Journal of Computer and System Sciences **64**, 48–75 (2002)
doi:10.1006/jcss.2001.1795, available online at http://www.idealibrary.com on IDEAL®

Adaptive and Self-Confident On-Line Learning Algorithms

Institute for Theoretical Computer Science, Graz University of Technology, Klosterwiesgasse 32/2, A-8010 Graz, Austria
E-mail: pauer@igi.tu-graz.ac.at

Nicolò Cesa-Bianchi

Department of Information Technologies, Università di Milano, Via Bramante 65, 26013 Crema, Italy
E-mail: nicolo.cesa-bianchi@unimi.it

and

Claudio Gentile

Department of Information Sciences, Università di Milano, Via Comelico 39, 20135 Milano, Italy
E-mail: gentile@dsi.unimi.it

Received October 6, 2000; revised March 12, 2001

We study on-line learning in the linear regression framework. Most of the performance bounds for on-line algorithms in this framework assume a constant learning rate. To achieve these bounds the learning rate must be optimized based on *a posteriori* information. This information depends on the whole sequence of examples and thus it is not available to any strictly on-line algorithm. We introduce new techniques for adaptively tuning the learning rate as the data sequence is progressively revealed. Our techniques allow us to prove essentially the same bounds as if we knew the optimal learning rate in advance. Moreover, such techniques apply to a wide class of on-line algorithms, including p-norm algorithms for generalized linear regression and Weighted Majority for linear regression with absolute loss. Our adaptive tunings are radically different from previous techniques, such as the so-called doubling trick. Whereas the doubling trick restarts the on-line algorithm several times using a constant learning rate for each run, our methods save information by changing the value of the learning rate very smoothly. In fact, for Weighted Majority over a finite set of experts our analysis provides a better leading constant than the doubling trick.

Key Words: on-line prediction; linear regression; quasi-additive algorithms; learning rate.

0022-0000/02 $35.00

1. INTRODUCTION

In this paper, we study on-line learning from labeled examples. Broadly speaking, the on-line learning task consists of the following. In each on-line trial the algorithm receives an *instance* and is required to output a prediction about the *label* associated with that instance. Then the label is revealed and the algorithm suffers a *loss*, quantifying the "distance" between the algorithm's prediction and the actual label. The goodness of the algorithm is measured by comparing its performance to the performance of the best off-line predictor in a given *comparison class* of predictors.

In this paper, we focus on the family of *quasi-additive algorithms* (Grove *et al.* [17], Kivinen and Warmuth [23]) applied to on-line linear regression problems. The algorithms in this family are efficient and easy to implement. They work well with various losses and, furthermore, they are very robust to noise; i.e., one can prove that their cumulative loss on *any* data sequence cannot grow much faster than the loss of the best off-line linear predictor on the same data sequence. However, to achieve these bounds, the parameters of the algorithms have to be set depending on features of the learning task that are typically not known a priori. For instance, for many on-line algorithms the optimal tuning of the learning rate does depend on information about the amount of noise in the data. In an on-line setting this information is typically not available.

In general, tuning is certainly one of the most critical aspects of an on-line learning algorithm and might affect its performance in a substantial way.

In this introductory section we begin by using a version of the Weighted Majority algorithm [7, 26, 29, 32, 34] as a motivating example to illustrate the tuning problem we are interested in. We then briefly overview the much more general class of quasi-additive algorithms [17, 23]. Finally, we introduce our tuning techniques and compare them to those already available.

1.1. An Illustrating Example

The Weighted Majority[1] algorithm processes the examples one at a time in trials. In each trial t the algorithm receives a vector $\mathbf{x}_t = (x_{t,1}, \ldots, x_{t,n}) \in [-1, +1]^n$ and is asked to predict the value of an unknown label $y_t \in [-1, +1]$ associated with $\mathbf{x}_t$. The algorithm keeps a weight vector $\mathbf{w}_t = (w_{t,1}, \ldots, w_{t,n})$ representing its current hypothesis. The vector $\mathbf{w}_t$ is an n-dimensional probability vector. The algorithm's prediction at time t is the linear combination $\hat{y}_t = \mathbf{w}_t \cdot \mathbf{x}_t \in [-1, +1]$. After receiving the correct label y_t, the algorithm incurs a loss $l_t = \frac{1}{2}|y_t - \hat{y}_t| \in [0, 1]$. Finally, the algorithm updates the weights according to the rule

$$w_{t+1,i} = w_{t,i} \exp\{-\eta \tfrac{1}{2}|y_t - x_{t,i}|\}/W_{t+1},$$

[1] The version of the Weighted Majority algorithm we are considering here is called WMC in [29]. This will be our leading example. As a matter of fact, the main points of the following discussion apply to many of the on-line learning algorithms which have been proposed in the literature.

where η is the *learning rate*, $\frac{1}{2}|y_t - x_{t,i}|$ is the loss of the ith input component, and W_{t+1} is the normalizing factor making $\mathbf{w}_{t+1}$ a probability vector. Now, let $L_{i,T} = \sum_{t=1}^{T} \frac{1}{2}|y_t - x_{t,i}|$ be the *cumulative* loss of the ith input component on a sequence of T trials. A (simplified) analysis of the Weighted Majority algorithm with fixed η shows that, up to lower-order terms, the cumulative loss $L_T = \sum_{t=1}^{T} l_t$ can be upper bounded as

$$L_T \leqslant (1+\eta)\, L_T^* + \frac{1+\eta}{\eta} \ln n, \tag{1}$$

where $L_T^* = \min_{1 \leqslant i \leqslant n} L_{i,T}$. To obtain an optimal bound (up to lower-order terms) the learning rate η has to be chosen with respect to L_T^*. For instance, bound (1) is optimized by $\eta = \sqrt{\ln n / L_T^*}$ which yields

$$L_T \leqslant L_T^* + 2\sqrt{L_T^* \ln n} + \ln n.$$

Thus, according to the last bound, the loss of the algorithm is asymptotically the same as the loss of the best fixed component, if we disregard lower-order terms. Obviously this tuning needs *a priori* knowledge about the optimal cumulative loss L_T^*, which is usually not available. We will solve this tuning problem in Section 2 and in Section 3 by using an adaptive learning rate η_t that varies over time, depending on the information the algorithm gains about L_T^* during the learning process.

1.2. Incremental Update vs the Doubling Trick

A standard method to deal with the above tuning problem is the so-called "doubling trick": An upper bound B on L_T^* is assumed and the learning rate is tuned with respect to this bound. For the simple version of the Weighted Majority algorithm we just mentioned this would be $\eta = \sqrt{\ln n / B}$. We call "round" a sequence of trials where B is constant. If during the learning process the loss of the best component exceeds bound B, then this bound is increased, the learning algorithm is restarted, and a new round begins. Doubling strategies for particular on-line algorithms have been analyzed in [7, 8].

We may say that the doubling trick makes an on-line algorithm coarsely adaptive, as the learning rate is constant within a round and makes big jumps between rounds. However, a major disadvantage is that the on-line algorithm is restarted from scratch at the beginning of each round, hence losing all the information collected during the past rounds. More disadvantages arise if the learning setting is made more general. For instance, in generalized linear regression [23] checking the termination condition for the current round might be computationally expensive (this involves computing the cumulative loss of the best regressor so far). Furthermore, it is not clear how the doubling trick could be analyzed when the loss in each trial can be arbitrarily large.

In contrast, this paper analyzes on-line learning algorithms that are "incrementally adaptive," as they modify their learning rates possibly in each trial, and

typically by a small amount. Via our approach we design algorithms whose performance bounds are in some cases better, and never significantly worse, than those proven for the doubling trick. Moreover, some of our techniques are efficiently applicable to general learning settings and can even handle unbounded loss functions.

Also, it is worth emphasizing that tuning techniques based on the doubling trick are not expected to work well in many practical situations. This is because in each round the algorithm does actually *achieve* (or even violate) the postulated worst-case bound before it is restarted from scratch. This feature seems to make doubling strategies attractive only in worst-case settings.

As an aside, we note that algorithms with an incrementally adaptive learning rate have also been investigated by Vovk [33] and Azoury and Warmuth [3] for the problem of on-line linear regression with square loss. The algorithms they study are related to the recursive least squares algorithm. The covariance matrices involved in those algorithms can be naturally interpreted as learning rates. However, their proof techniques are different from ours and do not seem easily extendible to regression problems using loss function different from the square loss.

1.3. The Formal Learning Model

In this section we describe the learning model more precisely and give our basic notation.

An *example* is a pair $(\mathbf{x}, y)$, where $\mathbf{x} \in \mathbb{R}^n$ is called an *instance*, and $y \in \mathbb{R}$ is the *label* associated with $\mathbf{x}$. On-line learning proceeds in trials. In the tth trial the on-line algorithm receives an instance $\mathbf{x}_t$ and is required to give a *prediction* $\hat{y}_t$ about the unknown label y_t associated with $\mathbf{x}_t$. Then y_t is revealed and the algorithm incurs a loss $L(y_t, \hat{y}_t)$, measuring the discrepancy between the prediction $\hat{y}_t$ and the label y_t. We call a sequence $S = ((\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots)$ of instances and labels processed by the algorithm in a run a *trial sequence*.

To analyze these algorithms, we adopt a well-established mathematical model which is a generalization of a learning model introduced by Littlestone and Warmuth [25, 26, 29] and Angluin [1]. We are given a *comparison class* of predictors and a loss function L. We measure the performance of A on the sequence S by the cumulative loss $L_{A,T}(S)$ the algorithm A suffers on S: $L_{A,T}(S) = \sum_{t=1}^{T} L(y_t, \hat{y}_t)$. We compare this loss to the loss of the comparison class, i.e., to the loss of the best predictor in the comparison class for the same trial sequence S.

This paper focuses on the linear regression problem with the square loss and the absolute loss. Such problems have been widely investigated in the last years (see, e.g., Littlestone [26, 27], Vovk [32–35], Littlestone and Warmuth [29], Littlestone *et al.* [28], Cesa-Bianchi *et al.* [7–9], Kivinen and Warmuth [22, 23], Yamanishi [37], Grove *et al.* [17], Gentile and Littlestone [14], Azoury and Warmuth [3], and references therein).

