Applied Intelligence (2024) 54:9429–9445
https://doi.org/10.1007/s10489-024-05681-x

Online learning from capricious data streams via shared and new feature spaces

Peng Zhou[1,2,3] · Shuai Zhang[4] · Lin Mu[1,2,3] · Yuanting Yan[1,2,3]

Accepted: 5 July 2024 / Published online: 16 July 2024

Abstract

Data streams refer to data sequences generated at a high rate over a continuous period, such as social media analysis, financial transaction monitoring, and sensor data processing. Most existing data stream mining methods make assumptions about the feature space, assuming it is either fixed or undergoes regular changes, such as trapezoidal or evolving data streams. However, these restrictions do not hold for real-world applications where data streams may exhibit arbitrary missing features. To address the issue of arbitrary missing features in the feature space, we propose the Online Learning from Capricious Data Streams (OLCDS) algorithm and its variant, OLCDS-I. Specifically, OLCDS first identifies the higher uncertainty features that can provide more information for the optimization model. Then, based on the shared and new feature space, we formulate the constrained optimization problem using the soft margin technique. We deduce the update rules and use model sparsity to retain the essential features for classifier learning. Compared to existing online learning approaches, our new method eliminates the need for feature space assumptions and avoids generating missing features. Extensive experiments compared with five state-of-the-art methods on ten real-world datasets demonstrate the effectiveness and efficiency of our new algorithms

Keywords Online learning · Dynamic feature spaces · Capricious data streams

1 Introduction

Online learning is an efficient method for dealing with dynamic data, which involves repeated interaction between a learner and an environment with an underlying state [1]. The learner predicts the question and the true answer is revealed. A loss is incurred based on the disparity between the learner's prediction and the answer. The primary objective of the learner is to minimize the cumulative loss over all rounds [2]. As the volume of data grows exponentially, many online learning methods have been developed for processing streaming data [3]. Meanwhile, most of these methods assume each streaming instance has a fixed feature space [4, 5]. However, their features may be missing in practical applications for some reasons. For example, when sensors are damaged, or the network is abnormal in ecosystem detection [6], real-time network intrusion [7], and environmental data monitoring [8], their feature spaces will continue to change with the continuous growth of instances. Therefore, online learning on doubly-streaming data (streaming features and instances) is a boiling research problem [9–13].

Doubly-streaming data refers to the increase of both data volume and data dimensions over time [9]. In general, doubly-streaming data has three essential characteristics [14]. (1) Lack of eigenvalue space is arbitrary and irregular, and the features that appeared before may disappear as the number of instances increases. For example, patient profiles are often derived from different testing devices and healthcare providers [15]. Due to factors such as individual medical characteristics, equipment failure, etc., the collected features

✉ Peng Zhou
doodzhou@ahu.edu.cn

Shuai Zhang
zs0920ahu@163.com

Lin Mu
mulin@ahu.edu.cn

Yuanting Yan
ytyan@ahu.edu.cn

[1] Key Laboratory of Intelligent Computing and Signal Processing, Ministry of Education, Hefei, Anhui Province 230601, P.R. China

[2] School of Computer Science and Technology, Anhui University, Hefei, Anhui Province 230601, P.R. China

[3] Information Materials and Intelligent Sensing Laboratory of Anhui Province, Hefei, Anhui Province 230601, P.R. China

[4] School of Information Engineering, Xuzhou Vocational College of Industrial Technology, Xuzhou, Jiangsu Province 221140, P.R. China

are usually incomplete and arbitrarily missing. (2) The distribution of dynamically unbalanced classes. For example, in an anomaly detection system [16], there are only a few anomalies, most of which are normal. (3) Concept drift often occurs, e.g., sudden or gradual drift due to arbitrary changes in data streams [17–19]. Therefore, the challenge of doubly-streaming online learning is dealing with the arbitrary absence of feature space, dynamically unbalanced class distribution, and concept drift simultaneously.

As shown in Fig. 1, assume that the instances at timestamps t_1 and t_2 are x_1 and x_2, respectively. From Fig. 1, it is apparent that x_2 exhibits disappearing features, indicated by the blank, which signifies the absence of certain features compared to the previous moment (x_1). Additionally, new features have emerged, denoted by the green. The gray represents the shared feature space, consistent with the previous moment. We refer to the above doubly-streaming data as capricious data streams where data dynamically changes in volume and feature dimension [20]. The problem of online learning from capricious data streams is much more complex than traditional online learning problems. In other words, traditional online learning algorithms cannot handle capricious data streams because the feature space is generally assumed to be unchanged [21–23]. Therefore, the main challenge of learning from capricious data streams is dynamically designing classifiers that can learn from increasing training instances with capricious feature spaces.

Recently, a few related studies have attempted to address doubly-streaming online learning problems, including Online Learning with Streaming Features (OLSF) [9], Feature Evolvable Streaming Learning (FESL) [10], Online Learning from varying Features (OLVF) [11], and Generative Learning with Streaming Capricious (GLSC) [20]. However, they have some limitations. Specifically, OLSF can only process the trapezoidal data streams, and FESL can only process the evolvable feature data streams (change regularly). OLVF is used to deal with varying feature spaces but cannot handle arbitrary missing features. GLSC can handle capricious data streams by constructing a model on a shared feature space that incorporates historical and new features. Consequently, GLSC's time complexity is relatively higher. Motivated by this, this paper proposes a novel online learning algorithm to handle arbitrarily missing feature spaces of doubly-streaming online learning. Our proposed algorithms, namely OLCDS and its variant, OLCDS-I, offer a novel approach that eliminates the need for any assumptions on the feature space and avoids generating missing features.

Specifically, we aim to construct an efficient classification model using the incomplete feature space. Upon the arrival of capricious data streams, we receive new streaming instances and extract the shared and newly emerging feature space between the current and previous instances. Then, we project the weight vector w_t of the classifier and the new arriving instance x_t onto the shared and new feature space, respectively. To make predictions, we compute confidences for the shared and the new feature space and subsequently calculate their predicted values. We employ a feature space confidence metric to identify the most informative observed features. By leveraging this metric, our algorithms effectively handle data streams with arbitrary changes in the feature space. We quantify the loss between predicted and actual values by employing a loss function. Finally, we update the model based on the calculated loss. This iterative process allows us to refine the classification model and improve its accuracy continually. The main contributions of this paper are as follows:

– Two new methods are proposed to handle the issue of online learning from capricious data streams. These two algorithms do not rely on any assumptions about the feature space and can effectively handle arbitrary missingness in the feature space. Since our new methods eliminate the need for feature space assumptions, they are closer to practical application requirements.
– We formulate the constrained online optimization problem based on the shared and new feature space by identifying higher uncertainty features. Since our new

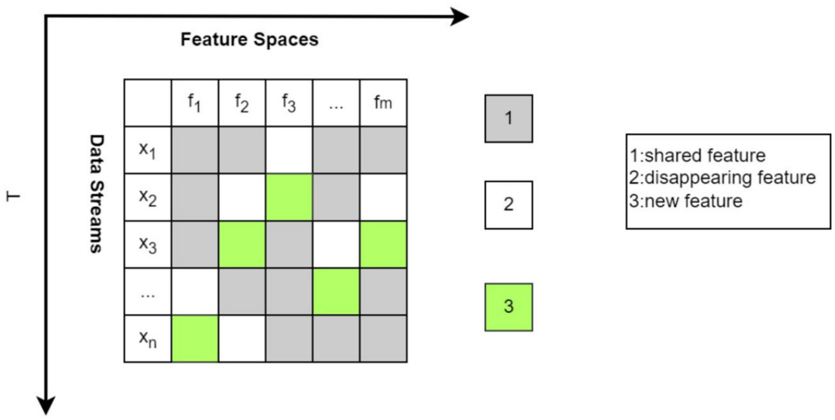


Fig. 1 Illustration of capricious data streams. $x_1, x_2, ..., x_n$ are the streaming instances, while $f_1, f_2, ..., f_m$ are the streaming features. The feature spaces may differ (features disappear or emerge) for different instances, such as x_1 and x_2

methods avoid generating missing features, they are more efficient than existing online learning algorithms. Additionally, we conducted a thorough theoretical analysis to determine the upper bound of the cumulative hinge loss.
- Extensive experiments are conducted on ten real-world datasets compared with online learning methods for fixed feature space, trapezoidal, and capricious data streams. Experimental results of statistical tests indicate the effectiveness of our proposed algorithms.

The rest of this article is organized as follows. In Section 2, we describe related work. Section 3 provides a formal definition of the problem. Section 4 presents the proposed OLCDS and OLCDS-I algorithms in detail. Section 5 gives the theoretical analysis. Section 6 conducts extensive experiments. Section 7 discusses the advantages and disadvantages of our new method. Finally, Section 8 concludes the paper.

2 Related work

This paper focuses on online learning from capricious data streams. We mainly review the related works from the following two aspects: online learning from static feature space and online learning from dynamic feature space.

2.1 Online learning from static feature space

Online learning is a dynamic process where the learner receives a training example in each round, akin to a question with a concealed answer [2]. Over the years, numerous algorithms have been devised and utilized for online learning to optimize the learning process and improve prediction accuracy. These algorithms encompass various techniques and methodologies, each with strengths and limitations. Their application in online learning has led to significant advancements in various domains.

In online learning, first-order algorithms utilize the information from the first-order derivatives to update the model parameters. Two commonly employed first-order algorithms are the Perceptron algorithm [24, 25] and the Online Gradient Descent (OGD) [26] algorithm. These algorithms have found wide application and have proven effective in various learning tasks.

On the other hand, second-order online learning algorithms leverage the second-order derivative information to explore the underlying structure between features better. By considering the curvature of the space, these algorithms enhance convergence and can minimize quadratic functions within a limited number of steps. Notable representatives of second-order online learning algorithms include the Second-Order Perceptron [27], the Normal Herding method via Gaussian Herding (NHERD) [28], the Confidence-Weighted learning and Soft Confidence-Weighted algorithm (SCW) [29], the Adaptive Regularization of Weight Vectors (AROW) [30], and a New variant of Adaptive Regularization (NAROW) [31]. These second-order algorithms have demonstrated their efficacy in capturing complex relationships and exploiting the underlying structure of the data. They offer improved convergence rates and can be advantageous when dealing with non-linear or highly correlated features. Their utilization of online learning has contributed to advancements in various research and application areas.

