ONLINE KERNEL-BASED CLASSIFICATION BY PROJECTIONS

Konstantinos Slavakis, Sergios Theodoridis

University of Athens,
Department of Informatics
and Telecommunications,
Ilissia, Athens 15784, Greece.
Emails: {slavakis,stheodor}@di.uoa.gr

Isao Yamada

Tokyo Institute of Technology,
Department of Communications
and Integrated Systems,
S3-60, Tokyo 152-8552, Japan.
Email: isao@comm.ss.titech.ac.jp

ABSTRACT

The goal of this paper is the development of a novel efficient *online* kernel-based algorithm for classification. The spirit of the algorithm stems from the recently introduced Adaptive Projected Subgradient Method. This is a general convex analytic tool that employs projections onto a sequence of convex sets and it can be considered as a generalization of the celebrated APA algorithm, widely used in classical adaptive filtering.

Keywords: Pattern classification, Adaptive systems.

1. INTRODUCTION

Kernel-based methods have been showing significant success in modern pattern analysis [1, 2]. They have been applied mostly to batch schemes, i.e., cases where all the training data are available beforehand (e.g., support vector machines). Although hybrid batch methods can be derived by using, for example, a sliding buffer, genuine online kernel-based learning algorithms for real-time applications are mostly desirable [3].

An efficient online kernel-based learning algorithm was recently introduced in [3] that builds upon stochastic gradient descent concepts. The algorithm in [3] surmounts the obstacles often appearing in real-time settings, generalizes the kernel perceptron method, and when applied to online classification problems exhibits low computational load and misclassification errors.

The present paper reformulates the general kernel-based classification problem as the problem of finding a point that belongs to a halfspace; a closed convex subset of a Hilbert space. In a real-time setting, new training data arrive at every time instant and online classification becomes the problem of finding a point that belongs to the *intersection* of a sequence of halfspaces. An algorithmic solution is given by the very recently introduced Adaptive Projected Subgradient Method (APSM) [4, 5]. This is a general convex analytic tool, which for the problem at hand, is realized by taking *simple projections* onto the sequence of the halfspaces. The resulting online classification algorithm exhibits a computational load of the same order as that of the method in [3] and the classical kernel perceptron algorithm but with a distinctly superior performance regarding convergence.

2. MATHEMATICAL PRELIMINARIES

We will denote the set of all integers, nonnegative integers, positive integers, and real numbers by $\mathbb{Z}$, $\mathbb{Z}_{\geq 0}$, $\mathbb{Z}_{>0}$, and $\mathbb{R}$ respectively.

The main stage of our discussion will be a real Hilbert space $\mathcal{H}$, equipped with an inner product denoted by $\langle f_1, f_2\rangle$, $\forall f_1, f_2 \in \mathcal{H}$ [6]. The induced norm will be denoted by $\|f\| := \langle f, f\rangle^{1/2}$, $\forall f \in \mathcal{H}$.

In modern pattern analysis [1, 2] a classification task of a set $\mathcal{X}$ of vectors in the data space $\mathbb{R}^m$, $m \in \mathbb{Z}_{>0}$, is usually reformed by mapping the data $\mathcal{X}$ into a higher dimensional space $\mathcal{H}$, which is a Reproducing Kernel Hilbert Space (RKHS) [1, 2, 7]. This RKHS $\mathcal{H}$, which is called feature space in the context of pattern analysis, is often of very high or even infinite dimension. The advantage of such a mapping is to make the task more tractable, by employing a linear classifier in the feature space, exploiting Cover's theorem [1]. At the heart of such a mapping lies a kernel function κ. Given an RKHS $\mathcal{H}$, the associated kernel function $\kappa : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ [1, 2, 7] defines the mapping from $\mathbb{R}^m$ to $\mathcal{H}$, i.e., $\phi : \mathbb{R}^m \to \mathcal{H} : x \mapsto \phi(x) := \kappa(x, \cdot)$. In this way the data $\mathcal{X} \subset \mathbb{R}^m$ are mapped to $\phi(\mathcal{X}) \subset \mathcal{H}$. We stress here that any point f of $\mathcal{H}$, such as $\kappa(x, \cdot)$ for any $x \in \mathbb{R}^m$, is a function from $\mathbb{R}^m$ into $\mathbb{R}$. That is, $\mathcal{H}$ is basically a Hilbert space of functions and is given by the closure of the linear span of $\{\kappa(x, \cdot) : x \in \mathbb{R}^m\}$ [2, 7]. Moreover, the following reproducing property is satisfied in an RKHS $\mathcal{H}$ [2, 7]:

$$\langle f, \kappa(x, \cdot)\rangle = f(x), \quad \forall f \in \mathcal{H}, \forall x \in \mathbb{R}^m. \tag{1}$$

This implies in particular $\|\kappa(x, \cdot)\|^2 = \langle \kappa(x, \cdot), \kappa(x, \cdot)\rangle = \kappa(x, x)$.