In linear regression the learner's hypothesis at time t is represented by a weight vector $\mathbf{w}_t \in \mathbb{R}^n$, and the prediction $\hat{y}_t$ is often a function of $\mathbf{w}_t \cdot \mathbf{x}_t$. For instance, if L is the square loss $L(y_t, \hat{y}_t) = \frac{1}{2}(y_t - \hat{y}_t)^2$ and the prediction is just $\hat{y}_t = \mathbf{w}_t \cdot \mathbf{x}_t$, then we compare the cumulative loss $L_{A,T}(S)$ of the algorithm with the least cumulative

square loss that could be incurred by predicting with a fixed weight vector $\mathbf{u}$ in the comparison class. In particular, setting $L_{\mathbf{u},T}(S) = \sum_{t=1}^{T} L(y_t, \mathbf{u} \cdot \mathbf{x}_t)$, our goal is to bound the loss difference (this is also called the *regret* or *relative loss*)

$$L_{A,T}(S) - \min_{\mathbf{u}} L_{\mathbf{u},T}(S),$$

for an arbitrary trial sequence S.

In the absolute loss setting we have $L(y_t, \hat{y}_t) = \frac{1}{2}|y_t - \hat{y}_t|$. In this paper we only consider the case when the labels y_t have range $[-1, +1]$. The prediction $\hat{y}_t$ is a suitable function of $\mathbf{w}_t \cdot \mathbf{x}_t$, whose range is again $[-1, +1]$. As a consequence, $L(y_t, \hat{y}_t) \in [0, 1]$. We are still aimed at bounding the cumulative (absolute) loss of A, but the way we measure the performance of the best off-line $\mathbf{u}$ might be different (see Section 3). The case $y_t \in \{-1, +1\}$ is of special interest, since it can be interpreted as a binary classification problem where the algorithm is allowed to make randomized predictions. The absolute loss $\frac{1}{2}|y_t - \hat{y}_t|$ is then the probability of a prediction mistake, i.e., the probability that $y_t \neq \hat{y}_t$, and the cumulative loss is just the expected number of mistakes. The Weighted Majority algorithm of Section 1.1 can be seen as an algorithm for the following restricted class of regression problems: the loss function is the absolute loss $L(y_t, \hat{y}_t) = \frac{1}{2}|y_t - \hat{y}_t|$, the prediction is $\hat{y}_t = \mathbf{w}_t \cdot \mathbf{x}_t$, and the comparison class is the set of the n unit vectors, i.e., the rows of the $n \times n$ identity matrix. These are often called "experts" in the literature (e.g., [7, 24] and references therein). See also Section 3.1.

1.4. Quasi-additive Learning Algorithms

In this section we briefly overview the class of *quasi-additive* algorithms, i.e., the class of on-line algorithms we are interested in.

This class of algorithms have been introduced by Grove *et al.* [17] in the context of binary classification and also, independently, by Kivinen and Warmuth [23] in the context of (generalized) linear regression. These algorithms are called *quasi-additive* in [17] and *general additive* in [23]. This class includes a wide variety of learning algorithms. For instance, in the binary classification setting it includes the Perceptron algorithm [4, 30, 31] and algorithms in the Winnow family [25, 26], such as the Weighted Majority algorithm called WM in [29]; in the regression setting it includes the Widrow–Hoff rule [36] and algorithms in the EG family [22].

All these algorithms have the same basic structure. In the generic trial t the algorithm stores the weight vector $\mathbf{w}_t$, lying in a suitable weight space. Combined with the current instance $\mathbf{x}_t$, the vector $\mathbf{w}_t$ determines the algorithm's prediction $\hat{y}_t$, which is a function of $\mathbf{w}_t \cdot \mathbf{x}_t$. Then, based on the label y_t, the algorithm performs the weight update step $\mathbf{w}_t \to \mathbf{w}_{t+1}$. At the core of the weight update lies the rule

$$\mathbf{w}_{t+1} = \mathbf{f}^{-1}(\mathbf{f}(\mathbf{w}_t) + \eta_t g(y_t, \hat{y}_t)\, \mathbf{x}_t), \tag{2}$$

where $\mathbf{f}$ is a smooth bijective mapping from the adopted weight space to $\mathbb{R}^n$, $\mathbf{f}^{-1}$ is the inverse of $\mathbf{f}$, and g is a suitable function of y_t and $\hat{y}_t$. For instance, in linear

regression with square loss we have $g(y_t, \hat{y}_t) = y_t - \hat{y}_t$ and the vector $g(y_t, \hat{y}_t)\,\mathbf{x}_t$ is just the gradient of the square loss $\frac{1}{2}(y_t - \mathbf{w}\cdot\mathbf{x}_t)^2$ w.r.t. vector $\mathbf{w}$. The Widrow–Hoff rule is then obtained when $\mathbf{f}$ is the identity mapping, while the EGU algorithm [22] is given by the componentwise logarithm $\mathbf{f}(\mathbf{w}_t) = (\ln w_{t,1}, \ldots, \ln w_{t,n})$. The version of the Weighted Majority algorithm we mentioned in Section 1.1 can be seen as a member of the quasi-additive family once we set $g(y_t, \hat{y}_t) = 1$ and interpret the n-component vector $\mathbf{x}_t$ as the vector of *losses* of the n predictors in the comparison class [24]. The bijective mapping $\mathbf{f}$ giving rise to the Weighted Majority algorithm is [23, Example 3] $\mathbf{f} = (f_1, \ldots, f_{n-1})$: $\mathscr{W} \subseteq \mathbb{R}^{n-1} \to \mathbb{R}^{n-1}$, where $\mathscr{W} = \{w_1, \ldots w_{n-1} \mid \sum_{i=1}^{n-1} w_i \leqslant 1\}$ and

$$f_i(\mathbf{w}) = \ln \frac{w_i}{1 - \sum_{i=1}^{n-1} w_i}. \tag{3}$$

The standard way to analyze these algorithms (e.g., [2, 3, 6–8, 13, 14, 17, 18, 22, 25, 26, 29]) is to define a measure of progress related to the mapping $\mathbf{f}$. The measure of progress we use here is the so-called *Bregman divergence* [5, 10] associated with $\mathbf{f}$. We denote the divergence by $d_{\mathbf{f}}(\mathbf{u}, \mathbf{w})$, where $\mathbf{u}$ and $\mathbf{w}$ are weight vectors. We can define $d_{\mathbf{f}}(\mathbf{u}, \mathbf{w})$ as follows. Assume $\mathbf{f}$ is the gradient of some convex function $P_{\mathbf{f}}$ on a convex weight space. Then $d_{\mathbf{f}}(\mathbf{u}, \mathbf{w})$ is the difference between $P_{\mathbf{f}}(\mathbf{u})$ and the first-order Taylor expansion of $P_{\mathbf{f}}$ around $\mathbf{w}$, i.e.,

$$d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}) = P_{\mathbf{f}}(\mathbf{u}) - P_{\mathbf{f}}(\mathbf{w}) - (\mathbf{u} - \mathbf{w}) \cdot \mathbf{f}(\mathbf{w}). \tag{4}$$

The convexity of $P_{\mathbf{f}}$ ensures $d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}) \geqslant 0$. Also, if $P_{\mathbf{f}}$ is *strictly* convex then $d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}) = 0$ if and only if $\mathbf{u} = \mathbf{w}$. For example, if $\mathbf{f}$ is the identity then $d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}) = \frac{1}{2}\|\mathbf{u} - \mathbf{w}\|_2^2$, whereas if $\mathbf{f}$ is the mapping (3) then it is not hard to show that setting $w_n = 1 - \sum_{i=1}^{n-1} w_i$ and $u_n = 1 - \sum_{i=1}^{n-1} u_i$ leads to the relative entropy divergence $d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}) = \sum_{i=1}^{n} u_i \ln(u_i / w_i)$.

A further example of Bregman divergence is provided in Section 3.1, when dealing with the sub-family of p-norm algorithms [14, 17]. A good reference to learn about Bregman divergences is [11]. Azoury and Warmuth [3] give a valuable summary of the main properties of Bregman divergences in the context of on-line learning with the exponential family of distributions.

1.5. Two Approaches to Incrementally Adaptive Learning Rates

In this section we distinguish two ways how we will tune the learning rates of the algorithms. The most obvious way is to set the learning rate η_t with respect to the loss of the comparison class observed so far. For the Weighted Majority algorithm of Section 1.1 this would be $\eta_t = \sqrt{\ln n / L_t^*}$, where

$$L_t^* = \min_{1 \leqslant i \leqslant n} \sum_{\tau=1}^{t} \tfrac{1}{2} |y_\tau - x_{\tau,i}|.$$

It should be clear that a tuning based on the true current loss of the comparison class can be applied in general only when the comparison class is *finite*. When

moving to harder regression frameworks (such as those considered in Section 3) the evaluation of the current loss of the comparison class might be computationally intensive: storing all past examples and resorting to numerical methods might be necessary for loss functions more difficult than the square loss. Observe that different loss functions give rise to different computational problems.

An alternative way for tuning the learning rate η_t is to use the loss of the learning algorithm to calculate η_t. For the Weighted Majority algorithm of Section 1.1 this would roughly be $\eta_t \simeq \sqrt{\ln n / L_t}$, where

$$L_t = \sum_{\tau=1}^{t} \tfrac{1}{2} |y_\tau - \hat{y}_\tau|.$$

This is done under the assumption that $L_t \approx L_t^*$, i.e., that the current cumulative loss of the algorithm closely matches the current cumulative loss of the comparison class. We call such algorithms *self-confident*, as they somehow "trust themselves" in tuning η_t.

The self-confident tuning might be viewed as a computationally efficient approximation to the tuning based on the loss of the comparison class. The self-confident tuning allows us to circumvent the computational problems deriving from hard loss functions and, furthermore, its analysis seems to be much simpler in most cases. In fact, we have been able to analyze the tuning based on the current loss of the comparison class only for the relatively simple Weighted Majority algorithm (Section 2). Self-confident learning algorithms will be analyzed in Section 3.

1.6. Overview of Proof Techniques

The analysis of quasi-additive algorithms with constant learning rate η often centers on inequalities of the form

$$(1-\alpha)\, l_t - l_{\mathbf{u},t} \leqslant \frac{1}{\alpha} \left(d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_t) - d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_{t+1})\right), \tag{5}$$

where l_t is the loss of the algorithm in trial t, $l_{\mathbf{u},t}$ is the loss of the generic off-line predictor $\mathbf{u}$ in trial t, and $\alpha \in (0, 1)$ is a constant proportional to η. The difference $d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_t) - d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_{t+1})$ might be considered as the one-trial progress of the algorithm's weight vector $\mathbf{w}_t$ towards the best off-line vector $\mathbf{u}$ within the comparison class. Therefore inequalities such as (5) connect the one-trial relative loss $l_t - l_{\mathbf{u},t}$ to the algorithm's progress in that trial. Set for brevity $L_t = \sum_{\tau=1}^{t} l_\tau$ and $L_{\mathbf{u},t} = \sum_{\tau=1}^{t} l_{\mathbf{u},\tau}$. A cumulative loss bound of the form[2]

$$L_T \leqslant \frac{1}{1-\alpha} L_{\mathbf{u},T} + \frac{1}{\alpha(1-\alpha)} d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_1)$$

[2] Observe that, as far as the dependence on the learning rate is concerned, this bound is qualitatively analogous to bound (1).

