Online Learning from Data Streams with Varying Feature Spaces

Ege Beyazit
University of Louisiana at Lafayette
Lafayette, LA, USA
exb6143@louisiana.edu

Jeevithan Alagurajah
University of Louisiana at Lafayette
Lafayette, LA, USA
jxa4540@louisiana.edu

Xindong Wu
University of Louisiana at Lafayette
Lafayette, LA, USA
xwu@louisiana.edu

Abstract

We study the problem of online learning with varying feature spaces. The problem is challenging because, unlike traditional online learning problems, varying feature spaces can introduce new features or stop having some features without following a pattern. Other existing methods such as online streaming feature selection (Wu et al. 2013), online learning from trapezoidal data streams (Zhang et al. 2016), and learning with feature evolvable streams (Hou, Zhang, and Zhou 2017) are not capable to learn from arbitrarily varying feature spaces because they make assumptions about the feature space dynamics. In this paper, we propose a novel online learning algorithm OLVF to learn from data with arbitrarily varying feature spaces. The OLVF algorithm learns to classify the feature spaces and the instances from feature spaces simultaneously. To classify an instance, the algorithm dynamically projects the instance classifier and the training instance onto their shared feature subspace. The feature space classifier predicts the *projection confidences* for a given feature space. The instance classifier will be updated by following the empirical risk minimization principle and the strength of the constraints will be scaled by the projection confidences. Afterwards, a feature sparsity method is applied to reduce the model complexity. Experiments on 10 datasets with varying feature spaces have been conducted to demonstrate the performance of the proposed OLVF algorithm. Moreover, experiments with trapezoidal data streams on the same datasets have been conducted to show that OLVF performs better than the state-of-the-art learning algorithm (Zhang et al. 2016).

Introduction

The goal of learning a classifier in an online setup, where the training instances arrive one by one, has been studied extensively and a wide variety of algorithms have been developed for online learning (Leite, Costa, and Gomide 2010; Yu, Neely, and Wei 2017; Agarwal, Saradhi, and Karnick 2008). These algorithms have made it possible to learn in applications where it is computationally not feasible to train over the entire dataset. Additionally, online learning algorithms have been used in applications where the data continuously grow over time and new patterns need to be extracted (Shalev-Shwartz and others 2012). However, online learning algorithms assume that the feature space they learn from remains constant. On the other hand, in a wide range of applications, feature spaces dynamically change over time. For example in a spam e-mail classification task, each e-mail can have a different set of words and words that are being used in spam e-mails can change over time. In social networks, the features provided by each user can be different than each other. In these scenarios, absence of a feature can also convey information and should not be treated as missing data. When the feature space in a data stream keeps changing, we refer the feature space as a *varying feature space*.

To enable learning from data with varying feature spaces, we propose the Online Learning from Varying Features (OLVF) algorithm, which will also be referred to as 'the algorithm' in this paper. To learn an instance classifier, the OLVF algorithm projects the existing classifier and the current instance onto their shared feature subspace, and makes a prediction. Based on the prediction, a loss will be suffered and the classifier will be updated according to an empirical risk minimization principle with adaptive constraints. The algorithm re-weights the constraints based on the confidence of the classifier and the training instance's projection given the feature space. To learn a feature space classifier, the algorithm follows a maximum likelihood principle, using the current feature space information and the output of the instance classifier. After each update, a sparsity step is applied to decrease the classifier size. The contributions of this paper can be listed as follows:

- The problem of real-time learning from data where the feature space arbitrarily changes has been studied.
- A new learning algorithm, Online Learning from Varying Features (OLVF), that is capable of learning from varying feature spaces on a single pass is proposed.
- Performance of the OLVF algorithm is empirically validated on 10 datasets in two different scenarios: data with varying feature spaces and trapezoidal data streams.

Related Work

Because of the dynamic nature of its feature space and the streaming instances, learning from data with varying feature spaces can be related to online learning, online learning from trapezoidal data streams and learning with feature evolvable streams.

Copyright © 2019, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Online Learning

The online learning is performed in consecutive rounds and at each round, the learner is given a training example which can be seen as a question with its answer hidden. The learner makes a prediction on this question and the answer is revealed. Based on the difference between the learner's prediction and the actual answer, a loss is suffered. The aim of the learner is to minimize the total loss through all rounds. Many different algorithms have been developed for learning in online setting (Tsang, Kocsor, and Kwok 2007; Roughgarden and Schrijvers 2017; Domingos and Hulten 2000; Kuzborskij and Cesa-Bianchi 2017). These algorithms can be broadly grouped into two categories: first-order and second-order methods. First-order algorithms use first-order derivative information for updates (Rosenblatt 1958; Ying and Pontil 2008). Many first order algorithms (Crammer et al. 2006; Gentile 2001; Kivinen, Smola, and Williamson 2004) use the maximum margin principle to find a discriminant function that maximizes the separation between different classes. By taking the curvature of the space into account, second-order methods improve the convergence and are able to minimize a quadratic function in a finite number of steps (Cesa-Bianchi, Conconi, and Gentile 2005; Crammer, Dredze, and Pereira 2009; Crammer, Kulesza, and Dredze 2009; Wang, Zhao, and Hoi 2016; Orabona and Crammer 2010). Second-order methods are more noise sensitive and more expensive than first-order methods, because they require the calculation of the second derivatives. Traditional online learning methods can not learn from varying feature spaces since they assume the feature space they learn from remains the same.

