Online Learning from Capricious Data Streams: A Generative Approach

Yi He[1], **Baijun Wu**[1], **Di Wu**[2], **Ege Beyazit**[1], **Sheng Chen**[1] and **Xindong Wu**[1]
[1]School of Computing and Informatics, University of Louisiana at Lafayette, USA
[2]Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, China
{yi.he1, bj.wu, exb6143, chen, xwu}@louisiana.edu, wudi@cigit.ac.cn

Abstract

Learning with streaming data has received extensive attention during the past few years. Existing approaches assume the feature space is fixed or changes by following explicit regularities, limiting their applicability in dynamic environments where the data streams are described by an arbitrarily varying feature space. To handle such capricious data streams, we in this paper develop a novel algorithm, named OCDS (Online learning from Capricious Data Streams), which does not make any assumption on feature space dynamics. OCDS 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. Specifically, 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 analysis. The experimental results demonstrate that OCDS achieves conspicuous performance on both synthetic and real datasets.

1 Introduction

To date, many online learning approaches have been developed to handle streaming data [Zinkevich, 2003; Crammer *et al.*, 2006; Nguyen *et al.*, 2017]. 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 later data instances should include increasingly more features [Zhang *et al.*, 2016a], or batchly evolving, where a few consecutive data instances must include all possible features from the feature space [Hou *et al.*, 2017]. Unfortunately, these assumptions do not always hold in real applications. For example, in a smart healthcare platform [Baig and Gholamhosseini, 2013], features describing the symptoms of patients vary across IoT devices (thermometers, pulse monitors, respiratory sensors, *etc.*) and 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*.

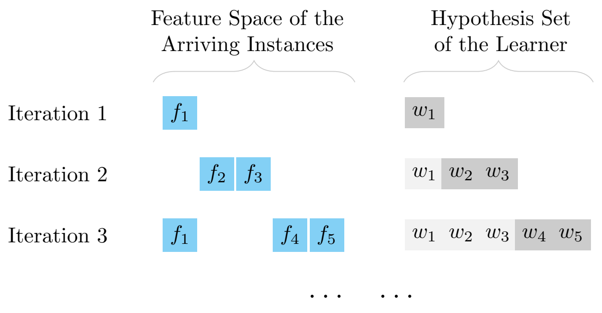


Figure 1: Naïve way of learning from a varying feature space. The weight coefficient w_i (marked in dark gray) of the corresponding new feature is initialized as zero.

At the first glance, one may think to adapt existing algorithms, such as Online Convex Optimization (OCO) [Zinkevich, 2003], for handling capricious data streams. Figure 1 depicts such a learning paradigm. The learner is trained on an *observable feature space* which comprises the features carried by an arriving instance at the t^{th} iteration. Alas, 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 Figure 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 [Duda *et al.*, 2012]. Second, when the features that have been observed by the learner become unavailable in latter iterations, the learned patterns regarding these features are ignored [Hou *et al.*, 2017]. As a result, the learner does not exert its full power to achieve the best prediction performance.

To overcome these limitations, we in this paper propose the Online learning from Capricious Data Streams (OCDS) 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 t^{th} 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 Figure 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.

We build upon a key insight that enables OCDS to infer unobservable features from observable ones: in practice there exist relatednesses among features [Zhang *et al.*, 2013; Zhang *et al.*, 2016b; Chen *et al.*, 2005]. Specifically, OCDS 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 OCDS is validated in three scenarios: trapezoidal data streams [Zhang *et al.*, 2016a], feature evolvable streams [Hou *et al.*, 2017], and capricious data streams.

Specific contributions in this paper are 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 prove that the obtained universal feature space can effectively improve the learning performance of OCDS.
3. We analyze the performance bound of OCDS.

Due to the page limitation, we present the detailed proofs (derivations), time complexity analysis, and complete experimental results in the supplementary material [He *et al.*, 2019].

2 Related Work

In this paper, we focus on learning data streams from a varying feature space. It is worthy pointing out that though concept-drift happens in streaming data where the underlying data distribution keeps changing [Gama and Rodrigues, 2009], the number of features carried by each instance is fixed in concept-drift, which is different from our learning task.