For $t = 1, 2, \ldots, T$:

- Calculate the weights as

$$w_{t,i} = \alpha^{-L_{i,t-1}}/W_t, \qquad i = 1, \ldots, n,$$

$$\varepsilon_t = \min\left\{\frac{1}{4}, \sqrt{2\frac{\ln n}{L^*_{t-1}}}\right\},$$

$$\alpha_t = \frac{1}{1-\varepsilon_t},$$

$$W_t = \sum_{i=1}^{n} \alpha_t^{-L_{i,t-1}};$$

- Get instance $\mathbf{x}_t \in [-1, +1]^n$;
- Predict with $\hat{y}_t = \mathbf{w}_t \cdot \mathbf{x}_t$;
- Get label $y_t \in [-1, +1]$;
  Incur loss $\frac{1}{2}|y_t - \hat{y}_t|$.

FIG. 1. The incrementally adaptive Weighted Majority algorithm IAMW.

is immediately obtained by summing (5) over all trials t, solving for L_T, and dropping the last (non-negative) divergence term $d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_{T+1})$.

Our adaptive learning rate analysis is based on more sophisticated versions of (5), where we let α depend on time. Denote the time-dependent α by α_t. In Section 2 we will deal with a version of the Weighted Majority algorithm tuned via the loss of the comparison class so far. The Bregman divergence associated with this algorithm is the relative entropy $d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_t) = \sum_{i=1}^{n} u_i \ln(u_i/w_{t,i})$. Since the comparison class is the set of the n unit vectors, the relative entropy reduces to $d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_t) = -\ln w_{t,i}$, where $\mathbf{w}_t = (w_{t,1}, \ldots w_{t,n})$ and $\mathbf{u}$ is the ith unit vector. We will bound a progress of the form

$$\frac{-\ln w_{t,i^*_t}}{\alpha_t} - \frac{-\ln w_{t+1,i^*_{t+1}}}{\alpha_{t+1}}, \tag{6}$$

where i^*_t is the component i with the minimal cumulative loss L^*_t up to that trial and α_t roughly equals $\sqrt{\ln n/L^*_t}$.

In Section 3 we will deal with the more general family of quasi-additive algorithms tuned via the loss of the algorithm so far (self-confident tuning). We will bound a progress of the form

$$\frac{d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_t)}{\alpha_t} - \frac{d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_{t+1})}{\alpha_{t+1}}, \tag{7}$$

where d_f is the Bregman divergence associated with the algorithm under consideration[3] and α_t is roughly proportional to $1/\sqrt{L_t}$.

Both progress (6) and progress (7) will be suitably related to their corresponding one-trial relative losses.

2. AN INCREMENTALLY ADAPTIVE WEIGHTED MAJORITY

In this section we present an incrementally adaptive version of the Weighted Majority algorithm described in Section 1.1. We call the algorithm IAWM. Recall the notation of Section 1. The structure of algorithm IAWM, see Fig. 1, is quite straightforward from the Weighted Majority algorithm of Section 1.1. Essentially, the only modification necessary is the introduction of a varying learning rate. (In this section we found it more convenient to set $\eta_t = \ln \alpha_t$ and focus on the tuning of α_t.) There is a subtle point here, though, since the learning rate is changed also retrospectively: In the first step of the algorithm the weight $w_{t,i}$ is set proportionally to $\alpha_t^{-L_{i,t-1}}$, whereas one might expect that it is set to $w_{t-1,i}\alpha_t^{-|y_t - x_{t,i}|}$. Setting the weight to $\alpha_t^{-L_{i,t-1}}$ actually means that the new learning rate parameter α_t is applied to all past trials. This is quite essential for the analysis. Notice that in Fig. 1 it is understood that $L_{i,0} = L_0^* = 0$. Thus $w_{1,i} = 1/n$ and $\varepsilon_1 = 1/4$. For this algorithm we have the following result.

THEOREM 2.1. *Let $S = ((\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots)$, where each $(\mathbf{x}_t, y_t)$ belongs to the set $[-1, +1]^n \times [-1, +1]$. Then the incrementally adaptive Weighted Majority algorithm IAWM of Fig. 1, run on a prefix of S of arbitrary length T, achieves the following cumulative absolute loss bound,*

$$L_T \leqslant L_T^* + 2\sqrt{2L_T^* \ln n} + 4(\ln n)(\ln(1 + L_T^*)) + 10 \ln n + \tfrac{3}{10},$$

where

$$L_T^* = \min_{1 \leqslant i \leqslant n} \sum_{t=1}^{T} \tfrac{1}{2} |y_t - x_{t,i}|$$

and $n \geqslant 2$.

Proof. We will bound

$$\frac{\ln w_{t+1, i_{t+1}^*}}{\varepsilon_{t+1}} - \frac{\ln w_{t, i_t^*}}{\varepsilon_t},$$

[3] Our self-confident analysis requires the divergence $d_f(\mathbf{u}, \mathbf{w}_t)$ be bounded for any t. In the case of p-norm algorithms (Section 3.1) this is achieved by both assuming the comparison vectors $\mathbf{u}$ are bounded and keeping the weight vectors $\mathbf{w}_t$ produced by rule (2) bounded through Bregman divergence projection. The case of algorithms in the Winnow/EG family is a bit more troublesome. See the discussion in Section 3.2.

where $i_t^* = \arg\max_i w_{t,i}$ is the component with the minimal cumulative loss up to the *beginning* of trial t (i.e., before receiving the tth label y_t). Observe that with this notation we have $L_{i_{t+1}^*,t} = L_t^*$, which is the loss of the component with the minimal cumulative loss up to the *end* of trial t (i.e., after receiving the tth label y_t).

Set $W'_t = \sum_{i=1}^n \alpha_t^{-L_{i,t}}$ so that $w'_{t,i} = \alpha_t^{-L_{i,t}}/W'_t$ would be the weights for the next trial if the learning rate were not changed. We get

$$\begin{aligned}
&\frac{\ln w_{t+1,i_{t+1}^*}}{\varepsilon_{t+1}} - \frac{\ln w_{t,i_t^*}}{\varepsilon_t} \\
&\quad = \ln w_{t+1,i_{t+1}^*}\left(\frac{1}{\varepsilon_{t+1}} - \frac{1}{\varepsilon_t}\right) + \frac{\ln w_{t+1,i_{t+1}^*} - \ln w'_{t,i_{t+1}^*}}{\varepsilon_t} + \frac{\ln w'_{t,i_{t+1}^*} - \ln w_{t,i_t^*}}{\varepsilon_t} \\
&\quad \geqslant -\ln n\left(\frac{1}{\varepsilon_{t+1}} - \frac{1}{\varepsilon_t}\right) + \frac{1}{\varepsilon_t}\ln\frac{\alpha_{t+1}^{-L_t^*}W'_t}{\alpha_t^{-L_t^*}W_{t+1}} + \frac{1}{\varepsilon_t}\ln\frac{\alpha_t^{-L_t^*}}{\alpha_t^{-L_{t-1}^*}} + \frac{1}{\varepsilon_t}\ln\frac{W_t}{W'_t},
\end{aligned} \tag{8}$$

where in the inequality we have used $w_{t+1,i_{t+1}^*} \geqslant 1/n$ and $\varepsilon_{t+1} \leqslant \varepsilon_t$.

We denote the last three logarithmic factors in the right-most side of (8) as

$$B_{1,t} = \ln\frac{\alpha_{t+1}^{-L_t^*}W'_t}{\alpha_t^{-L_t^*}W_{t+1}}, \qquad B_{2,t} = \ln\frac{\alpha_t^{-L_t^*}}{\alpha_t^{-L_{t-1}^*}}, \qquad B_{3,t} = \ln\frac{W_t}{W'_t}.$$

We proceed by lower bounding $B_{1,t}$, $B_{2,t}$, and $B_{3,t}$. To bound $B_{1,t}$ we use the following technical lemma whose proof is given in the Appendix.

LEMMA 2.1. *For $1 < \alpha \leqslant \beta$ and any $\ell_1, \ldots, \ell_n \geqslant 0$ such that $\sum_{i=1}^n \alpha^{-\ell_i} \geqslant 1$ it holds that*

$$\ln\left(\frac{\sum_{i=1}^n \beta^{-\ell_i}}{\sum_{i=1}^n \alpha^{-\ell_i}}\right) \geqslant -(\beta-\alpha)\frac{\ln n}{\ln\alpha}.$$

Since $1 < \alpha_{t+1} \leqslant \alpha_t$ and

$$\frac{\alpha_{t+1}^{-L_t^*}W'_t}{\alpha_t^{-L_t^*}W_{t+1}} = \frac{\sum_{i=1}^n \alpha_t^{-L_{i,t}+L_t^*}}{\sum_{i=1}^n \alpha_{t+1}^{-L_{i,t}+L_t^*}} = \frac{1+\sum_{i=1}^{n-1}\alpha_t^{-\ell_i}}{1+\sum_{i=1}^{n-1}\alpha_{t+1}^{-\ell_i}},$$

for some $\ell_1, \ldots, \ell_{n-1} \geqslant 0$, we get

$$B_{1,t} \geqslant -(\alpha_t - \alpha_{t+1})\frac{\ln n}{\ln\alpha_{t+1}}. \tag{9}$$

We distinguish three cases:

1. trial t is such that $\alpha_t \neq \alpha_{t+1}$ and $\varepsilon_t < 1/4$;
2. trial t is such that $\alpha_t \neq \alpha_{t+1}$ and $\varepsilon_t = 1/4$;
3. trial t is such that $\alpha_t = \alpha_{t+1}$.

