A kernel-based Perceptron with dynamic memory

Wenwu He, a, *, Si Wu b

a Department of Mathematics and Physics, Fujian University of Technology, Fuzhou, Fujian 350108, China
b Lab of Neural Information Processing, Institute of Neuroscience, Chinese Academy of Sciences, Shanghai 200031, China

A R T I C L E   I N F O

Article history:
Received 29 September 2010
Received in revised form 23 July 2011
Accepted 26 July 2011

Keywords:
On-line learning
Pattern classification
Kernel methods
Perceptron and projection

A B S T R A C T

In this study, we propose a dynamical memory strategy to efficiently control the size of the support set in a kernel-based Perceptron learning algorithm. The method consists of two operations, namely, the incremental and decremental projections. In the incremental projection, a new presented instance is either added to the support set or discarded depending on a predefined rule. To diminish information loss, we do not throw away those discarded examples cheaply, instead their impact to the discriminative function is sustained by a projection technique, which maps the modified discriminative function into the space spanned by the original support set. When a new example is added to the support set, the algorithm moves to the decremental projection. We evaluate the minimum information loss by deleting one instance from the support set. If this minimum information loss is less than a tolerable threshold, then the corresponding instance is removed, however, its contribution to the discriminative function is reserved by the projection technique. By this, our method can on one hand keep a relatively small size of the support set and on the other hand achieve a high classification accuracy. We also develop a method which sets a budget for the size of the support set. We test our approaches to four benchmark data sets, and find that our methods outperform others in either having higher classification accuracies when the sizes of their support sets are comparable or having smaller sizes of the support sets when their classification accuracies are similar.

© 2011 Elsevier Ltd. All rights reserved.

1. Introduction

Online learning, which aims to acquire a function of data based on sequentially presented examples in an open-ended way, is important in many practical applications, e.g., when data is non-stationary, the set of training examples is not available at a single moment, or the memory buffer of a learning machine is limited (see, e.g., Crammer, Dekel, Keshet, Shalev-Shwartz, and Singer (2006), Freund and Schapire (1999), He (2009), Kivinen and Warmuth (1997), Kivinen, Smola, and Williamson (2004)). How to properly control the memory size and in the mean time achieve high prediction accuracy is a big challenge in online learning.

To see this problem clearly, let us consider kernel-based on-line Perceptron as an example (Freund and Schapire, 1999; Rosenblatt, 1958). Suppose our task is to learn the mapping \( f : X \rightarrow \mathbb{R} \) based on a set of examples \( \{(x_i, y_i)\}_{i=1}^T \), where \( x_i \in X \) is an instance, \( y_i = \pm 1 \) the corresponding class label and \( T \) the number of examples. We consider the learning in a reproducing kernel Hilbert space (RKHS) \( \mathcal{H} \), that is \( f \in \mathcal{H} \). Denote the discriminating function learned by Perceptron at moment \( t \) to be

\[
 f_{t+1}(\cdot) = \sum_{x_i \in S_{t+1}} \alpha_i k(x_i, \cdot), \tag{1}
\]

where \( \alpha_i = y_i, k(x_i, \cdot) \) is the kernel function satisfying the reproducing property, \( (f(\cdot), k(x, \cdot)) = f(x) \), and the symbol \( \langle \cdot, \cdot \rangle \) denoting the inner product (Scholkopf & Smola, 2001). \( S_t = \{x_i, 1 \leq i \leq t \mid y_i \neq \hat{y}_i \} := \text{sign}[f_t(x_i)] \} \) is called the support set, which holds the instances the classifier keeps in the memory for building up the hypothesis.

On-line Perceptron updates its memory in the following way. When the classifier makes a prediction error for a new example, i.e., \( y_t \neq \hat{y}_t \) or the prediction confidence is not sufficient, i.e., \( l_t(f_t(x_t), y_t) := \max[0, \rho - y_t f_t(x_t)] > 0 \), where \( \rho \) is a small positive constant, the classifier adds the instance \( x_t \) to the support set, \( S_{t+1} = S_t \cup x_t \), and updates its hypothesis, \( f_{t+1} = f_t + y_t k(x_t, \cdot) \). If the prediction is correct, both the hypothesis and the support set are left intact, i.e., \( f_{t+1} = f_t \) and \( S_{t+1} = S_t \). Apparently, the support set \( S_t \) of kernel-based Perceptron grows unboundedly when the classification problem is inseparable or the function to be learned varies over time. This shortcoming limits the application of kernel-based Perceptron to problems where a life-long learning is necessary or the memory buffer of the learning machine is limited.