However, these online learning algorithms mentioned above, regardless of whether they use first-order or second-order gradients, face a limitation in handling data streams with arbitrary changes in feature space. This limitation arises from their assumption that the feature space remains fixed and does not undergo any changes. Although these algorithms can improve performance, this assumption does not hold in real-world applications.

2.2 Online learning from dynamic feature space

The dynamic feature space means that the feature space of the data streams is constantly changing. In general, online learning from a dynamic feature space is more challenging.

Specifically, Zhang et al. [9] were the first to deal with trapezoidal data streams where the volume and feature dimensions of the data increase simultaneously. They proposed OLSF with its Variants OLSF-I and OLSF-II. OLSF first divides the features of the current training instance into historical features and new features. Then, a classifier updates historical features and new features by following different update rules. Recently, a few works have been proposed to learn from data streams with varying feature space, where features would vanish or occur over time. For example, Hou et al. [10] proposed the FESL algorithm. FESL first recovers historical data features through a mapping function in which both the learned historical features and new features exist, and then, it learns two models from the features of the above two parts. Finally, ensemble learning is used to make the final prediction. Based on FESL, Zhang et al. [12] and Hou et al. [13] conducted in-depth exploration and expansion of FESL scenarios and proposed evolutionary difference minimization (EDM) [12] and prediction based on unpredictable feature evolution (PUFE) [13] to deal with feature space of algorithmic variations for different situations in data streams. Besides, Beyazit et al. [11] proposed the OLVF will project the existing model and the current instance onto shared feature space and make a prediction. He et al. [20] proposed the GLSC algorithm, which establishes the relationships between historical and new features by constructing a model on a universal feature space. Zhou et al. [32] proposed an online subspace learning method for uncertain feature data streams. Specifically, the method maps heterogeneous feature instances into a low-dimensional subspace and then learns a classifier in this latent subspace. Gu et al. [33] proposed an algorithm for solving the problem of incremental feature space learning with label scarcity (FLLS), which uses an active learning strategy to select valuable instances for annotation and build a superior prediction model with minimal supervision. He et al. [34] proposed an online learning algorithm called OCDS to handle data streams with constantly changing feature spaces. Unlike traditional methods, OCDS does not make any assumptions about the dynamics of feature spaces. It adds newly emerged features to the general feature space by building a correlation graph model between features. Liu et al. [35] proposed an online active learning algorithm that combines an active query strategy with a passive-aggressive (PA) update strategy to handle binary and multi-class online classification tasks on trapezoidal data streams.

In addition, Wang et al. [36] proposed an improved noise-resistant adaptive long short-term memory neural network (ANA-LSTM) model for reliable prediction of the remaining life of lithium-ion batteries. The model has highly robust feature extraction and optimal parameter characterization. Wang et al. [37] proposed an improved multi-time scale singular filter-Gaussian process regression-long short-term memory (SF-GPR-LSTM) model for estimating the remaining capacity of lithium-ion batteries throughout their life cycle, adapting to rapid aging and multiple current changes. The model achieves rapid evaluation of battery performance by optimizing the multi-task training strategy.

However, the existing algorithms mentioned above cannot effectively handle data streams that undergo arbitrary changes in the feature space. This limitation arises from the assumption made by these algorithms that the feature space remains constant or changes predictably. Although a few algorithms can address data streams with arbitrary changes in the feature space, they do so at the expense of additional computational time, as they require the generation of missing features.

3 Problem definition

This section presents the formal definition of traditional online learning and online learning from capricious data streams. We summarize some symbols used in this paper in Table 1.

Table 1 Summary on mathematical notations

Notations	Definition
x_t	the instance at round t
d_t	dimension of instance x_t
d_t^s	dimensions of x_t on shared feature space
d_t^n	dimensions of x_t on new feature space
R^{d_t}	the feature space of x_t
T	$T \in N^+$, the number of total iteration
u	$u \in R^{d_t}$, arbitrary vector in R^{d_t}
w_t	$w_t \in R^{d_{t-1}}$,classifer build at round t-1
y_t	y_t ∈{-1,+1},true label of x_t
l_t	l_t=max{0,1-$y_t(x_t \cdot w_t)$}, hinge loss on instance x_t
$\hat{y}_t$	$\hat{y}_t$ ∈{-1,+1},predicted label of x_t
x_t^d	the disappearing feature of x_t and x_{t-1}
x_t^s	the shared feature of x_t and x_{t-1}
x_t^n	the new feature of x_t and x_{t-1}
w_t^s	Projection of w_t on x_t^s
w_t^n	Projection of w_t on x_t^n
h_t^i	the informativeness of the i-th feature in instance x_t
p_t^s	the confidence of the feature on the shared feature space
p_t^n	the confidence of the feature on the new feature space
ξ	slack variable
τ	learning rate variable
ϵ	a constant about p_t^s and p_t^n
λ	$\lambda > 0$, regularization parameter of calssifier
C	$C > 0$, penalty cost parameter
B	B ∈(0,1], proportion of selected features

3.1 Traditional online learning

We consider using a linear classifier for binary classification. Over time, data instances gradually arrive. At each round t, the algorithm receives an instance x_t and predicts the true label y_t ∈{+1,-1} of the instance x_t using its classifier w_t. The prediction uses the function $\hat{y}_t = sign(x_t \cdot w_t)$. After prediction, display the true label of instance x_t. Calculate their loss $l(y_t, \hat{y}_t)$ based on the difference between the predicted value $\hat{y}_t$ and the true value y_t.

Currently, the widely used loss function is the Hinge loss function [38–40] defined as $l(y_t, \hat{y}_t) = max\{0, 1 - y(x_t \cdot w_t)\}$. It is based on the margin between the instance x_t and the classifier w_t. The margin is calculated by $y_t(x_t \cdot w_t)$. Since online learning involves incremental tasks, minimizing the loss while making minimal changes to the current model is crucial. This approach ensures that knowledge from previous instances is preserved. Moreover, in the presence of noisy data, insisting on perfect predictions for every instance can lead to overfitting and limited generalization. Instead, training a classifier that can effectively separate large amounts of data while disregarding noise is preferable. To achieve this, soft decision margins [41] are utilized, allowing the classifier to tolerate some errors. Based on these, many online learning algorithms combine these constraints and frame weight learning as an optimization problem as follows:

$$w_{t+1} = \underset{w:l_t<\xi}{\arg\min} \frac{1}{2} \|w - w_t\|^2 + C\xi, \tag{1}$$

where (1) introduces slack variables to the discriminator, introducing nonlinearity and allowing a controlled level of error, denoted by ξ. The parameter C adjusts the slackness of the constraint. Considering the presence of noise in real-life data, soft-edge methods are employed for model learning across diverse feature spaces.

3.2 Online learning from capricious data streams

First of all, we define capricious data streams as follows:

Definition 1 *[Capricious Data Streams]*
*Let $\mathcal{D} = \{x_1, x_2, \ldots, x_n\}$ be the streaming instances, and F_t= $\{f_1, f_2, \ldots, f_{d_t}\}$, $F_t \in R^{d_t}$ is the feature set carried by instance x_t. Two streaming instances, x_i and x_j, their feature spaces are denoted as F_i and F_j. If a feature, f_j, is present in F_j but absent in F_i, it is considered a **new feature**. Conversely, if a feature, f_i, exists in F_i but not in F_j, it is referred to as a **disappearing feature**. If F_i and F_j are not identical, we define $\mathcal{D}$ as **capricious data streams**.*

Then, we present the formal definition of online learning from capricious data streams as follows:

Definition 2 *[Online Learning from Capricious Data Streams]*
Consider capricious data streams $\mathcal{D} = \{(x_t, y_t)\big|t = 1, 2, \ldots T\}$ as a sequence of training instances, where $x_t \in R^{d_t}$ represents the instance x_t of d_t dimensions, and $y_t \in$ {-1,+1} represents the ture class label. During the t-th iteration, the learner observes the instance x_t and provides its prediction $\hat{y}_t$. Then, the true label y_t is revealed, and the learner suffers a momentary loss, reflecting the discrepancy between the prediction and the ground truth. Online Learning from Carpicious Data Streams aims to construct an efficient classification model using the incomplete feature spaces of data streams.

4 The proposed method

This section first presents the details of problem analysis and solutions for the issue of online learning from capricious data streams. Then, we propose the new algorithm OLCDS in detail and give the complexity analysis of it.

4.1 Problem analysis and solutions

Online learning from capricious data streams presents significant challenges due to the unpredictable nature of the data-the lack of consistency in the feature space results in considerable degradation of classifier performance. The primary reason for this decline is that the new feature space lacks prior knowledge and can only rely on the known feature space for making predictions. Additionally, disappearing features further compound the problem as they do not contribute valuable information to update the classifier. Online learning from capricious data streams is inherently difficult and poses substantial obstacles.

