THE *P*-NORM GENERALIZATION OF THE LMS ALGORITHM FOR ADAPTIVE FILTERING

Jyrki Kivinen [*,1] **Manfred K. Warmuth** [**,2] **Babak Hassibi** [***]

[*] *Research School of Information Sciences and Engineering, Australian National University, Canberra, ACT 0200, Australia*
[**] *Computer Science Department, 237 Baskin Engineering, University of California, Santa Cruz, CA 95064, USA*
[***] *Department of Electrical Engineering, California Institute of Technology, Pasadena, CA 91125, USA*

Abstract: Recently much work has been done analyzing online machine learning algorithms in a worst case setting, where no probabilistic assumptions are made about the data. This is analogous to the H^∞ setting used in adaptive linear filtering. Bregman divergences have become a standard tool for analyzing online Machine Learning algorithms. Using these divergences, we motivate a generalization of the the Least Mean Squared (LMS) algorithm. The loss bounds for these so-called p-norm algorithms involve other norms than the standard 2-norm. The bounds can be significantly better if a large proportion of the input variables are irrelevant, *i.e.*, if the weight vector we are trying to learn is sparse. We also prove result for nonstationary targets. We only know how to apply kernel methods to the standard LMS algorithm (*i.e.*, $p = 2$). However even in the general p-norm case we can handle generalized linear models where the output of the system is a linear function combined with a nonlinear transfer function (*e.g.*, the logistic sigmoid).

Keywords: adaptive filtering, online learning, H^∞ optimality, Least Mean Squares

1. INTRODUCTION

We focus on the following linear model of adaptive filtering:

$$y_t = \mathbf{u} \cdot \mathbf{x}_t + v_t. \tag{1}$$

Here $\mathbf{u}$ is the unknown target, $\mathbf{x}_t$ is a known input, v_t is unknown noise and y_t is the known output signal. We are interested in algorithms that maintain a weight vector $\mathbf{w}_t$ based on the past examples $(\mathbf{x}_j, y_j)$, $j = 1, \ldots, t$, and, over a sequence of T trials, get as close as possible to the target $\mathbf{u}$. As we shall see, closely related online problems have also been studied in machine learning.

[1] Supported by the Australian Research Council
[2] Supported by grant NSF CCR 9821087

More specifically, at trial t the algorithm receives $\mathbf{x}_t$ and y_t (in order) and has to commit to a weight vector at some point after seeing $\mathbf{x}_t$. We consider three problems depending on whether the algorithm needs to commit to its weight vector before or after seeing y_t and depending on how the loss of the algorithm is measured.

A priori filtering: Here we are interested in predicting the *uncorrupted* output $\mathbf{u} \cdot \mathbf{x}_t$ before the signal y_t is received. Therefore the algorithm needs to commit to its weight vector $\mathbf{w}_{t-1}$ right before seeing y_t and our loss is the energy of the a priori filtering error $\mathbf{u} \cdot \mathbf{x}_t - \mathbf{w}_{t-1} \cdot \mathbf{x}_t$, *i.e.*,

$$\sum_{t=1}^{T} (\mathbf{u} \cdot \mathbf{x}_t - \mathbf{w}_{t-1} \cdot \mathbf{x}_t)^2. \tag{2}$$

A posteriori filtering: Here we assume that for estimating the uncorrupted output $\mathbf{u} \cdot \mathbf{x}_t$, we also have

access to the measurement y_t. Thus, the algorithm needs to commit to its weight vector $\mathbf{w}_t$ only after seeing y_t and the loss is the square of the a posteriori error:

$$\sum_{t=1}^{T}(\mathbf{u}\cdot\mathbf{x}_t-\mathbf{w}_t\cdot\mathbf{x}_t)^2. \tag{3}$$

Prediction: Here we are interested in *predicting* the next observation y_t before receiving it. Thus the algorithm needs to commit to its weight vector $\mathbf{w}_{t-1}$ before seeing y_t. The prediction error is $y_t-\mathbf{w}_{t-1}\cdot\mathbf{x}_t$ and the loss

$$\sum_{t=1}^{T}(y_t-\mathbf{w}_{t-1}\cdot\mathbf{x}_t)^2. \tag{4}$$

The above prediction problem is also studied in machine learning. Note that in the filtering problems, the term v_t is regarded as a disturbance, so we are interested in estimating the *true* output $\mathbf{u}\cdot\mathbf{x}_t$ of the linear system for the input $\mathbf{x}_t$. In the prediction problem, however, we consider the y_t as the actual outcome of some event we are interested in predicting; in that case there is no particular value in matching the prediction $\mathbf{u}\cdot\mathbf{x}_t$ at those times where it is inaccurate.