There are numerous kernel functions and associated RKHS $\mathcal{H}$ with rich applications in pattern analysis [1, 2]. Examples of kernel functions are i) the linear kernel $\kappa(x, y) := x^t y, \forall x, y \in \mathbb{R}^m$, where the superscript $(\cdot)^t$ denotes transposition (in this case the RKHS $\mathcal{H}$ is the data space $\mathbb{R}^m$ itself), ii) the Gaussian kernel $\kappa(x, y) := \exp(-\frac{(x-y)^t(x-y)}{2\sigma^2})$, $\forall x, y \in \mathbb{R}^m$, where $\sigma > 0$, and iii) the polynomial kernel $\kappa(x, y) := (x^t y + 1)^d$, $\forall x, y \in \mathbb{R}^m$, where $d \in \mathbb{Z}_{>0}$.

A subset C of $\mathcal{H}$ will be called convex if $\forall f_1, f_2 \in C$, the point $\lambda f_1 + (1-\lambda) f_2 \in C, \forall \lambda \in (0, 1)$. Let us give an example of a closed convex set in a real Hilbert space. Given $a \neq 0$ of $\mathcal{H}$ and $\xi \in \mathbb{R}$, let a halfspace be the closed convex set $\Pi := \{h \in \mathcal{H} : \langle a, h\rangle \geq \xi\}$. A function $\Theta : \mathcal{H} \to \mathbb{R}$ will be called convex if $\forall f_1, f_2 \in \mathcal{H}$, and $\forall \lambda \in (0, 1)$, we have $\Theta(\lambda f_1 + (1-\lambda) f_2) \leq \lambda\Theta(f_1) + (1-\lambda)\Theta(f_2)$.

Given any point $f \in \mathcal{H}$, we can quantify its distance from a closed convex set C by the function

$$d(f, C) := \inf\{\|f - h\| : h \in C\}, \quad \forall f \in \mathcal{H}. \tag{2}$$

The function $d(\cdot, C)$ is nonnegative, continuous, and convex [8]. Note that any point $h \in C$ is of zero distance from C, i.e., $d(h, C) = 0$. Thus, the set of all minimizers of $d(\cdot, C)$ over $\mathcal{H}$ is C itself.

Given an element f and a closed convex set C of a Hilbert space $\mathcal{H}$, an optimal way to move from f to a point in C, i.e., to a minimizer of $d(\cdot, C)$, is by means of the metric projection mapping P_C onto C, which is defined as the mapping that takes f to the uniquely existing point $P_C(f)$ of C that achieves the infimum value in (2): $\|f - P_C(f)\| = d(f, C)$ [6]. Clearly, if $f \in C$ then $P_C(f) = f$.

For a halfspace Π in a real Hilbert space, the metric projection operator P_Π has a closed form expression, i.e., to move from f to a point in Π only one step is needed [4]:

$$P_\Pi(f) = f + \frac{(\xi - \langle a, h\rangle)^+}{\|a\|^2} a, \forall f \in \mathcal{H}, \tag{3}$$

where $\tau^+ := \max\{0, \tau\}$ stands for the positive part of $\tau \in \mathbb{R}$. The metric projection mapping P_Π is the most efficient way to find a point in the halfspace Π. Moreover, since it belongs to the wide class of metric projection mappings onto closed convex sets, it enjoys several remarkable properties that can lead to very effective iterative solutions (see [4, 5] and the references therein). Note also that (3) is basically a generalization of projecting a point onto a hyperplane in a Euclidean space.

3. KERNEL-BASED CLASSIFICATION REVISITED

Consider the data space $\mathbb{R}^m$ and a set $\mathcal{X} := \{x_1, x_2, \ldots\} \subset \mathbb{R}^m$ which consists of vectors drawn from two classes. Each vector, x_n, in $\mathcal{X}$ is associated to a label $y_n \in \mathcal{Y} := \{\pm 1\}$, depending on the respective class. In this way a set of pairs is formed $\mathcal{S} := \{(x_1, y_1), (x_2, y_2), \ldots\} \subset \mathcal{X} \times \mathcal{Y}$. Let also a kernel κ and $\mathcal{H}$ the associated RKHS.