The studies related to our learning task include online learning from a fixed feature space [Zinkevich, 2003; Nguyen *et al.*, 2017], from an incremental feature space [Zhou *et al.*, 2012; Zhang *et al.*, 2016a; Beyazit *et al.*, 2018], and from an evolvable feature space [Masud *et al.*, 2013; Hou *et al.*, 2017]. 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 by [Beyazit *et al.*, 2019] 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 own technical challenges and solutions.

The key idea of OCDS is to learn a reconstructive mapping by exploiting feature relatednesses. In this regard, our work is also related to the feature space reconstruction-based approaches [Li *et al.*, 2017; Huang *et al.*, 2018]. However, most of them focus on feature selection and extraction and to our best knowledge, none of them consider the varying of feature space during the learning process.

3 Preliminaries

We focus on binary classification in this paper. Multiclass problems could be decomposed to multiple binary classification subproblems, using One vs Rest [Xu, 2011] or One vs One [Sáez *et al.*, 2014] strategies.

Learning Task Setup

Let $\{(\mathbf{x}_t, y_t) | t = 1, 2, \ldots, T\}$ denote a sequence of arriving instances with labels, where $\mathbf{x}_t = [x_1, x_2, \ldots, x_{d_t}]^\top \in \mathbb{R}^{d_t}$ is a d_t-dimensional vector and $y_t \in \{-1, +1\}$ represents the class label. At t^{th} 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 $\mathcal{U}_t = \{\mathbb{R}^{d_1} \cup \mathbb{R}^{d_2} \cup \ldots \cup \mathbb{R}^{d_t}\}$ denote the universal feature space at the t^{th} iteration where the features of $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t$ are included. We in this paper restrict the discussion to a linear classifier $\mathbf{w}_t$ based on a vector of weight coefficients, which is a common setting in online learning. If new features appear in $\mathbf{x}_t$, their corresponding weight coefficients in $\mathbf{w}_t$ are initialized as zeros. As a result, the dimension of $\mathbf{w}_t$ matches that of $\mathcal{U}_t$, namely $\mathbf{w}_t \in \mathbb{R}^{|\mathcal{U}_t|}$.

Generative Graphical Model

A generative graph is to embed a reconstructive mapping ψ: $\mathbb{R}^{d_t} \mapsto \mathcal{U}_t$. 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. We define *vertex approximator* Φ_i as a vector containing the weights of all out-edges of a vertex i. The graph $\mathcal{G}$ thus can be represented by a matrix that comprises a set of vertex approximators, namely $\mathbf{G} = [\Phi_1, \ldots, \Phi_{|\mathcal{U}_t|}]^\top \in \mathbb{R}^{|\mathcal{U}_t| \times |\mathcal{U}_t|}$.

We define the desired reconstruction of $\mathbf{x}_t$ in the universal feature space as $\mathbf{u}_t = [x_1, \ldots, x_{d_t}, \tilde{x}_{d_t+1}, \ldots, \tilde{x}_{|\mathcal{U}_t|}]^\top \in \mathbb{R}^{|\mathcal{U}_t|}$, where x_i and $\tilde{x}_j$ represent an original observable feature and a reconstructed unobservable feature, respectively. We infer $\mathbf{u}_t$ by maximizing a log-likelihood function:

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

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

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

where u_j denotes the j^{th} feature in $\mathbf{u}_t$.

Without loss of generality, we let $\mathbb{P}(u_j|x_i, \Phi_i)$ follow a Laplacian distribution [Liu *et al.*, 2014; Park and Lee, 2005]:

$$\mathbb{P}(u_j|x_i, \Phi_i) = \frac{1}{2\sigma} \exp\Big(- \frac{|u_j - \mathbb{E}(u_j)|}{\sigma} \Big), \tag{3}$$

where σ is a fixed variance [Gerven *et al.*, 2009], and $\mathbb{E}(u_j)$ is approximated based on ψ given x_i and Φ_i (see Section 4.1).

4 Our Proposed Approach

The objective function of OCDS takes the form:

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

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 parameter α is introduced to absorb the different scales between the first two terms. The third term bounds the maximal dimension of the learning model by penalizing its complexity, with a regularization parameter λ.

In the rest of this section, we first present the building blockings of OCDS by elaborating the components in the objective function (4). The updating rules are derived thereafter. We finally discuss the prediction strategy in OCDS.

