IDIAP RESEARCH REPORT

The Projectron: a Bounded Kernel-Based Perceptron

Francesco Orabona [a b] Joseph Keshet [a b]
Barbara Caputo [a b]

IDIAP–RR 08-30

May 2008

to appear in
Proceedings of the 25th International Conference on Machine Learning
(ICML 2008)

[a] IDIAP Research Institute, P.O.Box 592, 1920, Martigny, Switzerland
[b] Ecole Polytechnique Fdrale de Lausanne (EPFL), Switzerland

IDIAP Research Report 08-30

The Projectron: a Bounded Kernel-Based Perceptron

Francesco Orabona Joseph Keshet Barbara Caputo

May 2008

to appear in
Proceedings of the 25th International Conference on Machine Learning (ICML 2008)

Abstract. We present a discriminative online algorithm with a bounded memory growth, which is based on the kernel-based Perceptron. Generally, the required memory of the kernel-based Perceptron for storing the online hypothesis is not bounded. Previous work has been focused on discarding part of the instances in order to keep the memory bounded. In the proposed algorithm the instances are not discarded, but projected onto the space spanned by the previous online hypothesis. We derive a relative mistake bound and compare our algorithm both analytically and empirically to the state-of-the-art Forgetron algorithm (Dekel et al, 2007). The first variant of our algorithm, called Projectron, outperforms the Forgetron. The second variant, called Projectron++, outperforms even the Perceptron.

1 Introduction

One of the most important aspects of online learning methods is their ability to work in an open-ended fashion. Autonomous agents, for example, need to learn continuously from their surroundings, to adapt to the environment and maintain satisfactory performances. A recent stream of work on artificial cognitive systems have signaled the need for life-long learning methods and the promise of discriminative classifiers for this task (Orabona et al., 2007, and references therein).

Kernel-based discriminative online algorithms have been shown to perform very well on binary classification problems (see for example (Kivinen et al., 2004; Crammer et al., 2006)). Most of them can be seen as belonging to the Perceptron algorithm family. They construct their classification function incrementally, keeping a subset of the instances called *support set*. Each time an instance is misclassified it is added to the support set, and the classification function is defined as a kernel combination of the observations in this set. It is clear that if the problem is not linearly separable, they will never stop updating the classification function. This leads eventually to a memory explosion, and it concretely limits the usage of these methods for all those applications where data must be acquired continuously in time.

Several authors tried in the past to address this problem, mainly by bounding a priori the memory requirements. The first algorithm to overcome the unlimited growth of the support set was proposed by Crammer et al. (2003). The algorithm was then refined by Weston et al. (2005). The idea of the algorithm was to discard a vector of the solution, once the maximum dimension has been reached. The strategy was purely heuristic and no mistake bounds were given. A similar strategy has been used also in NORMA (Kivinen et al., 2004) and SILK (Cheng et al., 2007). The very first online algorithm to have a fixed memory "budget" and at the same time to have a relative mistake bound has been the Forgetron (Dekel et al., 2007). A stochastic algorithm that on average achieves similar performances, and with a similar mistake bound has been proposed by Cesa-Bianchi et al. (2006).

In this paper we take a different route. We modify the Perceptron algorithm so that the number of stored samples is always bounded. Instead of fixing a priori the maximum dimension of the solution, we introduce a parameter that can be tuned by the user, to trade accuracy for sparseness, depending on the needs of the task at hand. We call the algorithm, that constitutes the first contribution of this paper, *Projectron*. The Projectron is an online, Perceptron-like method that is bounded in space and in time complexity. We derive for it a mistake bound, and we show experimentally that it outperforms consistently the Forgetron algorithm. The second contribution of this paper is the derivation of a second algorithm, that we call *Projectron++*. It achieves better performances than the Perceptron, retaining all the advantage of the Projectron listed above. Note that this is opposite to previous budget online learning algorithms, delivering performances at most as good as the original Perceptron.

The rest of the paper is organized as follows: in Section 2 we state the problem and we introduce the necessary background theory. Section 3 introduces the Projectron, Section 4 derives its properties and Section 4.1 derives the Projectron++. We report experiments in Section 5, and we conclude the paper with an overall discussion.

