Online Learning with Kernels

Jyrki Kivinen, Alexander J. Smola, and Robert C. Williamson, *Member, IEEE*

***Abstract*—Kernel-based algorithms such as support vector machines have achieved considerable success in various problems in batch setting, where all of the training data is available in advance. Support vector machines combine the so-called kernel trick with the large margin idea. There has been little use of these methods in an online setting suitable for real-time applications. In this paper, we consider online learning in a reproducing kernel Hilbert space. By considering classical stochastic gradient descent within a feature space and the use of some straightforward tricks, we develop simple and computationally efficient algorithms for a wide range of problems such as classification, regression, and novelty detection.**

In addition to allowing the exploitation of the kernel trick in an online setting, we examine the value of large margins for classification in the online setting with a drifting target. We derive worst-case loss bounds, and moreover, we show the convergence of the hypothesis to the minimizer of the regularized risk functional.

We present some experimental results that support the theory as well as illustrating the power of the new algorithms for online novelty detection.

***Index Terms*—Classification, condition monitoring, large margin classifiers, novelty detection, regression, reproducing kernel Hilbert spaces, stochastic gradient descent, tracking.**

I. Introduction

KERNEL methods have proven to be successful in many batch settings (support vector machines, Gaussian processes, regularization networks) [1]. While one can apply batch algorithms by utilizing a sliding buffer [2], it would be much better to have a truely online algorithm. However, the extension of kernel methods to online settings where the data arrives sequentially has proven to provide some hitherto unsolved challenges.

A. Challenges for Online Kernel Algorithms

First, the standard online settings for linear methods are in danger of overfitting when applied to an estimator using a Hilbert space method because of the high dimensionality of the weight vectors. This can be handled by use of regularization (or exploitation of prior probabilities in function space if the Gaussian process view is taken).

Second, the functional representation of classical kernel-based estimators becomes more complex as the number of observations increases. The Representer Theorem [3] implies that the number of kernel functions can grow up to linearly with the number of observations. Depending on the loss function used [4], this will happen in practice in most cases. Thus, the complexity of the estimator used in prediction increases linearly over time (in some restricted situations, this can be reduced to logarithmical cost [5] or constant cost [6], yet with linear storage requirements). Clearly, this is not satisfactory for genuine online applications.

Third, the training time of batch and/or incremental update algorithms typically increases superlinearly with the number of observations. Incremental update algorithms [7] attempt to overcome this problem but cannot guarantee a bound on the number of operations required per iteration. Projection methods [8], on the other hand, will ensure a limited number of updates per iteration as well as keep the complexity of the estimator constant. However they can be computationally expensive since they require a matrix multiplication at each step. The size of the matrix is given by the number of kernel functions required at each step and could typically be in the hundreds in the smallest dimension.

In solving the above challenges it is highly desirable to be able to theoretically prove convergence rates and error bounds for any algorithms developed. One would want to be able to relate the performance of an online algorithm after seeing m observations to the quality that would be achieved in a batch setting. It is also desirable to be able to provide some theoretical insight in drifting target scenarios when a comparison with a batch algorithm makes little sense.

In this paper we present algorithms that deal effectively with these three challenges as well as satisfying the above desiderata.

B. Related Work

Recently several algorithms have been proposed [5], [9]–[11] that perform perceptron-like updates for classification at each step. Some algorithms work only in the noise-free case, others not for moving targets, and others assume an upper bound on the complexity of the estimators. In the present paper, we present a simple method that allows the use of kernel estimators for classification, regression, and novelty detection and copes with a large number of kernel functions efficiently.

The stochastic gradient descent algorithms we propose (collectively called NORMA) differ from the tracking algorithms of Warmuth, Herbster, and Auer [5], [12], [13] insofar as we do not require that the norm of the hypothesis be bounded beforehand. More importantly, we explicitly deal with the issues described earlier that arise when applying them to kernel-based representations.

Concerning large margin classification (which we obtain by performing stochastic gradient descent on the soft margin loss function), our algorithm is most similar to Gentile's ALMA [9], and we obtain similar loss bounds to those obtained for ALMA.

Manuscript received June 29, 2003; revised October 21, 2003. This work was supported by the Australian Research Council. Parts of this work were presented at the 13th International Conference on Algorithmic Learning Theory, November 2002, and the 15th Annual Conference on Neural Information Processing Systems, December 2001. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Alfred O. Hero.

J. Kivinen was with the Research School of Information Sciences and Engineering, The Australian National University, Canberra 0200, Australia. He is now with the University of Helsinki, Helsinki, Finland.

A. J. Smola and R. C. Williamson are with the Research School of Information Sciences and Engineering, The Australian National University and National ICT Australia, Canberra, Australia.

Digital Object Identifier 10.1109/TSP.2004.830991

One of the advantages of a large margin classifier is that it allows us to track changing distributions efficiently [14].

In the context of Gaussian processes (an alternative theoretical framework that can be used to develop kernel based algorithms), related work was presented in [8]. The key difference to our algorithm is that Csató and Opper repeatedly project on to a low-dimensional subspace, which can be computationally costly, requiring as it does a matrix multiplication.

Mesterharm [15] has considered tracking arbitrary linear classifiers with a variant of Winnow [16], and Bousquet and Warmuth [17] studied tracking of a small set of experts via posterior distributions.

Finally, we note that although not originally developed as an online algorithm, the sequential minimal optimization (SMO) algorithm [18] is closely related, especially when there is no bias term, in which case [19] it effectively becomes the Perceptron algorithm.

C. Outline of the Paper

In Section II, we develop the idea of stochastic gradient descent in Hilbert space. This provides the basis of our algorithms. Subsequently we show how the general form of the algorithm can be applied to problems of classification, novelty detection, and regression (Section III). Next we establish mistake bounds with moving targets for linear large margin classification algorithms in Section IV. A proof that the stochastic gradient algorithm converges to the minimum of the regularized risk functional is given in Section V, and we conclude with experimental results and a discussion in Sections VI and VII.

II. Stochastic Gradient Descent in Hilbert Space

We consider a problem of function estimation, where the goal is to learn a mapping $f : \mathcal{X} \to \mathbb{R}$ based on a sequence $S = ((x_1, y_1), \ldots, (x_m, y_m))$ of *examples* $(x_t, y_t) \in \mathcal{X} \times \mathcal{Y}$.

Moreover we assume that there exists a loss function $l : \mathbb{R} \times \mathcal{Y} \to \mathbb{R}$, given by $l(f(x), y)$, which penalizes the deviation of estimates $f(x)$ from observed labels y. Common loss functions include the soft margin loss function [20] or the logistic loss for classification and novelty detection [21], and the quadratic loss, absolute loss, Huber's robust loss [22], and the ε-insensitive loss [23] for regression. We will discuss these in Section III.

The reason for allowing the range of f to be $\mathbb{R}$ rather than $\mathcal{Y}$ is that it allows for more refinement in evaluation of the learning result. For example, in *classification* with $\mathcal{Y} = \{-1, 1\}$, we could interpret $\operatorname{sgn}(f(x))$ as the prediction given by f for the class of x and $|f(x)|$ as the confidence in that classification. We call the output f of the learning algorithm an hypothesis and denote the set of all possible hypotheses by $\mathcal{H}$.

We will always assume that $\mathcal{H}$ is a *reproducing kernel Hilbert space* (RKHS) [1]. This means that there exists a kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and a dot product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ such that

1) k has the reproducing property

$$\langle f, k(x, \cdot) \rangle_{\mathcal{H}} = f(x), \text{ for } x \in \mathcal{X}. \tag{1}$$

2) $\mathcal{H}$ is the closure of the span of all $k(x, \cdot)$ with $x \in \mathcal{X}$.

In other words, all $f \in \mathcal{H}$ are linear combinations of kernel functions. The inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ induces a norm on $f \in \mathcal{H}$ in the usual way: $\|f\|_{\mathcal{H}} := \langle f, f \rangle_{\mathcal{H}}^{1/2}$. An interesting special case is $\mathcal{X} = \mathbb{R}^n$ with $k(x, y) = \langle x, y \rangle$ (the normal dot-product in $\mathbb{R}^n$), which corresponds to learning linear functions in $\mathbb{R}^n$, but much more varied function classes can be learned by using different kernels.

A. Risk Functionals

In *batch learning*, it is typically assumed that all the examples are immediately available and are drawn independently from some distribution P over $\mathcal{X} \times \mathcal{Y}$. One natural measure of quality for f in that case is the *expected risk*

$$R[f, P] := E_{(x,y)\sim P}[l(f(x), y)]. \tag{2}$$

Since P is unknown, given S drawn from P^m, a standard approach [1] is to instead minimize the *empirical risk*

$$R_{\text{emp}}[f, S] := \frac{1}{m} \sum_{t=1}^{m} l(f(x_t), y_t). \tag{3}$$

However, minimizing $R_{\text{emp}}[f]$ may lead to overfitting (complex functions that fit well on the training data but do not generalize to unseen data). One way to avoid this is to penalize complex functions by instead minimizing the *regularized risk*

$$R_{\text{reg}}[f, S] := R_{\text{reg},\lambda}[f, S] := R_{\text{emp}}[f] + \frac{\lambda}{2} \|f\|_{\mathcal{H}}^2 \tag{4}$$

where $\lambda > 0$, and $\|f\|_{\mathcal{H}} = \langle f, f \rangle_{\mathcal{H}}^{1/2}$ does indeed measure the complexity of f in a sensible way [1]. The constant λ needs to be chosen appropriately for each problem. If l has parameters (for example l_ρ—see later), we write $R_{\text{emp},\rho}[f, S]$ and $R_{\text{reg},\lambda,\rho}[f, S]$.

Since we are interested in online algorithms, which deal with one example at a time, we also define an instantaneous approximation of $R_{\text{reg},\lambda}$, which is the *instantaneous regularized risk* on a single example (x, y), by

$$R_{\text{inst}}[f, x, y] := R_{\text{inst},\lambda}[f, x, y] := R_{\text{reg},\lambda}[f, ((x, y))]. \tag{5}$$

B. Online Setting

In this paper, we are interested in *online learning*, where the examples become available one by one, and it is desired that the learning algorithm produces a sequence of hypotheses $\mathbf{f} = (f_1, \ldots, f_{m+1})$. Here f_1 is some arbitrary initial hypothesis and f_i for $i > 1$ is the hypothesis chosen after seeing the $(i-1)$th example. Thus $l(f_t(x_t), y_t)$ is the loss the learning algorithm makes when it tries to predict y_t, based on x_t and the previous examples $(x_1, y_1), \ldots, (x_{t-1}, y_{t-1})$. This kind of learning framework is appropriate for real-time learning problems and is, of course, analogous to the usual adaptive signal processing framework [24]. We may also use an online algorithm simply as an efficient method of approximately solving a

batch problem. The algorithm we propose below can be effectively run on huge data sets on machines with limited memory.

