Detection and Identification of Rare Audiovisual Cues

Insperata accident magis saepe quam quae speres.
(Things you do not expect happen more often than things you do expect) Plautus (ca 200(B.C.)

Project no: 027787

DIRAC

Detection and Identification of Rare Audio-visual Cues

Integrated Project
IST - Priority 2

DELIVERABLE NO: D4.12
Incremental learning with bounded memory growth

Date of deliverable: 31.12.2008
Actual submission date: 4.02.2009

Start date of project: 01.01.2006 Duration: 60 months

Organization name of lead contractor for this deliverable: **Idiap Research Institute**

Revision [1]

Project co-funded by the European Commission within the Sixth Framework Program (2002-2006)		
Dissemination Level		
PU	Public	X
PP	Restricted to other program participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

DIRAC
Detection and Identification of Rare Audiovisual Cues

Insperata accident magis saepe quam quae speres.
(Things you do not expect happen more often than things you do expect) Plautus (ca 200(B.C.)

D4.12 Incremental Learning with Bounded Memory Growth

Idiap Research Institute (IDIAP)

Abstract:

A common problem of kernel-based online algorithms, such as the kernel-based Perceptron algorithm, is the amount of memory required to store the online hypothesis, which may increase indefinitely as the algorithm progresses. Furthermore, the computation load of such algorithms grows linearly with the amount of memory used to store the hypothesis.

To attack these problems, most previous work focused on discarding part of the instances in order to keep the memory bounded. We present a new algorithm, in which the instances are not discarded, but projected onto the space spanned by the previous online hypothesis. We call this algorithm Projectron. While the memory size of the Projectron solution cannot be predicted before training, we prove that its solution is guaranteed to be bounded. We derive a relative mistake bound for the proposed algorithm, and deduce from it a slightly different algorithm which outperforms the Perceptron. We call this second algorithm Projectron++. We show that this algorithm can be extended to handle the multiclass and the structured output settings, resulting, as far as we know, the first online bounded algorithm that can learn complex classification tasks. The method of bounding the hypothesis representation can be applied to any conservative online algorithms and to other online algorithms, as it is demonstrated for ALMA2. Experimental results on various datasets show the empirical advantage of our technique compared to various bounded online algorithms, both in terms of memory and accuracy.

Table of Content

1 Introduction

Kernel-based discriminative online algorithms have been shown to perform very well on binary and multiclass classification problems [see, for example, Freund and Schapire, 1999, Crammer and Singer, 2003, Kivinen et al., 2004, Crammer et al., 2006]. Each of these algorithms works in rounds, where at each round a new instance is provided. On rounds where the online algorithm makes a prediction mistake or when the confidence in the prediction is not sufficient, the algorithm adds the instance to a set of stored instances, called *support set*. The online classification function is defined as a weighted sum of kernel combination of the instances in the support set. It is clear that if the problem is not linearly separable or the target hypothesis is changing over time, the classification function will never stop being updated, and consequently, the support set will grow unboundedly. This leads, eventually, to a memory explosion, and it concretely limits the usage of these algorithms for all those applications, such as autonomous agents, for example, where data must be acquired continuously in time.

Several authors have tried to address this problem, mainly by bounding a priori the size of the support set with a fixed value, called *budget*. The first algorithm to overcome the unlimited growth of the support set was proposed by Crammer et al. [2003], and refined by Weston et al. [2005]. In these algorithms, once the size of the support set reaches the budget, an instance from the support set that meets some criteria is removed, and replaced by the new instance. The strategy is purely heuristic and no mistake bounds is given. A similar strategy is also used in NORMA [Kivinen et al., 2004] and SILK [Cheng et al., 2007]. The very first online algorithm to have a fixed memory budget and a relative mistake bound is the Forgetron [Dekel et al., 2007]. A stochastic algorithm that on average achieves similar performances, and with a similar mistake bound was proposed by Cesa-Bianchi et al. [2006]. Opposed to all the previous work, the analysis presented in the last work is within the probabilistic context, and all the bounds derived there are in expectation. A different approach to address this problem for online Gaussian Processes proposed in [Csató and Opper, 2001], where, similar to our approach, the instances are not discarded but rather projected onto the space spanned by the instances from the support set. However, in that paper no mistake bounds is derived and the there is no use of the hinge loss, which may produce sparser solution. A recent work by Langford et al. [2008] proposed a parameter that trades accuracy for sparseness in the weights of online learning algorithms. Nevertheless, this approach cannot induce sparsity for online algorithms with kernels.

In this paper we take a different route. While previous work focused on discarding part of the instances in order to keep the support set bounded, in this work the instances are not discarded. Either they are projected onto the space spanned by the support set, or they are added to the support set. By using this method, we show that the support set and, hence, the online hypothesis, is guaranteed to be bounded, although we cannot predict its size before training. Instead of using a budget parameter, representing the maximum size of the support set, we introduce a parameter trading accuracy for sparseness, depending on the needs of the task at hand. The main advantage of this setup is that by using all training samples, we are able to provide an online hypothesis with high online accuracy. Empirically, as suggested by the experiments, the output hypotheses are represented with relatively small number of instances, and have high accuracy.

We start with the most simple and intuitive kernel-based algorithm, namely the kernel-based Perceptron. We modify the Perceptron algorithm so that the number of stored samples needed to represent the online hypothesis is always bounded. We call this new algorithm *Projectron*. The empirical performance of the Projectron algorithm is on a par with the original Perceptron algorithm. We present a relative mistake bound for the Projectron algorithm, and deduce from it a new online bounded algorithm which outperforms Perceptron, but still retains all of its advantages. We call this second algorithm *Projectron++*. We then extend Projectron++ to the more general cases of multiclass and structured output. As far as we know, this is the first bounded multiclass and structured output online algorithm, with a relative mistake bound[1]. Our technique for bounding the size of the support set can be applied to any conservative kernel-based online algorithms and to other online algorithms, as we demonstrate for ALMA_2 [Gentile, 2001]. Finally, we present some experiments with common datasets, which suggest that Projectron is comparable to Perceptron in performance, but uses much smaller support set. Moreover, experiments with Projectron++ shows that it outperforms all other bounded algorithms, while uses the smallest support set. We also present experiments on the task of phoneme classification, which is considered to be hard and with a relatively very high number of support vectors. When comparing the Projectron++ algorithm with the Passive-Aggressive multiclass algorithm [Crammer et al., 2006], it turns out that the cumulative online error and the test error, after online-to-batch

[1]Note that converting the budget algorithms presented by other authors, such as the Forgetron, to the multiclass or the structured output setting is not trivial, since these algorithms are inherently binary in nature.

conversion, of both algorithms are comparable, although Projectron++ uses much less supports.

In summary, the contributions of this paper are (1) new algorithm, called Projecton, which is derived from the kernel-based Perceptron algorithm, empirically performs the same, but has a bounded support set; (2) relative mistake bound for this algorithm; (3) another algorithm, called Projectron++, based on the notion of large margin, which outperforms the Perceptron algorithm and the proposed Projectron algorithm; (4) multiclass and structured output Projectron++ online algorithm with a bounded support set; and (5) extension of our technique to other online algorithms, exemplified in this paper for ALMA_2.

The rest of the paper is organized as follows: in Section 2 we state the problem definition and the kernel-based Perceptron algorithm. Section 3 introduces Projectron, along with its theoretical analysis. Next, in Section 4 we derived Projectron++. Section 5 presets the multiclass and structured learning variant of Projectron++. In Section 6 we apply our technique for another kernel-based online algorithm, ALMA_2. Section 7 describes experimental results of the algorithms presented on different datasets. Section 8 concludes the paper with a short discussion.

2 Problem Setting and the Kernel-Based Perceptron Algorithm

The basis of our study is the well known *Perceptron* algorithm [Rosenblatt, 1958, Freund and Schapire, 1999]. The Perceptron algorithm learns the mapping $f : \mathcal{X} \to \mathbb{R}$ based on a set of examples $\mathcal{T} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots\}$, where $\mathbf{x}_t \in \mathcal{X}$ is called an *instance* and $y_t \in \{-1, +1\}$ is called a *label*. We denote the prediction of the Perceptron algorithm as $\text{sign}(f(\mathbf{x}))$ and we interpret $|f(\mathbf{x})|$ as the confidence in the prediction. We call the output f of the Perceptron algorithm a *hypothesis*, and we denote the set of all attainable hypotheses by $\mathcal{H}$. In this paper we assume that $\mathcal{H}$ is a Reproducing Kernel Hilbert Space (RKHS) with a positive definite kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ implementing the inner product $\langle \cdot, \cdot \rangle$. The inner product is defined so that it satisfies the reproducing property, $\langle k(\mathbf{x}, \cdot), f(\cdot) \rangle = f(\mathbf{x})$.