Given a margin $\rho \geq 0$, the (binary) classification problem is defined as selecting a function $f \in \mathcal{H}$ and an offset $b \in \mathbb{R}$ such that $y(f(x) + b) \geq \rho, \forall (x, y) \in \mathcal{S}$ [1, 2]. For convenience we can merge the unknown quantities $f \in \mathcal{H}$ and $b \in \mathbb{R}$ as a single vector $u := (f, b) \in \mathcal{H} \times \mathbb{R}$. Henceforth, we will call the point $u \in \mathcal{H} \times \mathbb{R}$ a classifier, and $\mathcal{H} \times \mathbb{R}$ the space of all classifiers.

The space $\mathcal{H} \times \mathbb{R}$ of all classifiers can be endowed with an inner product as follows: for any $u_1 := (f_1, b_1), u_2 := (f_2, b_2) \in \mathcal{H} \times \mathbb{R}$, let $\langle u_1, u_2\rangle := \langle f_1, f_2\rangle + b_1 b_2$. Thus, the space $\mathcal{H} \times \mathbb{R}$ of all classifiers becomes a Hilbert space.

A standard penalty function used for classification problems is the *soft margin loss function* [3, 1] defined on $\mathcal{H} \times \mathbb{R}$ as follows: given $(x, y) \in \mathcal{S}$ and the margin parameter $\rho \geq 0$,

$$l_\rho^{x,y}(u) : (f, b) \mapsto (\rho - y(f(x) + b))^+ = (\rho - yg(x))^+, \tag{4}$$

where $g(\cdot) := f(\cdot) + b$. By the above definition and given $(x, y) \in \mathcal{S}$, if the classifier $u = (f, b)$ is such that $yg(x) < \rho$, then this classifier fails to achieve the margin ρ and (4) scores a penalty. In such a case we say that the classifier committed a margin error. The boundary hypersurface of the data space $\mathbb{R}^m$ that separates the two classes is given by $\{x \in \mathbb{R}^m : g(x) = 0\}$. A misclassification occurs if there exists an $(x, y) \in \mathcal{S}$ such that $yg(x) < 0$.

It is natural therefore to look for those classifiers that minimize the soft margin error function. A stochastic gradient descent algorithmic solution towards this direction is presented in [3]. The scheme not only generalizes the kernel perceptron method but also offers an efficient *truly online* learning method as opposed to hybrid batch algorithms [2].

Here we will approach the classification task from a slight different perspective. Our goal is to seek for classifiers or points in $\mathcal{H} \times \mathbb{R}$ that belong to the set $\Pi_\rho^{x,y} := \{u \in \mathcal{H} \times \mathbb{R} : y(f(x) + b) \geq \rho\}$. If we recall the reproducing property (1), a desirable classifier, i.e., a classifier that belongs to $\Pi_\rho^{x,y}$, would satisfy $\langle f, y\kappa(x, \cdot)\rangle + by \geq \rho$. By the definition of the inner product in $\mathcal{H} \times \mathbb{R}$, we clearly have that the set of all *desirable* classifiers (that do not make a margin error) is

$$\Pi_\rho^{x,y} = \{u \in \mathcal{H} \times \mathbb{R} : \langle u, v_{x,y}\rangle \geq \rho\}, \tag{5}$$

where $v_{x,y} := (y\kappa(x, \cdot), y) = y(\kappa(x, \cdot), 1) \in \mathcal{H} \times \mathbb{R}$. Moreover, by (4) the set of all minimizers of the soft margin loss function is $\arg\min\{l_\rho^{x,y}(u) : u \in \mathcal{H} \times \mathbb{R}\} = \Pi_\rho^{x,y}$.

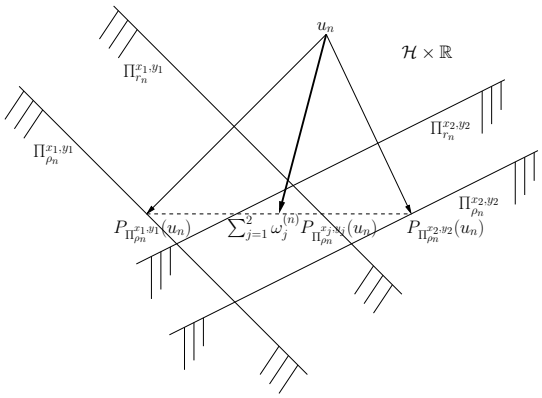

Fig. 1. Illustration of the projection onto a halfspace. The concurrent processing in (7) averages the projections onto the halfspaces. By projecting onto $\Pi_\rho^{x,y}$ instead of $\Pi_r^{x,y}$, where $\rho = \gamma r$ ($\gamma \geq 1$), we move deeper into $\Pi_r^{x,y}$. This and concurrent processing take us faster to the intersection $\bigcap_{j \in \mathcal{J}_n} \Pi_{r_n}^{x_j,y_j}$.