2 Problem Setting

The basis of our study is the well known Perceptron algorithm (Rosenblatt, 1958). The Perceptron algorithm learns the mapping $f : \mathcal{X} \to \mathbb{R}$ based on a set of examples $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)\}$, 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 Perceptron as $\mathrm{sign}(f(\mathbf{x}))$ 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, $\langle k(\mathbf{x}, \cdot), f(\cdot) \rangle = f(\mathbf{x})$

Algorithm 1 Perceptron Algorithm

Initialize: $\mathcal{S}_0 = \emptyset$, $f_0 = \mathbf{0}$
for $t = 1, 2, \ldots, T$ **do**
  Receive new instance $\mathbf{x}_t$
  Predict $\hat{y}_t = \text{sign}(f_{t-1}(\mathbf{x}_t))$
  Receive label y_t
  if $y_t \neq \hat{y}_t$ **then**
    $f_t = f_{t-1} + y_t k(\mathbf{x}_t, \cdot)$
    $\mathcal{S}_t = \mathcal{S}_{t-1} \cup \{t\}$
  else
    $f_t = f_{t-1}$
    $\mathcal{S}_t = \mathcal{S}_{t-1}$
  end if
end for

The Perceptron algorithm is an online algorithm, in which the 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 $f_0 = \mathbf{0}$. On each round t, an instance $\mathbf{x}_t \in \mathcal{X}$ is presented to the algorithm. The algorithm predicts 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, $f_t = f_{t-1}$. Practically, the hypothesis f_t can be written as a kernel expansion (Schölkopf et al., 2000),

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

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

Although the Perceptron is a very simple algorithm, it is considered to produce very good results. Our goal is to derive and analyze a new algorithm which attains the same results as the Perceptron but with a minimal size of support set. In the next section we present our Projectron algorithm.

3 The Projectron Algorithm

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 the Perceptron algorithm 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, if the instance can be spanned by the support set, namely, $\mathbf{x}_t = \sum_{i=1}^{t-1} d_i \mathbf{x}_i$, it is not added to the support set. Instead, the coefficients $\{\alpha_i\}$ in the expansion Eq. (1) are not merely y_i, $i \in \mathcal{S}_{t-1}$, but they are changed to reflect the addition of this instance to the hypothesis, that is, $\alpha_i = y_i + y_t d_i$, $1 \le i \le t-1$. 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 in any way, and was used by Downs at al. (2001) to simplify Support Vector Machine 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

Algorithm 2 Projectron Algorithm

```
Initialize: S_0 = ∅, f_0 = 0
for t = 1, 2, ..., T do
  Receive new instance x_t
  Predict ŷ_t = sign(f_{t−1}(x_t))
  Receive label y_t
  if y_t ≠ ŷ_t then
    f'_t = f_{t−1} + y_t k(x_t, ·)
    f''_t = f'_t projected onto the space S_{t−1}
    δ_t = f''_t − f'_t
    if ‖δ_t‖ ≤ η then
      f_t = f''_t
      S_t = S_{t−1}
    else
      f_t = f'_t
      S_t = S_{t−1} ∪ {t}
    end if
  else
    f_t = f_{t−1}
    S_t = S_{t−1}
  end if
end for
```

independence with a finite number of vectors (Csató & Opper, 2001; Engel et al., 2002; Orabona et al., 2007). In particular 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. 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, which is the projection of f'_t onto the space spanned by $\mathcal{S}_{t-1}$. That is, the projected hypothesis is a hypothesis from the support set $\mathcal{S}_{t-1}$ which is the closest to the temporal 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 next section, 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 Algorithm 2.

In our algorithm the parameter η plays an important role. 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 will also derive mistake bounds to analyze the effect of η on the classification accuracy.

We now consider the problem of deriving the projected hypothesis f''_t in a Hilbert space $\mathcal{H}$, induced by a kernel function $k(\cdot, \cdot)$. Denote by $P_{t-1} f_t$ the projection of $f_t \in \mathcal{H}$ onto the subspace $\mathcal{H}_{t-1} \subset \mathcal{H}$ spanned by the set $\mathcal{S}_{t-1}$. The projected hypothesis f''_t is defined as $f''_t = P_{t-1} f'_t$. Expanding f'_t we have

$$f''_t = P_{t-1} f'_t = P_{t-1} \left(f_{t-1} + y_t k(\mathbf{x}_t, \cdot) \right) \ . \tag{2}$$

The projection is an idempotent ($P_{t-1}^2 = P_{t-1}$) and linear operator, hence,

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

Recall that $\delta_t = f_t'' - f_t'$. Substitute f_t'' from Eq. (3) and f_t' we have

$$\delta_t = f_t'' - f_t' = y_t P_{t-1} k(\mathbf{x}_t, \cdot) - y_t k(\mathbf{x}_t, \cdot) \; . \tag{4}$$

Recall that the projection of $f_t' \in \mathcal{H}$ onto a subspace $\mathcal{H}_{t-1} \subset \mathcal{H}$ is the hypothesis in $\mathcal{H}_{t-1}$ closest to f_t'. Hence, let $\sum_{j \in \mathcal{S}_{t-1}} d_j k(\mathbf{x}_j, \cdot)$ be an hypothesis in $\mathcal{H}_{t-1}$, where $(d_1, \ldots, d_{t-1})$ is a set of coefficients. The closest hypothesis is the one for which

$$\|\delta_t\|^2 = \min_{(d_1, \ldots, d_{t-1})} \left\| \sum_{j \in \mathcal{S}_{t-1}} d_j k(\mathbf{x}_j, \cdot) - k(\mathbf{x}_t, \cdot) \right\|^2 \; . \tag{5}$$