The Perceptron algorithm is an online algorithm, where learning takes place in rounds. At each round a new hypothesis function is estimated, based on the previous one. We denote the hypothesis estimated after the t-th round by f_t. The algorithm starts with the zero hypothesis, $f_0 = \mathbf{0}$. At each round t, an instance $\mathbf{x}_t \in \mathcal{X}$ is presented to the algorithm, that predicts

Initialize: $\mathcal{S}_0 = \emptyset,\ f_0 = \mathbf{0}$
For $t = 1, 2, \ldots$
Receive new instance $\mathbf{x}_t$
Predict $\hat{y}_t = \text{sign}(f_{t-1}(\mathbf{x}_t))$
Receive label y_t
If $y_t \neq \hat{y}_t$
$f_t = f_{t-1} + y_t k(\mathbf{x}_t, \cdot)$ *(update the hypothesis)*
$\mathcal{S}_t = \mathcal{S}_{t-1} \cup \mathbf{x}_t$ *(add instance $\mathbf{x}_t$ to the support set)*
Else
$f_t = f_{t-1}$
$\mathcal{S}_t = \mathcal{S}_{t-1}$

Figure 1: The kernel-based Perceptron Algorithm.

a label $\hat{y}_t \in \{-1, +1\}$ by using the current function, $\hat{y}_t = \text{sign}(f_t(\mathbf{x}_t))$. Then, the correct label y_t is revealed. If the prediction $\hat{y}_t$ differs from the correct label y_t, it updates the hypothesis $f_t = f_{t-1} + y_t k(\mathbf{x}_t, \cdot)$, otherwise the hypothesis is left intact, $f_t = f_{t-1}$. The hypothesis f_t can be written as a kernel expansion according to the representer theorem [Schölkopf et al., 2000],

$$f_t(\mathbf{x}) = \sum_{\mathbf{x}_i \in \mathcal{S}_t} \alpha_i k(\mathbf{x}_i, \mathbf{x}) \ , \tag{1}$$

where $\alpha_i = y_i$ and $\mathcal{S}_t$ is defined to be the set of instances for which an update of the hypothesis occurred, i.e., $\mathcal{S}_t = \{\mathbf{x}_i, 0 \leq i \leq t \,|\, \hat{y}_i \neq y_i\}$. The set $\mathcal{S}_t$ is called the *support set*. The Perceptron algorithm is summarized in Figure 1.

Although the Perceptron algorithm is a very simple algorithm, it produces an online hypothesis with a good performance. Our goal is to derive and analyze a new algorithm, which outputs a hypothesis that attains almost the same performance as the Perceptron hypothesis, but can be represented by much less instances, that is, an online hypothesis that is "close" to the Perceptron hypothesis but represented by a smaller support set. Recall that the hypothesis f_t is represented as a weighted sum over all the instances in the support set. The size of this representation is the cardinality of the support set, $|\mathcal{S}_t|$.

3 The Projectron Algorithm

This section starts by deriving the Projectron algorithm, motivated by an example of a finite dimensional kernel space. It continues with a description of how to calculate the projected hypothesis and described some other computational aspects of the algorithm. The section concludes with a theoretical analysis of the algorithm.

3.1 Definition and Derivation

Let us first consider a finite dimensional RKHS $\mathcal{H}$ induced by a kernel such as the polynomial kernel. Since $\mathcal{H}$ is finite dimensional, there is a finite number of linearly independent hypotheses in this space. Hence, any hypothesis in this space can be expressed using a finite number of examples. We can modify Perceptron to use only one set of independent instances as follows. On each round the algorithm receives an instance and predicts its label. On a prediction mistake, we check if the instance $\mathbf{x}_t$ can be spanned by the support set, namely, for scalars $d_i \in \mathbb{R}, 1 \leq i \leq |\mathcal{S}_{t-1}|$, not all zeros, such that

$$k(\mathbf{x}_t, \cdot) = \sum_{\mathbf{x}_i \in \mathcal{S}_{t-1}} d_i k(\mathbf{x}_i, \cdot) \ .$$

If we can find such scalars, the instance is not added to the support set, but instead, the coefficients $\{\alpha_i\}$ in the expansion Eq. (1) are changed to reflect the addition of this instance to the support set, that is, for every i

$$\alpha_i = y_i + y_t d_i.$$

On the other hand, if the instance and the support set are linearly independent, the instance is added to the set with $\alpha_t = y_t$ as before. This technique reduces the size of the support set without changing the hypothesis. A similar approach was used by Downs et al. [2001] to simplify SVM solutions.

Let us consider now the more elaborate case of an infinite dimensional RKHS $\mathcal{H}$ induced by kernels such as the Gaussian kernel. In this case, it is not possible to find a finite number of linearly independent vectors which span the whole space, and hence there is no guarantee that the hypothesis can be expressed by a finite number of instances. However, we can approximate the concept of linear independence with a finite number of vectors [Csató and Opper, 2001, Engel et al., 2004, Orabona et al., 2007].

In particular let us assume that at round t of the algorithm there is a prediction mistake and the mistaken instance $\mathbf{x}_t$ should be added to the

support set, $\mathcal{S}_{t-1}$. Let $\mathcal{H}_{t-1}$ be an RKHS space which is the span of the kernel images of the instances in the set $\mathcal{S}_{t-1}$. Formally,

$$\mathcal{H}_{t-1} = \text{span}\left(\{k(\mathbf{x}, \cdot) | \mathbf{x} \in \mathcal{S}_{t-1}\}\right). \tag{2}$$

Before adding the instance to the support, we construct two hypotheses: a *temporal hypothesis*, f'_t, using the function $k(\mathbf{x}_t, \cdot)$, that is, $f'_t = f_{t-1} + y_t k(\mathbf{x}_t, \cdot)$, and a *projected hypothesis*, f''_t, that is the projection of f'_t onto the space $\mathcal{H}_{t-1}$. That is, the projected hypothesis is a hypothesis from the space $\mathcal{H}_{t-1}$ which is the *closest* to the temporal hypothesis. In the sequel we will describe an efficient way to calculate the projected hypothesis. Denote by δ_t the distance between the hypotheses, $\delta_t = f''_t - f'_t$. If the norm of distance $\|\delta_t\|$ is below some threshold η, we use the projected hypothesis as our next hypothesis, i.e., $f_t = f''_t$, otherwise we use the temporal hypothesis as our next hypothesis, i.e., $f_t = f'_t$. As we show in the following theorem, this strategy assures that the maximum size of the support set is always finite, regardless of the dimension of the RKHS $\mathcal{H}$. Guided by these considerations we can design a new Perceptron-like algorithm that projects the solution onto the space spanned by the previous support vectors whenever possible. We call this algorithm *Projectron*. The algorithm is given in Figure 2.

The parameter η plays an important role in our algorithm. If η is equal to zero, we obtain exactly the same solution of the Perceptron algorithm. In this case, however, the Projectron solution can still be sparser when some of the instances are linearly dependent or when the kernel induces a finite dimensional RKHS $\mathcal{H}$. In case η is greater than zero we trade precision for sparseness. Moreover, as shown in the next section, this implies a bounded algorithmic complexity, namely, the memory and time requirements for each step are bounded. We analyze the effect of η on the classification accuracy in Subsection 3.3.

3.2 Practical Considerations

We now consider the problem of deriving the projected hypothesis f''_t in a Hilbert space $\mathcal{H}$, induced by a kernel function $k(\cdot, \cdot)$. Recall that f'_t is defined as $f'_t = f_t + y_t k(\mathbf{x}_t, \cdot)$. Denote by $P_{t-1} f'_t$ the projection of $f'_t \in \mathcal{H}$ onto the subspace $\mathcal{H}_{t-1} \subseteq \mathcal{H}$. The projected hypothesis f''_t is defined as $f''_t = P_{t-1} f'_t$. Schematically, this is depicted in Figure 3.

Expanding f'_t we have

$$f''_t = P_{t-1} f'_t = P_{t-1}\left(f_{t-1} + y_t k(\mathbf{x}_t, \cdot)\right) . \tag{3}$$

Initialize: $\mathcal{S}_0 = \emptyset$, $f_0 = \mathbf{0}$

For $t = 1, 2, \ldots$

Receive new instance $\mathbf{x}_t$

Predict $\hat{y}_t = \text{sign}(f_{t-1}(\mathbf{x}_t))$

Receive label y_t

If $y_t \neq \hat{y}_t$

Set $f'_t = f_{t-1} + y_t k(\mathbf{x}_t, \cdot)$ (*temporal hypothesis*)

Set $f''_t = f'_t$ projected onto the space $\mathcal{H}_{t-1}$ (*projected hypothesis*)

Set $\delta_t = f''_t - f'_t$

If $\|\delta_t\| \leq \eta$

$f_t = f''_t$

$\mathcal{S}_t = \mathcal{S}_{t-1}$

Else

$f_t = f'_t$

$\mathcal{S}_t = \mathcal{S}_{t-1} \cup \mathbf{x}_t$ *(add $\mathbf{x}_t$ to the support set)*

Else

$f_t = f_{t-1}$

$\mathcal{S}_t = \mathcal{S}_{t-1}$

Figure 2: The Projectron Algorithm.