Case 1. Since α_t as a function of L^*_{t-1} is convex, a first-order Taylor expansion yields

$$\alpha_{t+1}-\alpha_t \geqslant (L^*_t-L^*_{t-1})\frac{\partial}{\partial L^*_{t-1}}\alpha_t = -(L^*_t-L^*_{t-1})\,\alpha_t^2\sqrt{\frac{\ln n}{2(L^*_{t-1})^3}}$$
$$= -(L^*_t-L^*_{t-1})\,\alpha_t^2\frac{\varepsilon_t^3}{4\ln n}.$$

For similar reasons

$$\frac{1}{\ln \alpha_{t+1}} \leqslant \frac{1}{\varepsilon_{t+1}} \leqslant \frac{1}{\varepsilon_t}+(L^*_t-L^*_{t-1})\frac{\partial}{\partial L^*_{t-1}}\frac{1}{\varepsilon_t}$$
$$= \frac{1}{\varepsilon_t}+(L^*_t-L^*_{t-1})\frac{1}{\varepsilon_t^2}\sqrt{\frac{\ln n}{2(L^*_{t-1})^3}}$$
$$\leqslant \frac{1}{\varepsilon_t}\left(1+\frac{\varepsilon_t^2}{4\ln n}\right),$$

since $L^*_t-L^*_{t-1}\leqslant 1$. Putting together as in (9) yields

$$B_{1,t} \geqslant -(L^*_t-L^*_{t-1})\frac{\alpha_t^2\varepsilon_t^2}{4}\left(1+\frac{\varepsilon_t^2}{4\ln n}\right) \geqslant -(L^*_t-L^*_{t-1})\frac{\alpha_t^2\varepsilon_t^2}{4}\left(1+\frac{\varepsilon_t^2}{4\ln 2}\right)$$
$$\geqslant -(L^*_t-L^*_{t-1})\frac{\varepsilon_t^2}{4}(1+4\varepsilon_t),$$

where in the second inequality we used $n\geqslant 2$ and in the last inequality we used the bound $(1/(1-\varepsilon)^2)(1+\varepsilon^2/(4\ln 2))\leqslant 1+4\varepsilon$, for $0\leqslant\varepsilon\leqslant 1/4$.

Case 2. First of all, notice that there is at most one trial such that at $\alpha_t\neq\alpha_{t+1}$ and $\varepsilon_t=1/4$. Recall (9). For such a trial we have

$$-(\alpha_t-\alpha_{t+1})\frac{\ln n}{\ln \alpha_{t+1}} = \frac{1/4-\varepsilon_{t+1}}{3/4(1-\varepsilon_{t+1})}\frac{\ln n}{\ln(1-\varepsilon_{t+1})}, \tag{10}$$

where $1/4\geqslant\varepsilon_{t+1}\geqslant\sqrt{2\ln n/(1+32\ln n)}$. By a derivative argument it is not hard to see that (10) is increasing in ε_{t+1}. Thus the minimal value is obtained when we set $\varepsilon_{t+1}=\sqrt{2\ln n/(1+32\ln n)}$ therein. This substitution yields a function which is decreasing in n. Thus the minimal value is achieved when $n\to\infty$. Computing the limit gets

$$B_{1,t} \geqslant \frac{1}{144}\frac{1}{\ln(3/4)} > -\frac{1}{40}.$$

Case 3. It is trivially verified that $B_{1,t}=0$.

To summarize:

$$B_{1,t} \geqslant \begin{cases} -(L_t^* - L_{t-1}^*)\dfrac{\varepsilon_t^2}{4}(1+4\varepsilon_t) & \text{if } \alpha_t \neq \alpha_{t+1} \text{ and } \varepsilon_t < 1/4; \\ -\dfrac{1}{40} & \text{if } \alpha_t \neq \alpha_{t+1} \text{ and } \varepsilon_t = 1/4; \\ 0 & \text{otherwise.} \end{cases}$$

Next, we turn to bounding $B_{2,t}$ from below. Using the inequality $\ln(1-\varepsilon) \geqslant -\varepsilon - \varepsilon^2/2 - \varepsilon^3$, for $0 \leqslant \varepsilon \leqslant 1/4$, we obtain

$$B_{2,t} = \ln \frac{\alpha_t^{-L_t^*}}{\alpha_t^{-L_{t-1}^*}} \geqslant -\varepsilon_t(L_t^* - L_{t-1}^*)(1+\varepsilon_t/2+\varepsilon_t^2).$$

Finally, $B_{3,t}$ is lower bounded through the following chain of (in)equalities,

$$\begin{aligned} B_{3,t} &= \ln \frac{W_t}{W_t'} = -\ln\left(\sum_{i=1}^n w_{t,i}\alpha_t^{-|y_t - x_{t,i}|/2}\right) \\ &= -\ln\left(1 + \sum_{i=1}^n w_{t,i}(\alpha_t^{-|y_t - x_{t,i}|/2} - 1)\right) \\ &\geqslant -\sum_{i=1}^n w_{t,i}(\alpha_t^{-|y_t - x_{t,i}|/2} - 1) \\ &\geqslant \varepsilon_t \sum_{i=1}^n w_{t,i}\,|y_t - x_{t,i}|/2 \\ &\geqslant \varepsilon_t \left| y_t - \sum_{i=1}^n w_{t,i}x_{t,i} \right| \Big/ 2 \\ &= \varepsilon_t(L_t - L_{t-1}), \end{aligned}$$

where we used the very definitions of $B_{3,t}$, W_t, and W_t', the facts $\ln(1+x) \leqslant x$ for $x > -1$, $1-(1-\varepsilon)^x \geqslant \varepsilon x$ for $0 \leqslant \varepsilon, x \leqslant 1$, and the convexity of $|\cdot|$.

We now use the bounds on $B_{1,t}$, $B_{2,t}$, $B_{3,t}$ in (8) and sum for $t = 1, \ldots, T$. We obtain

$$\begin{aligned} 4\ln n &= -\frac{\ln w_{1,1}}{\varepsilon_1} \\ &\geqslant \frac{\ln w_{T+1,i_{T+1}^*}}{\varepsilon_{T+1}} - \frac{\ln w_{1,1}}{\varepsilon_1} \\ &\geqslant -\ln n\left(\frac{1}{\varepsilon_{T+1}} - \frac{1}{\varepsilon_1}\right) - \sum_{t:\,\alpha_{t+1} \neq \alpha_t,\, \varepsilon_t < 1/4} (L_t^* - L_{t-1}^*)\frac{\varepsilon_t}{4}(1+4\varepsilon_t) - \frac{1}{10} \\ &\quad - \sum_{t=1}^T (L_t^* - L_{t-1}^*)(1+\varepsilon_t/2+\varepsilon_t^2) + L_T. \end{aligned}$$

Further manipulation yields

$$
\begin{aligned}
4\ln n &\geqslant -\frac{\ln n}{\varepsilon_{T+1}} + 4\ln n - \sum_{t:\, L_t^* \neq L_{t-1}^*,\, \varepsilon_t < 1/4} (L_t^* - L_{t-1}^*)\,\frac{\varepsilon_t}{4}\,(1+4\varepsilon_t) - \frac{1}{10} \\
&\quad - L_T^* - \sum_{t:\, L_t^* \neq L_{t-1}^*} (L_t^* - L_{t-1}^*)(\varepsilon_t/2 + \varepsilon_t^2) + L_T \\
&= -\frac{\ln n}{\varepsilon_{T+1}} + 4\ln n - \sum_{t:\, L_t^* \neq L_{t-1}^*,\, \varepsilon_t < 1/4} (L_t^* - L_{t-1}^*)\,\frac{\varepsilon_t}{4}\,(1+4\varepsilon_t) - \frac{1}{10} \\
&\quad - L_T^* - \sum_{t:\, L_t^* \neq L_{t-1}^*,\, \varepsilon_t < 1/4} (L_t^* - L_{t-1}^*)(\varepsilon_t/2 + \varepsilon_t^2) \\
&\quad - \sum_{t:\, L_t^* \neq L_{t-1}^*,\, \varepsilon_t = 1/4} (L_t^* - L_{t-1}^*)(\varepsilon_t/2 + \varepsilon_t^2) + L_T \\
&= -\frac{\ln n}{\varepsilon_{T+1}} + 4\ln n - \sum_{t:\, L_t^* \neq L_{t-1}^*,\, \varepsilon_t < 1/4} (L_t^* - L_{t-1}^*)\left(\frac{3\varepsilon_t}{4} + 2\varepsilon_t^2\right) - \frac{1}{10} \\
&\quad - L_T^* - \sum_{t:\, L_t^* \neq L_{t-1}^*,\, \varepsilon_t = 1/4} (L_t^* - L_{t-1}^*)(\varepsilon_t/2 + \varepsilon_t^2) + L_T . \qquad (11)
\end{aligned}
$$

We continue by bounding from above the three terms

$$\frac{\ln n}{\varepsilon_{T+1}}, \qquad (12)$$

$$\sum_{t:\, L_t^* \neq L_{t-1}^*,\, \varepsilon_t < 1/4} (L_t^* - L_{t-1}^*)\left(\frac{3\varepsilon_t}{4} + 2\varepsilon_t^2\right), \qquad (13)$$

$$\sum_{t:\, L_t^* \neq L_{t-1}^*,\, \varepsilon_t = 1/4} (L_t^* - L_{t-1}^*)(\varepsilon_t/2 + \varepsilon_t^2). \qquad (14)$$

To bound (12) we argue the following. If $L_T^* \leqslant 32 \ln n$ then $\varepsilon_{T+1} = \frac{1}{4}$. Hence in this case

$$\frac{\ln n}{\varepsilon_{T+1}} = 4\ln n.$$

On the other hand, if $L_T^* > 32 \ln n$ then

$$\frac{\ln n}{\varepsilon_{T+1}} = \frac{\ln n}{\sqrt{\frac{2\ln n}{L_T^*}}} = \sqrt{\frac{1}{2} L_T^* \ln n}.$$

To bound (13) observe that if $L_T^* \leqslant 32 \ln n$ holds then $\varepsilon_t = 1/4$ for $t = 1, \ldots, T$. Thus in this case (13) is trivially zero.