Notice that $\Pi_\rho^{x,y}$ is a halfspace of the Hilbert space $\mathcal{H} \times \mathbb{R}$. Given a point $u \in \mathcal{H} \times \mathbb{R}$, the most efficient way to move to a desirable classifier in $\Pi_\rho^{x,y}$ is by means of the metric projection mapping $P_{\Pi_\rho^{x,y}}$ as follows: by (3) and the fact that $\|v_{x,y}\|^2 = 1 + \kappa(x, x)$ we obtain

$$P_{\Pi_\rho^{x,y}}(u) = u + y\frac{(\rho - yg(x))^+}{1 + \kappa(x, x)}(\kappa(x, \cdot), 1). \tag{6}$$

Recall that $d(u, \Pi_\rho^{x,y}) = \left\| u - P_{\Pi_\rho^{x,y}}(u) \right\|$. As such,

$$d(u, \Pi_\rho^{x,y}) = \frac{(\rho - yg(x))^+}{\sqrt{1 + \kappa(x, x)}} = \frac{l_\rho^{x,y}(u)}{\sqrt{1 + \kappa(x, x)}},$$

which suggests that the problem of minimizing the soft margin loss function $l_\rho^{x,y}(\cdot)$ is equivalent to minimizing $d(\cdot, \Pi_\rho^{x,y})$. Recall now (3) to see that a minimizer of $d(\cdot, \Pi_\rho^{x,y})$ can be found by just a *single step*. In this way a faster learning method than the stochastic gradient descent approach in [3] is anticipated.

4. ONLINE CLASSIFICATION BY THE ADAPTIVE PROJECTED SUBGRADIENT METHOD

Assume now a sequence of pairs $(x_n, y_n)_{n \in \mathbb{Z}_{\geq 0}} \subset \mathcal{X} \times \mathcal{Y}$ denoting the sequence of incoming data from two classes in $\mathbb{R}^m$ together with their associated labels. Consider also a sequence of nonnegative margin parameters $(\rho_n)_{n \in \mathbb{Z}_{\geq 0}}$. According to the discussion of the previous section, each pair (x_n, y_n) and margin ρ_n define the halfspace $\Pi_n := \Pi_{\rho_n}^{x_n,y_n} := \{u = (f, b) \in \mathcal{H} \times \mathbb{R} : y_n(f(x_n) + b) \geq \rho_n\}$ or in other words the set of all those classifiers in $\mathcal{H} \times \mathbb{R}$ that achieve ρ_n for given (x_n, y_n) (see Fig. 1). Since we deal with a sequence of halfspaces $(\Pi_n)_{n \in \mathbb{Z}_{\geq 0}}$, our objective is to find a classifier $u_* := (f_*, b_*) \in \mathcal{H} \times \mathbb{R}$ that belongs to $\bigcap_{n \geq n_0} \Pi_n$ for, let's say, some $n_0 \in \mathbb{Z}_{\geq 0}$, provided of course that the set of all such classifiers $\bigcap_{n \geq n_0} \Pi_n$ is nonempty.

To tackle such problems, the Adaptive Projected Subgradient Method (APSM) was very recently introduced in [4, 5]. The algorithm solves the problem of asymptotically minimizing a sequence of nonnegative, continuous, convex but not necessarily differentiable functions over a closed convex set in a Hilbert space. The closed convex set was originally handled as the fixed point set of a metric projection in [5]. This idea was very recently extended in [4] by handling the closed convex set as the fixed point set of a more general nonexpansive mapping. By this generalization, we can solve the

problem defined, for example, over the intersection of fixed point sets of multiple metric projections. To see the connection of asymptotic minimization with the present classification problem, recall that finding a classifier in Π_n is equivalent to minimizing $d(\cdot, \Pi_n)$ which is a nonnegative, continuous, and convex function [8]. Although $d(\cdot, \Pi_n)$ is not differentiable, it is everywhere subdifferentiable [8, 4] and its subderivatives [8] are given by means of the metric projection operator P_{Π_n}. In adaptive filtering, APSM generalizes the classical Normalized Least Mean Squares (NLMS) and the celebrated Affine Projection Algorithm (APA) [9, 4, 5]. The APSM enjoys several remarkable theoretical properties and has been showing low computational cost, robustness, and fast convergence for several adaptive filtering problems [4].