Online Learning from Trapezoidal Data Streams

Online learning from trapezoidal data streams (Zhang et al. 2015; 2016) is closely related to data streams with varying feature spaces. Trapezoidal data streams are streams with a doubly growing nature: the number of instances and the number of features provided by instances can grow simultaneously. In trapezoidal streams, the current training instance either has the same features as previous instances or additional new features. In other words the feature space at each iteration contains the feature spaces from previous iterations. The proposed algorithms $OL_{SF}-I$ and $OL_{SF}-II$ learn a linear classifier from trapezoidal data streams by following the passive-aggressive update rule. At each iteration, they receive a training instance and try to make a prediction. If the prediction is correct, they do not do any classifier update, else they update their classifier with the weights that minimize the error and are close to the previous classifier weights. While doing this, they also learn additional weights for the new features, if any introduced by the current training instance. Moreover, sparsity is introduced by applying projection and truncation to the classifier at each round, to decrease the model complexity and improve generalization. However OL_{SF} and its variants can not handle data with varying feature spaces, where the feature space can also shrink and the instances can have completely different sets of features from each other.

Learning with Feature Evolvable Streams

Learning with feature evolvable streams (Hou, Zhang, and Zhou 2017) tackle the problems of classification and regression for data streams where features vanish and new features occur. However, they assume that the feature space evolves in a strictly sequential manner: after the new features arrive, there exists a time period that both the new and old features are available, before some of the old features vanish. Based on this assumption, they use the overlapping time period to learn a projection matrix from old features to the new ones by minimizing a least squares loss. Then to learn a predictor, they propose two methods: weighted combination and dynamic selection. Weighted combination follows an ensemble strategy by combining the outputs of a set of predictors with weights based on exponential of the cumulative loss. On the other hand, dynamic selection selects the output of the predictor of larger weight with higher probability. Even though learning with feature evolvable streams focuses on feature spaces that can dynamically grow and shrink, the assumption of sequential evolution is too strong and application specific. If there is no overlap period of or linear relationship between the old and new features, the projection does not carry any useful information. Consequently, the algorithm reduces to an ensemble based online learning method with a runtime overhead. Moreover, it is not always computationally feasible to learn a projection matrix and multiple predictors if the data stream is high dimensional.

Learning from data with varying feature spaces is more challenging than the problems stated above since the classifier needs to adapt itself to a dynamically changing feature space without any assumptions on the dynamics of the feature space or interactions of features. The algorithm needs to continuously learn from data while extracting knowledge when arbitrary features stop being generated as well as new features arrive, in one pass. The existing learning methods are not able to solve learning from varying feature spaces since none of them handle such dynamic and unconstrained feature spaces while learning a classifier.

Preliminaries

In this section, we first present two preliminary concepts that we make use of to design our algorithm: online learning of linear classifiers, and soft-margin classification. Then, we discuss the problem setting for learning binary classifiers from varying feature spaces.

Online Learning of Linear Classifiers

We consider binary classification using linear classifiers. At each round t, the algorithm receives an instance $\mathbf{x}_t \in \mathbb{R}^d$ and tries to predict the true label $y_t \in \{-1, +1\}$ of the instance by using its classifier $w_t \in \mathbb{R}^d$. Prediction is done by using the function $\hat{y}_t = sign(\mathbf{x}_t \cdot \mathbf{w}_t)$. After prediction, the true label of the instance is revealed. Based on the difference between prediction $\hat{y}_t$ and true label y_t, a loss $l(y_t, \hat{y}_t)$ is suffered. One of the widely used loss functions is the Hinge loss (Gentile and Warmuth 1999; Wu and Liu 2007; Bartlett and Wegkamp 2008) defined as $l(y_t, \hat{y}_t) = max\{0, 1-y(\mathbf{x}_t \cdot \mathbf{w}_t)\}$. It is based on the margin

between the example $\mathbf{x_t}$ and the classifier $\mathbf{w_t}$. The margin is calculated by $y(\mathbf{x}_t \cdot \mathbf{w}_t)$. Because online learning is an incremental task, it is important to minimize the loss while keeping the change to the current model minimum, in order to preserve the knowledge from previous instances. Moreover when the data is noisy, forcing the classifier to correctly predict for every instance leads to overfitting and poor generalization. Therefore, it is more preferable to learn a classifier which is able to separate the bulk of the data while ignoring the noise. To accomplish this, a soft decision margin (Shawe-Taylor and Cristianini 2002) is used to let the classifier make a few mistakes. Many online learning algorithms combine the constraints stated above and formulate the learning of the weights as an optimization problem:

$$w_{t+1} = \underset{\substack{w: \\ l(y_t, \hat{y}_t) \leq \xi}}{\operatorname{argmin}} \quad \frac{1}{2} \|w - w_t\|^2 + C\xi. \tag{1}$$

By introducing a slack variable, Equation (1) adds nonlinearity to the discriminator to allow a certain amount of error to be made. This error is bounded by ξ. The parameter C adjusts the slackness of the constraint. Since real life data is noisy, we use the soft-margin approach to model learning from varying feature spaces.

The Problem Setting and Notation

A feature space that keeps changing on different instances in a data stream is referred to as a varying feature space. We consider the problem of learning a classifier from data with varying feature spaces, by doing single pass over each instance. Let input to the learning algorithm consist of a sequence of examples (x_t, y_t) where $t = 1, ..., T$. Each instance $x_t \in \mathbb{R}^{x_t}$ is a vector that contains an arbitrary set of features and $y_t \in \{-1, +1\}$ is a binary class label for all t. Let w_t be the weight vector for the instance classifier at round t and $\hat{y}_t = sign(x_t \cdot w_t)$ be the prediction of the instance classifier for the instance x_t. Let $\bar{w}_t$ be the weight vector for the feature space classifier at round t and $p_t = \sigma(\mathbb{R}^{x_t} \cdot \bar{w}_t)$ is the prediction of the feature space classifier for the feature space $\mathbb{R}^{x_t}$.

