Learning additive models online with fast evaluating kernels*

Mark Herbster

Department of Computer Science
University College London
Gower Street
London WC1E 6BT, UK
M.Herbster@cs.ucl.ac.uk

Abstract. We develop three new techniques to build on the recent advances in online learning with kernels. First, we show that an exponential speed-up in prediction time per trial is possible for such algorithms as the Kernel-Adatron, the Kernel-Perceptron, and ROMMA for specific additive models. Second, we show that the techniques of the recent algorithms developed for online linear prediction when the best predictor changes over time may be implemented for kernel-based learners at no additional asymptotic cost. Finally, we introduce a new online kernel-based learning algorithm for which we give worst-case loss bounds for the ϵ-insensitive square loss.

Introduction

The aim of this research is to make online learning with kernels more practical. We do this in three ways. Initially, in Part 1 we present an algorithmic innovation which speeds computation time for certain kernels designed for additive modeling. This works with a broad class of algorithms such as those in [14, 13, 25, 15] and the algorithms presented in Part 2. Specifically, with an additive spline kernel, on trial t we may predict and then update our hypothesis with $O(\log t)$ computations as opposed to the usual $O(t)$ computation time. In Part 2 we present a unified analysis of two algorithms, one for classification and one for regression. The classification algorithm is a simple variant of the well-known Perceptron [34] algorithm. We then present a novel algorithm for regression with the useful property that it updates *conservatively*. We give a simple total loss bound for this algorithm with the ϵ-insensitive square loss. Finally, in Part 2 we show that the recent results of [21] may be applied and implemented efficiently for online learning with kernels; this allows us to prove *local* loss bounds. These bounds differ from total loss bounds in that they hold for any segment of the sequence. Thus they are particularly relevant to online learning.

As discussed above the paper is divided into two parts. Each part is wholly self-contained. However, the techniques presented in both parts are easily combined.

* Mark Herbster was supported by ESPRC grant GR/M15972.

1 Fast kernel evaluation for additive models

Reproducing kernels were initially introduced in the machine literature for use with the perceptron algorithm in [1]. The current popularity of kernel-based methods is due to the successful integration of the "optimal separating hyperplane" method [41] with a kernel transformation to create the support vector machine [7]. Recently there has a been a renewed interest in online kernel-based algorithms both as a proxy for batch learning and in their own right. The use of online kernel algorithms for batch learning [13, 14, 25, 33] has been proposed for two essential reasons: first, their simplicity of implementation; and second, whereas typical batch methods require memory quadratic in the dataset size, typical online algorithms require memory linear in the dataset size. Thus for the largest of datasets online algorithms may be the best option. A key obstacle facing online kernel-based learners is the fact that on trial t prediction typically requires $O(t)$ time. In this part of the paper we tread a middle ground, in that we restrict the hypothesis class of kernel-based learners to a class of *additive* models, but in compensation on trial t we may predict in $O(\log t)$ time, a per-trial exponential speed-up. Another major problem with online kernel-based learners is that the internal hypothesis representation on trial t is potentially of $O(t)$ size; thus an important goal of online kernel-based learners is to restrict the representation to a finite size. In this paper we do not address this problem, but in preliminary research [38] a broad framework was presented for online kernel-based learners motivated by a regularized stochastic gradient descent update; it was also pointed out that the online regularization led to weight decay and thus potentially to bounds for an algorithm with finite memory size.

Additive modeling is a well-known technique from statistics [17] (also see [16]) where we suppose the underlying hypothesis class is essentially additive, i.e.,

$$f(\boldsymbol{x}) = \sum_{i=1}^{d} f_i(x_i)$$

for d-dimensional data. In effect we attempt to learn a different smooth function for each dimension. If we expect lower order correlations among the components, it may be reasonable to explicitly expand the coordinate vectors by multiplication. However, the technique presented in this section is inappropriate for highly correlated data, as in the classic postal digit recognition task [24].

The kernels and data structures presented in this part are compatible with a number of online kernel-based learners such as presented in [13, 14, 25][1]. The key features that are required to make use of techniques of this part is an online algorithm that changes its dual representation a single term at a time, and that the computations lead to a single term being changed are not "too complex". In Part 2 we give two sample algorithms which are compatible with the speed-up technique presented in this part.

[1] For both the Kernel-Perceptron and ROMMA, we refer to their non-voted versions.

For simplicity we illustrate the idea in the batch setting. The well-known dual representation of a kernel learner's hypothesis is

$$f(x) = \sum_{i=1}^{m} \alpha_i K(x_i, x), \tag{1}$$

a regression prediction is simply $f(x)$, whereas for classification the prediction is $\text{sign}(f(x))$. In either case the straightforward method of evaluating $f(x)$ requires m steps. In this paper we show that for certain kernels only $O(\log m)$ steps are needed after a preprocessing step. For example, consider the single coordinate radial basis kernel $K(p, q) = e^{-\sigma|p-q|}$. Now consider that a batch learning procedure has determined the $2m$ parameters $\alpha_1, \ldots, \alpha_m, x_1, \ldots, x_m$ of the hypothesis f. In Figure 1 we show that $f(x)$ as represented in Equation (1) may be repeatedly evaluated for any x in $O(\log m)$ steps after a $O(m \log m)$ preprocessing step. Thus rapid evaluation of the dual form (Equation (1)) with

Preprocessing:

1. Sort in ascending order $x_1, \ldots, x_m$ then relabel as $x'_1 < \ldots < x'_m$ and set $x'_0 = -\infty$.
2. Set $\alpha'_j = \alpha_i$ when $x'_j = x_i$ for $i = 1, \ldots, m$.
3. Set $\text{lhs}_{\{e^{\sigma p}\}}[i] = \sum_{j=1}^{i} \alpha'_j e^{\sigma x'_j}$ for $i = 0, \ldots, m$.
4. Set $\text{rhs}_{\{e^{-\sigma p}\}}[i] = \sum_{j=i+1}^{m} \alpha'_j e^{-\sigma x'_j}$ for $i = 0, \ldots, m$.

Evaluation:
We may evaluate f on an arbitrary x as follows:

1. With a binary search let $j^* = \max\{j : x'_j \leq x\}$.
2. Let $f(x) = \text{lhs}_{\{e^{\sigma p}\}}[j^*] e^{-\sigma x} + \text{rhs}_{\{e^{-\sigma p}\}}[j^*] e^{\sigma x}$.

Fig. 1. Evaluating $f(x) = \sum_{i=1}^{m} \alpha_i e^{\sigma|x_i - x|}$ in $O(\log m)$ time

$K(p, q) = e^{-\sigma|p-q|}$ requires $O(m)$ space but only $O(\log m)$ time. For the sake of comparison this kernel may be viewed as intermediate between the the kernels $e^{-\sigma(-p-q)}$ and $e^{-\sigma(p-q)^2}$. The first kernel is a special case of kernels in the form $K_k(p, q) = k(p)k(q)$, which after preprocessing may be evaluated in $O(1)$ time and represented in $O(1)$ space, since

$$f(x) = \sum_{i=1}^{m} \alpha_i K_k(x_i, x) = \left[\sum \alpha_i k(x_i)\right] k(x).$$

Whereas for the kernel $e^{-\sigma(p-q)^2}$ it is an open problem if Equation (1) can be evaluated for any x in less than $O(m)$ steps with a polynomial preprocessing step[2].