4.1 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 [Pan *et al.*, 2010; Sun, 2013] is thus unrealistic. We restrict our interest in finding a linear mapping relationship between two features. Specifically, we define $\mathbb{E}(u_j) = \mathbf{G}_{i,j} x_i$ in (3), where $\mathbf{G}_{i,j}$ represents the weight of the out-edge from vertex i to j. Accordingly, maximizing (1) is equivalent to minimizing the following optimization problem *w.r.t.* $\mathbf{G}$:

$$\begin{aligned} \min \mathcal{H} &= \|\mathbf{u}_t - \frac{1}{d_t} \mathbf{Gr}^\top \mathbf{x}_t\|_2^2 \\ &= \|\mathbf{x}_t - \frac{1}{d_t} \Pi_{\mathbb{R}^{d_t}} (\mathbf{Gr}^\top \mathbf{x}_t)\|_2^2, \end{aligned} \tag{5}$$

where $\mathbf{Gr} = \mathbf{I}_t\mathbf{G}$. $\mathbf{I}_t \in \mathbb{R}^{d_t \times |\mathcal{U}_t|}$ is an indicator matrix that represents which features in $\mathcal{U}_t$ are carried by $\mathbf{x}_t$. $\|\cdot\|_2$ is an ℓ_2-norm, and $\Pi_{\mathbb{R}^{d_t}}(\cdot)$ is an operator that orthogonally projects $\mathbf{u}_t$ and $\mathbf{Gr}^\top\mathbf{x}_t$ onto the $\mathbb{R}^{d_t}$ feature space. Surprisingly, we can observe that the reconstruction error is minimized by (5) if the reconstructive mapping is defined as $\psi(\mathbf{x}_t) = (1/d_t)(\mathbf{Gr}^\top \mathbf{x}_t)$. In Algorithm 1, we show how $\mathbf{Gr}$ is retrieved for solving (5), such that $\psi(\mathbf{x}_t)$ is obtained. The main procedure of our retrieval strategy is to update $\mathbf{G}$ at each iteration. To improve the learning performance and save the computational cost, we 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. The weights of these edges are then randomly initialized, comprising Φ_j.

Algorithm 1: Retrieval Strategy

```
Input: An arriving instance $\mathbf{x}_t$, $\mathbf{I}_t$, and $\mathbf{G}$
1  foreach feature $x_i \in \mathbf{x}_t$ do
2      if $\Phi_i$ exists in $\mathbf{G}$ then
3          foreach feature $f_j \in \mathcal{U}_t$ do
4              if $x_j \in \mathbf{x}_t$ and $\mathbf{G}_{i,j}$ not exists then
5                  $\mathbf{G}_{i,j} \leftarrow$ a random number $r$;
6      else if $\Phi_i$ not exists in $\mathbf{G}$ then
7          Initialize $\Phi_i = [\ ]$;
8          while $|\Phi_i| < |\mathcal{U}_t|$ do
9              $\Phi_i$.append (a random number $r$);
10         $\mathbf{G}$.append ($\Phi_i$);
11 return $\mathbf{Gr} \leftarrow \mathbf{I}_t\mathbf{G}$;
```

4.2 Learning from Supervised Loss

The accuracy of the universal feature space recovered by minimizing (5) 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. 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 $\psi(\mathbf{x}_t) = (1/d_t)(\mathbf{Gr}^\top\mathbf{x}_t)$, the supervised loss function is:

$$\mathcal{L}\big(y_t, \mathbf{w}_t^\top \psi(\mathbf{x}_t)\big) = \big(y_t - \frac{1}{d_t} \mathbf{w}_t^\top \mathbf{Gr}^\top \mathbf{x}_t\big)^2. \tag{6}$$

The dimension of $\mathbf{w}_t$ will go infinite as the data keep streaming with new features. To bound the maximum dimension, penalizing the classifier 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.