The projection is a linear operator, hence

$$f''_t = f_{t-1} + y_t P_{t-1} k(\mathbf{x}_t, \cdot) \ . \tag{4}$$

Recall that $\delta_t = f''_t - f'_t$. By substituting f''_t from Eq. (4) and f'_t we have

$$\delta_t = f''_t - f'_t = y_t P_{t-1} k(\mathbf{x}_t, \cdot) - y_t k(\mathbf{x}_t, \cdot) \ . \tag{5}$$

The projection of $f'_t \in \mathcal{H}$ onto a subspace $\mathcal{H}_{t-1} \subset \mathcal{H}$ is defined as the hypothesis in $\mathcal{H}_{t-1}$ closest to f'_t. Hence, let $\sum_{\mathbf{x}_j \in \mathcal{S}_{t-1}} d_j k(\mathbf{x}_j, \cdot)$ be an hypothesis in $\mathcal{H}_{t-1}$, where $\mathbf{d} = (d_1, \ldots, d_{|\mathcal{S}_{t-1}|})$ is a set of coefficients, with $d_i \in \mathbb{R}$. The closest hypothesis is the one for which it holds

$$\|\delta_t\|^2 = \min_{\mathbf{d}} \left\| \sum_{\mathbf{x}_j \in \mathcal{S}_{t-1}} d_j k(\mathbf{x}_j, \cdot) - k(\mathbf{x}_t, \cdot) \right\|^2 \ . \tag{6}$$

Expanding Eq. (6) we get

$$\|\delta_t\|^2 = \min_{\mathbf{d}} \left(\sum_{\mathbf{x}_i,\mathbf{x}_j \in \mathcal{S}_{t-1}} d_j d_i k(\mathbf{x}_j,\mathbf{x}_i) - 2 \sum_{\mathbf{x}_j \in \mathcal{S}_{t-1}} d_j k(\mathbf{x}_j,\mathbf{x}_t) + k(\mathbf{x}_t,\mathbf{x}_t) \right) . \tag{7}$$

Let us define $\mathbf{K}_{t-1} \in \mathbb{R}^{t-1 \times t-1}$ to be the matrix generated by the instances in the support set $\mathcal{S}_{t-1}$, that is, $\{\mathbf{K}_{t-1}\}_{i,j} = k(\mathbf{x}_i,\mathbf{x}_j)$ for every $\mathbf{x}_i,\mathbf{x}_j \in \mathcal{S}_{t-1}$. Let us also define $\mathbf{k}_t \in \mathbb{R}^{t-1}$ to be the vector whose i-th element is $k_{t_i} = k(\mathbf{x}_i,\mathbf{x}_t)$. We have

$$\|\delta_t\|^2 = \min_{\mathbf{d}} \left(\mathbf{d}^T \mathbf{K}_{t-1} \mathbf{d} - 2\mathbf{d}^T \mathbf{k}_t + k(\mathbf{x}_t,\mathbf{x}_t) \right) , \tag{8}$$

Solving Eq. (8), that is, applying the extremum conditions with respect to $\mathbf{d}$, we obtain

$$\mathbf{d}^\star = \mathbf{K}_{t-1}^{-1} \mathbf{k}_t \tag{9}$$

and, by substituting Eq. (9) into Eq. (8),

$$\|\delta_t\|^2 = k(\mathbf{x}_t,\mathbf{x}_t) - \mathbf{k}_t^T \mathbf{d}^\star . \tag{10}$$

Furthermore, by substituting Eq. (9) back into Eq. (4) we get

$$f_t'' = f_{t-1} + y_t \sum_{\mathbf{x}_j \in \mathcal{S}_{t-1}} \mathbf{d}_j^\star \, k(\mathbf{x}_j,\cdot) . \tag{11}$$

We have shown how to calculate both the distance δ_t and the projected hypothesis f_t''. In summary, one needs to compute $\mathbf{d}^\star$ according to Eq. (9), plugs the result either into Eq. (10) and obtains δ_t, or into Eq. (11) and obtains the projected hypothesis.

In order to make the computation more tractable, we need an efficient method to calculate the matrix inversion $\mathbf{K}_t^{-1}$ iteratively. A first method, used by Cauwenberghs and Poggio [2000] for incremental training of SVMs, directly updates the inverse matrix. An efficient way to do it, exploiting the incremental nature of the approach, is to recursively update the inverse matrix: after the addition of a new sample, $\mathbf{K}_t^{-1}$ becomes

$$\mathbf{K}_t^{-1} = \begin{bmatrix} & & & 0 \\ & \mathbf{K}_{t-1}^{-1} & & \vdots \\ & & & 0 \\ 0 & \cdots & 0 & 0 \end{bmatrix} + \frac{1}{\|\delta_t\|^2} \begin{bmatrix} \mathbf{d}^\star \\ -1 \end{bmatrix} \begin{bmatrix} \mathbf{d}^{\star T} & -1 \end{bmatrix} \tag{12}$$

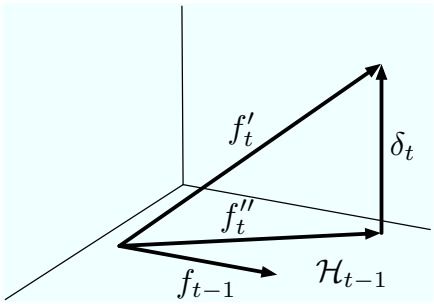


Figure 3: Geometrical interpretation of the projection of the hypothesis f''_t onto the subspace $\mathcal{H}_{t-1}$.

where $\mathbf{d}^\star$ and $\|\delta_t\|^2$ are already evaluated during the previous steps of the algorithm, as given by Eq. (9) and Eq. (10). Thanks to this incremental evaluation, the time complexity of the linear independence check is $O(|\mathcal{S}_{t-1}|^2)$, as one can easily see from Eq. (9). Note that the matrix $\mathbf{K}_{t-1}$ can be safely inverted since, by incremental construction, it is always full-rank. An alternative way to derive the inverse matrix is to use the Cholesky decomposition of $\mathbf{K}_{t-1}$ and to update it recursively. This is known to be numerically more stable than directly updating the inverse. In our experiments, however, we found out that the method presented here is as stable as the Cholesky decomposition.

Overall, the time complexity of the algorithm is $O(|\mathcal{S}_t|^2)$, as describe above, and the space complexity is $O(|\mathcal{S}_t|^2)$, due to the storage of the matrix $\mathbf{K}_t^{-1}$, similar to the second-order Perceptron algorithm [Cesa-Bianchi et al., 2005]. A summary of the derivation of f''_t, the projection of f'_t onto the space spanned by $\mathcal{S}_{t-1}$ is describe in Figure 4.

3.3 Analysis

We now analyze the theoretical aspects of the proposed algorithm. First, we present a theorem which states that the size of the support set of the Projectron algorithm is bounded.

Theorem 1 *Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a continuous Mercer kernel, with $\mathcal{X}$ a compact subset of a Banach space. Then, for any training sequence $\mathcal{T} = \{(\mathbf{x}_i, y_i)\}, i = 1, \cdots, \infty$ and for any $\eta > 0$, the size of the support set of the Projectron algorithm is finite.*

Input: new instance $\mathbf{x}_t$, $\mathbf{K}_{t-1}^{-1}$, and the support set $\mathcal{S}_{t-1}$

- Set $\mathbf{k}_t = \big(k(\mathbf{x}_1, \mathbf{x}_t), k(\mathbf{x}_2, \mathbf{x}_t), \ldots, k(\mathbf{x}_{|\mathcal{S}_{t-1}|}, \mathbf{x}_t)\big)$
- Solve $\mathbf{d}^\star = \mathbf{K}_{t-1}^{-1}\mathbf{k}_t$
- Set $\|\delta_t\|^2 = k(\mathbf{x}_t, \mathbf{x}_t) - \mathbf{k}_t^T \mathbf{d}^\star$
- The projected hypothesis is $f_t'' = f_{t-1} + y_t \sum_{\mathbf{x}_j \in \mathcal{S}_{t-1}} \mathbf{d}_j^\star \, k(\mathbf{x}_j, \cdot)$
- Kernel inverse matrix for the next round

$$\mathbf{K}_t^{-1} = \begin{bmatrix} & & & 0 \\ & \mathbf{K}_{t-1}^{-1} & & \vdots \\ & & & 0 \\ 0 & \cdots & 0 & 0 \end{bmatrix} + \frac{1}{\|\delta_t\|^2} \begin{bmatrix} \mathbf{d}^\star \\ -1 \end{bmatrix} \begin{bmatrix} \mathbf{d}^{\star T} & -1 \end{bmatrix}$$

Output: the projected hypothesis f_t'', the measure δ_t and the kernel inverse matrix $\mathbf{K}_t^{-1}$.

Figure 4: Calculation of the projected hypothesis f_t''.

Proof The proof of this theorem goes along the same lines as the proof of Theorem 3.1 in [Engel et al., 2004]. From the Mercer theorem it follows that there exists a mapping $\phi : \mathcal{X} \to \mathcal{H}'$, where $\mathcal{H}'$ is an Hilbert space, $k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$ and ϕ is continuous. Given that ϕ is continuous and that $\mathcal{X}$ is compact, we obtain that $\phi(\mathcal{X})$ is compact. From the definition of δ_t in Eq. (6) we get

$$\|\delta_t\|^2 = \min_{\mathbf{d}} \left\| \sum_{\mathbf{x}_j \in \mathcal{S}_{t-1}} d_j k(\mathbf{x}_j, \cdot) - k(\mathbf{x}_t, \cdot) \right\|^2 \leq \min_{d_j} \|d_j k(\mathbf{x}_j, \cdot) - k(\mathbf{x}_t, \cdot)\|^2 \tag{13}$$

$$= \min_{d_j} \|d_j \phi(\mathbf{x}_j) - \phi(\mathbf{x}_t)\|^2 \tag{14}$$

for any $1 \leq j \leq |\mathcal{S}_{t-1}|$. Recall that $\|\delta_t\|$ is less than or equal to η, hence from the definition of packing numbers [Cucker and Zhou, 2007, Definition 5.17], we get that the maximum size of the support set in the Projectron algorithm is bounded by the packing number at scale η of $\phi(\mathcal{X})$. This number, in turn, is bounded by the covering number at scale $\eta/2$, and it is finite because the

set is compact [Cucker and Zhou, 2007, Preposition 5.18]. ■

Note that this theorem guarantees that the size of the support set is bounded, however it does not state that the size of the support set is fixed or that it can be estimated before training.