The flow chart of our new method is shown in Fig. 2. To enhance prediction accuracy, the feature space of x_t is partitioned into two segments based on the presence of the feature

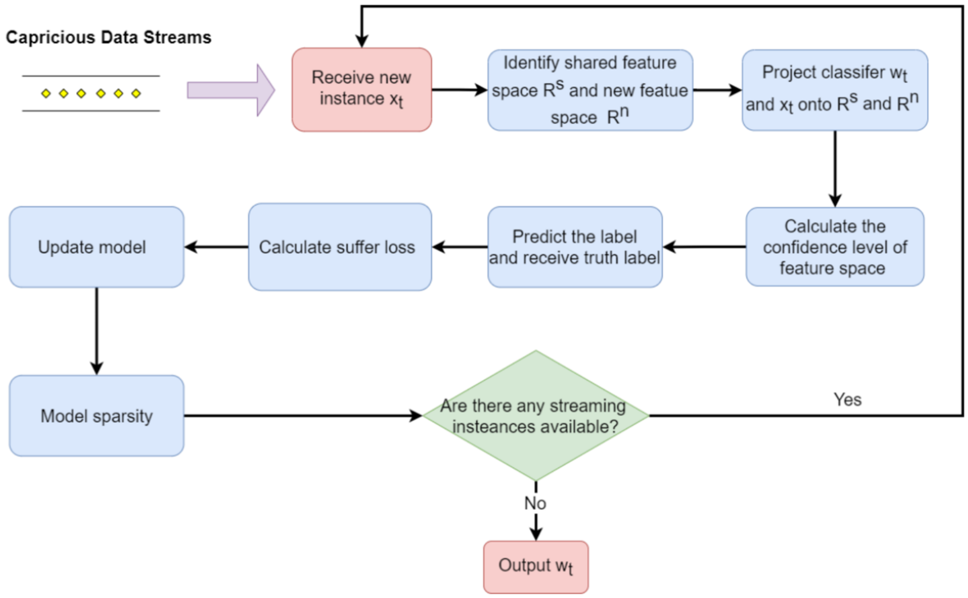

Capricious Data Streams
Receive new instance x_t
Identify shared feature space R^S and new featue space R^n
Project classifer w_t and x_t onto R^S and R^n
Calculate the confidence level of feature space
Predict the label and receive truth label
Calculate suffer loss
Update model
Model sparsity
Are there any streaming insteances available?
Yes
No
Output w_t

Fig. 2 The flow chart of our new method

in x_t and the preceding instance. These segments are referred to as shared and new feature spaces denoted as R^s and R^n respectively. At the t-th iteration, we project w_t on R^s and R^n, which are represented as w_t^s and w_t^n respectively. A step-by-step, in-depth analysis of the problem and corresponding solutions follow.

4.1.1 Estimating the informativeness of features from arbitrarily varying feature spaces

To enhance the learning process and optimize the model, we propose an uncertainty-based adaptive weighting strategy for dynamically learning the capricious data streams. The rationale behind this strategy is that features with higher uncertainty tend to contain more valuable information for model optimization [42, 43]. We leverage the variance of features during the iterative process to assess their uncertainty. A feature with a more considerable variance indicates higher uncertainty and can offer more information for the optimization model. Notably, the variance solely depends on the instance and remains unaffected by external factors. Thus, we utilize variance as a proxy for uncertainty to evaluate feature informativeness.

Specifically, after projecting the training instances into the feature space, we compute the cumulative average of feature information as the confidence level of the feature space. This confidence level is the weight assigned to the feature space in the prediction results. During iteration t, we denote h_t^i as the amount of information associated with the i-th feature in the instance x_t. Consequently, the confidence levels for the shared feature space p_t^s and the new feature space p_t^n can be defined as follows:

$$p_t^s = \sum_{i=1}^{d_t^s} \frac{h_t^i}{\sum_{j=1}^{d_t} h_t^j}, \tag{2}$$

$$p_t^n = \sum_{i=1}^{d_t^n} \frac{h_t^i}{\sum_{j=1}^{d_t} h_t^j}, \tag{3}$$

where d_t^s and d_t^n are the dimensions of the shared feature space and the new feature space respectively.

4.1.2 Learning from supervised loss

Linear classifiers are generally used for binary classification. When receiving an instance x_t, predict the true label y_t ∈{-1,+1} of the instance x_t bt using its classifier w_t. After the prediction is completed, the true label of the instance x_t is displayed. Then, we update the model by calculating the difference between the predicted value $\hat{y}_t$ and the true value y_t using the loss function $l(y_t, \hat{y}_t)$. One of the most widely used loss functions is the Hinge loss defined as l(y_t,$\hat{y}_t$)=max{0,1-$y_t(x_t \cdot w_t)$}, which is based on the margin between the instance x_t and the classifier w_t.

In order to adapt the classifier to the dynamic nature of the capricious data streams, we modify the hinge loss to train a classifier w_t. We define the loss l_t of the classifier at iteration t as :

$$l_t = l(y_t, \hat{y}_t) = max\{0, 1 - y_t(p_t^s \cdot w_t^s \cdot x_t^s + p_t^n \cdot w_t^n \cdot x_t^n)\}. \tag{4}$$

In online learning tasks, the goal is to minimize the cumulative loss by constructing and adjusting the weights of each observed instance. However, this approach can be sensitive to noise and may result in overfitting. Therefore, a soft margin technique has been widely adopted to overcome this limitation. This technique introduces a slack variable ξ, where ξ ∈[0, 1] allows for a certain level of misclassification. Many online learning algorithms combine the abovementioned constraints to address the trade-off between maximizing accuracy and minimizing the loss. They formulate the learning of weights as an optimization problem, where the objective is to find the optimal weights that minimize the loss while considering the soft margin constraint. By formulating the problem, these algorithms balance fitting the training data and generalizing well to unseen instances.

Therefore, by using (1) and (4), our learning task can be formulated as a constrained optimization problem:

$$w_{t+1} = \underset{\substack{w=[w^s, w^n]: \\ l_t \leq \xi}}{\arg\min} \frac{1}{2} \left\| w^s - w_t^s \right\|^2 + \frac{1}{2} \left\| w^n \right\|^2 + C\xi, \tag{5}$$

where w^s 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), w^n denotes the vector consisting of elements of w_{t+1} which are not in the feature space of w_t, w_t^s represents projection of w_t on x_t^s, $C > 0$ is a penalty cost parameter, ξ is a slack variable, and $l_t = l(y_t, \hat{y}_t) = max\{0, 1 - y_t(p_t^s \cdot w_t^s \cdot x_t^s + p_t^n \cdot w_t^n \cdot x_t^n)$.

4.1.3 Derive updating rules

We use a Lagrangian function with K.K.T. conditions to solve the optimization problem of the above inequality and get the following updated rules:

$$\begin{aligned} w^s &= w_t^s + \tau p_t^s y_t x_t^s \\ w^n &= \tau p_t^n y_t x_t^n \\ \eta &= C - \tau \end{aligned} \tag{6}$$

Using these conditions, we can get the representation of our problem and find τ:

$$L(\tau) = \frac{1}{2}\tau^2(p_t^s)^2\left\|x_t^s\right\|^2 + \frac{1}{2}\tau^2(p_t^n)^2\left\|x_t^n\right\|^2 \\ +\tau(1 - y_t(p_t^s \cdot w_t^s \cdot x_t^s + p_t^n \cdot w_t^n \cdot x_t^n)) \tag{7}$$

$$\tau = min\{C, \frac{l_t}{(p_t^s)^2\left\|x_t^s\right\|^2 + (p_t^n)^2\left\|x_t^n\right\|^2}\} \tag{8}$$

The objective function of (5) is linearly proportional to ξ. Although it can solve the over-fitting problem, its numerical stability is poor, and its punishment for significant errors is minor, thus affecting classification accuracy. Because the square of the slack variable ξ can effectively solve these problems. Therefore, our optimization problem uses the square of the slack variable ξ as follows:

$$w_{t+1} = \underset{\substack{w=[w^s,w^n]:\\ l_t\leq\xi}}{\arg\min}\ \frac{1}{2}\left\|w^s - w_t^s\right\|^2 + \frac{1}{2}\left\|w^n\right\|^2 + C\xi^2. \tag{9}$$

We also use a Lagrangian function with K.K.T. conditions to solve the optimization problem of the above inequality and get the following updated rules:

$$w^s = w_t^s + \tau p_t^s y_t x_t^s \\ w^n = \tau p_t^n y_t x_t^n \\ 2C\xi = \tau \tag{10}$$

Putting (10) into (9), we can get the following formula:

$$L(\tau) = \frac{1}{2}\tau^2(p_t^s)^2\left\|x_t^s\right\|^2 + \frac{1}{2}\tau^2(p_t^n)^2\left\|x_t^n\right\|^2 \tag{11}$$

$$+\tau(1 - y_t(p_t^s \cdot w_t^s \cdot x_t^s + p_t^n \cdot w_t^n \cdot x_t^n) - \frac{\tau}{4C}),$$

$$\tau = min\{C, \frac{l_t}{(p_t^s)^2\left\|x_t^s\right\|^2 + (p_t^n)^2\left\|x_t^n\right\|^2 + \frac{1}{2C}}\}. \tag{12}$$

To sum up, we can derive the update rules as follows:

$$w_{t+1} = [w_{t+1}^s, w_{t+1}^n] = [w_{t+1}^s + \tau_t p_t^s y_t x_t^s, \tau_t p_t^n y_t x_t^n], \tag{13}$$

$$\text{where } \tau_t = \begin{cases} \min\{C, \frac{l_t}{(p_t^s)^2\left\|x_t^s\right\|^2 + (p_t^n)^2\left\|x_t^n\right\|^2}\} & \text{(OLCDS)} \\ \min\{C, \frac{l_t}{(p_t^s)^2\left\|x_t^s\right\|^2 + (p_t^n)^2\left\|x_t^n\right\|^2 + \frac{1}{2C}}\} & \text{(OLCDS-I)} \end{cases}$$

Based on these two cases of τ_t, we propose two new algorithms named OLCDS and OLCDS-I in Section 4.3.

4.2 Model sparsity

The infinite nature and high dimensions of data streams present challenges when retaining all features in a classifier. Doing so can lead to a degradation in classifier performance due to increased memory requirements and computational expenses. To address this issue, it becomes necessary to select and retain only the most essential features for classifier learning by intercepting w_t.

One commonly used approach is to truncate the classifier after projecting it onto the $L1$ sphere. However, this truncation strategy can introduce a bias towards infrequent features in the data stream. These features tend to have smaller weights and are more easily truncated. Moreover, small changes in the weights of features with high uncertainty can result in different outcomes. Therefore, retaining the most significant features is crucial while avoiding bias towards features that occur in only a few instances.

We incorporate relative uncertainty into the feature selection process to address this concern. When features exhibit higher uncertainty, we prioritize retaining their weights to capture their potential significance. In practice, this is achieved through a projection process that involves the following steps:

$$w_t = min\left\{1, \frac{\lambda}{\langle w_t \cdot H_t\rangle}\right\} w_t, \tag{14}$$

where λ is a regularization parameter that truncates w_t to give the model good sparsity. H_t=[h_t^1, h_t^2, h_t^3, ..., h_t^p] represents the relative uncertainty vector of the public feature space at the t-th iteration, which is composed of the information content of all observed features. After the update operation is completed, projection and truncation are introduced to prune redundant features based on parameter B, where $B \in (0, 1]$.