In data with varying feature spaces, the feature space can dynamically change; new features can be introduced or some of the existing features can cease to exist. At each round t, when an instance arrives, we divide the explored feature space into three groups: existing, shared and new features. The existing features are only contained by the current classifiers, the new features are only contained by the current instance, and the shared features are the intersection of current classifiers' and the current instance's feature spaces. To make a prediction, the learning algorithm projects its current classifiers and the current training instance onto their shared feature space. We denote the projection of a classifier at round t onto the existing feature space as ${w_t}^e$, shared feature space as ${w_t}^s$ and the new feature space as ${w_t}^n$. This notation also applies for the projections of other vectors such as training instances (${x_t}^e$, ${x_t}^s$, ${x_t}^n$), and feature space classifiers ($\bar{w}_t^e$, $\bar{w}_t^s$, $\bar{w}_t^n$).

We define a feature space $\mathbb{R}$ as a binary *bag of features* vector, where each feature in the shared and new feature spaces is represented by a one and, each feature in the existing feature space represented by a zero.

Online Learning from Varying Feature Spaces

In this section, we explain the building blocks of the OLVF algorithm and discuss the motivation behind their design. Then, we derive the soft-margin classifier update rules for the instance and feature space classifiers in a binary classification setting. Note that the algorithm can easily be generalized to a multiclass setting by converting the problem to multiple binary classification problems (Bolon-Canedo, Sanchez-Marono, and Alonso-Betanzos 2011), using the One vs Rest or One vs One (Sáez et al. 2014; Xu 2011) strategies. Finally, we combine the building blocks to form the main steps and discuss the running time complexity of the OLVF algorithm.

Learning to Predict Projection Confidences

In traditional online learning problems the feature space remains constant. When a classifier makes a poor prediction, it needs to be updated aggressively. However in data with varying feature spaces, our instance classifier can make poor predictions because of the unfair conditions caused by varying feature spaces: the instance classifier often needs to predict the labels for instances that belong to feature spaces different than its own. For example, when our learner receives an instance that does not have some of the features included by the current instance classifier, the learner will not able to use its full potential, since the instance classifier will only be using the features in the shared feature space. Similarly, if the learner receives an instance with new features, the instance loses the information provided by the new feature space. Hence, if the instance classifier makes a bad prediction, the reason can also be the high difference between the feature spaces of itself and the current instance.

We measure the loss of information after projecting a vector onto a feature space with *projection confidence*: probability that the instance classifier makes a correct prediction given the vectors projected feature space. We learn to estimate projection confidences by training a feature space classifier $\bar{w}_t$. Let $I(y, \hat{y})$ be an indicator function defined as:

$$I(y, \hat{y}) = \begin{cases} 1, & \text{if } y = \hat{y} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

We model the probability that the instance classifier w_t makes a correct prediction, $y = \hat{y}$, given the feature space $\mathbb{R}^{x_t}$ parameterized by the feature space classifier $\bar{w}_t$ with the likelihood function

$$\begin{aligned} &P(I(y_t, \hat{y}_t) | \mathbb{R}^{x_t}; \bar{w}_t) = \\ &\quad \sigma(\mathbb{R}^{x_t^e} \cdot \bar{w}_t^e + \mathbb{R}^{x_t^s} \cdot \bar{w}_t^s + \mathbb{R}^{x_t^n} \cdot \bar{w}_t^n)^{I(y_t, \hat{y}_t)} + \\ &\quad \sigma(\mathbb{R}^{x_t^e} \cdot \bar{w}_t^e + \mathbb{R}^{x_t^s} \cdot \bar{w}_t^s + \mathbb{R}^{x_t^n} \cdot \bar{w}_t^n)^{1 - I(y_t, \hat{y}_t)}. \end{aligned} \tag{3}$$

After taking the negative logarithm of the likelihood function above, we rearrange and define our loss function to train the feature space classifier $\bar{w}_t$ as:

$$\begin{aligned} &\bar{l}_t(\mathbb{R}^{x_t}, I(y_t, \hat{y}_t), \bar{w}_t) = \\ &\qquad \log(1 + e^{-I(y_t, \hat{y}_t) \bar{w}_t^s \mathbb{R}^{x_t^s}} e^{-I(y_t, \hat{y}_t) \bar{w}_t^n \mathbb{R}^{x_t^n}}) \end{aligned} \tag{4}$$

Since we learn the feature space classifier in an online setting, merely minimizing the loss function can lead to a poor performance. As the learner receives new instances and updates the feature space classifier, it is important to preserve the knowledge of previously seen feature spaces. Moreover when a set of new features arrive and need to be added to the current feature space classifier, there can be infinitely many solutions because of the lack of information about those new features. To avoid overfitting, we need to learn the smallest weights possible that minimize the loss. Also, because real-life data is noisy, we want our algorithm to follow a soft-margin strategy to avoid poor generalization. As a result, we formulate the problem of learning the feature space classifier as constrained optimization, using the loss function defined in Equation (4):

$$\bar{w}_{t+1} = \underset{\substack{\bar{w}=[\bar{w}^e,\bar{w}^s,\bar{w}^n]:\\ \bar{l}_t\leq\xi,\xi\geq 0}}{\operatorname{argmin}} \frac{1}{2}\left\|\bar{w}^e-\bar{w}_t^e\right\|^2 + \frac{1}{2}\left\|\bar{w}^s-\bar{w}_t^s\right\|^2 + \frac{1}{2}\left\|\bar{w}^n\right\|^2 + \bar{C}\xi, \quad (5)$$

