The Forgetron: A Kernel-Based Perceptron on a Fixed Budget

Ofer Dekel Shai Shalev-Shwartz Yoram Singer
School of Computer Science & Engineering
The Hebrew University, Jerusalem 91904, Israel
{oferd,shais,singer}@cs.huji.ac.il

Abstract

The Perceptron algorithm, despite its simplicity, often performs well on online classification problems. The Perceptron becomes especially effective when it is used in conjunction with kernels. However, a common difficulty encountered when implementing kernel-based online algorithms is the amount of memory required to store the online hypothesis, which may grow unboundedly. In this paper we describe and analyze a new infrastructure for kernel-based learning with the Perceptron while adhering to a strict limit on the number of examples that can be stored. We first describe a template algorithm, called the Forgetron, for online learning on a fixed budget. We then provide specific algorithms and derive a unified mistake bound for all of them. To our knowledge, this is the first online learning paradigm which, on one hand, maintains a *strict* limit on the number of examples it can store and, on the other hand, entertains a relative mistake bound. We also present experiments with real datasets which underscore the merits of our approach.

1 Introduction

The introduction of the Support Vector Machine (SVM) [8] sparked a widespread interest in kernel methods as a means of solving (binary) classification problems. Although SVM was initially stated as a batch-learning technique, it significantly influenced the development of kernel methods in the online-learning setting. Online classification algorithms that can incorporate kernels include the Perceptron [7], ROMMA [6], ALMA [4], NORMA [5] and the Passive-Aggressive family of algorithms [1]. Each of these algorithms observes examples in a sequence of rounds, and constructs its classification function incrementally by storing a subset of the observed examples in its internal memory. The classification function is then defined by a kernel-dependent combination of the stored examples. This set of stored examples is the online equivalent of the *support set* in SVMs, however in contrast to the support it continually changes as learning progresses. In this paper, we call this set the *active set*, as it includes those examples that actively define the current classifier. Typically, an example is added to the active set every time the online algorithm makes a prediction mistake or when its confidence in a prediction is inadequately low. A rapid growth of the active set can lead to significant computational difficulties. Naturally, since computing devices have bounded memory resources, there is the danger that an online al-

gorithm would require more memory than is physically available. This problem becomes especially eminent in cases where the online algorithm is implemented as part of a specialized hardware system with a small memory, such as a mobile telephone or an autonomous robot. Moreover, the growth of the active set can lead to unacceptably long running times, as the time-complexity of each online round scales linearly with the size of the active set.

Crammer, Kandola, and Singer [2] first addressed this problem by describing an online kernel-based modification of the Perceptron algorithm in which the active set does not exceed a predefined *budget*. Their algorithm removes redundant examples from the active set so as to make the best use of the limited memory resource. Weston, Bordes and Bottou [9] followed with their own online kernel machine on a budget. Both techniques work relatively well in practice, however they both lack a theoretical guarantee on prediction accuracy. In this paper we present online kernel-based classifiers which are restricted to a budget of active examples and for which we can give a formal learning-theoretic analysis. To the best of our knowledge, these are the first online algorithms with a fixed predetermined budget which also entertain formal worst-case mistake bounds. Some of our proofs are omitted due to the lack of space. Detailed proofs of all of the theorems stated in this paper can be found in [3].

This paper is organized as follows. In Sec. 2 we begin with a more formal presentation of our problem and discuss some difficulties in proving mistake bounds for kernel-methods on a budget. In Sec. 3 we present an algorithmic framework for online prediction with a predefined budget of active examples. Then in Sec. 4 we derive various algorithms from this framework. We name our algorithmic framework the *Forgetron*, since its update builds on that of the Perceptron, and since it gradually forgets active examples. Finally, we present an empirical evaluation of our algorithms in Sec. 5.

2 Problem Setting

Online learning is performed in a sequence of consecutive rounds, where on round t the online algorithm observes an instance $\mathbf{x}_t$, which is drawn from some predefined instance domain $\mathcal{X}$. The algorithm predicts the binary label associated with that instance and is then given the correct label $y_t \in \{-1, +1\}$. At this point, the algorithm may use the instance-label pair $(\mathbf{x}_t, y_t)$ to improve its prediction mechanism. The goal of the algorithm is to correctly predict as many labels as possible.

