Toward Mining Capricious Data Streams: A Generative Approach

Yi He, Baijun Wu, Di Wu, Ege Beyazit, Sheng Chen, and Xindong Wu, *Fellow, IEEE*

***Abstract*— Learning with streaming data has received extensive attention during the past few years. Existing approaches assume that the feature space is fixed or changes by following explicit regularities, limiting their applicability in real-time applications. For example, in a smart healthcare platform, the feature space of the patient data varies when different medical service providers use nonidentical feature sets to describe the patients' symptoms. To fill the gap, we in this article propose a novel learning paradigm, namely, Generative Learning With Streaming Capricious (GLSC) data, which does not make any assumption on the feature space dynamics. In other words, GLSC handles the data streams with a varying feature space, where each arriving data instance can arbitrarily carry new features and/or stop carrying partial old features. Specifically, GLSC trains a learner on a universal feature space that establishes relationships between old and new features, so that the patterns learned in the old feature space can be used in the new feature space. The universal feature space is constructed by leveraging the relatednesses among features. We propose a generative graphical model to model the construction process, and show that learning from the universal feature space can effectively improve the performance with theoretical guarantees. The experimental results demonstrate that GLSC achieves conspicuous performance on both synthetic and real data sets.**

***Index Terms*— Biconvex optimization, capricious data streams, graphical model, online learning.**

NOMENCLATURE

t	$t \in [1, 2, \ldots, T]$, current iteration number.
T	$T \in N^+$, total number of data instances.
$\mathbb{R}^{d_t}$	Observable feature space of data instance $\mathbf{x}_t$.
$\mathbb{R}^{\lvert\mathcal{U}_t\rvert}$	Hypothesis space of learner $\mathbf{w}_t$.
$\mathcal{U}_t$	Universal feature space at the tth iteration.
ψ	Reconstructive mapping: $\mathbb{R}^{d_t} \mapsto \mathcal{U}_t$.
$\mathcal{G}$	$\mathcal{G} = (\mathcal{V}, \mathcal{E})$, generative graph that embeds ψ.
$\mathcal{V}$	Vertices in $\mathcal{G}$, each of which represents a feature in $\mathcal{U}_t$.
$\mathcal{E}$	Edges in $\mathcal{G}$, each of which represents a relatedness between two vertices.
$\mathbf{G}_{i,j}$	Out-edge from vertex i to j.
Φ_i	$\Phi_i = [\mathbf{G}_{i,1}, \mathbf{G}_{i,2}, \ldots, \mathbf{G}_{i,\lvert\mathcal{U}_t\rvert}]^\top \in \mathbb{R}^{\lvert\mathcal{U}_t\rvert}$, vertex approximator corresponding to vertex i.
$\mathbf{G}$	$\mathbf{G} = [\Phi_1, \Phi_2, \ldots, \Phi_{\lvert\mathcal{U}_t\rvert}]^\top \in \mathbb{R}^{\lvert\mathcal{U}_t\rvert \times \lvert\mathcal{U}_t\rvert}$, concrete matrix representation of $\mathcal{G}$.
$\mathbf{L}$	Laplacian matrix corresponding to $\mathbf{G}$.
$\mathbf{I}_t$	$\mathbf{I}_t \in \mathbb{R}^{d_t \times \lvert\mathcal{U}_t\rvert}$, indicator matrix indicating which features in $\mathcal{U}_t$ are carried by $\mathbf{x}_t$.
$\Pi_{\mathbb{R}^n}(\cdot)$	Orthogonal projection operator. For any vector $\mathbf{b}$, its orthogonal projection onto an $\mathbb{R}^n$ space is $\Pi_{\mathbb{R}^n}(\mathbf{b}) = \arg\min_{\mathbf{a}\in\mathbb{R}^n} \lVert\mathbf{a} - \mathbf{b}\rVert_2$.
$\mathbf{Gr}$	$\mathbf{Gr} = \mathbf{I}_t\mathbf{G}$, concrete matrix representation of ψ.
$\mathbf{x}_t$	$\mathbf{x}_t = [\mathrm{f}_1, \mathrm{f}_2, \ldots, \mathrm{f}_{d_t}]^\top \in \mathbb{R}^{d_t}$, arriving data instance carrying observable features.
$\psi(\mathbf{x}_t)$	$\psi(\mathbf{x}_t) = [\tilde{\mathrm{f}}_1, \tilde{\mathrm{f}}_2, \ldots, \tilde{\mathrm{f}}_{\lvert\mathcal{U}_t\rvert}]^\top \in \mathcal{U}_t$, reconstructed data instance carrying universal features.
$\tilde{\mathbf{x}}_t$	$\tilde{\mathbf{x}}_t = [\tilde{\mathrm{f}}_{d_{t+1}}, \ldots, \tilde{\mathrm{f}}_{\lvert\mathcal{U}_t\rvert}]^\top \in \mathcal{U}_t \setminus \mathbb{R}^{d_t}$, reconstructed data instance carrying unobservable features.
$\mathbf{u}_t$	$\mathbf{u}_t = [\mathbf{x}_t, \tilde{\mathbf{x}}_t]^\top \in \mathbb{R}^{\lvert\mathcal{U}_t\rvert}$, recovery of data instance $\mathbf{x}_t$ in the universal feature space.
$\mathbf{w}_t$	$\mathbf{w}_t = [w_1, w_2, \ldots, w_{\lvert\mathcal{U}_t\rvert}]^\top \in \mathbb{R}^{\lvert\mathcal{U}_t\rvert}$, learner built at tth iteration.
$\bar{\mathbf{w}}_t$	$\bar{\mathbf{w}}_t = [w_1, w_2, \ldots, w_{d_t}]^\top \in \mathbb{R}^{d_t}$, weight coefficients in learner $\mathbf{w}_t$ corresponding to $\mathbf{x}_t$.
$\tilde{\mathbf{w}}_t$	$\tilde{\mathbf{w}}_t = [w_{d_{t+1}}, \ldots, w_{\lvert\mathcal{U}_t\rvert}]^\top \in \mathbb{R}^{\lvert\mathcal{U}_t\rvert - d_t}$, weight coefficients in learner $\mathbf{w}_t$ corresponding to $\tilde{\mathbf{x}}_t$.
$\langle\bar{\mathbf{w}}_t, \mathbf{x}_t\rangle$	Confidence degree of predicting $\mathbf{x}_t$ with respect to $\bar{\mathbf{w}}_t$.
$\langle\tilde{\mathbf{w}}_t, \tilde{\mathbf{x}}_t\rangle$	Confidence degree of predicting $\tilde{\mathbf{x}}_t$ with respect to $\tilde{\mathbf{w}}_t$.
y_t	Groundtruth label of data instance $\mathbf{x}_t$.
$\hat{y}_t$	Predicted label of data instance $\mathbf{x}_t$.
$\ell(y_t, \hat{y}_t)$	Instantaneous loss reflecting the discrepancy between the prediction and the groundtruth.
L_T^{obs}	$L_T^{\mathrm{obs}} = \sum_{t=1}^{T} \ell(y_t, \langle\bar{\mathbf{w}}_t, \mathbf{x}_t\rangle)$, cumulative loss suffered by making predictions on $\mathbf{x}_t$ over T iterations.

Manuscript received July 19, 2019; revised January 3, 2020; accepted March 3, 2020. This work was supported in part by the National Key Research and Development Program of China under Grant 2016YFB1000901, in part by the National Natural Science Foundation of China (NSFC) under Grant 91746209, and in part by the U.S. National Science Foundation (NSF) under Grant IIS-1652107, Grant IIS-1763620, and Grant CCF-1750886. *(Corresponding author: Xindong Wu.)*

Yi He, Baijun Wu, Ege Beyazit, and Sheng Chen are with the Center for Advanced Computer Studies, School of Computing and Informatics, University of Louisiana at Lafayette, Lafayette, LA 70503 USA (e-mail: yi.he1@louisiana.edu; bj.wu@louisiana.edu; exb6143@louisiana.edu; chen@louisiana.edu).

Di Wu is with the Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 376348, China (e-mail: wudi@cigit.ac.cn).

Xindong Wu is with the Key Laboratory of Knowledge Engineering with Big Data, Ministry of Education, Hefei University of Technology, Hefei 230009, China, and also with the Mininglamp Academy of Sciences, Mininglamp Technology, Beijing 100084, China (e-mail: xwu@hfut.edu.cn).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2020.2981386

L_T^{rec} $L_T^{\text{rec}} = \sum_{t=1}^{T} \ell(y_t, \langle \tilde{\mathbf{w}}_t, \tilde{\mathbf{x}}_t \rangle)$, cumulative loss suffered by making predictions on $\tilde{\mathbf{x}}_t$ over T iterations.

I. Introduction

TO DATE, many online learning approaches have been developed to handle streaming data [1]–[3]. Most of them assume that each data stream has a fixed feature space. Only a few recent studies have explored to learn from a dynamic feature space, yet they all make strong assumptions on the feature space dynamics, such as monotonically increasing, where the later data instances should include increasingly more features [4], or batchly evolving, where a few consecutive data instances must include all possible features from the feature space [5]. Unfortunately, these assumptions do not always hold in real applications. For example, in a smart healthcare platform [6]–[8], features describing the symptoms of patients can vary across the IoT devices (thermometers, pulse monitors, respiratory sensors, etc.) and medical service providers (hospitals, labs, insurance companies, etc.). This means that the patient data are streaming with an arbitrarily varying feature space. We refer to such data streams as capricious data streams.

At first glance, one may think to adapt existing algorithms, such as online convex optimization (OCO) [1], for handling capricious data streams. Fig. 1 depicts such a learning paradigm. The learner is trained on an observable feature space which only comprises the features carried by an arriving instance at the tth iteration. Therefore, this approach does not work well and is limited in two aspects. First, although the new features (e.g., features 2 and 3 at the second iteration in Fig. 1) enlarge the dimension of the learner's hypothesis set, they may not be described by a sufficient number of instances, leading to the curse of dimensionality [9]. Second, when the features that have been observed by the learner become unavailable in latter iterations, the learned patterns regarding these features are ignored. As a result, the learner does not exert the full power to achieve the best prediction performance.

To overcome these limitations, we in this article propose the Generative Learning With Streaming Capricious (GLSC) data algorithm by training a learner based on a universal feature space that includes the features appeared at each iteration. Introducing a universal feature space provides several advantages over an observable feature space. In the training phase, since the newly appeared features at the tth iteration are maintained in the universal feature space in all following iterations, the learner could benefit from being continuously provided information from them. In the predicting phase, the universal feature space is wider than the observable one, conveying additional information, so that the learner's prediction performance is improved.

The question, then, is how to obtain the universal feature space. On capricious data streams, an instance may not carry some features that are already included in the universal feature space. Taking the second iteration in Fig. 1 as an example, the universal feature 1 is missing in the arriving instance. We call such missing features unobservable features, and the problem of obtaining the universal feature space is thus recast as reconstructing them.