Expanding Eq. (5) we get

$$\|\delta_t\|^2 = \min_{(d_1, \ldots, d_{t-1})} \left(\sum_{i,j \in \mathcal{S}_{t-1}} d_j d_i k(\mathbf{x}_j, \mathbf{x}_i) - 2 \sum_{j \in \mathcal{S}_{t-1}} d_j k(\mathbf{x}_j, \mathbf{x}_t) + k(\mathbf{x}_t, \mathbf{x}_t) \right) \; . \tag{6}$$

Define $\mathbf{K}_{t-1}$ 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 $i, j \in \mathcal{S}_{t-1}$. Define $\mathbf{k}_t$ to be the vector whose i-th element is $\mathbf{k}_{t_i} = k(\mathbf{x}_i, \mathbf{x}_t)$. We have

$$\|\delta_t\|^2 = \min_{\mathbf{d}} \left(\mathbf{d}^T \mathbf{K}_{t-1} \mathbf{d} - 2\mathbf{d}^T \mathbf{k}_t + k(\mathbf{x}_t, \mathbf{x}_t) \right) \; , \tag{7}$$

where $\mathbf{d} = (d_1, \ldots, d_{t-1})^T$. Solving Eq. (7), that is, applying the extremum conditions with respect to $\mathbf{d}$, we obtain

$$\mathbf{d}^\star = \mathbf{K}_{t-1}^{-1} \mathbf{k}_t \tag{8}$$

and, by substituting Eq. (8) into Eq. (7),

$$\|\delta_t\|^2 = k(\mathbf{x}_t, \mathbf{x}_t) - \mathbf{k}_t^T \mathbf{d}^\star \; . \tag{9}$$

Furthermore, substituting Eq. (8) into Eq. (3) we get

$$f_t'' = f_{t-1} + y_t \sum_{j \in \mathcal{S}_{t-1}} \mathbf{d}_j^\star \, k(\mathbf{x}_j, \cdot) \; . \tag{10}$$

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. (8), plug the result either into Eq. (9) and obtain δ_t or into Eq. (10) and obtain the projected hypothesis.

In order to make the computation more tractable, we introduce an efficient method to calculate the matrix inversion $\mathbf{K}_t^{-1}$ iteratively. This method was first introduced in (Cauwenberghs & Poggio, 2000), and we give it here only for completeness. We would like to note in passing that the matrix $\mathbf{K}_{t-1}$ can be safely inverted since, by incremental construction, it is always full-rank. After the addition of a new sample, $\mathbf{K}_t^{-1}$ becomes

$$\left[\begin{array}{ccc|c} & & & 0 \\ & \mathbf{K}_{t-1}^{-1} & & \vdots \\ & & & 0 \\ 0 & \cdots & 0 & 0 \end{array} \right] + \frac{1}{\|\delta_t\|^2} \begin{bmatrix} \mathbf{d}^\star \\ -1 \end{bmatrix} \begin{bmatrix} \mathbf{d}^{\star T} & -1 \end{bmatrix} \tag{11}$$

where $\mathbf{d}^\star$ and $\|\delta_t\|^2$ are already evaluated during the previous steps of the algorithm. 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. (8).

4 Analysis

In this section we analyze the performance of the Projectron algorithm in the usual framework of online learning with a competitor. First, we present a theorem which states that the size of the support set is bounded.

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

The proof of this theorem goes along the same lines as the proof of Theorem 3.1 in (Engel et al., 2002), and we omit it for brevity. 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 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 best hypothesis $g \in \mathcal{H}$ chosen in hindsight. Let us define D_1 as

$$D_1 = \sum_{t=1}^{T} \ell(g(\mathbf{x}_t), y_t) \tag{12}$$

where $\ell(g(\mathbf{x}_t), y_t)$ is the hinge loss suffered by the function g on the example $(\mathbf{x}_t, y_t)$, that is, $\max\{0, 1 - y_t g(\mathbf{x}_t)\}$. With these definitions we can state the following bound for the Projectron Algorithm.

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 $k(\mathbf{x}_t, \mathbf{x}_t) \leq R$ for all t. Let g be an arbitrary function in $\mathcal{H}$. Assume that the Projectron algorithm is run with $0 \leq \eta < \frac{2-R^2}{2\|g\|}$. Then the number of prediction mistakes the Projectron makes on the sequence is at most*

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

The proof of this theorem is based on the following lemma.