On the other hand, if $L_T^* > 32 \ln n$ we can write

$$\begin{aligned}
&\sum_{t:\, L_t^* \neq L_{t-1}^*,\, \varepsilon_t < 1/4} (L_t^* - L_{t-1}^*)\left(\frac{3\varepsilon_t}{4} + 2\varepsilon_t^2\right) \\
&\quad = \sum_{t:\, L_t^* \neq L_{t-1}^*,\, \varepsilon_t < 1/4} (L_t^* - L_{t-1}^*)\left(\frac{3}{4}\sqrt{\frac{2\ln n}{L_{t-1}^*}} + \frac{4\ln n}{L_{t-1}^*}\right) \\
&\quad \leqslant \sum_{t:\, L_t^* \neq L_{t-1}^*,\, \varepsilon_t < 1/4} (L_t^* - L_{t-1}^*)\left(\frac{3}{4}\sqrt{\frac{2\ln n}{L_t^* - 1}} + \frac{4\ln n}{L_t^* - 1}\right) \\
&\quad \leqslant \int_{32\ln n}^{L_T^*} \left(\frac{3}{4}\sqrt{\frac{2\ln n}{L-1}} + \frac{4\ln n}{L-1}\right) dL \\
&\quad < \int_{32(\ln n)-1}^{L_T^*} \left(\frac{3}{4}\sqrt{\frac{2\ln n}{L}} + \frac{4\ln n}{L}\right) dL \\
&\quad = \frac{3}{2}\sqrt{2L_T^* \ln n} - \frac{3}{2}\sqrt{2\ln n}\sqrt{32(\ln n) - 1} + 4(\ln n)\ln\frac{L_T^*}{32(\ln n)-1} \\
&\quad \leqslant \frac{3}{2}\sqrt{2L_T^* \ln n} + \frac{1}{5} - 12\ln n + 4(\ln n)\ln L_T^*,
\end{aligned}$$

where in the last inequality we used $\frac{3}{2}\sqrt{2x}\sqrt{32x-1} \geqslant 12x - \frac{1}{5}$, for $x \geqslant \ln 2$, and we dropped the denominator from the last logarithm.

Finally, a simple bound on (14) is

$$\begin{aligned}
\sum_{t:\, L_t^* \neq L_{t-1}^*,\, \varepsilon_t = 1/4} (L_t^* - L_{t-1}^*)(\varepsilon_t/2 + \varepsilon_t^2) &= \sum_{t:\, L_t^* \neq L_{t-1}^*,\, \varepsilon_t = 1/4} (L_t^* - L_{t-1}^*)(\tfrac{1}{8} + \tfrac{1}{16}) \\
&\leqslant (\tfrac{1}{8} + \tfrac{1}{16})\, 32 \ln n \\
&= 6 \ln n.
\end{aligned}$$

We plug these bounds back into (11) and solve for L_T. We obtain

$$L_T \leqslant L_T^* + 10 \ln n + \tfrac{1}{10},$$

in the case that $L_T^* \leqslant 32 \ln n$, and

$$L_T \leqslant L_T^* + 2\sqrt{2L_T^* \ln n} - 6\ln n + 4(\ln n)(\ln L_T^*) + \tfrac{3}{10},$$

in the case that $L_T^* > 32 \ln n$. Therefore in both cases

$$L_T \leqslant L_T^* + 2\sqrt{2L_T^* \ln n} + 4(\ln n)(\ln(1 + L_T^*)) + 10\ln n + \tfrac{3}{10},$$

thereby proving Theorem 2.1. ∎

The last theorem improves on results obtained by the doubling trick in [7]. In that paper the authors perform a sophisticated doubling trick analysis with a slightly different version of the Weighted Majority algorithm we consider here. They prove the bound $L_T \leqslant L_T^* + (3.3302 + o(1))\sqrt{L_T^* \ln n}$ as $T \to \infty$. Theorem 2.1 gives $L_T \leqslant L_T^* + (2.83 + o(1))\sqrt{L_T^* \ln n}$ as $T \to \infty$. It is probably possible to modify the constants in the expression for ε_t in Fig. 1 to allow in our bound a limited trade-off between the constant $2\sqrt{2}$ and the leading constants of the subsequent terms. We have not investigated this any further.

A bound similar to the one in Theorem 2.1 (but with slightly larger constants) can be obtained by a suitable choice of the parameter p in the self-confident p-norm algorithm of Section 3.1 (see the discussion in that section).

3. SELF-CONFIDENT ALGORITHMS

We now move on to analyze self-confident algorithms. Though we focus on self-confident p-norm algorithms, in Section 3.2 we also briefly discuss how our self-confident analysis could be applied to the algorithms in the Winnow/EG family.

3.1. Self-Confident p-Norm Algorithms

This section defines and analyzes our self-confident p-norm algorithms. We deal with two linear regression settings: (1) the square loss setting; (2) the absolute loss setting with binary labels (i.e., the binary classification problem where the algorithm makes randomized predictions). Our results for square loss are easily extended to more general regression frameworks, such as the generalized linear regression model of Helmbold *et al.* [18,23].

We first need to recall some preliminaries about the dual norms technology we will be using in this section. Given a vector $\mathbf{w} = (w_1, \ldots, w_n) \in \mathbb{R}^n$ and $p \geqslant 1$ we denote by $\|\mathbf{w}\|_p$ the p-norm of $\mathbf{w}$, i.e., $\|\mathbf{w}\|_p = (\sum_{i=1}^n |w_i|^p)^{1/p}$ (also, $\|\mathbf{w}\|_\infty = \lim_{p \to \infty} (\sum_{i=1}^n |w_i|^p)^{1/p} = \max_i |w_i|$). We say that p and q are *dual* if $\frac{1}{p} + \frac{1}{q} = 1$ holds. For example, the 1-norm is dual to the ∞-norm and the 2-norm is self-dual. For the rest of this section we assume that p and q are some pair of dual values with $p \geqslant 2$.

The p-norm algorithms are defined [14] in terms of the following bijective mapping $\mathbf{f}$ (a p indexing for $\mathbf{f}$ is understood): $\mathbf{f}$: $\mathbb{R}^n \to \mathbb{R}^n$, $\mathbf{f} = (f_1, \ldots, f_n)$, where

$$f_i(\mathbf{w}) = \frac{\operatorname{sign}(w_i)\,|w_i|^{q-1}}{\|\mathbf{w}\|_q^{q-2}}, \qquad \mathbf{w} = (w_1, \ldots, w_n) \in \mathbb{R}^n. \tag{15}$$

The function $\mathbf{f}$ is just the gradient of $P_{\mathbf{f}}(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_q^2$. The inverse $\mathbf{f}^{-1}$ of $\mathbf{f}$ is given by [14] $\mathbf{f}^{-1}$: $\mathbb{R}^n \to \mathbb{R}^n$, $\mathbf{f}^{-1} = (f_1^{-1}, \ldots, f_n^{-1})$, where

$$f_i^{-1}(\boldsymbol{\theta}) = \frac{\operatorname{sign}(\theta_i)\,|\theta_i|^{p-1}}{\|\boldsymbol{\theta}\|_p^{p-2}}, \qquad \boldsymbol{\theta} = (\theta_1, \ldots, \theta_n) \in \mathbb{R}^n,$$

i.e., $\mathbf{f}^{-1}$ is obtained from $\mathbf{f}$ by replacing q with p. Note that if $p = 2$ then $\mathbf{f}$ is the identity function.

Recall the definition of Bregman divergence given in Section 1.4. It is easy to check [14] (see also Gordon [16]) that the Bregman divergence $d_{\mathbf{f}}(\mathbf{u}, \mathbf{w})$ associated with the gradient mapping $\mathbf{f}$ given in (15) can be rewritten as

$$d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}) = \tfrac{1}{2}\|\mathbf{u}\|_q^2 + \tfrac{1}{2}\|\mathbf{w}\|_q^2 - \mathbf{u} \cdot \mathbf{f}(\mathbf{w}). \tag{16}$$

Also, note that the special case $p = 2$ yields $d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}) = \frac{1}{2}\|\mathbf{u} - \mathbf{w}\|_2^2$.

The following lemma is a Bregman divergence version of a classical result about projection operators. This lemma has also been used in the context of on-line learning by Herbster and Warmuth [20].

LEMMA 3.1 (Bregman [5], Censor and Lent [10]). *Let $\mathscr{W}$ be a closed and convex subset of $\mathbb{R}^n$, let $\mathbf{w} \in \mathbb{R}^n$, and let $\mathbf{w}' = \operatorname{argmin}_{\mathbf{u} \in \mathscr{W}} d_{\mathbf{f}}(\mathbf{u}, \mathbf{w})$ be the projection of $\mathbf{w}$ onto $\mathscr{W}$ w.r.t. the Bregman divergence $d_{\mathbf{f}}$ given in* (16). *Then for any $\mathbf{u} \in \mathscr{W}$ the following holds:*

$$d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}) \geqslant d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}') + d_{\mathbf{f}}(\mathbf{w}', \mathbf{w}) \geqslant d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}').$$

In this section we take our comparison class to be the convex set $\mathscr{W}_U = \{\mathbf{w} \in \mathbb{R}^n : \|\mathbf{w}\|_q \leqslant U\}$ and we will always be projecting onto $\mathscr{W}_U$. By a simple Kuhn–Tucker analysis it is not hard to verify that in such a case, $\mathbf{w}' = (\mathbf{w}U)/\|\mathbf{w}\|_q$ if $\|\mathbf{w}\|_q > U$ and $\mathbf{w}' = \mathbf{w}$, otherwise. (This specific projection occurs in the algorithms of Figs. 2 and 3.) We also have the following lemma.

LEMMA 3.2. *If $\mathbf{u}, \mathbf{w} \in \mathscr{W}_U$ then $d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}) \leqslant 2U^2$.*

Proof. The assertion follows from (16) and Hölder's inequality on the term $\mathbf{u} \cdot \mathbf{f}(\mathbf{w})$, for $\mathbf{u} \cdot \mathbf{f}(\mathbf{w}) \leqslant \|\mathbf{u}\|_q \|\mathbf{f}(\mathbf{w})\|_p = \|\mathbf{u}\|_q \|\mathbf{w}\|_q \leqslant U^2$, where the equality uses the fact that $\|\mathbf{f}(\mathbf{w})\|_p = \|\mathbf{w}\|_q$ (see Lemma 1, part 3 in [14]). ∎

The p-norm algorithms are a versatile on-line learning tool. It is noted in [14, 17] that by varying p these algorithms can behave in a radically different manner. Consider, for instance, the case of the square loss. Here $p = 2$ yields the Widrow–Hoff rule, while $p = 2 \ln n$ gives rise to an algorithm which is very similar to EG.

We now describe the self-confident p-norm algorithms for square loss and absolute loss. Both algorithms are assumed to know[4] bound U on the q-norm of the comparison vector. This assumption could be removed by applying the results we mention in Section 3.2.