A suitable measure of performance for online algorithms in an online setting is the *cumulative loss*

$$L_{\mathrm{cum}}[\mathbf{f}, S] = \sum_{t=1}^{m} l(f_t(x_t), y_t). \tag{6}$$

(Again, if l has such parameters as ρ, we write $L_{\mathrm{cum},\rho}[f]$, etc.) Notice that here f_t is tested on the example (x_t, y_t), which was not available for training f_t; therefore, if we can guarantee a low cumulative loss, we are already guarding against overfitting. Regularization can still be useful in the online setting: If the target we are learning changes over time, regularization prevents the hypothesis from going too far in one direction, thus hopefully helping recovery when a change occurs. Furthermore, if we are interested in large margin algorithms, some kind of complexity control is needed to make the definition of the margin meaningful.

C. General Idea of the Algorithm

The algorithms we study in this paper are classical stochastic gradient descent—they perform gradient descent with respect to the instantaneous risk. The general form of the update rule is

$$f_{t+1} := f_t - \eta_t \partial_f R_{\mathrm{inst},\lambda}[f, x_t, y_t]\big|_{f=f_t} \tag{7}$$

where for $i \in \mathbb{N}$, $f_i \in \mathcal{H}$, ∂_f is shorthand for $\partial/\partial f$ (the gradient with respect to f), and $\eta_t > 0$ is the *learning rate*, which is often constant $\eta_t = \eta$. In order to evaluate the gradient, note that the evaluation functional $f \mapsto f(x_i)$ is given by (1), and therefore

$$\partial_f l(f(x_t), y_t) = l'(f(x_t), y_t) k(x_t, \cdot) \tag{8}$$

where $l'(z, y) := \partial_z l(z, y)$. Since $\partial_f \|f\|_{\mathcal{H}}^2 = 2f$, the update becomes

$$f_{t+1} := (1 - \eta\lambda) f_t - \eta_t l'(f_t(x_t), y_t) k(x_t, \cdot). \tag{9}$$

Clearly, given $\lambda > 0$, η_t needs to satisfy $\eta_t < 1/\lambda$ for all t for the algorithm to work.

We also allow loss functions l that are only piecewise differentiable, in which case, ∂ stands for subgradient. When the subgradient is not unique, we choose one arbitrarily; the choice does not make any difference either in practice or in theoretical analyses. All the loss functions we consider are convex in the first argument.

Choose a zero initial hypothesis $f_1 = 0$. For the purposes of practical computations, one can write f_t as a kernel expansion (cf. [25])

$$f_t(x) = \sum_{i=1}^{t-1} \alpha_i k(x_i, x) \quad x \in \mathcal{X} \tag{10}$$

where the coefficients are updated at step t via

$$\alpha_t := -\eta_t l'(f_t(x_t), y_t), \text{ for } i = t \tag{11}$$

$$\alpha_i := (1 - \eta_t \lambda)\alpha_i, \qquad \text{for } i < t. \tag{12}$$

Thus, at step t, the tth coefficient may receive a nonzero value. The coefficients for earlier terms decay by a factor (which is constant for constant η_t). Notice that the cost for training at each step is not much larger than the prediction cost: Once we have computed $f_t(x_t)$, α_t is obtained by the value of the derivative of l at $(f_t(x_t), y_t)$.

D. Speedups and Truncation

There are several ways of speeding up the algorithm. Instead of updating all old coefficients α_i, $i = 1, \ldots, t-1$, one may simply cache the power series $1, (1 - \lambda\eta), (1 - \lambda\eta)^2, (1 - \lambda\eta)^3, \ldots$ and pick suitable terms as needed. This is particularly useful if the derivatives of the loss function l will only assume discrete values, say $\{-1, 0, 1\}$ as is the case when using the soft-margin type loss functions (see Section III).

Alternatively, one can also store $\tilde{\alpha}_t = (1 - \eta)^{-t}\alpha_t$ and compute $f_t(x) = (1 - \eta)^t \sum_{i=1}^{t-1} \tilde{\alpha}_i k(x_i, x_t)$, which only requires rescaling once $\tilde{\alpha}_t$ becomes too large for machine precision—this exploits the exponent in the standard floating point number representation.

A major problem with (11) and (12) is that without additional measures, the kernel expansion at time t contains t terms. Since the amount of computation required for predicting grows linearly in the size of the expansion, this is undesirable. The regularization term helps here. At each iteration, the coefficients α_i with $i \neq t$ are shrunk by $(1 - \lambda\eta)$. Thus, after τ iterations, the coefficient α_i will be reduced to $(1 - \lambda\eta)^\tau \alpha_i$. Hence one can drop small terms and incur little error, as the following proposition shows.

Proposition 1 (Truncation Error): Suppose $l(z, y)$ is a loss function satisfying $|\partial_z l(z, y)| \leq C$ for all $z \in \mathbb{R}$, $y \in \mathcal{Y}$, and k is a kernel with bounded norm $\|k(x, \cdot)\| \leq X$, where $\|\cdot\|$ denotes either $\|\cdot\|_{L_\infty}$ or $\|\cdot\|_{\mathcal{H}}$. Let $f_{\mathrm{trunc}} := \sum_{i=\max(1,t-\tau)}^{t-1} \alpha_i k(x_i, \cdot)$ denote the kernel expansion truncated to τ terms. The truncation error satisfies

$$\|f - f_{\mathrm{trunc}}\| \leq \sum_{i=1}^{t-\tau} \eta(1 - \lambda\eta)^{t-i} CX < (1 - \lambda\eta)^\tau \frac{CX}{\lambda}.$$

Obviously, the approximation quality increases exponentially with the number of terms retained.

The regularization parameter λ can thus be used to control the storage requirements for the expansion. In addition, it naturally allows for distributions $P(x, y)$ that change over time in which case it is desirable to *forget* instances (x_i, y_i) that are much older than the average time scale of the distribution change [26].

We call our algorithm the Naive Online R_{reg} Minimization Algorithm (NORMA) and sometimes explicitly write the parameter λ: NORMA $_\lambda$. NORMA is summarized in Fig. 1. In the applications discussed in Section III, it is sometimes necessary to introduce additional parameters that need to be updated. We nevertheless refer somewhat loosely to the whole family of algorithms as NORMA.

III. APPLICATIONS

The general idea of NORMA can be applied to a wide range of problems. We utilize the standard [1] addition of the constant

Given: A *sequence* $S = ((x_i, y_i))_{i\in\mathbb{N}} \in (\mathcal{X}\times\mathcal{Y})^\infty$; a *regularisation parameter* $\lambda > 0$; a *truncation parameter* $\tau \in \mathbb{N}$; a *learning rate* $\eta \in (0, 1/\lambda]$; a piecewise differentiable convex *loss function* $l\colon \mathbb{R}\times\mathcal{Y}\to\mathbb{R}$; and a *Reproducing Kernel Hilbert Space* $\mathcal{H}$ with reproducing *kernel* k, NORMA$_\lambda(S, l, k, \eta, \tau)$ outputs a sequence of hypotheses $\mathbf{f} = (f_1, f_2, \ldots) \in \mathcal{H}^\infty$.

Initialise $t := 1$; $\beta_i := (1-\lambda\eta)^i$ for $i = 0, \ldots, \tau$;

Loop

$f_t(\cdot) := \sum_{i=\max(1,t-\tau)}^{t-1} \alpha_i \beta_{t-i-1} k(x_i, \cdot)$;

$\alpha_t := -\eta l'(f_t(x_t), y_t)$;

$t := t + 1$;

End Loop

Fig. 1. NORMA $_\lambda$ with constant learning rate η exploiting the truncation approximation.

offset b to the function expansion, i.e., $g(x) := f(x) + b$, where $f \in \mathcal{H}$ and $b \in \mathbb{R}$. Hence, we also update b via

$$b_{t+1} := b_t - \eta\, \partial_b R_{\text{inst}}[g, x_t, y_t]\big|_{g=f_t+b_t}.$$

A. Classification

In (binary) classification, we have $\mathcal{Y} = \{\pm 1\}$. The most obvious loss function to use in this context is $l(f(x), y) = 1$ if $yf(x) \le 0$ and $l(f(x), y) = 0$ otherwise. Thus, no loss is incurred if $\operatorname{sgn}(f(x))$ is the correct prediction for y; otherwise, we say that f makes a *mistake* at (x, y) and charge a unit loss.

However, the mistake loss function has some drawbacks.

a) It fails to take into account the *margin* $yf(x)$ that can be considered to be a measure of confidence in the correct prediction: a nonpositive margin meaning an actual mistake.

b) The mistake loss is discontinuous and nonconvex and, thus, is unsuitable for use in gradient-based algorithms.

In order to deal with these drawbacks, the main loss function we use here for classification is the *soft margin loss*

$$l_\rho(f(x), y) := \max(0, \rho - yf(x)) \tag{13}$$

where $\rho \ge 0$ is the *margin parameter*. The soft margin loss $l_\rho(f(x), y)$ is positive if f fails to achieve a margin at least ρ on (x, y); in this case, we say that f made a *margin error*. If f made an actual mistake, then $l_\rho(f(x), y) \ge \rho$.

Let σ_t be an indicator of whether f_t made a margin error on (x_t, y_t), i.e., $\sigma_t = 1$ if $y_t f_t(x_t) \le \rho$ and zero otherwise. Then

$$l'_\rho(f_t(x_t), y_t) = -\sigma_t y_t = \begin{cases} -y_t, & \text{if } y_t f_t(x_t) \le \rho \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

and the update (9) becomes

$$f_{t+1} := (1-\eta\lambda) f_t + \eta\sigma_t y_t k(x_t, \cdot) \tag{15}$$

$$b_{t+1} := b_t + \eta\sigma_t y_t. \tag{16}$$

Suppose now that $X > 0$ is a bound such that $k(x_t, x_t) \le X^2$ holds for all t. Since $\|f_1\|_{\mathcal{H}} = 0$ and

$$\begin{aligned} \|f_{t+1}\|_{\mathcal{H}} &\le (1-\eta\lambda)\|f_t\|_{\mathcal{H}} + \eta\|k(x_t, \cdot)\|_{\mathcal{H}} \\ &= (1-\eta\lambda)\|f_t\|_{\mathcal{H}} + \eta k(x_t, x_t)^{1/2} \end{aligned}$$

we obtain $\|f_t\|_{\mathcal{H}} \le X/\lambda$ for all t. Furthermore

$$|f_t(x_t)| = |\langle f_t, k(x_t, \cdot)\rangle_{\mathcal{H}}| \le \frac{X^2}{\lambda}. \tag{17}$$

Hence, when the offset parameter b is omitted (which we consider particularly in Sections IV and V), it is reasonable to require $\rho \le X^2/\lambda$. Then the loss function becomes effectively bounded, with $l_\rho(f_t(x_t), y_t) \le 2X^2/\lambda$ for all t.