In contrast to the loss function used by the prediction problem, the loss functions for the two filtering problems are not measurable by the online algorithm. This is because the true value of $\mathbf{u}$ is unknown. To partially alleviate this, one possibility is to consider the *worst-case* losses by maximizing (2) and (3) over $\mathbf{u}$. However, this is not really meaningful since the the losses can be made arbitrarily large by scaling $\mathbf{u}$. A much more reasonable route is to look at the *normalized* loss, normalized by the energy of the unknown sequence v_t and the unknown vector $\mathbf{u}$. Thus, we can consider

$$\frac{\sum_{t=1}^{T}(\mathbf{u}\cdot\mathbf{x}_t-\mathbf{w}_{t-1}\cdot\mathbf{x}_t)^2}{\sum_{t=1}^{T}(\mathbf{u}\cdot\mathbf{x}_t-y_t)^2+\frac{1}{\eta}||\mathbf{u}||_2^2},$$

where we have used $v_t=y_t-\mathbf{u}\cdot\mathbf{x}_t$ and where $\eta>0$ is a normalization constant that weighs the relative contributions of $\sum_{t=1}^{T}(\mathbf{u}\cdot\mathbf{x}_t-y_t)^2$ and $||\mathbf{u}||_2^2$. In particular, we can look at the *worst-case* of this normalized loss by maximizing over $\mathbf{u}$:

$$\max_{\mathbf{u}}\frac{\sum_{t=1}^{T}(\mathbf{u}\cdot\mathbf{x}_t-\mathbf{w}_{t-1}\cdot\mathbf{x}_t)^2}{\sum_{t=1}^{T}(\mathbf{u}\cdot\mathbf{x}_t-y_t)^2+\frac{1}{\eta}||\mathbf{u}||_2^2}.$$

This choice of $\mathbf{u}$ is now well-behaved and can be determined when all examples are processed. In control-theoretic jargon, the above maximum energy gain is referred to as the H^∞ norm.

Our starting point is the Least Mean Squares (LMS) algorithm (also known as the Widrow-Hoff algorithm), defined by the update rule

$$\mathbf{w}_t=\mathbf{w}_{t-1}-\eta\,(\mathbf{w}_{t-1}\cdot\mathbf{x}_t-y_t)\mathbf{x}_t$$

where $\eta>0$ is a learning rate parameter. The basic result for a priori filtering (Hassibi *et al.*, 1996) states that for the LMS algorithm,

$$\frac{\sum_{t=1}^{T}(\mathbf{u}\cdot\mathbf{x}_t-\mathbf{w}_{t-1}\cdot\mathbf{x}_t)^2}{\sum_{t=1}^{T}(\mathbf{u}\cdot\mathbf{x}_t-y_t)^2+\frac{1}{\eta}||\mathbf{u}||_2^2}\le 1 \tag{5}$$

for all $\mathbf{u}$, provided $\eta\le 1/\max||\mathbf{x}_t||_2^2$. Further, no algorithm can in general guarantee a ratio less than one. Therefore we say that LMS is H^∞ *optimal.* (For the above, and as done throughout the paper, we assumed $\mathbf{w}_0=\mathbf{0}$; if $\mathbf{w}_0\neq\mathbf{0}$, then $||\mathbf{u}||_2^2$ must be replaced by $||\mathbf{u}-\mathbf{w}_0||_2$.)

To compare this with results from machine learning, assume there is a known upper bound X_2 such that $||\mathbf{x}_t||_2\le X_2$ for all t, and write $\eta=\alpha/X_2^2$. Then Cesa-Bianchi *et al.* (1996) have shown that for $0<\alpha<1$,

$$\sum_{t=1}^{T}(y_t-\mathbf{w}_{t-1}\cdot\mathbf{x}_t)^2 \le\frac{1}{1-\alpha}\sum_{t=1}^{T}(\mathbf{u}\cdot\mathbf{x}_t-y_t)^2+\frac{1}{\alpha}X_2^2||\mathbf{u}||_2^2. \tag{6}$$

To compare the prediction and filtering results, we write (5) as

$$\sum_{t=1}^{T}(\mathbf{u}\cdot\mathbf{x}_t-\mathbf{w}_{t-1}\cdot\mathbf{x}_t)^2 \le\sum_{t=1}^{T}(\mathbf{u}\cdot\mathbf{x}_t-y_t)^2+\frac{1}{\alpha}X_2^2||\mathbf{u}||_2^2 \tag{7}$$

where X_2 and η are as above and $0<\alpha\le 1$. The factor $1/(1-\alpha)$ in (6) is a source of many difficulties in machine learning, where the goal is to tune the learning rate so as to obtain the smallest possible bound. However, the filtering bound (7) is optimized at $\alpha=1$. Thus we omit the α parameter from the filtering bounds when the norm of instances is bounded.