The predictions of the online algorithm are determined by a function which is stored in its internal memory and is updated from round to round. We refer to this function as the *hypothesis* of the algorithm and denote the hypothesis used on round t by f_t. Our focus in this paper is on margin based hypotheses, namely, f_t is a function from $\mathcal{X}$ to $\mathbb{R}$ where $\text{sign}(f_t(\mathbf{x}_t))$ constitutes the actual binary prediction and $|f_t(\mathbf{x}_t)|$ is the confidence in this prediction. The term $y f(\mathbf{x})$ is called the *margin* of the prediction and is positive whenever y and $\text{sign}(f(\mathbf{x}))$ agree. We can evaluate the performance of an hypothesis on a given example $(\mathbf{x}, y)$ in one of two ways. First, we can check whether the hypothesis makes a prediction mistake, namely determine if $y = \text{sign}(f(\mathbf{x}))$. Throughout this paper, we use M to denote the number of prediction mistakes made by an online algorithm on a sequence of examples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$. The second way to evaluate the predictions of an hypothesis is by using the *hinge-loss* function, defined by,

$$\ell\big(f; (\mathbf{x}, y)\big) \;=\; \begin{cases} 0 & \text{if } \; y f(\mathbf{x}) \geq 1 \\ 1 - y f(\mathbf{x}) & \text{otherwise} \end{cases} \quad . \tag{1}$$

The hinge-loss penalizes an hypothesis for any margin less than 1. Additionally, if $y \neq \text{sign}(f(\mathbf{x}))$ then $\ell(f, (\mathbf{x}, y)) \geq 1$ and therefore the *cumulative hinge-loss* suffered over a sequence of examples upper bounds M. The algorithms discussed in this paper use kernel-based hypotheses that are defined with respect to a kernel operator $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ which

adheres to Mercer's positivity conditions [8]. A kernel-based hypothesis takes the form,

$$f(\mathbf{x}) = \sum_{i=1}^{k} \alpha_i K(\mathbf{x}_i, \mathbf{x}) \ , \tag{2}$$

where $\mathbf{x}_1, \ldots, \mathbf{x}_k$ are members of $\mathcal{X}$ and $\alpha_1, \ldots, \alpha_k$ are real weights. To facilitate the derivation of our algorithms and their analysis, we associate a reproducing kernel Hilbert space (RKHS) with K in the standard way common to all kernel methods. Formally, let $\mathcal{H}_K$ be the closure of the set of all hypotheses of the form given in Eq. (2). For any two functions, $f(\mathbf{x}) = \sum_{i=1}^{k} \alpha_i K(\mathbf{x}_i, \mathbf{x})$ and $g(\mathbf{x}) = \sum_{j=1}^{l} \beta_j K(\mathbf{z}_j, \mathbf{x})$, define the inner product between them to be, $\langle f, g \rangle = \sum_{i=1}^{k} \sum_{j=1}^{l} \alpha_i \beta_j K(\mathbf{x}_i, \mathbf{z}_j)$. This inner-product naturally induces a norm defined by $\|f\| = \langle f, f \rangle^{1/2}$ and a metric $\|f - g\| = (\langle f, f \rangle - 2\langle f, g \rangle + \langle g, g \rangle)^{1/2}$. These definitions play an important role in the analysis of our algorithms. Online kernel methods typically restrict themselves to hypotheses that are defined by some subset of the examples observed on previous rounds. That is, the hypothesis used on round t takes the form, $f_t(\mathbf{x}) = \sum_{i \in I_t} \alpha_i K(\mathbf{x}_i, \mathbf{x})$, where I_t is some subset of $1, \ldots, (t\text{-}1)$ and $\mathbf{x}_i$ is the example observed by the algorithm on round i. As stated above, I_t is called the active set, and we say that $\mathbf{x}_i$ is *active* on round t if $i \in I_t$.

