Document downloaded from:

http://hdl.handle.net/10251/125683

This paper must be cited as:

Jorge-Cano, J.; Paredes Palacios, R. (2018). Passive-Aggressive online learning with nonlinear embeddings. Pattern Recognition. 79:162-171. https://doi.org/10.1016/j.patcog.2018.01.019

The final publication is available at

http://doi.org/10.1016/j.patcog.2018.01.019

Copyright Elsevier

Additional Information

Passive-Aggressive online learning with nonlinear embeddings

Javier Jorge*, Roberto Paredes

Pattern Recognition and Human Language Technologies Research Center
Universitat Politècnica de València
Campus de Vera, Camino de Vera, s/n, 46022 Valencia

Abstract

Nowadays, there is an increasing demand for machine learning techniques which can deal with problems where the instances are produced as a stream or in real time. In these scenarios, online learning is able to learn a model from data that comes continuously. The adaptability, efficiency and scalability of online learning techniques have been gaining interest last years with the increasing amount of data generated every day. In this paper, we propose a novel binary classification approach based on nonlinear mapping functions under an online learning framework. The non-convex optimization problem that arises is split into three different convex problems that are solved by means of Passive-Aggressive Online Learning. We evaluate both the adaptability and generalization of our model through several experiments comparing with the state of the art techniques. We improve significantly the results in several datasets widely used previously by the online learning community.

Keywords:
online learning, nonlinear functions, Passive-Aggressive, binary classification, nonlinear embedding
2010 MSC: 00-01, 99-00

*Corresponding author
Email addresses: jjorge@dsic.upv.es (Javier Jorge), rparedes@dsic.upv.es (Roberto Paredes)

1. Introduction

Nowadays, there is an increasing demand for machine learning techniques which can deal with problems where instances are produced in real time. Under these conditions, online learning, a set of machine learning algorithms, can learn models from data that comes continuously. Moreover, online learning techniques are used in large-scale problems as an alternative to batch learning, in order to alleviate the computational cost. Additionally, there are environments such as social media or stock markets where it is essential to deal with the changes of the underlying probability distribution over time. These requirements are adopted by online learning techniques, aiming at learning from each example, by using it once and updating the model at that moment. Therefore, according to this pure online learning scenario, it is not necessary to store or revisit the previous examples. This idea provides a set of simple, fast and efficient techniques regarding computation and memory as well. Recent successful approaches based on this technique have been used to medical diagnosis [1], detecting topics on text streams [2], action recognition [3] or face recognition [4] among others.

1.1. State of the art

Among well-known online learning algorithms, Passive-Aggressive (PA) [5] offers an interesting analytical closed solution for this kind of tasks. This technique in its simplest formulation within a classification framework aims at finding for every new instance a linear model, that is, a weight vector which is close to the current one, but guaranteeing the correct classification of the present instance. This scheme represents a trade-off between the passive behaviour, where the algorithm wants to update the model the least possible, and the aggressive behaviour, where it tries to classify the current instance correctly.

Due to PA was proposed based on linear models, some extensions have been developed to take advantage of the underlying relationships between features. Following this idea, several techniques have been developed assuming that the weight vector follows a Gaussian distribution, such as Confidence-

Weighted Linear Classification (CW) [6], Adaptive Regularization of Weight Vectors (AROW) [7] and Exact Soft Confidence-Weighted Learning (SCW) [8].

PA, as many other linear algorithms, has been extended using the kernel trick as in the Support Vector Machines (SVM) [9, 10, 11] to cope with non-linear problems. However, it is important to note that this extension relies on storing all the previous instances where the model failed, calling them Support Vectors (SV). After that, these methods must compute the kernel of the current sample against all these SV to classify a new instance. For this reason, using the kernel extension leads to a very intensive computational model. Several techniques followed the idea of obtaining a maximum margin solution, as in SVM, in an online manner, approximating [12] or relaxing the conditions to obtain the hyperplane [13].

This drawback is normally alleviated by applying strategies to control the number of these SV. In the literature, the size of this set is seen as a budget that has to be administered. According to this, these techniques are called budget strategies. Several approaches based on the Perceptron algorithm [14] were developed to control the growth of the SV's set, for instance Random Budget Perceptron (RBP) [15], Forgetron [16] and Projectron [17]. In addition, based on PA, in [18] the Budget Passive Aggressive (BPA) learning is proposed. In this paper, authors provide some constraints, related to the budget, that are included in the optimization. Using an index, they decide which vectors to store or discard, obtaining a closed-form solution that explicitly limits the growth of SV. A different approach using PA is presented in [19], in this case with a stochastic sampling that creates new SV from the examples. This probabilistic decision is weighted by the loss suffered from the current instance. This set of techniques represents different heuristic measures to deal with the management of SV.

Other strategies to reduce the computational cost of these kernel-based techniques are focused on Online Gradient Descent (OGD) algorithms [20] such as Bounded Online Gradient Descent (BOGD) [21], where authors propose uniform and non-uniform sampling for discarding SV. Pursuing the optimum projection

is another strategy, as it was proposed in [22], combining this with a budget. A similar idea was used in Budgeted Stochastic Gradient Descent (BSGD) [23] where the number of SV is limited by implementing merging or projecting strategies over this set.

Recently, other techniques have appeared with a different perspective beyond the disadvantage of budget maintenance. In [24] the Fourier Online Gradient Descent (FOGD) and Nÿstrom Online Gradient Descent (NOGD) are proposed. These approaches work as follows: first, they try to approximate kernel functions or the kernel matrix with Fourier/Nÿstrom methods, second they transform data from one space to another using these approximations, and finally, the data is classified in the transformed space performing OGD.

In the same line of avoiding the use of kernel-based strategies, in [25] the Local Online Learning (LOL) is proposed, which poses a model that carries out multiple hyperplanes learning with the combination of PA and clustering. Additionally, there is an extension where the independence between hyperplanes is assumed which is called Independent Local Online Learning (I-LOL). A different approach that avoids the use of the kernel-based methods is the On-Line Sequential Extreme Learning Machine (OS-ELM) [26, 27, 28], which is based on the Extreme Machine Learning framework [29], where the output weights of a single layer neural network are learned by regularized least square method and the input weights are randomly assigned. This last family of techniques has been closely related to neural networks, more than with the online learning community.

In the present paper an approach based on the PA online learning procedure is presented, to deal with binary classification problems by means of nonlinear mapping functions. By using these nonlinear projections, a new space can be obtained where a linear model can manage the problem. Both the projection and the classification process are performed as data arrives, modifying the whole model in order to follow the changes in the distribution accordingly.