[2] All three kernels are reproducing kernels, however the Hilbert space induced by the kernel $e^{-\sigma(-p-q)}$ is 1-dimensional. While the other two induced Hilbert spaces are infinite-dimensional.

A spline kernel $K : (0,\infty) \times (0,\infty) \to \Re$ of order z with an infinite number of knots [39, 22] is defined by

$$K_d(p,q) = \sum_{r=0}^{z} \frac{\binom{z}{r}}{2z-r+1} \min(p,q)^{2z-r+1} |q-p|^r + \sum_{r=0}^{z} p^r q^r. \tag{2}$$

In the short version of the paper we will only consider linear splines ($z = 1$) which fit the data with a piecewise cubic polynomial between knots. The linear spline kernel is then

$$K_1(p,q) = 1 + pq + \frac{1}{2}|q-p| \min(p,q)^2 + \frac{1}{3}\min(p,q)^3 \tag{3}$$

$$= 1 + pq + \begin{cases} \frac{1}{2}p^2 q - \frac{5}{6}p^3 & p \le q \\ \frac{1}{2}pq^2 - \frac{5}{6}q^3 & p > q. \end{cases} \tag{4}$$

Following the schema of Figure 1, we may perform evaluations of the dual representation (see Equation (1)) in $O(\log m)$ time after an $O(m \log m)$ preprocessing step by creating 4 cached values for each data point and two globally cached values. Thus assuming the notation of steps 1 and 2 of Figure 1, we define for $i = 0, \ldots, m$,

$$\begin{array}{lll} \mathrm{lhs}_{\{\frac{1}{2}p^2\}}[i] = \frac{1}{2}\sum_{j=1}^{i} \alpha'_j (x'_j)^2 & ; & \mathrm{lhs}_{\{-\frac{5}{6}p^3\}}[i] = -\frac{5}{6}\sum_{j=1}^{i} \alpha'_j (x'_j)^3 \\ \mathrm{rhs}_{\{\frac{1}{2}p\}}[i] = \frac{1}{2}\sum_{j=i+1}^{m} \alpha'_j x'_j & ; & \mathrm{rhs}_{\{-\frac{5}{6}\}}[i] = -\frac{5}{6}\sum_{j=i+1}^{m} \alpha'_j \\ \mathrm{gs}_{\{1\}} = \sum_{j=1}^{m} \alpha'_j & ; & \mathrm{gs}_{\{p\}} = \sum_{j=1}^{m} \alpha'_j x'_j. \end{array}$$

We can then evaluate Equation (1) with linear splines in $O(\log m)$ time by finding j^* (see Figure 1), then computing

$$f(x) = \mathrm{gs}_{\{1\}} + \mathrm{gs}_{\{p\}} x + \mathrm{lhs}_{\{\frac{1}{2}p^2\}}[j^*]x + \mathrm{lhs}_{\{-\frac{5}{6}p^3\}}[j^*] + \mathrm{rhs}_{\{\frac{1}{2}p\}}[j^*]x^2 + \mathrm{rhs}_{\{-\frac{5}{6}\}}[j^*]x^3.$$

The key to our method, very loosely, is that the kernel must be separable into left-hand sides and right-hand sides such that a linear combination of either side is quick to evaluate; a technical discussion follows. Let $\chi_p(q) = \begin{cases} 1 & q \le p \\ 0 & q > p \end{cases}$, denote a heavyside function. Then given a kernel[3] $K(p,q) : \Re \times \Re \to \Re$ we split it into "left" and "right" functions, $k_p^L = K(p,\cdot)\chi_p(\cdot)$ and $k_p^R = K(p,\cdot)(1-\chi_p(\cdot))$; thus $K(a,b) = k_a^L(b) + k_a^R(b)$. Define the vector space $F_x^L = \text{span}\,\{k_y^L \chi_x : y \in [x,\infty) \in \Re\}$. Suppose there exists a vector space F^L with basis $\phi_1^L, \ldots, \phi_d^L$ such that $F_x^L \subset \text{span}\,\{\phi_i^L \chi_x : i = 1, \ldots, d\}$ for all $x \in \Re$ (further suppose F^L is of the least dimension such that the former holds). Then we say that the Hilbert space $\mathcal{H}_K$ induced by K has *left dimension* d; the *right dimension* is defined analogously. Without loss of generality assume that both the left and right dimension are d; then $K(p,\cdot)$ may be expanded in terms of the left and right bases, i.e.,

$$K(p,\cdot) = [\sum_{i=1}^{d} \beta_i^L \phi_i^L(\cdot)]\chi_p(\cdot) + [\sum_{j=1}^{d} \beta_j^R \phi_j^R(\cdot)](1-\chi_p(\cdot))$$

[3] The domain may be any ordered set for concreteness we choose $\Re$.

and if the constants (implicitly dependent on p) $\beta_1^L, \ldots, \beta_d^L, \beta_1^R, \ldots, \beta_d^R$ are easily computed then the techniques described below may be applied to compute (1) on trial m in $O(d \log m)$ steps.

The extension of this method to additive multicoordinate kernels is straightforward. Given a 1-coordinate kernel $K(p, q)$ the additive c-coordinate kernel is simply

$$K(\boldsymbol{p}, \boldsymbol{q}) = \sum_{i=1}^{c} K(p_i, q_i).$$

Since each coordinate is additively independent we can apply the above technique in each coordinate independently, which leads to a cost of $O(cd \log m)$ to evaluate a point after preprocessing.

In an online setting it is possible to use a balanced binary tree (e.g., a red-black tree) as the base data structure, to evaluate Equation (1) for the above kernels in $O(d \log m)$ steps; then if the sum consists of m terms a new term may be added to the structure in $O(d \log m)$ steps. The following is a sketch of how this may be done. In the balanced binary tree each node $i = 1, \ldots, m$ will contain a *key* x_i, and $2d$ *values* $\alpha_i \beta_{i,j}^L, \alpha_i \beta_{i,j}^R$ for each $(j = 1, \ldots, d, \{L, R\})$. For each of the values there is also an *augmented value* which is the sum of the values $\alpha_{i'} \beta_{i',j}$ in the subtree rooted at i (in [9, Theorem 15.1] it is demonstrated that these augmented values may be implemented at no additional asymptotic cost for operations on the balanced tree). Given the existence of these augmented value sums, the $2d$ *derived sums* $\sum_{i:x_i \leq x} \alpha_i \beta_{i,j}^L$ and $\sum_{i:x_i > x} \alpha_i \beta_{i,j}^R$ may each then be computed $O(d \log m)$ steps. Thus evaluation of (1) and the addition of a new term may be accomplished in $O(d \log m)$ steps. We provide further details in the full paper.

2 Online algorithms

2.1 Preliminaries

A Hilbert space, in this paper, denotes a complete inner product space, which may be finite or infinite dimensional; thus $\Re^n$ is a Hilbert space. The notation $\langle \boldsymbol{v}, \boldsymbol{w} \rangle$ indicates the inner product between $\boldsymbol{v}$ and $\boldsymbol{w}$. The set $\mathcal{H}$ always denotes an arbitrary Hilbert space.

