One-pass Online Learning: A Local Approach

Zhaoze Zhou[‡♯], Wei-Shi Zheng[‡†], Jian-Fang Hu[⋆], Yong Xu[⋄], Jane You[⊕]

[‡] School of Information Science and Technology,
Sun Yat-Sen University, Guangzhou, China.
[♯]Collaborative Innovation Center of High Performance Computing,
National University of Defense Technology, Changsha 410073, China
[⋆] School of Mathematics and Computational Science,
Sun Yat-Sen University, Guangzhou, China.
[⋄] Shenzhen Graduate School, Harbin Institute of Technology, Shenzhen, China
[⊕] Department of Computing, The Hong Kong Polytechnic University, Hong Kong
[†]Guangdong Provincial Key Laboratory of Computational Science, China
zhouzhaoze@gmail.com, wszheng@ieee.org, hujianf@mail2.sysu.edu.cn,
yongxu@ymail.com, csyjia@comp.polyu.edu.hk

Abstract

Online learning is very important for processing sequential data and helps alleviate the computation burden on large scale data as well. Especially, one-pass online learning is to predict a new coming sample's label and update the model based on the prediction, where each coming sample is used only once and never stored. So far, existing one-pass online learning methods are globally modeled and do not take the local structure of the data distribution into consideration, which is a significant factor of handling the nonlinear data separation case. In this work, we propose a local online learning (LOL) method, a multiple hyperplane passive aggressive algorithm integrated with online clustering, so that all local hyperplanes are learned jointly and working cooperatively. This is achieved by formulating a common component as information traffic among multiple hyperplanes in LOL. A joint optimization algorithm is proposed and theoretical analysis on the cumulative error is also provided. Extensive experiments on 11 datasets show that LOL can learn a nonlinear decision boundary, overall achieving notably better performance without using any kernel modeling and second order modeling.

Keywords: One-pass Online learning, Local modeling, Classification

1. Introduction

We are concerning the one-pass online learning without keeping any tracks of passed samples. The key idea is to update current model by retaining the new learned model close to the current one and meanwhile imposing a margin separation on the most recent sample. After prediction and updating current model, the sample is abandoned, which means it could not be used for training again. Therefore, one-pass training reduces the consuming of memory so greatly that it is very practical in some circumstances. For example, a closed-circuit television camera with very limited resources is allowed to learn from video stream in one-pass manner to enhance its performance of recognition. Hence, one-pass online learning reduces the burden of the learning system and makes machine learning models more applicable and flexible.

One of the most widely known one-pass online learning methods is the first-order Passive-Aggressive (PA) method [1], which updates a marginal classifier according to the feedback of the prediction of each sequential data point. In order to explicitly consider the uncertainty of weights of linear classifier, confidence-weighted(CW) learning [2] as well as its variants soft confidence-weighted (SCW-I, SCW-II)[3], adaptive regularization of weight vectors (AROW)[4] have been recently investigated. However, the passive-aggressive and its related confidence-weighted learning methods still assume that samples are almost linearly separable, which is not always true since data points are always nonlinearly separable in the original input space.

To address the nonlinear separation problem in online learning, there are indeed some nonlinear algorithms, but not all of them are one-pass based. These algorithms include direct application of kernel trick on online linear classifiers [5, 6], budget-based online models [7, 8, 9, 10], and kernel approximation mapping based models [11] by using random Fourier features or Nyström method.

However, limitations exist in these methods, including (1) memory overflow after processing a large amount of data due to keeping historical wrong classified samples as support vectors (SVs) in [5, 6], (2) large computational burden caused by processing on SVs in the budget in [7, 8, 9, 10], and (3) data-independence and not adaptation to data stream in [11].

In offline learning, besides kernelizing linear classification models to solve the linearly non-separable problems, local classifiers have also been investigated recently to assign data samples to a set of prototypes and then infer the weights for model combination of local classifiers. Locally linear support vector machine(LLSVM)[12], and local deep kernel learning(LDKL)[13] were proposed to solve non-linear classification tasks using a weighted combination of multiple local hyperplanes, whose weights can be determined by local coding. As an extreme, the 1-nearest prototype classifier(1-NN)[14] combines prototype learning and nearest neighbor search to perform classification. Local classifiers has better adaptability to various types of data distribution than kernel classifiers, and the advantage of combining local classifiers is to avoid kernel modeling, so as to avoid computational expensive for large scale data. However, existing works mentioned above are not specifically designed for online learning tasks. Hence how to derive online local learning and how local online approach works for on-the-fly classification are still unknown.

In this paper, we propose a novel online approach by jointly learning multiple local hyperplanes to nonlinearly process sequential data in an one-pass manner. In particular, we extend the single hyperplane passive-aggressive method to a multiple local hyperplanes one. All local hyperplanes will be connected by a common component and optimized simultaneously. In our modeling, the local specific components of hyperplanes allow our model to make more accurate prediction locally (i.e. sensitive to a probe data point), while the common component shared by these local hyperplanes alleviates the over-fitting caused by the local information. A novel optimization algorithm is proposed and the theoretical relationship between the single and multiple hyperplane passive aggressive algorithms is derived from the aspect of cumulative error. We call our proposed model the *local online learning* (LOL) method. Our method achieves notable better performance on various tasks, especially on multi-class classification tasks, such as Multi-PIE Identity recognition of 249 subjects, achieving more than 95% accuracy and at least 13% higher than the compared methods. The details will be discussed in the experiment section (Sec. 5).

It is indeed that there are related online developments in pattern recognition such as online tracking [15], online face recognition [16, 17], incremental object matching [18], and online action recognition [19]. There are also works on using sequential algorithm to learn a classifier when only limited source is available for computation such as [20]. However, these models are not generally designed to make themselves be applicable for other applications, or they only conduct the learning when the size of dataset is increasing but are not one-pass based suitable for locality sensitive classification. Compared to these work, we aim to learn an online classifier for general purpose and make it suitable for one-pass online learning.

The remainder of this paper is organized as follows: Sec. 2 briefly reviews the related one-pass online learning algorithms. Sec. 3 and Sec. 4 detail the proposed local online learning algorithms and the corresponding theoretical analysis. Experimental results are presented in Sec. 5. Finally we draw the conclusion in Sec. 6.

2. Related Work

2.1. Passive Aggressive Algorithm (PA)

Like Perceptron[21], the first order Passive-Aggressive algorithm (PA)[1] focuses on learning linear classification model for each new sample $\mathbf{x} \in \mathbb{R}^d$, formed as:

$$g(\mathbf{x}) = sign(\mathbf{w}^T\mathbf{x}), \tag{1}$$

where *sign* function outputs the prediction label (-1 or +1) of the input and $\mathbf{w}$ is a weight vector. Passive-Aggressive method aims to use the pre-learned globally linear hyperplane to guide the prediction of new observed sample $\mathbf{x}_t$ labeled with $y_t \in \{-1, 1\}$ at time step t by solving the following problem:

$$\begin{aligned} \mathbf{w}_{t+1} &= \arg\min_{\mathbf{w}} \frac{1}{2} \left\| \mathbf{w} - \mathbf{w}_t \right\|^2 \\ &s.t.\ \ell^{pa}(\mathbf{w}; (\mathbf{x}_t, y_t)) = 0 \end{aligned} \tag{2}$$

where $\mathbf{w}_t$ is the model/hyperplane at time step t before update. Here, ℓ^{pa} is the hinge loss function, i.e. $\ell^{pa}(\mathbf{w}; (\mathbf{x}_t, y_t)) = max\{0, 1 - y_t \cdot \mathbf{w}^T \mathbf{x}_t\}$. In practice, a non-negative slack variable ξ is introduced so that the above optimization problem becomes:

$$\begin{aligned} \mathbf{w}_{t+1} &= \arg\min_{\mathbf{w}} \frac{1}{2} \left\| \mathbf{w} - \mathbf{w}_t \right\|^2 + C\xi, \\ &s.t.\ \ell^{pa}(\mathbf{w}; (\mathbf{x}_t, y_t)) \leq \xi \quad \text{and} \quad \xi \geq 0 \end{aligned} \tag{3}$$

where C is a positive parameter. This optimization problem can be understood as searching a new optimal hyperplane that does not differ from the current one too much in order to meet the loss constraint with respect to a new input sample. The solution of the problem (3) is given by

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \eta_t y_t \mathbf{x}_t, \quad \eta_t = \min\{C, \frac{\ell^{pa}}{\left\| \mathbf{x}_t \right\|^2}\} \tag{4}$$

where η_t is the learning rate. Obviously, Passive-Aggressive method learns a globally linear decision function without considering the local distribution of data.