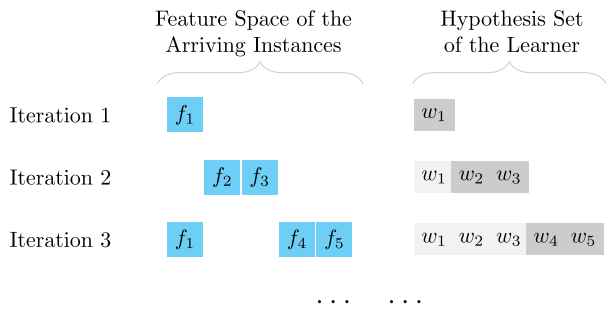


Fig. 1. Naïve way of learning from a varying feature space. The weight coefficient w_i (marked in dark gray) of the corresponding new feature is initialized as zero.

We build upon a key insight that enables GLSC to infer unobservable features from observable ones: in practice there exist relatednesses among features [10]–[12]. Specifically, GLSC uses a graph to capture feature relatednesses. Each vertex in the graph denotes a feature in the universal feature space, and all out-edges of a vertex together represent the relationship between the corresponding feature and the others. We embed the graph learning process into the online learning task. The effectiveness of GLSC is validated in three scenarios: trapezoidal data streams [4], feature evolvable streams [5], and capricious data streams.

Specific contributions of this article are summarized as follows.

1) This is the first work to learn with capricious data streams where data come with an arbitrarily varying feature space. We want to emphasize that our learning task does not make any assumption on the feature space dynamics, which is different from existing studies.
2) We introduce a generative graphical model, which takes the observable feature space as the input and outputs a universal feature space. We analyze the performance bound of GLSC and prove that the obtained universal feature space can effectively improve the learning performance.
3) Extensive experiments on both synthetic and real-world data sets demonstrate the superiority of GLSC.

The remainder of this article is organized as follows. Section II discusses related work. Section III introduces preliminaries. Section IV presents the building blocks of GLSC. Section V analyzes the performance bound of our approach. Section VI reports experimental results. We conclude the work in Section VII. Due to the page limitation, we put the detailed derivations and proofs, time complexity analysis, and complete experimental results in the Supplementary Material.[1]

II. Related Work

In this article, we focus on learning data streams from a varying feature space, which is closely related to the following literatures. It is worth pointing out that though concept-drift happens in streaming data where the underlying data distribution keeps changing [13], the number of features carried by each instance is fixed in concept-drift, which is different from our learning task.

[1] bit.ly/3aAzDJF

A. Online Learning

The works that are most related to our learning task include online learning from a fixed feature space [1], [3], from an incremental feature space [4], [14], and from an evolvable feature space [5], [15]. Those approaches tackling the streaming data problem under different settings, however, rely on the assumptions that the feature space is fixed or changes by following explicit regularities. Thus, they cannot handle an arbitrarily varying feature space. A recent work in [16] lifted these assumptions, but it makes an implicit assumption that there are overlapping features among arriving instances. Our work makes no such assumption and thus has its technical challenges and solutions.

B. Feature Reconstruction Learning

The key idea of GLSC is to learn reconstructive mapping by exploiting feature relatednesses. As such, our work is also related to the feature space reconstruction-based approaches. Specifically, Boutsidis *et al.* [17], Farahat *et al.* [18], and Tang and Liu [19] use the capability of features to approximate original data as a novel criterion for unsupervised feature selection. They assume that the most informative subset of features can reconstruct the whole feature space with few reconstruction errors. Similarly, Huang *et al.* [20] considers that, in completing a highly sparse matrix, one can actively query the missing features that have the strongest capability to recover the other features. Moreover, Mairal *et al.* [21] and Ruvolo and Eaton [22] propose to learn sparse representations of data streams via reconstructing original features from extracted latent features. However, these methods mainly focus on feature selection and extraction, and to the best of our knowledge, none of them consider the varying of feature space during the learning process.

C. Graphical Models

Our proposed graphical model can also be viewed as a type of Markov random field and is thus related to other graphical models such as the Boltzmann machines [23]–[26]. At the structural level, our graphical model and the Boltzmann machine both have visible (observable) and hidden (unobservable) vertices. The main difference is how vertex visibility is determined. In the Boltzmann machine, hidden vertices are predefined and fixed and are independent of the data. In our graph, however, the status of vertices are determined by the arriving data in a stochastic manner. The vertices corresponding to features in the arriving data turn to observable and all other vertices remain unobservable. At the algorithm level, training a Boltzmann machine in general does not entail label information, which is more similar to feature extraction or dimension reduction rather than supervised learning. To see this, we can deem hidden vertices as latent features. Instead, our graphical model is trained jointly with the learner, serving directly for the purpose of supervised learning, and the label information, in turn, contributes to a better-trained graph.

III. Preliminaries

In this article, we focus on binary classification. Multiclass problems could be decomposed to multiple binary classification subproblems, using One-Versus-Rest [27] or One-Versus-One [28] strategies.

A. Learning Task Setup

Let $\{(\mathbf{x}_t, y_t)|t = 1, 2, \ldots, T\}$ denote a sequence of arriving data instances with labels, where $\mathbf{x}_t = [\mathrm{f}_1, \mathrm{f}_2, \ldots, \mathrm{f}_{d_t}]^\top \in \mathbb{R}^{d_t}$ is a d_t-dimensional vector and $y_t \in \{-1, +1\}$ represents the class label. At the tth iteration, the learner observes the instance $\mathbf{x}_t$ and then returns its prediction. The true label y_t is revealed thereafter, and the learner suffers an instantaneous loss reflecting the discrepancy between the prediction and the groundtruth.

We define feature space as a set of features. Let $\mathcal{U}_t = \{\mathbb{R}^{d_1} \cup \mathbb{R}^{d_2} \cup \cdots \cup \mathbb{R}^{d_t}\}$ denote the universal feature space at the tth iteration where the features of $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_t$ are included. Note that $|\mathcal{U}_t| \leq d_1 + d_2 + \ldots + d_t$ as we treat the feature shared by multiple instances as a single feature. For example, in Fig. 1, $\mathcal{U}_1 = \{f_1\}$, $\mathcal{U}_2 = \{f_1, f_2, f_3\}$, and $\mathcal{U}_3 = \{f_1, f_2, f_3, f_4, f_5\}$.

For simplicity, we denote a linear classifier by $\mathbf{w}_t$, a vector of weight coefficients. If new features appear in $\mathbf{x}_t$, their corresponding weight coefficients in $\mathbf{w}_t$ can be initialized as small scalars (or zeros). As a result, the dimension of $\mathbf{w}_t$ matches that of $\mathcal{U}_t$, namely, $\mathbf{w}_t \in \mathbb{R}^{|\mathcal{U}_t|}$.

B. Generative Graphical Model

A generative graph is to embed a reconstructive mapping $\psi\colon \mathbb{R}^{d_t} \mapsto \mathcal{U}_t$. Let $\mathcal{G}$ denote the graph whose vertices represent the features in $\mathcal{U}_t$. The weight of each edge in $\mathcal{G}$ encodes a feature-wise relatedness. In this article, we have $\mathbf{G}_{i,j} > 0$ if feature i and feature j are related, and $\mathbf{G}_{i,j} = 0$ otherwise.

We define vertex approximator Φ_i as a vector containing the weights of all out-edges of a vertex i. The adjacency matrix of $\mathcal{G}$ can thus be viewed as a matrix whose column vectors are the vertex approximators, namely, $\mathbf{G} = [\Phi_1, \ldots, \Phi_{|\mathcal{U}_t|}]^\top \in \mathbb{R}^{|\mathcal{U}_t| \times |\mathcal{U}_t|}$. Viewing $\mathbf{G}$ as a gathering of vertex approximators instead of an adjacency matrix facilitates the understanding of the later derivations.

We define the desired reconstruction of $\mathbf{x}_t$ in the universal feature space as $\mathbf{u}_t = [\mathrm{f}_1, \ldots, \mathrm{f}_{d_t}, \tilde{\mathrm{f}}_{d_{t+1}}, \ldots, \tilde{\mathrm{f}}_{|\mathcal{U}_t|}]^\top \in \mathbb{R}^{|\mathcal{U}_t|}$, where f_i and $\tilde{\mathrm{f}}_j$ represent an original observable feature and a reconstructed unobservable feature, respectively. We infer $\mathbf{u}_t$ by maximizing a log-likelihood function

$$\mathcal{Q} = \sum_{i=1}^{d_t} \log \mathbb{P}(\mathbf{u}_t|\mathrm{f}_i, \Phi_i). \tag{1}$$

The features in $\mathcal{U}_t$ are inferred by given a vertex i and the corresponding Φ_i as follows:

$$\mathbb{P}(\mathbf{u}_t|\mathrm{f}_i, \Phi_i) = \prod_{j=1}^{|\mathcal{U}_t|} \mathbb{P}(u_j|\mathrm{f}_i, \Phi_i) \tag{2}$$

where u_j denotes the jth universal feature in $\mathbf{u}_t$. Note, although universal features are inferred independently in (2),

the values of the inferred universal features are indeed related, and are determined by values of the observable features (e.g., f_i) and the vertex approximators (e.g., Φ_i). When a different f_i is observed, Φ_i will be updated correspondingly, resulting in different values of the inferred universal features. In this sense, the relatednesses among universal features are captured via observable features that are used in the reconstruction.

For the sake of simplicity of mathematical expression and without loss of generality, assume $\mathbb{P}(u_j|\mathrm{f}_i, \Phi_i)$ follows a Laplacian distribution [29], [30]. We can also choose, for example, Gaussian distribution prior to the data distribution to handle data from different sources [26], [31], [32]

$$\mathbb{P}(u_j|\mathrm{f}_i, \Phi_i) = \frac{1}{2\sigma} \exp\left(- \frac{|u_j - \mathbb{E}(u_j)|}{\sigma} \right) \tag{3}$$

where σ is a fixed variance [33], and $\mathbb{E}(u_j)$ is approximated based on ψ given f_i and Φ_i (see Section IV-A).

IV. Our Proposed Approach

The objective function of GLSC takes the form

$$\min_{\mathbf{w}_t, \psi} \frac{1}{T} \sum_{t=1}^{T} \big(\mathcal{L}\big(y_t, \mathbf{w}_t^\top \psi(\mathbf{x}_t)\big) + \alpha\ \mathcal{H} + \lambda\ \Omega(\mathbf{w}_t, \psi) \big) \tag{4}$$

where the first term as a supervised loss function is minimized for classification purpose. The second term, a reconstruction error function, indicates the approximation between the desired $\mathbf{u}_t$ and the reconstructed $\psi(\mathbf{x}_t)$. The third term $\Omega(\mathbf{w}_t, \psi)$ is a model regularizer (penalty function), which on the one hand penalizes the model if it becomes too complex to avoid over-fitting and on the other hand encourages a sparse model representation to bound the maximal dimension of the model parameters. α and λ are the tradeoff parameters for controlling the significance of the second and third terms, respectively.