2.2 Introduction

We consider the following on-line learning model based on a model introduced by Littlestone [27, 26, 28]. Learning proceeds in trials $t = 1, 2, \ldots, \ell$. The algorithm maintains a parameter vector (hypothesis), denoted by $\boldsymbol{w}_t$. In each trial the algorithm receives a *pattern* $\mathbf{x}_t$. It then produces some action or a prediction denoted $\hat{y}_t$, a function of current pattern $\mathbf{x}_t$ and hypothesis $\boldsymbol{w}_t$. Finally, the algorithm receives an *outcome* y_t, and incurs a loss $L(y_t, \hat{y}_t)$ measuring the discrepancy between y_t and $\hat{y}_t$.

In this part we give algorithms for classification and regression. For classification we predict with $\hat{y}_t = \text{sign}(\langle \boldsymbol{w}_t, \mathbf{x}_t \rangle)$, while for regression $\hat{y}_t = \langle \boldsymbol{w}_t, \mathbf{x}_t \rangle$ and we assign loss with

$$L_{\text{m}}(y_t, \hat{y}_t) = \begin{cases} 0 & y_t = \hat{y}_t \\ 1 & y_t \neq \hat{y}_t \end{cases} \tag{5}$$

$$L_{\text{sq},\epsilon}(y_t, \hat{y}_t) = \begin{cases} 0 & |y_t - \hat{y}_t| \leq \epsilon \\ (|y_t - \hat{y}_t| - \epsilon)^2 & |y_t - \hat{y}_t| > \epsilon \end{cases} \tag{6}$$

for the mistake counting loss and the ϵ-insensitive square loss for classification and regression, respectively. The mistake counting loss is a natural measure of discrepancy. The ϵ-insensitive square loss may appear less natural, but consider the following example. Suppose that a robot arm must place a peg into a hole slightly larger than the peg. If the peg is placed in the hole there is no loss; otherwise it is necessary to pay the squared distance from the boundary of the hole to reorient the arm. Thus the ϵ-insensitive square loss is potentially appropriate for situations where there is a natural tolerance for the "correct" response. The ϵ-insensitive linear or quadratic loss is often used in batch learning for support vector regression [40].

In the usual methodology of worst-case loss bounds the total loss of the algorithm is expressed as a function of the total loss of any member in a comparison class of predictors [27]. Surprisingly, such bounds are achievable even when there are no probabilistic assumptions made on the sequence of examples; some prominent results are found in [26, 42, 8, 18, 23, 19, 44]. In this paper we consider a simplification of the above goal, i.e., we give bounds on the loss of algorithm in terms of any member of the *realizable* set of predictors, rather than the whole comparison class. The realizable set are those predictors that "perfectly" fit the data. For classification the realizable set are those predictors that separate the data with a given minimum margin. For regression, it is the set of predictors for which the ϵ-insensitive square loss is zero over the data sequence. The *realizabilty* condition is certainly a limitation on the bounds, particularly in the classification case. However, it is less of a limitation in the regression case, as there necessarily exists an ϵ such that the data will be realizable. In both cases, however, the realizability restriction is less onerous with the *kernel* transformation, since the hypotheses' classes are then much richer and there is the recent technique in [37] to incorporate noise tolerance into a kernel by mixing it with a delta function.

The key tool which we use to repeatedly construct updates for our algorithms is *projection* as is defined below.

Definition 1. *Given a Hilbert space $\mathcal{H}$, the projection of a point $\boldsymbol{w} \in \mathcal{H}$ onto a closed convex nonempty set $\Gamma \subset \mathcal{H}$ is defined by:*

$$\mathcal{P}_\Gamma(\boldsymbol{w}) = \underset{\boldsymbol{u} \in \Gamma}{arg\,min} \|\boldsymbol{u} - \boldsymbol{w}\|. \tag{7}$$

The existence and uniqueness of projection is well-known (e.g. [35, Theorem 4.1]); in this paper the needed projections are always simple to compute. In the

full version we give proofs for the methods of computation given for various projections. Given the definition of the projection above, we may give the following well-known version of the Pythagorean Theorem.

Theorem 1. *Given a Hilbert space $\mathcal{H}$, a point $\boldsymbol{w} \in \mathcal{H}$, a closed convex set $\Gamma \subset \mathcal{H}$, and $\boldsymbol{u} \in \Gamma$, then*

$$\|\boldsymbol{u} - \boldsymbol{w}\|^2 \geq \|\boldsymbol{u} - \mathcal{P}_\Gamma(\boldsymbol{w})\|^2 + \|\mathcal{P}_\Gamma(\boldsymbol{w}) - \boldsymbol{w})\|^2. \tag{8}$$

In the special case where Γ is an affine set the above becomes an equality.

A hyperplane is an example of an affine set. The Pythagorean Theorem is the main tool used to prove bounds for the algorithms given in this part.

2.3 Online algorithms for regression and classification

In this section we give two online algorithms (see Figure 2) for classification and regression, and prove worst-case loss bounds. These algorithms are based on the Prototypical projection algorithm (see Figure 3) which is a relatively well-known technique from the convex optimization community. An early reference for a version of this algorithm is found in the work of Von Neumann [31]. Bauschke and Borwein [5] present a broadly generalized version of this algorithm and a review of its many applications. The first application of this algorithm in the machine learning literature was by Faber and Mycielski [11] to proving worst-case square loss bounds for regression in the noise-free case; Cesa-Bianchi et. al. [8] generalized this work to noisy data with the GD algorithm. In this section we will discuss the application of the prototypical projection algorithm to classification and regression, producing a simple variant of the Perceptron algorithm [34] and a new online algorithm for regression with noisy data.

The following lemma regarding the convergence of the prototypical projection algorithm is well-known.

Lemma 1. *Given a sequence of convex set $\{\mathcal{U}_1, \ldots, \mathcal{U}_\ell\}$ and a start vector $\boldsymbol{w}_1$ as input to the prototypical projection algorithm (see Figure 3) the following inequality holds*

$$\sum_{t=1}^{\ell} \|\boldsymbol{w}_t - \boldsymbol{w}_{t+1}\|^2 \leq \|\boldsymbol{u} - \boldsymbol{w}_1\|^2 \tag{13}$$

for all $\boldsymbol{u} \in \bigcap_{t=1}^{\ell} \mathcal{U}_t$.