We observe that, unlike previous on-line regression analyses [14, 18, 22, 23], our algorithms do not have any prior knowledge about the norm of the instances. The algorithms are given in Figs. 2 and 3. In Fig. 2 we denote by L_t the cumulative

[4] It is worth noticing that, in order to make a constant learning rate algorithm achieve bounds of the form of Theorems 3.1 and 3.2, both the norm of the best $\mathbf{u}$ and its loss seem to be a necessary prior knowledge [2, 22].

square loss of the algorithm up to trial t, i.e., $L_t = \sum_{i=1}^{t} l_i$, where $l_t = \frac{1}{2}(y_t - \hat{y}_t)^2$. In Fig. 3 we denote by L_t the cumulative absolute loss of the algorithm up to trial t, i.e., $L_t = \sum_{i=1}^{t} l_i$, where $l_t = \frac{1}{2}|y_t - \hat{y}_t|$. Also, in both figures (and throughout the rest of this section) we set

$$k_t = (p-1)\, X_t^2 U^2,$$

where

$$X_t = \max_{i:\, i \leqslant t,\, l_i > 0} \|\mathbf{x}_i\|_p.$$

The algorithms maintain an n-dimensional weight vector. They start from $\mathbf{w}_1 \in \mathscr{W}_U$, and in the generic trial t they are required to predict the unknown label y_t associated with the instance $\mathbf{x}_t$. The square loss algorithm predicts the label $y_t \in \mathbb{R}$ through the linear combination $\hat{y}_t = \mathbf{w}_t \cdot \mathbf{x}_t$, while the absolute loss algorithm

Initialization: Initial weight vector $\mathbf{w}_1 \in \mathscr{W}_U$;
For $t = 1, 2, \ldots, T$:

- Get instance $\mathbf{x}_t \in \mathbb{R}^n$;
- Predict with $\hat{y}_t = \mathbf{w}_t \cdot \mathbf{x}_t$;
- Get label $y_t \in \mathbb{R}$;
  Incur loss $l_t = \frac{1}{2}(y_t - \hat{y}_t)^2$;
- **If** $l_t > 0$ **then** update weights as follows.

Set

$$X_t = \max_{i:\, i \leqslant t,\, l_i > 0} \|\mathbf{x}_i\|_p,$$

$$k_t = (p-1)\, X_t^2 U^2,$$

$$c_t = \frac{\sqrt{k_t}}{\sqrt{k_t + L_t} - \sqrt{k_t}},$$

$$\eta_t = \frac{c_t}{1 + c_t} \frac{1}{(p-1)\, X_t^2}.$$

Let

$$\mathbf{w}_t^m = \mathbf{f}^{-1}(\mathbf{f}(\mathbf{w}_t) + \eta_t (y_t - \hat{y}_t)\, \mathbf{x}_t),$$

$$\mathbf{w}_{t+1} = \begin{cases} \mathbf{w}_t^m & \text{if} \quad \|\mathbf{w}_t^m\|_q \leqslant U, \\ \dfrac{\mathbf{w}_t^m U}{\|\mathbf{w}_t^m\|_q} & \text{otherwise.} \end{cases}$$

FIG. 2. The self-confident p-norm algorithm for square loss.

Initialization: Initial weight vector $\mathbf{w}_1 \in \mathcal{W}_U$.
For $t = 1, 2, \ldots, T$:

- Get instance $\mathbf{x}_t \in \mathbb{R}^n$;
- Let

$$X_t = \max_{i : i \leqslant t, l_i > 0} \|\mathbf{x}_i\|_p,$$

$$k_t = (p-1)\, X_t^2 U^2,$$

$$\alpha_t = \sqrt{\frac{k_t/4}{k_t/4 + L_{t-1} + 1}},$$

$$c_t = 1 - \alpha_t;$$

- Predict with $\hat{y}_t = \sigma_{c_t}(\mathbf{w}_t \cdot \mathbf{x}_t) = \begin{cases} 1 & \text{if} \quad \mathbf{w}_t \cdot \mathbf{x}_t \geqslant c_t, \\ \dfrac{\mathbf{w}_t \cdot \mathbf{x}_t}{c_t} & \text{if} \quad \mathbf{w}_t \cdot \mathbf{x}_t \in (-c_t, c_t), \\ -1 & \text{if} \quad \mathbf{w}_t \cdot \mathbf{x}_t \leqslant -c_t; \end{cases}$
- Get label $y_t \in \{-1, +1\}$;

  Incur loss $l_t = \frac{1}{2}|y_t - \hat{y}_t|$;
- **If** $l_t > 0$ **then** update weights as follows:

$$\mathbf{w}_t^m = \mathbf{f}^{-1}(\mathbf{f}(\mathbf{w}_t) + \eta_t y_t \mathbf{x}_t),$$

$$\mathbf{w}_{t+1} = \begin{cases} \mathbf{w}_t^m & \text{if} \quad \|\mathbf{w}_t^m\|_q \leqslant U, \\ \dfrac{\mathbf{w}_t^m U}{\|\mathbf{w}_t^m\|_q} & \text{otherwise,} \end{cases}$$

where $\eta_t = \dfrac{2\alpha_t}{(p-1)\, X_t^2}$.

FIG. 3. The self-confident p-norm algorithm for absolute loss and binary labels.

predicts the label $y_t \in \{-1, +1\}$ through the "clipped" linear combination $\hat{y}_t = \sigma_{c_t}(\mathbf{w}_t \cdot \mathbf{x}_t)$, as specified in Fig. 3. Note that the knot c_t of the clipping function σ_{c_t} tends to get close to 1 as the cumulative absolute loss L_t grows. When the label y_t is received, the algorithms incur a loss l_t. As we already said, this loss is the square loss for the algorithm in Fig. 2 and the absolute loss for the algorithm in Fig. 3. Finally, the algorithms update their weights as indicated. In both figures the update has two steps. The first step computes $\mathbf{w}_t^m$ by the conventional update of the p-norm algorithms, as in [14]. The second step computes $\mathbf{w}_{t+1}$ by projecting $\mathbf{w}_t^m$ onto $\mathcal{W}_U$ w.r.t. $d_{\mathbf{f}}$. Note that weights are not updated if $l_t = 0$.[5]

The analysis of the algorithms is summarized by the following results.

[5] The algorithms do not update also in the degenerate case that $\mathbf{x}_t = \mathbf{0}$.

THEOREM 3.1. *Let* $\mathscr{W}_U = \{\mathbf{w} \in \mathbb{R}^n : \|\mathbf{w}\|_q \leqslant U\}$ *and* $S = ((\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots)$, *where each* $(\mathbf{x}_t, y_t) \in \mathbb{R}^n \times \mathbb{R}$. *Then for any* $\mathbf{u} \in \mathscr{W}_U$ *the regression algorithm in Fig. 2, run on a prefix of* S *of arbitrary length* T, *achieves the following cumulative square loss bound*

$$L_T \leqslant L_{\mathbf{u},T} + 4k_T + 4\sqrt{k_T L_{\mathbf{u},T} + k_T^2},$$

where $L_{\mathbf{u},T} = \sum_{t=1}^{T} \frac{1}{2}(y_t - \mathbf{u} \cdot \mathbf{x}_t)^2$.

THEOREM 3.2. *Let* $\mathscr{W}_U = \{\mathbf{w} \in \mathbb{R}^n : \|\mathbf{w}\|_q \leqslant U\}$ *and* $S = ((\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots)$, *where each* $(\mathbf{x}_t, y_t) \in \mathbb{R}^n \times \{-1, +1\}$. *Then for any* $\mathbf{u} \in \mathscr{W}_U$ *the algorithm in Fig. 3, run on a prefix of* S *of arbitrary length* T, *achieves the following cumulative absolute loss bound*

$$\begin{aligned} L_T &\leqslant \tfrac{1}{2} D_{\mathbf{u},T} + \tfrac{3}{2} k_T - A + 1 + \sqrt{2k_T D_{\mathbf{u},T} + 4k_T - 4k_T A + 3k_T^2} \\ &= \tfrac{1}{2} D_{\mathbf{u},T} + O(k_T + \sqrt{k_T D_{\mathbf{u},T}}), \end{aligned}$$

where

$$A = \frac{1}{2} k_1 - \frac{(p-1) X_1^2 d_f(\mathbf{u}, \mathbf{w}_1)}{4}, \qquad 0 \leqslant A \leqslant \frac{1}{2} k_1,$$

$$D_{\mathbf{u},T} = \sum_{t=1, l_t > 0}^{T} D_{\mathbf{u},t},$$

$$D_{\mathbf{u},t} = \max\{0, 1 - y_t \mathbf{u} \cdot \mathbf{x}_t\}.$$

The bounds of Theorems 3.1 and 3.2 above have the same form of those proven for algorithms whose constant learning rate has been optimized in terms of the total loss of the comparison class (e.g., [2, 22, 23]). It should be clear that analogous bounds could also be obtained by applying the doubling trick to the corresponding p-norm algorithms with constant learning rate. Comparing these three groups of bounds—i.e., ours, those for the optimally tuned constant learning rate, and those for the doubling trick—requires a careful analysis of the leading constants that we did not carry out in this paper.

Note that the bound for the absolute loss algorithm is in terms of the *deviation* $D(\mathbf{u}; (\mathbf{x}_t, y_t))$ of a linear threshold classifier $\mathbf{u}$ with threshold 0 on example $(\mathbf{x}_t, y_t)$, defined as $D(\mathbf{u}; (\mathbf{x}_t, y_t)) = \max\{0, 1 - y_t \mathbf{u} \cdot \mathbf{x}_t\}$. The quantity $\frac{1}{2} D(\mathbf{u}; (\mathbf{x}_t, y_t))$ is related to a "loss" of $\mathbf{u}$ on example $(\mathbf{x}_t, y_t)$. For instance, if $\mathbf{u}$ is the ith unit vector and $\mathbf{x}_t \in [-1, +1]^n$, we obtain the finite expert framework considered in [7]. Here the tth prediction of the ith expert is $x_{t,i} = \mathbf{u} \cdot \mathbf{x}_t$, and $\frac{1}{2} D(\mathbf{u}; (\mathbf{x}_t, y_t)) = \frac{1}{2} |y_t - x_{t,i}| \in [0, 1]$ is exactly the absolute loss suffered by the ith expert in the ith trial. As another example, if $\mathbf{u}$ and $\mathbf{x}_t$ are n-dimensional $\{0, 1\}$-vectors then $\frac{1}{2} D(\mathbf{u}; (\mathbf{x}_t, y_t))$ corresponds to the so-called attribute error [27] of $\mathbf{u}$ on $(\mathbf{x}_t, y_t)$ This quantity counts the

minimal number of components of $\mathbf{x}_t$ that need to be flipped to make $\mathbf{u}$ classify $(\mathbf{x}_t, y_t)$ correctly.