Note that we use the slack variable $\bar{C}$ to bound the loss $\bar{l}_t$. We solve the optimization problem with the inequalities $\bar{l}_t \leq \xi$ and $\xi \geq 0$ defined in Equation (5) by using a Lagrangian function with K.K.T. conditions (Luo and Yu 2006):

$$L(\bar{w},\tau,\alpha,\xi) = \frac{1}{2}\left\|\bar{w}^e-\bar{w}_t^e\right\|^2 + \frac{1}{2}\left\|\bar{w}^s-\bar{w}_t^s\right\|^2 + \frac{1}{2}\left\|\bar{w}^n\right\|^2 + \bar{C}\xi + \tau(\bar{l}_t-\xi) - \alpha\xi, \quad (6)$$

where τ and α are Lagrange multipliers. Setting the derivatives of $L(\bar{w},\tau,\alpha,\xi)$ with respect to $\bar{w}^e$, $\bar{w}^s$, $\bar{w}^n$ and ξ, we obtain the following conditions:

$$\bar{w}^e = \bar{w}_t^e$$
$$\bar{w}^s = \bar{w}_t^s + \tau\frac{\partial\bar{l}_t(\mathbb{R}^{x_t},I(y_t,\hat{y}_t),\bar{w}_t)}{\partial\bar{w}^s}$$
$$w^n = -\tau\frac{\partial\bar{l}_t(\mathbb{R}^{x_t},I(y_t,\hat{y}_t),\bar{w}_t)}{\partial\bar{w}^n}$$
$$\alpha = C-\tau,$$

where

$$\frac{\partial\bar{l}_t(\mathbb{R}^{x_t},I(y_t,\hat{y}_t),\bar{w}_t)}{\partial\bar{w}^s} = -\frac{\log(e^{-I(y_t,\hat{y}_t)(\bar{w}_t^s\mathbb{R}^{x_t^s}+\bar{w}_t^n\mathbb{R}^{x_t^n})})}{\log(1+e^{-I(y_t,\hat{y}_t)(\bar{w}_t^s\mathbb{R}^{x_t^s}+\bar{w}_t^n\mathbb{R}^{x_t^n})})}I(y_t,\hat{y}_t)\mathbb{R}^{x_t^s}, \quad (7)$$

and

$$\frac{\partial\bar{l}_t(\mathbb{R}^{x_t},I(y_t,\hat{y}_t),\bar{w}_t)}{\partial\bar{w}^n} = -\frac{\log(e^{-I(y_t,\hat{y}_t)(\bar{w}_t^s\mathbb{R}^{x_t^s}+\bar{w}_t^n\mathbb{R}^{x_t^n})})}{\log(1+e^{-I(y_t,\hat{y}_t)(\bar{w}_t^s\mathbb{R}^{x_t^s}+\bar{w}_t^n\mathbb{R}^{x_t^n})})}I(y_t,\hat{y}_t)\mathbb{R}^{x_t^n} \quad (8)$$

Using the conditions and partial derivatives defined above, we obtain the following representation of our problem and find τ:

$$L(\tau) = \frac{1}{2}\tau^2\left\|\frac{\partial\bar{l}_t}{\partial\bar{w}^s}\right\|^2 + \frac{1}{2}\tau^2\left\|\frac{\partial\bar{l}_t}{\partial\bar{w}^n}\right\|^2 + \tau\bar{l}_t(\mathbb{R}^{x_t},I(y_t,\hat{y}_t),\bar{w}_t) \quad (9)$$

$$\tau = min\{C, \frac{-\bar{l}_t(\mathbb{R}^{x_t},I(y_t,\hat{y}_t),\bar{w}_t)}{\left\|\frac{\partial\bar{l}_t}{\partial\bar{w}^s}\right\|^2+\left\|\frac{\partial\bar{l}_t}{\partial\bar{w}^n}\right\|^2}\}. \quad (10)$$

To make a prediction given an instance x_t, the learner makes projections of both the instance classifier and the instance onto the current shared feature space. We use the feature space classifier $\bar{w}_t$ to estimate the projection confidences and adaptively re-weight the different parts of the objective function corresponding to the different pieces of the instance classifier weights $w_t = [w_t^e, w_t^s, w_t^n]$. This re-weighting strategy helps to adjust the priorities of each constraint in the objective function, scales the regularization weights for each instance x_t, and improves the learning convergence.

Learning to Classify Instances from Varying Feature Spaces

In order to adapt the instance classifier to the dynamic nature of the varying feature spaces, we modify the Hinge loss by using the projections of the instance classifier and the instance:

$$l_t = max\{0, 1-y({x_t}^s\cdot w^s)-y({x_t}^n\cdot w^n)\}. \quad (11)$$

Motivated by the same reasons discussed for the objective function defined in Equation (5), we formulate the problem of learning the instance classifier from varying features as

$$w_{t+1} = \underset{\substack{w=[w^e,w^s,w^n]:\\ l_t\leq\xi,\xi\geq 0}}{\operatorname{argmin}} \frac{1}{2}\left\|w^e-{w_t}^e\right\|^2 + \frac{1}{2}p_t^w\left\|w^s-{w_t}^s\right\|^2 + \frac{1}{2}p_t^x\left\|w^n\right\|^2 + C\xi, \quad (12)$$

where $p_t^w = \sigma(\mathbb{R}^{w_t}\cdot\bar{w}_t)$ is the projection confidence of the instance classifier w_t and $p_t^x = \sigma(\mathbb{R}^{x_t}\cdot\bar{w}_t)$ is the projection confidence of the instance x_t. Note that the objectives for w^s and w^n are weighted using these projection confidences. When the projection confidence of the classifier onto the shared subspace p_t^w is close to 1, then the shared feature space is not very different than the feature space of the current classifier. Therefore, the objective of minimizing $\left\|w^s-{w_t}^s\right\|^2$ is important in order to remember the information received from previous instances. Moreover if the projection confidence of the training instance onto the shared subspace p_t^x is close to 1, the objective of minimizing $\left\|w^n\right\|^2$ is easier and can be emphasized more to avoid overfitting. On the other hand low projection confidences imply a sudden change in the feature space. In this case, the objective function must focus on minimizing the loss more than trying to keep the classifier weights small or close to the previous weights. We solve the optimization problem with inequalities defined above by using another Lagrangian function with K.K.T. conditions and obtain the following update

rules:

$$w^e = {w_t}^e$$
$$w^s = {w_t}^s + \tau p_t^w y_t {x_t}^s$$
$$w^n = \tau p_t^x y_t {x_t}^n$$
$$\alpha = C - \tau.$$