We are interested in taking machine learning techniques that have recently been used to generalize (6) and applying them in the filtering setting, leading to generalizations of (7) and new interpretations of the filtering algorithms. Techniques we are interested in include

(1) motivating algorithms in terms of minimization problems based on Bregman divergences (Kivinen and Warmuth, 1997; Kivinen and Warmuth, 2001),
(2) replacing the 2-norms in the bounds by other norms (Kivinen and Warmuth, 1997; Grove *et al.*, 2001; Gentile and Littlestone, 1999), and
(3) allowing for nonstationary targets (Herbster and Warmuth, 2001) and nonlinear predictors (Helmbold *et al.*, 1999).

In Section 2 we introduce Bregman divergences and show how a Bregman divergence can be used to derive

two subtly different algorithms, the *implicit* and *explicit* algorithm. When the squared Euclidean distance is used as the Bregman divergence, these algorithms become the standard *LMS* and *normalized LMS* algorithm (Hassibi *et al.*, 1996), respectively. In Section 3 we give filtering loss bounds for the explicit and implicit algorithms in the case of Bregman divergences based on squared q-norms (Grove *et al.*, 2001). These bounds generalize the results of Hassibi *et al.* (1996) about the H^∞ optimality of LMS and normalized LMS for the a priori and a posteriori filtering problems. The generalization replaces the product $||\mathbf{x}||_2||\mathbf{u}||_2$ in the bound by another product of dual norms $||\mathbf{x}||_p||\mathbf{u}||_q$ where p and q are such that $1/p+1/q=1$ and $2\le p<\infty$. The new bounds are significantly stronger when the target $\mathbf{u}$ is sparse, *i.e.*, has few nonzero components. In Section 4 we generalize the q-norm based algorithms to allow for nonstationary targets $\mathbf{u}_t$. The loss bounds in the nonstationary case include an extra term that depends on the total distance $\mathbf{u}_t$ travels during the whole sequence, as measured by the q-norm. Again there are no distribution assumptions about this movement. Finally, in Section 5 we give bounds for generalized linear regression where the linear predictor is fed through a nonlinear transfer function (such as the logistic sigmoid).

2. DERIVATION OF ALGORITHMS

In this section we give the basic definitions of Bregman divergences and explain their use in deriving online algorithms. See Azoury and Warmuth (2001) and references therein for more background on these divergences.

Assume that $F:\mathbf{R}^n\to\mathbf{R}^n$ is a strictly convex twice differentiable function. Denote its gradient by $\mathbf{f}=\nabla F$; notice that $\mathbf{f}$ is one-to-one. The *Bregman divergence* $d_F(\mathbf{u},\mathbf{w})$ (Bregman, 1967) is defined for $\mathbf{u},\mathbf{w}\in\mathbf{R}^n$ as the error in approximating $F(\mathbf{u})$ by its first order Taylor polynomial around $\mathbf{w}$. More formally,

$$d_F(\mathbf{u},\mathbf{w})=F(\mathbf{u})-F(\mathbf{w})-\mathbf{f}(\mathbf{w})\cdot(\mathbf{u}-\mathbf{w}).$$

The Bregman divergence $d_F(\mathbf{u},\mathbf{w})$ is always nonnegative, and zero only for $\mathbf{u}=\mathbf{w}$. It is (strictly) convex in $\mathbf{u}$, but might not be convex in $\mathbf{w}$. Usually, d_F is not symmetric.

Example 1. Given $q>1$, define $F(\mathbf{w})=\frac{1}{2}||\mathbf{w}||_q^2$ where $||\mathbf{u}||_q=(\sum_i|u_i|^q)^{1/q}$. We denote the corresponding Bregman divergence by d_q. Thus

$$d_q(\mathbf{u},\mathbf{w})=\frac{1}{2}||\mathbf{u}||_q^2-\frac{1}{2}||\mathbf{w}||_q^2-\mathbf{f}(\mathbf{w})\cdot(\mathbf{u}-\mathbf{w})$$

where the gradient is given by

$$f_i(\mathbf{w})=\frac{\mathrm{sign}(w_i)|w_i|^{q-1}}{||\mathbf{w}||_q^{q-2}}.$$

It is easy to verify that the inverse of the gradient is given by

$$f_i^{-1}(\theta)=\frac{\mathrm{sign}(\theta_i)|\theta_i|^{p-1}}{||\theta||_p^{p-2}}$$

where $1/p+1/q=1$. Grove *et al.* (2001) have shown that if $\mathbf{f}(\mathbf{w}')=\mathbf{f}(\mathbf{w})+\mathbf{x}$, we have

$$d_q(\mathbf{w},\mathbf{w}')\le\frac{p-1}{2}||\mathbf{x}||_p^2. \tag{8}$$

The important special case $p=q=2$ gives $d_2(\mathbf{u},\mathbf{w})=\frac{1}{2}||\mathbf{u}-\mathbf{w}||_2^2$, with $\mathbf{f}$ the identity function.