Lemma 1. *Let $(\mathbf{x}, y)$ be an example, with $\mathbf{x} \in \mathcal{X}$ and $y \in \{+1, -1\}$. Denote by f an hypothesis in $\mathcal{H}$, such that $yf(\mathbf{x}) < 1$. Let $f' = f + \tau y q(\cdot)$, where $q(\cdot) \in \mathcal{H}$. Then the following bound holds for any $\tau \geq 0$:*

$$\begin{aligned}
&\|f - g\|^2 - \|f' - g\|^2 \geq \\
&\qquad \tau\big(2\ell(f(\mathbf{x}), y) - 2\ell(g(\mathbf{x}), y) - \tau\|q(\cdot)\|^2 - 2\langle f, q(\cdot) - k(\mathbf{x}, \cdot)\rangle - 2\|q(\cdot) - k(\mathbf{x}, \cdot)\| \cdot \|g\|\big)
\end{aligned}$$

Proof.

$$\begin{aligned}
&\|f - g\|^2 - \|f' - g\|^2 = 2\tau y\langle g - f, q(\cdot)\rangle - \tau^2\|q(\cdot)\|^2 \\
&= 2\tau y(g(x) - f(x)) - \tau^2\|q(\cdot)\|^2 + 2\tau y\langle g - f, q(\cdot) - k(x, \cdot)\rangle \\
&\geq \tau\big(2\ell(f(\mathbf{x}), y) - 2\ell(g(\mathbf{x}), y) - \tau\|q(\cdot)\|^2 - 2y\langle f, q(\cdot) - k(x, \cdot)\rangle - 2\|q(\cdot) - k(x, \cdot)\| \cdot \|g\|\big)
\end{aligned}$$

□

With this bound we are ready to prove Thm. 2.

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 in which there is no mistake Δ_t is 0. On rounds in which 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 bound the progress from below, when the update is of the former type (the same bound can be obtained for the latter type as well, but the derivation is omitted). 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. (4)

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

Note that $\langle f_{t-1}, \delta_t \rangle = 0$, because f_{t-1} belongs to the space spanned by the functions indexed by $\mathcal{S}_{t-1}$. Moreover, on every projection update $\|\delta_t\| \leq \eta$ and using the theorem assumption $\|P_{t-1}k(\mathbf{x}_t, \cdot)\| \leq R$, we then have

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

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

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

We sum over t 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

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

where M is the number of mistakes. □

To compare with other similar algorithms it can be useful to change the formulation of the algorithm in order to use the maximum norm of g as parameter instead of η. Hence we can fix an upper bound, U, on $\|g\|$ and then we set η to have a positive progress. Specifically, on each round we set η to be

$$\frac{1}{2U} \Big(2\ell(f_{t-1}(\mathbf{x}_t), y_t) - \|P_{t-1}k(\mathbf{x}_t, \cdot)\|^2 - 0.5 \Big) \ . \tag{13}$$

The next corollary, based on Thm. 2, provides a mistake bounds in terms of U rather than η.

Corollary 1. *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 $\mathcal{H}$, whose norm $\|g\|$ is bounded by U. Assume that the Projectron algorithm is run with a parameter η, which is set in each round according to Eq. (13). Then, the number of prediction mistakes the Projectron makes on the sequence is at most*

$$2\|g\|^2 + 4D_1 \ .$$

Notice that the bound in Corollary 1 is similar to Thm. 5.1 in (Dekel et al., 2007) of the Forgetron algorithm. The difference is in the assumptions made: in the Forgetron, the size of the support set is guaranteed to be less than a fixed size B that depends on U, while in the Projectron we choose the value of η or, equivalently, U, and there is no guarantee on the exact size of the support set. However, the experimental results suggest that, with the same assumptions used in the derivation of the Forgetron bound, the Projectron needs a smaller support set and produces less mistakes.

It is also possible to give yet another bound by slightly changing the proof of Thm. 2. This theorem is a worst-case mistake bound for the Projectron algorithm. We state it here without the proof, leaving it for a long version of this paper.

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 $k(\mathbf{x}_t, \mathbf{x}_t) \leq R$ for all t. Let g an arbitrary function in $\mathcal{H}$. Assume that the Projectron algorithm is run with $0 \leq \eta < \frac{1}{\|g\|}$. Then, M, the number of prediction mistakes the Projectron makes on the sequence is at most*

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

The last theorem suggests that the performance of the Projectron are slightly worse than the Perceptron (Shalev-Shwartz & Singer, 2005). Specifically the degradation in the performance of Projectron compared to the Perceptron are related to $1/(1-\eta\|g\|)^2$. In the next subsection we present a variant to the Projectron algorithm, which attains even better performance.

4.1 Going Beyond the Perceptron

The proof of Thm. 2 and Corollary 1 direct us how to improve the Projectron algorithm to go beyond the performance of the Perceptron algorithm, while maintaining a bounded support set.