The APSM allows concurrent processing of multiple data for every time instant. That is, instead of projecting each time onto a single halfspace, one can choose to project concurrently onto a number of halfspaces. This speeds up convergence, and it is basically the same idea behind APA [4, 5]. To do so, we assume for every $n \in \mathbb{Z}_{\geq 0}$ an index set $\mathcal{J}_n \subset \mathbb{Z}_{\geq 0}$, of finite cardinality $\mathrm{card}(\mathcal{J}_n)$, which determines the incoming data $\{(x_j, y_j)\}_{j\in\mathcal{J}_n}$ to be processed at time n. Since each index $j \in \mathcal{J}_n$ associates to a halfspace and to explicitly show the dependence of the halfspaces on the index set $\mathcal{J}_n$, we introduce a new notation for Π_n by $\Pi_j^{(n)}$ for any $j \in \mathcal{J}_n$ and for any $n \in \mathbb{Z}_{\geq 0}$. We can also assign weights to each one of the halfspaces by associating to each $j \in \mathcal{J}_n$ an $\omega_j^{(n)}$ such that $\omega_j^{(n)} > 0, \forall j \in \mathcal{J}_n$, and $\sum_{j\in\mathcal{J}_n} \omega_j^{(n)} = 1$, that may further improve performance.

By following similar arguments to those in [4, 5], we generate the following algorithmic solution to the online classification problem: for an arbitrary initial offset $b_0 \in \mathbb{R}$, consider as an initial classifier the point $u_0 := (0, b_0)$ and generate the point sequence $\forall n \in \mathbb{Z}_{\geq 0}$ in $\mathcal{H} \times \mathbb{R}$ by

$$u_{n+1} := u_n + \mu_n \left(\sum_{j\in\mathcal{J}_n} \omega_j^{(n)} P_{\Pi_j^{(n)}}(u_n) - u_n \right). \qquad (7)$$

That is, given the point u_n we employ a number of $\mathrm{card}(\mathcal{J}_n)$ metric projection mappings in order to obtain the next estimate u_{n+1}. The summation term in (7) denotes the concurrent processing of the data. Recall also that the positive weights $\{\omega_j^{(n)}\}_{j\in\mathcal{J}_n}$ add to one and note that the summation term in (7) is nothing but an average of the projections onto the halfspaces $\{\Pi_j^{(n)}\}_{j\in\mathcal{J}_n}$ as can be seen also for example in Fig. 1.

By (6), the algorithmic process (7) can be written equivalently as follows; $\forall n \in \mathbb{Z}_{\geq 0}$,

$$(f_{n+1}, b_{n+1}) = (f_n, b_n) + \mu_n \sum_{j\in\mathcal{J}_n} \omega_j^{(n)} y_j \frac{(\rho_j^{(n)} - y_j g_n(x_j))^+}{1 + \kappa(x_j, x_j)} (\kappa(x_j, \cdot), 1).$$

It can be shown that the relaxation coefficient $\mu_n \in [0, 2\mathcal{M}_n]$ where

$$\mathcal{M}_n := \begin{cases} \frac{\sum_{j\in\mathcal{J}_n} \omega_j^{(n)} \|P_{\Pi_j^{(n)}}(u_n) - u_n\|^2}{\|\sum_{j\in\mathcal{J}_n} \omega_j^{(n)} P_{\Pi_j^{(n)}}(u_n) - u_n\|^2} \\ \quad = \frac{\sum_{j\in\mathcal{J}_n} \omega_j^{(n)} [(\rho_j^{(n)} - y_j g_n(x_j))^+]^2/(1+\kappa(x_j,x_j))}{\sum_{i,j\in\mathcal{J}_n} \beta_i^{(n)} \beta_j^{(n)} (1+\kappa(x_i,x_j))}, \\ \qquad \text{if } u_n \notin \bigcap_{j\in\mathcal{J}_n} \Pi_j^{(n)}, \\ 1, \qquad \text{otherwise}, \end{cases} \qquad (8)$$

and

$$\beta_j^{(n)} := \omega_j^{(n)} y_j \frac{(\rho_j^{(n)} - y_j g_n(x_j))^+}{1 + \kappa(x_j, x_j)}, \ \forall j \in \mathcal{J}_n, \ \forall n \in \mathbb{Z}_{\geq 0}. \quad (9)$$