Perhaps the most well known online algorithm for binary classification is the Perceptron [7]. Stated in the form of a kernel method, the hypotheses generated by the Perceptron take the form $f_t(\mathbf{x}) = \sum_{i \in I_t} y_i K(\mathbf{x}_i, \mathbf{x})$. Namely, the weight assigned to each active example is either $+1$ or -1, depending on the label of that example. The Perceptron initializes I_1 to be the empty set, which implicitly sets f_1 to be the zero function. It then updates its hypothesis only on rounds where a prediction mistake is made. Concretely, on round t, if $f_t(\mathbf{x}_t) \neq y_t$ then the index t is inserted into the active set. As a consequence, the size of the active set on round t equals the number of prediction mistakes made on previous rounds. A relative mistake bound can be proven for the Perceptron algorithm. The bound holds for any sequence of instance-label pairs, and compares the number of mistakes made by the Perceptron with the cumulative hinge-loss of any fixed hypothesis $g \in \mathcal{H}_K$, even one defined with prior knowledge of the sequence.

Theorem 1. *Let K be a Mercer kernel and let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ be a sequence of examples such that $K(\mathbf{x}_t, \mathbf{x}_t) \leq 1$ for all t. Let g be an arbitrary function in $\mathcal{H}_K$ and define $\hat{\ell}_t = \ell\big(g; (\mathbf{x}_t, y_t)\big)$. Then the number of prediction mistakes made by the Perceptron on this sequence is bounded by, $M \leq \|g\|^2 + 2 \sum_{t=1}^{T} \hat{\ell}_t$.*

The proof can be found in [3]. Although the Perceptron is guaranteed to be competitive with any fixed hypothesis $g \in \mathcal{H}_K$, the fact that its active set can grow without a bound poses a serious computational problem, as noted in the previous section. In fact, this problem is common to most kernel-based online methods that do not explicitly monitor the size of I.

As discussed above, our goal is to derive and analyze an online prediction algorithm which resolves these problems by enforcing a *fixed* bound on the size of the active set. Formally, let B be a positive integer which we refer to as the *budget parameter*. We would like to devise an algorithm which enforces $|I_t| \leq B$ on every round t. Furthermore, we would like to prove a relative mistake bound for this algorithm analogous to the bound stated in Thm. 1. Regretfully, this goal turns out to be impossible without making additional assumptions. Concretely, for any kernel-based algorithm which is restricted by $|I_t| \leq B$, we can find an hypotheses $g \in \mathcal{H}_K$ and an arbitrarily long sequence of examples such that the algorithm makes a prediction mistake on every single round whereas g suffers no loss at all. We show this inherent limitation by presenting a simple counterexample which applies to any online algorithm which uses a prediction function of the form given in Eq. (2), and for which $|I_t| \leq B$ for all t. In our example, we choose the instance space $\mathcal{X}$ to be the set

of $B+1$ standard unit vectors in $\mathbb{R}^{B+1}$, namely $\mathcal{X} = \{e_i\}_{i=1}^{B+1}$ where e_i is the vector with 1 in its i'th coordinate and zeros elsewhere. K is set to be the standard dot product in $\mathbb{R}^{B+1}$, that is $K(\mathbf{x}, \mathbf{x}') = \mathbf{x} \cdot \mathbf{x}'$. Now for every t, f_t is a linear combination of at most B vectors from $\mathcal{X}$. Since $|\mathcal{X}| = B + 1$, there exists a vector $\mathbf{x}_t \in \mathcal{X}$ which is not currently active. Furthermore, $\mathbf{x}_t$ is orthogonal to all of the active vectors and therefore $f_t(\mathbf{x}_t) = 0$. Assume without loss of generality that the online algorithm we are using predicts y_t to be -1 when $f_t(\mathbf{x}) = 0$. If on every round we were to present the online algorithm with the example $(\mathbf{x}_t, +1)$ then the online algorithm would make a prediction mistake on every round. On the other hand, the hypothesis $\bar{g} = \sum_{i=1}^{B+1} e_i$ is a member of H_K and attains a hinge-loss of 0 on every round. We have found a sequence of examples and a fixed hypothesis (which is indeed defined by more than B vectors from $\mathcal{X}$) that attains a cumulative loss of zero on this sequence, while the number of mistakes made by our online algorithm equals the number of rounds. Clearly, a theorem along the lines of Thm. 1 cannot be proven.