Let us start from the algorithm in Corollary 1. We change it so an update takes place not only if there is a prediction mistake, but also when the prediction is correct with a low confidence. We indicate this latter case as a *margin error*, that is, $0 < y_t f_{t-1}(\mathbf{x}_t) < 1$. This strategy improves 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 every round a margin error occurs, but also when the new instance *can be projected* onto the support set. Hence, the update on margin error rounds would be in the general form

$$f_t = f_{t-1} + y_t \tau_t P_{t-1} k(\mathbf{x}_t, \cdot) \ , \tag{14}$$

with $0 < \tau_t \leq 1$. The last constraint comes from proofs of Thm. 2 and Corollary 1 in which we upper bound τ_t by 1. Note that setting τ_t to 0 is equivalent to leave the hypothesis unchanged. The bound in Corollary 1 becomes

$$M \leq 2(\|g\|^2 + 2D_1 - \sum_{\{t:0<y_t f_{t-1}(\mathbf{x}_t)<1\}} \beta_t) \ , \tag{15}$$

where β_t bounds the progress made on margin error round t. In particular it is easy to see from Lemma 1 that β_t is

$$\tau_t\big(2\ell(f_{t-1}(\mathbf{x}_t), y_t) - \tau_t \|P_{t-1} k(\mathbf{x}_t, \cdot)\|^2 - 2U\|\delta_t\|\big) \ , \tag{16}$$

for $0 < \tau_t \leq 1$, and is 0 when there is no update. Whenever β_t is non-negative the worst-case number of mistakes in Eq. (15) decreases, hopefully along with the classification error rate of the algorithm. Hence, we determine the optimal τ_t which maximizes β_t. In particular, the expression of β_t in Eq. (16) is quadratic in τ_t, and is maximized for $\tau_t = \ell(f_{t-1}(\mathbf{x}_t), y_t)/\|P_{t-1}k(\mathbf{x}_t, \cdot)\|^2$. Constraining τ_t to be less than or equal to 1, we have[1]

$$\tau_t = \min\{\ell(f_{t-1}(\mathbf{x}_t), y_t)/\|P_{t-1}k(\mathbf{x}_t, \cdot)\|^2, 1\} \ . \tag{17}$$

In summary, at every round t with margin error we calculate τ_t according to Eq. (17), and check that β_t is non-negative. If so we update the hypothesis using Eq. (14), otherwise we leave it untouched.

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 variant Projectron++, and in the next section we compare it to the original version.

5 Experimental Results

In this section we present experimental results that demonstrate the effectiveness of the Projectron and the Projectron++. We compare both algorithms to the Perceptron and to the budget algorithms Forgetron (Dekel et al., 2007) and Randomized Budget Perceptron (RBP) (Cesa-Bianchi et al., 2006). For the Forgetron, we choose the state-of-the-art "self-tuned" variant, which outperforms its other variants. We also use two other baseline algorithms: the first one is a Perceptron algorithm which stops updating the solution once the support size has reached some limit, and it is used to verify that

[1] This update rule gives $\tau_t = 1$ on rounds in which there is a mistake.

the Projectron is better than just stop learning. We name it *Stoptron*. The second baseline algorithm is the PA-I variant of the Passive-Aggressive learning algorithm (Crammer et al., 2006), which gives an upper bound to the classification performance that Projectron++ can reach.

We tested the algorithms with two standard machine learning datasets: *Adults9* and *Vehicle*[2] and a synthetic dataset, all of them with more than 10000 samples. The synthetic dataset is built in the same way as in (Dekel et al., 2007). It is composed with 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.

All the experiments were performed over 5 different permutations of the training set. All algorithms used a Gaussian kernel with σ^2 equals 25, 4, and 0.5 for *Adults9*, *Vehicle*, and the synthetic datasets, respectively. The C parameter of the PA-I was set to 1, to have an update similar to the Perceptron and Projectron. Due to the different nature of our algorithm compared to the budget ones, we cannot select the support set size in hindsight. Hence, we compared them using the proper conditions to obtain the same bounds. That is, we selected the maximum support size B for the Forgetron algorithm, which implies a maximum value U, the norm of g, for its bound to hold. In particular U is equal to $1/4\sqrt{(B + 1)/ \log (B + 1)}$ (Dekel et al., 2007), where B is the budget parameter that sets the maximum size of the support set. We then selected the parameter η in the Projectron in each round according to Eq. (13). Hence the final size of the Projectron solution will depend on U and on the particular classification problem at hand. We have set B on each dataset roughly to $1/2$ and $1/4$ of the size of the Perceptron support set, for a total of 6 experiments. Note that Projectron can also be used without taking into account the norm of the competitor and considering η just as a parameter. In particular η should be set to trade accuracy for sparseness.

In Tables 1–3 we summarize the results of our experiments. The cumulative number of mistakes as percentage of the training size (mean $\pm$ std) and the size of the support set are reported. In all the experiments both the Projectron and the Projectron++ outperform the Forgetron and the RBP with a smaller support size. Moreover, the Projectron++ always outperforms the Projectron and has smaller support set. Due to its theoretically derived formulation, it achieves better results even if being bounded, and it has better performance than the Perceptron. In particular it gets closer to the classification rate of the PA-I, without paying the price of a large support set. It is interesting to note the performances of the Stoptron: it has an accuracy close to the other bounded algorithms in average, but with much bigger variance. This indicates that all the examined strategies for bounded learning are always better than the simple procedure to stop learning, at least to have stable performances.