The next theorem provides a mistake bound. The main idea is to bound the maximum number of mistakes of the algorithm, relatively to the any hypothesis $g \in \mathcal{H}$, even chosen in hindsight. First, we define the loss with a margin $\gamma \in \mathbb{R}$ of the hypothesis g on the example $(\mathbf{x}_t, y_t)$ as

$$\ell_\gamma(g(\mathbf{x}_t), y_t) = \max\{0, \gamma - y_t g(\mathbf{x}_t)\}, \tag{15}$$

and we define the cumulative loss, D_γ, of g on the first T examples as

$$D_\gamma = \sum_{t=1}^{T} \ell_\gamma(g(\mathbf{x}_t), y_t) \tag{16}$$

Before stating the bound, we present a lemma that will be used in the rest of our proofs. We will use its first statement to bound the scalar product between a projected sample and the competitor, and its second statement to derive the scalar product between the current hypothesis and the projected sample.

Lemma 1 *Let $(\hat{\mathbf{x}}, \hat{y})$ be an example, with $\hat{\mathbf{x}} \in \mathcal{X}$ and $\hat{y} \in \{+1, -1\}$. Denote by $f(\cdot)$ an hypothesis in $\mathcal{H}$, and denote $q(\cdot)$ any function in $\mathcal{H}$, then the following holds for any $\tau \geq 0$*

$$\hat{y}\tau\langle f, q\rangle \geq \tau\gamma - \tau\ell_\gamma(f(\hat{\mathbf{x}}), \hat{y}) - \tau\|f\| \cdot \|q - k(\hat{\mathbf{x}}, \cdot)\| \ . \tag{17}$$

Moreover, if $f(\cdot)$ can be written as $\sum_{i=1}^{m} \alpha_i k(\mathbf{x}_i, \cdot)$ with $\alpha_i \in \mathbb{R}$ and $\mathbf{x}_i \in \mathcal{X}, i = 1, \cdots, m$, and $q(\cdot)$ is the projection of $k(\hat{\mathbf{x}}, \cdot)$ onto the space spanned by $k(\mathbf{x}_i, \cdot), i = 1, \cdots, m$, then

$$\hat{y}\tau\langle f, q\rangle = \hat{y}\tau f(\hat{\mathbf{x}}) \tag{18}$$

Proof The first inequality comes from an application of the Cauchy-Schwarz inequality and the definition of the hinge loss in (15). The second equality follows from the fact that $\langle f, q - k(\hat{\mathbf{x}}, \cdot)\rangle = 0$, because $f(\cdot)$ is orthogonal to the difference between $k(\hat{\mathbf{x}}, \cdot)$ and its projection onto the space where $f(\cdot)$ lives. ■

With these definitions at hand, we can state the following bound for Projectron.

Theorem 2 *Let* $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_T, y_T)$ *be a sequence of instance-label pairs where* $\mathbf{x}_t \in \mathcal{X}$*,* $y_t \in \{-1, +1\}$*, and* $k(\mathbf{x}_t, \mathbf{x}_t) \leq 1$ *for all* t*. Assume that the Projectron algorithm is run with* $\eta \geq 0$*, then the number of prediction mistakes it makes on the sequence is bounded by*

$$\frac{\|g\|^2 + 2D_1}{1 - 2\eta\|g\|}$$

where g *is an arbitrary function in* $\mathcal{H}$*, such that* $\|g\| \leq \frac{1}{2\eta}$*.*

Proof Define the relative progress in each round as $\Delta_t = \|f_{t-1} - g\|^2 - \|f_t - g\|^2$. We bound the progress from above and below. On rounds where there is no mistake Δ_t equals 0. On rounds where there is a mistake there are two possible updates: either $f_t = f_{t-1} + y_t P_{t-1} k(\mathbf{x}_t, \cdot)$ or $f_t = f_{t-1} + y_t k(\mathbf{x}_t, \cdot)$. In the following we start bounding the progress from below, when the update is of the former type. In particular we set $q(\cdot) = P_{t-1} k(\mathbf{x}_t, \cdot)$ in Lemma 1 and use $\delta_t = y_t P_{t-1} k(\mathbf{x}_t, \cdot) - y_t k(\mathbf{x}_t, \cdot)$ from Eq. (5)

$$\begin{aligned}\Delta_t = \|f_{t-1} - g\|^2 - \|f_t - g\|^2 &= 2\tau_t y_t \langle g - f_{t-1}, P_{t-1} k(\mathbf{x}_t, \cdot)\rangle - \tau_t^2 \|P_{t-1} k(\mathbf{x}_t, \cdot)\|^2 \\ &\geq \tau_t \Big(2 - 2\ell_1(g(\mathbf{x}_t), y_t) - \tau_t \|P_{t-1} k(\mathbf{x}_t, \cdot)\|^2 - 2\|g\| \cdot \|\delta_t\| - 2 f_{t-1}(\mathbf{x}_t)\Big) \,. \end{aligned} \tag{19}$$

Moreover, on every projection update $\|\delta_t\| \leq \eta$ and using the theorem assumption $\|P_{t-1} k(\mathbf{x}_t, \cdot)\| \leq 1$ we then have

$$\Delta_t \geq \tau_t \Big(2 - 2\ell_1(g(\mathbf{x}_t), y_t) - \tau_t - 2\eta\|g\| - 2 f_{t-1}(\mathbf{x}_t)\Big) \,.$$

We can further bound Δ_t by noting that on every prediction mistake $f_{t-1}(\mathbf{x}_t) \leq 0$. Overall we have

$$\|f_{t-1} - g\|^2 - \|f_t - g\|^2 \geq \tau_t \Big(2 - 2\ell_1(g(\mathbf{x}_t), y_t) - \tau_t - 2\eta\|g\|\Big) \,. \tag{20}$$

When there is an update without projection, in the same way it can be derived that

$$\|f_{t-1} - g\|^2 - \|f_t - g\|^2 \geq \tau_t \Big(2 - 2\ell_1(g(\mathbf{x}_t), y_t) - \tau_t\Big) \,.$$

hence the bound in (20) holds in both cases.

We sum over t on both sides. Let τ_t be an indicator function for a mistake on the t-th round, that is, τ_t is 1 if there is a mistake on round t and 0 otherwise, hence it can be upper bounded by 1. The left hand side of

the equation is a telescopic sum, hence it collapses to $\|f_0 - g\|^2 - \|f_T - g\|^2$, which can be upper bounded by $\|g\|^2$, using the fact that $f_0 = \mathbf{0}$ and that $\|f_T - g\|^2$ is non-negative. Finally, we have

$$\|g\|^2 + 2D_1 \geq M\left(1 - 2\eta\|g\|\right) \ ,$$

where M is the number of mistakes. ■

The next theorem states a worst-case mistake bound using the difference of scalar products, instead of norms, as a measure of progress. The proof of the theorem is given in the appendix.

Theorem 3 *Let $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_T, y_T)$ be a sequence of instance-label pairs where $\mathbf{x}_t \in \mathcal{X}$, $y_t \in \{-1, +1\}$, and $k(\mathbf{x}_t, \mathbf{x}_t) \leq 1$ for all t. Assume that Projectron is run with $\eta \geq 0$, then the number of prediction mistakes it makes on the sequence is bounded by*

$$\frac{1}{(1 - \eta\|g\|)^2} \frac{\left(\|g\| + \sqrt{\|g\|^2 + 4D_1}\right)^2}{4}$$

where g is an arbitrary function in $\mathcal{H}$, such that $\|g\| \leq \frac{1}{\eta}$.

The last theorem suggests that the performance of the Projectron algorithm is slightly worse than that of the Perceptron algorithm. Specifically, if we set $\eta = 0$, we recover the best known bound for the Perceptron algorithm [see for example Gentile, 2003]. Hence the degradation in the performance of Projectron compared to Perceptron are related to $\frac{1}{(1-\eta\|g\|)^2}$. Empirically, the Projectron algorithm and the Perceptron algorithm perform in a similar way, with a wide range of settings of η.

4 The Projectron++ Algorithm

The proof of Theorem 2 suggests how to improve the Projectron algorithm to go beyond the performance of the Perceptron algorithm, while maintaining a bounded support set. We can change the Projectron algorithm so that an update takes place not only if there is a prediction mistake, but also when the confidence of the prediction is low. We refer to this latter case as a *margin error*, that is, $0 < y_t f_{t-1}(\mathbf{x}_t) < 1$. This strategy is known to improve the classification rate but also increases the size of the support set [Crammer et al., 2006]. A possible solution to this obstacle is not to update

on every round a margin error occurs, but only when there is a margin error and the new instance *can be projected* onto the support set. Hence, the update on margin error rounds would be in the general form

$$f_t = f_{t-1} + y_t \tau_t P_{t-1} k(\mathbf{x}_t, \cdot) \ , \tag{21}$$

with $0 < \tau_t \leq 1$. The last constraint comes from the proof of Theorem 2 where we upper bound τ_t by 1. Note that setting τ_t to 0 is equivalent to leave the hypothesis unchanged.

Let us denote the following progress on round t by β_t

$$\beta_t = \tau_t \left(2 - \tau_t \|P_{t-1} k(\mathbf{x}_t, \cdot)\|^2 - 2\|\delta_t\| \cdot \|g\| - 2 f_{t-1}(\mathbf{x}_t)\right) . \tag{22}$$

In particular it is easy to see from Eq. (19) that the progress on margin error round t, without considering the loss term, is at least β_t for $0 < \tau_t \leq 1$, and it is equal to 0 when there is no update. Whenever this progress is non-negative the worst-case number of mistakes decreases, hopefully along with the classification error rate of the algorithm. With this modification we expect better performance, that is, fewer mistakes, but without any increase of the support set size. We can even expect solutions with a smaller support set, since new instances can be added to the support set only if misclassified, hence having less mistakes should result in a smaller support set. We name this algorithm *Projectron++*. The following theorem states a mistake bound for Projectron++, and guides us how to choose τ_t.