To attack this problem, different approaches have been proposed. For instance, Crammer, Kandola, and Singer (2003) suggested a budget method to restrict the size of the support set by discarding redundant instances according to certain rules. A similar strategy can also be found in NORMA (Kivinen et al., 2004) or SILK (Cheng, Vishwanathan, Schuurmans, Wang, & Gaelli, 2006). Dekel, Shalev-Shwartz, and Singer (2007) further introduced a shrinking...
idea in the budget approach, called Forgetron, and gave an upper bound for the classification error. Cesa-Bianchi, Conconi, and Gentile (2006) proposed a stochastic version for budget algorithms. Recently, Orabona, Keshet, and Caputo (2009) proposed a more advanced method called Projectron, which utilizes the idea of projection to keep the information of removed examples (Csató & Opper, 2000, 2002; Downs, Gates, & Masters, 2001; Engel, Mannor, & Meir, 2002, 2004; Wang, Crammer, & Vucetic, 2010). In Projectron, a newly arriving example is either added to the support set or discarded according to a predefined rule. However, for those discarded examples, Projectron does not throw them away cheaply, instead, it tries to retain the information of these examples by projecting their contributions to the discriminative function into the space spanned by the original support set. By this, Projectron tends to achieve two goals of learning the target function and in the mean time keeping a relatively small size of the support set.

Projectron works quite well in many situations, however, this method may have a shortcoming of storing too many redundant examples in the memory, due to that it lacks an explicit mechanism of 'forgetting'. This can be a serious problem when data to be learned is non-stationary, i.e., the target function changes over time. Therefore, in the present study, we propose a dynamical memory strategy to further control the size of the support set in online learning.

Our approach consists of two operations, namely, the incremental and decremental projections. In the phase of incremental projection, which is similar to the Projectron, the newly presented instance is either added to the support set or discarded by the projection technique. Whenever the support set is increased, the algorithm moves on to the phase of decremental projection. We evaluate the minimum information loss by deleting one example from the support set. If this minimum information loss is less than a tolerable threshold, then the corresponding example is removed from the support set by the projection technique. Otherwise, the support set is left intact. We also develop an approach which sets a hard constraint on the size of the support set. We test our methods to four benchmark data sets and find out that our approaches achieve a good balance between high classification accuracy and small support set size. For the same level of accuracy, the support set sizes achieved by our approaches are significantly smaller than that of Projectron. For comparable support set sizes, our methods outperform others with higher classification accuracies. Computationally, our methods are also very efficient and easy to be implemented.

2. The projection technique

An easy way to constrain memory size is to set a budget for the support set and throw away extra instances whenever the budget is broken. This approach is, however, not optimal for learning, since by throwing away instances cheaply, the valuable information coming with these data is also lost. A more efficient approach to remove instances is the projection technique (Csató & Opper, 2000, 2002; Downs et al., 2001; Engel et al., 2002, 2004; Orabona et al., 2009; Wang et al., 2010).

Let us consider first a simple case, when the functional space $\mathcal{H}$ of the classifier is a finite dimensional RKHS (e.g., induced by a polynomial kernel). In this case, there is only a finite number of linearly independent hypotheses in $\mathcal{H}$, and any hypothesis in $\mathcal{H}$ can be expressed using a finite number of instances. Thus, if a new arriving instance $x_t$ can be spanned by the current support set $S_t$, i.e., $k(x_t, \cdot) = \sum_{x_i \in S_t} d_i k(x_i, \cdot)$, then this instance does not have to be added to the support set. However, we can use a projection technique to learn the contribution of $x_t$ to the discriminative function, and update the hypothesis according to $\alpha_{t+1} = \alpha_t + y_t d_t, i \in \{j | x_j \in S_t\}$. By this, the information of $x_t$ is extracted by the classifier without the cost of increasing the support set size. If $x_t$ and $S_t$ are linearly independent in feature space, $x_t$ needs to be added to the support set, i.e., $S_{t+1} = S_t \cup x_t$, and the hypothesis is also updated accordingly. Apparently, since the dimensionality of $\mathcal{H}$ is finite, the support set size $|S|$ is bounded.