4.3 The proposed algorithm

Based on the above analyses and solutions, we propose a novel Online Learning algorithm specifically designed to handle Capricious Data Streams, named OLCDS, as shown in Algorithm 1.

Specifically, our new algorithms have three parameters, λ, C, and B, that indicate the penalty parameter, the regularization parameter, and the proportion of selected features, respectively. w_t is the weight vector of the latest classifier, where t is the timestamp during online learning. In Step 2, we receive a new instance x_t at timestamp t. Step 3 identifies the shared and new feature space as R^s and R^n between x_{t-1} and x_t. In Step 4, the weight vector w_t and instance x_t are projected to the shared feature space R^s and the new feature space R^n, respectively. Next, Step 5 calculates the confidence values in the shared and new feature space using formula (2) and formula (3), respectively. In terms of these confidence values, the predicted value $\hat{y}_t$ of the instance x_t is calculated in Step 6. Then, the true label y_t of the instance x_t

Algorithm 1 OLCDS and Its Variants OLCDS-I.

```
Input:
    $C > 0$: the penalty parameter;
    $\lambda > 0$: the regularization parameter;
    $B \in (0, 1]$: the proportion of selected features;
Onput:
    $w_t$: the latest classifier;
Initialize:
    $w_1 = \{0, 0, \ldots, 0\} \in R^{d_1}$;
1: For $t$=1, 2, …, $T$ do
2:     Receive instance: $x_t \in R^{d_t}$;
3:     Identify shared and new feature spaces as: $R^s = R^{x_{t-1}} \cap R^{x_t}$, $R^n = R^{x_t} - R^s$;
4:     Project $w_t, x_t$ onto $R^s$, $R^n$ as: $w_t^s$, $w_t^n$, $x_t^s$, $w_t^n$;
5:     Calculate the confidence of features as: $p_t^s$ and $p_t^n$;
6:     Predict: $\hat{y}_t$ =sign($p_t^s \cdot w_t^s \cdot x_t^s$+$p_t^n \cdot w_t^n \cdot x_t^n$);
7:     Receive true label: $y_t$ ∈{-1,+1};
8:     Suffer loss: $l_t$=max{0,1-$y_t(p_t^s \cdot w_t^s \cdot x_t^s + p_t^n \cdot w_t^n \cdot x_t^n)$};
       //Model Update
9:         set:$\tau_t = \begin{cases} \min\{C, \frac{l_t}{(p_t^s)^2 \|x_t^s\|^2 + (p_t^n)^2 \|x_t^n\|^2}\} & \text{(OLCDS)} \\ \min\{C, \frac{l_t}{(p_t^s)^2 \|x_t^s\|^2 + (p_t^n)^2 \|x_t^n\|^2 + \frac{1}{2C}}\} & \text{(OLCDS-I)} \end{cases}$
10:        $w_{t+1} = [w_t^s + \tau_t p_t^s y_t x_t^s, \tau_t p_t^n y_t x_t^n]$;
       //Model Sparsity
11:        Project $w_t = min\left\{1, \frac{\lambda}{\langle w_t \cdot H_t \rangle}\right\} w_t$
12:        truncate $w_t$ based on $B$;
13: End For
```

is obtained in Step 7. Subsequently, the loss between the predicted value $\hat{y}_t$ and the true value y_t is calculated by formula (4) in Step 8. To update the model in real-time, Steps 9-10 outline the specific update strategies of the OLCDS and OLCDS-I algorithms. These strategies adaptively adjust the model based on the observed instances and their associated losses. Finally, in Steps 11-12, the weight vector w_t is truncated using parameter B. The model only processes part of the critical data. Therefore, introducing the sparsity of the model can improve its efficiency.

In summary, Algorithm 1 presents a comprehensive framework for effectively handling capricious data streams in online learning. The algorithm achieves accurate predictions by dynamically adapting the model based on incoming instances while maintaining model sparsity through feature selection.

4.4 Algorithm complexity

Algorithm 1 presents the pseudocode for OLCDS and its variant, OLCDS-I. Regarding time complexity, for a single iteration, $|x_t|$ is the number of features observed in the current training instance, and $|w_t|$ is the number of features in the current classifier. Since identifying and projecting features into feature space is related to the classifier and the training instance, the time complexity of identifying features and projecting them into feature space is O($|w_t| + |x_t|$). Therefore, for a single iteration, we know that the time complexity of the OLCDS and OLCDS-I algorithm are O($|w_t| + |x_t|$). This indicates that the runtime of both two algorithms is linear to the size of the weight vector w_t and the incoming instance x_t.

Regarding space complexity, the main space requirements of the entire algorithm in each iteration come from two aspects: storing the weight vector w_t and storing the current instance x_t. Suppose the maximum dimension is m, so the space complexity of our new algorithm is O(m). Since our algorithm does not need to cache all samples, its space complexity is very low.

OLCDS and OLCDS-I are designed to process and update the model efficiently based on the observed instances, ensuring that the computational overhead remains manageable even with large-scale datasets. The linear runtime complexity of OLCDS and OLCDS-I allow for effective online learning, making them suitable for real-time applications where timely model updates are crucial.

5 Theoretical analysis

In this section, we utilize regret from online learning as a metric to evaluate the performance of the OLCDS algorithm.

Initially, we examine the upper bound of the cumulative hinge loss of OLCDS in an ideal scenario where the learner can accurately predict each instance. We then extend and derive this upper bound as linearly inseparable. Additionally, we present error rate bounds for each class in OLCDS. These bounds ensure that our algorithm consistently achieves a lower cumulative hinge loss compared to the best-fixed prediction, which is chosen retrospectively for any sequence of instances.

When a learner incorrectly predicts an instance during iteration t, we observe that $y_t(p_t^s w_t^s x_t^s + p_t^n w_t^n x_t^n)$ >0, and the loss function l_t >1. As a result, the cumulative hinge loss $\sum_{t=1}^{T} l_t$ is an upper bound of the number of misclassified instances. We represent the loss of offline predictor during iteration t as l_t^x, which is defined as follows:

$$l_t^x = l(w^{x_t}; (x_t, y_t)) \tag{15}$$

Where $w \in R^u$ represents an arbitrary vector, and w^{x_t} represents the projection of w on x_t. This notation also applies to w^{w_t}, $w^{w_{t+1}}$, $w^{w_t^s}$, $w^{x_t^n}$, $w^{x_t^s}$ and $w^{x_t^n}$. Then, we have Lemma 1 as follows.

Lemma 1 *Let $(x_1, y_1), \ldots, (x_T, y_T)$ be a sequence of training data, where $x_t \in R^{d_t}$ and y_t ∈{+1,-1} for all t. Let* *learning rate* $\tau_t = min\{C, \frac{l_t}{(p_t^s)^2\|x_t^s\|^2+(p_t^n)^2\|x_t^n\|^2}\}$*, as given in* (8)*. The following bound holds for any* $w \in R^u$*:*

$$\sum_{t=1}^{T}\{\tau_t\{2l_t - \frac{2}{\epsilon}l_t^x - \frac{2(\epsilon-1)}{\epsilon} - \tau_t[(p_t^s)^2\,\|x_t^s\|^2 + \ (p_t^n)^2\,\|x_t^n\|^2]\}\} \leq \|w\|^2\,. \tag{16}$$

Proof 1 *Let's define* Δ_t *to be* $\|w_t - w^{w_t}\|^2$-$\|w_{t+1} - w^{w_{t+1}}\|^2$*. In order to prove the lemma, we need to bound the sum of all* Δ_t *over T. It is worth noting that the* $\sum\Delta_t$ *can be simplified using the telescoping property, leading to the following expression:*

$$\begin{aligned}\sum_{t=1}^{T-1}\Delta_t &= \sum_{t=1}^{T-1}(\|w_t - w^{w_t}\|^2 - \|w_{t+1} - w^{w_{t+1}}\|^2)\\ &= \|w_1 - w^{w_1}\|^2 - \|w_{t+1} - w^{w_{t+1}}\|^2\\ &\leq \left\|w^{w_1}\right\|^2.\end{aligned} \tag{17}$$

This establishes an upper limit for $\sum\Delta_t$*. Moving forward, we will present a lower limit for individual* Δ_t*.*

If an example x_t *is classified correctly, it means that the values of* l_t *and* τ_t *are both 0, and the classifier remains unchanged. However, let's now consider the scenario where the learner fails to satisfy the minimum margin requirement, indicated by* $l_t > 0$*. In such cases, we can gather insights from the confidences obtained through solving the optimization problem and the sparse step. After truncation, we find that* w_{t+1}=$[w_t^s + \tau_t p_t^s y_t x_t^s, \tau_t p_t^n y_t x_t^n]$*. Thus, we can conclude that:*

$$\begin{aligned}\Delta_t &= \|w_t - w^{w_t}\|^2 - \|w_{t+1} - w^{w_{t+1}}\|^2\\ &\geq \|w_t^s - w^{w_t^s}\|^2 - \|w_t^s + \tau_t p_t^s y_t x_t^s - w^{w_t^s}\|^2\\ &\quad -\|\tau_t p_t^n y_t x_t^n - w^{w_t^n}\|^2\\ &= -2\|\tau_t p_t^s y_t x_t^s\|^2 \cdot \|w_t^s - w^{w_t^s}\|^2 - \|\tau_t p_t^s y_t x_t^s\|^2\\ &\quad -\|\tau_t p_t^n y_t x_t^n - w^{w_t^n}\|^2\end{aligned} \tag{18}$$