Last, we show the behavior of the algorithms over time. In Fig. 1 we show the average online error rate, that is, the total numbers of errors on the examples seen as a function of the number of samples for all algorithms on the Adult9 dataset with $B = 1500$. Note how the Projectron algorithm closely tracks the Perceptron. On the other hand the Forgetron and the RBP stop improving after reaching the support set size B, around 7500 samples. The growth of the support set as a function of the number of samples is depicted in Fig. 2. While for PA-I and Perceptron the growth is clearly linear, it is sub-linear for Projectron and for the Projectron++ and they will reach a maximum size and then they will stop growing (as stated in Thm. 1). In Fig. 3 we show the average online error rate as a function of the size of the support set. It is clear that the Projectron and the Projectron++ outperform the Perceptron with smaller support set.

6 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, therefore it solves the memory explosion problem of the Perceptron and similar algorithms. Although the size of the support set is guaranteed to be bounded,

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

Table 1: Adult9 dataset, 32561 samples.

Algorithm	% Mistakes	Size Support Set
Perceptron	$20.99\% \pm 0.06$	6835.6 ± 20.28
PA-I	$18.11\% \pm 0.10$	12537 ± 36.2
	B=1500	
Projectron	$20.95\% \pm 0.12$	1094.6 ± 16.06
Projectron++	$\mathbf{20.04\% \pm 0.14}$	$\mathbf{992.8 \pm 9.73}$
Forgetron	$21.90\% \pm 0.23$	1500
RBP	$22.05\% \pm 0.21$	1500
Stoptron	$22.73\% \pm 2.82$	1500
	B=3000	
Projectron	$20.97\% \pm 0.13$	1499.6 ± 13.58
Projectron++	$\mathbf{20.16\% \pm 0.11}$	$\mathbf{1364.2 \pm 4.76}$
Forgetron	$21.41\% \pm 0.13$	3000
RBP	$21.49\% \pm 0.11$	3000
Stoptron	$21.04\% \pm 1.54$	3000

Table 2: Vehicle dataset, 78823 samples.

Algorithm	% Mistakes	Size Support Set
Perceptron	$19.58\% \pm 0.09$	15432.0 ± 69.62
PA-I	$15.27\% \pm 0.05$	30131.4 ± 21.07
	B=4000	
Projectron	$19.63\% \pm 0.08$	3496.4 ± 18.39
Projectron++	$\mathbf{18.27\% \pm 0.06}$	$\mathbf{3187.0 \pm 13.64}$
Forgetron	$20.40\% \pm 0.04$	4000
RBP	$20.32\% \pm 0.04$	4000
Stoptron	$19.49\% \pm 3.56$	4000
	B=8000	
Projectron	$19.62\% \pm 0.04$	4668.2 ± 32.88
Projectron++	$\mathbf{18.53\% \pm 0.07}$	$\mathbf{4309.6 \pm 28.67}$
Forgetron	$19.98\% \pm 0.06$	8000
RBP	$19.94\% \pm 0.06$	8000
Stoptron	$20.17\% \pm 2.03$	8000

the actual size of the support set cannot be determined in advance, like in the Forgetron algorithm, and it is not fixed. Practically, the size of the support set of the Projectron algorithms is much smaller than that of the budget algorithms.

Compared to budget algorithms it has the advantage of a bounded support set size without removing or scaling instances in the set. This keeps performance high. We call this algorithm Projectron. Its second variant, the Projectron++, always outperforms the standard Perceptron algorithm, while assuring a bounded solution. Another advantage over budget algorithms is the possibility to obtain bounded batch solutions using standard online-to-batch conversion. In fact using the averaging conversion (Cesa-Bianchi et al., 2004) we get a bounded solution. This is not true for budget algorithms, where more sophisticated techniques have to be used (Dekel & Singer, 2005). A similar approach has been used in (Csató & Opper, 2001) in the framework of the Gaussian Processes. However in that paper no mistake bounds were derived and the use of the hinge loss allows us to have sparser solution.

Acknowledgments. This work was supported by EU project DIRAC (FP6-0027787).

References

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

Table 3: Synthetic dataset, 10000 samples.