2. Problem setting

Under the prism of an online learning binary classification problem, data is provided sequentially. After the observation presented in the round t, $\mathbf{x}_t \in \mathbb{R}^d$, the aim is to predict the correct label, $y_t \in \{-1, +1\}$. To solve this problem, we propose a nonlinear mapping function in order to project data onto a new space where the problem could be managed using a linear model. The prediction $\hat{y}_t$ will be provided as follows:

$$\hat{y}_t = sgn(\mathbf{w} \cdot f_{\mathbf{\Theta}}(\mathbf{x}_t)) = sgn(\mathbf{w} \cdot \mathbf{z}) \tag{1}$$

Where $\mathbf{x}_t$ is the current input vector and $\mathbf{z} \in \mathbb{R}^h$ is the nonlinear projection in the h dimensional space. This projection is performed by the function $f_{\mathbf{\Theta}}(\cdot)$, with parameters $\mathbf{\Theta}$. For this classification task, a weight vector $\mathbf{w} \in \mathbb{R}^h$ will be used and it is considered that $\mathbf{w} \cdot f_{\mathbf{\Theta}}(\mathbf{x}_t) \geq 0$ implies $\hat{y}_t = +1$, and $\mathbf{w} \cdot f_{\mathbf{\Theta}}(\mathbf{x}_t) < 0$ implies $\hat{y}_t = -1$. This process is illustrated in Figure 1.

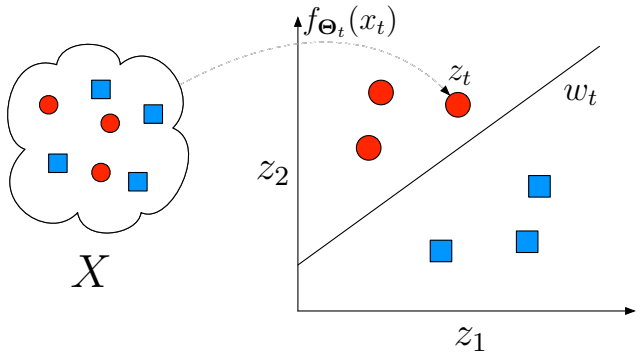

Figure 1: The objective is to learn a good projection for the problem through the nonlinear mapping $f_{\mathbf{\Theta}}(\cdot)$ that ease the classification procedure through the linear model $\mathbf{w}$. Combining these two steps, we will be able to cope with nonlinear problems in an online fashion, updating both nonlinear projection and linear classifier.

2.1. Training with Passive-Aggressive Online Learning

The Passive-Aggressive online learning framework proposes an optimization problem, and it provides a closed form solution for model's update. In the case

of binary classification task, there is the instance $\mathbf{x}$, the label y and the weight vector $\mathbf{w}$, as they have been presented previously. With these elements, the hinge loss is defined as follows:

$$\ell_{\mathbf{w}}(\mathbf{x}, y) = \begin{cases} 0 & y(\mathbf{w} \cdot \mathbf{x}) \geq 1 \\ 1 - y(\mathbf{w} \cdot \mathbf{x}) & \text{otherwise} \end{cases} \tag{2}$$

The purpose of this algorithm is to find the new weight vector $\mathbf{w}_{t+1}$ that is near to the current one but obtaining zero loss with the present instance. Regarding linear models, the quantification of this change is represented by the squared norm of the difference between models. With this distance to minimize the objective and the loss' restriction, the problem is formulated as an optimization. Moreover, we could include a slack variable ξ and the parameter C to cope with label noise and avoiding restricting the update too much, following the steps of [11]. The final constrained optimization problem that results is the following:

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

This optimization pursues updating the model to get another one near to the current, while the loss is under certain threshold which is adjusted by the parameter C. This represents the room for the aggressive behaviour. Solving this optimization, we obtain the closed forms to update the model:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \tau y_t \mathbf{x}_t \tag{4}$$

$$\tau = min \left\{ C, \frac{\ell_{\mathbf{w}_t}(\mathbf{x}_t, y_t)}{||\mathbf{x}_t||^2} \right\} \tag{5}$$

As can be seen, the parameter C ends up controlling the aggressiveness of the process by means of limiting the amount of change. Both a detailed derivation

of these process and theoretical lower bounds obtained for the proposed loss can be found in [5].

2.2. Nonlinear embedding learning with PA

In order to cope with nonlinear problems, projecting the original representation space into a new representation space is required. Eventually, this projection allows using a linear PA model. To this end we propose the following PA optimization problem:

$$\mathbf{\Phi}_{t+1} = \arg\min_{\mathbf{\Phi}} \mathbf{\Omega}(\mathbf{\Phi}, \mathbf{\Phi}_t) \;\; \text{s.t} \;\; \ell(\mathbf{\Phi}; (\mathbf{x}_t, y_t)) = 0 \tag{6}$$

Where $\mathbf{\Phi}$ denotes the set of parameters that compose our model, $\mathbf{\Phi} = \{\mathbf{w}, \mathbf{\Theta}\}$, where $\mathbf{w}$ is the weight vector and $\mathbf{\Theta}$ denotes the parameters for nonlinear mapping functions. Regarding the divergence between current and new parameters, this is denoted by $\mathbf{\Omega}(\mathbf{\Phi}, \mathbf{\Phi}_t)$. The expression $\ell_{\mathbf{\Phi}}(\mathbf{x}_t, y_t)$ represents the loss suffered with the current prediction. By adapting the hinge loss, following plain PA, with the projection $f_{\mathbf{\Theta}}(\mathbf{x}_t)$ we obtain:

$$\ell_{\mathbf{\Phi}}(\mathbf{x}_t, y_t) = \begin{cases} 0 & y_t(\mathbf{w} \cdot f_{\mathbf{\Theta}}(\mathbf{x}_t)) \geq 1 \\ 1 - y_t(\mathbf{w} \cdot f_{\mathbf{\Theta}}(\mathbf{x}_t)) & \text{otherwise} \end{cases} \tag{7}$$

The behaviour of these steps could vary between a conservative/non-conservative regime using an additional parameter C, in the same way as in the original version of PA. If this is included in the optimization, the initial formula will change:

$$\mathbf{\Phi}_{t+1} = \arg\min_{\mathbf{\Phi}} \mathbf{\Omega}(\mathbf{\Phi}, \mathbf{\Phi}_t) \;+\; C\xi \tag{8}$$

$$\text{s.t} \;\; \ell_{\mathbf{\Phi}}(\mathbf{x}_t, y_t) \leq \xi \;\; \text{and} \;\; \xi \geq 0 \tag{9}$$

With the weight vector $\mathbf{w}$ in the projected space and the parameters of this projection $\mathbf{\Theta}$, the formula could be interpreted as below:

$$\mathbf{w}_{t+1}, \mathbf{\Theta}_{t+1} = \arg\min_{\mathbf{w}} \frac{1}{2} \mid\mid \mathbf{w} - \mathbf{w}_t \mid\mid^2 \; + \arg\min_{\mathbf{\Theta}} \mathbf{\Psi}(\mathbf{\Theta}, \mathbf{\Theta}_t) \; + \; C\xi \tag{10}$$

$$\text{s.t} \;\; \ell_{\mathbf{w},\mathbf{\Theta}}(\mathbf{x}_t, y_t) \leq \xi \;\; \text{and} \;\; \xi \geq 0 \tag{11}$$

Where $\mathbf{\Psi}(\mathbf{\Theta}, \mathbf{\Theta}_t)$ represents the divergence between using the current parameters $\mathbf{\Theta}_t$ and the new ones $\mathbf{\Theta}$, that is, between the present embedding and the updated version.