The update in terms of α_i is (for $i = 1, \ldots, t-1$)

$$(\alpha_i, \alpha_t, b) := ((1-\eta\lambda)\alpha_i, \eta\sigma_t y_t, b + \eta\sigma_t y_t). \tag{18}$$

When $\rho = 0$ and $\lambda = 0$ we recover the kernel perceptron [27]. If $\rho = 0$ and $\lambda > 0$ we have a kernel perceptron with regularization.

For *classification with the* ν-trick [4], we also have to take care of the margin ρ since there (recall $g(x) = f(x) + b$)

$$l(g(x), y) := \max(0, \rho - yg(x)) - \nu\rho. \tag{19}$$

Since one can show [4] that the specific choice of λ has no influence[1] on the estimate in ν-SV classification, we may set $\lambda = 1$ and obtain the update rule (for $i = 1, \ldots, t-1$)

$$(\alpha_i, \alpha_t, b, \rho) := ((1-\eta)\alpha_i, \eta\sigma_t y_t, b + \eta\sigma_t y_t, \rho + \eta(\nu - \sigma_t)).$$

B. Novelty Detection

Novelty detection [21] is like classification without labels. It is useful for condition monitoring tasks such as network intrusion detection. The absence of labels y_i means that the algorithm is not precisely a special case of NORMA as presented earlier, but one can derive a variant in the same spirit.

The *ν-setting* is most useful here as it allows one to specify an upper limit on the frequency of alerts $f(x) < \rho$. The loss function to be utilized is

$$l(f(x), x, y) := \max(0, \rho - f(x)) - \nu\rho$$

and usually [21] one uses $f \in \mathcal{H}$, rather than $g = f + b$ where $b \in \mathbb{R}$, in order to avoid trivial solutions. The update rule is (for $i = 1, \ldots, t-1$)

$$(\alpha_i, \alpha_t, \rho) := \begin{cases} ((1-\eta)\alpha_i, \eta, \rho + \eta(1-\nu)), & \text{if } f(x) < \rho \\ ((1-\eta)\alpha_i, 0, \rho - \eta\nu), & \text{otherwise.} \end{cases} \tag{20}$$

Consideration of the update for ρ shows that on average, only a fraction ν of the observations will be considered for updates. Thus, it is necessary to store only a small fraction of the x_is.

C. Regression

We consider the following three settings: squared loss, the ε-insensitive loss using the ν-trick, and Huber's robust loss function, i.e., trimmed mean estimators. For convenience, we will only use estimates $f \in \mathcal{H}$, rather than $g = f + b$, where $b \in \mathbb{R}$. The extension to the latter case is straightforward.

1) Squared Loss: Here, $l(f(x), y) := (1/2)(y - f(x))^2$. Consequently the update equation is (for $i = 1, \ldots, t-1$)

$$(\alpha_i, \alpha_t) := ((1-\lambda\eta)\alpha_i, \eta(y_t - f(x_t))). \tag{21}$$

This means that we have to store *every* observation we make or, more precisely, the prediction error we made on the observation.

[1]Note that the relative scale of ρ, b versus $\sum_i \alpha_i k(x_i, x)$ may make it more convenient to rescale the problem to some $\lambda \ne 1$ to improve convergence.

2) ε-Insensitive Loss: The use of the loss function $l(f(x), y) = \max(0, |y - f(x)| - \varepsilon)$ introduces a new parameter—the width of the insensitivity zone ε. By making ε a variable of the optimization problem, we have

$$l(f(x), y) := \max(0, |y - f(x)| - \varepsilon) + \nu\varepsilon.$$

The update equations now have to be stated in terms of α_i, α_t, and ε, which is allowed to change during the optimization process. Setting $\delta_t := y_t - f(x_t)$, the updates are (for $i = 1, \ldots, t-1$)

$$(\alpha_i, \alpha_t, \varepsilon) := \begin{cases} ((1-\lambda\eta)\alpha_i, \eta \operatorname{sgn} \delta_t, \varepsilon + (1-\nu)\eta), & \text{if } |\delta_t| > \varepsilon \\ ((1-\lambda\eta)\alpha_i, 0, \varepsilon - \eta\nu), & \text{otherwise.} \end{cases} \tag{22}$$

This means that every time the prediction error exceeds ε, we increase the insensitive zone by $\eta(1-\nu)$. If it is smaller than ε, the insensitive zone is decreased by $\eta\nu$.

3) Huber's Robust Loss: This loss function was proposed in [22] for robust maximum likelihood estimation among a family of unknown densities. It is given by

$$l(f(x), y) := \begin{cases} |y - f(x)| - \frac{1}{2}\sigma, & \text{if } |y - f(x)| \geq \sigma \\ \frac{1}{2\sigma}(y - f(x))^2, & \text{otherwise.} \end{cases} \tag{23}$$

Setting $\delta_t := y_t - f(x_t)$, the updates are (for $i = 1, \ldots, t-1$)

$$(\alpha_i, \alpha_t) := \begin{cases} ((1-\eta)\alpha_i, \eta \operatorname{sgn} \delta_t), & \text{if } |\delta_t| > \sigma \\ ((1-\eta)\alpha_i, \sigma^{-1}\delta_t), & \text{otherwise.} \end{cases} \tag{24}$$

Comparing (24) with (22) leads to the question of whether σ might also be adjusted adaptively. This is a desirable goal since we may not know the amount of noise present in the data. Although the ν setting allowed the formation of adaptive estimators for batch learning with the ε-insensitive loss, this goal has proven elusive for other estimators in the standard batch setting.

In the online situation, however, such an extension is quite natural (see also [28]). It is merely necessary to make σ a variable of the optimization problem, and the updates become (for $i = 1, \ldots, t-1$)

$$(\alpha_i, \alpha_t, \sigma) := \begin{cases} ((1-\eta)\alpha_i, \eta \operatorname{sgn} \delta_t, \sigma + \eta(1-\nu)), & \text{if } |\delta_t| > \sigma \\ ((1-\eta)\alpha_i, \sigma^{-1}\delta_t, \sigma - \eta\nu), & \text{otherwise.} \end{cases}$$

IV. Mistake Bounds for Nonstationary Targets

In this section we theoretically analyze NORMA for classification with the soft margin loss with margin ρ. In the process, we establish relative bounds for the soft margin loss. A detailed comparative analysis between NORMA and Gentile's ALMA [9] can be found in [14].

A. Definitions

We consider the performance of the algorithm for a fixed sequence of observations $S := ((x_1, y_1), \ldots, (x_m, y_m))$ and study the sequence of hypotheses $\mathbf{f} = (f_1, \ldots, f_m)$ produced by the algorithm on S. Two key quantities are the number of *mistakes*, given by

$$M(\mathbf{f}, S) := |\{1 \leq t \leq m \mid y_t f_t(x_t) \leq 0\}| \tag{25}$$

and the number of *margin errors*, given by

$$M_\rho(\mathbf{f}, S) := |\{1 \leq t \leq m \mid y_t f_t(x_t) \leq \rho\}|. \tag{26}$$

Notice that margin errors are those examples on which the gradient of the soft margin loss is nonzero; therefore $M_\rho(\mathbf{f}, S)$ gives the size of the kernel expansion of final hypothesis f_{m+1}.

We use σ_t to denote whether a margin error was made at trial t, i.e., $\sigma_t = 1$ if $y_t f_t(x_t) \leq \rho$ and $\sigma_t = 0$ otherwise. Thus the soft margin loss can be written as $l_\rho(f_t(x_t), y_t) = \sigma_t(\rho - y_t f_t(x_t))$ and consequently $L_{\mathrm{cum},\rho}[\mathbf{f}, S]$ denotes the total soft margin loss of the algorithm.

In our bounds, we compare the performance of NORMA with the performance of function sequences $\mathbf{g} = (g_1, \ldots, g_m)$ from some *comparison class* $\mathcal{G} \subset \mathcal{H}^m$.

Notice that we often use a different margin $\mu \neq \rho$ for the comparison sequence, and σ_t always refers to the margin errors of the actual algorithm with respect to its margin ρ. We always have

$$l_\mu(g(x), y) \geq \mu - yg(x). \tag{27}$$

We extend the notations $M(\mathbf{g}, S)$, $M_\mu(\mathbf{g}, S)$, $l_\mu(g_t, y_t)$, and $L_{\mathrm{cum},\mu}[\mathbf{g}, S]$ to such comparison sequences in the obvious manner.

B. Preview

To understand the form of the bounds, consider first the case of a stationary target, with comparison against a constant sequence $\mathbf{g} = (g, \ldots, g)$. With $\rho = \lambda = 0$, our algorithm becomes the kernelized Perceptron algorithm. Assuming that some g achieves $M_\mu(\mathbf{g}, S) = 0$ for some $\mu > 0$, the kernelized version of the Perceptron Convergence Theorem [27], [29] gives

$$M(\mathbf{f}, S) \leq \|g\|_{\mathcal{H}}^2 \max_t \frac{k(x_t, x_t)}{\mu^2}.$$

Consider now the more general case where the sequence is not linearly separable in the feature space. Then, ideally, we would wish for bounds of the form

$$M(\mathbf{f}, S) \leq \min_{\mathbf{g}=(g,\ldots,g)} M(\mathbf{g}, S) + o(m)$$

which would mean that the mistake rate of the algorithm would converge to the mistake rate of the best comparison function. Unfortunately, even approximately minimizing the number of mistakes over the training sequence is very difficult; therefore, such strong bounds for simple online algorithms seem unlikely. Instead, we settle for weaker bounds of the form

$$M(\mathbf{f}, S) \leq \min_{\mathbf{g}=(g,\ldots,g), \|g\|_{\mathcal{H}} \leq B} \frac{L_{\mathrm{cum},\mu}[\mathbf{g}, S]}{\mu} + o(m) \tag{28}$$

where $L_{\mathrm{cum},\mu}[\mathbf{g}, S]/\mu$ is an upper bound for $M(\mathbf{g}, S)$, and the norm bound B appears as a constant in the $o(m)$ term. For earlier bounds of this form, see [30] and [31].

In the nonstationary case, we consider comparison classes that are allowed to change slowly, that is

$$\mathcal{G}(B, D_1, D_2) := \left\{ (g_1, \ldots, g_m) | \sum_{t=1}^{m-1} \|g_t - g_{t+1}\|_{\mathcal{H}} \leq D_1 \right. \\ \left. \sum_{t=1}^{m-1} \|g_t - g_{t+1}\|_{\mathcal{H}}^2 \leq D_2 \text{ and } \|g_t\|_{\mathcal{H}} \leq B \right\}.$$

The parameter D_1 bounds the total distance travelled by the target. Ideally, we would wish the target movement to result in an additional $O(D_1)$ term in the bounds, meaning there would be a constant cost per unit step of the target. Unfortunately, for technical reasons, we also need the D_2 parameter, which restricts the changes of speed of the target. The meaning of the D_2 parameter will become clearer when we state our bounds and discuss them.