After initializing w_t^n *as a zero vector, the disappeared feature space remains unobserved. By considering the expression* $l_t = 1 - y_t(p_t^s w_t^s x_t^s + p_t^n w_t^n x_t^n)$*, and* $l_t^x \geq 1 - y_t(w^{x_t} \cdot x_t)$*, We introduce* $\epsilon = (w^{w_t^s}x_t^s + w^{w_t^n}x_t^n)/(p_t^s w^{x_t^s}x_t^s + p_t^n w^{x_t^n}x_t^n)$*. Consequently, we have* $y_t(p_t^s w^{x_t^s}x_t^s) = 1 - l_t$ *and* $y_t(p_t^s w^{x_t^s}x_t^s + p_t^n w^{x_t^n}x_t^n) \geq (1 - l_t^x)/\epsilon$*. Here,* ϵ *represents a constant related to* p_t^s *and* p_t^n*. Combining these and* (18)*, we obtain:*

$$\begin{aligned}\Delta_t &\geq 2\tau_t\|(p_t^s y_t w^{x_t^s}x_t^s + p_t^n y_t w^{x_t^n}x_t^n) - p_t^s y_t w_t^s x_t^s\|^2\\ &\quad -\tau_t^2[(p_t^s)^2\|x_t^s\|^2 + (p_t^n)^2\|x_t^n\|^2 - \|w^{w_t^n}\|^2]\\ &= \tau_t\{2l_t - \tau_t[(p_t^s)^2\|x_t^s\|^2 + (p_t^n)^2\|x_t^n\|^2]\\ &\quad -\frac{2}{\epsilon}(l_t^x - 1) - 2\} - \|w^{w_t^n}\|.\end{aligned} \tag{19}$$

By considering the upper bound defined in (17) *and the lower bound specified in* (19)*, we can conclude that:*

$$\begin{aligned}&\sum_{t=1}^{T}\{\tau_t\{2l_t - \frac{2}{\epsilon}l_t^x - \frac{2(\epsilon-1)}{\epsilon} - \tau_t[(p_t^s)^2\,\|x_t^s\|^2 + (p_t^n)^2\,\|x_t^n\|^2]\}\}\\ &\leq \|w^{w_1}\|^2 + \sum_{t=1}^{T-1}\|w^{w_t^n}\|^2\\ &= \|w\|^2.\end{aligned} \tag{20}$$

Hence, Lemma 1 *is proved.*

6 Experiments

In this section, we conduct an empirical evaluation to validate the performance of our proposed algorithms, OLCDS and OLCDS-I. Firstly, we present the details of the experimental setup. Then, we compare the performance of our new algorithms with some baseline algorithms, including 1) traditional online learning algorithms for fixed feature space, 2) online learning algorithms for trapezoidal data streams; and 3) online learning algorithms for capricious data streams. Finally, we give the parameter analysis of our new method.

6.1 Experimental setup

6.1.1 Datasets

In order to assess the performance of these competing algorithms, we conducted experiments on ten datasets obtained from machine learning repositories. The specific datasets used in our study are listed in Table 2 and are readily available for download from the UCI machine learning repository,[1] accessible free of charge.

6.1.2 Evaluation metrics

In this paper, we adopt the F_β-measure with $\beta = 1$ as the default evaluation metric to assess the performance of our algorithm. In many cases, Accuracy is a commonly used metric in classification, which represents the ratio of the number of correctly classified samples to the total number of samples. However, in the case of class imbalance, Accuracy may give misleading results. F-measure is a metric that considers both Precision and Recall. Precision represents the proportion of samples classified as positive that are truly positive, while Recall represents the proportion of all truly positive samples

[1] http://archive.ics.uci.edu/ml/

Table 2 Experimental datasets

Data Set	Instances	Features
wdbc	569	30
splice	3190	60
credit-a	690	15
svmguide3	1243	22
spambase	4601	57
ionosphere	351	33
spect	267	22
libras	360	90
dermatology	358	34
arrhythmia	452	279

that are correctly classified as positive. F-measure is the harmonic mean of Precision and Recall, which is more robust to class imbalance. Therefore, choosing the F-measure as a metric can better reflect the algorithm's performance when processing data streams. The specific calculation formula of the F-measure is as follows:

$$F-measure = \frac{2 * Precision * Recall}{Precision + Recall}, \tag{21}$$

where $Precision = \frac{TP}{TP+FP}$, $Recall = \frac{TP}{TP+FN}$, TP represents true positives, FP denotes false positives, and FN represents false negatives.

To determine if there are significant differences in the prediction of F-measure among these competing algorithms, we conducted a Friedman test at a 95% significance level [44]. Rejecting the null hypothesis of the Friedman test indicates a noteworthy disparity in performance among these competing algorithms. Upon rejecting the null hypothesis, we employed the Nemenyi test as a post-hoc test [44].

6.1.3 Competing algorithms

There are three types of competing algorithms: 1) Traditional online learning algorithms for fixed feature space data streams: OGD [26] and RDA [45]; 2) Online learning algorithms for trapezoidal data streams: OLSF [9]; 3) Online learning algorithm for capricious data streams: OLVF [11] and GLSC [20]. The parameter settings of these competing algorithms are as follows:

- **OGD(Online Gradient Descent)**: OGD is an iterative optimization algorithm designed for large-scale and dynamic datasets. Unlike traditional batch methods, OGD updates the model parameters using single samples or small subsets, making it well-suited for real-time learning scenarios. In OGD, we set α=0.01.
- **RDA(Regularized Dual Averaging)**: RDA is a dual-averaging method that combines stochastic gradient descent and regularization techniques to solve machine learning and optimization problems on large-scale data sets. This method provides an efficient optimization algorithm through smooth and stable parameter updates and regularization terms that control model complexity. In RDA, we set λ=0.01.
- **OLSF(Online Learning with Streaming Features)**: OLSF is an algorithm specifically developed to optimize the processing of trapezoidal data streams. These data streams are characterized by a continuous and monotonically increasing feature space. In experiments, we set λ=30, B=0.1, C=0.1 for OLSF as the suggested values in the original paper.
- **OLVF(Online Learning from Varying Features)**: OLVF expands trapezoidal data streams by transforming them into Varying Feature Spaces. This expansion is accomplished by training a feature-space classifier, indicating that the simultaneous inclusion of unobservable and novel features can offer valuable discriminant insights. In OLVF, we set B=0.5, C=0.01 as the suggested values.
- **GLSC(Generative Learning with Streaming Capricious)**: GLSC refines unobserved features by building a generative graphical model, where a latent matrix is maintained to establish correlations between observed features. In GLSC, we set $p = 0.5$.

Our new algorithms have three parameters: λ, C, and B. According to the parameter analysis in Section 6.5, the values of C and B are specified in {0.001, 0.01}, and {0.16, 0.32}, respectively. Besides, we set λ=30 as an experience value, which truncates w_t to give the model good sparsity.

6.1.4 Implementation details

To simulate trapezoidal data streams, we split the dataset into ten chunks, each carrying only 10% of the instances and a different number of features. For example, the first data block carries the top 10% of instances with the top 10% of features. The second data block carries the second 10% of the instances and another 10% of the features (20% of the features in total).

To simulate capricious data streams, we randomly remove features from each arriving instance x_t. The ratio of the removed features is denoted as α. For example, $\alpha = 0.5$ means that at most 50% of the features in x_t are randomly removed. In our experiments, the default value of α is 0.5.

Performance is measured in terms of F-measure. All experiments were repeated 10 times with random permutations on each data set, and all results reported here are the average values.

6.1.5 Computational device

All experimental results are conducted on a PC with AMD 5800X, 3.8 GHz CPU, and 16 GB memory.

6.2 OLCDS vs. fixed feature space data streams methods

In this section, we compare our proposed algorithm OLCDS and its variant OLCDS-I with traditional online learning methods OGD [26] and RDA [45]. Our proposed algorithms, OLCDS and OLCDS-I, are conducted in experiments on capricious data streams, while OGD and RDA are in the complete feature space data streams. All algorithms are implemented in PYTHON. The parameters involved in the comparison algorithm adopt the default values mentioned in the paper.

Table 3 presents the F-measure values of these competing algorithms. Based on the Friedman test, the p-value of the F-measure is 0.7083. Therefore, there is no significant difference in the predictive values among these competing algorithms in the case of F-measure. The critical difference (CD) value is 1.48.

From Table 3, we can indicate that:

- OLCDS *vs.* OGD: According to the statistical test, no significant difference exists between our new methods and OGD in the case of F-measure. On datasets credit-a, svmguide3, spambase, and ionosphere, both OLCDs and OLCDS-I perform better than OGD. Meanwhile, the performance of OGD is 2% higher than OLCDS on average. However, it is worth mentioning that OGD is applied on complete feature space while our new methods are conducted on capricious data streams. Our new algorithms, OLCDS and OLCDS-I, possess the distinct advantage of accommodating unaltered feature spaces and arbitrary changes in the feature space. As a result, our method offers a broader scope of applicability.
- OLCDS *vs.* RDA: There is no significant difference between our new methods and RDA in F-measure. On datasets wdbc, splice, credit-a, svmguide3, and spambase, both OLCDs and OLCDS-I perform better than OGD. RDA is about 2% higher than OLCDS on average. RDA is a dual-averaging method that combines stochastic gradient descent and regularization techniques to tackle machine learning and optimization problems on data streams. However, like OGD, RDA is limited to traditional online learning scenarios with fixed feature space.

In summary, the performance of our algorithms is comparable to traditional online learning methods. However, these competing algorithms can only handle data streams with fixed feature space, while our new algorithms can deal with data streams with arbitrary changes in feature space. Therefore, the applicability of our new algorithms is broader.

6.3 OLCDS vs. trapezoidal data streams methods

This section compares the proposed OLCDS and OLCDS-I with an online learning method for trapezoidal data streams (OLSF) [9]. All algorithms are implemented in PYTHON. The parameters involved in the comparison algorithm use the default values mentioned in the paper.

Table 4 gives these competing algorithms' average F-measure values in trapezoidal data streams. Based on the Friedman test, the p-value of the F-measure is 2.06E-06. Therefore, there is a significant difference in the predictive values of these competing algorithms. The critical difference (CD) value is 1.046. Figure 3 visually represents the statistical test conducted among these competing algorithms, highlighting the observed variations.