2.3. Combined learning of embeddings and hyperplane

In order to obtain zero loss globally, and following the configuration that is shown before, the modification of $\mathbf{w}_t$ and $\mathbf{\Theta}_t$ is needed.

The whole procedure works as follows: modifying $\mathbf{w}_t$ in order to reduce a ratio $\alpha, 0 < \alpha < 1$, of the current loss, obtaining $\mathbf{w}_{t+1}$. After this step and with this new $\mathbf{w}_{t+1}$, finding a new value for the nonlinear projection, that is, a new vector $\mathbf{z}_{t+1}$, an updated version of the previous one, which leads to a zero loss. Finally, a new set of parameters $\mathbf{\Theta}_{t+1}$ has to be found to provide this updated projection $\mathbf{z}_{t+1}$. The decrease of the loss according to these steps is illustrated in Figure 2.

It is required to tackle this optimization step by step. First, the parameters of the nonlinear function could be fixed. Maintaining the same parameters for the nonlinear mapping on each step, the optimization changes as follows:

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}} \frac{1}{2} \mid\mid \mathbf{w} - \mathbf{w}_t \mid\mid^2 \; + \; C\xi \tag{12}$$

$$\text{s.t} \;\; \ell_{\mathbf{w},\mathbf{\Theta}}(\mathbf{x}_t, y_t) \leq \xi \;\; \text{and} \;\; \xi \geq 0 \tag{13}$$

This is equivalent to the original PA's optimization problem, but using our adapted loss. Therefore, solving this optimization, we obtain the following formulation:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \tau y_t f_{\mathbf{\Theta}}(\mathbf{x}_t) \tag{14}$$

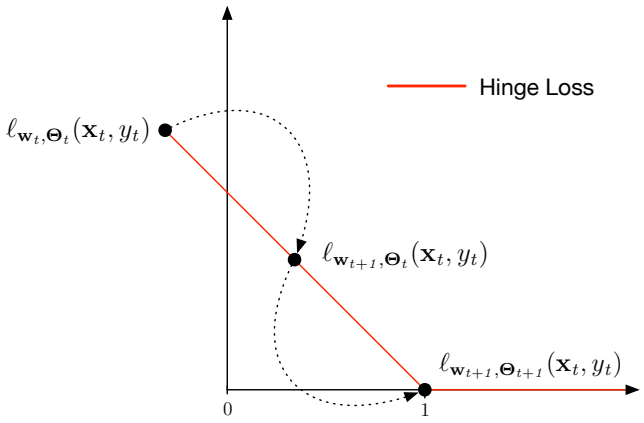

Figure 2: These optimization processes decrease the loss in two steps, first, reducing an α portion of the loss using the changes included in the linear model $\mathbf{w_{t+1}}$, and second, reducing the missing loss with the update of the projection that is performed using the parameters $\mathbf{\Theta_{t+1}}$.

$$\tau = min\left\{C, \frac{\ell_{\mathbf{w}_t,\mathbf{\Theta}}(\mathbf{x}_t, y_t)}{||f_{\mathbf{\Theta}}(\mathbf{x}_t)||^2}\right\} \tag{15}$$

This procedure updates the model over the projected space as can be seen in Figure 3, to get the minimum loss with the current instance. However, the aim is to follow a projection learning approach where the model learns how to project data in another space. It can be done applying PA algorithm to the part that carries out the projection, leaving some loss after updating the weight vector. This remaining loss will be solved changing the projection, that is, updating the parameters of the function $f_{\mathbf{\Theta}}(\cdot)$ that performs the nonlinear mapping.

These steps in the procedure imply different optimization problems that have to be solved in the following sequence. The first step, related to the update of the linear model $\mathbf{w}_t$, leads to the following optimization problem, where an α

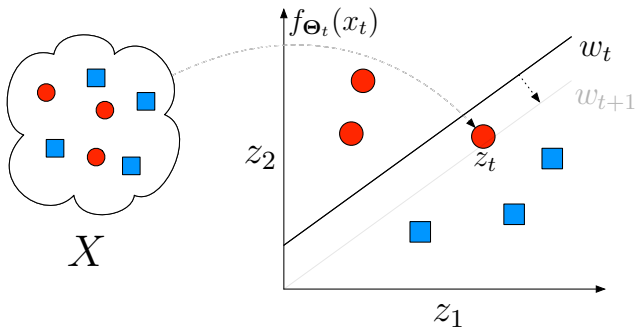

Figure 3: The constrained optimization results in the update of the model controlled by the parameter τ, a by-product of the optimization, and the current tuple $(\mathbf{x}_t, y_t)$. In this case, it is only changed the linear model over the projected space, resulting in the same procedure as in the original PA.

proportion of the total loss is reduced:

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}} \frac{1}{2} \mid\mid \mathbf{w} - \mathbf{w}_t \mid\mid^2 \; + \; C\xi \tag{16}$$

$$\text{s.t} \;\; \ell_{\mathbf{w},\boldsymbol{\Theta}}(\mathbf{x}_t, y_t) = \alpha \cdot \ell_{\mathbf{w}_t,\boldsymbol{\Theta}_t}(\mathbf{x}_t, y_t) \;\; \text{and} \;\; \ell_{\mathbf{w},\boldsymbol{\Theta}}(\mathbf{x}_t, y_t) \leq \xi \;\; \text{and} \;\; \xi \geq 0 \tag{17}$$

Solving this constrained optimization, the updating rule and $\tau_{\mathbf{w}_t}$ result in the following:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_{\mathbf{w}_t} y_t \mathbf{z}_t \tag{18}$$

$$\tau_{\mathbf{w}_t} = min \left\{ C, \frac{(1-\alpha)\ell_{\mathbf{w}_t,\boldsymbol{\Theta}_t}(\mathbf{x}_t, y_t)}{||\mathbf{z}_t||^2} \right\} \tag{19}$$

To alleviate the notation, $f_{\boldsymbol{\Theta}_t}(\mathbf{x}_t)$ and $\mathbf{z}_t$ are used equally. Once $\mathbf{w}_{t+1}$ is obtained, a new value for the projection, $\mathbf{z}_{t+1}$, is required in order to reduce the remaining loss:

$$\mathbf{z}_{t+1} = \arg\min_{\mathbf{z}} \frac{1}{2} ||\mathbf{z} - \mathbf{z}_t||^2 \;\; \text{s.t} \;\; \ell_{\mathbf{w}_t,\boldsymbol{\Theta}}(\mathbf{x}_t, y_t) = 0 \tag{20}$$