Notice by the convexity of $\|\cdot\|^2$ and by the first fraction in (8) that $\mathcal{M}_n \geq 1, \forall n \in \mathbb{Z}_{\geq 0}$, so that the relaxation parameter μ_n can take values bigger than or equal to 2. The update rules can be summarized as follows; if we define $\alpha_j^{(n)} := \mu_n \beta_j^{(n)}, \forall j \in \mathcal{J}_n, \forall n \in \mathbb{Z}_{\geq 0}$, and if we recall that $f_0 := 0$, then $\forall n \in \mathbb{Z}_{>0}$,

$$(f_n, b_n) = (\sum_{k=0}^{n-1} \sum_{j\in\mathcal{J}_k} \alpha_j^{(k)} \kappa(x_j, \cdot), b_0 + \sum_{k=0}^{n-1} \sum_{j\in\mathcal{J}_k} \alpha_j^{(k)}). \quad (10)$$

Note that the sequence of margins $(\rho_j^{(n)})$, $j \in \mathcal{J}_n$, $n \in \mathbb{Z}_{\geq 0}$, is treated as a sequence of parameters. An adaptive scheme for their selection is given in the next section.

We stress also that due to lack of space no arguments are demonstrated regarding the sparsification of the functional representation in (10). However, any sparsification method like [10] can be used for the proposed algorithm. The details are left for a future work.

5. AN ADAPTIVE SELECTION SCHEME FOR THE MARGIN PARAMETER

The selection schemes for the margin parameters are designer dependent. Here we give an example.

Assume a margin $r \geq 0$ and consider by (5) the set $\Pi_r^{x,y}$ of all those classifiers that achieve r for some $(x, y) \in \mathcal{X} \times \mathcal{Y}$. Notice that for any $\gamma \geq 1$ and for a margin $\rho := \gamma r$ we have $\Pi_\rho^{x,y} \subset \Pi_r^{x,y}$. This means that if we project onto $\Pi_\rho^{x,y}$ by the mapping $P_{\Pi_\rho^{x,y}}$ not only we obtain a classifier of $\Pi_r^{x,y}$ but we also move deeper into $\Pi_r^{x,y}$ (see Fig. 1).

Consider for simplicity that $\rho_j^{(n)} := \rho_n, \forall j \in \mathcal{J}_n, \forall n \in \mathbb{Z}_{\geq 0}$, i.e., all the halfspaces processed at time n assume the same margin $\rho_n \geq 0$. If $r_n \geq 0$ is a margin we want to achieve, by setting $\rho_n := \gamma r_n$ for some $\gamma \geq 1$ and by the argument just stated above, the average mapping $\sum_{j\in\mathcal{J}_n} \omega_j^{(n)} P_{\Pi_{\rho_n}^{x_j,y_j}}$ appearing in (7) will take us closer to $\bigcap_{j\in\mathcal{J}_n} \Pi_{r_n}^{x_j,y_j} \neq \emptyset$ than $\sum_{j\in\mathcal{J}_n} \omega_j^{(n)} P_{\Pi_{r_n}^{x_j,y_j}}$. For an illustration of this refer again to Fig. 1.

The initial value r_0 is arbitrarily chosen, so we let it $r_0 := 1$ here. The strategy for choosing r_n is given in Table 1. Let us give now a way to select $(\rho_n)_{n\in\mathbb{Z}_{\geq 0}}$. The parameters $(\rho_n)_{n\in\mathbb{Z}_{\geq 0}}$, that determine the halfspaces to be used by the APSM in (7), are defined as $\gamma(\geq 1)$ multiples of $(r_n)_{n\in\mathbb{Z}_{\geq 0}}$ due to the arguments just stated above and illustrated in Fig. 1. The basic idea for the selection scheme is as follows. If our current estimate u_n achieves the margin r_n, we assume that most likely our next estimate u_{n+1} will also do so such that we can change the parameter ρ_n to a slightly bigger value ρ_{n+1}. On the contrary, if the current classifier u_n does not achieve r_n, we assume that a small decrease of ρ_n to ρ_{n+1} will increase the probability for the next estimate u_{n+1} to achieve $r_{n+1} := r_n$. This idea shares the same rationale with the scheme in [3, (19)] for the adaptation of the margin. For the present design, the variations of the parameters $(\rho_n)_{n\in\mathbb{Z}_{\geq 0}}$ will be governed by the linear parametric model $\nu(\theta - \gamma r) + \gamma r$, where $\theta \in \mathbb{R}$ is a parameter and ν is a sufficiently small positive slope. In this way, an increase of θ will increase ρ, whereas a decrease of θ will force ρ to take smaller values.