The drawback of using the ℓ_1-norm regularizer is that, since the feature space of capricious data streams varies arbitrarily, the generated solution becomes sensitive to the sequence of arriving instances. To obtain a robust solution, we take the structure of graph $\mathcal{G}$, which represents the relationship between different features, into consideration. When a pair of features show strong dependency, 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 add a graph regularizer onto ℓ_1-norm. The formulation is:

$$\begin{aligned} &\beta\|\mathbf{w}_t\|_1 + (1-\beta) \sum_{i=1}^{|\mathcal{U}_t|} \sum_{j=1}^{|\mathcal{U}_t|} \mathbf{G}_{i,j}(w_i - w_j)^2 \\ =&\beta\|\mathbf{w}_t\|_1 + 2(1-\beta)\operatorname{Tr}(\mathbf{w}_t^\top \mathbf{L} \mathbf{w}_t), \end{aligned} \tag{7}$$

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

4.3 Updating Rules

By plugging (5), (6), and (7) into (4), our learning task is reduced to solve the *Empirical Risk Minimization* problem as below:

$$\arg\min_{\mathbf{G},\mathbf{w}_t} \frac{1}{T}\sum_{t=1}^{T}\bigg(\big(y_t - \mathbf{w}_t^\top \psi(\mathbf{x}_t)\big)^2 + \alpha\|\mathbf{x}_t - \Pi_{\mathbb{R}^{d_t}}\psi(\mathbf{x}_t)\|_2^2 + \beta_1\|\mathbf{w}_t\|_1 + \beta_2 \operatorname{Tr}(\mathbf{w}_t^\top \mathbf{L}\mathbf{w}_t)\bigg), \tag{8}$$

where $\psi(\mathbf{x}_t) = (1/d_t)((\mathbf{I}_t\mathbf{G})^\top \mathbf{x}_t)$, $\beta_1 = \lambda\beta$, and $\beta_2 = 2\lambda(1-\beta)$. We prove that the main function (denoted by $\mathcal{F}$) in (8) is bi-convex, providing a theoretical guarantee for convergence. To solve (8), we follow the common steps of solving a bi-convex optimization problem: (i) We divide (8) into two convex optimization subproblems, which are with respect to $\mathbf{w}_t$ and $\mathbf{G}$, respectively. (ii) The two subproblems are simultaneously solved at each iteration.

In this paper, we use *Coordinate Gradient Descend* [Tseng and Yun, 2009] as the optimizer for the subproblems. The updating rules for $\mathbf{w}_t$ and $\mathbf{G}$ are defined accordingly: $\mathbf{w}_{t+1} = \mathbf{w}_t - \tau\nabla_{\mathbf{w}_t}\mathcal{F}$ and $\mathbf{G} = \mathbf{G} - \tau\nabla_{\mathbf{G}}\mathcal{F}$, where $\tau = \sqrt{1/t}$ is a varied step size. In the updating rules, the partial derivatives (gradients) of $\mathcal{F}$ *w.r.t.* $\mathbf{w}_t$ and $\mathbf{G}$ are calculated as below.

$$\nabla_{\mathbf{w}_t}\mathcal{F} = -2\big(y_t - \mathbf{w}_t^\top\psi(\mathbf{x}_t)\big)\psi(\mathbf{x}_t) + \beta_1\partial\|\mathbf{w}_t\|_1 + \beta_2(\mathbf{L}+\mathbf{L}^\top)\mathbf{w}_t \tag{9}$$

$$\nabla_{\mathbf{G}}\mathcal{F} = (-2/d_t)\big(y_t - \mathbf{w}_t^\top\psi(\mathbf{x}_t)\big)\mathbf{I}_t^\top\mathbf{x}_t\mathbf{w}_t^\top - (2\alpha/d_t)\mathbf{I}_t^\top\mathbf{x}_t\big(\mathbf{x}_t - \Pi_{\mathbb{R}^{d_t}}\psi(\mathbf{x}_t)\big)^\top\mathbf{I}_t, \tag{10}$$

where $\partial\|\cdot\|_1$ denotes an entry-wise subdifferential operator.

4.4 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 $\tilde{\mathbf{x}}_t = [\tilde{x}_{d_t+1}, \dots, \tilde{x}_{|\mathcal{U}_t|}]^\top \in \mathbb{R}^{|\mathcal{U}_t|-d_t}$, which contains the reconstructed unobservable features:

$$\hat{y}_t = p\langle\bar{\mathbf{w}}_t, \mathbf{x}_t\rangle + (1-p)\langle\tilde{\mathbf{w}}_t, \tilde{\mathbf{x}}_t\rangle, \tag{11}$$

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 (11) based on exponential of the cumulative loss [Cesa-Bianchi and Lugosi, 2006].

$$p = \frac{\exp(-\eta L_T^{\text{obs}})}{\exp(-\eta L_T^{\text{obs}}) + \exp(-\eta L_T^{\text{rec}})}, \tag{12}$$

where η is a tuned parameter and its value assignment is discussed in Section 5. Intuitively, 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.