Table 3 F-measure of OLCDS vs. fixed feature space data streams methods

Data Set	OGD	RDA	OLCDS	OLCDS-I
wdbc	**0.937±0.010**	0.894±0.006	0.932±0.010	0.930±0.007
splice	**0.789±0.004**	0.731±0.010	0.780±0.008	0.770±0.006
credit-a	0.787±0.014	0.753±0.012	0.795±0.010	**0.800±0.012**
svmguide3	0.346±0.008	0.320±0.004	0.521±0.014	**0.524±0.010**
spambase	0.676±0.003	0.656±0.007	**0.847±0.005**	0.842±0.004
ionosphere	0.714±0.011	**0.787±0.013**	0.772±0.017	0.746±0.013
spect	0.839±0.010	**0.863±0.014**	0.669±0.013	0.663±0.018
libras	0.848±0.014	**0.901±0.011**	0.671±0.014	0.679±0.012
dermatology	0.940±0.013	**0.980±0.017**	0.827±0.019	0.843±0.018
arrhythmia	**0.907±0.014**	0.893±0.021	0.774±0.017	0.732±0.011
AVG.	**0.7784**	0.7777	0.7588	0.7529
AVG. RANKS	**2.2**	2.6	2.5	2.7

The best results are highlighted in bold face in the tables

Table 4 F-measure of OLCDS vs. trapezoidal data streams methods

Data Set	OLSF	OLCDS	OLCDS-I
wdbc	0.903±0.005	**0.914±0.008**	0.904±0.013
splice	0.643±0.004	**0.687±0.006**	0.637±0.009
credit-a	0.618±0.013	**0.642±0.012**	0.631±0.015
svmguide3	0.478±0.017	**0.511±0.019**	0.479±0.013
spambase	0.754±0.006	**0.787±0.004**	0.744±0.005
ionosphere	0.721±0.016	**0.778±0.013**	0.735±0.031
spect	0.635±0.015	0.638±0.034	**0.641±0.014**
libras	0.593±0.018	**0.667±0.012**	0.639±0.022
dermatology	0.741±0.012	**0.810±0.008**	0.808±0.011
arrhythmia	0.697±0.014	**0.748±0.011**	0.747±0.014
AVG.	0.6783	**0.7182**	0.6965
AVG. RANKS	2.8	**1.1**	2.1

The best results are highlighted in bold face in the tables

From Table 4 and Fig. 3, we can indicate that:

- OLCDS *vs*. OLSF: According to the statistical test, both OLCDS and OLCDS-I demonstrate significantly superior performance compared to OLSF in the case of F-measure. OLCDS consistently outperforms OLSF on all datasets, indicating its effectiveness in handling trapezoidal data streams. The key reason behind OLCDS's exceptional performance lies in its ability to retain the most essential weights and learn the most valuable information. By prioritizing the most informative features, OLCDS excels at capturing the essential characteristics of the data streams, enabling it to handle trapezoidal patterns effectively.
- OLCDS *vs*. OLCDS-I: Based on the experimental results, no significant difference exists between OLCDS and OLCDS-I on performance in F-measure. However, OLCDS performs better than OLCDS-I on nine of ten datasets. The reason behind this performance advantage lies in the initial stages of trapezoidal data streams. During this phase, the feature set consists of only 10% of the total features, resulting in limited information for model learning. The OLCDS algorithm adapts and learns from the expanding feature set over time by effectively leveraging the available information. This adaptive learning approach enables the model to enhance its understanding of the data and ultimately achieve superior performance compared to OLCDS-I.

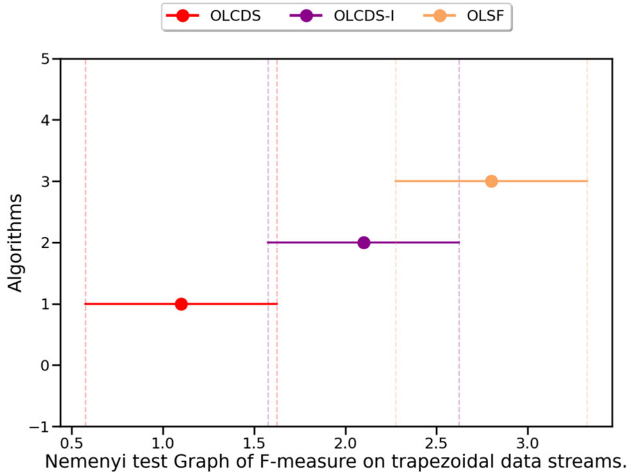


Fig. 3 The statistical test graph of OLCDS and OLCDS-I vs. trapezoidal data streams methods

In sum, by retaining the essential weights and learning the most valuable information, OLCDS significantly improves performance compared to OLSF for trapezoidal data streams.

6.4 OLCDS vs. Capricious data streams methods

In this section, we compare our proposed algorithms with OLVF [11] and GLSC [20] on capricious data streams. All algorithms are implemented in PYTHON. The parameters involved in the comparison algorithms use the default values mentioned in the papers.

Table 5 shows the average F-measure values of OLCDS, OLCDS-I, and the comparison algorithms OLVF and GLSC. The Friedman test was conducted to analyze the performance of competing algorithms on capricious data streams. The obtained p-value for the F-measure is 9.86E-07. Therefore, there is a significant difference between these competing algorithms. The value of the critical difference (CD) is 1.4819. Figure 4 visually presents the outcomes of the statistical test, emphasizing the observed variations among these competing algorithms.

From Table 5 and Fig. 4, we can indicate that:

- OLCDS *vs*. OLVF: Both algorithms perform better than OLVF on all these datasets. OLVF is primarily designed for processing data streams with regularly changing feature spaces and may struggle to handle data streams with arbitrary changes in feature space effectively. On the other hand, OLCDS excels in handling data streams with changing feature spaces and those with arbitrary changes in feature space.
- OLCDS *vs*. GLSC: According to the statistical test, OLCDS and OLCDS-I significantly outperform GLSC in the F-measure. OLCDS and OLCDS-I achieve higher performance on nine of ten datasets than GLSC. It is important to note that GLSC utilizes a graphical model for classifier training, which requires a more significant number of instances and features to achieve comparable performance to OLCDS. Additionally, GLSC requires more computing time to generate model classifiers.

Table 5 F-measure of OLCDS vs. capricious data streams methods

Data Set	OLVF	GLSC	OLCDS	OLCDS-I
wdbc	0.909±0.013	0.905±0.010	**0.932±0.010**	0.930±0.007
splice	0.748±0.007	0.752±0.006	**0.780±0.008**	0.770±0.006
credit-a	0.783±0.011	0.794±0.012	0.795±0.010	**0.800±0.012**
svmguide3	0.495±0.012	0.484±0.007	0.521±0.014	**0.524±0.010**
spambase	0.810±0.003	**0.851±0.004**	0.847±0.005	0.842±0.004
ionosphere	0.730±0.011	0.728±0.014	**0.772±0.017**	0.746±0.013
spect	0.651±0.018	0.634±0.013	0.669±0.027	**0.663±0.024**
libras	0.593±0.011	0.578±0.017	0.671±0.017	**0.679±0.015**
dermatology	0.760±0.010	0.742±0.013	0.827±0.016	**0.843±0.017**
arrhythmia	0.690±0.005	0.687±0.011	**0.774±0.016**	0.732±0.013
AVG.	0.7169	0.7155	**0.7588**	0.7529
AVG. RANKS	3.3	3.5	**1.5**	1.7

The best results are highlighted in bold face in the tables

- OLCDS *vs*. OLCDS-I: There is no significant difference between OLCDS and OLCDS-I on these datasets in the case of F-measure. Meanwhile, the average values of OLCDS and OLCDS-I are very close. The main difference between these two algorithms lines in the slack variables ξ in the quadratic objective function. Introducing these slack variables allows the algorithm to penalize more significant errors and facilitate larger step sizes during iterations, ultimately enhancing the classifier's performance.

In summary, OLCDS and OLCDS-I offer significant performance improvements over OLVF and GLSC when dealing with capricious data streams by preserving crucial weights and learning the most valuable information.

6.5 Parameter analysis

In this section, we investigate the parameter sensitivity of our algorithm using four data sets: wdbc, splice, credit-a, and svmguide3. The algorithm relies on two crucial parameters: the penalty cost parameter C and the proportion of selected features B.

We conduct two sets of experiments to analyze the impact of these parameters. In the first set, we keep B fixed at 1, and vary C from 10^{-4} to 10^{4}, as shown in Fig. 5. In the second set, we fix C at 1, and adjust B using values from the set $\{0.04, 0.08, 0.16, 0.32, 0.64\}$, as shown in Fig. 6. By systematically exploring the effects of different parameter values, we aim to understand how C and B influence the performance of our algorithm.

From Fig. 5, it is evident that our algorithm, OLCDS, and OLCDS-I, are sensitive to changes in the value of C from 10^{-4} to 10^{4}. In total, the performance of both two algorithms first increases and then declines. In the range of $[10^{-4}, 10^{-2}]$, the algorithms consistently are performing better and better and achieving optimal performance. However, as the value of C increases beyond this range, OLCDS and OLCDS-I gradually decline in performance. This sensitivity can be attributed to the fact that C determines the update step size τ. A smaller value of C enables a smoother change in the update step size, preventing abrupt changes during optimization.