At this level, the change in the resulting projected sample is not bounded, because it is pursued a solution that leads to the lowest loss. The previous

optimization problem is solved with the following closed form of the updating rule and $\tau_{\mathbf{z}_t}$:

$$\mathbf{z}_{t+1} = \mathbf{z}_t + \tau_{\mathbf{z}_t} y_t \mathbf{w}_t \tag{21}$$

$$\tau_{\mathbf{z}_t} = \frac{\ell_{\mathbf{w}_t, \mathbf{\Theta}_t}(\mathbf{x}_t, y_t)}{||\mathbf{z}_t||^2} \tag{22}$$

This could imply an aggressive displacement but it will be controlled by the following step. After updating $\mathbf{z}_t$ to $\mathbf{z}_{t+1}$, the third optimization problem requires another set of parameters $\mathbf{\Theta}_{t+1}$ in order to provide the projection $\mathbf{z}_{t+1}$ that leads to zero loss with the current sample. For doing this, a function that can provide any value in the range of this last modification is required. A representation of updating this projection is presented in Figure 4.

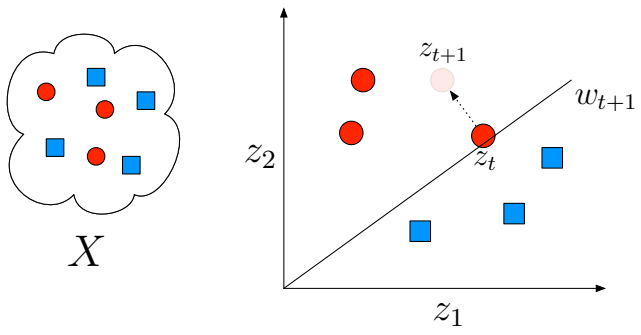

Figure 4: After updating the weight vector, modifying the position of the current instance in the projected space (light circle) to a new one (bold circle) leads to a loss equal to zero.

2.4. Passive Aggressive on Max-Out functions

Max-Out nonlinear functions [30] were proposed to improve the performance of Neural Networks with the ability to mimic different activation functions on demand. Inspired by this, the same principle based on the combination of different hyperplanes could be applied, to learn the shape of a nonlinear function

that fits the problem. The Max-Out function is composed of k hyperplanes, and the output is the maximum scalar product among all these hyperplanes.

Similar to this, it is proposed a set of hyperplanes, $\mathbf{u}_{i,j}$, defining the following formula for the general case where $\mathbf{x}, \mathbf{u}_{i,j} \in \mathbb{R}^d$ and $\mathbf{z} \in \mathbb{R}^h$:

$$f_{\mathbf{\Theta}}(\mathbf{x}) = \mathbf{z} = \tag{23}$$

$$\left[\underbrace{max(\mathbf{x} \cdot \mathbf{u}_{1,1}, \mathbf{x} \cdot \mathbf{u}_{1,2}, \ldots, \mathbf{x} \cdot \mathbf{u}_{1,k})}_{z_1}, \ldots, \underbrace{max(\mathbf{x} \cdot \mathbf{u}_{h,1}, \mathbf{x} \cdot \mathbf{u}_{1,2}, \ldots, \mathbf{x} \cdot \mathbf{u}_{h,k})}_{z_h} \right] \tag{24}$$

Being h the dimensionality of the projection $\mathbf{z}$, and k the number of hyperplanes per each dimension. The set of parameters for this nonlinear projection are summarized as below:

$$\mathbf{\Theta} = \{\mathbf{u}_{i,j} : i \in [1, \ldots, h]; j \in [1, \ldots, k]; \mathbf{u}_{i,j} \in \mathbb{R}^d\} \tag{25}$$

With this incorporation, a third optimization problem is posed to obtain the required value for the projection $\mathbf{z}_{t+1}$ solving h different regression problems:

$$\mathbf{u}_{i,j}^{t+1} = \arg\min_{\mathbf{u}_{i,j}} \frac{1}{2} ||\mathbf{u}_{i,j} - \mathbf{u}_{i,j}^{t}||^2 \; + \; C_r \xi \tag{26}$$

$$\text{s.t} \;\; \ell_\epsilon(\mathbf{u}_{i,j}^{t}; (\mathbf{x}_t, z_t^i)) \leq \xi \;\; \text{and} \;\; \xi \geq 0 \tag{27}$$

Where $\mathbf{u}_{i,j}$ is the $j-$hyperplane associated with the dimension i of the projection, while z_{t+1}^i is the particular output on this dimension, and $\mathbf{x}_t$ is the current sample. Here appears a new parameter, C_r to control the aggressiveness of the projection in this regression procedure. An example of this last step is shown in Figure 5. For this optimization, the epsilon loss function, commonly applied in regression problems, is used. This function is defined as detailed

below:

$$\ell_\epsilon(\mathbf{u}_{i,j}; (\mathbf{x}_t, z^i_{t+1})) = \begin{cases} 0 & |(\mathbf{u}_{i,j} \cdot \mathbf{x}_t) - z^i_{t+1}| \leq \epsilon \\ |(\mathbf{u}_{i,j} \cdot \mathbf{x}_t) - z^i_{t+1}| - \epsilon & \text{otherwise} \end{cases} \tag{28}$$

This optimization has also a closed solution provided by PA formulation:

$$\mathbf{u}^{t+1}_{i,j} = \mathbf{u}_{i,j} + sign(z^i_{t+1} - \mathbf{u}_{i,j} \cdot \mathbf{x}_t)\tau_{\mathbf{u}_{i,j}}\mathbf{x}_t \tag{29}$$

$$\tau_{\mathbf{u}_{i,j}} = min\left\{C_r, \frac{\ell_\epsilon(\mathbf{u}_{i,j}; (\mathbf{x}_t, z^i_{t+1}))}{||\mathbf{x}_t||^2}\right\} \tag{30}$$

Regarding what hyperplane has to be used for each dimension, two strategies have been evaluated. First, selecting the vector that provides the product $\mathbf{u}_{i,j} \cdot \mathbf{x}_t$ nearest to z^i_{t+1}, in order to minimize the divergence of the update, $\mathbf{\Psi}(\mathbf{\Theta}, \mathbf{\Theta}_t)$, following a conservative behaviour according to the changes in the model. Second, selecting the hyperplane that had provided the maximum value among the other in the same dimension, focusing on changing directly parameters that have more influence in the model. The second method has been chosen due to the better results obtained empirically.