Choosing the parameters is an issue in the bounds we have. The bounds depend on the choice of the learning rate and margin parameters, and the optimal choices depend on quantities (such as $\min_{\mathbf{g}} L_{\mathrm{cum},\mu}[\mathbf{g}, S]$) that would not be available when the algorithm starts. In our bounds, we handle this by assuming an upper bound $K \geq \min_{\mathbf{g}} L_{\mathrm{cum},\mu}[\mathbf{g}, S]$ that can be used for tuning. By substituting $K = \min_{\mathbf{g}} L_{\mathrm{cum},\mu}[\mathbf{g}, S]$, we obtain the kind of bound we discussed above; otherwise, the estimate K replaces $\min_{\mathbf{g}} L_{\mathrm{cum},\mu}[\mathbf{g}, S]$ in the bound. In a practical application, one would probably be best served to ignore the formal tuning results in the bounds and just tune the parameters by whatever empirical methods are preferred. Recently, online algorithms have been suggested that dynamically tune the parameters to almost optimal values as the algorithm runs [9], [32]. Applying such techniques to our analysis remains an open problem.

C. Relative Loss Bounds

Recall that the update for the case we consider is

$$f_{t+1} := (1 - \eta\lambda) f_t + \eta \sigma_t y_t k(x_t, \cdot). \tag{29}$$

It will be convenient to give the parameter tunings in terms of the function

$$h(x, K, C) = \sqrt{\frac{C}{K}\left(x + \frac{C}{K}\right)} - \frac{C}{K} \tag{30}$$

where we assume x, K, and C to be positive. Notice that $0 \leq h(x, K, C) \leq x$ holds, and $\lim_{K \to 0+} h(x, K, C) = x/2$. Accordingly, we define $h(x, 0, C) = x/2$.

We start by analyzing margin errors with respect to a given margin ρ.

Theorem 2: Suppose $\mathbf{f}$ is generated by (29) on a sequence S of length m. Let $X > 0$, and suppose that $k(x_t, x_t) \leq X^2$ for all t. Fix $K \geq 0$, $B > 0$, $D_1 \geq 0$, and $D_2 \geq 0$. Let

$$C = \frac{1}{4} X^2 \left(B^2 + B\left(\sqrt{m D_2} + D_1\right)\right) \tag{31}$$

and, given parameters $\mu > \rho \geq 0$, let $\eta' = 2h(\mu - \rho, K, C)/X^2$. Choose the regularization parameter

$$\lambda = (B\eta')^{-1} \sqrt{\frac{D_2}{m}} \tag{32}$$

and the learning rate parameter $\eta = \eta'/(1 + \eta'\lambda)$. If, for some $\mathbf{g} \in \mathcal{G}(B, D_1, D_2)$, we have $L_{\mathrm{cum},\mu}[\mathbf{g}, S] \leq K$, then

$$M_\rho(\mathbf{f}, S) \leq \frac{K}{\mu - \rho} + \frac{2C}{(\mu - \rho)^2} \\ + 2\left(\frac{C}{(\mu - \rho)^2}\right)^{1/2} \left(\frac{K}{\mu - \rho} + \frac{C}{(\mu - \rho)^2}\right)^{1/2}.$$

The proof can be found in Appendix A.

We now consider obtaining mistake bounds from our margin error result. The obvious method is to set $\rho = 0$, turning margin errors directly to mistakes. Interestingly, it turns out that a subtly different choice of parameters allows us to obtain the same mistake bound using a nonzero margin.

Theorem 3: Suppose $\mathbf{f}$ is generated by (29) on a sequence S of length m. Let $X > 0$, and suppose that $k(x_t, x_t) \leq X^2$ for all t. Fix K, B, D_1, $D_2 \geq 0$, and define C as in (31), and given $\mu > 0$, let $\eta' = 2r/X^2$, where $r = h(\mu, K, C)$. Choose the regularization parameter as in (32) and the learning rate $\eta = \eta'/(1 + \eta'\lambda)$, and set the margin to either $\rho = 0$ or $\rho = \mu - r$. Then, for either of these margin settings, if there exists a comparison sequence $\mathbf{g} \in \mathcal{G}(B, D_1, D_2)$ such that $L_{\mathrm{cum},\mu}[\mathbf{g}, S] \leq K$, we have

$$M(\mathbf{f}, S) \leq \frac{K}{\mu} + \frac{2C}{\mu^2} + 2\left(\frac{C}{\mu^2}\right)^{1/2} \left(\frac{K}{\mu} + \frac{C}{\mu^2}\right)^{1/2}.$$

The proof of Theorem 3 is also in Appendix A.

To gain intuition about Theorems 2 and 3, consider first the separable case $K = 0$ with a stationary target ($D_1 = D_2 = 0$). In this special case, Theorem 3 gives the familiar bound from the Perceptron Convergence Theorem. Theorem 2 gives an upper bound of $X^2 B^2/(\mu - \rho)^2$ margin errors. The choices given for ρ in Theorem 3 for the purpose of minimizing the mistake bound are, in this case, $\rho = 0$ and $\rho = \mu/2$. Notice that the latter choice results in a bound of $4X^2 B^2/\mu$ margin errors. More generally, if we choose $\rho = (1 - \epsilon)\mu$ for some $0 < \epsilon < 1$ and assume μ to be the largest margin for which separation is possible, we see that the algorithm achieves in $O(\epsilon^{-2})$ iterations a margin within a factor $1 - \epsilon$ of optimal. This bound is similar to that for ALMA [9], but ALMA is much more sophisticated in that it automatically tunes its parameters.

Removing the separability assumption leads to an additional K/μ term in the mistake bound, as we expected. To see the effects of the D_1 and D_2 terms, assume first that the target has constant speed: $\|g_t - g_{t+1}\|_{\mathcal{H}} = \delta$ for all t, where $\delta > 0$ is a constant. Then $D_1 = m\delta$, and $D_2 = m\delta^2$; therefore, $\sqrt{mD_2} = D_1$. If the speed is not constant, we always have $\sqrt{mD_2} > D_1$. An extreme case would be $\|g_1 - g_2\|_{\mathcal{H}} = D_1$, $g_{t+1} = g_t$ for $t > 1$. Then $\sqrt{mD_2} = \sqrt{m} D_1$. Thus the D_2 term increases the bound in case of changing target speed.

V. Convergence of Norma

A. Preview

Next we study the performance of Norma when it comes to minimizing the regularized risk functional $R_{\mathrm{reg}}[f,S]$, of which $R_{\mathrm{inst}}[f,x_t,y_t]$ is the stochastic approximation at time t. We show that under some mild assumptions on the loss function, the average instantaneous risk $(1/m)\sum_{t=1}^{m} R_{\mathrm{inst}}[f_t,x_t,y_t]$ of the hypotheses f_t of Norma converges toward the minimum regularized risk $\min_g R_{\mathrm{reg}}[g,S]$ at rate $O(m^{-1/2})$. This requires no probabilistic assumptions. If the examples are i.i.d., then with high probability, the expected regularized risk of the average hypothesis $(1/m)\sum_{t=1}^{m} f_t$ similarly converges toward the minimum expected risk. Convergence can also be guaranteed for the truncated version of the algorithm that keeps its kernel expansion at a sublinear size.

B. Assumptions and Notation

We assume a bound $X>0$ such that $k(x_t,x_t)\le X^2$ for all t. Then for all $g\in\mathcal{H}$, $|g(x_t)| = |\langle g,k(x_t,\cdot)\rangle_{\mathcal{H}}| \le X\|g\|_{\mathcal{H}}$.

We assume that the loss function l is convex in its first argument and satisfies, for some constant $c>0$, the Lipschitz condition

$$|l(z_1,y)-l(z_2,y)|\le c|z_1-z_2| \tag{33}$$

for all $z_1,z_2\in\mathbb{R}$, and $y\in\mathcal{Y}$.

Now fix $\lambda>0$. The hypotheses f_t produced by (9)

$$\begin{aligned}\|f_{t+1}\|_{\mathcal{H}} &= \|(1-\eta_t\lambda)f_t-\eta_t l'(f(x_t),y_t)k(x_t,\cdot)\|_{\mathcal{H}}\\ &\le (1-\eta_t\lambda)\|f_t\|_{\mathcal{H}}+\eta_t cX\end{aligned}$$

and since $f_1=0$, we have, for all t, the bound $\|f_t\|_{\mathcal{H}}\le U$, where

$$U:=\frac{cX}{\lambda}. \tag{34}$$

Since $|l'(f(x_t),y_t)|\le c$, we have $\|\partial_f l(f(x_t),y_t)\|_{\mathcal{H}}\le cX$ and $\|\partial_f R_{\mathrm{inst}}[f,x_t,y_t]\|_{\mathcal{H}}\le cX+\lambda\|f\|_{\mathcal{H}}\le 2cX$ for any f such that $\|f\|_{\mathcal{H}}\le U$.

Fix a sequence S, and for $0<\epsilon<1$, define

$$\hat{g}:=\operatorname*{argmin}_{g\in\mathcal{H}} R_{\mathrm{reg}}[g,S],\quad g:=(1-\epsilon)\hat{g}.$$

$$\begin{aligned}\text{Then } 0 &\le R_{\mathrm{reg}}[g,S]-R_{\mathrm{reg}}[\hat{g},S]\\ &= \frac{1}{m}\sum_{t=1}^{m}(l(g(x_t),y_t)-l(\hat{g}(x_t),y_t))\\ &\quad+\frac{\lambda}{2}(\|g\|_{\mathcal{H}}^2-\|\hat{g}\|_{\mathcal{H}}^2)\\ &\le cX\|g-\hat{g}\|_{\mathcal{H}}+\frac{\lambda}{2}((1-\epsilon)^2-1)\|\hat{g}\|_{\mathcal{H}}^2\\ &= cX\epsilon\|\hat{g}\|_{\mathcal{H}}-\lambda\epsilon\|\hat{g}\|_{\mathcal{H}}^2+\frac{\lambda\epsilon^2}{2}\|\hat{g}\|_{\mathcal{H}}^2.\end{aligned}$$

Considering the limit $\epsilon\to 0+$ shows that $\|\hat{g}\|_{\mathcal{H}}\le U$, where U is as in (34).

C. Basic Convergence Bounds

We start with a simple cumulative risk bound. To achieve convergence, we use a decreasing learning rate.

Theorem 4: Fix $\lambda>0$ and $0<\eta<1/\lambda$. Assume that l is convex and satisfies (33). Let the example sequence $S=((x_t,y_t))_{t=1}^{m}$ be such that $k(x_t,x_t)\le X^2$ holds for all t, and let $(f_1,\ldots,f_{m+1})$ be the hypothesis sequence produced by Norma with learning rate $\eta_t=\eta t^{-1/2}$. Then, for any $g\in\mathcal{H}$, we have

$$\sum_{t=1}^{m} R_{\mathrm{inst},\lambda}[f_t,x_t,y_t]\le mR_{\mathrm{reg},\lambda}[g,S]+am^{1/2}+b \tag{35}$$

where $a=2\lambda U^2(2\eta\lambda+1/(\eta\lambda))$, $b=U^2/(2\eta)$, and U is as in (34).