In the rest of this section, we begin by presenting the building blocks of GLSC (such as $\mathcal{H}$ and $\Omega(\mathbf{w}_t, \psi)$ in the objective function). The updating rules and the prediction strategy are thereafter scrutinized. We end by analyzing the complexity of our proposal.

A. Learning From Reconstruction Error

In capricious data streams, the feature spaces between any two consecutive instances could be different, leading to a highly dynamic environment. Learning a complex reconstructive mapping based on existing methods [34], [35] is thus unrealistic. We restrict our interest in finding a linear mapping relationship between two features. As a motivating example to justify the appropriateness of using a linear mapping, we consider an online text classification task where the universal features are vocabularies from multiple languages [36]. Typically, when the word "Sano" (SP) is unobservable, it can be reconstructed with a linear combination of its related words, such as "Saine" (FR), "Great" (EN), and "Wholesome" (EN), through an equation of "Sano" = 0.4 × "Saine" + 0.4 × "Great" + 0.2 × "Wholesome." The coefficient of the related words (i.e., "Saine," "Great," and "Wholesome") indicates the relatedness between "Sano" and each of these words—the larger the value, the higher the relatedness. The non-related words (e.g., "Awful," whose semantic meaning is opposite to "Sano") are thus assigned zero-valued coefficients, and shall not be involved in the reconstruction of "Sano."

Specifically, we define $\mathbb{E}(u_j) = \mathbf{G}_{i,j}\mathrm{f}_i$ in (3), where $\mathbf{G}_{i,j} \geq 0$ represents the weight of the out-edge from vertex i to j. Accordingly, the log-likelihood maximization problem in (1) can be rewritten as

$$\begin{aligned} \max \mathcal{Q} &= \sum_{i=1}^{d_t} \log \left(\prod_{j=1}^{|\mathcal{U}_t|} \mathbb{P}(u_j|\mathrm{f}_i, \Phi_i) \right) \\ &= \sum_{i=1}^{d_t} \sum_{j=1}^{|\mathcal{U}_t|} \left(- \frac{|u_j - \mathbf{G}_{i,j}\mathrm{f}_i|}{\sigma} + \log \frac{1}{2\sigma} \right). \end{aligned} \tag{5}$$

Evidently, maximizing (5) is equivalent to minimizing the following optimization problem with respect to $\mathbf{G}$:

$$\min \mathcal{H} = \sum_{i=1}^{d_t} \sum_{j=1}^{|\mathcal{U}_t|} |u_j - \mathbf{G}_{i,j}\mathrm{f}_i| = \left\| \mathbf{u}_t - \frac{1}{d_t} \mathbf{G}\mathbf{r}^\top \mathbf{x}_t \right\|_2^2 \tag{6}$$

where $\|\cdot\|_2$ is an ℓ_2-norm. Here, $\mathbf{G}\mathbf{r}$ denotes $[\Phi_1, \ldots, \Phi_{d_t}]^\top$, where the vertex approximator Φ_i is selected according to the features f_i in $\mathbf{x}_t$. The use of $\mathbf{G}\mathbf{r}$ enables us to update $\mathbf{G}$ more efficiently. In particular, instead of updating the whole $\mathbf{G}$, we update only $\mathbf{G}\mathbf{r}$, which is a part of $\mathbf{G}$, determined by the features carried by $\mathbf{x}_t$.

Optimizing (6) requires the knowledge of the unknown part of $\mathbf{u}_t$ (the $\tilde{\mathbf{x}}_t$ part). To make (6) optimizable, we orthogonally project both $\mathbf{u}_t$ and $\mathbf{G}\mathbf{r}^\top \mathbf{x}_t$ onto the $\mathbb{R}^{d_t}$ feature space, transforming the (6) into the following:

$$\begin{aligned} &\min \left\| \Pi_{\mathbb{R}^{d_t}} \left(\mathbf{u}_t - \frac{1}{d_t} \mathbf{G}\mathbf{r}^\top \mathbf{x}_t \right) \right\|_2^2 \\ &\quad = \min \left\| \Pi_{\mathbb{R}^{d_t}}(\mathbf{u}_t) - \frac{1}{d_t} \Pi_{\mathbb{R}^{d_t}}(\mathbf{G}\mathbf{r}^\top \mathbf{x}_t) \right\|_2^2. \end{aligned}$$

The aforementioned transformation is realized by distributing the orthogonal projection operator over the minus operation. By the definition of $\mathbf{u}_t$, we have $\Pi_{\mathbb{R}^{d_t}}(\mathbf{u}_t) = \mathbf{x}_t$, allowing us to simplify the aforementioned equation to the following:

$$\min \left\| \mathbf{x}_t - \frac{1}{d_t} \Pi_{\mathbb{R}^{d_t}}(\mathbf{G}\mathbf{r}^\top \mathbf{x}_t) \right\|_2^2. \tag{7}$$

Surprisingly, we can tell from (7) that the reconstruction error is minimized if the reconstructive mapping is defined as $\psi(\mathbf{x}_t) = (1/d_t)(\mathbf{G}\mathbf{r}^\top \mathbf{x}_t)$. Alternatively, we can understand the minimization problem in (7) as follows. Suppose there is an accurate mapping ψ that can reconstruct $\mathbf{x}_t$ in $\mathcal{U}_t$ without any noise, then the values of the observable features in the reconstructed $\mathbf{x}_t$ should be numerically identical to those in the original $\mathbf{x}_t$.

It is worth noting that, since $\mathbf{G}\mathbf{r}$ is a discrete variable determined by $\mathbf{x}_t$, the problem in (7) is an integer program and it is difficult to solve. We introduce an indicator matrix $\mathbf{I}_t \in \mathbb{R}^{d_t \times |\mathcal{U}_t|}$ to represent which features in $\mathcal{U}_t$ are carried by $\mathbf{x}_t$. Such kind of indicator matrix is also known as binary selector matrix [37], whose nonzero entries are pointers to the universal features carried by $\mathbf{x}_t$. For example, having an

arriving $\mathbf{x}_t = [\mathrm{f}_1, \mathrm{f}_3, \mathrm{f}_6]^\top$ and $\mathcal{U}_t = \{f_1, f_2, f_3, f_4, f_5, f_6\}$, we construct $\mathbf{I}_t$ as

$$\mathbf{I}_t = \begin{array}{c} \begin{matrix} f_1 & f_2 & f_3 & f_4 & f_5 & f_6 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \begin{matrix} \mathrm{f}_1 \\ \mathrm{f}_3 \\ \mathrm{f}_6 \end{matrix}$$

which satisfies $\mathbf{G}\mathbf{r} = \mathbf{I}_t\mathbf{G}$ and $\Pi_{\mathbb{R}^{d_t}}(\mathbf{G}\mathbf{r}^\top\mathbf{x}_t) = \mathbf{I}_t(\mathbf{G}\mathbf{r}^\top\mathbf{x}_t)$. As such, the optimization problem in (7) with respect to $\mathbf{G}$ is tightly relaxed and becomes differentiable, which provides great convenience for our further derivation.

B. Learning From Supervised Loss

The accuracy of the universal feature space recovered by minimizing (7) may be affected when $\mathbf{x}_t$ does not convey sufficient information, for example, $\mathbf{x}_t$ only carries new features or the number of features in $\mathbf{x}_t$ is few. In this case, there could exist arbitrarily many possible reconstruction choices of $\psi(\mathbf{x}_t)$ that perfectly match $\mathbf{x}_i$ on the observable entries, among which searching the optimal one requires external information. To address this issue, we utilize the class label, an abstract representation of the reconstructed features, to provide extra supervised information for learning a better mapping ψ.

We train the learner by minimizing the supervised loss jointly along with the reconstruction error. With $\psi(\mathbf{x}_t) = (1/d_t)(\mathbf{G}\mathbf{r}^\top\mathbf{x}_t)$, in this article, the supervised loss function is implemented with the squared loss

$$\begin{aligned} \mathcal{L}\big(y_t, \mathbf{w}_t^\top\psi(\mathbf{x}_t)\big) &= \big(y_t - \mathbf{w}_t^\top\psi(\mathbf{x}_t)\big)^2 \\ &= \left(y_t - \frac{1}{d_t}\mathbf{w}_t^\top\mathbf{G}\mathbf{r}^\top\mathbf{x}_t\right)^2. \end{aligned} \tag{8}$$

Model Sparsity: The dimension of $\mathbf{w}_t$ will go infinite as the data keep streaming with new features. To bound the maximum dimension, penalizing the learning model with an ℓ_1-norm regularizer is a common choice. This is because it encourages a sparse solution of $\mathbf{w}_t$ in which the values of many weight coefficients are forced to be small or even zero. The dimension of $\mathbf{w}_t$ could thus be bounded by truncating the smallest weight coefficients, with a ratio of γ.

Unfortunately, directly adopting the ℓ_1-norm regularizer for dealing with capricious data streams faces the following issue. The issue is, since the feature space of capricious data streams varies arbitrarily, the features are described by the different numbers of data instances as data streams flow in. The smaller the number of instances describing a feature, the more biased the distribution is learned regarding this feature. Such biases may escalate with an increasing number of features being described by few instances, rendering the ℓ_1-norm regularizer to be.

1) *Ineffective:* In real applications, the dimension of the universal feature space can increase rapidly, asking for timely truncation. The ℓ_1-norm regularizer, in this case, is inefficient since it may require a very large number of learning iterations between two truncations to offset the biases.
2) *Invalid:* When a new feature just appears, it is a convention to initialize its weight coefficient as a small scalar as we do not have any prior knowledge about this feature. If one iteration concurrently involves features-appearing and learner-truncating, the weight coefficients of these newly appeared features are truncated by coincident, among which we may miss the important features. In this case, the ℓ_1-norm regularizer becomes functionally invalid, as it is sensitive to the sequence of arriving instances.

To address the aforementioned drawback, we take the structure of graph $\mathcal{G}$, which represents the relationship between features, into consideration. When a pair of features show strong relatedness, the weight of the edge between them, i.e., $\mathbf{G}_{i,j}$, is large, and their feature coefficients, i.e., w_i and w_j, should be similar. To achieve this, we draw insights from graph spectral analysis to design a graph regularizer onto ℓ_1-norm. The regularization term in (4) is finally defined as

$$\begin{aligned} \Omega(\mathbf{w}_t, \psi) &= \beta\|\mathbf{w}_t\|_1 + (1-\beta)\sum_{i=1}^{|\mathcal{U}_t|}\sum_{j=1}^{|\mathcal{U}_t|}\mathbf{G}_{i,j}(w_i - w_j)^2 \\ &= \beta\|\mathbf{w}_t\|_1 + 2(1-\beta)\,\mathrm{Tr}\left(\mathbf{w}_t^\top\mathbf{L}\mathbf{w}_t\right) \end{aligned} \tag{9}$$

where $\mathbf{L}$ is the graph Laplacian of $\mathbf{G}$ and β is a tradeoff parameter.