On the other hand, Fig. 6 depicts the performance of the F-measure of our proposed OLCDS and OLCDS-I algorithms under different values of B. From these two figures, we can observe that the performance of our algorithms remains largely unaffected by variations in the size of B. This robustness demonstrates that our algorithm can handle different

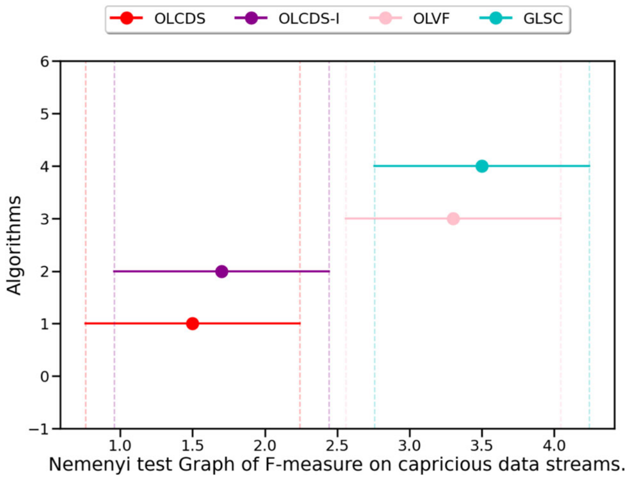
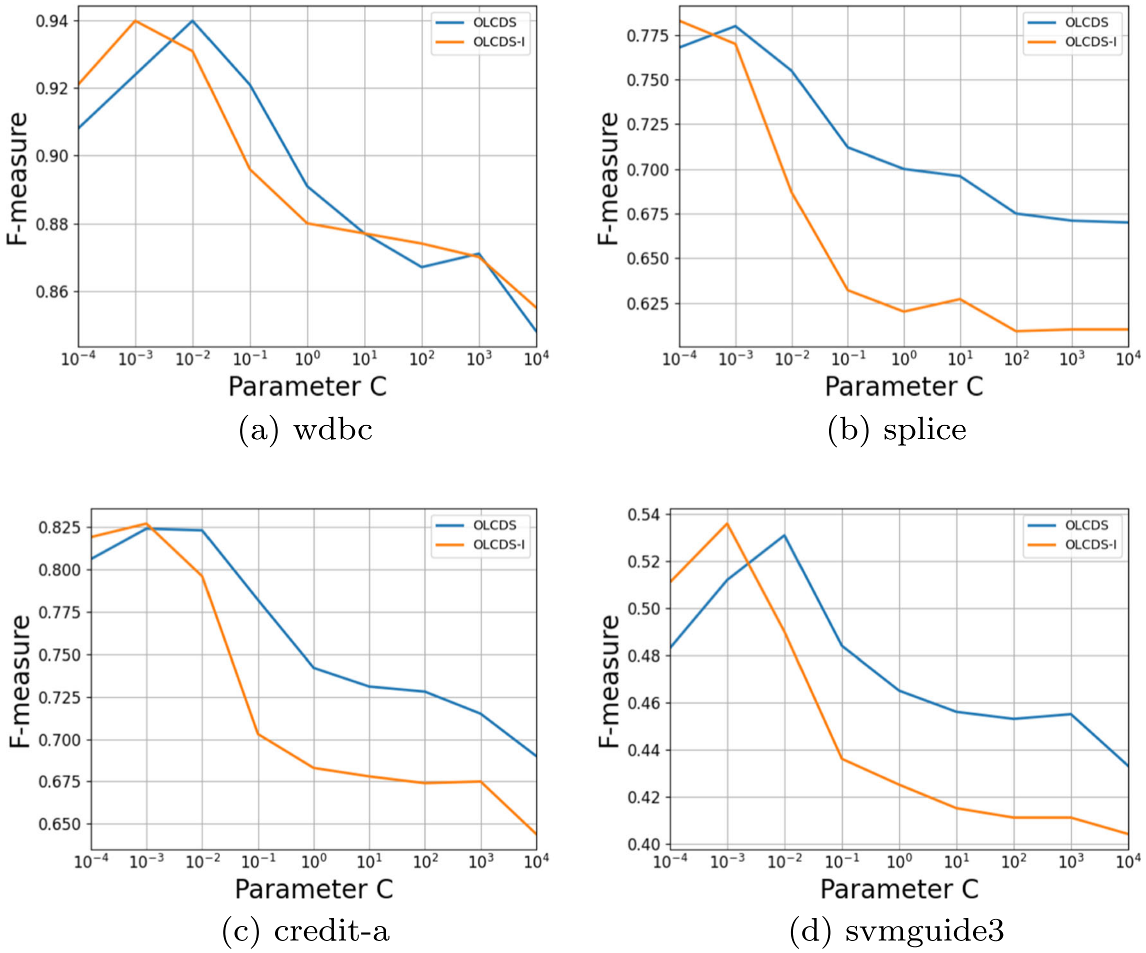


Fig. 4 The statistical test graph of OLCDS and OLCDS-I vs. capricious data streams methods



(a) wdbc

(b) splice

(c) credit-a

(d) svmguide3

Fig. 5 F-measure of OLCDS and OLCDS-I varies with the penalty parameter C

feature subspaces effectively, highlighting its versatility and reliability.

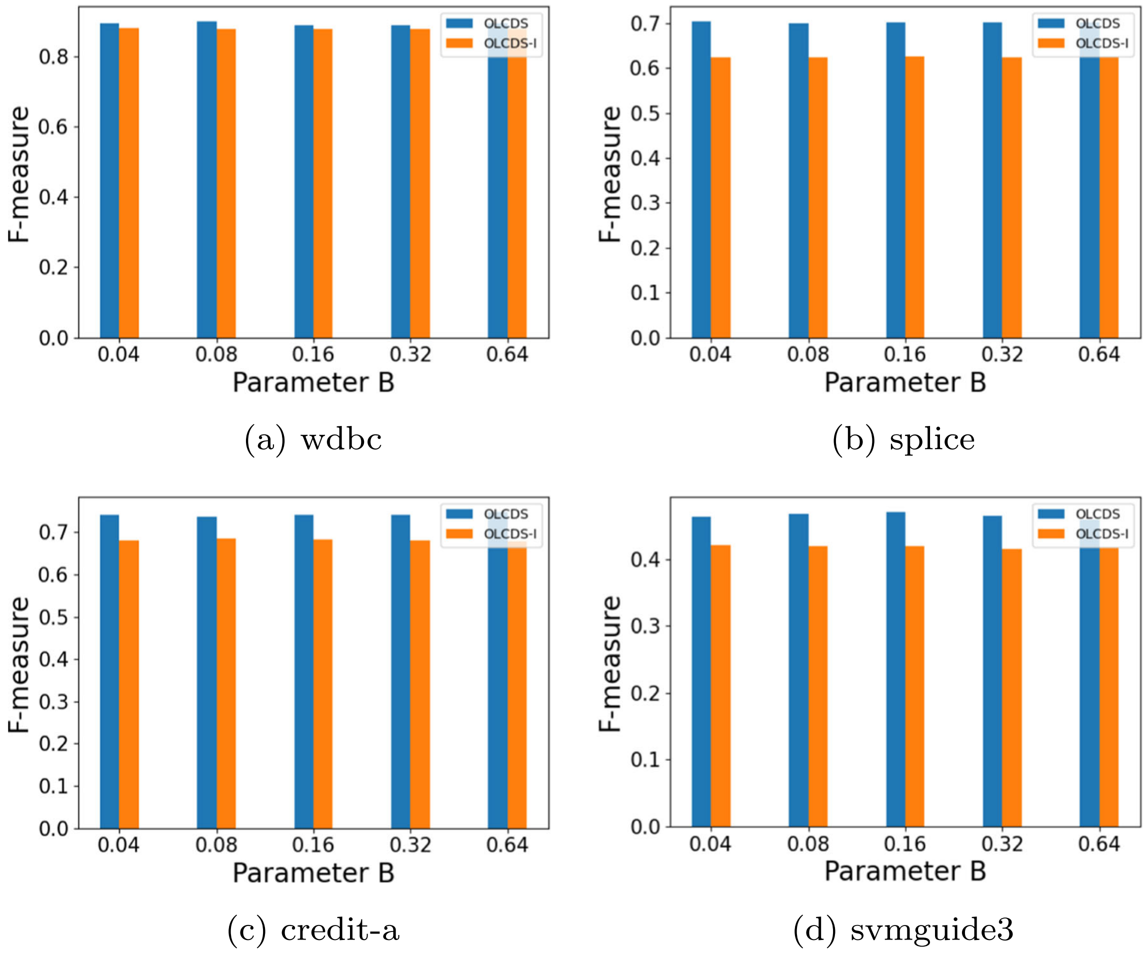
In sum, we set the parameter values of C and B in $\{0.001, 0.01\}$ and $\{0.16, 0.32\}$ respectively in our experiments.

7 Discussion

Based on the above analysis of theoretical and experimental results, our new method has the following advantages:

- Effectiveness with superior online learning performance. Experimental results from Tables 3, 4, and 5 and the statistical test Figs. 3 and 4 indicate the superiority of our new methods in F-measure on these datasets compared with other competing algorithms. Meanwhile, Lemma 1 theoretically proves the effectiveness of our proposed methods.
- Efficiency with low time and space complexity. Since our new methods avoid generating missing features, they are more efficient than existing online learning algorithms. Besides, after identifying higher uncertainty features, we formulate the constrained online optimization problem based on the shared and new feature space between adjacent instances. In other words, our new methods do not need to cache all instance data. The detailed time and space complexity analysis indicates the efficiency of our new methods.
- Adaptability for practical applications. Traditional online learning methods assume the feature space of the streaming instance remains fixed during learning. However, some features may be missing in practical applications for some reasons. Meanwhile, assuming that the feature space changes regularly in practical applications is also unreasonable. Therefore, since our new methods eliminate the need for feature space assumptions, they are more adaptive to practical application requirements.

Of course, our new methods also have some limitations. Although the empirical risk minimization principle is used to solve the problem of missing or adding feature spaces in capricious data streams, OLCDS and its variant OLCDS-I algorithm are limited to learning linear decision boundaries,



(a) wdbc

(b) splice

(c) credit-a

(d) svmguide3

Fig. 6 F-measure of OLCDS and OLCDS-I varies with the proportion of selected features *B*

which may not perform well on nonlinearly separable data. In addition, trapezoidal data streams and arbitrary data streams represent only two examples of data streams with dynamic feature spaces. In practical problems, datasets may involve instances with completely different feature sets, i.e., mixed feature sets, or their feature spaces may shrink over time. Therefore, our future work will explore new methods to adapt to irregularly changing feature spaces in these cases.

8 Conclusion

In this paper, we address the challenge of online learning from capricious data streams, where the feature space of streaming instances undergoes arbitrary changes. To tackle this problem, we propose a novel online learning algorithm named OLCDS and its variant OLCDS-I for capricious data streams. By comparing the feature space of adjacent instances, we get the shared and new feature space at each timestamp. Then, we identify the most informative features and formulate the optimization problem based on shared and new feature space. Our new method eliminates the need for feature space assumptions and avoids generating missing features. To evaluate the effectiveness of our approach, we conduct extensive experiments on ten real-world datasets and compare our new method with three different types of online learning algorithms. Experiment results and statistical tests validate the superiority of our new algorithms.