Using these conditions, we obtain the following representation of our problem and find τ:

$$L(\tau) = \frac{1}{2}\tau^2 (p_t^w)^2 \left\|{x_t}^s\right\|^2 + \frac{1}{2}\tau^2 (p_t^x)^2 \left\|{x_t}^n\right\|^2 + \tau(1 - y_t({w_t}^s \cdot {x_t}^s)), \quad (13)$$

$$\tau = min\{C, \frac{-l_t}{\left\|x_t\right\|^2}\}. \quad (14)$$

After rearranging, we can define the update strategy as:

$$w_{t+1} = [w_{t+1}^e, w_{t+1}^s, w_{t+1}^n] = [w^e, {w_t}^s + min\{Cp_t^w y_t {x_t}^s, \frac{p_t^w l_t y_t {x_t}^s}{\left\|x_t\right\|^2}\}, min\{Cp_t^x y_t {x_t}^n, \frac{p_t^x l_t y_t {x_t}^n}{\left\|x_t\right\|^2}\}]. \quad (15)$$

Model Sparsity

Because the feature space can dynamically change, the number of features kept in the classifiers are not bounded. In its current design, the learner keeps remembering even if the features stop being generated. This can act as a bias on feature space and instance classifier weights when the features are reintroduced. Moreover, when vanishing features re-emerge, there is a possibility that their meanings are different. As a result, this bias can lead to a poor performance. Furthermore, because there is no feature selection mechanism in the learner, the learner keeps using even the least important features. Therefore, to improve the memory usage and running time efficiency, we do not want to keep the classifier weights for every feature received by the learner. Simply truncating the smallest weights from the instance classifier leads to poor performance because it introduces a sudden change to the result of the dot product. On the other hand, the feature space classifier can tolerate this change because of the saturating sigmoid function. Therefore to sparsify the instance classifier, we introduce the following projection step before truncation:

$$w_t := min\{1, \frac{\lambda}{w_t \cdot \bar{w}_t}\} w_t, \quad (16)$$

where λ is a regularization parameter. With a ratio of B, we truncate the smallest elements from the classifiers. Since $\bar{w}_t$ carries the information of which features contribute to the classification task, we use it to scale the weights of the instance classifier. This strategy helps to identify and truncate the features that are either redundant or vanishing.

Algorithm 1: The OLVF Algorithm

Input : $C, \bar{C} > 0$: Loss bounding parameters
$\lambda > 0$: Regularization parameter
$B \in (0, 1]$: Proportion of selected features
Initialize: $w_1 = (0, ..., 0) \in \mathbb{R}^{x_1}$
Initialize: $\bar{w}_1 = (0, ..., 0) \in \mathbb{R}^{x_1}$
for *t = 1,2,... T* **do**
  Receive the current instance $x_t \in \mathbb{R}^{x_t}$
  Identify the shared feature space $\mathbb{R}^s = \mathbb{R}^{w_t \cap x_t}$
  Project w_t, x_t onto $\mathbb{R}^s$: ${w_t}^s, {x_t}^s$
  Predict the class label: $\hat{y}_t = sign({w_t}^s \cdot {x_t}^s)$
  Receive the correct label: $y_t \in \{+1, -1\}$
  Update feature space classifier with $y_t, \hat{y}_t, \mathbb{R}^{x_t}$; using Equation (7).
  Predict the projection confidences:
  $p_t^w = \sigma(\mathbb{R}^{w_t} \cdot \bar{w}_t)$
  $p_t^x = \sigma(\mathbb{R}^{x_t} \cdot \bar{w}_t)$
  Update the instance classifier with x_t, y_t, p_t^w, p_t^x; using Equation (15).
  Project w_{t+1} using Equation (16) and λ.
  Truncate w_{t+1} and $\bar{w}_{t+1}$ using B.
end

Time Complexity of OLVF

The pseudocode of our online learning from varying features (OLVF) algorithm is shown in Algorithm 1. The time complexity of the OLVF algorithm is as follows. Assuming at round t, $|w_t| = |\bar{w}_t|$ is the number of features in the current classifiers, $|x_t|$ is the number of features arrived with the training instance, and $|\mathbb{R}^s|$ is the number of features in the shared feature space. Since the operations are being done for both the weight vectors and the current training instance, the time complexity for identifying the shared feature space and projecting the weight vectors with the training instance onto their shared feature space is O($|w_t| + |x_t|$). The time complexity for making a prediction, calculating the projection confidences and suffering losses is O($|\mathbb{R}^s|$), because all of these steps use the shared feature space. Finally, the time complexity for the sparsity step is O($|w_t|$), since the size of the instance classifier is always equal to the size of the feature space classifier. For a single round, the worst case time complexity of the OLVF algorithm is $O(|w_t| + |x_t| + |\mathbb{R}^s|)$. Because $|\mathbb{R}^s|$ is always less than $|w_t| + |x_t|$, we can further simplify the complexity as $O(|w_t| + |x_t|)$. Considering the fact that at any round t the feature sets of w_t and x_t can be completely disjoint, we do not apply further simplification for the time complexity. In other words, the running time of the algorithm linearly scales with the number of features that are being used at each round t.