In cases when $\mathcal{H}$ is an infinite dimensional RKHS (e.g., induced by a Gaussian kernel), the projection technique cannot guarantee a bounded support size any more. However, in this case, we can approximate the concept of linear independence in a feature space with a finite support size (Csató & Opper, 2000; Engel et al., 2004; Orabona et al., 2009). Specifically, on updating a hypothesis, we consider two hypotheses, namely, a temporal one, $f_t^{(t+1)} = f_t + y_t k(x_t, \cdot)$ and a projected one, $f_t^{(t+1)} = P_t f_{t+1}^{(t+1)}$, where $P_t f_{t+1}$ denotes the projection of $f_{t+1}^{(t+1)} \in \mathcal{H}$ onto the subspace $\mathcal{H}_{t- \mathcal{H}}$ spanned by $S_t$, that is, we try to use the projected hypothesis to approximate the temporal one. Then we introduce a scalar $\eta > 0$ to bounded the discrimination $\mathcal{H}$.

The above method is called Projectron. Orabona et al. (2009) proved that the maximum support set size in Projectron is finite. In the below, we will develop approaches to more efficiently control the support set size.

3. Projection with dynamic memory

Our method consists of two operations, namely, the incremental and decremental projections. It tries to memorize new informative examples and forget old redundant ones. Compared with Projectron, our approach has an additional ‘forgetting’ mechanism, i.e., the decremental projection.

3.1. Incremental projection

The step of incremental projection is the same as that in Projectron. Recalling the definition of the approximation error, we have

$$
\| \delta_t \|^2 = \min_{d} \sum_{x_i \in S_t} d_i k(x_i, \cdot) - k(x_t, \cdot) \|^2.
$$

where $d \in \mathbb{R}^{|S_t|}$ is a vector of expanding coefficients. Solving it gives us

$$
\delta_t = K_t^{-1} k_t,
$$

where $K_{t,i,j} = k(x_t, x_i), K_{t,i,j} = k(x_t, x_i)$. $k_t = k(x_t, x_i)$ with $x_t \in S_t$. Then we can update the coefficients by $\alpha_{t+1}^i = \alpha_t^i + y_t d_i$

One choice to calculate $K_t^{-1}$ efficiently is to use the Sherman–Morrison–Woodbury formula (Cauwenberghs & Poggio, 2000; Orabona et al., 2009) and solve the problem iteratively with $K_t^{-1} = k_t^{-1}$, where we assume that $k_t^{-1} > 0$, and

$$
K_{t+1}^{-1} = \begin{bmatrix} K_t & k_t \n \k_t^T & k_t \end{bmatrix}^{-1} = \begin{bmatrix} K_t & k_t \\
0 & -1 \end{bmatrix}^{-1} = \begin{bmatrix} K_t^{-1} & 0 \n 0 & -1 \end{bmatrix}^{-1}. \end{bmatrix}
$$

3.2. Decremental projection

When a new instance is added to the support set, we move to the step of decremental projection. An instance is removed from the support set by using the projection technique if two criteria are satisfied: (1) deleting this instance will cause the minimum
information loss to the discriminative function (measured by the projection error); and (2) the information loss caused is less than a tolerable threshold.

The decremental projection works in the following way. First we consider a projected hypothesis \( f_{t+1}^{*} = P_{t+1-r} f_{t+1} \), where \( P_{t+1-r} f_{t+1} \) denotes the projection of \( f_{t+1} \) onto the subspace \( \mathcal{H}_{t+1-r} \), \( \mathcal{H} \) spanned by the set \( S_{t+1-r} = S_{t} \cup x_{i} \setminus x_{i} \) with \( x_{i} \) representing the instance to be removed. Let \( \| \delta_{t}^{i} \| = \| f_{t+1}^{*} - f_{t+1} \| \) denote the approximation error and \( \eta' > 0 \) a threshold, then the hypothesis is updated according to

\[
f_{t+1} = f_{t+1}^{*}, \quad S_{t+1} = S_{t} \cup x_{i}, \quad \text{if } \| \delta_{t}^{i} \| \leq \eta', (7)
\]


\[
f_{t+1} = f_{t+1}, \quad S_{t+1} = S_{t} \cup x_{i}, \quad \text{otherwise}.
\]

Eqs. (7) and (8) are similar to Eqs. (2) and (3), while the former can remove any instance in the support set which becomes uninformative.