2.2. Confidence Weighted Online Learning Algorithm Family

Confidence weighted algorithm (CW)[2], soft confidence weighted algorithms (SCW-I, SCW-II) [3] and adaptive regularization of weight vectors (AROW)[4] were proposed to explore the underlying structure of features. Confidence-weighted learning is actually inspired by Passive-Aggressive learning but holds a Gaussian distribution assumption over the weights. Confidence weighted algorithm assumes that a Gaussian distribution with mean $\mu \in \mathbb{R}^d$ and covariance matrix $\mathbf{\Sigma} \in \mathbb{R}^{d \times d}$ is imposed on linear weights . The distribution is obtained by minimizing the Kullback-Leiber divergence between the new distribution (parameterized by $\mathcal{N}(\mu_{t+1}, \mathbf{\Sigma}_{t+1})$) and the old one (parameterized by $\mathcal{N}(\mu_t, \mathbf{\Sigma}_t)$) in a passive-aggressive way, forcing the probability of accurate prediction of current sample $\mathbf{x}_t$ greater than a threshold η below:

$$(\mu_{t+1}, \mathbf{\Sigma}_{t+1}) = \underset{(\mu,\mathbf{\Sigma})}{\arg\min} D_{KL}(\mathcal{N}(\mu, \mathbf{\Sigma}), \mathcal{N}(\mu_t, \mathbf{\Sigma}_t)) \tag{5}$$
$$s.t. Pr_{\mathbf{w}\sim\mathcal{N}(\mu,\mathbf{\Sigma})}[y_t \cdot (\mathbf{w}^T \mathbf{x}_t) \geq 0] \geq \eta$$

The problem (5) has a solution of the following form:

$$\mu_{t+1} = \mu_t + \alpha_t y_t \mathbf{\Sigma}_t \mathbf{x}_t, \quad \mathbf{\Sigma}_{t+1} = \mathbf{\Sigma}_t - \beta_t \mathbf{\Sigma}_t \mathbf{x}_t^T \mathbf{x}_t \mathbf{\Sigma}_t \tag{6}$$

Soft confidence-weighted (SCW-I, SCW-II)[3] and adaptive regularization of weight vectors (AROW)[4] are extensions of confidence weighted algorithm [2], sharing the same update form but with different rules to learn the coefficients. These methods model the globally linear weight with a distribution and thus introduce variations of linear hyperplanes into the modeling. However, this modeling is still globally linear.

2.3. Online Kernel (Approximation) Algorithms

Inspired by the successful application of kernel tricks [22] to enhance the linear classifiers, some researchers intend to employ kernel tricks for online linear classifier learning. Correspondingly, a lot of kernel approximation methods have been developed to reduce the computational complexity caused by the employment of kernel tricks [23] [24]. However, these method need to keep all historical wrong classified samples as support vectors (SVs). Alternatively, online learning with a budget[7][8][9] is also proposed by maintaining a limit on the number of SVs, based on some strategies for balancing sacrifice of accuracy and computational burden. However, keeping SVs albeit limited in a budget makes these methods not strictly one-pass because these methods would revisit the passed samples to adjust their model. For example, budgeted online learning algorithm with a certain removal strategy needs to check which SVs to remove and therefore the cost of each update is expensive.

While kernelization provides opportunity for solving nonlinear classification problem, it introduces heavy computational burden with the growth of numbers of support vectors (SVs). To solve the problem, budgeted online learning algorithms such as Budgeted Stochastic Gradient Descent(BSGD)[10] which limits the number of SVs were proposed to bound the number of SVs. BSGD describes a budgeted version of kernelized SGD for SVM by controlling the number of SVs through one of several budget maintenance strategies, such as removal of SVs.

An alternative way to solve the problem is two recently proposed methods, namely Fourier Online Gradient Descent(FOGD) and Nyström Gradient Descent(NGOD) [11], which integrate kernel approximation techniques into online learning. These methods adopt kernel feature mappings on input space, where the corresponding inner product space is induced by the desired mercer kernel, such as Gaussian kernel. After the mapping, they learn a linear prediction model in the feature space afterwards. The key difference between FOGD and NOGD is that FOGD adopts Random Fourier feature sampling technique, while NOGD adopts Nyström method to set up the kernel feature mapping. More specifically, for processing the sample $\mathbf{x}_t$ at time step t, FOGD performs online learning in the following space:

$$\mathbf{z}_t = [cos(\mathbf{u}_1^T \mathbf{x}_t), sin(\mathbf{u}_1^T \mathbf{x}_t), ..., cos(\mathbf{u}_D^T \mathbf{x}_t), sin(\mathbf{u}_D^T \mathbf{x}_t)], \tag{7}$$

where $\mathbf{u}_i$ is sampled from a Gaussian distribution $N(\mathbf{0}, \sigma^{-2}\mathbf{I})$. NOGD learns in another space below:

$$\mathbf{z}_t = [\mathbf{K}(\mathbf{x}_1, \mathbf{x}_t), \mathbf{K}(\mathbf{x}_2, \mathbf{x}_t), ..., \mathbf{K}(\mathbf{x}_B, \mathbf{x}_t)]\mathbf{V}_k \mathbf{D}_k^{-1/2}, \tag{8}$$

where the Nyström method samples k columns from kernel matrix $\mathbf{K}$ consisting of the first B samples, i.e. $\mathbf{x}_1, \cdots, \mathbf{x}_B$, $(B \ll T)$, and $\mathbf{V}_k$, $\mathbf{D}_k$ are derived from the singular value Decomposition(SVD) of $\mathbf{K}_k = \mathbf{V}_k \mathbf{D}_k \mathbf{V}_k^T$ where $\mathbf{K}_k$ is the best rank-k approximation[11] of $\mathbf{K}$. Fourier feature sampling is obviously data independent, while Nyström sampling is only applied to the first portion (small if the data stream is large) of the data stream. Therefore, there is no guarantee that these methods could adapt to the change of data distribution, since the kernel approximation methods they used are only based on the early data distribution in a data stream and do not take any future data distribution change into consideration as more and more data points are accessed.

3. Local Online Learning

3.1. The Model

The proposed local online learning (LOL) aims to learn multiple hyperplanes locally on one-pass data points. LOL consists of two parts: 1) developing a multiple hyperplane passive-aggressive algorithm to learn multiple hyperplanes locally and jointly, making the online learning locally sensitive to data distribution; 2) incorporating online clustering in order to assign each probe sample to its specific prototype domain, which is the region of the prototype closest to the probe sample. The prototype domain is the description of data distribution for our online learning method. We detail our model below.

Inspired by recent local prototype methods [25, 12, 14] and the joint local multiple hashing modeling [26] which are offline methods, we develop a new online learning approach based on the following k local prototype model:

$$f(\mathbf{x}) = \sum_{i=1}^{k} f_i(\mathbf{x}) \cdot \mathbf{1}(\mathbf{x} \in \mathcal{D}(\mathcal{P}_i)) \tag{9}$$

where k is the number of prototypes, $\mathcal{D}(\mathcal{P}_i)$ denotes domain of the i^{th} prototype $\mathcal{P}_i$, $\mathbf{1}(\cdot)$ is an indicator function whose value is 1 if $\mathbf{x} \in \mathcal{D}(\mathcal{P}_i)$ or 0 otherwise, and $f_i(\mathbf{x})$ is defined as:

$$f_i(\mathbf{x}) = \mathbf{w}_i^T \mathbf{x} \tag{10}$$

Although data is not always globally linearly separable, it is still possible that they are locally linearly separable. Eq.(9) assumes that the combined linear prediction model achieves locally sensitive classification depending on where the test data point locates.