Algorithm 2: The OCDS algorithm

```
Initialize: $\mathbf{w}_1 = [0,\dots,0]^\top \in \mathbb{R}^{d_1}, \mathcal{U}_t = \varnothing, \mathbf{G} = \varnothing$,
            $p = 0.5$, and $L_T^{\text{obs}} = L_T^{\text{rec}} = 0$.
1 for $t = 1,\dots,T$ do
2     Receive instance $\mathbf{x}_t$, and $\mathcal{U}_t = \mathcal{U}_{t-1} \cup \mathbb{R}^{d_t}$;
3     Retrieve Gr using Algorithm 1;
4     Predict the label as $\text{sign}(\hat{y}_t)$ using (11);
5     $L_T^{\text{obs}}$ += $\ell(y_t, \langle\bar{\mathbf{w}}_t, \mathbf{x}_t\rangle)$, $L_T^{\text{rec}}$ += $\ell(y_t, \langle\tilde{\mathbf{w}}_t, \tilde{\mathbf{x}}_t\rangle)$;
6     Reweight the parameter $p$ using (12) where $\eta = 8\sqrt{1/\ln T}$;
7     Update $\mathbf{w}_{t+1}$ and $\mathbf{G}$ using (9) and (10), respectively;
8     Truncate $\mathbf{w}_{t+1}$ based on $\gamma$;
```

The details of OCDS are presented in Algorithm 2. The running time complexity of OCDS is $\mathcal{O}(d_t^2 \times |\mathcal{U}_t|)$. To fully capture the feature relatednesses without being affected by the data scale, we in the implementation update $\mathbf{Gr}$ since $\mathbf{G}$ can be easily restored based on it.

5 Theoretical Analysis

In this section, we borrow the the *regret* from online learning to measure the performance of OCDS. We derive a cumulative loss bound and show that the recovered feature space effectively enhances the performance. For the sake of soundness, the proofs of this section are provided in Section 3 of supplementary material.

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

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

Let $\Delta = T/\sqrt{\ln T} + (\ln 2/8)\sqrt{\ln T}$, which is linearly bounded by the number of iterations. Theorem 1 indicates that L_T is comparable to the minimum of L_T^{obs} and L_T^{rec}. 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:*

$$L_T < L_T^{obs} + C,$$

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

Theorem 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 OCDS 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 the assistance from the universal feature space achieves better performance than that without the assistance. ☐

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	a8a	32,560	123

Table 1: Characteristics of the studied datasets.

6 Experiments

We use 15 UCI datasets [Dua and Karra Taniskidou, 2017] and 1 real-world IMDB dataset [Maas *et al.*, 2011] to evaluate the performance of OCDS. The task in IMDB dataset is to classify the movie reviews into positive and negative sentiments. Each individual word in the reviews is considered as a feature. Since the words used in each and every user review could be different, we formulate the task as learning from an arbitrarily varying feature space. In UCI datasets, we simulate 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 details of the used datasets are provided in Table 1.

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 memory and running time efficiency, we let $|\mathcal{U}_t| \leq 150$ by setting γ in different datasets.

6.1 Comparisons with State-of-the-arts

Table 2 presents the results of performance comparison in terms of classification accuracy. Three baseline algorithms, OLSF [Zhang *et al.*, 2016a], FESL [Hou *et al.*, 2017], and OCO [Zinkevich, 2003], as well as the proposed OCDS 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 OCDS with OLSF and FESL on trapezoidal data streams and feature evolvable streams, respectively. On capricious data streams, OCDS is compared with OCO, which, as mentioned in Section 1, is a naïve online learning algorithm that makes prediction based on the observable feature space only.

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

On feature evolvable streams, OCDS and FESL achieve 87.98% and 77.12% accuracy on average, respectively, and OCDS outperforms FESL on 11 datasets. This is because FESL trains a learner mainly with the help of the time period in which old and new features exist simultaneously, while OCDS can keep updating the learned reconstructive mapping over all iterations. The way that OCDS 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 OCDS is 19.78% higher than that of FESL on large scale datasets such as splice and HAPT.