Proof. On any trial t the Pythagorean Theorem 1 implies the inequality

$$\|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\|^2 \leq \|\boldsymbol{u} - \boldsymbol{w}_t\|^2 - \|\boldsymbol{u} - \boldsymbol{w}_{t+1}\|^2$$

for all $\boldsymbol{u}$ such that $\boldsymbol{u} \in \mathcal{U}_t$. Summing the previous inequality over all trials $t = 1, \ldots, \ell$ we have

$$\sum_{t=1}^{\ell} \|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\|^2 \leq \|\boldsymbol{u} - \boldsymbol{w}_1\|^2 - \|\boldsymbol{u} - \boldsymbol{w}_{\ell+1}\|^2$$

<table>
<tr><th colspan="3">Algorithms for arbitrary Hilbert Space $\mathcal{H}$</th></tr>
<tr><td></td><td>Classification</td><td>ϵ-insensitive Regression</td></tr>
<tr><td>Input:</td><td>$\{(\boldsymbol{x}_1,y_1),\ldots,(\boldsymbol{x}_\ell,y_\ell)\}\in(\mathcal{H},\{-1,1\})^\ell$</td><td>$\{(\boldsymbol{x}_1,y_1),\ldots,(\boldsymbol{x}_\ell,y_\ell)\}\in(\mathcal{H},\Re)^\ell$</td></tr>
<tr><td>Initialization:</td><td>$\boldsymbol{w}_1=\mathbf{0}$</td><td>$\boldsymbol{w}_1=\mathbf{0}$, choose $\epsilon>0$</td></tr>
<tr><td>Prediction:</td><td colspan="2">Upon receiving the tth instance $\boldsymbol{x}_t$,
set $\overline{y}_t=\langle\boldsymbol{w}_t,\boldsymbol{x}_t\rangle$ then give the prediction:</td></tr>
<tr><td></td><td>$\hat{y}_t=\mathrm{sign}(\overline{y}_t)$</td><td>$\hat{y}_t=\overline{y}_t$</td></tr>
<tr><td>Update:</td><td colspan="2">Project $\boldsymbol{w}_t$ onto the tth feasible set $\mathcal{U}_t$, $\boldsymbol{w}_{t+1}=\mathcal{P}_{\mathcal{U}_t}(\boldsymbol{w}_t)$</td></tr>
<tr><td>Feasible set:</td><td>$\mathcal{U}_t=\{\boldsymbol{v}:\langle\boldsymbol{v},\mathbf{x}_t\rangle y_t\geq 1\}$</td><td>$\mathcal{U}_t=\{\boldsymbol{v}:\langle\boldsymbol{v},\mathbf{x}_t\rangle\in[y_t-\epsilon,y_t+\epsilon]\}$</td></tr>
<tr><td>Update Eq:</td><td>if $\overline{y}_t y_t\geq 1$ then
$\quad\boldsymbol{w}_{t+1}=\boldsymbol{w}_t$
else
$\quad\boldsymbol{w}_{t+1}=\boldsymbol{w}_t+\frac{y_t-\overline{y}_t}{\|\boldsymbol{x}_t\|^2}\boldsymbol{x}_t$ (9)</td><td>if $\overline{y}_t\in[y_t-\epsilon,y_t+\epsilon]$ then
$\quad\boldsymbol{w}_{t+1}=\boldsymbol{w}_t$
else
$\quad s=\mathrm{sign}(\overline{y}_t-(y_t-\epsilon))$
$\quad\boldsymbol{w}_{t+1}=\boldsymbol{w}_t+\frac{y_t+s\epsilon-\overline{y}_t}{\|\boldsymbol{x}_t\|^2}\boldsymbol{x}_t$ (10)</td></tr>
<tr><th colspan="3">Algorithms for data mapped to RKHS $\mathcal{H}_K$ via kernel $K:\mathcal{E}\times\mathcal{E}\rightarrow\Re$</th></tr>
<tr><td>Input:</td><td>$\{(x_1,y_1),\ldots,(x_\ell,y_\ell)\}\in(\mathcal{E},\{-1,1\})^\ell$</td><td>$\{(x_1,y_1),\ldots,(x_\ell,y_\ell)\}\in(\mathcal{E},\Re)^\ell$</td></tr>
<tr><td>Initialization:</td><td>$\boldsymbol{w}_1=\mathbf{0}$ ($\alpha_1=0$)</td><td>$\boldsymbol{w}_1=\mathbf{0}$ ($\alpha_1=0$), choose $\epsilon>0$</td></tr>
<tr><td>Prediction:</td><td colspan="2">Upon receiving the tth instance x_t,
set $\overline{y}_t=\boldsymbol{w}_t(\boldsymbol{x}_t)=\sum_{i=1}^{t-1}\alpha_i K(x_i,x_t)$ then give the prediction:</td></tr>
<tr><td></td><td>$\hat{y}_t=\mathrm{sign}(\overline{y}_t)$</td><td>$\hat{y}_t=\overline{y}_t$</td></tr>
<tr><td>Update:</td><td colspan="2">Project $\boldsymbol{w}_t$ onto the tth feasible set $\mathcal{U}_t$, $\boldsymbol{w}_{t+1}=\mathcal{P}_{\mathcal{U}_t}(\boldsymbol{w}_t)$</td></tr>
<tr><td>Feasible set:</td><td>$\mathcal{U}_t=\{\boldsymbol{v}:\boldsymbol{v}(x_t)y_t\geq 1\}$</td><td>$\mathcal{U}_t=\{\boldsymbol{v}:\boldsymbol{v}(x_t)\in[y_t-\epsilon,y_t+\epsilon]\}$</td></tr>
<tr><td>Update Eq:</td><td>if $\overline{y}_t y_t\geq 1$ then
$\quad\alpha_t=0$
else
$\quad\alpha_t=\frac{y_t-\overline{y}_t}{K(x_t,x_t)}$ (11)</td><td>if $\overline{y}_t\in[y_t-\epsilon,y_t+\epsilon]$ then
$\quad\alpha_t=0$
else
$\quad s=\mathrm{sign}(\overline{y}_t-(y_t-\epsilon))$
$\quad\alpha_t=\frac{y_t+s\epsilon-\overline{y}_t}{K(x_t,x_t)}$ (12)</td></tr>
<tr><td></td><td colspan="2">$\boldsymbol{w}_{t+1}=\boldsymbol{w}_t+\alpha_t K(x_t,\cdot)$</td></tr>
</table>

Fig. 2. Projection algorithms for classification and regression

for all $\boldsymbol{u}\in\bigcap_{t=1}^{\ell}\mathcal{U}_t$. Dropping the final term of the above inequality proves the lemma.

An implication of the above lemma is that if $\bigcap_{t=1}^{\ell}\mathcal{U}_t$ is nonempty, and if we repeatedly cycle through the input $\{\mathcal{U}_1,\ldots,\mathcal{U}_\ell\}$, then the Cauchy sequence $\{\boldsymbol{w}_1,\boldsymbol{w}_2,\ldots\}$ generated by the prototypical projection algorithm necessarily converges to a point in the above intersection.