The PAMO (Passive-Aggressive Max-Out) algorithm 1 provides a whole picture of the proposed method[1]. Note that we have normalized the input $\mathbf{x}_t$ and projected vector $\mathbf{z}_t$ to use unit vectors in order to ease the PA steps. We have proposed two versions of our algorithm, one where the parameters $\mathbf{\Theta}$ are updated only when there is loss and another where these parameters will be always updated, independently of the loss. Performing this update in every step is forcing our model into learning the inner normalization, enabling a fast adaptation of the parameters to this normalized space. We named this two versions as PAMO-I and PAMO-II respectively.

[1] Code: https://goo.gl/dWdYbf

Algorithm 1: PAMO, Passive-Aggressive Max-Out algorithm

Input : $\mathcal{X}, h, k, C, C_r, \alpha$

Output: $\mathbf{w}, \mathbf{\Theta} = \{\mathbf{u}_{i,j} : i \in [1, \cdots, h]; j \in [1, \cdots, k]\}$

$\mathbf{w}_0, \mathbf{\Theta}_0 \leftarrow initialize()$

for $(\mathbf{x}_t, y_t) \in \mathcal{X}$ **do**

  #Normalize the input

  $\mathbf{x}_t \leftarrow \mathbf{x}_t / ||\mathbf{x}_t||$

  #Perform the nonlinear projection

  $\mathbf{z}_t \leftarrow f_{\mathbf{\Theta}_t}(\mathbf{x}_t)$

  #Normalize the projection

  $\mathbf{z}_t \leftarrow \mathbf{z}_t / ||\mathbf{z}_t||$

  #Predict

  $\hat{y}_t \leftarrow sign(\mathbf{z}_t^{\mathbf{T}} \cdot \mathbf{w}_t)$

  #Suffer loss $\ell_{\mathbf{w},\mathbf{\Theta}}(\mathbf{x}_t, y_t)$

  if $\ell_{\mathbf{w}_t,\mathbf{\Theta}_t}(\mathbf{x}_t, y_t) > 0$ **then**

    #PA modifying hyperplane and the projected sample

    $\mathbf{w}_{t+1} \leftarrow PA\_class(\mathbf{z}_t, \mathbf{w}_t, \hat{y}_t, y_t, C, \alpha)$

    $\mathbf{z}_{t+1} \leftarrow PA\_class(\mathbf{w}_{t+1}, \mathbf{z}_t, \hat{y}_t, y_t)$

    #PA modifying the projection (always performed in PAMO-II)

    for $i \in [1, .., h]$ **do**

      $\mathbf{u}_{i,j}^t \leftarrow select\_vector(\{\mathbf{u}_{i,n}^t : n \in [1, \cdots, k]\})$

      $\mathbf{u}_{i,j}^{t+1} \leftarrow PA\_regr(\mathbf{x}_t, \mathbf{u}_{i,j}^t, z_{t+1}^i, C_r)$

    end

  end

end

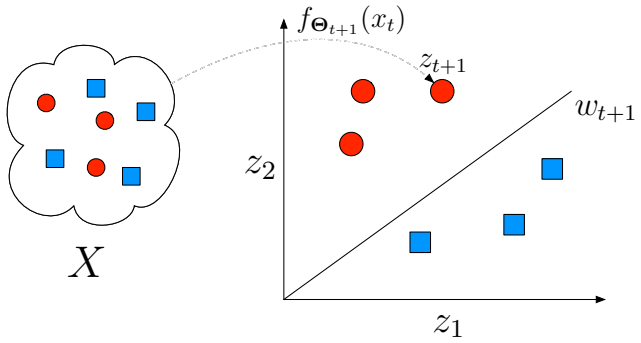

Figure 5: Finally, it is only left the update of the parameters of the nonlinear mapping, in order to obtain the new position in the projected space, $\mathbf{z_{t+1}}$. This step encourages the process to modify the projection to get a good space for performing the classification using $\mathbf{w_{t+1}}$

Summing up, the parameters of the whole model are the following: the dimension of the projection, h, the number of hyperplanes for each dimension, k, the aggressiveness of updates that are controlled by parameters C and C_r, and α that controls the proportion of loss that is solved by each part of the model.

Regarding the computational complexity, with every instance's update there are involved different vector/matrix multiplications. First, the projection, that is a vector-matrix multiplication and maximum selection, where the dimension depends on the number of dimensions for the projection and for hyperplanes that are considered. Once this projection is performed, the dot product between $\mathbf{z}$ and $\mathbf{w}$ provides the label. Considering the dimensions that are involved in this operations, the complexity is $O(d{\cdot}k{\cdot}h{+}h)$. The difference in complexity between versions involves the projection's update, which will be omitted if there is no loss in PAMO-I, while it is always carried out in PAMO-II. All these operations are done efficiently with the modern architectures and they are extremely fast. This advantage enables the model process instances as they come.

3. Analysis and limitations of the model

Similarly to other online learning techniques, we could analyze the bound for the accumulated loss. Unfortunately, the non-convex nature of the proposed method makes extremely difficult to provide any useful bound, beyond to establish two different scenarios:

- If $\alpha = 0$, it implies a projection using the initial weights and a standard PA over the resulting random projection, that is modifying only $\mathbf{w}$. If we set this value in Formula 17, we obtain the Formula 3, that is the basic PA with the same error bound analysis but performing it over a random projection of the input.

- If $\alpha = 1$, it implies modifying the projection to solve the problem with a fixed $\mathbf{w}$. The solution involves a regression PA at every z_i, represented by the Formula 27 that coincides with the original formulation of the PA for regression.

On the other hand, one of the limitations of the model emerges when we consider using more than just one non-linear projection. According to this, this PAMO's configuration could be considered as a neural network model with more than one hidden layer h. Under the neural network perspective, regarding Deep Learning in particular, representation learning could be performed by more than one hidden layer. Therefore we could be interested in including these ideas in our model to solve much more complicated problems.

However, adding more than one non-linear functions makes even more difficult to obtain a closed-form solution, as it was shown in 3. To solve this problem there are some possibilities, one of them is to expand this first projection to be composed of groups of non-linear functions, each one assigned to each dimension of the next projection. This approach will widen the model exponentially. According to this, our model becomes quite large and impractical in order to provide a closed-form solution.

Nevertheless, beyond the above difficulties, the here proposed technique has demonstrated good performance in all the experiments, as we can see in the next section.

4. Experiments and results

On this section, we will perform experiments to evaluate three main characteristics: the ability for tracking changes of data distributions, the capacity to learn nonlinear problems and the generalization provided by the model. For these tasks, we will use a synthetic and some widely used datasets for binary problems.

4.1. Sensitivity analysis of parameter settings

In this section, we have conducted experiments to analyse the influence of parameters on two different datasets, to evaluate how they affect the model. We have focused our analysis on parameters C, C_r and α which control and balance the behaviour of the algorithm. To limit the exploration to a two-dimensional space of parameters, we have decided to set up C equal to C_r from these experiments and onwards, calling it C.