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

6.2 Impact of Universal Feature Space

In this section, we compare OCDS with three approaches. One is OCO, which could work on capricious data streams, as a baseline algorithm. The other two are the variants of OCDS, named OCDS-o(bservable) and OCDS-r(econstructed), respectively. The difference between them is that, when making prediction, OCDS-o uses the observable features while OCDS-r uses the reconstructed features. To investigate the impact of universal feature space, we aim to answer the following two 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 addition, due to the bi-convexity of our objective optimization function (8), the reconstruction error is positively correlated to 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 Figure 2. At the iteration T, $acl = L_T/T$. Based on the results, we find that although the curve of OCDS-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 OCDS and OCDS-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 Figure 2, we make the following observations. (i) After convergence, the average cumulative loss of OCDS is significantly smaller than that of OCO, especially on large datasets. OCDS enjoys better performance because the universal feature space can provide more information. (ii) OCO may surpass OCDS-o when the number of instances is small, but the average cumulative loss of OCDS-o becomes smaller

	Trapezoidal Data Streams		Feature Evolvable Streams		Capricious Data Streams	
Dataset	OLSF	OCDS	FESL	OCDS	OCO	OCDS
wpbc	.586 ± .040 •	**.790 ± .014**	.719 ± .010 •	**.768 ± .005**	.510 ± .033 •	**.813 ± .024**
ionosphere	.856 ± .018 •	**.891 ± .003**	.833 ± .016 •	**.865 ± .007**	.638 ± .018 •	**.906 ± .014**
wdbc	.932 ± .012	**.945 ± .005**	.953 ± .022	**.968 ± .003**	.919 ± .063	**.951 ± .009**
australian	.739 ± .014 •	**.888 ± .002**	.849 ± .009 •	**.892 ± .004**	.804 ± .007 •	**.922 ± .011**
credit-a	.729 ± .011 •	**.859 ± .004**	.831 ± .009 •	**.888 ± .004**	.783 ± .010 •	**.926 ± .014**
wbc	.951 ± .009	**.968 ± .002**	.927 ± .071	**.972 ± .001**	.616 ± .002 •	**.939 ± .005**
diabetes	.679 ± .004 •	**.797 ± .004**	.652 ± .009 •	**.817 ± .005**	.672 ± .009 •	**.892 ± .017**
dna	.713 ± .004 •	**.893 ± .002**	.692 ± .021 •	**.940 ± .005**	.549 ± .006 •	**.960 ± .003**
german	.682 ± .007 •	**.828 ± .004**	.703 ± .004 •	**.751 ± .004**	.627 ± .001 •	**.827 ± .021**
svmguide3	.722 ± .013 •	**.854 ± .007**	.779 ± .010	**.804 ± .012**	.654 ± .017 •	**.882 ± .018**
splice	.773 ± .002 •	**.847 ± .003**	.612 ± .022 •	**.895 ± .003**	.491 ± .004 •	**.934 ± .003**
kr-vs-kp	.621 ± .005 •	**.860 ± .007**	.630 ± .016 •	**.938 ± .002**	.666 ± .008 •	**.912 ± .006**
HAPT	.980 ± .003	**.989 ± .050**	.749 ± .026 •	**.908 ± .016**	.811 ± .002 •	**.968 ± .002**
magic04	.692 ± .004 •	**.826 ± .005**	.806 ± .003	**.848 ± .019**	.717 ± .006 •	**.886 ± .013**
IMDB	.695 ± .023 •	**.813 ± .019**	.860 ± .004	**.883 ± .007**	.562 ± .025 •	**.891 ± .006**
a8a	.714 ± .022 •	**.823 ± .008**	.744 ± .013 •	**.851 ± .005**	.452 ± .004 •	**.925 ± .005**
OCDS: w/t/l	13 / 3 / 0	—	11 / 5 / 0	—	15 / 1 / 0	—

Table 2: Experimental results (Mean Accuracy ± Standard Deviation) on 16 datasets in the settings of Trapezoidal Data Streams, Feature Evolvable Streams, and Capricious Data Streams. We apply a random permutation to each dataset and repeat the experiment 10 times. The best results are bold. • indicates OCDS 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 OCDS are summarized in the last row.