Now, we start to develop local online learning (LOL) based on Eq.(9). Assume that each local component $\mathbf{w}_i$ can be factorized into two components: a common component $\mathbf{w}$ shared by all local components and prototype specific component $\mathbf{u}_i$, i.e. $\mathbf{w}_i = \mathbf{w} + \mathbf{u}_i$, $i = 1, \cdots, k$. The key idea of our model is to optimize the common component $\mathbf{w}$ and the local components $\mathbf{u}_i$ separately and connect them by balancing the extent of their update using a balance parameter λ. For online learning, both common component and local components are updated by passively neglecting correct classified samples but aggressively adjusting all components to fit the wrong classified samples. Updating the common component $\mathbf{w}$ bridges all local hyperplanes, while updating each local components if necessary helps refine prototype specific hyperplane (i.e. the local hyperplane). To this end, for a new input data $\mathbf{x}_t$ labeled with $y_t \in \{-1, 1\}$ at time step t, we propose to optimize the following objective function for our LOL problem:

$$\begin{aligned} & \mathbf{w}_{t+1}, \mathbf{u}_{1,t+1}, \ldots, \mathbf{u}_{k,t+1} = \\ & \underset{\mathbf{w},\mathbf{u}_1,\ldots,\mathbf{u}_k}{\arg\min} \frac{\lambda}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 + \frac{1}{2}\sum_{i=1}^{k}\|\mathbf{u}_i - \mathbf{u}_{i,t}\|^2 \\ & s.t. \quad \ell^{lol}(\mathbf{w}, \mathbf{u}_1, \ldots, \mathbf{u}_k; (\mathbf{x}_t, y_t)) = 0 \end{aligned} \tag{11}$$

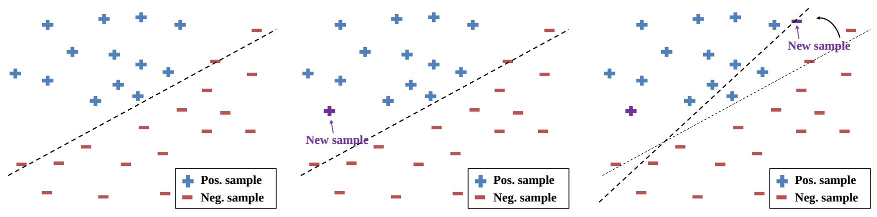
where $\mathbf{w}_t$ and $\mathbf{u}_{i,t}$ are the common component and the i^{th} local component at time step t before update, respectively, and the loss function is:

$$\begin{aligned} & \ell^{lol}(\mathbf{w}, \mathbf{u}_1, \ldots, \mathbf{u}_k; (\mathbf{x}_t, y_t)) \\ = & \begin{cases} 0 & y_t \cdot f(\mathbf{x}_t) \geq 1 \\ 1 - y_t \cdot f(\mathbf{x}_t) & \text{otherwise} \end{cases} \end{aligned} \tag{12}$$

To make Eq. (11) more practical, a slack variable is introduced into the above optimization problem to achieve soft-margin maximization:

$$\begin{aligned} & \mathbf{w}_{t+1}, \mathbf{u}_{1,t+1}, \ldots, \mathbf{u}_{k,t+1} = \\ & \underset{\mathbf{w},\mathbf{u}_1,\ldots,\mathbf{u}_k}{\arg\min} \frac{\lambda}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 + \frac{1}{2}\sum_{i=1}^{k}\|\mathbf{u}_i - \mathbf{u}_{i,t}\|^2 + C\xi \\ & s.t. \quad \ell^{lol}(\mathbf{w}, \mathbf{u}_1, \ldots, \mathbf{u}_k; (\mathbf{x}_t, y_t)) \leq \xi, \ \xi \geq 0 \end{aligned} \tag{13}$$

Since $\mathbf{w}$ is the common component of our target local hyperplanes that require the hyperplane of each local prototype to align with, $\mathbf{w}$ builds the information traffic and controls the shared information among different prototypes.



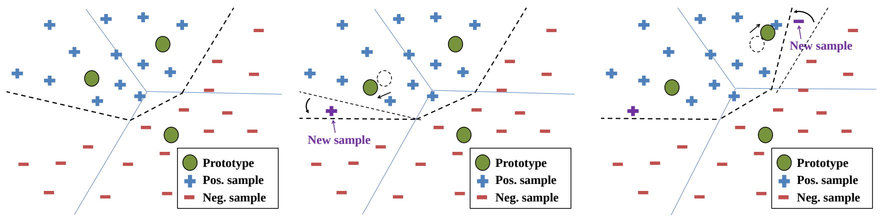
(a) (b) (c)

Figure 1: Example of how PA processes new input samples: (b) is the decision boundary (hyperplane) updated from (a), and (c) is the hyperplane update from (b).

In this way, the learned local hyperplanes are not separated in each prototype. It is therefore able to avoid over-fitting in each local prototype.

Figure 1 shows that Passive-Aggressive method (PA) updates the model when a wrong prediction is made. In comparison Figure 2 shows that in local online learning (LOL) a new sample is assigned to a prototype domain first and then the local model is updated if the local model makes a wrong prediction. Figure 3 shows an example of decision boundary learned by PA and LOL, which shows LOL is able to learn nonlinear boundary locally.

Independent local online learning (I-LOL). When we eliminate the common component $\mathbf{w}$ from the Criterion (13), i.e. setting $\mathbf{w} = \mathbf{0}$, then the proposed local online learning (LOL) becomes a direct extension of Passive-Aggressive method learned on different prototype domains adaptively. Here, we denote such a simplified version as independent local online learning (I-LOL). However, we show in our experiments that such a simplified local online learning is not always optimal. As shown in our experiments, on 8 datasets among all, lower test errors could be obtained if all these local hyperplanes are learned jointly rather than independently.



(a) (b) (c)

Figure 2: Example of how local online learning (LOL) processes new input samples: (b) is the decision boundary (hyperplane) updated from (a), and (c) is the hyperplane update from (b)

3.2. Optimization

Updating Hyperplanes for local online learning (LOL). Although LOL has to update quite a lot of local components and a common component, we show in the following that after certain transformation, LOL can be easily solved by the Passive-Aggressive algorithm (PA) itself, although LOL is a local model and PA is a global model.

Let

$$\tilde{\mathbf{x}}_t = \left[\frac{\mathbf{x}_t}{\sqrt{\lambda}}^T, \mathbf{0}^T, \mathbf{0}^T, ..., \mathbf{x}_t{}^T, ..., \mathbf{0}^T \right]^T \tag{14}$$

where the $(i+1)^{th}$ component of $\tilde{\mathbf{x}}_t$ is $\mathbf{x}_t$. Accordingly, the global and local components could be represented as follow:

$$\tilde{\mathbf{w}} = \left[\sqrt{\lambda}\mathbf{w}^T, \mathbf{u}_1^T, \mathbf{u}_2^T, ..., \mathbf{u}_i^T, ..., \mathbf{u}_k^T \right]^T \tag{15}$$

By using Eq.(14)and Eq. (15), the objective function(13) could be rewritten as:

$$\begin{aligned} \tilde{\mathbf{w}}_{t+1} &= \arg\min_{\tilde{\mathbf{w}}} \left(\frac{1}{2} \left\| \tilde{\mathbf{w}} - \tilde{\mathbf{w}}_t \right\|^2 + C\xi \right) \\ s.t. \quad & \ell^{lol}(\tilde{\mathbf{w}}; (\tilde{\mathbf{x}}_t, y_t)) \leq \xi, \;\; \xi \geq 0. \end{aligned} \tag{16}$$

In this way, we convert local online learning (LOL) into the same mathematical form as Eq.(3). However, we shall emphasize that LOL is intrinsically different from Passive-Aggressive method (PA), LOL learns locally sensitive online

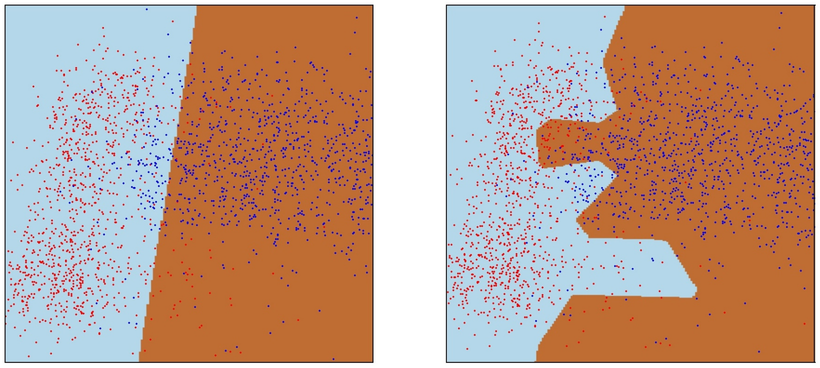


Figure 3: Decision boundaries learned on SVMGUIDE1 dataset after PCA Projection into a two-dimensional space: a globally linear one learned by Passive-Aggressive method (left) and a piece-linear one by our local online learning (right).

classifier but PA method is not. Then, as in [1], we first define the Lagrangian:

$$\mathcal{L}(\tilde{\mathbf{w}}, \xi, \eta, \gamma) = \arg\min_{\tilde{\mathbf{w}}} \frac{1}{2} \|\tilde{\mathbf{w}} - \tilde{\mathbf{w}}_t\|^2 + C\xi - \eta(\ \ell^{lol}(\tilde{\mathbf{w}}; (\tilde{\mathbf{x}}_t, y_t)) - \xi) - \gamma \cdot \xi \tag{17}$$

where η and γ are Lagrange multipliers. The optimal solution is obtained when the gradients equal to zero. By solving Eq.(17), we obtain

$$\tilde{\mathbf{w}}_{t+1} = \tilde{\mathbf{w}}_t + \eta_t \tilde{\mathbf{x}}_t \tag{18}$$

$$\eta_t = \min\{C, \frac{\ell^{lol}}{\|\tilde{\mathbf{x}}_t\|^2}\} \tag{19}$$

The optimal update for the new $\tilde{\mathbf{w}}$ has the form of a gradient descent step with a step size η_t.