The Figure 6 shows the variations between the parameters C and α for two different datasets and the effect on test error rate. These results are remarking that in approximately separable data, classic PA algorithms get good results. Variations of parameters have less influence than in a non-separable, as in SVMGUIDE dataset, where plain PA obtains 25.2% of error, providing the noisy surface that is seen in the Figure 6. However, in both cases, large α or low C values provide good results, limiting the influence of one over the other.

The Figure 7 represents the cumulative error rate over these training sets considering different values for α and C separately, showing the α's stability when C is fixed, and vice-versa with a C's value among 0.1 and 100.

As a consequence of these results, we have selected the default values $C = C_r = 0.125$ and $\alpha = 0.9$. With these parameters set, we have evaluated different

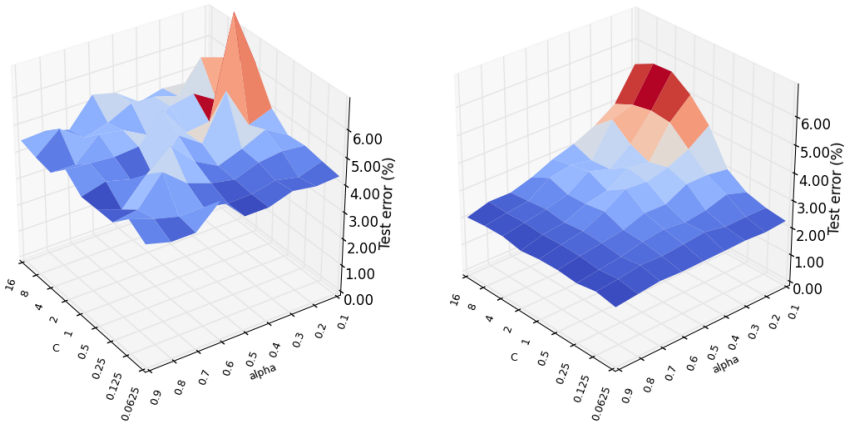

Figure 6: Test error rate for different values of the parameters $\alpha \in [0.1, 0.2, ...0.9]$ and $C, C_r \in [2^{-4}, 2^{-3}, ..., 2^3, 2^4]$, for SVMGUIDE1 (left) and CBLCFace dataset (right).

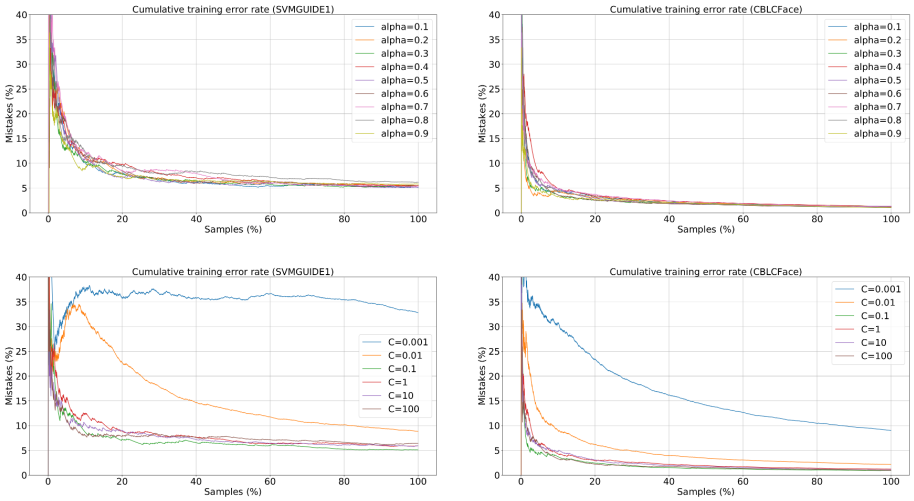

Figure 7: Cumulative training error rate for different values of the parameters α and C, for SVMGUIDE1 and CBLCFace datasets.

dimensions among $\{2^1, 2^2, ..., 2^8\}$, choosing those that provide the best results in validation. We have fixed to 2 the number of hyperplanes to the Max-Out projection.

4.2. Online learning adaptation

In order to evaluate the capacity of online adaptation and as a sanity check of our algorithm, we have generated a two dimensional XOR synthetic problem with 1000 samples using 4 Gaussian distributions. To simulate changes along time the means of Gaussians have been rotated counterclockwise every 100 samples. With the synthetic data generated, we have evaluated if our model might follow these changes over time.

Regarding the parameters of our model, we use the following: $h = 3$ and $k = 2$. The learning parameters C and C_r have been selected by validation from values in $\{2^{-4}, 2^{-3}, ..., 2^3, 2^4\}$, and the balancing parameter α has been evaluated over the range $\{0.1, 0.2, ...0.9\}$. The parameters that have shown better results have been selected. For this experiment and the following, all the weights have been initialized randomly on the interval $\{-0.1, 0.1\}$, distributed uniformly, and posteriorly an orthogonalization of the hyperplanes for each dimension has been performed. Furthermore, we have normalized data to a mean of 0 and a standard deviation of 1.

In order to analyse the adaptability to changes in the original distribution, we have generated plots every 100 samples with the resulting decision boundary and plotting the last 100 instances as well. We can see on Figure 8 the evolution of decision boundaries following the modifications of underlying class-dependent data distributions.

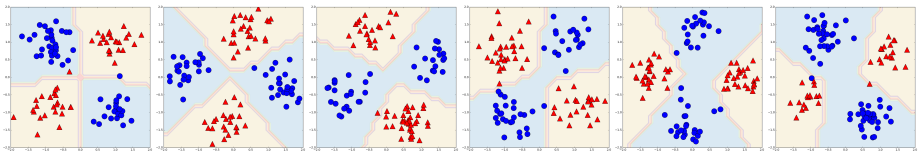

Figure 8: From left to right, decision boundaries and samples every 100 samples where the means are rotated counterclockwise.

The model tracks the changes of class-distributions despite the relatively low amount of samples that produces these changes (100). The nonlinear projection has modified the representation in order to adapt the distribution and forget

the influence on the class boundaries of previous samples.

4.3. Batch and Online Binary Classification tasks

We have performed experiments[2] with datasets widely used on online learning benchmarks. These are in the LIBSVM repository[3] and in the CBCL-Face website [4]. We have considered the same data partitions provided by the repositories, but in the case of WEBSPAM, we have used the 200.000 first samples for training, and the rest for testing.

We have evaluated two measures that are commonly used in previous online learning references. First, test error rate over unseen samples and second, cumulative misclassification over the training partition.