One way to resolve this problem is to limit the set of competing hypotheses to a subset of $\mathcal{H}_K$, which would naturally exclude $\bar{g}$. In this paper, we limit the set of competitors to hypotheses with small norms. Formally, we wish to devise an online algorithm which is competitive with every hypothesis $g \in \mathcal{H}_K$ for which $\|g\| \leq U$, for some constant U. Our counterexample indicates that we cannot prove a relative mistake bound with U set to $\sqrt{B+1}$ or greater, since that was the norm of $\bar{g}$ in our counterexample. However, in this paper we come close to this upper bound by proving that our algorithms can compete with any hypothesis with a norm bounded by $\frac{1}{4}\sqrt{B/\log(B)}$.

3 Forgetron: A Template Algorithm

In this section we present a general framework for online prediction which builds upon the basic Perceptron algorithm. We intentionally leave some important details unspecified in this section and defer their definition to a later section. This strategy will make our framework general and will enable us to derive various different online algorithms from it. Furthermore, the general framework also enables us to provide a unified analysis of all of the algorithms presented in this paper. The pseudo-code in Fig. 1 defines the generic skeleton of any algorithm that fits in our framework.

The online hypothesis used on round t is defined by $f_t(\mathbf{x}) = \sum_{i \in I_t} \sigma_{i,t\text{-}1}\, y_i K(\mathbf{x}_i, \mathbf{x})$, where $\sigma_{i,t\text{-}1}$ are weights already defined from previous rounds. The algorithm starts round t by outputting the prediction $\text{sign}(f_t(\mathbf{x}_t))$. It then receives the correct label y_t and modifies the current hypothesis f_t only if $\text{sign}(f_t(\mathbf{x}_t)) \neq y_t$. Step (1) of the update is simply the Perceptron update, that is, the current example is added to the active set. We denote the resulting active set by I'_t, and the hypothesis obtained after the application of step (1) by f'_t, namely, $f'_t(\mathbf{x}) = f_t(\mathbf{x}) + y_t K(\mathbf{x}_t, \mathbf{x})$. Step (2) of the update is a rescaling step, where the updated hypothesis f'_t is multiplied by a scalar $\phi_t \in (0, 1)$. The specific choice of ϕ_t is provided in Sec. 4, where we derive three specific algorithms from our general framework. We use f''_t to denote the scaled version of f'_t. Since the hypothesis is represented as a weighted sum of kernel functions, multiplying f' by ϕ_t is achieved by adjusting the weights of these functions, setting $\sigma_{i,t} = \phi_t \sigma_{i,t\text{-}1}$. The idea behind decreasing the weights of the active examples is to lessen the influence of older active examples on the current hypothesis. Thus, examples that have been active for a long time can be removed from the active set without drastically changing the current hypothesis. Finally, on step (3) the active set is modified so that its size will not exceed the budget B. If the current number of active examples is less than the budget B, then we can simply set I_{t+1} to be I'_t. Otherwise, the algorithm has already reached its budget and some index $r_t \in I'_t$ must be removed. As with ϕ_t, the specifics of how to choose r_t are discussed in Sec. 4. In general, r_t is an example whose weight has decreased enough, such that its influence on the hypothesis is negligible.

INPUT: Mercer kernel $K(\cdot,\cdot)$; budget $B > 0$

INITIALIZE: $I_1 = \emptyset$; $f_1 \equiv 0$

For $t = 1, 2, \ldots$

receive an instance $\mathbf{x}_t$ s.t. $K(\mathbf{x}_t, \mathbf{x}_t) \leq 1$

predict sign$(f_t(\mathbf{x}_t))$

receive correct label y_t

If $y_t f_t(\mathbf{x}_t) > 0$ set $I_{t+1} = I_t$ and $\forall (i \in I_t)$ set $\sigma_{i,t} = \sigma_{i,t-1}$

Else

(1) set $I'_t = I_t \cup \{t\}$

// *define* $f'_t = f_t + y_t K(\mathbf{x}, \cdot)$.

(2) choose $\phi_t \in (0, 1)$

set $\sigma_{t,t} = \phi_t$ and $\forall (i \in I_t)$ set $\sigma_{i,t} = \phi_t \, \sigma_{i,t-1}$

// *define* $f''_t = \phi_t f'_t$.