C. Updating Rules

By plugging (7), (8), and (9) into (4), our learning task is reduced to solve the empirical risk minimization problem as follows:

$$\begin{aligned} \arg\min_{\mathbf{w}_t, \mathbf{G}} \frac{1}{T}\sum_{t=1}^{T}\Bigg(\left(y_t - \frac{1}{d_t}\mathbf{w}_t^\top(\mathbf{I}_t\mathbf{G})^\top\mathbf{x}_t\right)^2 \\ + \alpha\|\mathbf{x}_t - \frac{1}{d_t}\Pi_{\mathbb{R}^{d_t}}(\mathbf{I}_t\mathbf{G})^\top\mathbf{x}_t\|_2^2 \\ + \beta_1\|\mathbf{w}_t\|_1 + \beta_2\,\mathrm{Tr}\left(\mathbf{w}_t^\top\mathbf{L}\mathbf{w}_t\right)\Bigg) \end{aligned} \tag{10}$$

where $\beta_1 = \lambda\beta$, and $\beta_2 = 2\lambda(1-\beta)$. We prove that the main function (denoted by $\mathcal{F}$) in (10) is bi-convex, providing a theoretical guarantee for convergence (see Section I in the Supplementary Material for details). To solve (10), we follow the common steps of solving a bi-convex optimization problem: 1) we divide (10) into two convex optimization subproblems, which are with respect to $\mathbf{w}_t$ and $\mathbf{G}$, respectively, and 2) the two subproblems are simultaneously solved at each iteration.

In this article, we use block-coordinate gradient descent [38], [39] to optimize the subproblems. For updating $\mathbf{w}_t$, we simply employ the first-order gradient descent as $\mathbf{w}_{t+1} = \mathbf{w}_t - \tau\nabla_{\mathbf{w}_t}\mathcal{F}$. For updating $\mathbf{G}$, to guarantee its nonnegativity, we follow the spirit in [40] to use the multiplicative updating rules in a generic form as: $\mathbf{G} = \mathbf{G} \circ (\nabla_{\mathbf{G}}^-\mathcal{F})/(\nabla_{\mathbf{G}}^+\mathcal{F})$, where we put all the negative terms of the gradient in the numerator $\nabla_{\mathbf{G}}^-\mathcal{F}$ and all the positive terms in the denominator $\nabla_{\mathbf{G}}^+\mathcal{F}$. $\circ$ denotes the Hadamard (element-wise) product. In implementing algorithms, any

negative value appears in $\mathbf{G}$ is set to zero. The gradients of $\mathcal{F}$ with respect to $\mathbf{w}_t$ and $\mathbf{G}$ are

$$\nabla_{\mathbf{w}_t}\mathcal{F} = -(2/d_t)\big(y_t - (1/d_t)\mathbf{w}_t^\top(\mathbf{I}_t\mathbf{G})^\top\mathbf{x}_t\big)(\mathbf{I}_t\mathbf{G})^\top\mathbf{x}_t + \beta_1\partial\|\mathbf{w}_t\|_1 + \beta_2(\mathbf{L}+\mathbf{L}^\top)\mathbf{w}_t \quad (11)$$

$$\nabla_{\mathbf{G}}\mathcal{F} = (-2/d_t)\big(y_t - (1/d_t)\mathbf{w}_t^\top(\mathbf{I}_t\mathbf{G})^\top\mathbf{x}_t\big)\mathbf{I}_t^\top\mathbf{x}_t\mathbf{w}_t^\top - (2\alpha/d_t)\mathbf{I}_t^\top\mathbf{x}_t\big(\mathbf{x}_t - (1/d_t)\mathbf{I}_t(\mathbf{I}_t\mathbf{G})^\top\mathbf{x}_t\big)^\top\mathbf{I}_t. \quad (12)$$

D. Ensemble Prediction

Given $\psi(\mathbf{x}_t)$ and the corresponding learner $\mathbf{w}_t$, conventionally the prediction is defined in an inner product form, namely, $\langle\mathbf{w}_t, \psi(\mathbf{x}_t)\rangle$. To further improve the prediction performance, we can combine two base predictions based on $\mathbf{x}_t$, which contains the original observable features, and $\tilde{\mathbf{x}}_t = [\tilde{f}_{d_t+1}, \ldots, \tilde{f}_{|\mathcal{U}_t|}]^\top \in \mathbb{R}^{|\mathcal{U}_t|-d_t}$, which contains the reconstructed unobservable features

$$\hat{y}_t = p\langle\bar{\mathbf{w}}_t, \mathbf{x}_t\rangle + (1-p)\langle\tilde{\mathbf{w}}_t, \tilde{\mathbf{x}}_t\rangle \quad (13)$$

where $\bar{\mathbf{w}}_t$ and $\tilde{\mathbf{w}}_t$, together forming $\mathbf{w}_t$, are the weight coefficients of $\mathbf{x}_t$ and $\tilde{\mathbf{x}}_t$, respectively. The value of p decides the impact of $\mathbf{x}_t$ and $\tilde{\mathbf{x}}_t$ in making predictions. Such an ensemble prediction can eliminate the prediction errors caused by potential noises in the reconstructed observable features, which is likely to happen in the initial iterations when few data instances have been seen.

The prediction loss function $\ell(\cdot)$ is convex in its first argument. In the implementation, we choose logistic loss for classification task, namely, $\ell(y, \hat{y}) = (1/\ln 2)\ln(1+\exp(-y\hat{y}))$. Let $L_T^{\text{obs}} = \sum_{t=1}^{T}\ell(y_t, \langle\bar{\mathbf{w}}_t, \mathbf{x}_t\rangle)$ and $L_T^{\text{rec}} = \sum_{t=1}^{T}\ell(y_t, \langle\tilde{\mathbf{w}}_t, \tilde{\mathbf{x}}_t\rangle)$ denote the cumulative losses suffered by making predictions on $\mathbf{x}_t$ and $\tilde{\mathbf{x}}_t$ over T iterations, respectively. At the iteration $T+1$, we update the parameter p in (13) based on exponential of the cumulative loss [41]

$$p = \frac{\exp\big(-\eta L_T^{\text{obs}}\big)}{\exp\big(-\eta L_T^{\text{obs}}\big) + \exp\big(-\eta L_T^{\text{rec}}\big)} \quad (14)$$

where η is a tuned parameter and its value assignment is discussed in Section V. The intuition behind such ensemble prediction strategy is that, when L_T^{obs} (or L_T^{rec}) is larger than L_T^{rec} (or L_T^{obs}), the impact of $\mathbf{x}_t$ (or $\tilde{\mathbf{x}}_t$) is negatively rewarded by our learning system.

E. Complexity Analysis

The details of GLSC are presented in Algorithm 1. A step-by-step running time complexity analysis is provided in Section III of the Supplementary Material. It is worth noting that the largest number of operations occurs at step 7, where the gradients in (11) and (12) are calculated, and correspondingly, the worst running time complexity is $\mathcal{O}(d_t^3 \times |\mathcal{U}_t|^2)$. Clearly, such complexity is unacceptable when the data streams flow in with high dimensionality. In response, this section analyzes the main causes of the poor efficiency and introduces a retrieval strategy used in the implementation which effectively improves the efficiency of GLSC.

The poor efficiency of GLSC is mainly caused by the inappropriate treatments in handling **Gr** and $\mathbf{G}$. We in (7) introduce $\mathbf{I}_t$ to relax the discreteness of the optimization problem. However, such a treatment brings more inner product operations, making GLSC less efficient. On the other hand, if the data streams come continually without stopping, it becomes infeasible to store and update $\mathbf{G}$ which consumes huge space resources.

Algorithm 1 GLSC Algorithm

Initialize: $\mathbf{w}_1 = [0, \ldots, 0]^\top \in \mathbb{R}^{d_1}$, $\mathcal{U}_t = \varnothing$, $\mathbf{G} = \varnothing$, $p = 0.5$, and $L_T^{\text{obs}} = L_T^{\text{rec}} = 0$.

1 **for** $t = 1, \ldots, T$ **do**
2 Receive instance $\mathbf{x}_t$, and $\mathcal{U}_t = \mathcal{U}_{t-1} \cup \mathbb{R}^{d_t}$;
3 Retrieve **Gr** from $\mathbf{G}$;
4 Predict the label as $\text{sign}(\hat{y}_t)$ using (13);
5 $L_T^{\text{obs}} \mathrel{+}= \ell(y_t, \langle\bar{\mathbf{w}}_t, \mathbf{x}_t\rangle)$, $L_T^{\text{rec}} \mathrel{+}= \ell(y_t, \langle\tilde{\mathbf{w}}_t, \tilde{\mathbf{x}}_t\rangle)$;
6 Reweight the parameter p using (14), where $\eta = 8\sqrt{1/\ln T}$;
7 Update $\mathbf{w}_{t+1}$ and $\mathbf{G}$ using (11) and (12), respectively;
8 Truncate $\mathbf{w}_{t+1}$ based on γ;

To save the computational cost and resources, we implement $\mathbf{G}$ using a nested hashmap, so that we do not need to initialize $\mathbf{I}_t$ at each iteration; Instead, we can retrieve **Gr** from $\mathbf{G}$, and update **Gr** directly. The idea of our retrieval strategy is to reuse Φ_i of x_i if x_i has already been observed before. For any new feature x_j carried by $\mathbf{x}_t$, we first build the out-edges between x_j and the others, representing the relatednesses among features. By doing this, the feature relatednesses are fully captured without being affected by the data scale—indeed, the big-O complexity of GLSC is reduced from $\mathcal{O}(d_t^3 \times |\mathcal{U}_t|^2)$ to $\mathcal{O}(d_t^2 \times |\mathcal{U}_t|)$.

We now analyze the running time complexity of the three algorithms that we will conduct comparative experiments with (see Section VI), i.e., OCO, OLSF, and FESL. For OLSF and FESL, we are not able to adapt them to work for capricious data streams. To make the complexity comparison meaningful, we first analyze their complexities in their own settings, namely, OLSF in trapezoidal data streams and FESL in feature evolvable streams. After this, we discuss the complexity of GLSC when adapted to these settings. For simplicity, we assume the matrices are invertible when the inversion operations are needed (for the singular matrices, we compute their Moore–Penrose pseudo-inverse instead).