Theorem 4 *Let $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_T, y_T)$ be a sequence of instance-label pairs where $\mathbf{x}_t \in \mathcal{X}$, $y_t \in \{-1, +1\}$, and $k(\mathbf{x}_t, \mathbf{x}_t) \leq 1$ for all t. Let g be an arbitrary function in $\mathcal{H}$. Assume that Projectron++ is run with $\eta \geq 0$, then the number of prediction mistakes it makes on the sequence is bounded by*

$$\frac{\|g\|^2 + 2D_1 - \sum_{\{t:0<y_t f_{t-1}(\mathbf{x}_t)<1\}} \beta_t}{1 - 2\eta\|g\|} \ ,$$

where g is an arbitrary function in $\mathcal{H}$, such that $\|g\| \leq \frac{1}{2\eta}$, β_t is defined in (22) and

$$0 < \tau_t \leq \min \left\{ \frac{2\ell_1(f_{t-1}(\mathbf{x}_t), y_t) - \frac{\delta_t}{\eta}}{\|P_{t-1} k(\mathbf{x}_t, \cdot)\|^2}, 1 \right\} .$$

Proof The proof is similar to the proof of Theorem 2, where the difference is that during margin error rounds we update the solution only if a projection

with positive relative progress is possible. In these rounds we bound the progress with

$$\begin{aligned}\beta_t &= \tau_t \left(2 - \tau_t \|P_{t-1}k(\mathbf{x}_t, \cdot)\|^2 - 2\|\delta_t\| \cdot \|g\| - 2f_{t-1}(\mathbf{x}_t)\right) \\ &\leq \tau_t \left(2 - \tau_t \|P_{t-1}k(\mathbf{x}_t, \cdot)\|^2 - \frac{\|\delta_t\|}{\eta} - 2f_{t-1}(\mathbf{x}_t)\right) \\ &= \tau_t \left(\ell_1(f_{t-1}(\mathbf{x}_t), y_t) - \tau_t \|P_{t-1}k(\mathbf{x}_t, \cdot)\|^2 - \frac{\|\delta_t\|}{\eta}\right)\end{aligned}$$

where the first inequality comes from the hypothesis $\|g\| \leq \frac{1}{2\eta}$. Hence the progress is positive if

$$\tau_t \leq \frac{2\ell_1(f_{t-1}(\mathbf{x}_t), y_t) - \frac{\delta_t}{\eta}}{\|P_{t-1}k(\mathbf{x}_t, \cdot)\|^2}.$$

Constraining τ_t to be less than or equal to 1 we have the update rule in the theorem. Hence the mistake bound can be derived as in Theorem 2. ■

The theorem gives us some freedom for the choice of τ_t. Experimentally we have observed that we obtain the best performance if the update is done with the following rule

$$\tau_t = \min \left\{ \frac{\ell_1(f_{t-1}(\mathbf{x}_t), y_t)}{\|P_{t-1}k(\mathbf{x}_t, \cdot)\|^2}, \frac{2\ell_1(f_{t-1}(\mathbf{x}_t), y_t) - \frac{\delta_t}{\eta}}{\|P_{t-1}k(\mathbf{x}_t, \cdot)\|^2}, 1 \right\}.$$

The added term in the minimum comes from ignoring the term $-\frac{\delta_t}{\eta}$ and in finding the maximum of the quadratic equation. Notice that the term $\|P_{t-1}k(\mathbf{x}_t, \cdot)\|^2$ in the last equation can be practically computed as $\mathbf{k}_t^T \mathbf{d}^\star$, as can be derived from the same techniques presented in Subsection 3.2.

We note in passing that, the condition whether $\mathbf{x}_t$ can be projected onto $\mathcal{H}_{t-1}$ on margin error may stated as $\ell_1(f_{t-1}(\mathbf{x}_t), y_t) \geq \frac{\|\delta_t\|}{2\eta}$. This means that if the loss is relatively large, the progress is also large and the algorithm can afford "wasting" a bit of it for the sake of projecting.

The algorithm is summarized in Figure 2. The performance of the Projectron++ algorithm, the Projectron algorithm and several other bounded online algorithms are compared and reported in Section 7.

5 Extension to Multiclass and Structured Output

In this section we extend Projectron++ to the multiclass and the structured output setting (note that Projectron can be generalized in a similar way).

We start by presenting the more complex decision problem, namely the structured output, and then we derive the multiclass decision problem as a private case.

In structured output decision problems the set of possible labels has a unique and defined structure, such as trees, graphs and sequences [Collins, 2000, Taskar et al., 2003, Tsochantaridis et al., 2004]. Denote the set of all labels as $\mathcal{Y} = \{1, \ldots, k\}$. Each instance is associated with a label from $\mathcal{Y}$. Generally, in structured output problems there may be dependencies between the instance and the label, as well as between labels. Hence, to capture these dependencies, the input and the output pairs are represented in a common feature representation. The learning task is therefore defined as finding a function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ such that

$$y'_t = \arg\max_{y \in Y} f(\mathbf{x}_t, y) \tag{23}$$

Let us generalize the definition of the RKHS $\mathcal{H}$ introduced in Section 2 to the case of structured learning. A kernel function in this setting should reflect the dependencies between the instances and the labels, hence we define the structured kernel function as a function on the domain of the instances and the labels, namely, $k^S : (\mathcal{X} \times \mathcal{Y})^2 \to \mathbb{R}$. This kernel function induces the RKHS $\mathcal{H}^S$, where the inner product in this space is defined such that it satisfies the reproducing property, $\langle k^S((\mathbf{x}, y), \cdot), f \rangle = f(\mathbf{x}, y)$.

As in the binary classification algorithm presented earlier, the structured output online algorithm receives instances in a sequential order. Upon receiving an instance, $\mathbf{x}_t \in \mathcal{X}$, the algorithm predicts a label, y'_t, according to Eq. (23). After making its prediction, the algorithm receives the correct label, y_t. We define the loss suffered by the algorithm on round t for the example $(\mathbf{x}_t, y_t)$ as

$$\ell^S_\gamma(f, \mathbf{x}_t, y_t) = \max\{0, \gamma - f(\mathbf{x}_t, y_t) + \max_{y'_t \neq y_t} f(\mathbf{x}_t, y'_t)\}. \tag{24}$$

Note that sometimes it is useful to define γ as a function $\gamma : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ describing the discrepancy between the predicted label and the true label. Our algorithm can handle such a label cost function, but we will not discuss this issue here [see Crammer et al., 2006, for further details]. As in the binary case, on rounds where there is a prediction mistake, $y'_t \neq y_t$, or where there is a margin mistake, $f(\mathbf{x}_t, y_t) - f(\mathbf{x}_t, y'_t) < \gamma$, the algorithm updates the hypothesis f_{t-1} by adding to it either $\tau_t \left(k((\mathbf{x}_t, y_t), \cdot) - k((\mathbf{x}_t, y'_t), \cdot)\right)$ or its projection, where $0 < \tau_t \leq 1$ and will be define shortly.

The analysis of the structured output Projectron++ algorithm is similar to the one provided for the binary case. We can easily obtain the generalization of Lemma 1 as follows

Lemma 2 *Let $(\hat{\mathbf{x}}, \hat{y})$ be an example, with $\hat{\mathbf{x}} \in \mathcal{X}$ and $\hat{y} \in \mathcal{Y}$. Denote by $f(\cdot)$ an hypothesis in $\mathcal{H}^S$. Let $q(\cdot) \in \mathcal{H}^S$ then the following holds for any $\tau \geq 0$, $y' \in \mathcal{Y}$:*

$$\tau\langle f, q\rangle \geq \tau\gamma - \tau\ell_\gamma^S(f, \hat{\mathbf{x}}, \hat{y}) - \tau\|f\| \cdot \left\|q - \left(k((\hat{\mathbf{x}}, \hat{y}), \cdot) - k((\hat{\mathbf{x}}, y'), \cdot)\right)\right\| \ . \quad (25)$$

Moreover if $f(\cdot)$ can be written as $\sum_{i=1}^m \alpha_i k((\mathbf{x}_i, y_i), \cdot)$ with $\alpha_i \in \mathbb{R}$ and $\mathbf{x}_i \in \mathcal{X}, i = 1, \cdots, m$, and q is the projection of $k((\hat{\mathbf{x}}, \hat{y}), \cdot) - k((\hat{\mathbf{x}}, y'), \cdot)$ in the space spanned by $k((\mathbf{x}_i, y_i), \cdot), i = 1, \cdots, m$, we have that

$$\tau\langle f, q\rangle = \tau f(\hat{\mathbf{x}}, \hat{y}) \quad (26)$$

Define the cumulative loss D_γ^S for the structured output classification problem as

$$D_\gamma^S = \sum_{t=1}^T \ell_\gamma^S(f(\mathbf{x}_t), y_t). \quad (27)$$

We can now derive the following mistake bound

Theorem 5 *Let $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_T, y_T)$ be a sequence of instance-label pairs where $\mathbf{x}_t \in \mathcal{X}$, $y_t \in \mathcal{Y}$, and $k((\mathbf{x}_t, y), \cdot) \leq 0.5$ for all t and $y \in \mathcal{Y}$. Let g be an arbitrary function in $\mathcal{H}^S$. Assume that Projectron++ is run with $\eta \geq 0$, then the number of prediction mistakes it makes on the sequence is bounded by*

$$\frac{\|g\|^2 + 2D_1^S - \sum_{\{t:0<y_t f_{t-1}(\mathbf{x}_t)<1\}} \beta_t}{1 - 2\eta\|g\|},$$

where g is an arbitrary function in $\mathcal{H}^S$, such as $\|g\| \leq \frac{1}{2\eta}$, β_t is defined in (22) and

$$0 < \tau_t \leq \min\left\{\frac{2\ell_1^S(f_{t-1}(\mathbf{x}_t), y_t) - \frac{\delta_t}{\eta}}{\|a\|^2}, 1\right\},$$

where $a = P_{t-1}k((\mathbf{x}_t, y_t), \cdot) - P_{t-1}k((\mathbf{x}_t, y_t'), \cdot)$.