Updating Prototypes for local online learning (LOL). Different from Passive-Aggressive method (PA), local online learning (LOL) is relying on quantizing input data to different prototype domains. Since LOL is developed for online learning, not all samples are provided in advance and thus it is necessary to update the prototype domain online. In order to process data points $\mathbf{x}_t$ for prototyping in a sequential order: $t = 1, 2, \ldots$, we adopt a sequential version of K-means[27] which uses the first k data points as initial prototypes, then assigns a new data point $\mathbf{x}_t, t>k$ to the closest prototype learned from step $t-1$, and finally updates the corresponding prototype after the assignment. The rule of online update of prototypes could be written as:

$$\mathcal{P}_i = \mathcal{P}_i + \frac{1}{n_i}(\mathbf{x}_t - \mathcal{P}_i) \tag{20}$$

where $\mathcal{P}_i$ is the i^{th} prototype and n_i is the total number of previous samples assigned to the i^{th} prototype. After we predict the nearest prototype of $\mathbf{x}_t$, the i^{th} one for example, then the original prototype $\mathcal{P}_i$ is updated. It could be viewed that the prototype $\mathcal{P}_i$ moves towards the sample $\mathbf{x}_t$ at time t. It is worth mentioning that due to the sequential manner of clustering we use, it might be probably that noisy clustering could affect the performance of our online learning. However, our experiments in a latter section will show that the gap caused by sequential K-means and the offline K-means (if available) is small enough to be tolerated after sufficient training.

4. Theoretical Analysis

We analyze the relative error rate of local online learning (LOL) as compared to one of Passive-Aggressive method [1]. Let

$$\tilde{\mathbf{w}}^{pa} = \left[\sqrt{\lambda}\mathbf{w}^T, \mathbf{0}^T, \mathbf{0}^T, ..., \mathbf{0}^T, ..., \mathbf{0}^T\right] \tag{21}$$

Based on the definition of $\tilde{\mathbf{x}}_t$ in Eq. (14), we have $\tilde{\mathbf{w}}^{paT}\tilde{\mathbf{x}}_t = \mathbf{w}^T\mathbf{x}_t$ and thus $\ell^{pa}(\mathbf{w}; (\mathbf{x}_t, y_t)) = \ell^{lol}(\tilde{\mathbf{w}}^{pa}; (\tilde{\mathbf{x}}_t, y_t))$, where ℓ^{pa} is the hinge loss function defined in Eq. (2) and ℓ^{lol} is hinge loss function defined in Eq. (15). Hence, local online learning (LOL) (Eq.(16)) is actually equivalent to linear passive-aggressive method if we constrain the projection in Eq.(11) in the set $\mathcal{Q} = \left\{[\sqrt{\lambda}\mathbf{w}^T, \mathbf{0}^T, \mathbf{0}^T, ..., \mathbf{0}^T, ..., \mathbf{0}^T] \mid \mathbf{w} \in \mathbb{R}^d\right\}$ as follows:

$$\begin{aligned} \tilde{\mathbf{w}}^{pa}_{t+1} &= \arg\min_{\tilde{\mathbf{w}}^{pa} \in \mathcal{Q}} \left(\frac{1}{2}\|\tilde{\mathbf{w}}^{pa} - \tilde{\mathbf{w}}^{pa}_t\|^2\right) \\ s.t. \quad & \ell^{lol}(\tilde{\mathbf{w}}^{pa}; (\tilde{\mathbf{x}}_t, y_t)) = 0. \end{aligned} \tag{22}$$

```
Algorithm
Input: aggressiveness parameter: $C>0$
    the number of prototypes: $k$
1: initialize $\tilde{\mathbf{w}}_0$: $\tilde{\mathbf{w}}_0 = \mathbf{0}$
2: for $t = 1, 2, \ldots$ do
3:     receive instance: $\mathbf{x}_t \in \mathbb{R}^d$
4:     if $t \leq k$ then
5:         initialize prototype: $\mathcal{P}_t \leftarrow \mathbf{x}_t, n_t = 1$ /* initialize $k$ prototypes*/
6:     else
7:         get the nearest prototype: $i \leftarrow \arg\min_j (Dist(\mathcal{P}_j, \mathbf{x}_t))$
8:         update prototype: $\mathcal{P}_i \leftarrow \mathcal{P}_i + \frac{1}{n_i}(\mathbf{x}_t - \mathcal{P}_i), n_i \leftarrow n_i + 1$
9:     end if
10:    $\tilde{\mathbf{x}}_t = [\mathbf{x}^T, \mathbf{0}^T, \ldots, \mathbf{x}^T((i+1)^{th} \text{ position}), \ldots, \mathbf{0}^T]^T, \tilde{\mathbf{x}}_t \in \mathbb{R}^{kd}$
11:    predict: $\hat{y}_t = sign(\tilde{\mathbf{w}}_t^T \tilde{\mathbf{x}}_t)$
12:    receive correct label: $y_t \in \{-1, 1\}$
13:    compute loss: $\ell_t^{lol} = \max\{0, 1 - y_t(\tilde{\mathbf{w}}_t^T \tilde{\mathbf{x}}_t)\}$
14:    set: $\eta_t = \min\{C, \frac{\ell_t^{lol}}{\|\tilde{\mathbf{x}}_t\|^2}\}$
15:    update: $\tilde{\mathbf{w}}_{t+1} = \tilde{\mathbf{w}}_t + \eta_t y_t \tilde{\mathbf{x}}_t$
16: end for
Output: model: $\tilde{\mathbf{w}} \in \mathbb{R}^{kd}$
```

Hence, the best solution $\tilde{\mathbf{w}}_t^{pa}$ of Passive-Aggressive method (PA) may be actually a suboptimal solution of local online learning (LOL), i.e.

$$\ell^{lol}(\tilde{\mathbf{w}}_t^{lol}; (\tilde{\mathbf{x}}_t, y_t)) \leq \ell^{lol}(\tilde{\mathbf{w}}_t^{pa}; (\tilde{\mathbf{x}}_t, y_t)) = \ell^{pa}(\mathbf{w}_t^{pa}; (\mathbf{x}_t, y_t)). \tag{23}$$

This shows LOL would not obtain a higher error rate than PA. Beyond the above, we can further have the following relationship between the cumulative errors of LOL and PA with respect to Eq. (2) and Eq. (11).

Theorem 1. *Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ be a series of samples where $\mathbf{x}_t \in \mathbb{R}^d$, $y_t \in \{+1, -1\}$ and $\|\mathbf{x}_t\| \leq R$ for all t. Assume that there exists a vector $\tilde{\mathbf{u}}^{lol}$ such that $\ell^{lol\star}(\tilde{\mathbf{u}}^{lol}; (\tilde{\mathbf{x}}_t, y_t)) = 0$ for all $t = 1, 2, \cdots, T$. Then the relation between cumulative errors of local online learning (LOL) and that of Passive-Aggressive* *method (PA) can be further described below:*

$$\sum_{t=1}^{T} (\ell_t^{lol})^2 \le \frac{1}{2}(1+\frac{1}{\lambda})R^2 \left(\| \tilde{\mathbf{u}}^{lol} \|^2 + \sum_{t=1}^{T} \frac{(\ell_t^{pa})^2}{\|\tilde{\mathbf{x}}_t\|^2} \right), \tag{24}$$

where $\ell_t^{lol} = \ell^{lol}(\tilde{\mathbf{w}}_t^{lol}; (\tilde{\mathbf{x}}_t, y_t))$, $\ell_t^{pa} = \ell^{lol}(\tilde{\mathbf{w}}_t^{pa}; (\tilde{\mathbf{x}}_t, y_t))$, *and* $\tilde{\mathbf{x}}_t$ *is a mapping of* $\mathbf{x}_t$ *as defined in Eq.(14).*

Proof. By using the Lemma 1 in [1], we first have

$$\sum_{t=1}^{T} \eta_t^{lol}(2\ell_t^{lol} - \eta_t^{lol} \| \tilde{\mathbf{x}}_t \|^2) \le \| \tilde{\mathbf{u}}^{lol} \|^2,$$

where $\eta_t^{lol} = \frac{\ell^{lol}(\tilde{\mathbf{w}}_t^{lol};(\tilde{\mathbf{x}}_t,y_t))}{\|\tilde{\mathbf{x}}_t\|^2}$. By using Eq.(23), we have $\ell_t^{lol} \le \ell_t^{pa}$ and thus $\eta_t^{lol} \le \frac{\ell^{pa}(\tilde{\mathbf{w}}_t^{pa};(\tilde{\mathbf{x}}_t,y_t))}{\|\tilde{\mathbf{x}}_t\|^2} = \eta_t^{'}$. Then

$$\sum_{t=1}^{T} 2\eta_t^{lol}\ell_t^{lol} - \eta_t^{'} \cdot \eta_t^{'} \| \tilde{\mathbf{x}}_t \|^2 \le \| \tilde{\mathbf{u}}^{lol} \|^2, \tag{25}$$