Algorithm	% Mistakes	Size Support Set
Perceptron	18.80% $\pm$ 0.25	1880.0 $\pm$ 25.12
PA-I	12.58% $\pm$ 0.05	3986.8 $\pm$ 42.83
	B=1000	
Projectron	18.71% $\pm$ 0.14	108.6 $\pm$ 2.97
Projectron++	**14.09% $\pm$ 0.10**	**104.2 $\pm$ 2.39**
Forgetron	18.96% $\pm$ 0.32	1000
RBP	18.86% $\pm$ 0.29	1000
Stoptron	17.49% $\pm$ 1.77	1000
	B=500	
Projectron	18.70% $\pm$ 0.21	**98.6 $\pm$ 3.05**
Projectron++	**14.23% $\pm$ 0.10**	**98.6 $\pm$ 2.30**
Forgetron	19.20% $\pm$ 0.19	500
RBP	19.27% $\pm$ 0.20	500
Stoptron	21.96% $\pm$ 4.62	500

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

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

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

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

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

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

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

Dekel, O., & Singer, Y. (2005). Data-driven online to batch conversions. *Advances in Neural Information Processing Systems 18*.

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

Engel, Y., Mannor, S., & Meir, R. (2002). Sparse online greedy support vector regression. *Proceedings 13th European Conference on Machine Learning*.

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

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

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

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

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

Weston, J., Bordes, A., & Bottou, L. (2005). Online (and offline) on an even tighter budget. *Proceedings of AISTATS 2005* (pp. 413–420).

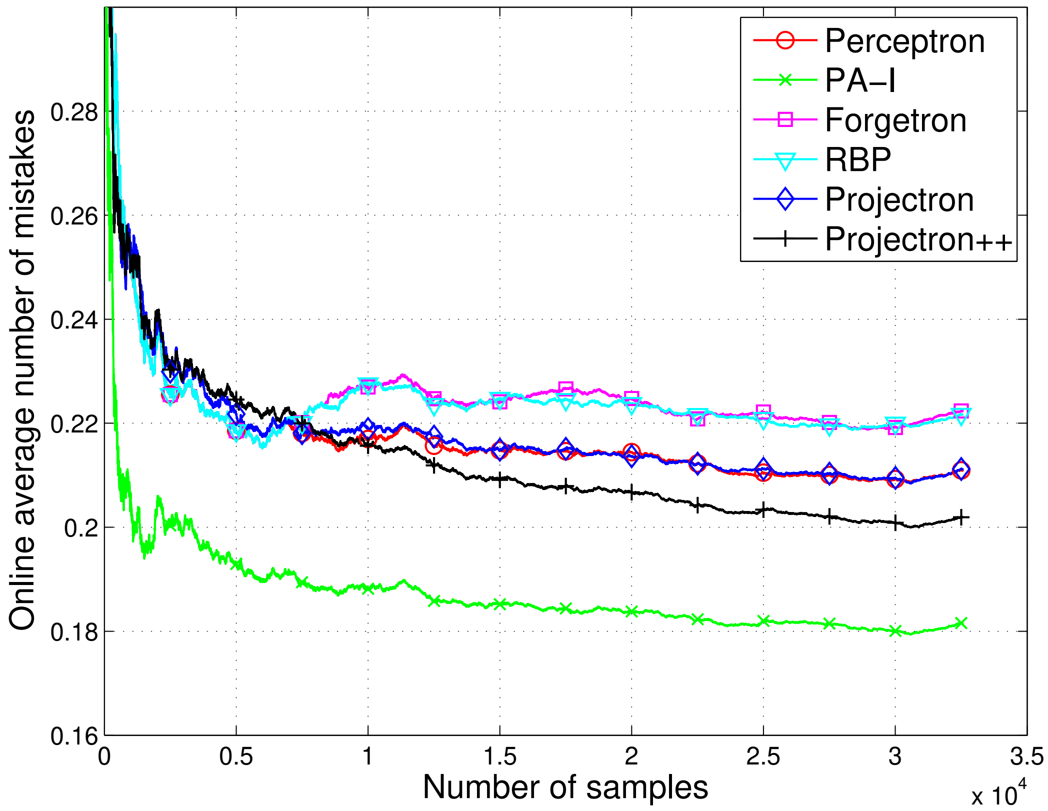

Figure 1: Average online error for the different algorithms on *Adult9* dataset as a function of the number of training samples. B is set to 1500.