Experiments

In this section we empirically evaluate the performance of the OLVF algorithm in three different scenarios: data with simulated varying feature spaces, simulated trapezoidal data streams (Zhang et al. 2016) and real-life varying feature spaces. In each scenario, the instances are provided to the

Table 1: Numbers of samples and features of 10 datasets.

Dataset	# Samples	# Features
wpbc	198	34
ionosphere	351	35
wdbc	569	31
wbc	699	10
german	1,000	24
svmguide3	1,234	21
spambase	4,601	57
magic04	19,020	10
imdb	25,000	7500
a8a	32,561	123

classifier incrementally and only a single pass is allowed. Similarly, the classifier does not have any initial knowledge about the full feature space. We use 9 different UCI datasets to simulate these scenarios. Additionally, we demonstrate the effectiveness of the proposed sparse strategy. Finally we evaluate the performance of OLVF using the real-world dataset IMDB movie reviews (Maas et al. 2011). The numbers of instances and features for each dataset are listed in Table 1. In all experiments, we measure the performance in terms of the average prediction accuracy on 20 random permutations of each dataset. The parameters C and $\bar{C}$ are chosen using grid search.

Experiments on Varying Feature Spaces

We simulate data with varying feature spaces by randomly removing features from every training instance. We denote the ratio of features removed from each training instance as *removing ratio (Rem.)*. At each round, we apply a random permutation to the complete dataset and randomly remove the features. We conduct two different types of experiments for varying features. The first set of experiments measure the performance of the proposed algorithm in varying feature spaces using removing ratios of 0.25, 0.5 and 0.75. The second set of experiments show the trade-off between the accuracy and the sparsity of the classifier, in data with varying feature spaces using 0.25 as the removing ratio. *1) Experiments using different removing ratios:* We test the performance of the proposed algorithm using 3 different removing ratios: 0.25, 0.5 and 0.75. If the removing ratio for a varying feature space is 0.25, then $1/4$ of the features are randomly removed from each training instance, and so on. Table 2 shows the average number of wrong predictions made by OLVF with different removing ratios on 9 UCI datasets. For all datasets, OLVF achieves higher accuracy on the experiments with smaller removing ratio. This is expected since a small removing ratio means a high number of features sent to the classifier. *2) Experiments on sparse and non-sparse OLVF:* We observe the trade-off between using sparse classifiers to decrease the time and space usage, and using the full feature space to increase the accuracy. To demonstrate the effect of the sparsity step, we set the removing ratio to 0.25 while simulating the varying feature spaces. Table 3 shows the average number of errors made by OLVF and sparse OLVF. Table 4 shows the number of features

Table 2: Average number of errors made by $OLVF$ on 9 UCI datasets in simulated varying feature spaces.

Rem.	**german**	**ionosphere**	**spambase**
0.25	333.4 ± 9.7	77.2 ± 7.1	659.8 ± 14.5
0.5	350.9 ± 7.8	79.5 ± 7.4	864 ± 20.6
0.75	365.15 ± 3.6	79.7 ± 5.9	1375.8 ± 21.5
Rem.	**magic04**	**svmguide3**	**wbc**
0.25	6152.4 ± 54.7	346.4 ± 11.6	25.3 ± 1.4
0.5	6775 ± 27.4	367.2 ± 11.9	60.6 ± 5.3
0.75	7136.4 ± 2.7	371.2 ± 11.7	123.1 ± 3.6
Rem.	**wpbc**	**wdbc**	**a8a**
0.25	88.5 ± 5.8	40.8 ± 3.5	8993.8 ± 40.3
0.5	90.2 ± 5.1	55.2 ± 5.4	9585.8 ± 53.8
0.75	107.7 ± 7.7	202.85 ± 7.5	12453 ± 74.4

Table 3: Average number of errors made by non-sparse $OLVF$ and sparse $OLVF$ on 9 UCI datasets in simulated varying feature spaces.

Alg	**german**	**ionosphere**	**spambase**
$ns.$	333.4 ± 9.7	77.2 ± 7.1	659.8 ± 14.5
$s.$	358 ± 9.2	79 ± 15.9	825.1 ± 42.7
Alg	**magic04**	**svmguide3**	**wbc**
$ns.$	6154 ± 47.1	367.4 ± 11.6	25 ± 1.4
$s.$	6621.6 ± 46.6	361.4 ± 20.7	32.3 ± 2.6
Alg	**wpbc**	**wdbc**	**a8a**
$ns.$	90.2 ± 5.1	39.6 ± 5.2	8933.2 ± 28.5
$s.$	97.4 ± 4.6	131 ± 7.9	8588.6 ± 575.3

used by the non-sparse and sparse versions of the OLVF algorithm, along with their parameter settings. We can see that with datasets german, ionosphere, spambase, magic04, svmguide3, wbc and a8a sparse and non-sparse versions of the proposed algorithm perform similarly, while the sparse version of the algorithm uses a smaller subset of the provided feature space. With the wpbc dataset, the sparse version of the algorithm demonstrates a more consistent performance because the sparse strategy helps to reduce the noise and avoid overfitting. In the wdbc dataset, the sparse version of the proposed algorithm performs similarly to the non-sparse version. However, it follows a lower accuracy trend by $\sim 10\%$ because wdbc has a relatively less number of instances and less redundant features after randomly removing 1/4 of the features to simulate varying feature spaces.

Table 4: Parameters used by the sparse OLVF algorithm.