The two results above have other interesting properties. The dual norms quantity k_T is a function of the norm p. This affects the dependence of the p-norm algorithm on the dimension n of the input space. Recall that $\|\mathbf{x}_t\|_p \leqslant X$ for all t implies that $k_T \leqslant (p-1)\, X^2 U^2$. For instance, if $p = 2 \ln n$ [14] then[6] $k_T < 2e \ln n U_1^2 X_\infty^2$, where U_1 is an upper bound on the 1-norm of $\mathbf{u}$ and X_∞ is an upper bound on the ∞-norm of the instances. In the expert case we have $U_1 = X_\infty = 1$. Thus $p = 2 \ln n$ yields $k_T < 2e \ln n$. This gives rise to a loss bound which is similar to the one we have proven in Section 2 for the adaptive Weighted Majority algorithm. (As a matter of fact, the constants in the bound of Theorem 3.2 are slightly larger.)

To prove Theorems 3.1 and 3.2 we need the following three technical lemmas. The first lemma is taken from [14], but it essentially follows from a combination of [17, 22]. The second lemma appears in various forms in [13, 14, 21, 23]. The third lemma is a simple technical tool for our self-confident analysis.

LEMMA 3.3. *Let* $\mathbf{u}, \mathbf{w}_t \in \mathbb{R}^n$, $\mathbf{x}_t \in \mathbb{R}^n$, *with* $\|\mathbf{x}_t\|_p \leqslant X_t$, *and set* $\mathbf{w}_t^m = \mathbf{f}^{-1}(\mathbf{f}(\mathbf{w}_t) + \eta_t (y_t - \hat{y}_t)\, \mathbf{x}_t)$, *with* $\eta_t = c_t / ((1+c_t)(p-1)\, X_t^2)$, $c_t \geqslant 0$, $X_t > 0$, $y_t \in \mathbb{R}$, *and* $\hat{y}_t = \mathbf{w}_t \cdot \mathbf{x}_t$. *Then the following inequality holds:*[7]

$$\frac{c_t}{1+c_t} \frac{1}{2} (y_t - \hat{y}_t)^2 - c_t \frac{1}{2} (y_t - \mathbf{u} \cdot \mathbf{x}_t)^2 \leqslant (p-1)\, X_t^2 (d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_t) - d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_t^m)).$$

LEMMA 3.4. *Let* $\mathbf{u}$, $\mathbf{w}_t$, $\mathbf{x}_t \in \mathbb{R}^n$, $\eta_t > 0$, *and set* $\mathbf{w}_t^m = \mathbf{f}^{-1}(\mathbf{f}(\mathbf{w}_t) + \eta_t y_t \mathbf{x}_t)$. *Then the following equality holds:*

$$y_t(\mathbf{u} \cdot \mathbf{x}_t - \mathbf{w}_t \cdot \mathbf{x}_t) = \frac{1}{\eta_t} (d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_t) - d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_t^m) + d_{\mathbf{f}}(\mathbf{w}_t, \mathbf{w}_t^m)).$$

LEMMA 3.5. *Let* $l_1, l_2, \ldots, l_T$ *and* δ *be non-negative real numbers. Then*

$$\sum_{t=1}^{T} \frac{l_t}{\sqrt{\delta + \sum_{i=1}^{t} l_i}} \leqslant 2 \left(\sqrt{\delta + \sum_{t=1}^{T} l_t} - \sqrt{\delta} \right),$$

where $0/\sqrt{0} = 0$.

Proof of Lemma 3.5. Let $l_0 = \delta$ and $L_t = \sum_{i=0}^{t} l_i$, $t = 0, \ldots, T$. In the inequality $\frac{1}{2} x \leqslant 1 - \sqrt{1-x}$, $x \leqslant 1$, set $x = l_t / L_t$ and then multiply both terms of the resulting inequality by $\sqrt{L_t}$. This yields

$$\frac{1}{2} \frac{l_t}{\sqrt{L_t}} \leqslant \sqrt{L_t} - \sqrt{L_{t-1}}.$$

The claim is then obtained by summing over $t = 1, \ldots, T$. ∎

[6] Here e is the base of natural logarithms.

[7] In the degenerate case that $X_t = 0$ we have $\mathbf{w}_t^m = \mathbf{w}_t$. In such a case η_t is not defined, but the inequality of the lemma trivially holds true for any $c_t \geqslant 0$.

Proof of Theorem 3.1. We note that Lemma 3.3 applies. We can write

$$\frac{c_t}{1+c_t}\, l_t - c_t l_{\mathbf{u},t} \leqslant (p-1)\, X_t^2 (d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_t) - d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_t^m)).$$

We divide by c_t and apply Lemma 3.1. This lemma allows us to lower bound $d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_t^m)$ by $d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_{t+1})$. We obtain

$$\frac{1}{1+c_t}\, l_t - l_{\mathbf{u},t} \leqslant \frac{(p-1)\, X_t^2}{c_t} (d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_t) - d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_{t+1})). \tag{17}$$

We need to upper bound the right-hand side of (17). For this purpose, we first claim that the following inequalities hold:

$$\frac{X_{t+1}^2}{c_{t+1}} \geqslant \frac{X_t^2}{c_t}, \qquad t = 1, \ldots, T. \tag{18}$$

To prove this claim, we define the function $g(x, L) = x(\sqrt{x^2 + L} - x)$. Computing its first derivatives, it is easy to see that $g(x, L)$ is nondecreasing in $x \geqslant 0$ for any $L \geqslant 0$ and nondecreasing in $L \geqslant 0$ for any $x \geqslant 0$. Now simple algebra gives

$$\frac{X_{t+1}^2}{c_{t+1}} = g\left(X_{t+1}, \frac{L_{t+1}}{(p-1)\, U^2}\right) \geqslant g\left(X_t, \frac{L_t}{(p-1)\, U^2}\right) = \frac{X_t^2}{c_t},$$

thereby proving (18).

Therefore the right-hand side of (17) can be bounded as

$$\begin{aligned}
&\frac{X_t^2}{c_t} (d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_t) - d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_{t+1})) \\
&\quad = \frac{X_t^2}{c_t} d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_t) - \frac{X_{t+1}^2}{c_{t+1}} d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_{t+1}) + d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_{t+1}) \left(\frac{X_{t+1}^2}{c_{t+1}} - \frac{X_t^2}{c_t}\right) \\
&\quad \leqslant \frac{X_t^2}{c_t} d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_t) - \frac{X_{t+1}^2}{c_{t+1}} d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_{t+1}) + 2U^2 \left(\frac{X_{t+1}^2}{c_{t+1}} - \frac{X_t^2}{c_t}\right),
\end{aligned}$$

where in the inequality we applied (18) and Lemma 3.2.

Plugging back into (17) and summing over $t = 1, \ldots, T$ gives

$$\begin{aligned}
&\sum_{t=1}^{T} \left(1 - \frac{c_t}{1+c_t}\right) l_t - L_{\mathbf{u},T} \\
&\quad \leqslant \frac{(p-1)\, X_1^2}{c_1} d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_1) - \frac{(p-1)\, X_{T+1}^2}{c_{T+1}} d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_{T+1}) + 2\left(\frac{k_{T+1}}{c_{T+1}} - \frac{k_1}{c_1}\right).
\end{aligned}$$

Since $d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_{T+1}) \geqslant 0$ and $d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_1) \leqslant 2U^2$ this implies

$$L_T - \sum_{t=1}^{T} \frac{c_t}{1+c_t} l_t \leqslant L_{\mathbf{u},T} + \frac{2k_T}{c_T}, \tag{19}$$

where we set $c_{T+1} = c_T$ and $X_{T+1} = X_T$ (so that (18) is not violated and $k_{T+1} = k_T$).

Since

$$\frac{c_t}{1+c_t} = \sqrt{\frac{k_t}{k_t + L_t}} \leqslant \sqrt{\frac{k_T}{k_T + L_t}},$$

we can apply Lemma 3.5 to the second term of the left-hand side of (19), along with the bound on $c_t/(1+c_t)$ just given. We also substitute the value $c_T = \sqrt{k_T}/(\sqrt{k_T + L_T} - \sqrt{k_T})$ into the right-hand side. This results in

$$L_T - 2\sqrt{k_T}\,(\sqrt{k_T + L_T} - \sqrt{k_T}) \leqslant L_{\mathbf{u},T} + 2\sqrt{k_T}\,(\sqrt{k_T + L_T} - \sqrt{k_T}).$$

Simplifying and rearranging gets

$$k_T + L_T \leqslant 4\sqrt{k_T}\,\sqrt{k_T + L_T} + L_{\mathbf{u},T} - 3k_T.$$

We solve for $L_T + k_T$ and simplify. The larger of the roots of the equation obtained by using $=$ instead of $\leqslant$ gives the desired bound. ▌

Proof of Theorem 3.2. We denote by $\mathcal{M}$ the set of trials where the algorithm incurs a nonzero loss. Let us focus on a single trial $t \in \mathcal{M}$ and let $\mathbf{w}_t^m$ be as in Fig. 3. We apply Lemma 3.4 and upper bound the last term $d_{\mathbf{f}}(\mathbf{w}_t, \mathbf{w}_t^m)$ as in [17, 14]: $d_{\mathbf{f}}(\mathbf{w}_t, \mathbf{w}_t^m) \leqslant (\eta_t^2/2)(p-1)\,X_t^2$. This yields

$$y_t(\mathbf{u} \cdot \mathbf{x}_t - \mathbf{w}_t \cdot \mathbf{x}_t) \leqslant \frac{1}{\eta_t}\left(d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_t) - d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_t^m) + \frac{\eta_t^2}{2}(p-1)\,X_t^2\right).$$

Next, we exploit the definition of $D(\mathbf{u}; (\mathbf{x}_t, y_t))$, from which it follows that $y_t \mathbf{u} \cdot \mathbf{x}_t \geqslant 1 - D(\mathbf{u}; (\mathbf{x}_t, y_t))$. Then, by Lemma 3.1, we lower bound $d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_t^m)$ through $d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_{t+1})$ and rearrange:

$$1 - y_t \mathbf{w}_t \cdot \mathbf{x}_t - \frac{\eta_t}{2}(p-1)\,X_t^2 \leqslant D_{\mathbf{u},t} + \frac{1}{\eta_t}(d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_t) - d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_{t+1})).$$

Next, we claim that for any trial t such that $l_t > 0$ the left-hand side of the last inequality is at least $2c_t l_t$, where $c_t = 1 - \alpha_t$. This would give us the one-trial loss bound

$$2c_t l_t \leqslant D_{\mathbf{u},t} + \frac{1}{\eta_t}(d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_t) - d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_{t+1})), \tag{20}$$

holding for any t such that $l_t > 0$ and any $\mathbf{u} \in \mathscr{W}_U$. We prove this claim by a case analysis over $y_t = -1, +1$. We work out the details only for $y_t = -1$, the other case being similar. If $y_t = -1$ then $l_t = \frac{1}{2}(1 + \hat{y}_t)$. Hence it suffices to prove that for any $r > -c_t$ we have

$$1 + r - \frac{\eta_t}{2}(p-1)\, X_t^2 \leqslant c_t(1 + \sigma_{c_t}(r)). \tag{21}$$

(The case $r \leqslant -c_t$ is not our concern, as this would get $l_t = 0$.) We split into two subcases: $r \geqslant c_t$ and $-c_t < r < c_t$. If $r \geqslant c_t$ then the right-hand side of (21) is $2c_t$, since $\sigma_{c_t}(r) = 1$. On the other hand, recalling the value of η_t in Fig. 3, the left-hand side of (21) is at least $1 + c_t - \alpha_t = 2c_t$. If $-c_t < r < c_t$ then $\sigma_{c_t}(r) = r/c_t$. It is easy to see that in this case both sides of (21) are equal to $c_t + r$. This concludes the proof of (20).