The complexity for OCO, the most naïve OCO algorithm, is well known as $\mathcal{O}(D^2)$ for a fixed D-dimensional feature space [42]. Adapting OCO to handle capricious data streams requires to update a model whose number of parameters is $\mathcal{U}_t$ in order to handle each appeared feature. As a result, the time complexity of OCO is $\mathcal{O}(|\mathcal{U}_t|^2)$, which is worse than that of GLSC when $|\mathcal{U}_t| > d_t^2$. However, OCO uses sparse model representations in which coefficients corresponding to unobservable features are simply set to zeros at each iteration. Thus, in practice, OCO is faster than GLSC since sparse matrices are handled more efficiently at the architectural level of the machine. We will present more details in Section VI-D.

OLSF improves OCO by initializing model parameters of the new features using the principle of margin-maximum rather than padding zeros. Since the complexity of the initialization is linearly bounded by the number of new features, the time complexity of OLSF in the context of trapezoidal data streams is still $\mathcal{O}(|\mathcal{U}_t|^2)$. When GLSC is adapted to handle trapezoidal data streams, instead of calculating the feature mappings amongst all universal features, only the feature mapping from the old features to the new features is calculated. Thus, the complexity of GLSC is reduced to $\mathcal{O}(d_{t-1}^2 \times \Delta d_t)$, where $\Delta d_t = |\mathcal{U}_t| - d_{t-1}$ denotes the number of new features. We can observe that the complexity of GLSC is lower than that of OLSF when $|\mathcal{U}_t| > d_{t-1}^2$ and is higher otherwise. In all the evaluated data sets, GLSC takes about 20% more runtime than OLSF, with details in Section VI-D.

For FESL, the most complex part is computing feature space mapping that involves a matrix inversion. Since the complexity of this part is $\mathcal{O}(d_t \times (\Delta d_t)^2)$, so is the complexity of FESL itself. Note, for OCO, OLSF, and GLSC, the time complexity with T iterations can be obtained by multiplying the single iteration complexity with T. However, for FESL, the T iterations complexity is $\mathcal{O}(d_t \times (\Delta d_t)^2 \times B)$, where B denotes the number of instances that carry all features in $\mathcal{U}_t$. If we adapt GLSC to work for feature evolvable streams, the complexity of GLSC for T iterations is $\mathcal{O}(d_t^2 \times |\mathcal{U}_t| \times B)$. In practice, FESL usually sets d_t to be close to Δd_t, so the complexity of GLSC is slightly worse than FESL and is bounded by the ratio of $(\Delta d_t)/d_t$.

V. Theoretical Analysis

In this section, we borrow the idea of regret from online learning [41] to measure the performance of GLSC. We derive a cumulative loss bound and show that, over T iterations, our approach guarantees a lower cumulative loss than approaches that do not make use of the recovered feature space. To save space, the proofs for theorems in this section are provided in Section II of the Supplementary Material.

Theorem 1: Let $L_T = \sum_{t=1}^{T} \ell(y_t, \hat{y}_t)$ denote the overall cumulative loss of GLSC over T iterations. L_T with parameter $\eta = 8(1/\ln T)^{1/2}$ satisfies

$$L_T \leq \min\{L_T^{\text{obs}}, L_T^{\text{rec}}\} + \frac{T}{\sqrt{\ln T}} + \frac{\ln 2}{8}\sqrt{\ln T}.$$

Remark 1: This theorem indicates that the cumulative loss L_T is the lower of L_T^{obs} and L_T^{rec} bounded by the scalar $\Delta = T/(\ln T)^{1/2} + (\ln 2/8)(\ln T)^{1/2}$, which is sublinear to the number of iterations. The recovered feature space improves model accuracy when $\tilde{\mathbf{w}}_t$ is better than $\bar{\mathbf{w}}_t$ such that the relation $L_T^{\text{obs}} - L_T^{\text{rec}} > \Delta$ is satisfied. In this case, it is easy to verify that $L_T < L_T^{\text{obs}}$, indicating that the learner trained with the assistance of the recovered feature space yields a strictly lower cumulative loss than those without the assistance.

Furthermore, we have the following theorem.

Theorem 2: If $\bar{\mathbf{w}}_t$ is better than $\tilde{\mathbf{w}}_t$ over T iterations, then L_T is bounded as

$$L_T < L_T^{\text{obs}} + C$$

where C is a constant, and $C \ll \Delta$.

Remark 2: This theorem states that the cumulative loss L_T of GLSC is comparable to L_T^{obs} and is bounded by a constant. Note that the assumption of this theorem is very weak since it assumes $\ell_t^{\text{obs}} < \ell_t^{\text{rec}}$ for all iterations. In other words, the arriving instances always lie in a feature space that is more informative than the unobservable feature space. In practice, it is very likely that $\mathbf{x}_t$ carries very few features or $\mathbf{x}_t$ carries noisy features, making $\bar{\mathbf{w}}_t$ worse than $\tilde{\mathbf{w}}_t$ in some iteration. Therefore, Theorem 2 indeed gives an upper bound of the cumulative loss of GLSC over T iterations. In practice, GLSC enjoys lower cumulative losses.

Theorems 1 and 2 offer our learning algorithm a nice property as follows.

Corollary 1: The learning performance is improved by making use of the recovered universal feature space.

Proof: On the one hand, when $\bar{\mathbf{w}}_t$ is better than $\tilde{\mathbf{w}}_t$ over T iterations, Theorem 2 tells that the cumulative loss L_T of GLSC is comparable to L_T^{obs} and is bounded to a constant. On the other hand, when $\tilde{\mathbf{w}}_t$ is better than $\bar{\mathbf{w}}_t$ over T iterations, it is obvious that the recovered unobservable feature space is helpful. Furthermore, if $\tilde{\mathbf{w}}_t$ is better than $\bar{\mathbf{w}}_t$ to certain degree, satisfying $L_T^{\text{obs}} - L_T^{\text{rec}} > \Delta$, it is easy to verify that $L_T < L_T^{\text{obs}}$. To conclude, the learner with assistance from the universal feature space achieves better performance than that without the assistance. □

VI. Experiments

In this section, we first introduce the data sets used in this article along with the general settings (see Section VI-A). One data set is collected by ourselves as learning from capricious data streams has not been well explored and few well-established data sets are widely available. Section VI-B presents the experimental results, comparing GLSC with the state-of-the-art algorithms. Section VI-C investigates the helpfulness of constructing a universal feature space by comparing GLSC with its two variants, namely, GLSC-o(bservable) and GLSC-r(econstructed). Section VI-D presents the efficiency of relevant algorithms. We carry out an ablation study on parameters α, β_1, and β_2 of (10) in Section VI-E.

A. Data Sets and General Settings

We perform the experiments on 16 data sets consisting of 14 UCI data sets [43] and two real data sets—one is the IMDB data set [44], the other is our collected Communities that Care Youth Survey (CCYS) data set. The statistics of the used data sets are summarized in Table I.

The UCI data sets are randomly selected, spanning a broad range of applications such as economy, genetic research, medical science, etc. We synthesize capricious data streams by randomly removing features from each arriving instance $\mathbf{x}_t$. The ratio of the maximal removed features is denoted as *VI*. For example, $VI = 0.5$ means that at most 50% of features in $\mathbf{x}_t$ are randomly removed. The default value of *VI* is 0.5 in our experiments.

The task in the IMDB data set is to classify the movie reviews into positive and negative sentiments. Each word in the reviews is considered as a feature. Since the words used in

TABLE I
CHARACTERISTICS OF THE STUDIED DATA SETS

Dataset	# Inst.	# Feat.	Dataset	# Inst.	# Feat.
wpbc	198	33	german	1,000	20
ionosphere	351	34	svmguide3	1,284	21
wdbc	569	30	splice	3,190	60
australian	690	14	kr-vs-kp	3,196	36
credit-a	690	15	HAPT	10,929	561
wbc	699	9	magic04	19,020	10
diabetes	768	8	IMDB	25,000	7,500
dna	949	180	CCYS	33,000	355

each and every review could be different, we formulate the task as learning from a varying feature space. Each movie review is treated as a word vector comprising a bag of words. For those words that do not appear in the current review, we treat them as unobservable features. In OCO, the value of unobservable features in the word vector are simply set as zeros. As such, the OCO is equivalent to the 1-g bag-of-word (BOG) model known in the NLP tasks. The comparison between GLSC and OCO in the IMDB data set carries over to the comparison between GLSC and the 1-g BOG model.

We collected the real data set during the research project called the CCYS, which is administered and funded by the Louisiana government. The data are collected from 79 988 U-12 students enrolled in public schools throughout the State of Louisiana. In total, 355 features are designed and collected through questionnaires provided by eight independent agencies. These questionnaires assess the students' exposure to a set of risk and protective factors (e.g., family, neighborhood, school, etc.) which have impacts on the students' social behavior. For example, students who live in disorganized, crime-ridden neighborhoods are more likely to become involved in crime and drug abuse than those who live in safe ones. The students are not obligated to answer all the questionnaires, and in practice, they only answer the partial questions that they are interested, thus the feature space describing different students varies. Our task is to predict the students' involvement in anti-social behaviors (the original CCYS data are multilabeled, but in this article, we only focus on one label and delete the instances that are not associated with this label; after the deletion, we keep 33 000 instances).

To find the best settings of the parameters α, β_1, and β_2, we use grid searches ranging from 10^{-5} to 1. For efficiency purpose, we let $|\mathcal{U}_t| \leq 150$ by setting γ in different data sets. For more detailed settings, refer to the Supplementary Material.

B. Comparisons With State-of-the-Arts

Table II presents the results of performance comparison in terms of classification accuracy. Three baseline algorithms, OLSF [4], FESL [5], and OCO [1], as well as the proposed GLSC algorithm are evaluated in this section. In particular, OLSF can only handle trapezoidal data streams where the feature space monotonically augments as data flow in, while FESL can only handle feature evolvable streams where feature space batchly evolves by following an explicit pattern—both new and old features exist in an overlapping time period. The trapezoidal and feature evolvable data streams are the special cases of capricious data streams, and we simulate these two kinds of data streams by following the methods provided in the respective work. We compare GLSC with OLSF and FESL on trapezoidal data streams and feature evolvable streams, respectively. On capricious data streams, GLSC is compared with OCO, which, as mentioned in Section I, is a naïve online learning algorithm that makes a prediction based on the observable feature space only.

On trapezoidal data streams, the average accuracy values of GLSC and OLSF are 86.69% and 75.40%, respectively, and GLSC statistically achieves better results on 13 out of 16 data sets. Moreover, on 12 out of 16 data sets, the classification variances of GLSC are smaller than those of OLSF. The main reason is that GLSC considers the feature relatednesses in model penalty while OLSF does not, and therefore the classification accuracy of GLSC is more robust.