Acknowledgements This work is supported in part by the National Natural Science Foundation of China under grants (62376001, 62376002, 62206004), the Natural Science Foundation of Anhui Province of China under grants (2308085MF215, 2208085QF199).

Author Contributions **Peng Zhou**: Conceptualization, Methodology, Writing - review & editing, Funding acquisition. **Shuai Zhang**: Methodology, Software, Validation, Investigation, Writing-original draft. **Lin Mu**:Formal analysis, Funding acquisition. **Yuanting Yan** Formal analysis, Funding acquisition.

Availability of data and materials The datasets and code used during this study are available upon reasonable request to the authors.

Declarations

Ethics approval Not applicable

Consent to participate Not applicable

Consent for publication Not applicable

Competing interests The authors declare that they have no competing interests.

References

1. Bhatia K, Sridharan K (2020) Online learning with dynamics: A minimax perspective. Adv Neural Inf Process Syst 33:15020–15030
2. Zhao P, Wang D, Wu P, Hoi SC (2020) A unified framework for sparse online learning. ACM Trans Knowl Discovery Data (TKDD) 14(5):1–20
3. Yu H, Sun X, Wang J (2019) Ensemble os-elm based on combination weight for data stream classification. Appl Intell 49:2382–2390
4. De Lange M, Tuytelaars T (2021) Continual prototype evolution: Learning online from non-stationary data streams. In: Proceedings of the IEEE/CVF international conference on computer vision, pages 8250–8259
5. Vidhya M, Aji S (2022) Parallelized extreme learning machine for online data classification. Appl Intell 52(12):14164–14177
6. Fu X, Seo E, Clarke J, Hutchinson RA (2019) Link prediction under imperfect detection: Collaborative filtering for ecological networks. IEEE Trans Knowl Data Eng 33(8):3117–3128
7. Phadke A, Kulkarni M, Bhawalkar P, Bhattad R (2019) A review of machine learning methodologies for network intrusion detection. In: 2019 3rd International conference on computing methodologies and communication (ICCMC), IEEE, pages 272–275
8. Ullo SL, Sinha GR (2020) Advances in smart environment monitoring systems using iot and sensors. Sensors 20(11):3113
9. Zhang Q, Zhang P, Long G, Ding W, Zhang C, Wu X (2016) Online learning from trapezoidal data streams. IEEE Trans Knowl Data Eng 28(10):2709–2723
10. Hou B-J, Zhang L, Zhou Z-H (2017) Learning with feature evolvable streams. Adv Neural Inf Process Syst 30
11. Beyazit E, Alagurajah J, Wu X (2019) Online learning from data streams with varying feature spaces. Proceedings of the AAAI conference on artificial intelligence 33:3232–3239
12. Zhang Z-Y, Zhao P, Jiang Y, Zhou Z-H (2020) Learning with feature and distribution evolvable streams. In: International conference on machine learning, PMLR, pp 11317–11327
13. Hou B-J, Zhang L, Zhou Z-H (2021) Prediction with unpredictable feature evolution. IEEE Trans Neural Netw Learn Syst 33(10):5706–5715
14. You D, Xiao J, Wang Y, Yan H, Wu D, Chen Z, Shen L, Wu X (2023) Online learning from incomplete and imbalanced data streams. IEEE Trans Knowl Data Eng
15. Zhang D, Jin M, Cao P (2020) St-metadiagnosis: Meta learning with spatial transform for rare skin disease diagnosis. In: 2020 IEEE International conference on bioinformatics and biomedicine (BIBM), IEEE, pages 2153–2160
16. Zhou Y, Ren H, Li Z, Wu N, Al-Ahmari AM (2021) Anomaly detection via a combination model in time series data. Appl Intell 51:4874–4887
17. Lu J, Liu A, Dong F, Gu F, Gama J, Zhang G (2018) Learning under concept drift: A review. IEEE Trans Knowl Data Eng 31(12):2346–2363
18. Agrahari S, Singh AK (2022) Concept drift detection in data stream mining: A literature review. J King Saud University-Comput Inf Sci 34(10):9523–9540
19. Friedrich B, Sawabe T, Hein A (2023) Unsupervised statistical concept drift detection for behaviour abnormality detection. Appl Intell 53(3):2527–2537
20. He Y, Wu B, Wu D, Beyazit E, Chen S, Wu X (2020) Toward mining capricious data streams: A generative approach. IEEE Trans Neural Netw Learn Syst 32(3):1228–1240
21. Du H, Zhang Y, Gang K, Zhang L, Chen Y-C (2021) Online ensemble learning algorithm for imbalanced data stream. Appl Soft Comput 107:107378
22. Sun Y, Tang K, Minku LL, Wang S, Yao X (2016) Online ensemble learning of data streams with gradually evolved classes. IEEE Trans Knowl Data Eng 28(6):1532–1545
23. Wang B, Pineau J (2016) Online bagging and boosting for imbalanced data streams. IEEE Trans Knowl Data Eng 28(12):3353–3366
24. Jamshidi M, Yahya SI, Roshani S, Chaudhary MA, Ghadi YY, Roshani S (2023) A fast surrogate model-based algorithm using multilayer perceptron neural networks for microwave circuit design. Algorithms 16(7):324
25. Xie X, Pu Y-F, Wang J (2023) A fractional gradient descent algorithm robust to the initial weights of multilayer perceptron. Neural Netw 158:154–170
26. Zinkevich M (2003) Online convex programming and generalized infinitesimal gradient ascent. In: Proceedings of the 20th international conference on machine learning (icml-03), pages 928–936
27. Yao Z, Gholami A, Shen S, Mustafa M, Keutzer K, Mahoney M (2021) Adahessian: An adaptive second order optimizer for machine learning. Proceedings of the AAAI conference on artificial intelligence 35:10665–10673
28. Crammer K, Lee D (2010) Learning via gaussian herding. Adv Neural Inf Process Syst 23
29. Crammer K, Dredze M, Kulesza A (2009) Multi-class confidence weighted algorithms. In: Proceedings of the 2009 conference on empirical methods in natural language processing, pages 496–504
30. Orabona F, Crammer K (2010) New adaptive algorithms for online classification. Adv Neural Inf Process Syst 23
31. Crammer K, Kulesza A, Dredze M (2009) Adaptive regularization of weight vectors. Adv Neural Inf Process Syst 22
32. Zhou H, Matsushima S (2023) Online learning under capricious feature data streams. In: 37th Annual conference of the japanese society for artificial intelligence, pages 2D4GS201–2D4GS201
33. Gu S, Qian Y, Hou C (2022) Incremental feature spaces learning with label scarcity. ACM Trans Knowl Discovery Data (TKDD) 16(6):1–26
34. He Y, Wu B, Wu D, Beyazit E, Chen S, Wu X (2019) Online learning from capricious data streams: a generative approach. In: Proceedings of the 28th international joint conference on artificial intelligence, pages 2491–2497
35. Liu Y, Fan X, Li W, Gao Y (2022) Online passive-aggressive active learning for trapezoidal data streams. IEEE Trans Neural Netw Learn Syst
36. Wang S, Fan Y, Jin S, Takyi-Aninakwa P, Fernandez C (2023) Improved anti-noise adaptive long short-term memory neural network modeling for the robust remaining useful life prediction of lithium-ion batteries. Reliab Eng Syst Saf 230:108920
37. Wang S, Wu F, Takyi-Aninakwa P, Fernandez C, Stroe D-I, Huang Q (2023) Improved singular filtering-gaussian process regression-long short-term memory model for whole-life-cycle remaining capacity estimation of lithium-ion batteries adaptive to fast aging and multi-current variations. Energy 284:128677
38. Ozyildirim BM, Kiran M (2021) Levenberg-marquardt multi-classification using hinge loss function. Neural Netw 143:564–571

39. Singla M, Ghosh D, Shukla KK, Pedrycz W (2020) Robust twin support vector regression based on rescaled hinge loss. Pattern Recognit 105:107395
40. George E, Murray M, Swartworth W, Needell D (2024) Training shallow relu networks on noisy data using hinge loss: when do we overfit and is it benign? Adv Neural Inf Process Syst 36
41. Frei S, Vardi G, Bartlett P, Srebro N (2023) Benign overfitting in linear classifiers and leaky relu networks from kkt conditions for margin maximization. In: The Thirty Sixth annual conference on learning theory, PMLR, pp 3173–3228
42. Huang S-J, Jin R, Zhou Z-H (2010) Active learning by querying informative and representative examples. Adv Neural Inf Process Syst 23
43. Huang S-J, Xu M, Xie M-K, Sugiyama M, Niu G, Chen S (2018) Active feature acquisition with supervised matrix completion. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, pp 1571–1579
44. Rayner JCW, Livingston G Jr (2023) Relating the friedman test adjusted for ties, the cochran-mantel-haenszel mean score test and the anova f test. Commun Statistics-Theory Methods 52(12):4369–4378
45. Xiao L (2009) Dual averaging method for regularized stochastic learning and online optimization. Adv Neural Inf Process Syst 22

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Peng Zhou received the Ph.D. degree from the Hefei University of Technology, Hefei, China, in 2018. He is currently an associate professor in the School of Computer Science and Technology at Anhui University. His research interests include machine learning, data mining and stream learning.

Shuai Zhang received the M.Sc. degree from Anhui University, Hefei, China, in 2024. He is currently employed in the School of Information Engineering, Xuzhou Vocational College of Industrial Technology, Xuzhou. China His research interests include online machine learning and data stream mining.

Lin Mu received the PhD degree in the University of Science and Technology of China, in 2021. He is a lecturer with the School of Computer Science and Technology, Anhui University, Hefei, China. His research interests include information extraction, natural language processing, and large language models (LLM).

Yuanting Yan received his PhD degree in computer science from Anhui University, China, in 2016. He is currently an associate professor in the School of Computer Science and Technology at Anhui University. His current research interests include machine learning, granular computing and bioinformatics.