(3) **If** $|I_t| = B$ choose $r_t \in I_t$ **Else** set $r_t = \epsilon$

set $I_{t+1} = I'_t \setminus \{r_t\}$

define $f_{t+1} = \sum_{i \in I_{t+1}} \sigma_{i,t} y_i K(\mathbf{x}_i, \cdot)$

Figure 1: The template Forgetron algorithm.

Although we have not fully specified the steps of our update procedure, we can already prove a formal claim which applies to any algorithm that fits into the Forgetron framework. At this point we need to introduce some additional notation. Let T denote the number of examples presented to the online algorithm. For every i, let s_i be the round on which example i was removed from the active set, or $s_i = \epsilon$ if it was never removed. That is, s_i equals ϵ in one of two cases: either i was never inserted into the active set or $i \in I_{T+1}$. For every $s_i \neq \epsilon$, the weight σ_{i,s_i} is defined in Fig. 1, however, if $s_i = \epsilon$ then σ_{i,s_i} is not yet defined. For completeness of our notation, we set $\sigma_{i,\epsilon} = 0$ for all i. Finally, for every i such that $s_i \neq \epsilon$, define $\mu_i = 1 - y_i f''_{s_i}(\mathbf{x}_i)$. On round s_i, after completing step (2) of the update, we obtain the hypothesis f''_{s_i}, and our next step is to remove i from the active set. μ_i indicates how well f''_{s_i} classifies $\mathbf{x}_i$. It plays a central role in our analysis and motivates one of the algorithms presented in Sec. 4. We are now ready to state a general lemma from which we derive mistake bounds in the next section.

Lemma 1. *Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ be a sequence of examples such that $K(\mathbf{x}_t, \mathbf{x}_t) \leq 1$ for all t. Assume that this sequence is presented to the Forgetron algorithm in Fig. 1, and let J denote the set of rounds on which it makes a prediction mistake. Let g be an arbitrary function in $\mathcal{H}_K$ and define $\hat{\ell}_t = \ell\big(g; (\mathbf{x}_t, y_t)\big)$. If ϕ_t in step (2) of the algorithm is chosen to be at least $\min\{\frac{\|g\|}{\|f'_t\|}, B^{-\frac{1}{2B}}\}$ then,*

$$M \;\leq\; \|g\|^2 \left(1 + \frac{M \log(B)}{2B}\right) \;+\; 2 \sum_{t \in J} \hat{\ell}_t \;+\; \sum_{t \in J} (2\sigma_{t,s_t}\mu_t + \sigma^2_{t,s_t}) \;.$$

The proof is omitted due to the lack of space and can be found in [3].

4 Forgetron: Derived Algorithms

In this section we derive and analyze three specific algorithms from the Forgetron template algorithm defined in the previous section. Our goal is to prove that our algorithms are competitive with every fixed hypothesis $g \in \mathcal{H}_K$ for which $\|g\| \leq U$, for some constant

U. Recall that in Sec. 2 we proved that there does not exist an algorithm for which $U \geq \sqrt{B+1}$. We now present online algorithms for which,

$$U \;=\; \tfrac{1}{4}\sqrt{B/\log(B)} \;\; . \tag{3}$$

Forgetron.1 - Remove Oldest: We begin with the simplest of the three algorithms. On step (2) of the update in Fig. 1, the Forgetron.1 algorithm sets ϕ_t to,

$$\phi_t \;=\; \min\left\{\frac{U}{\|f'_t\|}\,,\, B^{-\frac{1}{2B}}\right\} \;\; , \tag{4}$$

and on step (3), on rounds where $|I_t| = B$, r_t is chosen to be,

$$r_t \;=\; \min I_t \;\; .$$

We can now prove a relative mistake bound for the Forgetron.1 algorithm, which holds when $B \geq 84$.