That is

$$\sum_{t=1}^{T} 2\frac{(\ell_t^{lol})^2}{\|\tilde{\mathbf{x}}_t\|^2} - \frac{(\ell_t^{pa})^2}{\|\tilde{\mathbf{x}}_t\|^2} \le \| \tilde{\mathbf{u}}^{lol} \|^2 \tag{26}$$

Since $\|\tilde{\mathbf{x}}_t\|^2 = (1+\frac{1}{\lambda})\|\mathbf{x}_t\|^2 \le (1+\frac{1}{\lambda})R^2$, we have

$$\sum_{t=1}^{T} (\ell_t^{lol})^2 \le \frac{1}{2}(1+\frac{1}{\lambda})R^2 \left(\| \tilde{\mathbf{u}}^{lol} \|^2 + \sum_{t=1}^{T} \frac{(\ell_t^{pa})^2}{\|\tilde{\mathbf{x}}_t\|^2} \right). \tag{27}$$

□

180 ## 5. Experimental Results

5.1. Datasets

We evaluated our method on 11 benchmark datasets from different fields, including two digit datasets (MNIST[1] and USPS[2]), two face datasets (CBCL-Face [3] and the first section of CMU Multi-PIE [28]), the dataset of KDDCUP

[1]`http://yann.lecun.com/exdb/MNIST/`

[2]`http://www-i6.informatik.rwth-aachen.de/~keysers/USPS.html`

[3]`http://cbcl.mit.edu/software-datasets/FaceData2.html`

2008, and other datasets as SVMGUIDE1, IJCNN1 and LETTER from LIBSVM website[4]. It is worth noting that in the following, the Multi-PIE dataset will be used for two purposes. One is for pose classification (15 different poses) and we denote the Multi-PIE dataset as "Multi-PIE Pose" in this case. Another is for classifying identities (249 subjects) and we denote the Multi-PIE dataset as "Multi-PIE Identity" in such a case. More characteristics about these datasets are summarized in Table 2. Since online learning is designed for large scale problem, we also introduced two large scale classification tasks: 1) malicious urls detection and 2) web spam detection. We denote the dataset we used for malicious urls detection as "URL"[5]. We performed classification on such a dataset to detect malicious urls. The dataset "URL" has 2.4 million urls and 3.2 million features collected in a real-time system[29], a typical task which offline kernel machines could not afford. As for the dataset we used for web spam detection, we denote it as "WEBSPAM" [6]. WEBSPAM contains 350,000 records (web spam pages) collected by a fully automated web spam collection method[30] and we adopted the normalized unigram version available on the LIBSVM site.

Since the update form of the second order methods (see Eq.(6)) needs to keep a covariance matrix of the weight vector, which makes the training slow and even infeasible. For example, computation of the URL dataset with 3.2 million features yields a very large full covariance matrix consuming 381 GB. In order to compare these methods on "URL", we have to perform dimension reduction. In this paper, we performed Gaussian random projection [31] to reduce the data dimension for all methods to 100 dimension on URL.

About the training and testing on all dataset except KDDCUP2008, Multi-PIE Pose, Multi-PIE Identity and URL, we followed the existing training and testing division rules in literatures[7]. Since there is no default training and

[4]`http://wwww.csie.ntu.edu.tw/cjlin/libsvmtools/datasets`
[5]`http://sysnet.ucsd.edu/projects/url/`
[6]`http://www.cc.gatech.edu/projects/doi/WebbSpamCorpus.html`
[7]`http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`

testing protocols for KDDCUP2008[8], Multi-PIE-Pose, and Multi-PIE-Identity, we conducted as follows. For KDDCUP2008, we randomly selected 3/4 of the samples for training and the rest for testing. For Multi-PIE-Pose and Multi-PIE-Identity, we randomly divided them into half for training and another half for testing. For URL, we used the data samples collected in the first 100 days for training and the remaining 21 days' data samples for testing. For all the datasets except WEBSPAM and URL, experiments were run 10 times with different permutations of training set (i.e. sequences in different orders) and the averaged results are reported. As for URL data samples which were captured already in the order of *time*, due to the use of random projection, we performed *10* performed Gaussian random projection, computed the performance, and the average result was finally reported. The first 200,000 records of WEBSPAM were used for training and the remaining 150,000 records were for testing.

5.2. Measurement

Cumulative Error. It is defined by $\frac{\sum_1^t \mathbf{I}(y_t != \hat{y}_t)}{t}$. This measure is to tell how the online learning is trained on sequential training data. It reflects 1) the cumulative error rate caused by the online model for processing new input training data points before time step t and 2) how the accuracy of prediction changes over the data sequence.

Test Error. Different from the Cumulative error, traditional test error (on the unseen test set) is also tested on the final model after online training (on training set) process. It is necessary because the test error indicates the generalization ability of the learned online model on unseen data.

Cumulative error sometimes will be much higher than test error on the same dataset, because it is largely affected by initial wrong predictions.

[8]The original test dataset does not contain label information.

Table 1: Characteristics of the compared methods

methods	Second order	Using kernel	Non-linear	Local
PA[1]	No	No	No	No
Pegasos[32]	No	No	No	No
CW[2]	YES	No	No	No
SCW-I[3]	YES	No	No	No
SCW-II[3]	YES	No	No	No
AROW[4]	YES	No	No	No
BSGD[10]	No	Yes	Yes	No
FOGD[11]	No	Yes	Yes	No
NOGD[11]	No	Yes	Yes	No
I-LOL	No	No	Yes	Yes
LOL	No	No	Yes	Yes

5.3. Compared Methods

We compared the proposed method with the first order Passive Aggressive Algorithm(PA) and the second order confidence weighted family such as confidence weighted algorithm (CW)[2], soft confidence-weighted algorithms (SCW-I, SCW-II)[3], and adaptive regularization of weight vectors (AROW)[4]. Additionally, we also compared our method with Pegasos[32], a widely used linear SVM. Different from the original Pegasos algorithm using sub-sampling k samples in each iteration, we used it in an online way, i.e. sampling *one* sample in each iteration and that sample is then removed from the sample pool. To fairly evaluate the performance with several online learning algorithms, we adopted the implementation of LIBOL[33] for the compared methods and followed similar parameter setting for the methods in [3]. As for budgeted stochastic gradient descent method (BSGD), we used the BudgetedSVM toolbox[34]. To show that the proposed method local online learning (LOL) is benefited from locality sensitive modeling without using kernel, we also compare some kernel approximation online learning methods Fourier Online Gradient Descent(FOGD) and Nyström Gradient Descent(NGOD) [35] and an budgeted stochastic gradient descent method (BSGD)[10]. Finally, we compare the proposed local online

Table 2: Description of the datasets

Dataset	#Training	#Testing	#Features	#Classes
SVMGUIDE1	3,089	4,000	4	2
CBCL Face	6,977	24,045	361	2
IJCNN1	49,990	91,701	22	2
KDDCUP08	95,470	31,824	127	2
WEBSPAM	200,000	150,000	254	2
URL	1,976,130	420,000	100	2
USPS	7,291	2,007	256	10
LETTER	15,000	5,000	16	26
MNIST	60,000	10,000	780	10
Multi-PIE Pose	74,700	74,700	100	15
Multi-PIE Identity	74,700	74,700	100	249

learning (LOL) with Independent local online learning (I-LOL), the independent version without a common component to bridge local hyperplanes, to show the benefit of modeling the common component shared by all local hyperplanes. The characteristics of the compared methods are summarized in Table 1.