A second important family of Bregman divergences is the relative entropy and its variants.

Example 2. Define $F(\mathbf{w})=\sum_i(w_i\ln w_i-w_i)$. Then

$$d_F(\mathbf{u},\mathbf{w})=\sum_i(u_i\ln\frac{u_i}{w_i}-u_i+w_i)$$

is the unnormalized relative entropy. (When $\sum_i u_i=\sum_i w_i=1$ this gives the standard relative entropy.) The gradient is given by $f_i(\mathbf{w})=\ln w_i$, the inverse obviously being $f_i^{-1}(\theta)=\exp(\theta_i)$.

The following generalization of the Pythagorean Theorem follows directly from the definition of a Bregman divergence:

$$\begin{aligned}d_F(\mathbf{u},\mathbf{w}')=d_F(\mathbf{u},\mathbf{w})+d_F(\mathbf{w},\mathbf{w}')\\+(\mathbf{f}(\mathbf{w}')-\mathbf{f}(\mathbf{w}))\cdot(\mathbf{w}-\mathbf{u}).\end{aligned} \tag{9}$$

Since the dot product $(\mathbf{f}(\mathbf{w}')-\mathbf{f}(\mathbf{w}))\cdot(\mathbf{w}-\mathbf{u})$ can be positive, this shows in particular that d_F does not satisfy the triangle inequality. We recover the standard Pythagorean Theorem when the divergence is the squared Euclidean distance (*i.e.*, $\mathbf{f}$ is identity) and the dot product is zero (*i.e.*, $(\mathbf{w}'-\mathbf{w})$ and $\mathbf{w}-\mathbf{u}$ are orthogonal).

We now use a Bregman divergence d_F as a regularizer for deriving an update rule. This framework for motivating updates was introduced by Kivinen and Warmuth (1997) in the prediction setting. Bregman divergences based on the squared q-norm were first used by Grove *et al.* (2001) to analyze algorithms for learning linear threshold functions.

Suppose an example $(\mathbf{x}_t,y_t)$ has been observed, and we wish to update our hypothesis $\mathbf{w}_{t-1}$ based on this example. We wish to decrease the squared loss $(y_t-\mathbf{w}\cdot\mathbf{x}_t)^2$ (other convex loss functions can also be considered; see Section 5), but we should not make big changes based on just a single example. Thus, we define

$$C_t(\mathbf{w})=d_F(\mathbf{w},\mathbf{w}_{t-1})+\frac{1}{2}\eta(y_t-\mathbf{w}\cdot\mathbf{x}_t)^2$$

where $\eta > 0$ is a trade-off parameter, and tentatively set $\mathbf{w}_t = \arg\min_{\mathbf{w}} C_t(\mathbf{w})$. Since C_t is convex, we can minimize by setting $\nabla C_t(\mathbf{w}_t) = 0$. By substituting the definition of d_F, this becomes

$$\mathbf{f}(\mathbf{w}_t) = \mathbf{f}(\mathbf{w}_{t-1}) - \eta(\mathbf{w}_t \cdot \mathbf{x}_t - y_t)\mathbf{x}_t. \qquad (10)$$

Note that $\mathbf{w}_t$ appears on both sides of the equality. Hence, we call the algorithm defined by this equality the *implicit algorithm* for divergence d_F. Notice that (10) can be solved numerically by a line search since $\mathbf{w} = \mathbf{f}^{-1}(\mathbf{f}(\mathbf{w}_{t-1} + \alpha\mathbf{x}))$ for some scalar α, and the inverse $\mathbf{f}^{-1}$ is easy to compute in the cases we consider. Also in the special case of two-norm ($d_F = d_2$), with $\mathbf{f}$ the identity function, we can solve (10) in closed form, which turns out to give the the algorithm called *normalized LMS* by Hassibi *et al.* (1996).