Now the task is to determine index \( i^{*} \) of the instance to be removed. For clarity, we define \( \| \delta_{t}^{i} \| = \| f_{t+1} - f_{t+1}^{i} \| \), thus, \( \| \delta_{t}^{i} \| = \| \delta_{t}^{i} \| = \min_{j \in [k \leq 5 \leq 5]} \| \delta_{t}^{j} \| \). Expanding the hypothesis gives us

\[
\| \delta_{t}^{i} \|^{2} = \min_{\alpha_{t}^{i}} \sum_{x_{j} \in S_{t+1-i}} d_{j}^{i} k(x_{j}, \cdot) - f_{i} - y_{i} k(x_{i}, \cdot) \|^2
\]

\[
= (\alpha_{t}^{i})^{2} \min_{\alpha_{t}^{i}} \sum_{x_{j} \in S_{t+1-i}} d_{j}^{i} - \frac{\alpha_{t}^{i}}{\alpha_{t}^{j}} k(x_{j}, \cdot) + \frac{\alpha_{t}^{i}}{\alpha_{t}^{j}} y_{i} k(x_{i}, \cdot) - k(x_{i}, \cdot) \|^2
\]

\[
= \left( \alpha_{t}^{i} \right)^{2} \min_{\alpha_{t}^{i}} \sum_{x_{j} \in S_{t+1-i}} d_{j}^{i} k(x_{j}, \cdot) - k(x_{i}, \cdot) \|^2 + \frac{\alpha_{t}^{i}}{\alpha_{t}^{j}} y_{i} k(x_{i}, \cdot), (9)
\]

where \( d_{j}^{i} = (d_{j}^{i} - \alpha_{t}^{i}) / \alpha_{t}^{i} \) for \( x_{j} \in S_{t+1-i} \) and \( \alpha_{t}^{i} = y_{i} \).

\[
\| \delta_{t}^{i} \|^{2} = \min_{\alpha_{t}^{i}} \sum_{x_{j} \in S_{t+1-i}} d_{j}^{i} k(x_{j}, \cdot) - k(x_{i}, \cdot) \|^2
\]

Similar to Eq. (5) we have

\[
d_{j}^{i} = \mathbf{K}_{t+1-i}^{*} k_{j}, \quad \| \delta_{t}^{i} \|^{2} = k_{j} - \mathbf{K}_{t+1-i}^{*} d_{j}^{i},
\]

where \( \mathbf{K}_{t+1-i}^{*} \) denotes the kernel matrix \( \mathbf{K}_{t+1-i} \) with row \( i \) and column \( i \) removed. \( \mathbf{K}_{t+1-i} \) is already available in the incremental step. We remove the ith row of \( \mathbf{K}_{t+1-i} \) to the last row, the ith column to the last column, and let \( \mathbf{K}_{t+1-i}^{*} \) denote it. Then we can get \( \mathbf{K}_{t+1-i}^{*} \) just by removing row \( i \) and column \( i \) of \( \mathbf{K}_{t+1-i}^{*} \).

Applying the Sherman–Morrison–Woodbury formula once more gives us

\[
[\mathbf{K}_{t+1-i}^{*}]^{-1} = \left[ \mathbf{K}_{t+1-i}^{*} - \mathbf{Q}_{t+1-i} \mathbf{Q}_{t+1-i}^{*} \right]^{-1}
\]

\[
\mathbf{Q}_{t+1-i} = \left[ \mathbf{Q}_{t+1-i}^{*} \mathbf{Q}_{t+1-i}^{*} \right]
\]

With \( \mathbf{K}_{t+1-i}^{*} \) at hand, we can also write

\[
[\mathbf{K}_{t+1-i}^{*}]^{-1} = \left[ \mathbf{K}_{t+1-i}^{*} - \frac{\alpha_{t}^{i} e_{i} e_{i}^{T}}{\| \delta_{t}^{i} \|^{2}} \right]^{-1}
\]

\[
\alpha_{t}^{i} = -\frac{\mathbf{Q}_{t+1-i}^{T} \mathbf{Q}_{t+1-i}^{T}}{\| \mathbf{Q}_{t+1-i} \|^{2}}
\]

This gives us

\[
\| \delta_{t}^{i} \|^{2} = \frac{1}{\| \mathbf{Q}_{t+1-i} \|^{2}} + \frac{\| \mathbf{Q}_{t+1-i} \|^{2}}{\| \mathbf{Q}_{t+1-i} \|^{2}}\mathbf{K}_{t+1-i}^{*}
\]

\[
\| \mathbf{Q}_{t+1-i} \|^{2} = 1/\| \mathbf{Q}_{t+1-i} \|^{2}
\]

\[
\| \delta_{t}^{i} \|^{2} = \frac{1}{\| \mathbf{Q}_{t+1-i} \|^{2}}\mathbf{K}_{t+1-i}^{*} \mathbf{Q}_{t+1-i}^{T}
\]

\[
\| \mathbf{Q}_{t+1-i} \|^{2} = 1/\| \mathbf{Q}_{t+1-i} \|^{2}
\]

\[
\mathbf{K}_{t+1-i}^{*} = \mathbf{Q}_{t+1-i}^{T} \mathbf{Q}_{t+1-i}^{T}
\]

\[
\mathbf{Q}_{t+1-i}^{T} \mathbf{Q}_{t+1-i}^{T}
\]