The proof, which is given in Appendix B, is based on analyzing the progress of f_t toward g at update t. The basic technique is from [32]–[34], and [32] shows how to adjust the learning rate (in a much more complicated setting than we have here).

Note that (35) holds in particular for $g=\hat{g}$; therefore

$$\frac{1}{m}\sum_{t=1}^{m} R_{\mathrm{inst},\lambda}[f_t,x_t,y_t]\le R_{\mathrm{reg},\lambda}[\hat{g},S]+O(m^{-1/2})$$

where the constants depend on X, c, and the parameters of the algorithm. However, the bound does not depend on any probabilistic assumptions. If the example sequence is such that some fixed predictor g has a small regularized risk, then the average regularized risk of the online algorithm will also be small.

Consider now the implications of Theorem 4 to a situation in which we assume that the examples (x_t,y_t) are i.i.d. according to some fixed distribution P. The bound on the cumulative risk can be transformed into a probabilistic bound by standard methods. We assume that $k(x,x)\le X^2$ with probability 1 for $(x,y)\sim P$. We say that the *risk is bounded by* L if with probability 1 we have $R_{\mathrm{inst},\lambda}[f,x_t,y_t]\le L$ for all t and $f\in\{\hat{g},f_1,\ldots,f_{m+1}\}$.

As an example, consider the soft margin loss. By the preceding remarks, we can assume $\|f\|_{\mathcal{H}}\le X/\lambda$. This implies $|f(x_t)|\le X^2/\lambda$; therefore, the interesting values of ρ satisfy $0\le\rho\le X^2/\lambda$. Hence, $l_\rho(f(x_t),y_t)\le 2X^2/\lambda$, and we can take $L=5X^2/(2\lambda)$. If we wish to use an offset parameter b, a bound for $|b|$ needs to be obtained and incorporated into L. Similarly, for regression-type loss functions, we may need a bound for $|y_t|$.

The result of Cesa-Bianchi *et al.* for bounded convex loss functions [35, Th. 2] now directly gives the following.

Corollary 5: Assume that P is a probability distribution over $\mathcal{X}\times\mathcal{Y}$ such that $k(x,x)\le X^2$ holds with probability 1 for $(x,y)\sim P$, and let the example sequence $S=((x_t,y_t))_{t=1}^{m}$ be drawn i.i.d. according to P. Fix $\lambda>0$ and $0<\eta<1/\lambda$. Assume that l is convex and satisfies (33) and that the risk is bounded by L. Let $\bar{f}_m=(1/m)\sum_{t=1}^{m-1} f_t$, where f_t is the tth hypothesis produced by Norma with learning rate $\eta_t=\eta t^{-1/2}$. Then, for any $g\in\mathcal{H}$ and $0<\delta<1$ and for a and b, as in Theorem 4 we have

$$\begin{aligned}&E_{(x,y)\sim P}R_{\mathrm{inst},\lambda}[\bar{f}_m,x,y]\\ &\quad\le R_{\mathrm{reg},\lambda}[g,S]+\frac{1}{m^{1/2}}\left(a+L\left(2\ln\left(\frac{1}{\delta}\right)\right)^{1/2}\right)+\frac{b}{m}\end{aligned}$$

with probability at least $1-\delta$ over random draws of S.

To apply Corollary 5, choose $g=g_*$, where

$$g_*=\operatorname*{argmin}_{f\in\mathcal{H}} E_{(x,y)\sim P}R_{\mathrm{inst},\lambda}[f,x,y]. \tag{36}$$

With high probability, $R_{\mathrm{reg},\lambda}[g_*, S]$ will be close to $\mathrm{E}_{(x,y)\sim P} R_{\mathrm{inst},\lambda}[g_*, x, y]$; therefore, with high probability, $\mathrm{E}_{(x,y)\sim P} R_{\mathrm{inst},\lambda}[\bar{f}_m, x, y]$ will be close to the minimum expected risk.

D. Effects of Truncation

We now consider a version of NORMA where at time t, the hypothesis consists of a kernel expansion of size s_t, where we allow s_t to slowly (sublinearly) increase as a function of t. Thus

$$f_t(x) = \sum_{\tau=1}^{s_t} \alpha_{t-\tau,t} k(x_{t-\tau}, x)$$

where $\alpha_{t,t'}$ is the coefficient of $k(x_t, \cdot)$ in the kernel expansion at time t'. For simplicity, we assume $s_{t+1} \in \{s_t, s_t + 1\}$ and include in the expansion even the terms where $\alpha_{t,t} = 0$. Thus, at any update, we add a new term to the kernel expansion; if $s_{t+1} = s_t$, we also drop the oldest previously remaining term. We can then write

$$f_{t+1} = f_t - \eta_t \partial_f R_{\mathrm{inst}}[f, x_t, y_t]|_{f=f_t} - \Delta_t$$

where $\Delta_t = 0$ if $s_{t+1} = s_t + 1$ and $\Delta_t = \alpha_{t-s_t,t} k(x_{t-s_t}, \cdot)$ otherwise. Since $\alpha_{t,t'+1} = (1 - \eta_{t'}\lambda)\alpha_{t,t'}$, we see that the kernel expansion coefficients decay almost geometrically. However, since we also need to use a decreasing learning rate $\eta_t = \eta t^{-1/2}$, the factor $1 - \eta_t \lambda$ approaches 1. Therefore, it is somewhat complicated to choose expansion sizes s_t that are not large but still guarantee that the cumulative effect of the Δ_t terms remains under control.

Theorem 6: Assume that l is convex and satisfies (33). Let the example sequence $S = ((x_t, y_t))_{t=1}^m$ be such that $k(x_t, x_t) \le X^2$ holds for all t. Fix $\lambda > 0$, $0 < \eta < 1/\lambda$, and $0 < \epsilon < 1/2$. Then there is a value $t_0(\lambda, \eta, \epsilon)$ such that the following holds when we define $s_t = t$ for $t \le t_0(\lambda, \eta, \epsilon)$ and $s_t = \lceil t^{1/2+\epsilon} \rceil$ for $t > t_0(\lambda, \eta, \epsilon)$. Let $(f_1, \ldots, f_{m+1})$ be the hypothesis sequence produced by truncated NORMA with learning rate $\eta_t = \eta t^{-1/2}$ and expansion sizes s_t. Then, for any $g \in \mathcal{H}$, we have

$$\sum_{t=1}^{m} R_{\mathrm{inst},\lambda}[f_t, x_t, y_t] \le m R_{\mathrm{reg},\lambda}[g, S] + a m^{1/2} + b \quad (37)$$

where $a = 2\lambda U^2 (10\eta\lambda + 1/(\eta\lambda))$, $b = U^2/(2\eta)$, and U is as in (34).

The proof, and the definition of t_0, is given in Appendix C.

Conversion of the result to a probabilistic setting can be done as previously, although an additional step is needed to estimate how the Δ_t terms may affect the maximum norm of f_t; we omit the details.

VI. EXPERIMENTS

The mistake bounds in Section IV are, of course, only worst-case upper bounds, and the constants may not be very tight. Hence, we performed experiments to evaluate the performance of our stochastic gradient descent algorithms in practice.

A. Classification

Our bounds suggest that some form of regularization is useful when the target is moving, and forcing a positive margin may give an additional benefit.

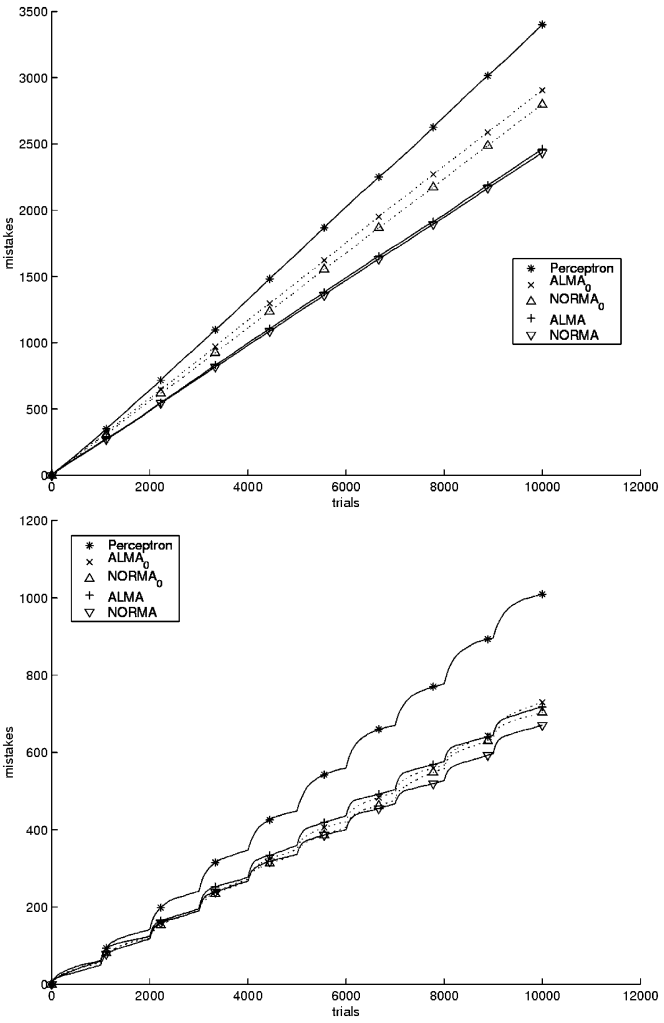


Fig. 2. Mistakes made by the algorithms on drifting data (top) and on switching data (bottom).

This hypothesis was tested using artificial data, where we used a mixture of two-dimensional Gaussians for the positive examples and another for negative ones. We removed all examples that would be misclassified by the Bayes-optimal classifier (which is based on the actual distribution known to us) or are close to its decision boundary. This gave us data that were cleanly separable using a Gaussian kernel.

In order to test the ability of NORMA to deal with changing underlying distributions, we carried out random changes in the parameters of the Gaussians. We used two movement schedules.

- In the *drifting* case, there is a relatively small parameter change after every ten trials.
- In the *switching* case, there is a very large parameter change after every 1000 trials.

Thus, given the form of our bounds, all other things being equal, our mistake bound would be much better in the drifting than in the switching case. In either case, we ran each algorithm for 10 000 trials and cumulatively summed up the mistakes made by them.