Instead of solving (10) numerically, we often find it sufficient to notice that for reasonable values of η, the values $\mathbf{w}_t \cdot \mathbf{x}_t$ and $\mathbf{w}_{t-1} \cdot \mathbf{x}_t$ should be fairly close to each other. Thus, we may approximate the solution of (10) by

$$\mathbf{w}_t = \mathbf{f}^{-1}(\mathbf{f}(\mathbf{w}_{t-1}) - \eta(\mathbf{w}_{t-1} \cdot \mathbf{x}_t - y_t)\mathbf{x}_t). \qquad (11)$$

We call this the *explicit algorithm* for divergence d_F. The special case $d_F = d_2$ gives the usual LMS algorithm.

3. BOUNDS IN TERMS OF DIFFERENT NORMS

Our interest in considering the generalization of LMS to the p-norm based algorithms comes from the fact that for these algorithms the term $||\mathbf{x}||_2||\mathbf{u}||_2$ in the LMS bound is replaced by another product of dual norms $||\mathbf{x}||_p||\mathbf{u}||_q$ (*i.e.*, $1/p + 1/q = 1$). We discuss the implications of this after giving the main result.

Theorem 3. Fix p and q such that $1/p + 1/q = 1$ and $2 \le p < \infty$. Assume that $||\mathbf{x}_t||_p \le X_p$ for all t. Then the explicit algorithm for d_q with learning rate $\eta = 1/((p-1)X_p^2)$ satisfies

$$\sum_{t=1}^{T} (\mathbf{u} \cdot \mathbf{x}_t - \mathbf{w}_{t-1} \cdot \mathbf{x}_t)^2 \le \sum_{t=1}^{T} (\mathbf{u} \cdot \mathbf{x}_t - y_t)^2 + (p-1)X_p^2||\mathbf{u}||_q^2$$

for any $\mathbf{u} \in \mathbf{R}^n$.

To see how the choice of p affects the bound, consider two cases: $p = 2$ (*i.e.*, LMS) and $p = 2\ln n$ (*i.e.*, p rather large). For the latter case, Gentile and Littlestone (1999) give the bound

$$(p-1)||\mathbf{x}||_p^2||\mathbf{u}||_q^2 \le (2e\ln n)||\mathbf{x}||_\infty^2||\mathbf{u}||_1^2$$

(where $||\mathbf{x}||_\infty = \max_i |x_i|$). Thus, we compare $||\mathbf{x}||_2^2||\mathbf{u}||_2^2$ with $(2e\ln n)||\mathbf{x}||_\infty^2||\mathbf{u}||_1^2$. The basic intuition is that the large p bound is better when the target $\mathbf{u}$ is sparse (few nonzero components) and the instances $\mathbf{x}_t$ are dense (many components of roughly equal size). As an extreme case, suppose $\mathbf{u} = (1,0,\ldots,0)$ and $\mathbf{x} = (1,\ldots,1)$. Then $||\mathbf{u}||_2^2||\mathbf{x}||_2^2 = n^2$ and $(2e\ln n)||\mathbf{x}||_\infty^2||\mathbf{u}||_1^2 = 2e\ln n$. So in this case, the large p case has a drastically better bound. On the other hand, when $\mathbf{u} = (1,\ldots,1)$ and the instances are permutations of $(1,0,\ldots,0)$, then $||\mathbf{x}||_2^2||\mathbf{u}||_2^2 = n^2$ and $(2e\ln n)||\mathbf{x}||_\infty^2||\mathbf{u}||_1^2 = 2en^2\ln n$. So now the $p = 2$ case is moderately better. See Kivinen and Warmuth (1997) for a more detailed discussion. They derive bounds of the $(\ln n)||\mathbf{x}||_\infty^2||\mathbf{u}||_1^2$ type for the prediction problem using an algorithm called the *exponentiated gradient* (EG) algorithm. EG uses (ignoring a normalization) the update (11) with $f_i(\mathbf{w}) = \ln w_i$. The analysis of EG can also be lifted to the filtering setting, giving again bounds of the $(\ln n)||\mathbf{x}||_\infty^2||\mathbf{u}||_1^2$ type; we omit the details.

For the a posteriori case we have the following theorem which generalizes the result about normalized LMS by Hassibi *et al.* (1996).

Theorem 4. Fix p and q such that $1/p + 1/q = 1$ and $2 \le p < \infty$. Assume that $||\mathbf{x}_t||_p \le X_p$ for all t. Then the implicit algorithm for d_q with learning rate $\eta = 1/((p-1)X_p^2)$ satisfies

$$\sum_{t=1}^{T} (\mathbf{u} \cdot \mathbf{x}_t - \mathbf{w}_t \cdot \mathbf{x}_t)^2 \le \sum_{t=1}^{T} (\mathbf{u} \cdot \mathbf{x}_t - y_t)^2 + (p-1)X_p^2||\mathbf{u}||_q^2.$$