Therefore, we have

\[
\| \delta_{t}^{i} \|^{2} = \frac{1}{\| \mathbf{Q}_{t+1-i} \|^{2}} + \frac{\| \mathbf{Q}_{t+1-i} \|^{2}}{\| \mathbf{Q}_{t+1-i} \|^{2}}\mathbf{K}_{t+1-i}^{*} \mathbf{Q}_{t+1-i}^{T}
\]

\[
\| \mathbf{Q}_{t+1-i} \|^{2} = 1/\| \mathbf{Q}_{t+1-i} \|^{2}
\]

\[
\mathbf{K}_{t+1-i}^{*} = \mathbf{Q}_{t+1-i}^{T} \mathbf{Q}_{t+1-i}^{T}
\]

\[
\mathbf{Q}_{t+1-i}^{T} \mathbf{Q}_{t+1-i}^{T}
\]

Table 1

| Projection with dynamical memory (PDM) |

| PDM |

| Initialize: \( S_{1} = \phi, f_{1} = 0 \) |

| for \( t = 1, 2, \ldots, T \) do |

| Receive instance \( x_{t} \); Predict \( y_{t} = \text{sign}(f_{t}(x_{t})) \); Receive outcome \( y_{t} \) |

| if \( y_{t} \neq y_{t} \) |

| Set \( f_{t+1} = f_{t} + y_{t} k(x_{t}, \cdot), f_{t+1} = P_{t+1-r} f_{t+1} \), \( \| \delta_{t+1} \| = \| f_{t+1} - f_{t+1} \| \) |

| else |

| \( f_{t+1} = f_{t}, S_{t+1} = S_{t} \) |

| end if |

| end for |


3.3 The algorithms

Putting the above two steps together, we get the algorithm realizing dynamic memory, called Projection with Dynamical Memory (PDM), Table 1 presents the code.

There are cases we may wish to have a fixed memory size, e.g., when the memory buffer of a learning machine is pre-determined. However, the exact size of the support set of PDM for a chosen tolerable threshold may still vary with different data sets. To this end, we also develop a modified version of PDM by setting a hard constraint on the maximum size of the support set as that in the budget algorithms, called PDM with a fixed budget (PDMB). The main difference of PDMB compared with PDM is that we remove an instance when the size violates the budget \( B \). Again, we choose the instance to be removed is the one having the minimum information loss, and retain its contribution to the discriminative function by the projection technique. The algorithm is summarized in Table 2.

One may concern that the computation of the matrix inverse \( \mathbf{K}_{t+1}^{-1} \) may cause the algorithms to be unstable. It turns out this is not the case. The incremental control ensures that the support set will not accept a new instance that can be expressed by others, therefore \( \mathbf{K} \) is always well-defined.

We note that the time consuming for the incremental projection is \( O(|S|^{2}) \), with \( |S| \) the size of the support set, and the space consuming for the storage of \( \mathbf{K}_{t+1}^{-1} \) is also \( O(|S|^{2}) \). The decremental control does not increase the computational cost significantly. The main time consuming task in the decremental step is to calculate the matrix inverse Eq. (13), which is already available in the incremental step.
Let $k$ be a kernel and $(\{\mathbf{x}_i, y_i\})_{i=1}^T$ a sequence of examples where $\mathbf{x}_i \in \mathcal{X}$, $y_i \in \{-1, +1\}$ and $k(\mathbf{x}_i, \mathbf{x}_j) \leq R^2$ for all $t$. Let $g$ be an arbitrary function in $\mathcal{H}$ so that $\|g\| < \frac{\eta}{\sqrt{T}}$, and define

$$D_\rho = \sum_{t=1}^T l_p(g(\mathbf{x}_t), y_t).$$

Then the number of prediction mistakes made by PDM with $\eta$, $\eta' > 0$ on the sequence is bounded by

$$M \leq \frac{D_\rho}{\rho - \eta\|g\|} + \frac{\|g\|^2R^2}{(\rho - \eta\|g\|)^2} + \frac{D_\rho}{\rho - \eta\|g\|},$$

where $M$ is the number of mistakes, $\eta = \max\{\eta, \eta'\}$.