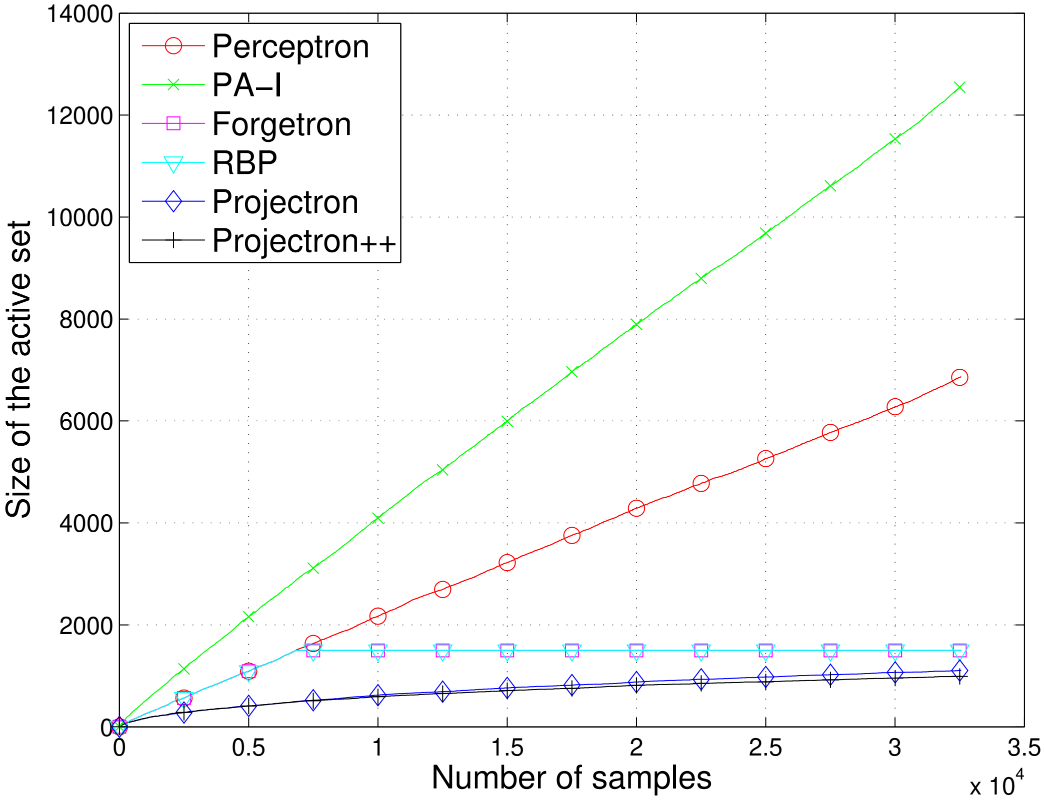

Figure 2: Size of the support set for the different algorithms on *Adult9* dataset as a function of the number of training samples. B is set to 1500.

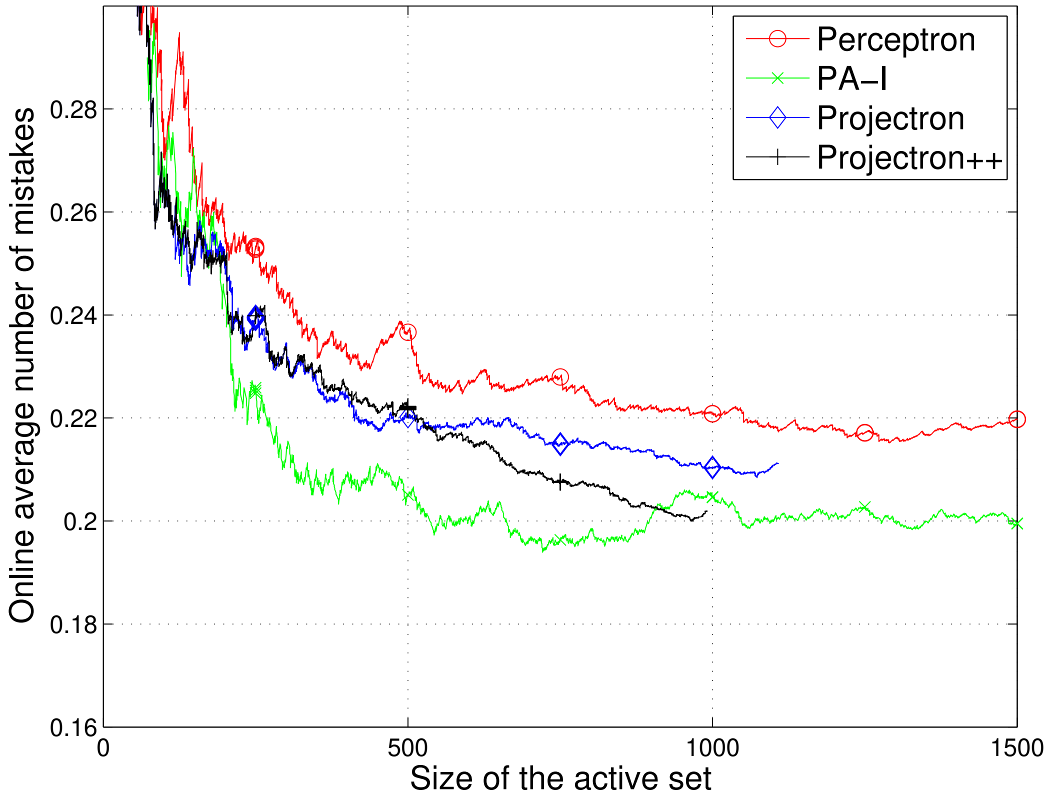

Figure 3: Average online error for the different algorithms on *Adult9* dataset as a function of the size of the support set. B is set to 1500.