Theorem 2. *Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ be a sequence of examples such that $K(\mathbf{x}_t, \mathbf{x}_t) \leq 1$ for all t. Assume that this sequence is presented to the Forgetron.1 algorithm, with a budget parameter of $B \geq 84$. Let U be as defined in Eq. (3) and let g be an hypothesis in $\mathcal{H}_K$ with $\|g\| \leq U$. Denoting $\hat{\ell}_t = \ell\big(g; (\mathbf{x}_t, y_t)\big)$, it holds that, $M \leq 2U^2 + 4\sum_{t=1}^{T} \hat{\ell}_t$.*

Proof. The definition of ϕ_t from Eq. (4) clearly satisfies the condition on ϕ_t stated in Lemma 1, so we get that,

$$M \;\leq\; \|g\|^2\left(1 + \frac{M\log(B)}{2B}\right) \;+\; 2\sum_{t\in J}\hat{\ell}_t \;+\; \sum_{t\in J}(2\sigma_{t,s_t}\mu_t + \sigma^2_{t,s_t}) \;\; . \tag{5}$$

We derive the mistake bound in the theorem from the above by proving that $2\sigma_{t,s_t}\mu_t + \sigma^2_{t,s_t} \leq 15/32$ for all $t \in J$ and that $\|g\|^2\log(B)/(2B) \leq 1/32$. Starting with the first inequality, we note that if $s_t = \epsilon$ then $\sigma_{t,s_t} = 0$ and the inequality clearly holds, so we focus on rounds where $s_t \neq \epsilon$. Since μ_t is defined to be $1 - y_t f''_{s_t}(\mathbf{x}_t)$ we can use the Cauchy-Shwartz inequality along with the assumption that $K(\mathbf{x}_t, \mathbf{x}_t) \leq 1$ to obtain the bound $\mu_t \leq 1 + \|f''_{s_t}\|$. Since $\phi_t \leq U/f'_{s_t}$ and $f''_{s_t} = \phi_{s_t} f'_{s_t}$, it holds that $\|f''_{s_t}\| \leq U$. We thus get that $\mu_t \leq 1 + U$ and therefore,

$$2\sigma_{t,s_t}\mu_t + \sigma^2_{t,s_t} \;\leq\; 2\sigma_{t,s_t}(1+U) + \sigma^2_{t,s_t} \;\; . \tag{6}$$

Next, we bound σ_{t,s_t}. Since the oldest active example is the one removed from the active set, then the algorithm performs exactly B updates from the time an example is inserted into the active set until the time it is removed. In addition, $\sigma_{t,t}$ is initialized to ϕ_t and on each update this weight is multiplied by ϕ_i. Since $\phi_i \leq B^{-\frac{1}{2B}}$, we have that, $\sigma_{t,s_t} \leq (B^{-\frac{1}{2B}})^{B+1} \leq 1/\sqrt{B}$. Plugging this inequality back into Eq. (6) and using the definition of U from Eq. (3) we get,

$$2\sigma_{r_t,t}\mu_t + \sigma^2_{r_t,t} \;\leq\; \frac{2+2U}{\sqrt{B}} + \frac{1}{B} \;=\; \frac{2}{\sqrt{B}} + \frac{1}{2\sqrt{\log(B)}} + \frac{1}{B} \;\; .$$

The right-hand side of the above is monotonically decreasing in B and is smaller than $15/32$ for $B \geq 84$. We have thus shown that $2\sigma_{t,s_t}\mu_t + \sigma^2_{t,s_t} \leq 15/32$ for all $t \in J$. Finally note that the inequality $\|g\|^2\log(B)/(2B) \leq 1/32$ follows directly from the fact that $\|g\| \leq U$ and the definition of U in Eq. (3). Plugging these two inequalities into Eq. (5) and rearranging terms gives the bound stated in the theorem. $\square$

Forgetron.2 - A Gentler Scaling Scheme: In the second algorithm we present, r_t is again chosen to be the minimal index in I_t. In contrast to the previous algorithm, the scaling parameter ϕ_t is now set to be,

$$\max \left\{ \phi \in \left[0, B^{-\frac{1}{2B}}\right] \,:\, (\phi\, \sigma_{r_t,t-1})^2 + 2(\phi \sigma_{r_t,t-1})(1 - y_{r_t} \phi f'_t(\mathbf{x}_{r_t})) \;\leq\; \frac{15}{32} \right\} \quad . \quad (7)$$