**Proof.** On each round there are four possible hypothesis updating, namely, $f_{t+1} = f_t, f_{t+1} = f_t + \eta k(\mathbf{x}_t, \cdot)$, and $f_{t+1} = f_{t+1-r}f_{t+1-r}$. Let $\Delta_t := \|f_t - \lambda g\|^2 - \|f_{t+1} - \lambda g\|^2$ represent the relative progress on round $t$. Here $\lambda$ is a positive scalar to be optimized. Observe, $\min(\lambda g(\mathbf{x})) = \min(\lambda g(\mathbf{x})).$

For the first case, $\Delta_t = 0$ and we consider other three cases. Actually, in these cases, it always holds that

$$\Delta_t = \|f_t - \lambda g\|^2 - \|f_{t+1} - \lambda g\|^2 - \eta_t$$

$$= \|f_t - \lambda g\|^2 - \|f_{t+1} - \lambda g\|^2 + \|k(\mathbf{x}_t, \cdot)\|^2 - 2\eta_t$$

$$\geq 2\eta_t \|k(\mathbf{x}_t, \cdot)\|^2 - \|k(\mathbf{x}_t, \cdot)\|^2 - \eta_t$$

$$\geq 2\eta_t \|k(\mathbf{x}_t, \cdot)\|^2 - \|k(\mathbf{x}_t, \cdot)\|^2 - \eta_t$$

$$\geq 2\lambda (\rho - l_p(g(\mathbf{x}_t), y_t)) - \lambda g\|g\| - R^2 - 2\lambda \|g\|,$$

(25)

where we used Lemma 1 with $\eta_t = \|P_t f_{t+1} - f_{t+1}\|^2 + 2\|P_{t+1} f_{t+1} - f_{t+1}\|^2 + 2\|P_{t+1} f_{t+1} - f_{t+1}\|^2 + 2\|P_{t+1} f_{t+1} - f_{t+1} - \lambda g\|$. In the second case, $\Delta_t = \rho - \eta\|g\|$ and the above inequality still holds.

Summing over $t$ on both sides of Eq. (25) and using the fact that $f_1 = 0$ give us

$$\lambda^2 \|g\|^2 \geq \sum_{t=1}^T \Delta_t \geq 2\lambda (\rho - \eta\|g\|) - R^2$$

(26)

Tuning the value of $\lambda$ we can get the best bound for $M$.

$$M \leq \frac{(\|g\| + \sqrt{\|g\|^2R^2 + 4(\rho - \eta\|g\|)D_\rho})^2}{4(\rho - \eta\|g\|)^2}.$$  

(27)

Over-approximating Eq. (27) concludes the proof. □

In particular, for $\rho = 1, R = 1$ and $\eta' \leq \eta$, we get a bound similar to that of Projectron (Orabona et al., 2009). This suggests that under the same relative mistake bound, PDM tends to give us a smaller support set compared with that of Projectron. We will test this by practical examples.

For the behaviors of PDDB, when its support set size is smaller than the budget $B$, PDDB performs like Projectron. When the support set size is larger than $B$, PDDB like PDM. We can set
We then apply PDM to other three data sets. To ensure that our results will not be affected by the special order of original data sets, we randomly permute the data sets 10 times to generate 10 training sets, and the performances of classifiers are measured by the averaged results (mean ± std), as shown in Table 4. We observe that compared with Perceptron and Projectron, PDM can either have a smaller size of S when the classification accuracy is comparable or have a higher classification accuracy when |S| is similar. To see these results more clearly, we present the online behaviors of the algorithms over time, as shown in Figs. 1 and 2. Fig. 1 shows that when the compared algorithms have similar classification accuracies (Fig. 1A), |S| of PDM is much smaller than that of others (Fig. 1B). Fig. 2 shows a case that when PDM and Projectron have comparable support set sizes, the classification accuracy of PDM is higher than that of Projectron.

5.2. Performances of PDMB

Now we test PDMB. We compare its performance to other two budget-based algorithms, namely, LRU and Forgetron. The results are summarized in Table 5, where two budget sizes are used and \( \eta = 0.1 \) in PDMB. We see that PDMB outperforms LRU and Forgetron significantly. Fig. 3 presents the online behaviors of three algorithms. Interestingly to note that when the support set sizes of LRU and Forgetron reach the budget constraints, their performances can no longer be improved. On the other hand, the performance of PDMB keeps improving due to arrival of new examples. This is because PDMB selectively removes examples from the memory and utilizes the projection idea to diminish information loss.