Dataset	**B**	**C**	$\bar{\mathbf{C}}$
wbc	0.6	1	0.01
svmguide3	0.4	0.1	0.001
wpbc	0.7	0.0001	0.01
ionosphere	0.1	0.01	0.1
magic04	0.6	0.1	0.0001
german	0.5	0.01	0.001
spambase	0.3	0.01	0.001
wdbc	0.9	1	0.1
a8a	0.05	0.001	0.00001

Table 5: Average number of errors made by OL_{SF} and $OLVF$ on 9 UCI datasets in simulated trapezoidal data streams, and on the IMDB dataset.

Algorithm	**german**	**ionosphere**
OL_{SF}	385.5 ± 10.2	57.9 ± 4.7
$OLVF$	$\mathbf{329.2 \pm 9.8}$	$\mathbf{51.8 \pm 3.1}$
Algorithm	**magic04**	**svmguide3**
OL_{SF}	6147.4 ± 65.3	361.7 ± 29.7
$OLVF$	$\mathbf{5784.0 \pm 52.7}$	$\mathbf{351.6 \pm 25.9}$
Algorithm	**wpbc**	**wdbc**
OL_{SF}	87.9 ± 5.6	52.9 ± 4.5
$OLVF$	$\mathbf{78.2 \pm 5.5}$	$\mathbf{45.4 \pm 2.9}$
Algorithm	**spambase**	**wbc**
OL_{SF}	993.5 ± 25	48.1 ± 12.6
$OLVF$	$\mathbf{825.8 \pm 20.2}$	$\mathbf{31.1 \pm 2.8}$
Algorithm	**a8a**	**imdb**
OL_{SF}	9420.4 ± 549.9	7851.0 ± 51.4
$OLVF$	$\mathbf{8649.8 \pm 526.7}$	$\mathbf{4474.2 \pm 39.1}$

Experiments on Trapezoidal Data Streams

We compare the prediction accuracy of OLVF with OL_{SF} algorithms (Zhang et al. 2016) on trapezoidal data streams. For each dataset, we use the version of OL_{SF} that performs the best to compare with our algorithm. For the sparsity step, we use the same λ and B parameters as in (Zhang et al. 2016). We used grid search to find the best C and $\bar{C}$ for each dataset. To simulate trapezoidal data streams, we split each dataset into 10 chunks where the number of features included by each chunk increases as the data flows in. For example, instances in the first chunk have the first 10 percent of features, instances in the second chunk have the first 20 percent of features and so on. Table 5 shows the average numbers of errors with variances of the OLVF and the OL_{SF} algorithms in trapezoidal streams. Note that because the existing and shared feature spaces in trapezoidal data streams never change, the only component that is effective while reweighting the objective function is the projection confidence of the current training instance. From these experiments, we observe that our OLVF achieves higher prediction accuracy while learning classifiers with the same sparsity as OLSF, because of its feature space adaptive constraint re-weighting strategy. In Table 5 we observe that in addition to lower numbers of errors in 9 UCI datasets, the standard deviation of the errors made by OLVF in 20 rounds is also lower than the OLSF algorithms. This shows that OLVF has consistently better performance than the OLSF algorithms on the 9 UCI datasets.

Application to Real-World Varying Feature Spaces

In IMDB Movie Reviews dataset, the task is to classify each movie review, provided in raw text, into positive or negative sentiment. Each new movie review can include words that the learner have never seen, or exclude the words exist in the learners feature space. Hence, we can formulate the problem as learning from varying feature spaces and use OLVF. The problem can also be seen as learning from trapezoidal data streams by assuming non-existing features are missing

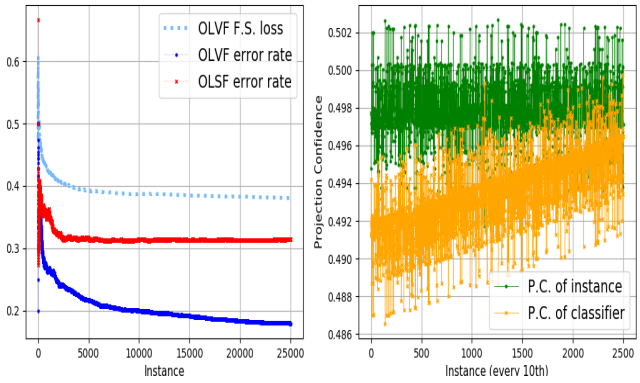

Figure 1: Mean error rates of $OLVF$ and OL_{SF} algorithms as the data streams in (left), and projection confidences estimated by the feature space classifier (right).

data. We evaluate these two approaches by comparing the prediction accuracies of OLVF and OL_{SF} algorithms. For both algorithms, we set B's to 0.1 and find their best setting for the C and $\bar{C}$ parameters by using grid search. We set the C to 0.1, and $\bar{C}$ to 10^{-5}.

Table 5 shows that average number of wrong predictions made by OLVF is lower than OL_{SF}. Moreover, Figure 1 (left) shows that OLVF has a faster learning trend and higher accuracy than OL_{SF} because of its feature space adaptive constraint weighting strategy. Figure 1 (left) also shows that the loss suffered by the feature space classifier of OLVF decreases as the data streams in. This indicates that the model learns how to classify feature spaces as it receives more instances. Figure 1 (right) shows the change of projection confidence predictions made by the feature space classifier. Note that the projection confidences randomly fluctuate because each instance comes from an arbitrary feature space. Additional to the fluctuations, the projection confidence of the instance classifier has an increasing trend. This is because the instance classifier learns new features and new information relevant to its existing features, therefore becomes more confident against unknown features as the data streams in.