As in Theorem 4 there is some freedom in the choice of τ_t, and again we set it to

$$\tau_t = \min\left\{\frac{\ell_1^S(f_{t-1}(\mathbf{x}_t), y_t)}{\|a\|^2}, \frac{2\ell_1^S(f_{t-1}(\mathbf{x}_t), y_t) - \frac{\delta_t}{\eta}}{\|a\|^2}, 1\right\}.$$

In the multiclass decision problem case, the kernel $k((\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2))$ is simplified to $\delta_{y_1 y_2} k(\mathbf{x}_1, \mathbf{x}_2)$, where $\delta_{y_1 y_2}$ is the Kronecker delta. This corresponds to use a different prototype for each class. This simplifies the projection step, in fact $k((\mathbf{x}_t, y_t), \cdot)$ can be projected only on the functions in S_{t-1} belonging to y_t, being zero the scalar product with the other functions. So instead of storing a single matrix $\mathbf{K}_{t-1}^{-1}$, we need to store m matrices, where m is the number of classes, each one with the inverse matrix of the Gram matrix of the functions of one class. This gives us a computational and memory gain. To see this suppose that we have m classes, each with n vectors in the support set. Storing a single matrix means having a spatial and algorithmic complexity of $O(m^2 n^2)$ (cf. Section 3), while in the second case the complexity is $O(mn^2)$. We use this method in the multiclass experiments presented in Section 7.

6 Bounding Other Online Algorithms

It is possible to apply the technique in the basis of the Projectron algorithm to any conservative online algorithm. A conservative online algorithm is an algorithm that updates its hypothesis only on rounds on which they made a prediction error. By applying Lemma 1 to a conservative algorithm, we can have a bounded version of it with worst case mistake bounds. As in the previous proofs, the idea is to use Lemma 1 to bound the scalar product of the competitor and the projected function. In this way we have an additional term that is subtracted from the margin γ of the competitor.

The technique presented here can be applied to other online kernel-based algorithms. As an example, we apply our technique to ALMA$_2$ [Gentile, 2001]. Again we define two hypotheses: a temporal hypothesis f'_t, which is the hypothesis of ALMA$_2$ after its update rule, and a projected hypothesis, which is the hypothesis f'_t projected on the set $\mathcal{H}_{t-1}$ as defined in Eq. (2). Define the projection error δ_t as $\delta_t = f'_t - f''_t$. The modified ALMA$_2$ algorithm uses the projected hypothesis f''_t whenever the projection error is smaller than a parameter η, otherwise it uses the temporal hypothesis f'_t. We can state the following bound

Theorem 6 *Let $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_T, y_T)$ be a sequence of instance-label pairs where $\mathbf{x}_t \in \mathcal{X}$, $y_t \in \{-1, +1\}$, and $k(\mathbf{x}_t, \mathbf{x}_t) \leq 1$ for all t. Let α, B and $C \in \mathbb{R}^+$ satisfy the equation*

$$C^2 + 2(1 - \alpha)BC = 1.$$

Assume $ALMA_2(\alpha;B,C)$ projects every time the projection error δ_t is less than $\eta \geq 0$, then the number of prediction mistakes it makes on the sequence is bounded by

$$\frac{D_\gamma}{\gamma-\eta} + \frac{\rho^2}{2} + \sqrt{\frac{\rho^4}{4} + \frac{\rho^2}{\gamma-\eta} D_\gamma + \rho^2}$$

where g is an arbitrary function in $\mathcal{H}$, such that $\|g\| \leq 1$, $\gamma > \eta$ and $\rho = \frac{1}{C^2(\gamma-\eta)^2}$.

Proof The proof follows the original proof presented in [Gentile, 2001]. Specifically, according to Lemma 1, one can replace the relation $y_t \langle g, k(\mathbf{x}_t, \cdot)\rangle \geq \gamma - \ell_\gamma(g(\mathbf{x}_t), y_t)$ with $y_t \langle g, P_{t-1} k(\mathbf{x}_t, \cdot)\rangle \geq \gamma - \eta - \ell_\gamma(g(\mathbf{x}_t), y_t)$, and further substitutes γ with $\gamma - \eta$. ■

7 Experimental Results

In this section we present experimental results that demonstrate the effectiveness of the Projectron algorithm and the Projectron++ algorithm. We compare both algorithms to the Perceptron algorithm, the Forgetron algorithm [Dekel et al., 2007] and the Randomized Budget Perceptron (RBP) algorithm [Cesa-Bianchi et al., 2006]. For Forgetron, we choose the state-of-the-art "self-tuned" variant, which outperforms all its other variants. We used the PA-I variant of the Passive-Aggressive algorithm [Crammer et al., 2006] as a baseline algorithm, as it gives an upper bound to the classification performance of the Projectron++ algorithm.

We tested the algorithms on several standard machine learning datasets[2]: *a9a*, *ijcnn1*, *news20.binary*, *SensIT Vehicle (combined)*, *usps*, *mnist* and a synthetic dataset. The synthetic dataset is built in the same way as in [Dekel et al., 2007]. It is composed of 10000 samples taken from two separate bi-dimensional Gaussian distributions. The means of the positive and negative samples are $(1, 1)$ and $(-1, -1)$, respectively, while the covariance matrix for both is diagonal matrix with $(0.2, 2)$ as its diagonal. Then the labels are flipped with a probability of 0.1 to introduce noise. The list of the datasets, their charateristics and the kernels used are given in Table 1. The parameters of the kernels were selected to have the best performance with the Perceptron and were used for all the other algorithms to have a fair comparison. The C parameter of PA-I was set to 1, to have an update

[2]Downloaded from `http://www.sie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`.

Table 1: Datasets used in the experiments

Dataset	# of samples	# of features	# classes	Kernel	Parameters
a9a	32561	123	2	Gaussian	0.04
ijcnn1	49990	22	2	Gaussian	8
news20.binary	19996	1355191	2	Linear	-
SensIT Vehicle	78823	100	2	Gaussian	0.125
Synthetic	10000	2	2	Gaussian	1
mnist	60000	780	10	Polynomial	7
usps	7291	256	10	Polynomial	13
timit (subset)	$\sim$ 150000	351	39	Gaussian	80

similar to Perceptron and Projectron. All the experiments were performed over five different permutations of the training set.

Experiments with one setting of η. In the first set experiments we compared the online average number of mistakes and the support set size of all algorithms. Both Forgetron and RBP works by discarding vectors from the support set, if the size of the support set reached the budget size, B. Hence for a fair comparison, we set η to some value and selected the budget size of Forgetron and of RBP to be equal to the final size of the support set of Projectron. In particular in Figure 6 we set $\eta = 0.1$ in Projectron and ended up with a support set of size 793, hence $B = 793$. In Figure 6(a) the average online error rate for all algorithms on the *Adult9* dataset is plotted. Note that Projectron closely tracks Perceptron. On the other hand Forgetron and RBP stop improving after reaching the support set size B, around 3400 samples. Moreover, due to its theoretically derived formulation, Projectron++ achieves better results than Perceptron, even if being bounded.

Figure 6(b) shows the growth of the support set as a function of the number of samples. While for the PA-I and the Perceptron the growth is clearly linear, it is sub-linear for Projectron and Projectron++: they will reach a maximum size and then they will stop growing, as stated in Theorem 1. Another important consideration is that Projectron++ outperforms Projectron *both* with respect to the size of the support set and number of mistakes. The same behaviour can be seen in Figure 7, for the synthetic database. Here the gain in performance of Projectron++ over Perceptron, Forgetron and RBP is even bigger.

Experiments with a range of values for η - Binary. To analyze in more detail the behavior of our algorithms we decided to run other tests using a range of values of η. For each value we obtain a different size of

the support set and a different number of mistakes. We have used these data to plot a curve corresponding to percentage of number of mistakes as a function of the support set size. The same curve was plotted for Forgetron and RBP, where the budget size selected as describe before. In this way we compared the algorithms along the continuous range of budget sizes, displaying the trade-off for each of them between sparseness and accuracy. For the remaining experiments we have chosen not to show the performance of Projectron, for it was always outperformed by Projectron++.

In Figure 8 we show the performance of the algorithms on different binary datasets: (a) ijcnn1, (b) a9a, (c) news20.binary, and (d) SensIT Vehicle (combined). Given that Projectron++ used a different support set size each permutation of the training samples, we plotted five curves related to five permutations. RBP and Forgetron have fixed budget size set in advance, hence for these algorithm we just plotted standard deviation bars. In all the experiments Projectron++ outperforms Forgetron and RBP, given they all have the same support set size. It can be noticed that there is always a point in all the graphs where the performance of Projectron++ is better than Perceptron, still with a smaller support set. In particular Projectron++ gets closer to the classification rate of the PA-I, without paying the price of a larger support set. Note that the performance of Projectron++ is consistently better than RBP and Forgetron, regardless of the kernel used, and in particular, on the database *news20.binary*, being a text classification task with linear kernel. In this task the samples are almost orthogonal one to the others, so a projection is hard. Nevertheless Projectron++ succeeded in obtaining better performance. The reason is probably due to the margin updates, done without increasing the size of the solution. Note that a similar modification would not be trivial in Forgetron and in RBP, because the proofs of their mistake bounds strongly depend on the rate of growth of the norm of the solution.