5.3. Performances of PDM for non-stationary data

A key advantage of PDM and PDMB is that they can control |S| efficiently when data is non-stationary due to the forgetting mechanism. To test this idea, we apply PDMB to non-stationary data sets (PDM has the similar performance and its result is not shown here). The non-stationary data sets are generated by manipulating the mean and the variance of Gaussian distribution in the Synthetic data set used above. Two operations are applied, namely, drifting and switching (Kivinen et al., 2004). In the drifting case, we add a Gaussian perturbation \( \mathcal{N}(0.2, 0.4; 0.1, 0.2; 0) \), where the first two parameters are the means of the distribution, the following two parameters are the variances and the last one is the correlation coefficient, to the instances of the Synthetic data in random in chosen direction every 5 steps. This models data changes fast and mildly. In the switching case, we add a Gaussian

| Table 3 |
|---|---|---|---|
| **Performance of PDM in the vehicle data set.** |
| Algorithm | \( \eta \) | \( \eta' \) | \( (M/T) \% \) | |\( S \) |
| Perceptron | – | – | 19.80 | 15610 |
| Projectron | 0.1 | – | 19.77 | 3110 |
| | 0.35 | – | 20.34 | 420 |
| PDM | 0.1 | 0.1 | 19.70 | 1470 |
| | 0.1 | 0.15 | 19.89 | 726 |
| | 0.1 | 0.25 | 20.09 | 310 |
| | 0.2 | 0.2 | 20.08 | 430 |

| Table 4 |
|---|---|---|---|
| **Performance of PDM in three data sets.** |
| Algorithm | \( \eta \) | \( \eta' \) | \( (M/T) \% \) | |\( S \) |
| Perceptron | – | – | 2071.4 | 34.0 |
| | 0.1 | – | 19.70 | 46.1 |
| | 0.15 | – | 19.02 | 40.1 |
| | 0.25 | – | 17.36 | 27.7 |
| | 0.35 | – | 15.91 | 19.2 |
| | 0.5 | – | 13.89 | 20.1 |
| | 0.7 | – | 11.28 | 21.3 |
| | 0.9 | – | 8.37 | 23.4 |
| PDM | 0.1 | 0.1 | 1899.39 | 5.1 |
| | 0.25 | 0.25 | 1677.49 | 4.4 |
| | 0.35 | 0.35 | 1542.27 | 5.5 |
| | 0.45 | 0.45 | 1379.28 | 4.2 |
| | 0.5 | 0.5 | 1220.20 | 9.9 |
| | 0.55 | 0.55 | 1151.16 | 5.6 |
| | 0.75 | 0.75 | 793.40 | 3.0 |

\( \eta' = \| \delta_t^\eta \| \) and let \( \eta' = \max_{S>1} |\delta_{t+1}^S| > \| |\delta_t^\eta| \| \), then the bound (Eq. (24)) also holds for PDDB, where \( \eta = \max(\| \delta_{t+1}^S \|, |\delta_t^\eta|) \). Obviously, \( \eta' \leq R \) always holds due to the fact that \( 0 \leq \| \delta_t \|^2, \| \delta_t^\eta \|^2 \leq k(x_1, x_2) \leq R^2 \). Hence we can set \( 0 < \eta, \eta' < R \) for Theorem 2. Theoretically, any budget size \( B > 0 \) can be used. While setting it to be reasonably large is required to keep the performance under control.

5. Experimental analysis

In this section we compare our methods to several pertinent algorithms in the field, which are (1) Perceptron, an algorithm without memory control; (2) LRU (last recently used), which discards the oldest instance in the support set whenever the support set size reaches the budget; (3) Forgetron, a budget control algorithm with the shrinking idea (we used the “self-tuned” version); and (4) Projectron, which utilizes the projection idea to diminish information loss but lacks a forgetting mechanism.

Four data sets are used, namely, Vehicle, Ijcnn1, Adults9 (the three data sets are all downloaded from http://www.sie.ntu.edu.tw/~cljnilabs/miletools) and a synthetic data set used in Orabona, Keshtet, and Caputo (2008). The Gaussian kernel is used. For a fair comparison, we set the width of the Gaussian kernel \( \sigma^2 \) to be 4, 0.0625, 25 and 0.5, respectively, in four data sets, so that Perceptron has its best performance.

5.1. Performances of PDM

We first apply PDM to a large data set, Vehicle, which has 78823 examples. The results are summarized in Table 3, where \( M/T \% \) represents the error rate and \( |S| \) the size of the support set. We see that, compared with Perceptron and Projectron, PDM can achieve similar classification accuracies but with a much smaller size of the support set.