Conclusion

In this paper we explored a new problem of online learning from data with varying feature spaces, where the feature space of each training instance can be arbitrarily different from other instances. We defined the concept of projection confidence for learning from varying feature spaces and derived an update rule for a projection confidence estimator. Then, we presented a new algorithm OLVF that learns feature space level and instance level classifiers from data with varying features by adaptively reweighting the constraints of the objective function with projection confidences. We evaluated the proposed OLVF algorithms on 9 UCI datasets and a real life dataset. Additionally, we compared the sparse and non-sparse versions of OLVF algorithm. Finally, we showed that our algorithm outperforms the state-of-the-art OL_{SF} algorithms.

Acknowledgments

This research is supported by the US National Science Foundation (NSF) under grants 1652107 and 1763620.

References

Agarwal, S.; Saradhi, V. V.; and Karnick, H. 2008. Kernel-based online machine learning and support vector reduction. *Neurocomputing* 71(7-9):1230–1237.

Bartlett, P. L., and Wegkamp, M. H. 2008. Classification with a reject option using a hinge loss. *Journal of Machine Learning Research* 9(Aug):1823–1840.

Bolon-Canedo, V.; Sanchez-Marono, N.; and Alonso-Betanzos, A. 2011. Feature selection and classification in multiple class datasets: An application to kdd cup 99 dataset. *Expert Systems with Applications* 38(5):5947–5957.

Cesa-Bianchi, N.; Conconi, A.; and Gentile, C. 2005. A second-order perceptron algorithm. *SIAM Journal on Computing* 34(3):640–668.

Crammer, K.; Dekel, O.; Keshet, J.; Shalev-Shwartz, S.; and Singer, Y. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research* 7(Mar):551–585.

Crammer, K.; Dredze, M.; and Pereira, F. 2009. Exact convex confidence-weighted learning. In *Advances in Neural Information Processing Systems*, 345–352.

Crammer, K.; Kulesza, A.; and Dredze, M. 2009. Adaptive regularization of weight vectors. In *Advances in Neural Information Processing Systems*, 414–422.

Domingos, P., and Hulten, G. 2000. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, 71–80. ACM.

Gentile, C., and Warmuth, M. K. 1999. Linear hinge loss and average margin. In *Advances in Neural Information Processing Systems*, 225–231.

Gentile, C. 2001. A new approximate maximal margin classification algorithm. *Journal of Machine Learning Research* 2(Dec):213–242.

Hou, B.-J.; Zhang, L.; and Zhou, Z.-H. 2017. Learning with feature evolvable streams. In *Advances in Neural Information Processing Systems*, 1417–1427.

Kivinen, J.; Smola, A. J.; and Williamson, R. C. 2004. Online learning with kernels. *IEEE Transactions on Signal Processing* 52(8):2165–2176.

Kuzborskij, I., and Cesa-Bianchi, N. 2017. Nonparametric online regression while learning the metric. In *Advances in Neural Information Processing Systems*, 667–676.

Leite, D.; Costa, P.; and Gomide, F. 2010. Evolving granular neural network for semi-supervised data stream classification. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, 1–8. IEEE.

Luo, Z.-Q., and Yu, W. 2006. An introduction to convex optimization for communications and signal processing. *IEEE Journal on Selected Areas in Communications* 24(8):1426–1438.

Maas, A. L.; Daly, R. E.; Pham, P. T.; Huang, D.; Ng, A. Y.; and Potts, C. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 142–150. Portland, Oregon, USA: Association for Computational Linguistics.

Orabona, F., and Crammer, K. 2010. New adaptive algorithms for online classification. In *Advances in Neural Information Processing Systems*, 1840–1848.

Rosenblatt, F. 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review* 65(6):386.

Roughgarden, T., and Schrijvers, O. 2017. Online prediction with selfish experts. In *Advances in Neural Information Processing Systems*, 1300–1310.

Sáez, J. A.; Galar, M.; Luengo, J.; and Herrera, F. 2014. 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.

Shalev-Shwartz, S., et al. 2012. Online learning and online convex optimization. *Foundations and Trends in Machine Learning* 4(2):107–194.

Shawe-Taylor, J., and Cristianini, N. 2002. On the generalization of soft margin algorithms. *IEEE Transactions on Information Theory* 48(10):2721–2735.

Tsang, I. W.; Kocsor, A.; and Kwok, J. T. 2007. Simpler core vector machines with enclosing balls. In *Proceedings of the 24th International Conference on Machine Learning*, 911–918. ACM.

Wang, J.; Zhao, P.; and Hoi, S. C. 2016. Soft confidence-weighted learning. *ACM Transactions on Intelligent Systems and Technology (TIST)* 8(1):15.

Wu, Y., and Liu, Y. 2007. Robust truncated hinge loss support vector machines. *Journal of the American Statistical Association* 102(479):974–983.

Wu, X.; Yu, K.; Ding, W.; Wang, H.; and Zhu, X. 2013. Online feature selection with streaming features. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35(5):1178–1192.

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

Ying, Y., and Pontil, M. 2008. Online gradient descent learning algorithms. *Foundations of Computational Mathematics* 8(5):561–596.

Yu, H.; Neely, M.; and Wei, X. 2017. Online convex optimization with stochastic constraints. In *Advances in Neural Information Processing Systems*, 1428–1438.

Zhang, Q.; Zhang, P.; Long, G.; Ding, W.; Zhang, C.; and Wu, X. 2015. Towards mining trapezoidal data streams. In *Data Mining (ICDM), 2015 IEEE International Conference on*, 1111–1116. IEEE.

Zhang, Q.; Zhang, P.; Long, G.; Ding, W.; Zhang, C.; and Wu, X. 2016. Online learning from trapezoidal data streams. *IEEE Transactions on Knowledge and Data Engineering* 28(10):2709–2723.