For completeness, we present the selection scheme in Table 1. A quantity that checks whether the desired margin r_n is reached at time n for all (x_j, y_j), $j \in \mathcal{J}_n$, is $\max\{(r_n - y_j g_n(x_j))^+ : j \in \mathcal{J}_n\}$. Indeed, if this is equal to 0, then by (6) we achieve r_n since we reached a point in $\bigcap_{j\in\mathcal{J}_n} \Pi_{r_n}^{x_j,y_j} \neq \emptyset$.

6. NUMERICAL EXAMPLE

Following [3], we generated data by mixing two 2-dimensional Gaussian distributions with means $\mu_{11} := [0, 3]^t$, $\mu_{12} := [1, 3]^t$, covariance matrices $\Sigma_{11} := \left[\begin{smallmatrix} 5 & 2.5 \\ 2.5 & 5 \end{smallmatrix}\right]$, $\Sigma_{12} := \left[\begin{smallmatrix} 5 & -1.5 \\ -1.5 & 5 \end{smallmatrix}\right]$, and equal

Define $r_0 \geq 0$, $\gamma \geq 1$, $\rho_0 := \gamma r_0$, $\theta_0 := \gamma r_0$, $\delta\theta > 0$, $\nu \geq 0$, $\texttt{NumFeas} := \texttt{NumInfeas} := 0$, $\texttt{FloorFeas}, \texttt{FloorInFeas} \in \mathbb{Z}_{>0}$, $n = 0$.
loop
  if $\max\{(r_n - y_j g_n(x_j))^+ : j \in \mathcal{J}_n\} = 0$ **then**
    $\texttt{NumFeas} := \texttt{NumFeas} + 1$.
    $\texttt{NumInfeas} := 0$.
    $\rho_{n+1} := (\nu(\theta_n + \delta\theta - \gamma r_n) + \gamma r_n)^+$.
    $\theta_{n+1} := \theta_n + \delta\theta$.
    $r_{n+1} := r_n$.
    if $\texttt{NumFeas} \geq \texttt{FloorFeas}$ **then**
      $r_{n+1} := 2r_n$.
      $\theta_{n+1} := \rho_{n+1} := \gamma r_{n+1}$.
      $\texttt{NumFeas} := 0$.
    end if
  else
    $\texttt{NumInfeas} := \texttt{NumInfeas} + 1$.
    $\texttt{NumFeas} := 0$.
    $\rho_{n+1} := (\nu(\theta_n - \delta\theta - \gamma r_n) + \gamma r_n)^+$.
    $\theta_{n+1} := \theta_n - \delta\theta$.
    $r_{n+1} := r_n$.
    if $\texttt{NumInfeas} \geq \texttt{FloorInfeas}$ **then**
      $r_{n+1} := r_n/2$.
      $\theta_{n+1} := \rho_{n+1} := \gamma r_{n+1}$.
      $\texttt{NumInfeas} := 0$.
    end if
  end if
  $n := n + 1$.
end loop

Table 1. The iterative process that selects the margin parameters.

weights of 0.5 for the first class, and two 2-dimensional Gaussian distributions with means $\mu_{21} := [0, -3]^t$, $\mu_{22} := [1, -3]^t$, covariance matrices $\Sigma_{21} := \left[\begin{smallmatrix} 5 & -2.5 \\ -2.5 & 5 \end{smallmatrix}\right]$, $\Sigma_{22} := \left[\begin{smallmatrix} 5 & 0 \\ 0 & 5 \end{smallmatrix}\right]$, and again equal weights of 0.5 for the second class. For the classification task we used a Gaussian kernel with $\sigma^2 := 0.1$.

We compared the proposed method with the stochastic gradient descent algorithm NORMA of [3]. We utilized the version of NORMA with the ν-trick appearing in [3, (19)] for a slope of $\nu := 0.01$. The learning rate was set equal to $0.9/\sqrt{n}$, $\forall n \in \mathbb{Z}_{\geq 0}$. Since the NORMA method generalizes the kernel perceptron, we also simulated the perceptron method with the same learning rate as in NORMA. Both of these methods were optimized with respect to the data at hand. The best results for NORMA were produced here without any regularization [3].

The label APSM refers to the proposed method of (7) with the index set $\mathcal{J}_n := \{n\}$, $\forall n \in \mathbb{Z}_{\geq 0}$. The relaxation parameter $\mu_n := 1$, $\forall n \in \mathbb{Z}_{\geq 0}$. In this way, the computational cost of APSM is the same as NORMA of [3] and kernel perceptron.