On feature evolvable streams, GLSC and FESL achieve 87.98% and 77.12% accuracy on average, respectively, and GLSC outperforms FESL on 11 data sets. This is because FESL trains a learner mainly with the help of the time period in which old and new features exist simultaneously, while GLSC can keep updating the learned reconstructive mapping over all iterations. The way that GLSC learns the mapping suggests that the classification accuracy could be improved when a large number of instances flow in, and the results support it. For example, we observe that the average accuracy of GLSC is 19.78% higher than that of FESL on large-scale data sets such as splice and HAPT.

On capricious data streams, the average accuracy of GLSC is 91.02%, while that of OCO is only 65.44%. In addition, GLSC wins over OCO on 15 data sets. We also find out that the classification result of GLSC is stable across different data sets. The results indicate that GLSC could effectively handle arbitrarily varying feature spaces.

Note, if we fix the feature space (i.e., no removed features), our proposed GLSC degrades to OCO with a spectral regularizer [45], [46]. Therefore, the performance of GLSC in handling a varying feature space is upper bounded by its performance in a fixed feature space. We show such upper bound performance in the rightmost column in Table II.

C. Impact of Universal Feature Space

In this section, we compare GLSC with three approaches. One is OCO, which could work on capricious data streams, as a baseline algorithm. The other two are the variants of GLSC, namely, GLSC-o(bservable) and GLSC-r(econstructed), respectively. Their difference is that, when making a prediction, GLSC-o uses the observable features while GLSC-r uses the reconstructed features. To investigate the impact of universal feature space, we aim to answer the following three questions:

Q1. How effectively can the universal feature space capture feature relatednesses?

The smaller reconstruction error the universal feature space has, the better the feature relatednesses are captured. In

TABLE II

EXPERIMENTAL RESULTS (MEAN ACCURACY ± STANDARD DEVIATION) ON 16 DATA SETS IN THE SETTINGS OF TRAPEZOIDAL DATA STREAMS, FEATURE EVOLVABLE STREAMS, CAPRICIOUS DATA STREAMS, AND FIXED FEATURE SPACE. WE APPLY A RANDOM PERMUTATION TO EACH DATA SET AND REPEAT THE EXPERIMENT TEN TIMES. THE BEST RESULTS ARE BOLD. • INDICATES GLSC HAS A STATISTICALLY SIGNIFICANT BETTER PERFORMANCE THAN THE COMPARED ALGORITHMS (HYPOTHESIS SUPPORTED BY PAIRED T-TESTS AT 95% SIGNIFICANCE LEVEL). THE WIN/TIE/LOSS COUNTS FOR GLSC ARE SUMMARIZED IN THE LAST ROW

	Trapezoidal Data Streams		Feature Evolvable Streams		Capricious Data Streams		Fixed Feature Space
Dataset	OLSF	GLSC	FESL	GLSC	OCO	GLSC	Upper Bound
wpbc	.586 ± .040 •	**.790 ± .014**	.719 ± .010 •	**.768 ± .005**	.510 ± .033 •	**.813 ± .024**	.872 ± .021
ionosphere	.856 ± .018 •	**.891 ± .003**	.833 ± .016 •	**.865 ± .007**	.638 ± .018 •	**.906 ± .014**	.937 ± .032
wdbc	.932 ± .012	**.945 ± .005**	.953 ± .022	**.968 ± .003**	.919 ± .063	**.951 ± .009**	.977 ± .015
australian	.739 ± .014 •	**.888 ± .002**	.849 ± .009 •	**.892 ± .004**	.804 ± .007 •	**.922 ± .011**	.951 ± .036
credit-a	.729 ± .011 •	**.859 ± .004**	.831 ± .009 •	**.888 ± .004**	.783 ± .010 •	**.926 ± .014**	.933 ± .004
wbc	.951 ± .009	**.968 ± .002**	.927 ± .071	**.972 ± .001**	.616 ± .002 •	**.939 ± .005**	.970 ± .006
diabetes	.679 ± .004 •	**.797 ± .004**	.652 ± .009 •	**.817 ± .005**	.672 ± .009 •	**.892 ± .017**	.931 ± .010
dna	.713 ± .004 •	**.893 ± .002**	.692 ± .021 •	**.940 ± .005**	.549 ± .006 •	**.960 ± .003**	.977 ± .002
german	.682 ± .007 •	**.828 ± .004**	.703 ± .004 •	**.751 ± .004**	.627 ± .001 •	**.827 ± .021**	.873 ± .025
svmguide3	.722 ± .013 •	**.854 ± .007**	.779 ± .010	**.804 ± .012**	.654 ± .017 •	**.882 ± .018**	.897 ± .081
splice	.773 ± .002 •	**.847 ± .003**	.612 ± .022 •	**.895 ± .003**	.491 ± .004 •	**.934 ± .003**	.962 ± .027
kr-vs-kp	.621 ± .005 •	**.860 ± .007**	.630 ± .016 •	**.938 ± .002**	.666 ± .008 •	**.912 ± .006**	.975 ± .008
HAPT	.980 ± .003	**.989 ± .050**	.749 ± .026 •	**.908 ± .016**	.811 ± .002 •	**.968 ± .002**	.997 ± .000
magic04	.692 ± .004 •	**.826 ± .005**	.806 ± .003	**.848 ± .019**	.717 ± .006 •	**.886 ± .013**	.904 ± .002
IMDB	.695 ± .023 •	**.813 ± .019**	.860 ± .004	**.883 ± .007**	.562 ± .025 •	**.891 ± .006**	.933 ± .012
CCYS	.714 ± .022 •	**.823 ± .008**	.744 ± .013 •	**.851 ± .005**	.652 ± .004 •	**.738 ± .005**	.802 ± .007
GLSC: w/t/l	13 / 3 / 0	—	11 / 5 / 0	—	15 / 1 / 0	—	—

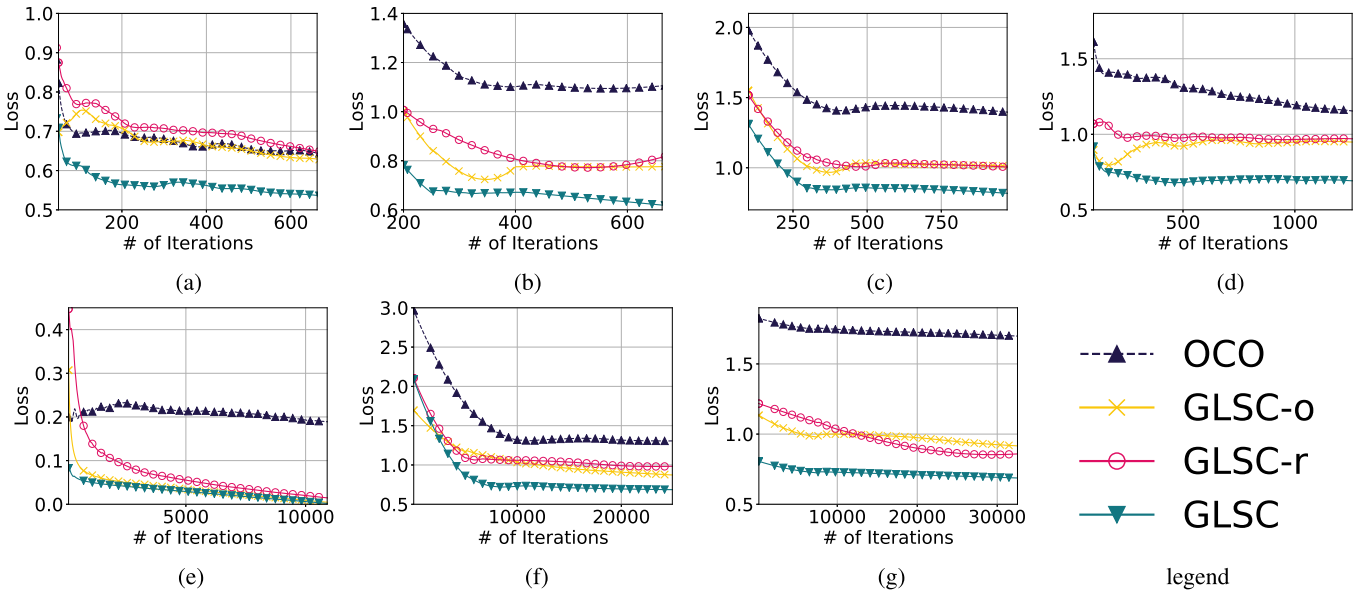


Fig. 2. Trends of average cumulative losses of GLSC and three baseline algorithms. (a) Australian. (b) Credit-a. (c) German. (d) Svmguide3. (e) HAPT. (f) IMDB. (g) CCYS.

addition, due to the bi-convexity of our objective optimization function (10), the reconstruction error is positively correlated with the prediction loss. Therefore, the prediction loss could be used in turn to measure the accuracy of the captured feature relatednesses.

Here, we present the trend of average cumulative loss (acl) in Fig. 2. At the iteration T, acl= L_T/T. Based on the results, we find that although the curve of GLSC-r may increase during the beginning iterations, it decreases as more data flow in and eventually converges. This intuitively makes sense because the more arriving instances the learner receives, the better the feature relatednesses are learned, reducing the value of acl. Moreover, the average cumulative losses of GLSC and GLSC-r both drop to small values after convergence. Thus, the reconstruction error in general is small, which suggests that the feature relatednesses are captured accurately.

Q2. Can the universal feature space help improve learning performance?

From Fig. 2, we make the following observations.

1) After convergence, the average cumulative loss of GLSC is significantly smaller than that of OCO. GLSC enjoys

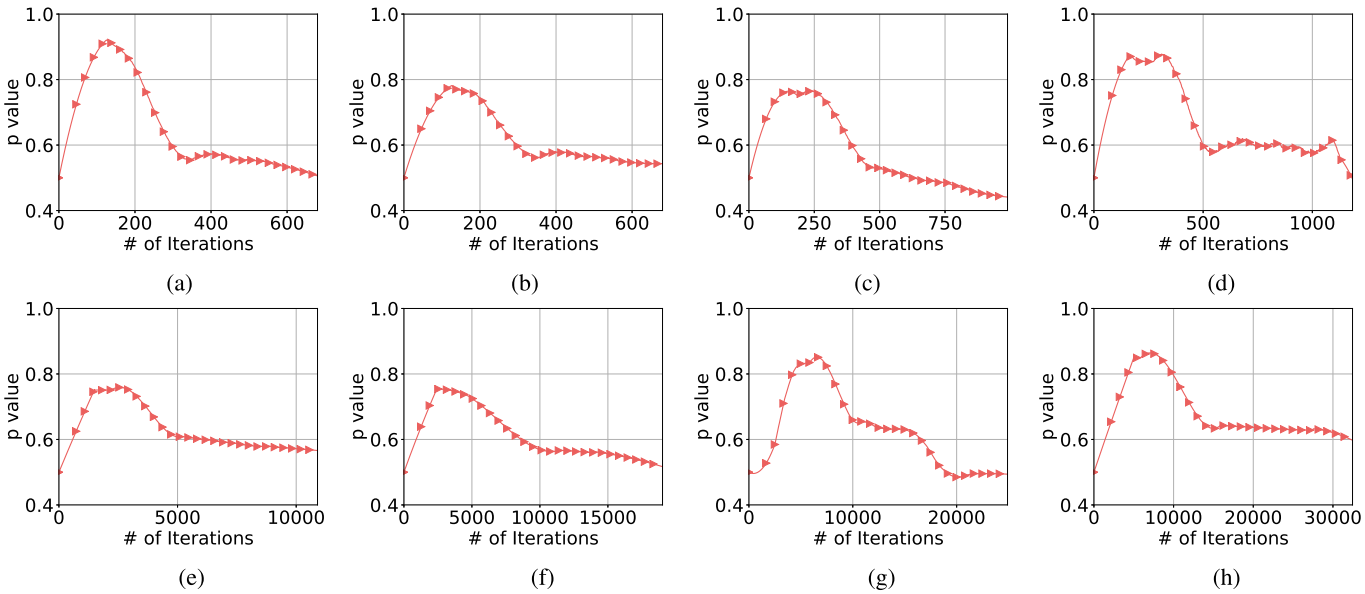


Fig. 3. Trends of p values in the ensemble prediction. (a) Australian. (b) Credit-a. (c) German. (d) Svmguide3. (e) HAPT. (f) Magic04. (g) IMDB. (h) CCYS.

better performance because the universal feature space can provide more information.