We use the prototypical projection algorithm to produce online learning algorithms by associating the *feasible set* sequence $\{\mathcal{U}_1,\ldots,\mathcal{U}_\ell\}$ with the example sequence $\{(\boldsymbol{x}_1,y_1),\ldots,(\boldsymbol{x}_\ell,y_\ell)\}$. Each feasible set $\mathcal{U}_t$ consists of those hypothesis vectors which are "compatible" with the last example. In the projection algorithm for classification, the feasible set is the halfspace of vectors

Input: A sequence of closed convex sets $\{\mathcal{U}_1, \ldots, \mathcal{U}_\ell\} \subset \mathcal{H}^\ell$ and a point $\boldsymbol{w}_1 \in \mathcal{H}$ where $\mathcal{H}$ is a Hilbert space.
Update: $\boldsymbol{w}_{t+1} = \mathcal{P}_{\mathcal{U}_t}(\boldsymbol{w}_t)$

Fig. 3. Prototypical projection algorithm

$\mathcal{U}_t = \{\boldsymbol{v} : \langle \boldsymbol{v}, \mathbf{x}_t \rangle y_t \geq 1\}$ which classify the last example correctly with a margin[4] greater than 1. For regression with the ϵ-insensitive square loss the feasible set is the "hyper-rectangle" of vectors $\mathcal{U}_t = \{\boldsymbol{v} : \langle \boldsymbol{v}, \mathbf{x}_t \rangle \in [y_t - \epsilon, y_t + \epsilon]\}$. These are the vectors which classify the last example correct up to an absolute error of most epsilon. In order to prove bounds for these algorithms, we lower bound each term of the sum in Equation (13) with a term that is the ratio of the loss of the algorithm on that example with the squared norm of the instance. Thus applying the lower bounds which are given in the Lemmas 2 and 3 in combination with Lemma 1 proves the worst-case loss bounds for classification and ϵ-insensitive regression in Theorems 2 and 3 respectively.

Lemma 2. *On any trial t the mistake-counting loss of the projection algorithm for classification may be bounded by*

$$\frac{L_m(y_t, \hat{y}_t)}{\|\mathbf{x}_t\|^2} \leq \|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\|^2 \tag{14}$$

Proof. Consider two cases. First, if $L_{\mathrm{m}}(y_t, \hat{y}_t) = 0$ the lemma is trivial, otherwise we have

$$\frac{|y_t - \overline{y}_t|^2}{\|\mathbf{x}_t\|^2} = \|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\|^2$$

by Update (9). Since $L_{\mathrm{m}}(y_t, \hat{y}_t) \leq |y_t - \overline{y}_t|^2$ when $y_t \neq \hat{y}_t$, the lemma is proven.

Lemma 3. *On any trial t the ϵ-insensitive square loss of the projection algorithm for regression may be bounded by*

$$\frac{L_{sq,\epsilon}(y_t, \hat{y}_t)}{\|\mathbf{x}_t\|^2} \leq \|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\|^2 \tag{15}$$

Theorem 2. *Given a sequence of examples $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_\ell, y_\ell)\} \in (\mathcal{H}, \{-1, 1\})^\ell$ and a start vector $\boldsymbol{w}_1 \in \mathcal{H}$, let $R = \max_{t=1,\ldots,\ell} \|\boldsymbol{x}_t\|$ then the cumulative mistakes of the projection algorithm for classification is bounded by*

$$\sum_{t=1}^{\ell} L_m(y_t, \hat{y}_t) \leq R^2 \|\boldsymbol{u} - \boldsymbol{w}_1\|^2 \tag{16}$$

[4] Margin here has a different meaning than typically used in discussion of the Perceptron algorithm or the Maximal margin algorithm. Generally the margin is allowed to vary while the norm of the classifier is fixed to less than 1. In our discussion the margin is fixed to be larger than 1 while the norm of the classifier is allowed to vary. We choose these semantics to indicate the parallels between classification and regression.

for all $\boldsymbol{u}$ *such that* $\langle \boldsymbol{u}, \boldsymbol{x}_t \rangle y_t \geq 1$ *for* $t = 1, \ldots, \ell$.

This algorithm for classification is a simple variant of Rosenblatt's perceptron algorithm [34], and the bound proven, though differing in form, is the same as that proven by Novikoff [32]. This algorithm is equivalent to the perceptron if the data is always normalized and we also update when correct but $y_t \overline{y}_t < 1$. As given in the conditions of theorem the algorithm only provides a bound when the data is linearly separable; recently, however, Freund and Schapire [13] have proven a bound for the perceptron algorithm (in $\Re^n$) when the data is linearly inseparable; this technique is further extended in [37] for inseparable data in more general kernel spaces.

Theorem 3. *Given a sequence of examples* $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_\ell, y_\ell)\} \in (\mathcal{H}, \Re)^\ell$ *and a start vector* $\boldsymbol{w}_1 \in \mathcal{H}$, *let* $R = \max_{t=1,\ldots,\ell} \|\boldsymbol{x}_t\|$. *Then the cumulative* ϵ*-insensitive square loss of the projection algorithm for regression is bounded by*

$$\sum_{t=1}^{\ell} L_{sq,\epsilon}(y_t, \hat{y}_t) \leq R^2 \|\boldsymbol{u} - \boldsymbol{w}_1\|^2 \tag{17}$$

for all $\boldsymbol{u}$ *such that* $\langle \boldsymbol{u}, \boldsymbol{x}_t \rangle \in [y_t - \epsilon, y_t + \epsilon]$ *for* $t = 1, \ldots, \ell$.

For the special case when $\epsilon = 0$ this theorem was first proven in [11]. The GD algorithm [8] is also designed for online regression on noisy data, a salient feature of the GD algorithm is that given an upper bound on R (as defined above) the algorithm may be tuned so that a worst-case bound on the usual square loss is given for any data sequence, whereas the projection algorithm for regression requires for its bound the assumption that $\sum_{t=1}^{\ell} \mathcal{U}_t$ is non-empty.

Two useful properties of the projection algorithm for regression are that it is convergent (see the discussion following Lemma 1), and that like the perceptron algorithm it is *conservative*, i.e., for a given example we only update if $|y_t - \hat{y}_t| \geq \epsilon$. This feature is particularly important when applying the algorithm in conjunction with a kernel transformation, since on any given example when there is a nonvacuous update (see Equation (12) in Figure 2) the representation of the hypothesis grows, and this increases the computation time for future predictions.

2.4 Methods for local loss bounds

In the traditional methodology of total loss bounds the performance over the whole sequence is bounded; but nothing is known about the performance over any particular contiguous subsequence of trials except in a very weak average sense. However, for many online learning applications what is needed is a *local* guarantee, i.e., a statement of this form: given an unbounded sequence of trials the loss over trials s to s' is bounded by X. Local bounds are thus appropriate when the best predictor for the example sequence is changing over time. There have been a number of papers [28, 20, 3, 43, 6, 21] which prove loss bounds in terms of a measure of the amount of change of the best predictor over time. These bounds have been called *shifting* or *switching* bounds. The local bounds of this

section are direct simplifications of the shifting bounds in [21]. Here we give local bounds rather than shifting bounds, however, since less introductory machinery is required, the bounds are easier to interpret, and weaker assumptions on the example sequence are possible in the theorem statements.