In words, ϕ_t is set such that the inequality $2\sigma_{r_t,t}\mu_{r_t} + \sigma^2_{r_t,t} \leq 15/32$ is satisfied, while ensuring that ϕ_t is at most $B^{-\frac{1}{2B}}$. We now briefly argue that the mistake bound of Thm. 2 still holds for this choice of ϕ_t. To do so, we need to verify that the condition $\phi_t \geq \min\{\|g\|/\|f'_t\|\,,\, B^{-\frac{1}{2B}}\}$ from Lemma 1 is still met. To distinguish between the current choice of ϕ_t and the value of ϕ_t chosen by Forgetron.1, we denote the latter by $\hat{\phi}$. Since the choice of $\hat{\phi}$ clearly ensures that $\hat{\phi} \geq \min\{\|g\|/\|f'_t\|\,,\, B^{-\frac{1}{2B}}\}$, it is enough to show that $\phi_t \geq \hat{\phi}$. This inequality immediately follows from the facts that $\hat{\phi}$ is in the range $[0, B^{-\frac{1}{2B}}]$, it satisfies the inequality $(\hat{\phi}\, \sigma_{r_t,t-1})^2 + 2(\hat{\phi} \sigma_{r_t,t-1})(1 - y_{r_t} \hat{\phi} f'_t(\mathbf{x}_{r_t})) \leq 15/32$, and ϕ_t is chosen to be the maximal value among those which satisfy these constraints. In summary, we have shown that the condition stated in Lemma 1 holds and thus the mistake bound of Thm. 2 is still applicable by using the same line of derivation we used for Forgetron.1.

Forgetron.3 - A Smarter Removal Strategy The above two algorithms always remove the oldest element in I_t. While this removal strategy ensures that the inequality $2\sigma_{r_t,t}\mu_{r_t} + \sigma^2_{r_t,t} \leq \frac{15}{32}$ holds, this inequality may also hold for other elements of I_t. Furthermore, the bound on the number of mistakes from Lemma 1 decreases with the term $2\sigma_{r_t,t}\mu_{r_t} + \sigma^2_{r_t,t}$. Our third algorithm removes the index in I_t for which this term is the smallest. Formally, for each $r \in I_t$ define $Q(r) = 2B^{-\frac{1}{2B}}\, \sigma_{r,t-1}\mu_r + B^{-\frac{1}{B}}\, \sigma^2_{r,t-1}$, and let $\hat{r} = \arg\min_{r \in I_t} Q(r)$. In words, $\hat{r}$ is the best candidate for removal had we set ϕ_t to be $B^{-\frac{1}{2B}}$. Finally, if $Q(\hat{r}) \leq \frac{15}{32}$ then we set $\phi_t = B^{-\frac{1}{2B}}$ and $r_t = \hat{r}$. Otherwise, we resort to the previous algorithm and set $r_t = \min I_t$ and ϕ_t as in Eq. (7). We conclude this section by briefly showing how the mistake bound in Thm. 2 can be adapted to our third algorithm. First note that the value of ϕ_t for this algorithm cannot exceed the value assigned by the second algorithm and therefore the condition on ϕ_t stated in Lemma 1 holds. In addition, the definition of r_t implies that the inequality $2\sigma_{r_t,t}\mu_{r_t} + \sigma^2_{r_t,t} \leq 15/32$ also holds and thus the mistake bound of Thm. 2 is also applicable here.

5 Experiments

In this section we present experimental results which demonstrate the merits of our Forgetron algorithms. For brevity, we refer to the three algorithms as **F.1**, **F.2**, and **F.3**. We compare the performance of our algorithms with the method described in [2], which we abbreviate by **CKS**, and with the standard kernel-based Perceptron. When the **CKS** algorithm exceeds its budget, it removes the active example whose margin is the largest after the removal. In the first experiment, we examine the accuracy of the different budget algorithms in the presence of label noise. Recall that the number of active examples used by the basic Perceptron algorithm grows with each prediction mistake. Therefore, we expect the Perceptron algorithm to require a large active set in the presence of noise. In this experiment, we chose the Gaussian kernel and used a synthetic dataset which was generated as follows. We randomly sampled 5000 positive examples from a two-dimensional Gaussian with a mean of $(1, 1)$ and a diagonal covariance matrix, with $(0.2, 2)$ on its diagonal. We sampled 5000 negative examples from a Gaussian with a mean of $(-1, -1)$ and the same covariance matrix as before. Finally, we flipped each label with a probability of 0.1, thus introducing label noise. We then presented the data in an arbitrary order to each of the algorithms. We repeated this for different values of the budget parameter B, ranging from 10 to