2) OCO may surpass GLSC-o when the number of instances is small, but the average cumulative loss of GLSC-o becomes smaller than that of OCO after convergence. This means that a better learner is obtained based on the universal feature space.
3) The average cumulative loss of GLSC is comparable to the best of GLSC-o and GLSC-r, and is smaller than them when the number of instances is large. The result validates Corollary 1 in Section V.

Q3. Can the ensemble prediction cancel the noise caused by the inaccurate feature space recovery?

It is intuitive to use the original and reconstructed features together in a single term, and give them the same importance. However, in the initial iterations during training, the reconstructed features are most likely to contain noise as few data instances have been seen, degrading the learning performance of GLSC. As such, we in (13) present an ensemble strategy to cancel the noise caused by the inaccurately reconstructed features. A self-adaptive parameter p is introduced to decide the significances of the original features (first term) and the reconstructed features (second term) in making predictions. Thus, it is interesting to know what values of p are learned during the learning process.

Fig. 3 illustrates the trends of the p values in the experiments on eight data sets. We start by setting $p = 0.5$, hoping that they contribute equally. Afterward, we observe: 1) the values of first p rise, meaning that the impact of the reconstructed features is negatively rewarded by our learning system, which validates that these reconstructed features contain noise in initial iterations as few data instances have been seen and 2) the values of the p drop, revealing that the larger the number of data instances feed to our learning system, the more precise the reconstructed features become. In most data sets, p converges to about 0.5, representing an ideal case where reconstructed and original features contribute equally. In other data sets, it is in $[0.4, 0.7]$, depending on the shape and noise in individual data sets.

TABLE III
COMPARISON OF RUNTIME PERFORMANCE (IN Seconds)

	Runtime (sec)			
Dataset	OCO	OLSF	FESL	GLSC
wpbc	2.38	4.76	3.08	5.24
ionosphere	3.21	7.30	5.35	8.03
wdbc	4.55	9.87	8.46	11.84
australian	3.76	8.47	9.35	10.16
credit-a	3.61	8.73	9.73	10.83
wbc	6.01	6.27	6.42	6.52
diabetes	6.43	6.63	6.63	6.89
dna	98.46	166.63	144.41	216.62
german	13.41	34.14	23.47	37.55
svmguide3	9.92	15.59	13.64	21.83
splice	63.54	158.84	136.15	190.61
kr-vs-kp	27.33	56.61	52.84	79.25
HAPT	462.50	925.01	751.57	1202.51
magic04	74.18	179.77	188.47	207.71
IMDB	1182.96	2001.93	1626.57	2602.51
CCYS	290.58	670.57	512.79	871.75

D. Comparisons on Computational Efficiency

In addition to the theoretical analysis of the time complexity provided in Section IV-E, a summary of the runtime performance for GLSC and the compared algorithms is reported in Table III, exhibiting the answer to the question.

Q4. At what cost do we achieve a specific increase in prediction accuracy?

For the data sets with a small number of features, e.g., WBC, diabetes, and magic04, the dimensions of the universal

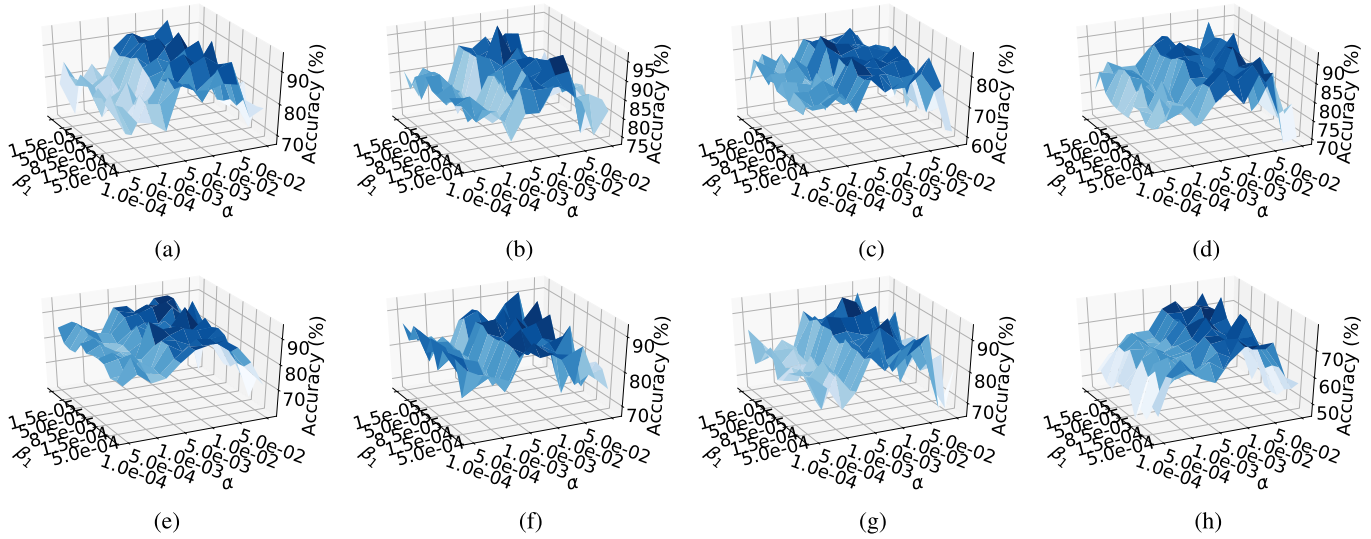


Fig. 4. Accuracy of GLSC with respect to different α's and β_1's. The darker the color, the higher the model accuracy. (a) Australian. (b) Credit-a. (c) German. (d) Svmguide3. (e) HAPT. (f) Magic04. (g) IMDB. (h) CCYS.

feature spaces constructed on these data sets are also small. GLSC correspondingly has similar runtimes with the compared algorithms. For the data sets with a large number of features, e.g., HAPT, IMDB, and CCYS, though the maximum dimension of the universal feature space is bounded in implementing algorithms, GLSC still takes more runtime than the others. Overall, the larger the total number of features owned by a data set, the higher the probability that arriving instances have large dimensions, and thus the longer the runtime that GLSC takes. For other data sets, the slowdowns are within a factor of 3 when compared with OCO, and within a factor of 1.5 when compared with OLSF and FESL.

E. Ablation Study

The objective function of GLSC is governed by three parameters α, β_1, and β_2. To investigate the impact of parameter values on the accuracy of our model, we have conducted an ablation study. This section provides the details of this article.

The parameter α controls how strongly the reconstruction function preserves the values of the original, observable feature. To investigate how it affects the prediction performance of GLSC as its value varies, we measure the performance as α is given each value from the set $\{1e{-}4, 5e{-}4, .001, .005, .01, .05, .1, .25, .5, .75, 1\}$. The other two parameters β_1 and β_2 encourage the classifier $\mathbf{w}_t$ to have sparse weight coefficients. The relationship between β_1 and β_2 is that $\beta_2 = 2\lambda - 2\beta_1$, which means that, when λ is fixed, the larger the value of β_1, the smaller the value of β_2. To grid-search the optimal values for β_1 and β_2, we let λ range over $\{.0001, .001, .01, .1\}$. For each λ value, we consider three β_1 values. Specifically, the corresponding values of β_1 to λ are $\{1.5e{-}5, 5e{-}5, 8.5e{-}5\}$, $\{1.5e{-}4, 5e{-}4, 8.5e{-}4\}$, $\{1.5e{-}3, 5e{-}3, 8.5e{-}3\}$, and $\{.015, .05, .085\}$. As an example, when $\lambda = 0.0001$, the values for β_1 are $\{1.5e{-}5, 5e{-}5, 8.5e{-}5\}$. The value for β_2 can be computed based on the aforementioned relationship.

Performance variance results are presented in Fig. 4. From the results, we can observe that the optimal value of α on different data sets is around 0.5 (the right-most tick on the α-axis). Thus, we can adopt such an empirical value in practice. Similar to other graph-based feature selection methods, the optimal value of β_1 (and β_2) in GLSC varies on different data sets. Making regularization parameters (β_1 and β_2 in our scenario) self-adaptive is under active research but remains an open issue [45], [47], [48]. We plan to address it in the future.

VII. Conclusion

In this article, we focus on a general and challenging setting—learning from a varying feature space. By utilizing the relatednesses among features, we learn a mapping from the observable feature space to a universal feature space in which both old and new features can be used for prediction. A learner is trained on the universal feature space, and theoretical results show that it can achieve better performance with strong guarantees. We carry out extensive experiments and the results demonstrate that our approach is effective.

Acknowledgment

The shorter version of this article, titled "Online Learning From Capricious Data Streams: A Generative Approach," was presented at the 28th International Joint Conference on Artificial Intelligence (IJCAI, 2019). The authors would like to thank anonymous IJCAI and TNNLS reviewers, whose reviews have improved both the content and the presentation of this article.

References

[1] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *Proc. ICML*, 2003, pp. 928–936.
[2] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithms," *J. Mach. Learn. Res.*, vol. 7, pp. 551–585, Dec. 2006.

[3] T. D. Nguyen, T. Le, H. Bui, and D. Phung, "Large-scale online kernel learning with random feature reparameterization," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Aug. 2017, pp. 2543–2549.

[4] Q. Zhang, P. Zhang, G. Long, W. Ding, C. Zhang, and X. Wu, "Online learning from trapezoidal data streams," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 10, pp. 2709–2723, Oct. 2016.