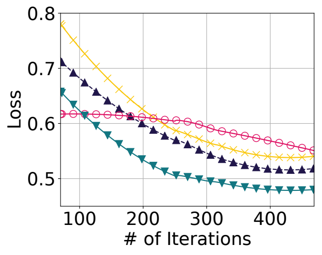
than that of OCO after convergence. This means that a better learner is obtained based on the universal feature space. (iii) The average cumulative loss of OCDS is comparable to the best of OCDS-o and OCDS-r, and is smaller than them when the number of instances is large. The result validates Corollary 1 in Section 5.

7 Conclusions

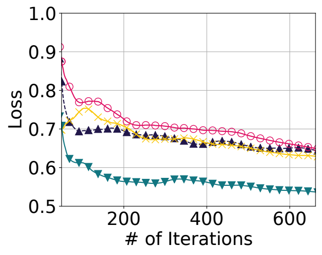
In this paper, we focus on a general and challenging setting – learning from an arbitrarily varying feature space. By utilizing the relatednesses among features, we learn a mapping from the observable feature space to a universal feature space in which both old and new features can be used for prediction. 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.

Acknowledgments

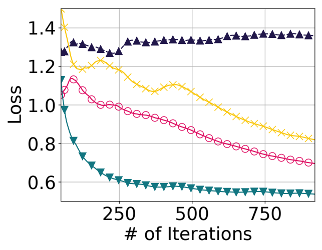
We thank IJCAI 2019 reviewers for their constructive feedback. This work is supported by the US National Science Foundation (NSF) under grants IIS-1652107, IIS-1763620, and CCF-1750886.



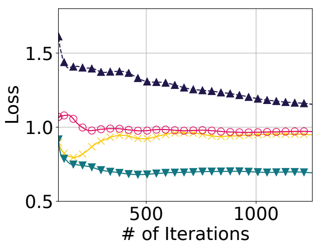
(a) wdbc



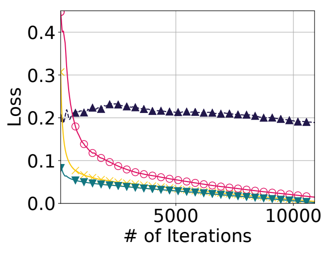
(b) australian



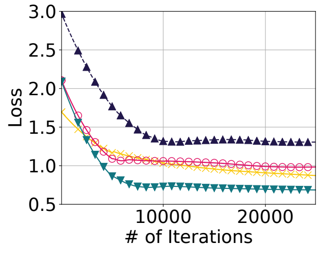
(c) dna



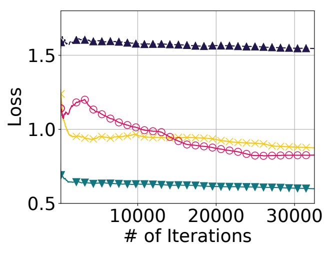
(d) svmguide3



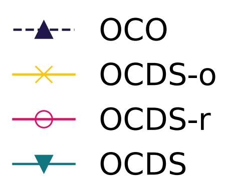
(e) HAPT



(f) IMDB



(g) a8a



legend

Figure 2: The trends of average cumulative losses of OCDS and three baseline algorithms on 7 datasets.

References

[Baig and Gholamhosseini, 2013] Mirza Mansoor Baig and Hamid Gholamhosseini. Smart health monitoring systems: an overview of design and modeling. *Journal of medical systems*, 37(2):9898, 2013.

[Beyazit *et al.*, 2018] Ege Beyazit, Matin Hosseini, Anthony Maida, and Xindong Wu. Learning simplified decision boundaries from trapezoidal data streams. In *ICANN*, pages 508–517. Springer, 2018.

[Beyazit *et al.*, 2019] Ege Beyazit, Jeevithan Alagurajah, and Xindong Wu. Online learning from data streams with varying feature spaces. In *AAAI*, 2019.

[Cesa-Bianchi and Lugosi, 2006] Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006.

[Chen *et al.*, 2005] Jinxiu Chen, Donghong Ji, Chew Lim Tan, and Zhengyu Niu. Unsupervised feature selection for relation extraction. In *CVPR*, 2005.

[Crammer *et al.*, 2006] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7(Mar):551–585, 2006.

[Dua and Karra Taniskidou, 2017] Dheeru Dua and Efi Karra Taniskidou. UCI machine learning repository, 2017.

[Duda *et al.*, 2012] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.