Test error rate means that the training partition is used for learning the model and the evaluation is performed with the test partition, thus simulating a batch learning but only using once each training sample. We have performed experiments with several techniques in the literature: First order approaches (PA-I and PA-II) as well as second order approaches (CW, AROW, SCW-I and SCW-II). For these methods we have performed cross-validation for selecting the parameters of each technique, using the implementation provided by LIBOL [31]. For the other approaches based on Budget or Kernel approximation (BSGD, FOGD and NOGD), LOL and I-LOL techniques, we have taken the results from [25]. Each experiment was repeated 20 times with a random permutation of the data and taking the average and the deviation.

For selecting the parameters in our model we have followed the same cross-validation method as in the previous experiment. First we have evaluated the influence of PA parameters: C, C_r and α. We have used C equal to C_r in all the experiments, referring them as only C. After the process, we have selected the default values $C = C_r = 0.125$ and $\alpha = 0.9$. With these parameters fixed, we have evaluated a different number of inner dimensions (h) among $\{2^1, 2^2, ..., 2^8\}$

[2] Code: https://goo.gl/dWdYbf

[3] https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/

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

and the number of hyperplanes per dimension (k) among $\{2^1, 2^2, ..., 2^5\}$, and we have selected the ones that give the best results in validation.

The results of the experiments for test error rate are summarized in Table 1. We also show in the last row the number of hyperplanes and dimensions of the nonlinear projection. The Figure 9 shows the cumulative training error rate for 10 repetitions for some datasets that were used for the experimentation, illustrating a similar evolution independently of the instances' order.

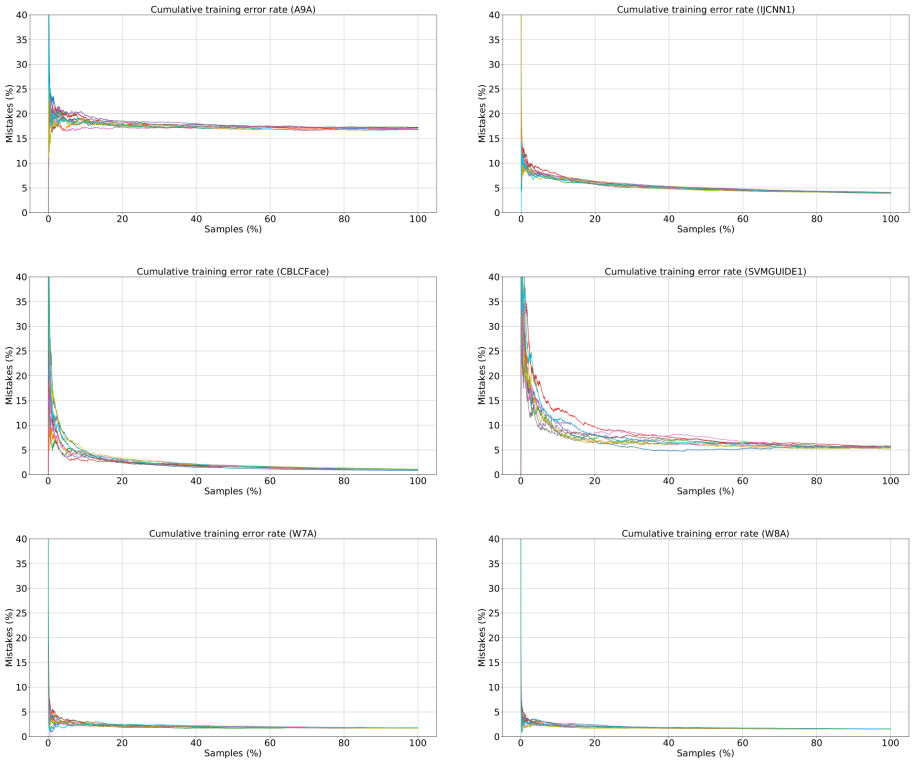

Figure 9: Cumulative training error rate for datasets A9A, .

We have improved the results provided in the literature of online learning techniques in terms of test error rate. In the datasets that seem nonlinear, according to the poor results of linear techniques such as PA, our model outperforms the approaches based on first and second order statistics and improves the results in comparison with other budget techniques. In these scenarios our model

Table 1: Comparative results of test error (%).

Algorithms/Data	SVMGUIDE1	CBCL Face	IJCNN1	WEBSPAM
(#tr/#te/d)	*3k /4k /4*	*6.9k /24k /361*	*49.9k /91.7k /22*	*200k /150k /254*
PA-I	25.19±7.12	3.75±1.31	7.85±0.33	7.94±0.72
PA-II	25.84±4.50	3.43±0.80	8.07±0.39	8.27±0.55
CW	22.92±2.34	3.09±0.15	11.20±2.09	10.62 ±0.79
AROW	20.99±0.09	2.88±0.09	8.22±0.08	7.28±0.01
SCW-I	21.08±0.20	2.73±0.09	6.44±0.08	6.58±0.02
SCW-II	21.29±0.18	2.64±0.06	6.93±0.15	6.59±0.04
BSGD	5.73±0.01	2.14±0.04	4.36±0.00	5.27±0.00
FOGD	7.68±0.01	6.47±0.03	11.48±0.07	5.68±0.00
NOGD	5.75±0.01	6.38±0.02	11.99±0.06	14.82±0.00
I-LOL	6.54±0.02	5.34±0.01	4.10±0.00	5.56±0.00
LOL	5.26±0.01	3.68±0.01	3.15±0.00	4.95±0.00
PAMO-I	**4.13±0.59**	**1.99±0.09**	**2.77±0.32**	**2.19±0.47**
PAMO-II	**4.35±0.67**	**1.79±0.08**	**2.53±0.23**	**1.82±0.09**
Dimension × Hyperplanes	64x2	64x4	256x2	256x2

can perform complex decision boundaries but with a limited model complexity. On the datasets with high dimensions, as CBCLFace and WEBSPAM with 361 and 254 features respectively, our method obtains also significant improvements. We have also obtained better results in datasets with a high number of samples, such as IJCNN1, with 49.990/91.701, and WEBSPAM, with 200.000/150.000. On the other hand, the number of projected dimension times the number of hyperplanes gives us an idea of the budget employed by PAMO. In this sense, our approach obtains significantly better results using a model complexity similar to the other budgeted algorithms.

For evaluating the online classification performance by the cumulative mistake rate over training we have selected techniques from the state of the art, as BPA-S, FOGD and NOGD, and we have included other recent approach based

on budget strategies as BOGD.

The budget used in these approaches is $B = 100$, as reported in the literature. Similarly, for FOGD the number of Fourier components have fixed to $D = 4 \cdot B$. Regarding our model, we have set a fixed configuration for all the experiments of 64 dimensions and 2 hyperplanes for each one, maintaining the other parameters as the previous experiments. For this evaluation, we have considered the same datasets that have been used previously with the mentioned techniques. These datasets also come from the LIBSVM site and they are publicly available. Table 2 shows the average and standard deviation over 20 experiments.