Parameter Setting: The parameter C in Passive-Aggressive method (PA), soft confidence-weighted (SCW-I, SCW-II) was determined by cross validation from $\{2^{-4}, 2^{-3}, \ldots, 2^3, 2^4\}$; the parameter η in CW, SCW-I, SCW-II was cross-validated from $\{0.5, 0.55, \ldots, 0.9, 0.95\}$. The balancing parameter λ in Pegasos was cross-validated in $\{10^{-4}, 10^{-3}, \ldots, 10^2, 10^3\}$. The size of budget B of budgeted stochastic gradient descent method (BSGD), Fourier Online Gradient Descent(FOGD), Nyström Gradient Descent(NGOD) was set to 200, and D in FOGD was set as $10 \times B$ and k in NOGD was set as B. The kernel bandwidth was set by cross validation over the range $\{1, 10, 100, 1000\}$ and the learning rate was set to 0.0001. In order to determine the optimal parameters for our online model on this dataset, we first performed a cross validation by repeatedly and randomly split the whole training set into two equal halves: one for training and the other for validation data. For each split, a set of online models with different parameter sets were learned on the training data and then tested (test

Table 3: Part I: Comparison of test errors (%) for different methods (Bold texts indicate the best results)

method	SVMGUDIE1	CBCL Face	IJCNN1	KDDCUP2008	WEBSPAM	URL
PA	30.35%±0.107	7.12%±0.059	34.98%±0.038	38.74%±0.047	7.56%	15.99% ± 0.052
Pegasos	37.49%±0.126	20.84%±0.182	25.48%±0.068	45.75%±0.030	11.58%	16.10%±0.023
CW	50.09%±0.243	3.64%±0.003	46.41%±0.028	34.52%±0.030	10.66%	36.46% ± 0.089
SCW-I	21.42%±0.010	3.20%±0.001	46.96%±0.079	25.95%±0.103	6.58%	9.01% ± 0.019
SCW-II	22.63%±0.002	3.08%±0.001	61.59%±0.314	44.89%±0.022	7.77%	9.31% ± 0.019
AROW	21.51%±0.002	3.22%±0.001	34.27%±0.005	43.98%±0.011	7.25%	9.28% ± 0.020
BSGD	5.73%±0.009	**2.14%±0.004**	4.36%±0.000	**0.57%±0.001**	5.27%	20.00%±0.023
FOGD	7.68%±0.013	6.47%±0.037	11.48%±0.070	0.70%±0.001	5.68%	9.72%±0.013
NOGD	5.75%±0.009	6.38%±0.017	11.99%±0.063	0.73%±0.003	14.82%	10.56% ±0.012
I-LOL	6.54%±0.021	5.34%±0.010	4.10%±0.006	0.65%±0.001	5.56%	8.70%±0.007
LOL	**5.26%±0.014**	3.68%±0.014	**3.15%±0.006**	0.69%±0.002	**4.95%**	**7.20%±0.007**

error) on the validation data. Then the parameters with the best average results over 10 random splits were selected to train our final online model on the whole training set. We report the test error of the learned model on the unseen test set in Table 3 and Table 4.

Since the local online learning (LOL) is not very sensitive to the parameters, we adopt a group of fixed parameters for all the tasks. The number of prototypes k is set to 60, the balancing parameter λ is set to 1.0 and the aggressive parameter C is set to 1.0. The influence of λ and k would be further evaluated in Section 5.4.4. In the Independent local online learning (I-LOL), parameters C and k are set as the same as LOL. In the multi-class classification problems, FOGD, NOGD , I-LOL and our LOL adopt one-vs-all strategy to predict the result.

To show our adopted sequential K-means holds a close performance to the "offline" K-means[9], we also performed experiment on 6 datasets with the pre-

[9]For comparison, we have to assume all the sequential data samples have been observed and the offline K-means can be run.

Table 4: Part II: Comparison of test errors (%) for different methods. (Bold texts indicate the best results)

method	USPS	LETTER	MNIST	Multi-PIE Pose	Multi-PIE Identity
PA	12.36%±0.015	34.17%±0.017	12.31%±0.011	4.22%±0.003	43.64%±0.003
Pegasos	20.87%±0.081	60.74%±0.049	21.73%±0.082	15.01%±0.025	83.35%±0.013
CW	10.78%±0.004	93.57%±0.038	12.56%±0.003	4.26%±0.001	31.17%±0.001
SCW-I	9.73%±0.003	37.33%±0.019	19.59%±0.008	4.37%±0.000	78.43%±0.004
SCW-II	9.29%±0.002	26.34%±0.003	12.04%±0.004	3.54%±0.000	29.04%±0.002
AROW	9.06%±0.004	33.03%±0.007	29.26%±0.090	6.63%±0.001	31.49%±0.002
BSGD	14.03%±0.008	29.28%±0.059	4.33%±0.002	8.99%±0.000	46.18%±0.102
FOGD	9.84%±0.024	19.24%±0.016	6.57%±0.002	3.58%±0.004	18.17%±0.007
NOGD	9.29%±0.015	23.16%±0.015	10.07%±0.005	10.78%±0.012	79.45%±0.014
I-LOL	10.19%±0.008	20.83%±0.012	4.82%±0.002	3.48%±0.002	31.95%±0.009
LOL	**7.88%±0.008**	**17.69%±0.011**	**3.03%±0.001**	**2.15%±0.001**	**4.62%±0.003**

trained prototypes via offline K-means, i.e. the prototypes are fixed before online training and no prototype update is made during the online learning process. The number of prototypes is set to the same value as the sequential one.

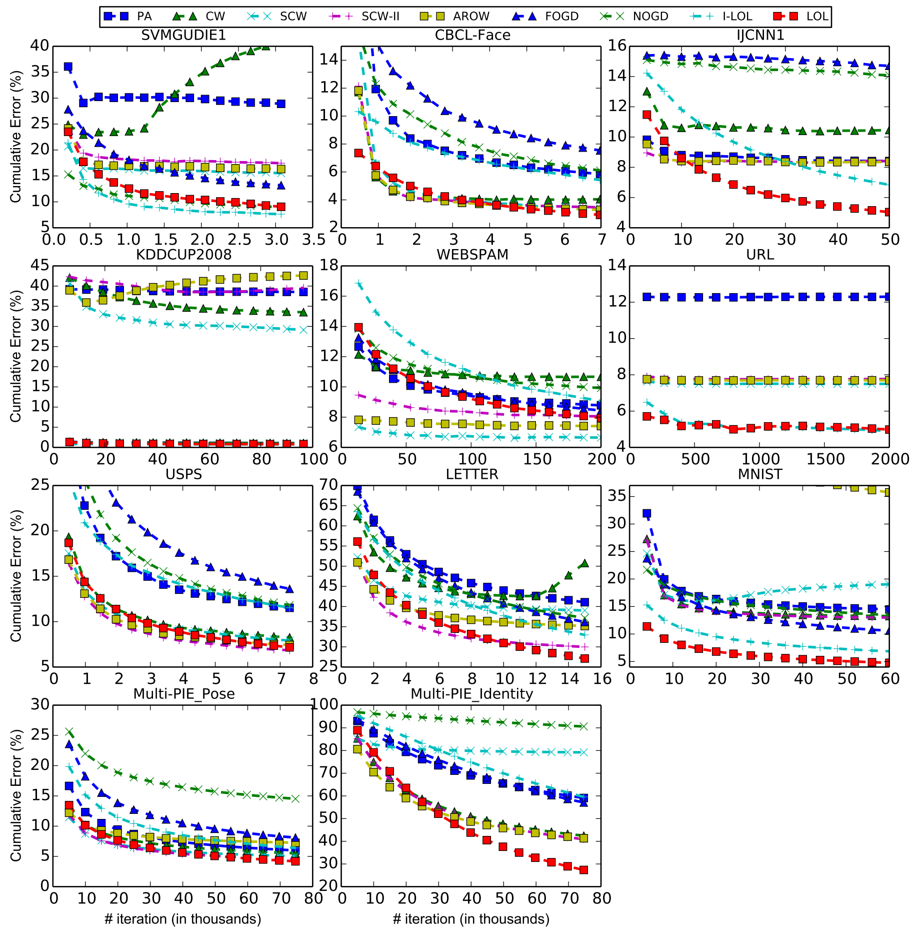


Figure 4: Cumulative error (%) on training data for different methods. For test error on probe, see Tables 3 and 4

5.4. Results

5.4.1. Comparison to Online Linear Classifiers

From Table 3, Table 4 and Figure 4, we show that local online learning (LOL) consistently outperforms Passive-Aggressive method (PA). Always, LOL algorithm has a much lower error rate than PA. Pegasos has a close performance to PA and thus is also clearly inferior to LOL. This shows that exploring local hyperplanes during online update is very important for addressing the nonlinearly separable data problem, while only preserving a global hyperplane is a notable weakness in PA. The results have provided empirical supports for our theoretical analysis in the previous section that LOL is able to achieve lower error rate.