[Gama and Rodrigues, 2009] João Gama and Pedro Pereira Rodrigues. An overview on mining data streams. In *Foundations of Computational, IntelligenceVolume 6*, pages 29–45. Springer, 2009.

[Gerven *et al.*, 2009] Marcel V Gerven, Botond Cseke, Robert Oostenveld, and Tom Heskes. Bayesian source localization with the multivariate laplace prior. In *NeurIPS*, pages 1901–1909, 2009.

[He *et al.*, 2019] Yi He, Baijun Wu, Di Wu, Ege Beyazit, Sheng Chen, and Xindong Wu. Supplementary Material of "Online Learning from Capricious Data Streams: A Generative Approach". 10.6084/m9.figshare.8316641.v1, 2019.

[Hou *et al.*, 2017] Bo-Jian Hou, Lijun Zhang, and Zhi-Hua Zhou. Learning with feature evolvable streams. In *NeurIPS*, pages 1417–1427, 2017.

[Huang *et al.*, 2018] Sheng-Jun Huang, Miao Xu, Ming-Kun Xie, Masashi Sugiyama, Gang Niu, and Songcan Chen. Active feature acquisition with supervised matrix completion. *SIGKDD*, 2018.

[Li *et al.*, 2017] Jundong Li, Jiliang Tang, and Huan Liu. Reconstruction-based unsupervised feature selection: an embedded approach. In *IJCAI*, 2017.

[Liu *et al.*, 2014] Lingqiao Liu, Chunhua Shen, Lei Wang, Anton Van Den Hengel, and Chao Wang. Encoding high dimensional local features by sparse coding based fisher vectors. In *NeurIPS*, pages 1143–1151, 2014.

[Maas *et al.*, 2011] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *ACL*, pages 142–150, Portland, Oregon, USA, June 2011.

[Masud *et al.*, 2013] Mohammad M Masud, Qing Chen, Latifur Khan, Charu C Aggarwal, Jing Gao, Jiawei Han, Ashok Srivastava, and Nikunj C Oza. Classification and adaptive novel class detection of feature-evolving data streams. *IEEE Transactions on Knowledge and Data Engineering*, 25(7):1484–1497, 2013.

[Nguyen *et al.*, 2017] Tu Dinh Nguyen, Trung Le, Hung Bui, and Dinh Q Phung. Large-scale online kernel learning with random feature reparameterization. In *IJCAI*, pages 2543–2549, 2017.

[Pan *et al.*, 2010] Sinno Jialin Pan, Qiang Yang, et al. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[Park and Lee, 2005] Hyun J Park and Te W Lee. Modeling nonlinear dependencies in natural images using mixture of laplacian distribution. In *NeurIPS*, pages 1041–1048, 2005.

[Sáez *et al.*, 2014] José A Sáez, Mikel Galar, Julián Luengo, and Francisco Herrera. Analyzing the presence of noise in multi-class problems: alleviating its influence with the one-vs-one decomposition. *Knowledge and information systems*, 38(1):179–206, 2014.

[Sun, 2013] Shiliang Sun. A survey of multi-view machine learning. *Neural Computing and Applications*, 23(7-8):2031–2038, 2013.

[Tseng and Yun, 2009] Paul Tseng and Sangwoon Yun. A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programming*, 117(1-2):387–423, 2009.

[Xu, 2011] Jianhua Xu. An extended one-versus-rest support vector machine for multi-label classification. *Neurocomputing*, 74(17):3114–3124, 2011.

[Zhang *et al.*, 2013] Wei Zhang, Wei Feng, and Jianyong Wang. Integrating semantic relatedness and words' intrinsic features for keyword extraction. In *IJCAI*, pages 2225–2231, 2013.

[Zhang *et al.*, 2016a] Qin Zhang, Peng Zhang, Guodong Long, Wei Ding, Chengqi Zhang, and Xindong Wu. Online learning from trapezoidal data streams. *IEEE Transactions on Knowledge and Data Engineering*, 28(10):2709–2723, 2016.

[Zhang *et al.*, 2016b] Xianchao Zhang, Xiaotong Zhang, and Han Liu. Self-adapted multi-task clustering. In *IJCAI*, pages 2357–2363, 2016.

[Zhou *et al.*, 2012] Guanyu Zhou, Kihyuk Sohn, and Honglak Lee. Online incremental feature learning with denoising autoencoders. In *AISTATS*, pages 1453–1461, 2012.

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