4. NONSTATIONARY TARGETS

Following Herbster and Warmuth (2001), we now consider a variant of the algorithm that keeps the q-norm of the weight vector bounded by U_q, where $U_q > 0$ is a parameter to the algorithm. We call this two-step update the *bounded explicit update* for d_F:

Explicit update step: Let

$$\mathbf{w}'_t = \mathbf{f}^{-1}(\mathbf{f}(\mathbf{w}_{t-1}) - \eta(\mathbf{w}_{t-1} \cdot \mathbf{x}_t - y_t)\mathbf{x}_t).$$

Out of bound update step: If $||\mathbf{w}'_t||_q > U_q$, then $\mathbf{w}_t = U_q\mathbf{w}'_t/||\mathbf{w}'_t||_q$; otherwise $\mathbf{w}_t = \mathbf{w}'_t$.

Thus if the algorithm tries to increase the q-norm of its weight vector above U_q, then we scale it back.

We now let the target $\mathbf{u}_t$ vary with time (nonstationary model):

$$y_t = \mathbf{u}_t \cdot \mathbf{x}_t + v_t \qquad (12)$$

As previously, our bound will include a penalty for the (maximum) norm of $\mathbf{u}_t$. Additionally, there is now also a penalty for the total distance the target moves during the process.

Theorem 5. Fix p and q such that $1/p + 1/q = 1$ and $2 \leq p < \infty$. Assume $||\mathbf{x}_t||_p \leq X_p$ and $||\mathbf{u}_t||_q \leq U_q$ for all t. Then the bounded explicit algorithm for d_q with learning rate $\eta = 1/((p-1)X_p^2)$ and parameter U_q satisfies

$$\sum_{t=1}^{T} (\mathbf{u}_t \cdot \mathbf{x}_t - \mathbf{w}_{t-1} \cdot \mathbf{x}_t)^2 \leq \sum_{t=1}^{T} (\mathbf{u}_t \cdot \mathbf{x}_t - y_t)^2 + (p-1)X_p^2 U_q^2 + 2(p-1)X_p^2 U_q \sum_{t=1}^{T-1} ||\mathbf{u}_{t+1} - \mathbf{u}_t||_q.$$

In the special case $\mathbf{u}_t = \mathbf{u}_{t+1}$ for all t, the result becomes Theorem 3 with the exception that the norm bound U_q must now be defined before the algorithm can be run.

A similar result can be obtained for a bounded version of the implicit algorithm in a posteriori filtering; we omit the details.

5. NONLINEAR MODELS

The kernel method is a well known technique for generalizing linear learning algorithm to nonlinear problems. The idea is that if the relationship between the inputs $\mathbf{x}_t \in \mathbf{R}^n$ and outputs y_t is nonlinear, it is often possible to transform the inputs by a *feature map* Ψ into a higher dimensional space $\mathbf{R}^N$, $N > n$, such that the relationship between the *feature vectors* $\Psi(\mathbf{x}_t)$ and outputs y_t becomes linear. We can then apply linear learning methods on the data $(\Psi(\mathbf{x}_t), y_t)$. In many interesting cases, there is a *kernel* $k: \mathbf{R}^n \times \mathbf{R}^n \to \mathbf{R}$ such that $\Psi(\mathbf{x}) \cdot \Psi(\mathbf{x}') = k(\mathbf{x}, \mathbf{x}')$. The kernel can then be used to compute dot products of feature vectors implicitly without evaluating Ψ.

For example, if we run the LMS algorithm on the examples $(\Psi(\mathbf{x}_t), y_t)$ then its weight vector (now a vector in $\mathbf{R}^N$) can be written as

$$\mathbf{w}_T = \sum_{t=1}^{T-1} \alpha_t \Psi(\mathbf{x}_t), \tag{13}$$

where $\alpha_t = \eta(y_t - \mathbf{w}_{t-1} \cdot \Psi(\mathbf{x}_t))$. Thus $\mathbf{w}_T \cdot \Psi(\mathbf{x}) = \sum_{t=1}^{T-1} \alpha_t \Psi(\mathbf{x}_t) \cdot \Psi(\mathbf{x}) = \sum_{t=1}^{T-1} \alpha_t k(\mathbf{x}_t, \mathbf{x})$. Hence, instead of explicitly maintaining a high dimensional weight vector $\mathbf{w}_T \in \mathbf{R}^N$, it is sufficient to keep track of the coefficients α_t and compute the dot products using the kernel.