Experiments with a range of values for η - Multiclass. We have also considered multiclass datasets, using the multiclass version of Projectron++. Due to the fact that there are no other bounded online algorithms with a mistake bound for multiclass, we have simply extended RBP to multiclass, discarding a vector at random from the solution each time a new instance added. We name it Multiclass Random Budget Perceptron (MRBP). It should be possible to prove a mistake bound for this algorithm, extending the proof in [Cesa-Bianchi et al., 2006]. In Figure 9 we show the results for Perceptron, Passive-Aggressive, Projectron++ and MRBP trained on (a) USPS, and (b) MNIST datasets. The results confirm the findings found for the binary case.

The last dataset used in our experiments is a corpus of continuous natural speech for the task of phoneme classification. The data we used is a subset of the TIMIT acoustic-phonetic dataset, which is a phonetically transcribed corpus of high quality continuous speech spoken by North American speakers [Lemel et al., 1986]. The features were generated from nine adjacent vectors of Mel-Frequency Cepstrum Coefficients (MFCC) along with their first and the second derivatives. The TIMIT corpus is divided into a training set and a test set in such a way that no speakers from the training set appear in the test set (speaker independent). We randomly selected 500 training utterances from the training set. The average online errors are shown in Figure 10 (a). We also tested the performance of the algorithm on the proposed TIMIT core test set composed of 192 utterances, the result are in Figure 10 (b). We did that by using online-to-batch conversion [Cesa-Bianchi et al., 2004] to have a bounded batch solution. We did not test the performance of MRBP on the test set because for this algorithm the online-to-batch conversion does not produce a bounded solution. We compare the batch solution to online-to-batch conversion of PA-I solution. The results of Projectron++ are comparable to that of PA-I, while the former uses smaller support set. The result also suggests that the batch solution is stable comparing to the value of η, as the difference in performance on test set is minor.

8 Discussion

This paper presented two different versions of a bounded online learning algorithm. The algorithms depend on a parameter that allows to trade accuracy for sparseness of the solution. The size of the solution is always guaranteed to be bounded, albeit unknown before the training begins, therefore it solves the memory explosion problem of the Perceptron and similar algorithms. Although the size of the support set cannot be determined before training, practically, for a given target accuracy, the size of the support set of Projectron or Projectron++ is much smaller than that of other budget algorithms such as Forgetron and RBP.

The first algorithm, the Projectron algorithm, is based on the Perceptron algorithm. While the empirical performance of Projectron is comparable to that of Perceptron, but with the advantage of a bounded solution. The second algorithm, Projectron++, introduces the notion of large margin and hence it always outperforms the Perceptron algorithm, while assuring a bounded solution. The experimental results suggest that Projectron++

outperforms other online bounded algorithms such as Forgetron and RBP, with the smallest hypothesis size.

These are two unique advantages of Projectron and Projectron++. First, these algorithm can be extended to the multiclass and the structured output settings. Second, a standard online-to-batch conversion can be applied to the online bounded solution of these algorithm, resulted a batch bounded solution. The major drawback of these algorithms is their time and space complexity, which is quadratic in the size of the support set. Trying to overcome this acute problem is left for future work.

Appendix

Proof of Theorem 3 We proceed as in [Shalev-Shwartz and Singer, 2005], bounding the quantity $\langle f_T, g\rangle$ from above and below. Remembering that we update only during errors round, taking into account that $\|P_{t-1}k(\mathbf{x}_t,\cdot)\|^2 \leq \|k(\mathbf{x}_t,\cdot)\|^2 = k(\mathbf{x}_t,\mathbf{x}_t) \leq 1$ and (18), we have

$$\begin{aligned}\|f_T\|^2 &= \|f_{T-1}\|^2 + \|P_{T-1}k(\mathbf{x}_T,\cdot)\|^2 + 2y_T\langle f_{T-1}, P_{T-1}k(\mathbf{x}_T,\cdot)\rangle \\ &= \|f_{T-1}\|^2 + \|P_{T-1}k(\mathbf{x}_T,\cdot)\|^2 + 2y_T f_{T-1}(\mathbf{x}_T) \leq \|f_T\|^2 + 1 \leq M\ .\end{aligned} \tag{28}$$

Hence we have that

$$\langle f_T, g\rangle \leq \|f_T\|\|g\| \leq \|g\|\sqrt{M} \tag{29}$$

For the lower bound we set $q(\cdot) = P_{t-1}k(\mathbf{x}_t,\cdot)$ in Lemma 1 and proceed analogously as in (19)

$$\begin{aligned}\langle f_T, g\rangle &= \langle f_{T-1} + y_T P_{T-1}k(\mathbf{x}_T,\cdot), g\rangle \geq \langle f_{T-1}, g\rangle + 1 - \ell_1(g(\mathbf{x}_T), y_T) - \eta\|g\| \\ &\geq M(1-\eta\|g\|) - \sum_{t=1}^{T}\ell_1(g(\mathbf{x}_t), y_t)\ .\end{aligned} \tag{30}$$

Putting together (28) and (30) we have

$$M(1-\eta\|g\|) - \|g\|\sqrt{M} - D_1 \leq 0$$

and solving for M concludes the proof. ■

References

G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In *Advances in Neural Information Processing Systems 14*, 2000.

N. Cesa-Bianchi, A. Conconi, and C. Gentile. On the generalization ability of on-line learning algorithms. *IEEE Trans. on Information Theory*, 50 (9):2050–2057, 2004.

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

N. Cesa-Bianchi, A. Conconi, and C. Gentile. Tracking the best hyperplane with a simple budget Perceptron. In *Proc. of the 19th Conference on Learning Theory*, pages 483–498, 2006.

L. Cheng, S. V. N. Vishwanathan, D. Schuurmans, S. Wang, and T. Caelli. Implicit online learning with kernels. In *Advances in Neural Information Processing Systems 19*, 2007.

M. Collins. Discriminative reranking for natural language parsing. In *Machine Learning: Proceedings of the Seventeenth International Conference*, 2000.

K. Crammer and Y. Singer. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991, 2003.

K. Crammer, J. Kandola, and Y. Singer. Online classification on a budget. In *Advances in Neural Information Processing Systems 16*, 2003.

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

L. Csató and M. Opper. Sparse representation for gaussian process models. In *Advances in Neural Information Processing Systems 13*, 2001.

F. Cucker and D.X. Zhou. *Learning Theory: An Approximation Theory Viewpoint*. Cambridge University Press, New York, NY, USA, 2007.

O. Dekel, S. Shalev-Shwartz, and Y. Singer. The Forgetron: A kernel-based perceptron on a budget. *SIAM Journal on Computing*, 37(5):1342–1372, 2007.

T. Downs, K. E. Gates, and A. Masters. Exact simplification of support vectors solutions. *Journal of Machine Learning Research*, 2:293–297, 2001.

Y. Engel, S. Mannor, and R. Meir. The kernel recursive least-squares algorithm. *IEEE Transactions on Signal Processing*, 52(8):2275 – 2285, 2004.

Y. Freund and R.E. Schapire. Large margin classification using the Perceptron algorithm. *Machine Learning*, pages 277–296, 1999.

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

C. Gentile. The robustness of the p-norm algorithms. *Mach. Learn.*, 53(3): 265–299, 2003.

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

J. Langford, L. Li, and T. Zhang. Sparse online learning via truncated gradient. In *To appear in Advances in Neural Information Processing Systems 21*, 2008.

L. Lemel, R. Kassel, and S. Seneff. Speech database development: Design and analysis. Report no. SAIC-86/1546, Proc. DARPA Speech Recognition Workshop, 1986.

F. Orabona, C. Castellini, B. Caputo, J. Luo, and G. Sandini. Indoor place recognition using online independent support vector machines. In *Proc. of the British Machine Vision Conference 2007*, pages 1090–1099, 2007.

F. Rosenblatt. The Perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–407, 1958.

B. Schölkopf, R. Herbrich, A. Smola, and R. Williamson. A generalized representer theorem. In *Proc. of the 13th Conference on Computational Learning Theory*, 2000.

S. Shalev-Shwartz and Y. Singer. A new perspective on an old perceptron algorithm. In *Proc. of the 18th Conference on Learning Theory*, pages 264–278, 2005.

B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *Advances in Neural Information Processing Systems 17*, 2003.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004.

J. Weston, A. Bordes, and L. Bottou. Online (and offline) on an even tighter budget. In Robert G. Cowell and Zoubin Ghahramani, editors, *Proceedings of AISTATS 2005*, pages 413–420, 2005.