We then compare local online learning (LOL) with the second-order algorithms, i.e. confidence weighted algorithm (CW), soft confidence-weighted (SCW-I, SCW-II) and adaptive regularization of weight vectors (AROW). They are all passive-aggressive like methods, but they are second-order, because they learn the weights of projection under Gaussian distribution. As shown in Table 3 and Figure 4, compared to LOL, LOL still performs better and obtains notable improvements on SVMGUIDE1, IJCNN1, KDDCUP2008 and WEBSPAM. Although all the linear models perform rather well on URL, LOL still has stable improvement as compared to the second order algorithms. On USPS and MNIST, which consist of 10 classes, the second order methods still work well. On the LETTER dataset that consists of 26 classes, the gap between LOL and these second order methods are large. In the Multi-PIE pose recognition task of 15 poses, their performances are still comparable to LOL, but for the identity classification of 249 people, they fail to keep high accuracy while the performance of LOL has only a slight increase of test error from 2.1% to 4.6%. Regarding the result of two real-world datasets: WEBSPAM and URL, LOL also outperforms the compared methods. Although the second order methods apply confidence weighting on features, they are still limited since they assume that data points are almost linearly separable. Hence, with the increase of the diversity of classes, the impact of local modeling is much more obvious.

5.4.2. Comparison to Online Kernel (Approximation) Classifiers

We also compare the recently proposed online kernel learning methods such as Fourier Online Gradient Descent (FOGD), Nyström Gradient Descent (N-GOD) and budgeted stochastic gradient descent method (BSGD)[10]. The results are presented in Table 3, Table 4 and Figure 4. For two-class classification task, the comparison results show that the proposed local online learning (LOL) obtains lower test error on all datasets except KDDCUP2008 , and notable improvements are gained on IJCNN1, e.g. LOL gaining 3.15% test error while FOGD and NOGD gain about 12%. Compared to BSGD, although BSGD achieves slightly better results on CBCL-Face and KDDCUP2008, LOL is only slightly inferior to BSGD especially on the KDDCUP2008 dataset. While FOGD, NOGD and BSGD have an improvement on WEBSPAM but close performance on URL comparing to the linear models, our proposed LOL holds steady and lower error on these two real-world datasets.

For the multi-class classification task, the gap is more clear. LOL significantly outperforms FOGD, NOGD and BSGD on all the multi-class datasets. On Multi-PIE, the proposed method achieves the lowest test error, obtaining a notable and clear margin. Especially, on Multi-PIE identity, there are 249 classes and only the proposed LOL achieves a test error about 5%, while the others are higher than 18%.

Although the proposed LOL is not a kernel based method, the gain of superiority of LOL is due to the fact that LOL can directly model the classification decision boundary locally and adapted to the change of data stream. Nevertheless, the comparison here shows that our local online learning modeling can be an robust alternative approach to generalize the online linear learning for handling nonlinearly separable sequential data.

5.4.3. Comparison to independent local online learning (Independent local online learning (I-LOL)) classifiers

To show the importance of the joint learning of local hyperplanes in local online learning , we compare local online learning (LOL) and independent local online learning (I-LOL). Table 3, Table 4 and Figure 4 show that although I-LOL could also model local information from data, it is obvious that I-LOL performs worse overall than LOL due to the lack of joint learning in I-LOL that there is no shared/common information between local prototype domains, especially in the Multi-PIE Identity task.

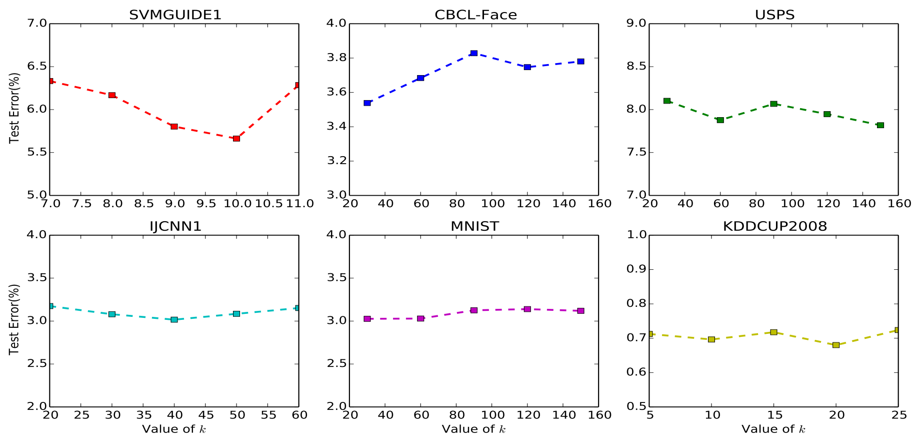


Figure 5: Test error of local online learning (LOL) with respect to the number of prototypes k with fixed $\lambda = 1$

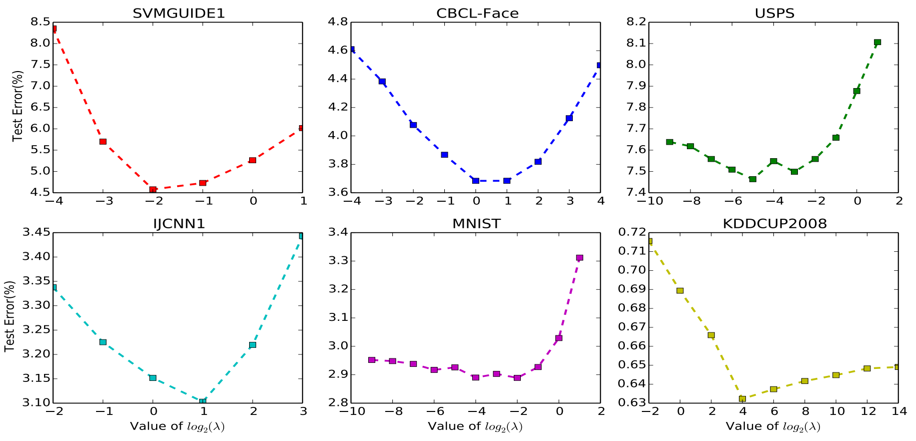


Figure 6: Test error of local online learning (LOL) with respect to the value of λ with fixed number of prototypes $k = 60$

5.4.4. Evaluation of parameters

Here, we would study the influence of two key parameters in local online learning (LOL) , namely the number of prototypes k and the balance parameter λ in Eq. (13) .

Regarding the number of prototypes used in LOL algorithm, Figure 5 shows how test error varies with different numbers of prototypes k. We here selected six of the datasets to display the results. It is indeed that a specific value of k is preferred for LOL to achieve higher accuracy on some datasets, such as SVMGUIDE1, IJCNN1. On some datasets, such as USPS and MNIST, their results are not so sensitive to the value of k.

But fortunately, in these cases, the difference of prediction ability is very small among different k values; for example, whatever value the k is, the results of MNIST are almost the same ($97 \pm 0.06\%$).

Regarding the balance parameter λ, Figure 5 shows that the test error first decreases as λ increases but raises again after λ keeps growing. As mentioned in the previous sections, λ balances the globally shared common component and those local ones. Hence, making a suitable amount of information shared across local hyperplanes will be beneficial to online performance.

5.4.5. Effect of Sequential Clustering in LOL

Figure 7 shows the performance comparison between our proposed LOL using sequential clustering and the LOL using the pre-trained prototypes learned by offline K-means. At the beginning of training, there is indeed clear difference between them. However, after training for a while, the gap becomes smaller and smaller, and the difference vanishes when sufficient training samples have been processed on large-scale dataset. On one hand, as long as more samples are processed online, sequential clustering is approximating the offline K-means and would not lower the effectiveness of LOL. On the other hand, for real world online learning, it is hard to perform an offline clustering over a large dataset in advance, and thus performing sequential clustering seems to be a proper and effective way for clustering sequential stream data.

5.4.6. Summary

From the experimental results above, we summarize the findings and observations as follows:

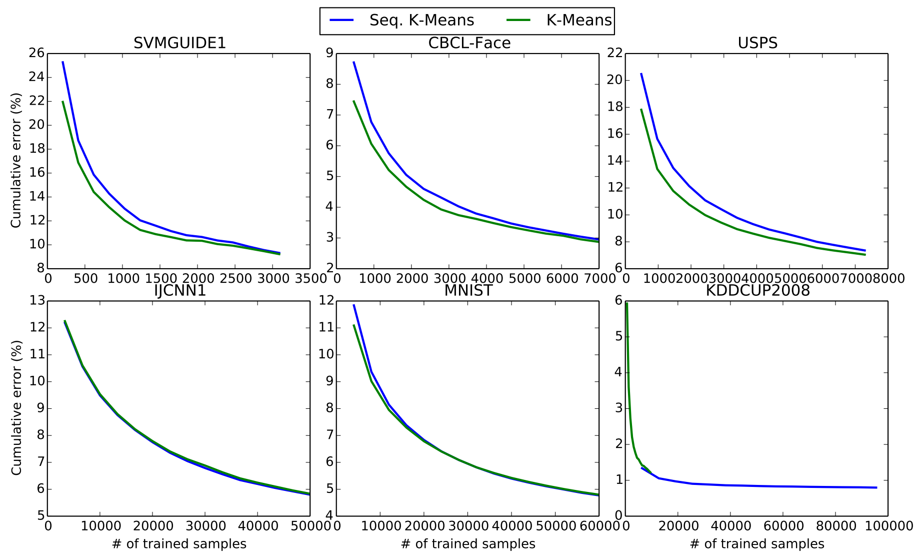


Figure 7: LOL with online (sequential) K-means vs. LOL with offline K-means ($\lambda = 1.0$, $k = 60$).