Examining the Theorems 2 and 3 it is clear that statements of the form

$$\sum_{t=s}^{s'} L(y_t, \hat{y}_t) \leq \|\boldsymbol{w}_s - \boldsymbol{u}\|^2 R^2, \tag{18}$$

for all $\boldsymbol{u} \in \bigcap_{t=s}^{s'} \mathcal{U}_t$ where $R = \max_{t=s,\ldots,s'} \|\mathbf{x}_t\|$ are provable. However, the weakness of bounds of the above form is that $\boldsymbol{w}_s$ is wholly unknown without reference to the example sequence prior to trial s. We resolve this by introducing an additional update step (first introduced in [21]) into the Prototypical projection algorithm which constrains the hypotheses vectors $\boldsymbol{w}_1, \ldots$ to an origin centered hypersphere Γ_γ with radius γ (see Figure 4) by projection. The projection cor-

Input: A constraint parameter $\gamma > 0$, a sequence of closed convex sets $\{\mathcal{U}_1, \ldots\} \subset \mathcal{H}^\infty$ and a point $\boldsymbol{w}_1 \in \Gamma_\gamma$ where $\Gamma_\gamma = \{\boldsymbol{v} : \|\boldsymbol{v}\| \leq \gamma\} \subset \mathcal{H}$ and $\mathcal{H}$ is a Hilbert space.
Update 1: $\boldsymbol{w}'_t = \mathcal{P}_{\mathcal{U}_t}(\boldsymbol{w}_t)$
Update 2: $\boldsymbol{w}_{t+1} = \mathcal{P}_{\Gamma_\gamma}(\boldsymbol{w}'_t)$

Fig. 4. Constrained prototypical projection algorithm

responding to the new update may be computed as follows:

$$\mathcal{P}_{\Gamma_\gamma}(\boldsymbol{w}) = \begin{cases} \boldsymbol{w} & \boldsymbol{w} \in \Gamma_\gamma \\ \gamma \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} & \boldsymbol{w} \notin \Gamma_\gamma. \end{cases} \tag{19}$$

We can now prove the analogue of Lemma 1.

Lemma 4. *Given a constraint parameter $\gamma > 0$, a sequence of convex sets $\{\mathcal{U}_1, \ldots\}$ and a start vector $\boldsymbol{w}_1 \in \Gamma_\gamma$ where $\Gamma_\gamma = \{\boldsymbol{v} : \|\boldsymbol{v}\| \leq \gamma\}$ as input to the constrained prototypical projection algorithm (see Figure 4); then for any positive integers s and s' the inequality*

$$\sum_{t=s}^{s'} \|\boldsymbol{w}_t - \boldsymbol{w}'_t\|^2 \leq (\gamma + \|\boldsymbol{u}\|)^2 \tag{20}$$

holds for all $\boldsymbol{u} \in \bigcap_{t=s}^{s'} \mathcal{U}_t$ such that $\|\boldsymbol{u}\| \leq \gamma$.

Proof. On any trial t the Pythagorean Theorem 1 implies the following two inequalities:

$$\|\boldsymbol{w}'_t - \boldsymbol{w}_t\|^2 \leq \|\boldsymbol{u} - \boldsymbol{w}_t\|^2 - \|\boldsymbol{u} - \boldsymbol{w}'_t\|^2$$

for all $\boldsymbol{u}$ such that $\boldsymbol{u} \in \mathcal{U}_t$, and

$$0 \leq \|\boldsymbol{u} - \boldsymbol{w}'_t\|^2 - \|\boldsymbol{u} - \boldsymbol{w}_{t+1}\|^2$$

for all $\boldsymbol{u}$ such that $\boldsymbol{u} \in \Gamma_\gamma$. Combining the above two inequalities gives

$$\|\boldsymbol{w}'_t - \boldsymbol{w}_t\|^2 \leq \|\boldsymbol{u} - \boldsymbol{w}_t\|^2 - \|\boldsymbol{u} - \boldsymbol{w}_{t+1}\|^2$$

for all $\boldsymbol{u}$ such that $\boldsymbol{u} \in \mathcal{U}_t \bigcap \Gamma_\gamma$. Summing the previous inequality over all trials $t = s, \ldots, s'$ we have

$$\sum_{t=s}^{s'} \|\boldsymbol{w}'_t - \boldsymbol{w}_t\|^2 \leq \|\boldsymbol{u} - \boldsymbol{w}_s\|^2 - \|\boldsymbol{u} - \boldsymbol{w}_{s'+1}\|^2$$

for all $\boldsymbol{u} \in [\bigcap_{t=s}^{s'} \mathcal{U}_t] \bigcap \Gamma_\gamma$. Maximizing the first term and dropping the second term of the right hand side of the above inequality proves the lemma.

We designate the modification of projection algorithms in Figure 2 with the additional constraint update (see Equation (19)) as *constrained* projection algorithms for classification and regression. The following two theorems give local loss bounds for these algorithms by combining the Lemma above with Lemmas 2 and 3.

Theorem 4. *Given a sequence of examples* $\{(\boldsymbol{x}_1, y_1), \ldots\} \in (\mathcal{H}, \{-1, 1\})^\infty$*, a constraint parameter* $\gamma > 0$*, a start vector* $\boldsymbol{w}_1 \in \Gamma_\gamma \subset \mathcal{H}$*, and two positive integers* s *and* s'*, where* $\Gamma_\gamma = \{\boldsymbol{v} : \|\boldsymbol{v}\| \leq \gamma\}$ *and* $R = \max_{t=s,\ldots,s'} \|\boldsymbol{x}_t\|$ *then the cumulative mistakes of the constrained projection algorithm for classification between trials* s *and* s' *is bounded by*

$$\sum_{t=s}^{s'} L_m(y_t, \hat{y}_t) \leq R^2(\gamma + \|\boldsymbol{u}\|)^2 \tag{21}$$

for all $\boldsymbol{u}$ *such that* $\langle \boldsymbol{u}, \boldsymbol{x}_t \rangle y_t \geq 1$ *for* $t = s, \ldots, s'$ *and* $\|\boldsymbol{u}\| \leq \gamma$*.*

Theorem 5. *Given a sequence of examples* $\{(\boldsymbol{x}_1, y_1), \ldots\} \in (\mathcal{H}, \Re)^\infty$*, a constraint parameter* $\gamma > 0$*, a start vector* $\boldsymbol{w}_1 \in \Gamma_\gamma \subset \mathcal{H}$*, two positive integers* s *and* s'*, where* $\Gamma_\gamma = \{\boldsymbol{v} : \|\boldsymbol{v}\| \leq \gamma\}$ *and* $R = \max_{t=s,\ldots,s'} \|\boldsymbol{x}_t\|$ *then the cumulative* ϵ*-insensitive square loss of the constrained projection algorithm for regression between trials* s *and* s' *is bounded by*

$$\sum_{t=s}^{s'} L_{sq,\epsilon}(y_t, \hat{y}_t) \leq R^2(\gamma + \|\boldsymbol{u}\|)^2 \tag{22}$$

for all $\boldsymbol{u}$ *such that* $\langle \boldsymbol{u}, \boldsymbol{x}_t \rangle \in [y_t - \epsilon, y_t + \epsilon]$ *for* $t = s, \ldots, s'$ *and* $\|\boldsymbol{u}\| \leq \gamma$*.*

2.5 Incorporating kernels

Reproducing kernel preliminaries We assume that the reader is already familiar with kernel-based methods (for an overview see [10]). This section is for notation and a cursory review of kernel concepts. For our purposes, given

an abstract set $\mathcal{E}$ a kernel is a function $K : \mathcal{E} \times \mathcal{E} \to \Re$ where, for every finite set $\{x_1, \ldots, x_n\} \subset \mathcal{E}^n$ and every set of scalars $\{\alpha_1, \ldots, \alpha_n\} \subset \Re^n$ the following holds:

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j K(x_i, x_j) \geq 0.$$

Such a kernel is known in the literature as a reproducing kernel [2], a positive definite kernel [29] and as a positive hermitian matrix [30]. An immediate consequence of the above property is that the kernel is symmetric, i.e., $K(x, y) = K(y, x)$. The associated Hilbert space $\mathcal{H}_K$ is the completion of the span of the set $\{K(x, \cdot) : x \in \mathcal{E}\}$ where the associated inner product between elements with finite representations $f = \sum_{i=1}^{n} \alpha_i K(x_i, \cdot), f' = \sum_{i=1}^{n'} \alpha'_i K(x'_i, \cdot)$ is given by

$$\langle f, f' \rangle = \sum_{i=1}^{n}\sum_{j=1}^{n'}\alpha_i\alpha'_j K(x_i, x'_j). \tag{23}$$

When the representations are not finite, the appropriate limits are taken. The key property of $\mathcal{H}_K$ which we will use repeatedly is the reproducing property, which states that, given any $f \in \mathcal{H}_K$ and any $x \in \mathcal{E}$ then

$$\langle f(\cdot), K(x, \cdot) \rangle = f(x). \tag{24}$$

The kernel may be viewed as a function that computes an inner product in a *feature space* [10]. None of the results in this paper depend explicitly on the existence of a feature space representation, thus it is not introduced.

The kernel transformation algorithmic details and bounds Given a data set $\{(x_1, y_1), \ldots, (x_\ell, y_\ell)\} \in (\mathcal{E}, \Re)^\ell$, a reproducing kernel $K : \mathcal{E} \times \mathcal{E} \to \Re$, and an algorithm A which accepts as input an example sequence $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_\ell, y_\ell)\} \in (\mathcal{H}, \Re)^\ell$, the new algorithm A_K simply executes algorithm A on the data set $\{(K(x_1, \cdot), y_1), \ldots, (K(x_\ell, \cdot), y_\ell)\}$. The kernel algorithms in Figure 2 follow directly by syntactic substitution of $K(x, \cdot)$ for $\boldsymbol{x}$ and application of the reproducing Property (24). The transformation of Theorems 2, 3, 4 and 5 follow from similar substitutions. We give as an example the transformation of Theorem 3 below, but we omit the other transforms since they follow the same schema.

Theorem 6. *Given a sequence of examples* $\{(x_1, y_1), \ldots, (x_\ell, y_\ell)\} \in (\mathcal{E}, \Re)^\ell$ *and a start vector* $\boldsymbol{w}_1 \in \mathcal{H}_K$*, let* $R = \max_{t=1,\ldots,\ell} K(x_t, x_t)$*, then the cumulative* ϵ*-insensitive square loss of the kernel projection algorithm for regression is bounded by*

$$\sum_{t=1}^{\ell} L_{sq,\epsilon}(y_t, \hat{y}_t) \leq R^2 \|\boldsymbol{u} - \boldsymbol{w}_1\|^2 \tag{25}$$

for all functions $\boldsymbol{u} \in \mathcal{H}_K$ *such that* $\boldsymbol{u}(x_t) \in [y_t - \epsilon, y_t + \epsilon]$ *for* $t = 1, \ldots, \ell$*.*

Recently strong total loss bounds have been proven for ridge regression [12, 44, 4], in [36] a method to perform kernel ridge regression is given. Unfortunately the loss bounds for ridge regression with kernels do not transform since the proofs rely on properties of $\Re^n$. A transformation of those bounds is an interesting open problem.

Computational issues When implementing the projection algorithms for regression and classification, significant computational shortcuts can be taken when the patterns of the example sequence are from $\Re^n$. This is because when summing two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ from $\Re^n$, the resultant $\boldsymbol{x}+\boldsymbol{y}$ has the same sized representation as $\boldsymbol{x}$ or $\boldsymbol{y}$ under a simplified model of computation, i.e., $\text{size}(\boldsymbol{x}+\boldsymbol{y}) = \text{size}(\boldsymbol{x}) = \text{size}(\boldsymbol{y})$. Whereas when the elements are drawn from an arbitrary Hilbert space, as with kernel-based algorithms, $\text{size}(\boldsymbol{x}+\boldsymbol{y}) = \text{size}(\boldsymbol{x}) + \text{size}(\boldsymbol{y})$. Thus for data from $\Re^n$ the projections algorithms take $O(n)$ time per trial. Whereas the kernel-based projection algorithms require for typical kernels and typical implementations $O(m)$ kernel computations on trial t (in order to predict), if there have been $m \leq t$ nonvacuous updates. In Part 1, there are presented particular kernels for which we require only $O(\log m)$ computation time on trial t after m nonvacuous updates.

The implementation of the constraint update, $\mathcal{P}_{\Gamma_\gamma}(\boldsymbol{w})$ (see Equation 19) requires $O(n)$ time when $\boldsymbol{w} \in \Re^n$. The naive implementation of the constraint update for the kernel-based algorithms requires t^2 kernel computations after t nonvacuous updates since the function $\boldsymbol{w}$ has a representation of length t, i.e., $\boldsymbol{w}_{t+1} = \sum_{i=1}^{t} \alpha_i K(x_i, \cdot)$ since the inner product (see Equation (23)) is

$$\|\boldsymbol{w}_{t+1}\|^2 = \langle \boldsymbol{w}_{t+1}, \boldsymbol{w}_{t+1} \rangle = \sum_{i=1}^{t} \sum_{j=1}^{t} \alpha_i \alpha_j K(x_i, x_j).$$

However, we may use a simple recurrence to track the value of $\|\boldsymbol{w}_{t+1}\|$, since after an update we have only one new α value, i.e.,

$$\begin{aligned}\|\boldsymbol{w}_{t+1}\|^2 &= \|\boldsymbol{w}_t\|^2 + \sum_{i=1}^{t-1} \alpha_i \alpha_t K(x_i, x_t) + \sum_{j=1}^{t-1} \alpha_t \alpha_j K(x_t, x_j) + \alpha_t^2 K(x_t, x_t) \\ &= \|\boldsymbol{w}_t\|^2 + 2\alpha_t \overline{y}_t + \alpha_t^2 K(x_t, x_t).\end{aligned}$$

Since in order to predict we already compute $\overline{y}_t$, we may keep track of $\|\boldsymbol{w}_{t+1}\|$ at no additional asymptotic cost. Implementing the constraint update then only requires the additional step of shrinking $\boldsymbol{w}_t$ by $\rho_t \in (0, 1]$ (see Equation 19). Rather than explicitly multiplying each term of $\boldsymbol{w}_t$ by ρ_t, we maintain the scale constant $\rho^{(t)} = \prod_{i=1}^{t} \rho_i$ ($\rho^0 = 1$), all arithmetic is then done with the scale constant implicitly. Thus it can be seen that the projection update leads to a version of weight decay, since at the start of trial t we have

$$\boldsymbol{w}_t = \sum_{i=1}^{t-1} \prod_{j=i}^{t-1} \rho_j \alpha_i K(x_i, \cdot),$$

internally, however, we maintain the representation

$$\boldsymbol{w}_t = \rho^{(t-1)} \sum_{i=1}^{t-1} \frac{1}{\rho^{(i-1)}} \alpha_i K(x_i, \cdot)$$

so that the constraint update may be implemented in $O(1)$ time rather than $O(t)$ time.