Initialize: $\mathcal{S}_0 = \emptyset,\ f_0 = \mathbf{0}$

For $t = 1, 2, \ldots$

Receive new instance $\mathbf{x}_t$

Predict $\hat{y}_t = \text{sign}(f_{t-1}(\mathbf{x}_t))$

Receive label y_t

If $y_t \neq \hat{y}_t$ (*prediction error*)

Set $f'_t = f_{t-1} + y_t k(\mathbf{x}_t, \cdot)$

Set $f''_t = P_{t-1} f'_t$

Set $\delta_t = f''_t - f'_t$

If $\|\delta_t\| \leq \eta$

$f_t = f''_t$

$\mathcal{S}_t = \mathcal{S}_{t-1}$

Else

$f_t = f'_t$

$\mathcal{S}_t = \mathcal{S}_{t-1} \cup \mathbf{x}_t$

Else If $y_t = \hat{y}_t$ and $y_t f_{t-1}(\mathbf{x}_t) \leq 1$ (*margin error*)

Set $\delta_t = P_{t-1} k(\mathbf{x}_t, \cdot) - k(\mathbf{x}_t, \cdot)$

If $\ell_1(f_{t-1}(\mathbf{x}_t), y_t) \geq \frac{\|\delta_t\|}{2\eta}$ (*check if the* $\mathbf{x}_t$ *can be projected onto* $\mathcal{H}_{t-1}$)

Set $\tau_t = \min \left\{ \frac{\ell_1(f_{t-1}(\mathbf{x}_t), y_t)}{\|P_{t-1} k(\mathbf{x}_t, \cdot)\|^2}, \frac{2\ell_1(f_{t-1}(\mathbf{x}_t), y_t) - \frac{\delta_t}{\eta}}{\|P_{t-1} k(\mathbf{x}_t, \cdot)\|^2}, 1 \right\}$

Set $f_t = f_{t-1} + y_t \tau_t P_{t-1} k(\mathbf{x}_t, \cdot)$

$\mathcal{S}_t = \mathcal{S}_{t-1}$

Else

$f_t = f_{t-1}$

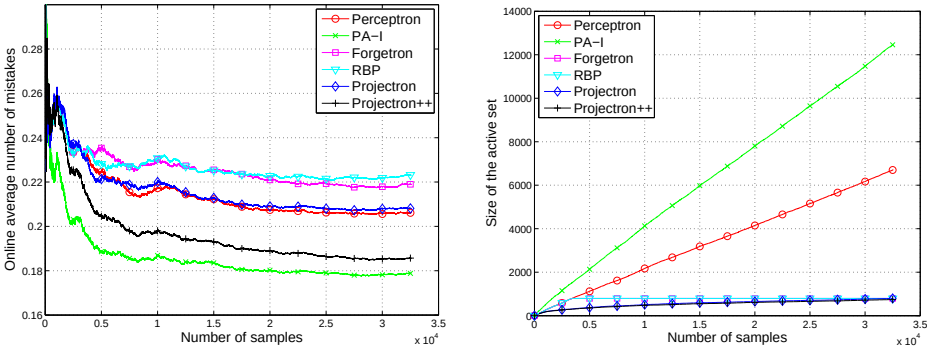
$\mathcal{S}_t = \mathcal{S}_{t-1}$

Else

$f_t = f_{t-1}$

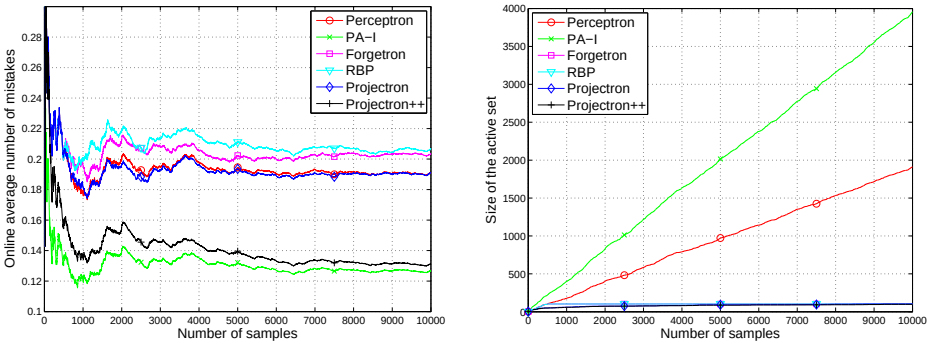
$\mathcal{S}_t = \mathcal{S}_{t-1}$

Figure 5: The Projectron++ Algorithm.



(a) (b)

Figure 6: Average online error (left) and size of the support set (right) for the different algorithms on *Adult9* dataset as a function of the number of training samples. B is set to 793, $\eta = 0.1$.



(a) (b)

Figure 7: Average online error (left) and size of the support set (right) for the different algorithms on the synthetic dataset as a function of the number of training samples. B is set to 103, $\eta = 0.04$.

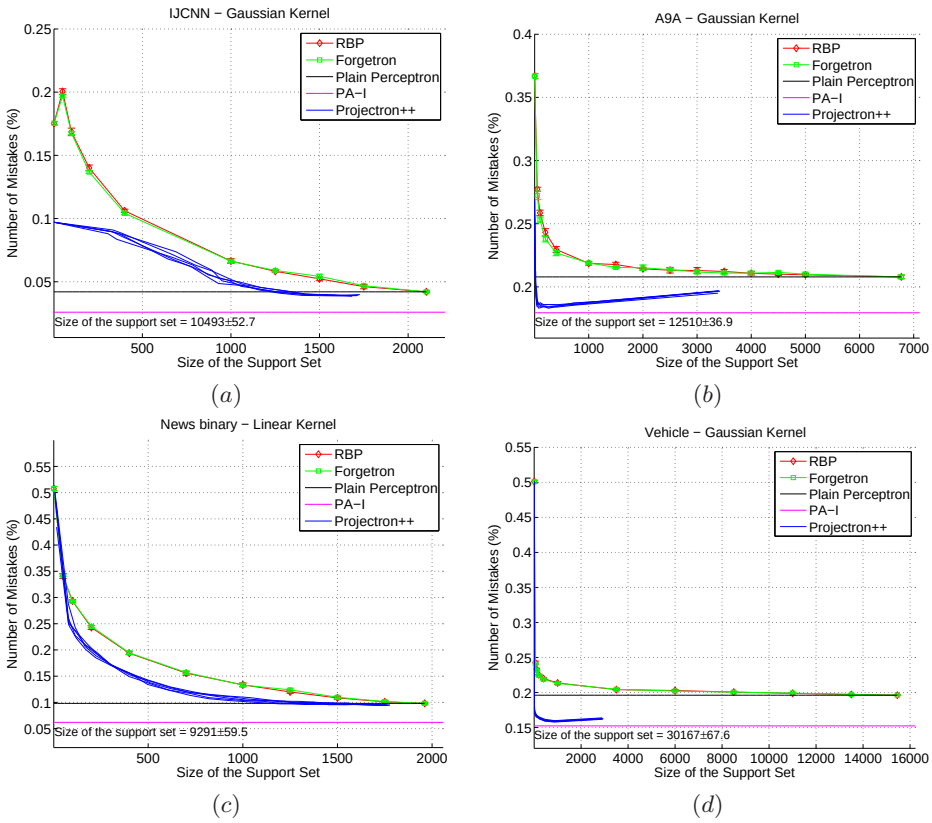
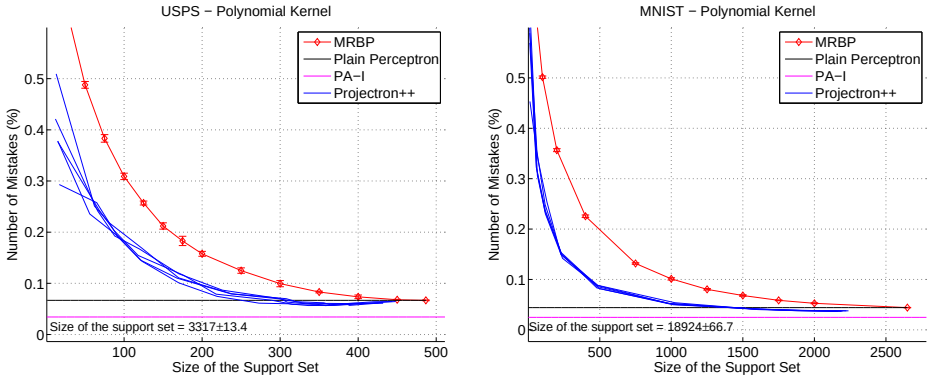
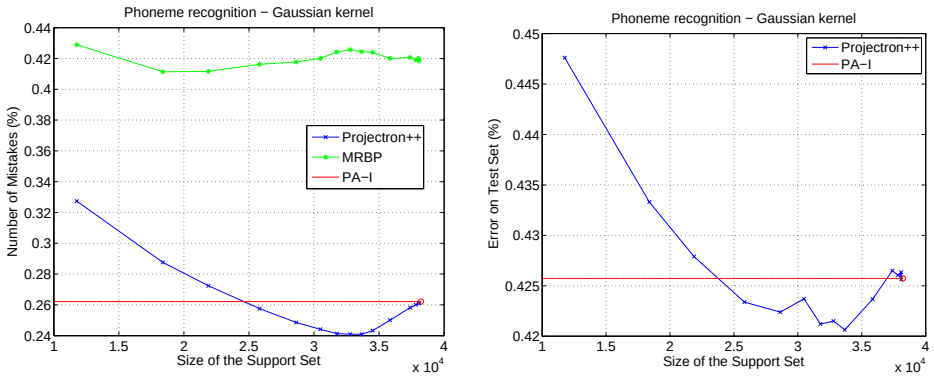


Figure 8: Average online error for the different algorithms as a function of the size of the support set on different binary datasets.



(a) (b)

Figure 9: Average online error for the different algorithms as a function of the size of the support set on different multiclass datasets.



(a) (b)

Figure 10: Average online error (a) and test error (b) for the different algorithms as a function of the size of the support set on a subset of the TIMIT dataset.