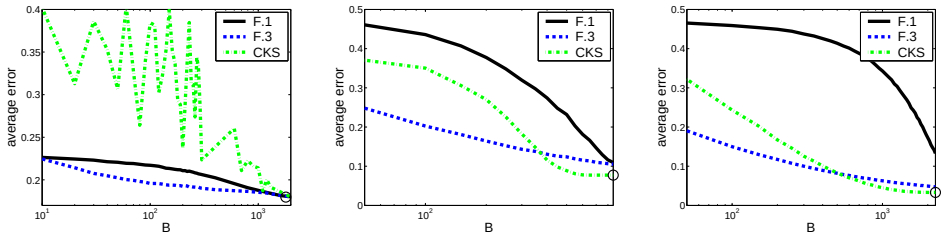

Figure 2: The average error of different budget algorithms as a function of the budget B on a synthetic dataset (left), the USPS dataset (middle) and the MNIST dataset (right). The average accuracy of the Perceptron and its budget requirements for each problem are marked by a circle.

2000. We repeated the entire experiment 10 times, where in each repetition we generated a new dataset, and averaged the results over the 10 repetitions. The average error attained by each algorithm for each choice of B is depicted on the left-hand side of Fig. 2. For clarity of presentation, we only depict the errors of the **F.1** and **F.3** algorithms, as the performance of **F.2** is rather similar to that of **F.3**. Since the standard Perceptron does not take a budget parameter, we mark its accuracy and active set size on the plot using a small circle. As can be seen from the plot, the Forgetron algorithms clearly outperform the **CKS** method. In fact, all of our algorithms achieve almost the same accuracy as the vanilla Perceptron algorithm while requiring less than a tenth of active set size required by the Perceptron.

Our next experiment was performed with two standard datasets: the MNIST dataset, which consists of 60,000 training examples, and the USPS dataset, with 10,000 examples. The instances in both datasets are handwritten images of digits, thus each image corresponds to one of the 10 digit classes. We generated 126 binary problems by splitting the 10 labels into two equal-size sets in all possible ways ($\binom{10}{5}/2 = 126$). For each budget value, we ran the various algorithms on all 126 binary problems and averaged the results. We chose a fifth degree non-homogeneous polynomial kernel for the MNIST dataset and a Gaussian kernel for the USPS dataset. The results of these experiments are summarized in the middle and on the right-hand side of Fig. 2. Note that regardless of the specific dataset and kernel used, the empirical results on the two datasets exhibit similar qualitative behavior. It is also apparent that both the **F.3** algorithm and the **CKS** method outperform the **F.1** algorithm. The relative inaccuracy of the **F.1** algorithm on these dataset might be explained by the rather aggressive scaling step it performs. Comparing the performance of the **F.3** algorithm with the **CKS** method, we note that the former performs better with small budgets while the latter is slightly better with large budgets. We leave further comparisons to future research.

References

[1] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive aggressive algorithms. *Leibniz TR* 2005-29. Available from http://leibniz.cs.huji.ac.il/tr/774.pdf.

[2] K. Crammer, J. Kandola, and Y. Singer. Online classification on a budget. *NIPS* 16, 2003.

[3] O. Dekel, S. Shalev-Shwartz and Y. Singer. The Forgetron: A Kernel-Based Perceptron on a Fixed Budget. *Leibniz TR* 2005-34. Available from http://leibniz.cs.huji.ac.il/tr/781.pdf.

[4] C. Gentile. A new approximate maximal margin classification algorithm. JMLR, 2003.

[5] J. Kivinen, A. J. Smola, and R. C. Williamson. Online learning with kernels. *IEEE TSP*, 2002.

[6] Yi Li and Phil M. Long. The relaxed online maximum margin algorithm. *NIPS* 13, 1999.

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

[8] V. N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[9] J. Weston, A. Bordes, and L. Bottou. Online (and offline) on an even tighter budget. *AISTATS* 10, 2005.