Acknowledgments: The author would like to thank Nello Cristianini for useful discussions and Mary Dubberly for the proofreading of an early draft. A portion of this research was undertaken while at the Computer Learning Research Centre at Royal Holloway University.

References

1. M. A. Aizerman, E. M. Braverman, and L. I. Rozonoér. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.
2. N. Aronszajn. Theory of reproducing kernels. *Trans. Amer. Math. Soc.*, 68:337–404, 1950.
3. P. Auer and M. K. Warmuth. Tracking the best disjunction. *Journal of Machine Learning*, 32(2):127–150, August 1998. Special issue on concept drift.
4. Katy S. Azoury and M. K. Warmuth. Relative loss bounds for on-line density estirnation with the exponential family of distributions. In Kathryn B. Laskey and Henri Prade, editors, *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 31–40, S.F., Cal., July 30–August 1 1999. Morgan Kaufmann Publishers.
5. Heinz H. Bauschke and Jonathan M. Borwein. On projection algorithms for solving convex feasibility problems. *SIAM Review*, 38(3):367–426, September 1996.
6. A. Blum and C. Burch. On-line learning and the metrical task system problem. *Machine Learning*, 39(1):35–58, 2000.
7. B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proc. 5th Annu. Workshop on Comput. Learning Theory*, pages 144–152. ACM Press, New York, NY, 1992.
8. N. Cesa-Bianchi, P. Long, and M.K. Warmuth. Worst-case quadratic loss bounds for on-line prediction of linear functions by gradient descent. *IEEE Transactions on Neural Networks*, 7(2):604–619, May 1996.
9. T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 1990.
10. N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, UK, 2000.
11. V. Faber and J. Mycielski. Applications of learning theorems. *Fundamenta Informaticae*, 15(2):145–167, 1991.
12. D. P. Foster. Prediction in the worst case. *The Annals of Statistics*, 19(2):1084–1090, 1991.
13. Yoav Freund and Robert E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, 1999.
14. Thilo-Thomas Frieß, Nello Cristianini, and Colin Campbell. The Kernel-Adatron algorithm: a fast and simple learning procedure for Support Vector machines. In *Proc. 15th International Conf. on Machine Learning*, pages 188–196. Morgan Kaufmann, San Francisco, CA, 1998.

15. C. Gentile. A new approximate maximal margin classification algorithm. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13, 2001.
16. Federico Girosi, Michael Jones, and Tomaso Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7(2):219–269, 1995.
17. T. Hastie and R. Tibshirani. Generalized additive models, 1990.
18. D. Haussler, J. Kivinen, and M. K. Warmuth. Sequential prediction of individual sequences under general loss functions. IEEE Transactions on Information Theory, 44(2):1906–1925, September 1998.
19. D. P. Helmbold, J. Kivinen, and M. K. Warmuth. Relative loss bounds for single neurons. *Journal of Machine Learning*, 2001. To appear.
20. Mark Herbster and Manfred Warmuth. Tracking the best expert. In *Proc. 12th International Conference on Machine Learning*, pages 286–294. Morgan Kaufmann, 1995.
21. Mark Herbster and Manfred K. Warmuth. Tracking the best regressor. In *Proc. 11th Annu. Conf. on Comput. Learning Theory*, pages 24–31. ACM Press, New York, NY, 1998.
22. G. S. Kimeldorf and G. Wahba. Some results on tchebycheffian spline functions. *J. Math. Anal. Applications*, 33(1):82–95, 1971.
23. J. Kivinen and M. K. Warmuth. Additive versus exponentiated gradient updates for linear prediction. *Information and Computation*, 132(1):1–64, January 1997.
24. Y. LeCun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger, P. Simard, and V. Vapnik. Comparison of learning algorithms for handwritten digit recognition, 1995.
25. Y. Li. and P. Long. The relaxed online maximum margin algorithm. *Machine Learning*, 2001.
26. N. Littlestone. Learning when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.
27. N. Littlestone. *Mistake Bounds and Logarithmic Linear-threshold Learning Algorithms*. PhD thesis, Technical Report UCSC-CRL-89-11, University of California Santa Cruz, 1989.
28. N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.
29. J. Mercer. Functions of a positive and negative type and their connection with the threory of integral equations. *Philosophical Transactions Royal Society London Ser. A.*, 209, 1909.
30. E. H. Moore. *General Analysis. Part I.* American Philosophical Society, Philadelphia, 1935.
31. J. Von Neumann. *Functional Operators, Vol II. The Geometry of orthogonal spaces*, volume 22. Princeton University Press, 1950.
32. A. Novikoff. On convergence proofs for perceptrons. In *Proc. Sympos. Math. Theory of Automata (New York, 1962)*, pages 615–622. Polytechnic Press of Polytechnic Inst. of Brooklyn, Brooklyn, N.Y., 1963.
33. J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 185–208, Cambridge, MA, 1999. MIT Press.
34. F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psych. Rev.*, 65:386–407, 1958. (Reprinted in *Neurocomputing* (MIT Press, 1988).).

35. Walter Rudin. *Real and Complex Analysis.* McGraw-Hill, New York, 3 edition, 1986.
36. G. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. In *Proc. 15th International Conf. on Machine Learning*, pages 515–521. Morgan Kaufmann, San Francisco, CA, 1998.
37. John Shawe-Taylor and Nello Cristianini. Further results on the margin distribution. In *Proc. 12th Annu. Conf. on Comput. Learning Theory*, pages 278–285. ACM Press, New York, NY, 1999.
38. A. Smola. Large scale and online learning with kernels. Talk given Dec 5, 2000 at Royal Holloway University, based on joint work with J. Kivinen, P. Wankadia, and R. Williamson.
39. V. Vapnik. *Statistical Learning Theory.* John_Wiley, 1998.
40. V. Vapnik, S. Golowich, and A. Smola. Support vector method for function approximation, regression estimation, and signal processing. In M. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 281–287, Cambridge, MA, 1997. MIT Press.
41. V. N. Vapnik and A. Y. Chervonenkis. *Teoriya raspoznavaniya obrazov. Statisticheskie problemy obucheniya. [Theory of Pattern Recognition].* Izdat. "Nauka", Moscow, 1974.
42. V. Vovk. Aggregating strategies. In *Proc. 3rd Annu. Workshop on Comput. Learning Theory*, pages 371–383. Morgan Kaufmann, 1990.
43. V. Vovk. Derandomizing stochastic prediction strategies. In *Proc. 10th Annu. Workshop on Comput. Learning Theory.* ACM Press, New York, NY, 1997.
44. Volodya Vovk. Competitive on-line linear regression. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10. The MIT Press, 1998.