[5] B.-J. Hou, L. Zhang, and Z.-H. Zhou, "Learning with feature evolvable streams," in *Proc. NeurIPS*, 2017, pp. 1417–1427.

[6] M. M. Baig and H. Gholamhosseini, "Smart health monitoring systems: An overview of design and modeling," *J. Med. Syst.*, vol. 37, no. 2, p. 9898, Apr. 2013.

[7] R. Haux, "Health information systems–past, present, future," *Int. J. Med. Inform.*, vol. 75, nos. 3–4, pp. 268–281, 2006.

[8] S. B. Baker, W. Xiang, and I. Atkinson, "Internet of Things for smart healthcare: Technologies, challenges, and opportunities," *IEEE Access*, vol. 5, pp. 26521–26544, 2017.

[9] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. Hoboken, NJ, USA: Wiley, 2012.

[10] W. Zhang, W. Feng, and J. Wang, "Integrating semantic relatedness and words' intrinsic features for keyword extraction," in *Proc. IJCAI*, 2013, pp. 2225–2231.

[11] X. Zhang, X. Zhang, and H. Liu, "Self-adapted multi-task clustering," in *Proc. IJCAI*, 2016, pp. 2357–2363.

[12] J. Chen, D. Ji, C. L. Tan, and Z. Niu, "Unsupervised feature selection for relation extraction," in *Proc. CVPR*, 2005, pp. 262–267.

[13] J. Gama and P. P. Rodrigues, "An overview on mining data streams," in *Foundations of Computational, Intelligence*, vol. 6. Berlin, Germany: Springer, 2009, pp. 29–45.

[14] G. Zhou, K. Sohn, and H. Lee, "Online incremental feature learning with denoising autoencoders," in *Proc. AISTATS*, 2012, pp. 1453–1461.

[15] M. M. Masud *et al.*, "Classification and adaptive novel class detection of feature-evolving data streams," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 7, pp. 1484–1497, Jul. 2013.

[16] E. Beyazit, J. Alagurajah, and X. Wu, "Online learning from data streams with varying feature spaces," *Proc. AAAI Conf. Artif. Intell.*, vol. 33, Jul. 2019, pp. 3232–3239.

[17] C. Boutsidis, M. W. Mahoney, and P. Drineas, "Unsupervised feature selection for principal components analysis," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2008, pp. 61–69.

[18] A. K. Farahat, A. Ghodsi, and M. S. Kamel, "An efficient greedy method for unsupervised feature selection," in *Proc. IEEE 11th Int. Conf. Data Mining*, Dec. 2011, pp. 161–170.

[19] J. Li, J. Tang, and H. Liu, "Reconstruction-based unsupervised feature selection: An embedded approach," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Aug. 2017, pp. 2159–2165.

[20] S.-J. Huang, M. Xu, M.-K. Xie, M. Sugiyama, G. Niu, and S. Chen, "Active feature acquisition with supervised matrix completion," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 1571–1579.

[21] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," in *Proc. 26th Annu. Int. Conf. Mach. Learn. (ICML)*, 2009, pp. 689–696.

[22] P. Ruvolo and E. Eaton, "Online multi-task learning via sparse dictionary optimization," in *Proc. AAAI*, 2014, pp. 2062–2068.

[23] R. Salakhutdinov and G. Hinton, "Deep Boltzmann machines," in *Proc. AISTATS*, 2009, pp. 448–455.

[24] D. Chen, J. Lv, and Z. Yi, "Graph regularized restricted Boltzmann machine," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2651–2659, Jun. 2018.

[25] H. D. Nguyen and I. A. Wood, "Asymptotic normality of the maximum pseudolikelihood estimator for fully visible Boltzmann machines," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 4, pp. 897–902, Apr. 2016.

[26] L. Shao, D. Wu, and X. Li, "Learning deep and wide: A spectral method for learning deep networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 12, pp. 2303–2308, Dec. 2014.

[27] J. Xu, "An extended one-versus-rest support vector machine for multi-label classification," *Neurocomputing*, vol. 74, no. 17, pp. 3114–3124, Oct. 2011.

[28] J. A. Sáez, M. Galar, J. Luengo, and F. Herrera, "Analyzing the presence of noise in multi-class problems: Alleviating its influence with the one-vs-one decomposition," *Knowl. Inf. Syst.*, vol. 38, no. 1, pp. 179–206, Jan. 2014.

[29] L. Liu, C. Shen, L. Wang, A. Van Den Hengel, and C. Wang, "Encoding high dimensional local features by sparse coding based Fisher vectors," in *Proc. NeurIPS*, 2014, pp. 1143–1151.

[30] H. J. Park and T. W. Lee, "Modeling nonlinear dependencies in natural images using mixture of Laplacian distribution," in *Proc. NeurIPS*, 2005, pp. 1041–1048.

[31] Z. Xu, B. Liu, S. Zhe, H. Bai, Z. Wang, and J. Neville, "Variational random function model for network modeling," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 1, pp. 318–324, Jan. 2019.

[32] D. Wu *et al.*, "Deep dynamic neural networks for multimodal gesture segmentation and recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 8, pp. 1583–1597, Aug. 2016.

[33] M. V. Gerven, B. Cseke, R. Oostenveld, and T. Heskes, "Bayesian source localization with the multivariate Laplace prior," in *Proc. NeurIPS*, 2009, pp. 1901–1909.

[34] S. Jialin Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[35] S. Sun, "A survey of multi-view machine learning," *Neural Comput. Appl.*, vol. 23, nos. 7–8, pp. 2031–2038, Dec. 2013.

[36] J. T. Zhou, I. W. Tsang, S. J. Pan, and M. Tan, "Heterogeneous domain adaptation for multiple classes," in *Proc. AISTATS*, 2014, pp. 1095–1103.

[37] A. Das, A. V. Rao, and A. Gersho, "Variable-dimension vector quantization," *IEEE Signal Process. Lett.*, vol. 3, no. 7, pp. 200–202, Jul. 1996.

[38] P. Tseng and S. Yun, "A coordinate gradient descent method for nonsmooth separable minimization," *Math. Program.*, vol. 117, nos. 1–2, pp. 387–423, Mar. 2009.

[39] Z. Lu and L. Xiao, "On the complexity analysis of randomized block-coordinate descent methods," *Math. Program.*, vol. 152, nos. 1–2, pp. 615–642, Aug. 2015.

[40] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Proc. NeurIPS*, 2001, pp. 556–562.

[41] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games*. Cambridge, U.K.: Cambridge Univ. Press, 2006.

[42] E. Hazan, "Introduction to online convex optimization," *Found. Trends. Optim.*, vol. 2, nos. 3–4, pp. 157–325, 2016.

[43] D. Dua and E. K. Taniskidou. (2017). *UCI Machine Learning Repository*. [Online]. Available: http://archive.ics.uci.edu/ml

[44] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proc. ACL*, Portland, Oregon, USA, Jun. 2011, pp. 142–150. [Online]. Available: http://www.aclweb.org/anthology/P11-1015

[45] J. Ye and J. Liu, "Sparse methods for biomedical data," *ACM SIGKDD Explor. Newslett.*, vol. 14, no. 1, pp. 4–15, Dec. 2012.

[46] Z. Zhao, X. He, D. Cai, L. Zhang, W. Ng, and Y. Zhuang, "Graph regularized feature selection with data reconstruction," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 3, pp. 689–700, Mar. 2016.

[47] Z. Zha *et al.*, "Compressed sensing image reconstruction via adaptive sparse nonlocal regularization," *Vis. Comput.*, vol. 34, no. 1, pp. 117–137, Jan. 2018.

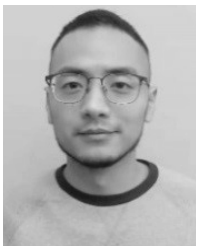
[48] R. Zhang, F. Nie, Y. Wang, and X. Li, "Unsupervised feature selection via adaptive multimeasure fusion," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9, pp. 2886–2892, Sep. 2019.

Yi He received the B.E. degree from the Harbin Institute of Technology, Harbin, China, in 2013, and the M.S. degree from the University of Louisiana at Lafayette, Lafayette, LA, USA, in 2017, where he is currently pursuing the Ph.D. degree with the Center for Advanced Computer Studies.

His research interests include data mining, machine learning, and optimization.

Baijun Wu received the B.S. and M.S. degrees in computer science from Sichuan University, Chengdu, China, in 2008 and 2011, respectively. He is currently pursuing the Ph.D. degree in computer science with the Center for Advanced Computer Studies, University of Louisiana at Lafayette, Lafayette, LA, USA.

He was a Software Engineer at Ericsson, Chengdu, from 2011 to 2012 and he worked for TP-LINK, Shenzhen, from 2012 to 2013. His current research interests include software engineering and machine learning.



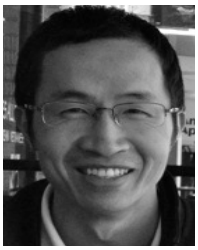
Di Wu received the B.S. degree in applied physics from the Nanjing University of Science and Technology, Nanjing, China, in 2009, the M.S. degree in optical engineering from Chongqing University, Chongqing, China, in 2012, and the Ph.D. degree in computer science from the Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Beijing, China, in 2019.

His research interests include data mining and machine learning.

Ege Beyazit received the bachelor's degree from the İzmir University of Economics, İzmir, Turkey, in July 2016. He is currently pursuing the Ph.D. degree with the Center for Advanced Computer Studies, University of Louisiana at Lafayette, Lafayette, LA, USA.

His research interests include online learning and deep learning.

Mr. Beyazit has served as a Reviewer for ICANN-18 and KDD-19.



Sheng Chen received the bachelor's degree in information technology from the Xi'an University of Technology, Xi'an, China, in 2005, the master's degree in software technology from Xidian University, Xi'an, in 2008, and the Ph.D. degree in computer science from Oregon State University, Corvallis, OR, USA, in 2015.

He is currently an Assistant Professor with the School of Computing and Informatics, University of Louisiana at Lafayette, Lafayette, LA, USA. His research interests include programming languages, software engineering, and applying machine learning in these areas.

Dr. Chen received the NSF CAREER Award in 2017.

Xindong Wu (Fellow, IEEE) received the Ph.D. degree in artificial intelligence from The University of Edinburgh, Edinburgh, U.K., in 1993.

He is currently the President of the Mininglamp Academy of Sciences, Mininglamp Technology, Beijing, China. He is also a Professor with the Key Laboratory of Knowledge Engineering with Big Data, Ministry of Education, Hefei University of Technology, Hefei, China. His research interests include data mining and knowledge engineering.

Dr. Wu is a fellow of the Association for American Association for the Advancement of Science (AAAS). He is also the Editor-in-Chief of *Knowledge and Information Systems* and *Advanced Information and Knowledge Processing* (Springer book series).