Table 2: Comparative results of mistake rate (%).

Algorithms/Data	a9a	w7a	w8a	IJCNN1
(#tr/d)	*48.8k/123*	*24.6k/300*	*64.7k/300*	*141k/22*
BPA-S	21.1 ± 0.20	2.99 ± 0.06	2.84 ± 0.03	11.33 ± 0.04
BOGD	27.9 ± 0.20	3.49 ± 0.16	3.43 ± 0.08	11.67 ± 0.13
FOGD	17.4 ± 0.10	2.75 ± 0.03	2.43 ± 0.03	9.06 ± 0.05
NOGD	17.4 ± 0.20	2.98 ± 0.01	2.92 ± 0.03	9.55 ± 0.01
PAMO (I)	**16.99** ± 0.07	**1.72** ± 0.04	**1.47** ± 0.02	**2.87** ± 0.05
PAMO (II)	**17.11** ± 0.10	**1.76** ± 0.04	**1.53** ± 0.01	**3.09** ± 0.11

PAMO improves the state of the art in this kind of evaluation that measures the cumulative error rate along the training process. Our approach overcomes the other techniques based on kernels using a similar complexity. It is important to note that our model does not require complex procedures as merging strategies, matrix decomposition or sampling distributions, as it is done in BPA-S, NOGD, FOGD and BOGD, and it results in a fast procedure that does not require either much memory nor computational resources.

5. Conclusions

In this paper, we propose a new online learning model based on nonlinear embeddings. Max-out functions have been chosen to provide such this nonlinearity because it provides two interesting properties. First, Max-out can provide any needed value of the alternative representation. Second, Max-out can be optimized in a linear and closed form. The whole optimization problem is solved using three different Passive-Aggressive procedures.

We have evaluated the adaptability of the model as well as its generalization capacity through synthetic data and widely used benchmarks. We provided a very fast online learning model, that does not rely on kernels. Moreover, the budget model complexity is fixed by means of determining the dimensionality of the alternative representation space and the number of Max-out pieces. We proposed two algorithms, PAMO-I and PAMO-II, carrying out experiments on datasets that are widely used by the online research learning community. Our proposed algorithms have improved the results regarding the state of the art such as first and second order methods as well as budget and kernel approximation techniques.

Although our model could be extended with more than one non-linear function, resembling a Neural Network with different hidden layers, deeper models are impractical due to the model's exponential growth. Future research could be oriented to provide an extension mechanism that will be able to alleviate this structural issue.

Acknowledgments

This work was developed in the framework of the PROMETEOII/2014/030 research project "Adaptive learning and multimodality in machine translation and text transcription", funded by the Generalitat Valenciana. The work of the first author is financed by Grant FPU14/03981, from the Spanish Ministry of Education, Culture and Sport.

References

[1] Y. Motai, N. A. Siddique, H. Yoshida, Heterogeneous data analysis: Online learning for medical-image-based diagnosis, Pattern Recognition 63 (Supplement C) (2017) 612 – 624.

[2] D. Tu, L. Chen, M. Lv, H. Shi, G. Chen, Hierarchical online nmf for detecting and tracking topic hierarchies in a text stream, Pattern Recognition 76 (Supplement C) (2018) 203 – 214.

[3] V. Bloom, V. Argyriou, D. Makris, Linear latent low dimensional space for online early action recognition and prediction, Pattern Recognition 72 (Supplement C) (2017) 532 – 547.

[4] A. Mian, Online learning from local features for video-based face recognition, Pattern Recognition 44 (5) (2011) 1068 – 1075.

[5] 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.

[6] 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.

[7] K. Crammer, A. Kulesza, M. Dredze, Adaptive regularization of weight vectors, in: Advances in neural information processing systems, 2009, pp. 414–422.

[8] K. Crammer, M. Dredze, F. Pereira, Exact convex confidence-weighted learning, in: Advances in Neural Information Processing Systems, 2009, pp. 345–352.

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

[10] N. Cristianini, J. Shawe-Taylor, An introduction to support vector machines and other kernel-based learning methods, Cambridge university press, 2000.

[11] V. N. Vapnik, Statistical learning theory, Vol. 1, J. Wiley & Sons, 1998.

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

[13] Y. Li, P. M. Long, The relaxed online maximum margin algorithm, in: Advances in neural information processing systems, 2000, pp. 498–504.

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

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

[16] O. Dekel, S. Shalev-Shwartz, Y. Singer, The forgetron: A kernel-based perceptron on a fixed budget, in: Advances in neural information processing systems, 2005, pp. 259–266.

[17] F. Orabona, J. Keshet, B. Caputo, Bounded kernel-based online learning, The Journal of Machine Learning Research 10 (2009) 2643–2666.

[18] Z. Wang, S. Vucetic, Online passive-aggressive algorithms on a budget, in: International Conference on Artificial Intelligence and Statistics, 2010, pp. 908–915.

[19] J. Lu, P. Zhao, S. C. Hoi, Online sparse passive aggressive learning with kernels, in: Proceedings of the 2016 SIAM International Conference on Data Mining, SIAM, 2016, pp. 675–683.

[20] J. Kivinen, A. J. Smola, R. C. Williamson, Online learning with kernels, in: Advances in neural information processing systems, 2001, pp. 785–792.

[21] P. Zhao, J. Wang, P. Wu, R. Jin, S. C. Hoi, Fast bounded online gradient descent algorithms for scalable kernel-based online learning, arXiv preprint arXiv:1206.4633.

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

[23] 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.

[24] J. Wang, P. Zhao, S. HOI, J. Zhuang, Z.-y. Liu, Large scale online kernel classification, IJCAI'13 Proceedings of the Twenty-Third international joint conference on Artificial Intelligence, 2013.

[25] Z. Zhou, W.-S. Zheng, J.-F. Hu, Y. Xu, J. You, One-pass online learning: A local approach, Pattern Recognition 51 (2016) 346–357.

[26] G.-B. Huang, N.-Y. Liang, H.-J. Rong, P. Saratchandran, N. Sundararajan, On-line sequential extreme learning machine., Computational Intelligence 2005 (2005) 232–237.

[27] X. Wang, M. Han, Online sequential extreme learning machine with kernels for nonstationary time series prediction, Neurocomputing 145 (2014) 90–97.

[28] Y. Lan, Y. C. Soh, G.-B. Huang, Ensemble of online sequential extreme learning machine, Neurocomputing 72 (13) (2009) 3391–3395.

[29] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, in: Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on, Vol. 2, IEEE, 2004, pp. 985–990.

[30] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, Y. Bengio, Maxout networks, arXiv preprint arXiv:1302.4389.

[31] S. C. Hoi, J. Wang, P. Zhao, Libol: A library for online learning algorithms, The Journal of Machine Learning Research 15 (1) (2014) 495–499.