In our experiments, we compared NORMA $_{\lambda,\rho}$ with ALMA [9] with $p = 2$ and the basic Perceptron algorithm (which is the same stochastic gradient descent with the margin ρ in the loss function (13) and weight decay parameter λ both set to zero). We also considered variants NORMA $_{\lambda,0}$ and ALMA $_0$, where the

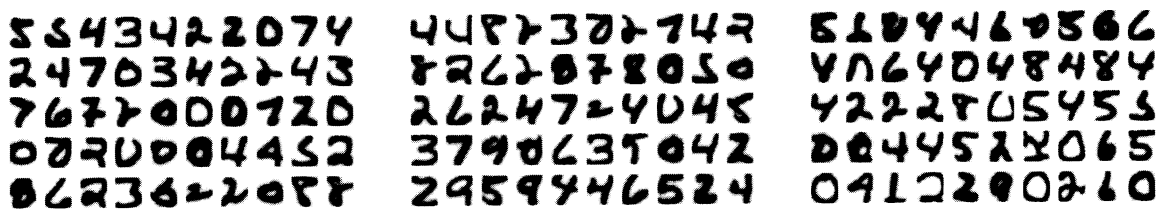


Fig. 3. Results of online novelty detection after one pass through the USPS database. The learning problem is to discover (online) novel patterns. We used Gaussian RBF kernels with width $2\sigma^2 = 0.5d = 128$ and $\nu = 0.01$. The learning rate was $(1/\sqrt{t})$. (Left) First 50 patterns that incurred a margin error—it can be seen that the algorithm at first finds even well-formed digits novel but later only finds unusually written ones. (Middle) Fifty worst patterns according to $f(x) - \rho$ on the training set—they are mostly badly written digits. (Right) Fifty worst patterns on an unseen test set.

margin ρ is fixed to zero. These algorithms are included to see whether regularization, either by weight decay as in NORMA or by a norm bound as in ALMA, helps to predict a moving target, even when we are not aiming for a large margin. We used Gaussian kernels to handle the nonlinearity of the data. For these experiments, the parameters of the algorithms were tuned by hand optimally for each example distribution.

Fig. 2 shows the cumulative mistake counts for the algorithms. There does not seem to be any decisive differences between the algorithms.

In particular, NORMA works quite well on switching data as well, even though our bound suggests otherwise (which is probably due to slack in the bound). In general, it seems that using a positive margin is better than fixing the margin to zero, and regularization, even with zero margin, is better than the basic Perceptron algorithm.

B. Novelty Detection

In our experiments, we studied the performance of the novelty detection variant of NORMA given by (20) for various kernel parameters and values of ν.

We performed experiments on the USPS database of handwritten digits (scanned images of digits at a resolution of 16×16 pixels; 7291 were chosen for training and 2007 for testing purposes).

Already after one pass through the database, which took in MATLAB less than 15 s on a 433 MHz Celeron, the results can be used to weed out badly written digits (cf. the left plot of Fig. 3). We chose $\nu = 0.01$ to allow for a fixed fraction of detected "outliers." Based on the theoretical analysis of Section V, we used a decreasing learning rate with $\eta_t \propto t^{-(1/2)}$.

Fig. 3 shows how the algorithm improves in its assessment of unusual observations (the first digits in the left table are still quite regular but degrade rapidly). It could therefore be used as an online data filter.

VII. DISCUSSION

We have shown how the careful application of classical stochastic gradient descent can lead to novel and practical algorithms for online learning using kernels. The use of regularization (which is essential for capacity control when using the rich hypothesis spaces generated by kernels) allows for truncation of the basis expansion and, thus, computationally efficient hypotheses. We explicitly developed parameterizations of our algorithm for classification, novelty detection, and regression. The algorithm is the first we are aware of for online novelty detection. Furthermore, its general form is very efficient computationally and allows for easy application of kernel methods to enormous data sets as well as, of course, to real-time online problems.

We also presented a theoretical analysis of the algorithm when applied to classification problems with soft margin ρ with the goal of understanding the advantage of securing a large margin when tracking a drifting problem. On the positive side, we have obtained theoretical bounds that give some guidance to the effects of the margin in this case. On the negative side, the bounds are not that well corroborated by the experiments we performed.

APPENDIX A
PROOFS OF THEOREMS 2 AND 3

The following technical lemma, which is proved by a simple differentiation, is used in both proofs for choosing the optimal parameters.

Lemma 7: Given $K > 0$, $C > 0$, and $\gamma > 0$, define $f(z) = K/(\gamma - z) + C/(z(\gamma - z))$ for $0 < z < \gamma$. Then, $f(z)$ is maximized for $z = h(\gamma, K, C)$, where h is as in (30), and the maximum value is

$$f(h(\gamma, K, C)) = \frac{K}{\gamma} + \frac{2C}{\gamma^2} + 2\left(\frac{K}{\gamma} + \frac{C}{\gamma^2}\right)^{1/2}\left(\frac{C}{\gamma^2}\right)^{1/2}.$$

The main idea in the proofs is to lower bound the *progress* at update t, which we define as $\|g_t - f_t\|_{\mathcal{H}}^2 - \|g_{t+1} - f_{t+1}\|_{\mathcal{H}}^2$. For notational convenience, we introduce $g_{m+1} := g_m$.

Proof of Theorem 2: Define $f'_{t+1} = f_t + \eta' \sigma_t y_t k(x_t, \cdot)$. We split the progress into three parts:

$$\begin{aligned}
&\|g_t - f_t\|_{\mathcal{H}}^2 - \|g_{t+1} - f_{t+1}\|_{\mathcal{H}}^2 \\
&\quad = (\|g_t - f_t\|_{\mathcal{H}}^2 - \|g_t - f'_{t+1}\|_{\mathcal{H}}^2) \\
&\qquad + (\|g_t - f'_{t+1}\|_{\mathcal{H}}^2 - \|g_t - f_{t+1}\|_{\mathcal{H}}^2) \\
&\qquad + (\|g_t - f_{t+1}\|_{\mathcal{H}}^2 - \|g_{t+1} - f_{t+1}\|_{\mathcal{H}}^2). \qquad (38)
\end{aligned}$$

By substituting the definition of f'_{t+1}, using (27), and applying $\sigma_t l_\mu(g_t(x_t), y_t) \le l_\mu(g_t(x_t), y_t)$, we can estimate the first part of (38) as

$$\begin{aligned}
&\|g_t - f_t\|_{\mathcal{H}}^2 - \|g_t - f'_{t+1}\|_{\mathcal{H}}^2 \\
&\quad = 2\eta' \sigma_t y_t \langle k(x_t, \cdot), g_t - f_t \rangle_{\mathcal{H}} - \|f_t - f'_{t+1}\|_{\mathcal{H}}^2 \\
&\quad = 2\eta' \sigma_t y_t (g_t(x_t) - f_t(x_t)) - \eta'^2 \sigma_t k(x_t, x_t) \\
&\quad \ge 2\eta'(\sigma_t \mu - l_\mu(g_t(x_t), y_t)) \\
&\qquad - 2\eta'(\sigma_t \rho - l_\rho(f_t(x_t), y_t)) - \eta'^2 \sigma_t X^2. \qquad (39)
\end{aligned}$$

For the second part of (38), we have

$$\|g_t - f'_{t+1}\|_{\mathcal{H}}^2 - \|g_t - f_{t+1}\|_{\mathcal{H}}^2 = \|f_{t+1} - f'_{t+1}\|_{\mathcal{H}}^2 + 2\left\langle f'_{t+1} - f_{t+1}, f_{t+1} - g_t\right\rangle_{\mathcal{H}}.$$

Since $f'_{t+1} - f_{t+1} = \eta\lambda f'_{t+1} = \eta\lambda f_{t+1}/(1-\eta\lambda)$, we have

$$\|f_{t+1} - f'_{t+1}\|_{\mathcal{H}}^2 = \left(\frac{\eta\lambda}{1-\eta\lambda}\right)^2 \|f_{t+1}\|_{\mathcal{H}}^2$$

and

$$\left\langle f'_{t+1} - f_{t+1}, f_{t+1} - g_t\right\rangle_{\mathcal{H}} = \frac{\eta\lambda}{1-\eta\lambda} \times \left(\|f_{t+1}\|_{\mathcal{H}}^2 - \langle f_{t+1}, g_t\rangle_{\mathcal{H}}\right).$$

Hence, recalling the definition of η, we get

$$\|g_t - f'_{t+1}\|_{\mathcal{H}}^2 - \|g_t - f_{t+1}\|_{\mathcal{H}}^2 = \left(2\eta'\lambda + \eta'^2\lambda^2\right)\|f_{t+1}\|_{\mathcal{H}}^2 - 2\eta'\lambda\langle f_{t+1}, g_t\rangle_{\mathcal{H}}. \quad (40)$$

For the third part of (38), we have

$$\|g_t - f_{t+1}\|_{\mathcal{H}}^2 - \|g_{t+1} - f_{t+1}\|_{\mathcal{H}}^2 = \|g_t\|_{\mathcal{H}}^2 - \|g_{t+1}\|_{\mathcal{H}}^2 + 2\langle g_{t+1} - g_t, f_{t+1}\rangle_{\mathcal{H}}. \quad (41)$$

Substituting (39)–(41) into (38) gives us

$$\begin{aligned}\|g_t - f_t\|_{\mathcal{H}}^2 - \|g_{t+1} - f_{t+1}\|_{\mathcal{H}}^2 &\geq 2\eta'(\sigma_t\mu - l_\mu(g_t(x_t), y_t)) \\ &\quad - 2\eta'(\sigma_t\rho - l_\rho(f_t(x_t), y_t)) - \eta'^2\sigma_t X^2 \\ &\quad + \|g_t\|_{\mathcal{H}}^2 - \|g_{t+1}\|_{\mathcal{H}}^2 + H[f_{t+1}] \end{aligned} \quad (42)$$

where

$$H[f] = \left(2\eta'\lambda + \eta'^2\lambda^2\right)\|f\|_{\mathcal{H}}^2 - 2\eta'\lambda\langle f, g_t\rangle_{\mathcal{H}} + 2\langle g_{t+1} - g_t, f\rangle_{\mathcal{H}}.$$

To bound $H[f_{t+1}]$ from below, we write

$$H[f] = a\|f\|_{\mathcal{H}}^2 - 2\langle r, f\rangle_{\mathcal{H}} = a\left\|f - \frac{r}{a}\right\|_{\mathcal{H}}^2 - \frac{\|r\|_{\mathcal{H}}^2}{a}$$

where $a = 2\eta'\lambda + \eta'^2\lambda^2$, and $r = (1+\eta'\lambda)g_t - g_{t+1}$. Hence

$$\begin{aligned} H[f_{t+1}] &\geq -\frac{\|r\|_{\mathcal{H}}^2}{a} \\ &\geq -\frac{1}{2\eta'\lambda + \eta'^2\lambda^2}\left(\|g_t - g_{t+1}\|_{\mathcal{H}} + \eta'\lambda\|g_t\|_{\mathcal{H}}\right)^2 \\ &= -\frac{1}{2+\eta'\lambda} \times \left(\frac{\|g_t - g_{t+1}\|_{\mathcal{H}}^2}{\eta'\lambda} + 2\|g_t - g_{t+1}\|_{\mathcal{H}}\|g_t\|_{\mathcal{H}} + \eta'\lambda\|g_t\|_{\mathcal{H}}^2\right). \end{aligned} \quad (43)$$