We then apply PDM to other three data sets. To ensure that our results will not be affected by the special order of original data sets, we randomly permute the data sets 10 times to generate 10 training sets, and the performances of classifiers are measured by the averaged results (mean ± std), as shown in Table 4. We observe that compared with Perceptron and Projectron, PDM can either have a smaller size of \( S \) when the classification accuracy is comparable or have a higher classification accuracy when \( |S| \) is similar. To see these results more clearly, we present the online behaviors of the algorithms over time, as shown in Figs. 1 and 2. Fig. 1 shows that when the compared algorithms have similar classification accuracies (Fig. 1A), \( |S| \) of PDM is much smaller than that of others (Fig. 1B). Fig. 2 shows a case that when PDM and Projectron have comparable support set sizes, the classification accuracy of PDM is higher than that of Projectron.
perturbation, $N(3, 6; 1, 3; 0)$, to the instances of the Synthetic data in a random chosen direction every 500 steps. This models data changes slow but sharply. We set $\eta = 0.1$ for Projectron and PDMB. The results are presented in Table 6 and Figs. 4 and 5. In the drifting case, PDMB has a higher accuracy than the budget-based algorithms, LRU and Forgetron (Table 6). Compared with Perceptron and Projectron, PDMB has a similar accuracy with a smaller support set size (Fig. 4). In the switching case, PDMB also outperforms others. Note that the support set size of Projectron becomes very large for a non-stationary data set (Fig. 5).

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Ijcnn1</th>
<th>Adults9</th>
<th>Synthetic</th>
</tr>
</thead>
<tbody>
<tr>
<td>LRU</td>
<td>5.00±0.31</td>
<td>1500</td>
<td>22.08±0.18</td>
</tr>
<tr>
<td>Forgetron</td>
<td>5.23±0.39</td>
<td>1500</td>
<td>22.01±0.14</td>
</tr>
<tr>
<td>PDMB</td>
<td>4.16±0.10</td>
<td>1500</td>
<td>20.96±0.11</td>
</tr>
<tr>
<td>LRU</td>
<td>6.20±0.67</td>
<td>1000</td>
<td>23.48±0.15</td>
</tr>
<tr>
<td>Forgetron</td>
<td>6.40±0.06</td>
<td>1000</td>
<td>23.09±0.17</td>
</tr>
<tr>
<td>PDMB</td>
<td>4.34±0.16</td>
<td>1000</td>
<td>20.99±0.08</td>
</tr>
</tbody>
</table>
Fig. 4. Online performances of the budget algorithms and PDMB in the drifting case. $B = 100$, $\eta = 0.1$. (A) Classification accuracy vs. number of samples presented; (B) Size of support set vs. number of samples presented.

Fig. 5. Online performances of the budget algorithms and PDMB in the switching case. $B = 100$, $\eta = 0.1$. (A) Classification accuracy vs. number of samples presented; (B) Size of support set vs. number of samples presented.

Table 6

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Drifting</th>
<th>Switching</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$(M/T)$ (%)</td>
<td>$(M/T)$ (%)</td>
</tr>
<tr>
<td>Perceptron</td>
<td>19.77±0.20</td>
<td>1977±0.19</td>
</tr>
<tr>
<td>Projecton</td>
<td>19.77±0.20</td>
<td>137±2.8</td>
</tr>
<tr>
<td>PDMB</td>
<td>19.64±0.16</td>
<td>100</td>
</tr>
<tr>
<td>LRU</td>
<td>21.28±0.29</td>
<td>100</td>
</tr>
<tr>
<td>Forgetron</td>
<td>21.26±0.23</td>
<td>100</td>
</tr>
</tbody>
</table>

6. Conclusions

In the present study we have proposed a dynamical memory idea for kernel-based online learning. This strategy is realized by performing two operations, namely, incremental and decremental projections. Two algorithms, PDM and PDMB, which implement the idea of dynamical memory, are introduced. We theoretically derived the mistake bounds of the two methods. Simulation experiments with real data show that PDM and PDMB outperform other memory-control strategies in either having smaller sizes of the support sets when the classification accuracies are comparable or having higher classification accuracies when the sizes of the support sets are similar. In particular, our methods work very well when data is non-stationary. In our future work, we will extend the dynamic memory idea to more general cases.

Acknowledgments

We thank the two anonymous reviewers for their valuable comments and suggestions. W.H. thanks Francesco Orabona for his unselfish help in understanding the Projectron algorithm and valuable comments to the paper. S.W is supported by the 973 program of China (grant number 2011CB300406). We acknowledge the support from the open project program of the State Key Laboratory of Neuroscience in Shanghai, China.

References


