H. Wang, and X. Zhu. Online feature selection with streaming features. *In TPAMI*, 35(5):1178–1192, 2013.

[17] X. Wu, K. Yu, H. Wang, and W. Ding. Online streaming feature selection. *In ICLM '10*, pages 1159–1166, 2010.

[18] Z. Xu, R. Jin, J. Ye, M. Lyu, and I. King. Non-monotonic feature selection. *In ICML '09*, page 144, 2009.

[19] L. Yang, R. Jin, and J. Ye. Online learning by ellipsoid method. *In ICML*, page 145, 2009.

[20] Y. Yang, H. Shen, Z. Ma, Z. Huang, and X. Zhou. $l_{2,1}$-norm regularized discriminative feature selection for unsupervised learning. *In IJCAI '11*, pages 1589–1594, 2011.

[21] K. Zhai and J. Boyd-Graber. Online latent dirichlet allocation with infinite vocabulary. *In the 30th ICML*, 28, 2013.

[22] Z. Zhao and H. Liu. Semi-supervised feature selection via spectral analysis. *In Proceeding of SIAM International Conference on Data Mining*, 2007.

[23] Z. Zhao and H. Liu. Spectral feature selection for supervised and unsupervised learning. *In ICML '07*, pages 1151–1157, 2007.

[24] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. *In ICML*, pages 928–936, 2003.