Since $-1/(2+\eta'\lambda) > -1/2$, (42) and (43) give

$$\begin{aligned}\|g_t - f_t\|_{\mathcal{H}}^2 - \|g_{t+1} - f_{t+1}\|_{\mathcal{H}}^2 &\geq -2\eta'(\sigma_t\rho - l_\rho(f_t(x_t), y_t)) \\ &\quad + 2\eta'(\sigma_t\mu - l_\mu(g_t(x_t), y_t)) \\ &\quad - \eta'^2\sigma_t X^2 + \|g_t\|_{\mathcal{H}}^2 - \|g_{t+1}\|_{\mathcal{H}}^2 \\ &\quad - \frac{1}{2}\left(\frac{\|g_{t+1} - g_t\|_{\mathcal{H}}^2}{\eta'\lambda} + 2\|g_t\|_{\mathcal{H}}\|g_{t+1} - g_t\|_{\mathcal{H}} + \eta'\lambda\|g_t\|_{\mathcal{H}}^2\right). \end{aligned} \quad (44)$$

By summing (44) over $t = 1, \ldots, m$ and using the assumption that $\mathbf{g} \in \mathcal{G}(B, D_1, D_2)$, we obtain

$$\begin{aligned}\|g_1 - f_1\|_{\mathcal{H}}^2 - \|g_{m+1} - f_{m+1}\|_{\mathcal{H}}^2 &\geq 2\eta' L_{\mathrm{cum},\rho}[\mathbf{f}, S] - 2\eta' L_{\mathrm{cum},\mu}[\mathbf{g}, S] \\ &\quad + \eta' M_\rho(\mathbf{f}, S)\left(2\mu - 2\rho - \eta' X^2\right) \\ &\quad + \|g_1\|_{\mathcal{H}}^2 - \|g_{m+1}\|_{\mathcal{H}}^2 \\ &\quad - \frac{1}{2}\left(\frac{D_2}{\eta'\lambda} + 2BD_1 + m\eta'\lambda B^2\right). \end{aligned} \quad (45)$$

Now, λ appears only in (45) as a subexpression $Q(\eta'\lambda)$, where $Q(z) = -(D_2/z) - zmB^2$. Since the function $Q(z)$ is maximized for $z = \sqrt{D_2/(mB^2)}$, we choose λ as in (32), which gives $Q(\eta'\lambda) = -2B\sqrt{mD_2}$. We assume $f_1 = 0$; therefore, $\|g_1 - f_1\|_{\mathcal{H}}^2 - \|g_{m+1} - f_{m+1}\|_{\mathcal{H}}^2 \leq \|g_1\|_{\mathcal{H}}^2$. By moving some terms around and estimating $\|g_{m+1}\|_{\mathcal{H}} \leq B$ and $L_{\mathrm{cum},\mu}[\mathbf{g}, S] \leq K$, we get

$$L_{\mathrm{cum},\rho}[\mathbf{f}, S] + M_\rho(\mathbf{f}, S)\left(\frac{\mu - \rho - \eta' X^2}{2}\right) \leq K + \frac{B^2 + B(\sqrt{mD_2} + D_1)}{2\eta'}. \quad (46)$$

To get a bound for margin errors, notice that the value η' given in the theorem satisfies $\mu - \rho - \eta' X^2/2 > 0$. We make the trivial estimate $L_{\mathrm{cum},\rho}[\mathbf{f}, S] \geq 0$, which gives us

$$M_\rho(\mathbf{f}, S) \leq \frac{K}{\mu - \rho - \eta'\frac{X^2}{2}} + \frac{B^2 + B(\sqrt{mD_2} + D_1)}{2\eta'\left(\mu - \rho - \eta'\frac{X^2}{2}\right)}.$$

The bound follows by applying Lemma 7 with $\gamma = \mu - \rho$ and $z = \eta' X^2/2$.

Proof of Theorem 3: The claim for $\rho = 0$ follows directly from Theorem 2. For nonzero ρ, we take (46) as our starting point. We choose $\eta' = 2(\mu - \rho)/X^2$; therefore, the term with $M_\rho(\mathbf{f}, s)$ vanishes, and we get

$$L_{\mathrm{cum},\rho}[\mathbf{f}, S] \leq K + \frac{X^2(B^2 + B(\sqrt{mD_2} + D_1))}{4(\mu - \rho)}. \quad (47)$$

Since $L_{\mathrm{cum},\rho}[\mathbf{f}, S] \geq \rho M(\mathbf{f}, S)$, this implies

$$M(\mathbf{f}, S) \leq \frac{K}{\rho} + \frac{X^2(B^2 + B(\sqrt{mD_2} + D_1))}{4\rho(\mu - \rho)}. \quad (48)$$

The claim follows from Lemma 7 with $\gamma = \mu$ and $z = \mu - \rho$.

Appendix B
Proof of Theorem 4

Without loss of generality, we can assume $g = \hat{g}$, and in particular, $\|g\|_{\mathcal{H}} \leq U$. First, notice that

$$\begin{aligned}\|f_t - g\|_{\mathcal{H}}^2 - \|f_{t+1} - g\|_{\mathcal{H}}^2 &= -\|f_{t+1} - f_t\|_{\mathcal{H}}^2 - 2\langle f_{t+1} - f_t, f_t - g\rangle_{\mathcal{H}} \\ &= -\eta_t^2\|\partial_f R_{\mathrm{inst}}[f, x_t, y_t]|_{f=f_t}\|_{\mathcal{H}}^2 \\ &\quad + 2\eta_t\langle \partial_f R_{\mathrm{inst}}[f, x_t, y_t]|_{f=f_t}, f_t - g\rangle_{\mathcal{H}} \\ &\geq -4\eta_t^2 c^2 X^2 - 2\eta_t(R_{\mathrm{inst}}[g, x_t, y_t] \\ &\quad - R_{\mathrm{inst}}[f_t, x_t, y_t]) \end{aligned} \quad (49)$$

where we used the Lipschitz property of l and the convexity of R_{inst} in its first argument. This leads to

$$\begin{aligned}
&\frac{1}{\eta_t}\|f_t - g\|_{\mathcal{H}}^2 - \frac{1}{\eta_{t+1}}\|f_{t+1} - g\|_{\mathcal{H}}^2 \\
&\quad = \frac{1}{\eta_t}\left(\|f_t - g\|_{\mathcal{H}}^2 - \|f_{t+1} - g\|_{\mathcal{H}}^2\right) \\
&\qquad + \left(\frac{1}{\eta_t} - \frac{1}{\eta_{t+1}}\right)\|f_{t+1} - g\|_{\mathcal{H}}^2 \\
&\quad \geq -4\eta_t c^2 X^2 - 2R_{\text{inst}}[g, x_t, y_t] + 2R_{\text{inst}}[f_t, x_t, y_t] \\
&\qquad + 4U^2\left(\frac{1}{\eta_t} - \frac{1}{\eta_{t+1}}\right)
\end{aligned}$$

since $\|f_{t+1} - g\|_{\mathcal{H}} \leq 2U$. By summing over $t = 1, \ldots, m+1$ and noticing that some terms telescope and $\sum_{t=1}^{m} \eta_t \leq 2\eta m^{1/2}$, we get

$$\begin{aligned}
&\frac{\|f_1 - g\|_{\mathcal{H}}^2}{\eta} - \frac{\|f_{m+1} - g\|_{\mathcal{H}}^2}{\eta_{m+1}} \\
&\quad \geq -8\eta c^2 X^2 m^{1/2} - 2\sum_{t=1}^{m} R_{\text{inst}}[g, x_t, y_t] \\
&\qquad + 2\sum_{t=1}^{m} R_{\text{inst}}[f_t, x_t, y_t] + 4U^2\left(\frac{1}{\eta} - \frac{(m+1)^{1/2}}{\eta}\right).
\end{aligned}$$

The claim now follows by rearranging terms and estimating $\|f_1 - g\|_{\mathcal{H}} \leq U$, $\|f_{m+1} - g\|_{\mathcal{H}}^2 \geq 0$, and $(m+1)^{1/2} - 1 \leq m^{1/2}$.

APPENDIX C
PROOF OF THEOREM 6

First, let us define $t_0(\lambda, \eta, \epsilon)$ to be the smallest possible such that all of the following hold for all $t \geq t_0(\lambda, \eta, \epsilon)$:

- $\eta\lambda t^{-1/2} \leq 1$;
- $\exp(-\eta\lambda t^{\epsilon}) \leq \eta\lambda t^{-1/2}$;
- $\lceil t^{1/2+\epsilon} \rceil \leq 3t/4$.

We use this to estimate $\|\Delta_t\|_{\mathcal{H}}$. If $s_{t+1} = t+1$, then clearly, $\Delta_t = 0$; therefore, we consider the case $t \geq t_0(\lambda, \eta, \epsilon)$. Let $r = t - s_t$ so that $\|\Delta_t\|_{\mathcal{H}} \leq X|\alpha_{r,t}|$. We have $|\alpha_{r,r}| \leq \eta_r c$ and $|\alpha_{r,r+\tau+1}| = (1 - \eta_{r+\tau}\lambda)|\alpha_{r,r+\tau}| \leq (1 - \eta_t\lambda)|\alpha_{r,r+\tau}|$ for $\tau = 0, \ldots, s_t - 1$. Hence

$$|\alpha_{r,t}| \leq \eta_r c(1 - \eta_t\lambda)^{s_t} \leq \eta_r c\left(\left(1 - \frac{\eta\lambda}{t^{1/2}}\right)^{t^{1/2}}\right)^{t^{\epsilon}}.$$

Since $\eta\lambda/t^{1/2} \leq 1$, we have

$$\left(1 - \frac{\eta\lambda}{t^{1/2}}\right)^{\left(t^{1/2}/\eta\lambda\right)} \leq \exp(-1).$$

Therefore, $|\alpha_{r,t}| \leq \eta_r c\exp(-\eta\lambda t^{\epsilon}) \leq \eta_r c\eta\lambda t^{-1/2}$. Finally, since $r \geq t/4$, we have $\eta_r \leq 2\eta_t$; therefore

$$\|\Delta_t\|_{\mathcal{H}} \leq 2\eta_t^2 \lambda c X.$$

In particular, we have $\|\Delta_t\|_{\mathcal{H}} \leq 2\eta_t c X$; therefore

$$\begin{aligned}
\|f_{t+1}\|_{\mathcal{H}} &\leq (1 - \eta_t\lambda)\|f_t\|_{\mathcal{H}} \\
&\quad + \eta_t|l'(f_t(x_t, y))|\|k(x_t, \cdot)\|_{\mathcal{H}} + \|\Delta_t\|_{\mathcal{H}} \\
&\leq (1 - \eta_t\lambda)\|f_t\|_{\mathcal{H}} + 3\eta_t c X.
\end{aligned}$$

Since $f_1 = 0$, we get $\|f_t\|_{\mathcal{H}} \leq 3cX/\lambda$. Again, without loss of generality, we can assume $g = \hat{g}$, and thus, in particular, $\|f_t - g\|_{\mathcal{H}} \leq 4cX/\lambda$.