The key property for the kernel method is that the prediction of the algorithm for instance $\mathbf{x}_T$ can be written (using a feature map Ψ) in terms of the dot products $\Psi(\mathbf{x}_T) \cdot \Psi(\mathbf{x}_t)$ for the other instances $\mathbf{x}_t$. This in particular implies that the algorithm must be rotation invariant, *i.e.*, its predictions must remain unchanged if we rotate the feature vectors by replacing $\Psi(\mathbf{x}_t)$ by $A\Psi(\mathbf{x}_t)$ where A is a fixed orthonormal matrix. For example, the LMS algorithm and Support Vector Machines are rotation invariant. Not surprisingly, these algorithms are motivated using 2-norms, the only rotation invariant p-norm.

On the other hand, applying the feature map with the more general updates (10) and (11) (assuming $\mathbf{f}(\mathbf{w}_0) = \mathbf{0}$ as usual) gives us $\mathbf{w}_T = \mathbf{f}^{-1}(\sum_{t=1}^{T-1} \alpha_t \Psi(\mathbf{x}_t))$ where $\mathbf{f}$ is typically nonlinear. Thus, it is at least not immediately clear how kernels could be applied. Furthermore, in classification it is known that rotation invariant algorithms cannot really take advantage of sparse weight vectors (Kivinen *et al.*, 1997). There are also computational hardness results suggesting there is a trade-off between having easy evaluation via kernels and having fast convergence for sparse targets via a nonlinear $\mathbf{f}$ (Khardon *et al.*, 2002). (See also (Kivinen and Warmuth, 1997) for a related discussion). As obtaining improved bounds for sparse targets is the main motivation for considering the case $p > 2$, looking for a general kernel version of the p-norm algorithms does not seem promising. However, Takimoto and Warmuth (2002) have shown that some specific kernels can be efficiently combined with an EG type algorithm. (EG corresponds roughly to taking $f_i(\mathbf{w}) = \ln w_i$ and gives bounds similar to the p-norm algorithm with $p = O(\ln n)$.)

Another slightly extended framework is that based on *generalized linear regression.* Here we replace the model (1) by

$$y_t = \phi(\mathbf{u} \cdot \mathbf{x}_t + v_t) \tag{14}$$

where ϕ is a continuous, strictly increasing *transfer function.* The logistic sigmoid $\phi(r) = 1/(1+\exp(-r))$ is a typical example of such a transfer function. In the prediction setting (where the learner tries to match y_t), the prediction becomes $\hat{y}_t = \phi(\mathbf{w}_{t-1} \cdot \mathbf{x}_t)$. In the filtering setting, we would naturally also include the transfer function in the prediction, giving $\hat{y}_t = \phi(\mathbf{w}_{t-1} \cdot \mathbf{x}_t)$ for the a priori and $\hat{y}_t = \phi(\mathbf{w}_t \cdot \mathbf{x}_t)$ for the a posteriori case. The algorithm then tries to match $\hat{y}_t$ to $\phi(\mathbf{u} \cdot \mathbf{x}_t)$. One could in principle still use the squared error $(\phi(\mathbf{u} \cdot \mathbf{x}_t) - \phi(\mathbf{w} \cdot \mathbf{x}_t))^2$ as the performance measure, but this is nonconvex in $\mathbf{u}$ and $\mathbf{w}$ and actually leads to a very badly behaved optimization problem (Auer *et al.*, 1995). We obtain a better behaved problem by using the *matching loss* for ϕ (Auer *et al.*, 1995), defined for y and y' in the range of ϕ as

$$L(y, y') = \int_{\phi^{-1}(y)}^{\phi^{-1}(y')} (\phi(r) - y) dr.$$

(Notice that by our assumptions ϕ is one-to-one.) This definition of loss may seem arbitrary, but it is actually a one-dimensional Bregman divergence: if we let $\Phi(r) = \int \phi(r) dr$, then

$$L(\phi(a),\phi(a')) = d_\Phi(a',a). \qquad (15)$$

It is easy to see that for the identity transfer function $\phi(r) = r$ we get $L(y,y') = (y-y')^2/2$, and for the logistic sigmoid $\phi(r) = 1/(1+\exp(-r))$ we get the logarithmic loss

$$L(y,y') = y\ln\frac{y}{y'} + (1-y)\ln\frac{1-y}{1-y'}.$$

Directly from the definition of matching loss, we obtain a simple expression for its gradient:

$$\nabla_{\mathbf{w}} L(y,\phi(\mathbf{w}\cdot\mathbf{x})) = (\phi(\mathbf{w}\cdot\mathbf{x}) - y)\mathbf{x}.$$

Therefore, the explicit update (11) naturally generalizes to

$$\mathbf{w}_t = \mathbf{f}^{-1}(\mathbf{f}(\mathbf{w}_{t-1}) - \eta(\hat{y}_t - y_t)\mathbf{x}_t)$$

where $\hat{y}_t = \phi(\mathbf{w}_{t-1}\cdot\mathbf{x}_t)$. The implicit algorithm can be generalized similarly; for it we use $\hat{y}_t = \phi(\mathbf{w}_t\cdot\mathbf{x}_t)$. For these algorithms we can now prove bounds which have as an additional factor an upper bound on the slope of the transfer function. The techniques are essentially those introduced by Helmbold *et al.* (1999).