1) Existing online linear models cannot tackle the nonlinear separation problem;

2) The local approach proposed in this work achieves notably better performance, especially on data with large variations (e.g. Multi-PIE datasets);

3) Compared to the kernel online learning, our results suggest that a local modeling can be an alternative way to deal with the nonlinear data separation problem for one-pass online learning, retaining better and more stable performance;

4) It is important to learn all the local hyperplanes jointly (via a common component shared among them), rather than learning them independently.

6. Conclusion

To address the nonlinearly separable case in the one-pass online learning, we propose a local online learning (LOL) method. LOL learns from sequential data and updates locally sensitive model, so that for each test point a local boundary would be explored for classification, thus solving the nonlinearly separable problem. More importantly, by formulating a share component as information traffic among them, LOL learns those local decision boundaries (i.e. local hyperplanes) jointly but not independently. Theoretical analysis on the connection between LOL and the global passive-aggressive is also given. Our results show that LOL can be an alternative approach that is easy to compute for generalizing online learning for solving nonlinear separation problem without using kernel approximation and any second order modeling.

Acknowledgment

The authors are most grateful for the constructive advice on the revision of the manuscript from the anonymous reviewers. This research was partly supported by Natural Science Foundation Of China (No. 61472456), Guangzhou Pearl River Science and Technology Rising Star Project under Grant 2013J2200068, and the Guangdong Natural Science Funds for Distinguished Young Scholar under Grant S2013050014265. The funding support from Hong Kong Government under its GRF scheme (Ref. No. 152202/14E) and the research grant from Hong Kong Polytechnic University are greatly appreciated.

References

[1] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, Y. Singer, Online passive-aggressive algorithms, The Journal of Machine Learning Research 7 (2006) 551–585.

[2] M. Dredze, K. Crammer, F. Pereira, Confidence-weighted linear classification, in: Proceedings of the 25th International Conference on Machine Learning, ACM, 2008, pp. 264–271.

[3] J. Wang, P. Zhao, S. C. Hoi, Exact soft confidence-weighted learning, in: Proceedings of the 29th International Conference on Machine Learning, 2012, pp. 121–128.

[4] K. Crammer, A. Kulesza, M. Dredze, Adaptive regularization of weight vectors, in: Advances in Neural Information Processing Systems, 2009, pp. 414–422.

[5] Y. Freund, R. E. Schapire, Large margin classification using the perceptron algorithm, Machine learning 37 (3) (1999) 277–296.

[6] J. Kivinen, A. J. Smola, R. C. Williamson, Online learning with kernels, in: Advances in Neural Information Processing Systems, 2001, pp. 785–792.

[7] G. Cavallanti, N. Cesa-Bianchi, C. Gentile, Tracking the best hyperplane with a simple budget perceptron, Machine Learning 69 (2-3) (2007) 143–167.

[8] J. H. Friedman, J. W. Tukey, A projection pursuit algorithm for exploratory data analysis, IEEE Transactions on Computers 100 (9) (1974) 881–890.

[9] O. Dekel, S. Shalev-Shwartz, Y. Singer, The forgetron: A kernel-based perceptron on a budget, SIAM Journal on Computing 37 (5) (2008) 1342–1372.

[10] Z. Wang, K. Crammer, S. Vucetic, Breaking the curse of kernelization: Budgeted stochastic gradient descent for large-scale svm training, The Journal of Machine Learning Research 13 (1) (2012) 3103–3131.

[11] J. Wang, S. C. Hoi, P. Zhao, J. Zhuang, Z.-y. Liu, Large scale online kernel classification, in: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, AAAI Press, 2013, pp. 1750–1756.

[12] L. Ladicky, P. Torr, Locally linear support vector machines, in: Proceedings of the 28th International Conference on Machine Learning, 2011, pp. 985–992.

[13] C. Jose, P. Goyal, P. Aggrwal, M. Varma, Local deep kernel learning for efficient non-linear svm prediction, in: Proceedings of the 30th International Conference on Machine Learning, 2013, pp. 486–494.

[14] P. Wohlhart, M. Kostinger, M. Donoser, P. M. Roth, H. Bischof, Optimizing 1-nearest prototype classifiers, in: IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2013, pp. 460–467.

[15] B. Babenko, M.-H. Yang, S. Belongie, Robust object tracking with online multiple instance learning, IEEE Transactions on Pattern Analysis and Machine Intelligence 33 (8) (2011) 1619–1632.

[16] C.-X. Ren, D.-Q. Dai, Incremental learning of bidirectional principal components for face recognition, Pattern Recognition 43 (2010) 318–330.

[17] A. Mian, Online learning from local features for video-based face recognition, Pattern Recognition 44 (2011) 1068C1075.

[18] H. Wang, X. Wang, J. Zheng, J. R. Deller, H. Peng, L. Zhu, W. Chen, X. Li, R. Liu, H. Bao, Video object matching across multiple non-overlapping camera views based on multi-feature fusion and incremental learning, Pattern Recognition 47 (12) (2014) 3841 – 3851.

[19] M. Barnachon, S. Bouakaz, B. Boufama, E. Guillou, Ongoing human action recognition with motion capture, Pattern Recognition 47 (1) (2014) 238 – 247.

[20] J.-X. Peng, S. Ferguson, K. Rafferty, V. Stewart, A sequential algorithm for sparse support vector classifiers, Pattern Recognition 46 (4) (2013) 1195 – 1208.

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

[22] C. Cortes, V. Vapnik, Support-vector networks, Machine learning 20 (3) (1995) 273–297.

[23] A. Rahimi, B. Recht, Random features for large-scale kernel machines, in: Advances in Neural Information Processing Systems, 2007, pp. 1177–1184.

[24] C. Williams, M. Seeger, Using the nyström method to speed up kernel machines, in: Proceedings of the 14th Annual Conference on Neural Information Processing Systems, no. EPFL-CONF-161322, 2001, pp. 682–688.

[25] Q. Gu, J. Han, Clustered support vector machines, in: proceedings of the sixteenth international conference on artificial intelligence and statistics, 2013, pp. 307–315.

[26] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, J. Attenberg, Feature hashing for large scale multitask learning, in: Proceedings of the 26th Annual International Conference on Machine Learning, ACM, 2009, pp. 1113–1120.

[27] J. MacQueen, et al., Some methods for classification and analysis of multivariate observations, in: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, Vol. 1, Oakland, CA, USA., 1967, pp. 281–297.

[28] R. Gross, I. Matthews, J. Cohn, T. Kanade, S. Baker, Multi-pie, Image and Vision Computing 28 (5) (2010) 807–813.

[29] J. Ma, L. K. Saul, S. Savage, G. M. Voelker, Identifying suspicious urls: an application of large-scale online learning, in: Proceedings of the 26th Annual International Conference on Machine Learning, ACM, 2009, pp. 681–688.

[30] D. Wang, D. Irani, C. Pu, Evolutionary study of web spam: Webb spam corpus 2011 versus webb spam corpus 2006, in: 8th International Conference on Collaborative Computing: Networking, Applications and Worksharing, CollaborateCom 2012, Pittsburgh, PA, USA, October 14-17, 2012, 2012, pp. 40–49.

[31] E. Bingham, H. Mannila, Random projection in dimensionality reduction: applications to image and text data, in: Proceedings of the seventh ACM

SIGKDD international conference on Knowledge discovery and data mining, ACM, 2001, pp. 245–250.

[32] S. Shalev-Shwartz, Y. Singer, N. Srebro, A. Cotter, Pegasos: Primal estimated sub-gradient solver for svm, Mathematical programming 127 (1) (2011) 3–30.

[33] S. C. Hoi, J. Wang, P. Zhao, Libol: A library for online learning algorithms (2012).

[34] N. Djuric, L. Lan, S. Vucetic, Z. Wang, Budgetedsvm: a toolbox for scalable svm approximations, The Journal of Machine Learning Research 14 (1) (2013) 3813–3817.

[35] J. Wang, S. C. Hoi, P. Zhao, J. Zhuang, Z.-y. Liu, Large scale online kernel classification, in: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, AAAI Press, 2013, pp. 1750–1756.