To estimate the progress at trial t, let $\tilde{f}_{t+1} = f_{t+1} + \Delta_t$ be the new hypothesis before truncation. We write

$$\begin{aligned}
&\|f_t - g\|_{\mathcal{H}}^2 - \|f_{t+1} - g\|_{\mathcal{H}}^2 \\
&\quad = \|f_t - g\|_{\mathcal{H}}^2 - \|\tilde{f}_{t+1} - g\|_{\mathcal{H}}^2 \qquad (50) \\
&\qquad + \|\tilde{f}_{t+1} - g\|_{\mathcal{H}}^2 - \|f_{t+1} - g\|_{\mathcal{H}}^2. \qquad (51)
\end{aligned}$$

To estimate (51), we write

$$\begin{aligned}
&\|\tilde{f}_{t+1} - g\|_{\mathcal{H}}^2 - \|f_{t+1} - g\|_{\mathcal{H}}^2 \\
&\quad = \|(\tilde{f}_{t+1} - f_{t+1}) + (f_{t+1} - g)\|_{\mathcal{H}}^2 - \|f_{t+1} - g\|_{\mathcal{H}}^2 \\
&\quad = 2\langle \Delta_t, f_{t+1} - g\rangle_{\mathcal{H}} + \|\Delta_t\|_{\mathcal{H}}^2 \\
&\quad \geq -2\|\Delta_t\|_{\mathcal{H}}\|f_{t+1} - g\|_{\mathcal{H}} \\
&\quad \geq -16\eta_t^2 c^2 X^2.
\end{aligned}$$

By combining this with the estimate (49) for (50), we get

$$\begin{aligned}
&\|f_t - g\|_{\mathcal{H}}^2 - \|f_{t+1} - g\|_{\mathcal{H}}^2 \\
&\quad \geq -20\eta_t^2 c^2 X^2 - 2\eta_t(R_{\text{inst}}[g, x_t, y_t] - R_{\text{inst}}[f_t, x_t, y_t]).
\end{aligned}$$

Notice the similarity to (49). The rest follows as in the proof of Theorem 4.

ACKNOWLEDGMENT

The authors would like to thank P. Wankadia for help with the implementation and to I. Steinwart and R. Herbrich for comments and suggestions.

REFERENCES

[1] B. Schölkopf and A. J. Smola, *Learning With Kernels*. Cambridge, MA: MIT Press, 2001.
[2] D. J. Sebald and J. A. Bucklew, "Support vector machine techniques for nonlinear equalization," *IEEE Trans. Signal Processing*, vol. 48, pp. 3217–3226, Nov. 2000.
[3] G. S. Kimeldorf and G. Wahba, "Some results on Tchebycheffian spline functions," *J. Math. Anal. Applic.*, vol. 33, pp. 82–95, 1971.
[4] B. Schölkopf, A. Smola, R. C. Williamson, and P. L. Bartlett, "New support vector algorithms," *Neural Comput.*, vol. 12, pp. 1207–1245, 2000.
[5] M. Herbster, "Learning additive models online with fast evaluating kernels," in *Proc. Fourteenth Annu. Conf. Comput. Learning Theory*, vol. 2111, Springer Lecture Notes in Computer Science, D. P. Helmbold and B. Williamson, Eds., 2001, pp. 444–460.
[6] S. V. N. Vishwanathan and A. J. Smola, "Fast kernels for string and tree matching," *Adv. Neural Inform. Process. Syst.*, vol. 15, pp. 569–576, 2003.
[7] G. Cauwenberghs and T. Poggio, "Incremental and decremental support vector machine learning," in *Advances in Neural Information Processing Systems 13*, T. K. Leen, T. G. Dietterich, and V. Tresp, Eds. Cambridge, MA: MIT Press, 2001, pp. 409–415.
[8] L. Csató and M. Opper, "Sparse representation for Gaussian process models," in *Advances in Neural Information Processing Systems 13*, T. K. Leen, T. G. Dietterich, and V. Tresp, Eds. Cambridge, MA: MIT Press, 2001, pp. 444–450.
[9] C. Gentile, "A new approximate maximal margin classification algorithm," *J. Machine Learning Res.*, vol. 2, pp. 213–242, Dec. 2001.
[10] T. Graepel, R. Herbrich, and R. C. Williamson, "From margin to sparsity," in *Advances in Neural Information Processing Systems 13*, T. K. Leen, T. G. Dietterich, and V. Tresp, Eds. Cambridge, MA: MIT Press, 2001, pp. 210–216.
[11] Y. Li and P. M. Long, "The relaxed online maximum margin algorithm," *Machine Learning*, vol. 46, no. 1, pp. 361–387, Jan. 2002.
[12] M. Herbster and M. Warmuth, "Tracking the best linear predictor," *J. Machine Learning Res.*, vol. 1, pp. 281–309, 2001.

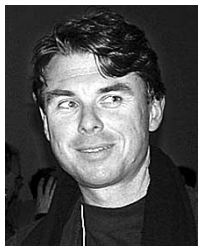
[13] P. Auer and M. Warmuth, "Tracking the best disjunction," *Machine Learning J.*, vol. 32, no. 2, pp. 127–150, 1998.
[14] J. Kivinen, A. J. Smola, and R. C. Williamson, "Large margin classification for moving targets," in *Proc. 13th Int. Conf. Algorithmic Learning Theory*, N. Cesa-Bianchi, M. Numao, and R. Reischuk, Eds., Berlin, Germany, Nov. 2002, pp. 113–127.
[15] C. Mesterharm, "Tracking linear-threshold concepts with Winnow," in *Proc. 15th Annu. Conf. Comput. Learning Theory*, J. Kivinen and B. Sloan, Eds., Berlin, Germany, July 2002, pp. 138–152.
[16] N. Littlestone, "Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm," *Machine Learning*, vol. 2, pp. 285–318, 1988.
[17] O. Bousquet and M. K. Warmuth, "Tracking a small set of experts by mixing past posteriors," *J. Machine Learning Res.*, vol. 3, pp. 363–396, Nov. 2002.
[18] J. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods—Support Vector Learning*, B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds. Cambridge, MA: MIT Press, 1999, pp. 185–208.
[19] M. Vogt, "SMO algorithms for support vector machines without bias term," Technische Univ. Darmstadt, Inst. Automat. Contr., Lab. Contr. Syst. Process Automat., Darmstadt, Germany, 2002.
[20] K. P. Bennett and O. L. Mangasarian, "Robust linear programming discrimination of two linearly inseparable sets," *Optimization Methods Software*, vol. 1, pp. 23–34, 1992.
[21] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Comput.*, vol. 13, no. 7, 2001.
[22] P. J. Huber, "Robust statistics: A review," *Ann. Statist.*, vol. 43, pp. 1041–1067, 1972.
[23] V. Vapnik, S. Golowich, and A. Smola, "Support vector method for function approximation, regression estimation, and signal processing," in *Advances in Neural Information Processing Systems 9*, M. C. Mozer, M. I. Jordan, and T. Petsche, Eds. Cambridge, MA: MIT Press, 1997, pp. 281–287.
[24] S. Haykin, *Adaptive Filter Theory*, Second ed. Englewood Cliffs, NJ: Prentice-Hall, 1991.
[25] B. Schölkopf, R. Herbrich, and A. J. Smola, "A generalized representer theorem," in *Proc. Annu. Conf. Comput. Learning Theory*, 2001, pp. 416–426.
[26] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," in *Advances in Neural Information Processing Systems 14*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds. Cambridge, MA: MIT Press, 2002, pp. 785–792.
[27] R. Herbrich, *Learning Kernel Classifiers: Theory and Algorithms*. Cambridge, MA: MIT Press, 2002.
[28] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: A statistical view of boosting," Dept. Statist., Stanford Univ., Stanford, CA, 1998.
[29] A. B. J. Novikoff, "On convergence proofs on perceptrons," in *Proc. Symp. Math. Theory Automata*, vol. 12, 1962, pp. 615–622.
[30] C. Gentile and N. Littlestone, "The robustness of the p-norm algorithms," in *Proc. 12th Annu. Conf. Comput. Learning Theory*, New York, NY, 1999, pp. 1–11.
[31] Y. Freund and R. E. Schapire, "Large margin classification using the perceptron algorithm," *Machine Learning*, vol. 37, no. 3, pp. 277–296, 1999.
[32] P. Auer, N. Cesa-Bianchi, and C. Gentile, "Adaptive and self-confident on-line learning algorithms," *J. Comput. Syst. Sci.*, vol. 64, no. 1, pp. 48–75, Feb. 2002.
[33] N. Cesa-Bianchi, P. Long, and M. Warmuth, "Worst-case quadratic loss bounds for on-line prediction of linear functions by gradient descent," *IEEE Trans. Neural Networks*, vol. 7, pp. 604–619, May 1996.
[34] M. K. Warmuth and A. Jagota. Continuous and discrete time nonlinear gradient descent: Relative loss bounds and convergence. presented at Fifth Int. Symp. Artif. Intell. Math.. [Online]. Available: http://rutcor.rutgers.edu/~amai
[35] N. Cesa-Bianchi, A. Conconi, and C. Gentile, "On the generalization ability of on-line learning algorithms," in *Advances in Neural Information Processing Systems 14*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds. Cambridge, MA: MIT Press, 2002, pp. 359–366.

Jyrki Kivinen received the M.Sc. degree in 1989 and the Ph.D. degree in 1992, both in computer science, from the University of Helsinki, Helsinki, Finland.

He has held various teaching and research appointments at University of Helsinki and has visited the University of California at Santa Cruz and the Australian National University, Canberra, as a postdoctoral fellow. Since 2003, he has been a Professor at the University of Helsinki. His scientific interests include machine learning and algorithm theory.

Alexander J. Smola received the Masters degree in physics from the TU Munich, Munich, Germany, in 1996 and the Ph.D. degree in computer science from the TU Berlin, Berlin, Germany, in 1998.

Since 1999, he has been at the Australian National University, Canberra, where is a fellow with the Research School of Information Sciences and Engineering. He is also a senior researcher at National ICT Australia, Canberra. He is a member of the editorial board of the *Journal of Machine Learning Research* and Kernel-Machines.org. He is also the coauthor of *Learning with Kernels* (Cambridge, MA: MIT Press, 2002). His scientific interests are in machine learning, vision, and bioinformatics.



Robert C. Williamson (M'91) received the Ph.D. degree in electrical engineering from the University of Queensland, Brisbane, Australia, in 1990.

Since then, he has been with the Australian National University, Canberra, where he is a Professor with the Research School of Information Sciences and Engineering. He is the director of the Canberra Laboratory of National ICT Australia, president of the Association for Computational Learning Theory, and a member of the the editorial board of the *Journal of Machine Learning Research*. His scientific interests include signal processing and machine learning.