To show the benefits of concurrent processing, we let also APSM with $\mathcal{J}_n := \{n, n+1, n+2, n+3\}$, $\forall n \in \mathbb{Z}_{\geq 0}$, in (7). The weights in (7) are set to $\omega_j^{(n)} := 1/4$, $\forall j \in \mathcal{J}_n$, $\forall n \in \mathbb{Z}_{\geq 0}$. We also let the relaxation parameter $\mu_n := 1.9\mathcal{M}_n$, $\forall n \in \mathbb{Z}_{\geq 0}$. This version of APSM increases the computational cost due to the calculation of $\mathcal{M}_n$ and due to the increased cardinality of the index set. However, the order of complexity remains the same as that of the simple case of APSM. The associated curve for this version of APSM is APSM(4) in Fig. 2. We initialized the process of Table 1 as follows: $r_0 := 1$, $\gamma := 3$, $\delta\theta := 10^{-2}$, $\nu := 0.1$, $\texttt{FloorFeas} := \texttt{FloorInfeas} := 30$.

For better visualization results we produce smooth curves in Fig. 2 by plotting the accumulated misclassification errors versus the time index. First, by misclassification errors we define the errors resulting by the classifier obtained at iteration n with respect to all the data collected till n. By the term accumulated misclassification errors we mean that each point in Fig. 2 at time n equals the value of the point at $n-1$ plus the errors occurred at time n. In other words, the misclassification errors at time n are given by the slope of the curves in Fig. 2. We let a total number of 500 data samples, conduct 100 experiments and uniformly average the results.

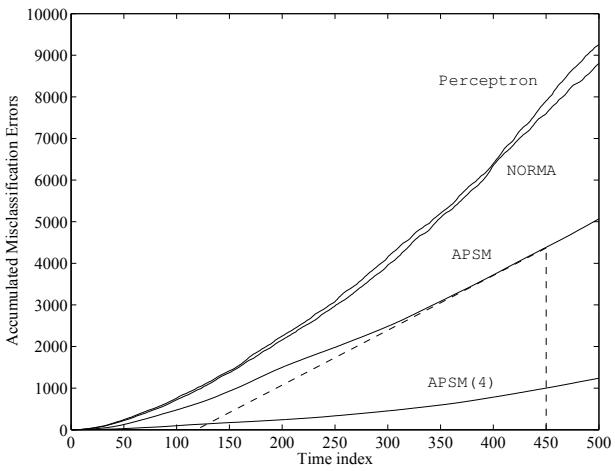

Fig. 2. The accumulated misclassification errors versus time. The number of errors at time n is given by the slope of the curve.

For example, after 450 data samples, the average missclasification errors are 30.7 for the kernel perceptron, 20.9 for NORMA, 12.7 for APSM, and 4.4 for APSM(4).

7. REFERENCES

[1] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, Academic Press, Amsterdam, 3rd edition, 2006.

[2] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*, Cambridge University Press, 2004.

[3] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE Trans. on Signal Processing*, vol. 52, no. 8, pp. 2165–2176, Aug. 2004.

[4] K. Slavakis, I. Yamada, and N. Ogura, "The Adaptive Projected Subgradient Method over the fixed point set of strongly attracting nonexpansive mappings," *Numerical Functional Analysis and Optimization*, vol. 27, no. 7&8, pp. 905–930, 2006.

[5] I. Yamada and N. Ogura, "Adaptive Projected Subgradient Method for asymptotic minimization of sequence of nonnegative convex functions," *Numerical Functional Analysis and Optimization*, vol. 25, no. 7&8, pp. 593–617, 2004.

[6] E. Kreyszig, *Introductory Functional Analysis with Applications*, Wiley Classics Library. John Wiley & Sons, New York, 1989.

[7] N. Aronszajn, "Theory of reproducing kernels," *Trans. of American Mathematical Society*, vol. 68, no. 3, pp. 337–404, 1950.

[8] J-B. Hiriart-Urruty and C. Lemaréchal, *Convex Analysis and Minimization Algorithms*, vol. 1, Springer-Verlag, Berlin, 1993.

[9] A. H. Sayed, *Fundamentals of Adaptive Filtering*, John Wiley & Sons, New Jersey, 2003.

[10] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least-squares algorithm," *IEEE Trans. Signal Processing*, vol. 52, no. 8, pp. 2275–2285, Aug. 2004.