Theorem 6. Fix p and q such that $1/p + 1/q = 1$ and $2 \le p < \infty$. Let ϕ be strictly increasing and continuously differentiable with c such that $0 < \phi(r) \le c$ holds for all r, and let L be the matching loss for ϕ. Assume that $||\mathbf{x}_t||_p \le X_p$ for all t. Then both the explicit algorithm and implicit algorithm for d_q with learning rate $\eta = 1/((p-1)cX_p^2)$ satisfy

$$\sum_{t=1}^{T} L(\hat{y}_t, \phi(\mathbf{u}\cdot\mathbf{x}_t)) \le \sum_{t=1}^{T} L(y_t, \phi(\mathbf{u}\cdot\mathbf{x}_t)) + (p-1)cX_p^2||\mathbf{u}||_q^2$$

for any $\mathbf{u} \in \mathbf{R}^n$.

Because of how we defined $\hat{y}_t$, the theorem gives an a priori filtering bound for the explicit algorithm and a posteriori bound for the implicit algorithm.

When ϕ is the identity function, we get the results of Section 3 with $c = 1$. For the logistic sigmoid, $c = 1/4$. Thresholded transfer functions, such as $\phi(r) = \mathrm{sign}(r)$, correspond to the limiting case $c \to \infty$, which makes the bound vacuous.

This result generalizes to the nonstationary case (Section 4) in the obvious manner; we omit the details.

6. REFERENCES

Auer, P., M. Herbster and M. K. Warmuth (1995). Exponentially many local minima for single neurons. In: *Proc. 1995 Neural Information Processing Conference*. MIT Press, Cambridge, MA. pp. 316–317.

Azoury, Katy S. and M. K. Warmuth (2001). Relative loss bounds for on-line density estimation with the exponential family of distributions. *Machine Learning* **43**(3), 211–246.

Bregman, L.M. (1967). The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Physics* **7**, 200–217.

Cesa-Bianchi, N., P. Long and M.K. Warmuth (1996). Worst-case quadratic loss bounds for on-line prediction of linear functions by gradient descent. *IEEE Transactions on Neural Networks* **7**(2), 604–619.

Gentile, Claudio and Nick Littlestone (1999). The robustness of the p-norm algorithms. In: *Proc. 12th Annu. Conf. on Comput. Learning Theory*. ACM Press, New York, NY. pp. 1–11.

Grove, Adam J., Nick Littlestone and Dale Schuurmans (2001). General convergence results for linear discriminant updates. *Machine Learning* **43**(3), 173–210.

Hassibi, Babak, Ali H. Sayed and Thomas Kailath (1996). H^∞ optimality of the LMS algorithm. *IEEE Transactions on Signal Processing* **44**(2), 267–280.

Helmbold, David P., Jyrki Kivinen and Manfred K. Warmuth (1999). Relative loss bounds for single neurons. *IEEE Transactions on Neural Networks* **10**(6), 1291–1304.

Herbster, Mark and Manfred K. Warmuth (2001). Tracking the best linear predictor. *Journal of Machine Learning Research* **1**, 281–309.

Khardon, R., D. Roth and R. A. Servedio (2002). Efficiency versus convergence of boolean kernels for on-line learning algorithms. In: *Advances in Neural Information Processing Systems 14* (T. G. Dietterich, S. Becker and Z. Ghahramani, Eds.). MIT Press. Cambridge, MA. pp. 423–430.

Kivinen, J. and M. K. Warmuth (1997). Additive versus exponentiated gradient updates for linear prediction. *Information and Computation* **132**(1), 1–64.

Kivinen, J. and M. K. Warmuth (2001). Relative loss bounds for multidimensional regression problems. *Machine Learning* **45**(3), 301–329.

Kivinen, J., M. K. Warmuth and P. Auer (1997). The Perceptron algorithm vs. Winnow: Linear vs. logarithmic mistake bounds when few input variables are relevant. *Artificial Intelligence* **97**, 325–343.

Takimoto, Eiji and Manfred K. Warmuth (2002). Path kernels and multiplicative updates. In: *Proceedings of the 15th Annual Conference on Computational Learning Theory* (Jyrki Kivinen and Bob Sloan, Eds.). Springer LNAI 2375. Berlin. pp. 74–89.