The right-hand side of (20) is upper bounded as

$$\begin{aligned}
&\frac{1}{\eta_t}\left(d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_t) - d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_{t+1})\right) \\
&\quad = \frac{(p-1)\, X_t^2}{2\alpha_t}\left(d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_t) - d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_{t+1})\right) \\
&\quad = \frac{p-1}{2}\left(\frac{X_t^2}{\alpha_t} d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_t) - \frac{X_{t+1}^2}{\alpha_{t+1}} d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_{t+1}) + d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_{t+1})\left(\frac{X_{t+1}^2}{\alpha_{t+1}} - \frac{X_t^2}{\alpha_t}\right)\right) \\
&\quad \leqslant \frac{p-1}{2}\left(\frac{X_t^2}{\alpha_t} d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_t) - \frac{X_{t+1}^2}{\alpha_{t+1}} d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_{t+1}) + 2U^2\left(\frac{X_{t+1}^2}{\alpha_{t+1}} - \frac{X_t^2}{\alpha_t}\right)\right),
\end{aligned}$$

where the last inequality follows from Lemma 3.2 and the fact (straightforward to check) that

$$\frac{X_{t+1}^2}{\alpha_{t+1}} \geqslant \frac{X_t^2}{\alpha_t}, \qquad t = 1, \ldots, T. \tag{22}$$

We plug this back into (20), sum over all $t \in \mathscr{M}$, drop the non-negative term involving $d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_{T+1})$, divide by 2, and rearrange. We obtain

$$\begin{aligned}
L_T - \sum_{t=1}^{T} \alpha_t l_t &\leqslant \frac{D_{\mathbf{u},T}}{2} + \frac{p-1}{4}\left(\frac{X_1^2}{\alpha_1} d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_1) + 2U^2\left(\frac{X_{T+1}^2}{\alpha_{T+1}} - \frac{X_1^2}{\alpha_1}\right)\right) \\
&= \frac{D_{\mathbf{u},T}}{2} + \frac{k_{T+1}}{2\alpha_{T+1}} - \frac{A}{\alpha_1},
\end{aligned} \tag{23}$$

where we have set

$$A = \frac{1}{2} k_1 - \frac{(p-1)\, X_1^2\, d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_1)}{4}.$$

We now handle (23) as follows. We set $X_{T+1} = X_T$, so that (22) is not violated and $\alpha_{T+1} = \sqrt{(k_T/4)/(k_T/4 + L_t + 1)}$. Next, since $A \geqslant 0$ (Lemma 3.2) and $\alpha_1 \leqslant 1$, we upper bound A/α_1 by A. Finally, we upper bound α_t in the left-most side of (23) by $\sqrt{(k_T/4)/(k_T/4 + L_t)}$ and then apply Lemma 3.5 with the bound on α_t just given. This leads to

$$L_T - \sqrt{k_T}\left(\sqrt{k_T/4 + L_T} - \frac{1}{2}\sqrt{k_T}\right) \leqslant \frac{D_{\mathbf{u},T}}{2} + \sqrt{k_T}\sqrt{k_T/4 + L_T + 1} - A.$$

By virtue of the inequality $\sqrt{1+x} \leqslant \sqrt{x} + 1/(2\sqrt{x})$, $x \geqslant 0$, the second term of the right-hand side is bounded from above by

$$\sqrt{k_T}\left(\sqrt{k_T/4 + L_T} + \frac{1}{2\sqrt{k_T/4 + L_T}}\right) \leqslant \sqrt{k_T}\sqrt{k_T/4 + L_T} + 1.$$

Simple algebra then gives

$$k_T/4 + L_T \leqslant \frac{D_{\mathbf{u},T}}{2} + 2\sqrt{k_T}\sqrt{k_T/4 + L_T} - k_T/4 + 1 - A.$$

We solve for $L_T + k_T/4$ and simplify. Again, we compute the larger of the roots of the equation obtained by using $=$ instead of $\leqslant$. This gives the bound of the theorem. ▮

3.2. Self-Confident Winnow/EG-like Algorithms

From the proofs of Theorems 3.1 and 3.2 the reader can see that our technique applies to a generic quasi-additive algorithm with mapping $\mathbf{f}$, as long as we can both find a constant upper bound[8] on the divergence terms $d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_{t+1})$, and show a one-trial loss bound of the form

$$(1-\alpha)\, l_t - l_{\mathbf{u},t} \leqslant \frac{c}{\alpha}\left(d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_t) - d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_{t+1})\right),$$

where l_t is the loss of the algorithm in trial t, $l_{\mathbf{u},t}$ is the loss of the generic off-line predictor $\mathbf{u}$ in trial t, $\alpha \in (0, 1)$ is a constant proportional to η, and $c > 0$.

Consider applying this proof technique to algorithms in the Winnow/EG family [22, 25, 26], such as Weighted Majority. The measure of progress typically associated with these algorithms is a relative entropy-like divergence. Proving the required one-trial loss bound is quite standard. Rather, the difficulty here stems from the fact that the relative entropy is hardly upper bounded, unless we introduce a lower bound constraint on the weights of the algorithm. Via this proof technique,

[8] In the case of the p-norm algorithms this is achieved through Lemma 3.2.

it is nevertheless possible to prove self-confident bounds for these algorithms. These bounds, however, have larger lower-order terms than those of the corresponding self-confident p-norm bounds in Theorems 3.1 and 3.2.

On the other hand, algorithms like Weighted Majority can be used in frameworks where the p-norm algorithms are harder to apply. As a relevant example, both the standard analysis [29, 34] and our self-confident analysis for Weighted Majority still hold when the dimension of the input space (the number of experts) is countably infinite. This is an interesting setting, since it can clearly formalize an on-line model selection problem with countably many models. (For instance, this can be used back in Section 3 to find a good value for the parameter U when we ignore a bound on the norm of the comparison vector.)

The question of obtaining sharp self-confident bounds for Winnow/EG-like algorithms remains an open problem.

4. CONCLUSIONS AND OPEN PROBLEMS

We have studied on-line learning algorithms with an incrementally adaptive learning rate. We have provided the first analysis of an adaptive Weighted Majority in the finite expert framework to date. This result compares favourably with previous bounds for Weighted Majority based on doubling strategies. For more general regression tasks we sketched a versatile and general technique to turn a constant learning rate algorithm into a variable learning rate algorithm, called self-confident algorithm. We focused on the analysis of self-confident p-norm algorithms proving regret bounds that are easily generalizable to a wide range of convex losses.

There are several directions in which this work could be extended. As we pointed out before, our analysis yields suboptimal results when applied to Winnow/EG-like algorithms, We would like to see if there is a self-confident analysis for such algorithms yielding bounds of the form

$$L_T \leqslant L_{\mathbf{u},T} + c_1 \sqrt{d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_1)\, X^2 L_{\mathbf{u},T}} + c_2\, d_{\mathbf{f}}(\mathbf{u}, \mathbf{w}_1)\, X^2,$$

where weight vectors and instances have countably many components, $d_{\mathbf{f}}$ is a relative entropy-like divergence, X is a bound on the ∞-norm of the instances, and c_1 and c_2 are positive constants. Also, it might be possible to extend our techniques to settings more general than the one studied here, e.g., the so-called shifting target setting [2, 19, 20].

We have recently applied on-line learning tools to the problem of approximating the maximal margin hyperplane for a set of linearly separable data (e.g., [12]). Part of the solution to this problem involves on-line tuning of parameters. As we have already pointed out, doubling strategies are not expected to work well in practice. Thus we have solved this tuning problem by a self-confident approach. We have tested our algorithms on a well-known handwritten digit recognition benchmark. The results we have obtained so far [15] are very encouraging.

APPENDIX A: PROOF OF LEMMA 2.1

Proof. Let L be the random variable taking values in the set $\{\ell_1, \ldots, \ell_n\}$ and whose distribution is given by

$$\Pr(L = \ell_i) = \frac{\alpha^{-\ell_i}}{Q},$$

where $Q = \sum_{j=1}^{n} \alpha^{-\ell_j}$. With this notation we can write

$$\frac{\sum_{i=1}^{n} \beta^{-\ell_i}}{\sum_{i=1}^{n} \alpha^{-\ell_i}} = \ln \mathbb{E}[(\beta/\alpha)^{-L}],$$

where $\mathbb{E}[\cdot]$ denotes the expectation. Since the logarithm is concave, Jensen's inequality gives

$$\ln \mathbb{E}[(\beta/\alpha)^{-L}] \geqslant -\ln(\beta/\alpha)\, \mathbb{E}[L]. \tag{A.1}$$

We continue by upper bounding the two factors $\ln(\beta/\alpha)$ and $\mathbb{E}[L]$. Using $\ln(1+x) \leqslant x$ and $1 < \alpha \leqslant \beta$ yields

$$\ln(\beta/\alpha) = \ln\left(1 + \frac{\beta-\alpha}{\alpha}\right) \leqslant \frac{\beta-\alpha}{\alpha} < \beta - \alpha.$$

As far as $\mathbb{E}[L]$ is concerned, we ague the following. Denote by $H(L)$ the entropy (in nats) of L. We have

$$H(L) = \sum_{i=1}^{n} \Pr(L = \ell_i) \ln \frac{1}{\Pr(L = \ell_i)} = \ln Q + \mathbb{E}[L] \ln \alpha. \tag{A.2}$$

Now, since L has a support of size n, $H(L) \leqslant \ln n$ holds. Moreover, the assumption $Q \geqslant 1$ is equivalent to $\ln Q \geqslant 0$. Thus from (A.2) we conclude that

$$\mathbb{E}[L] \leqslant \frac{\ln n}{\ln \alpha}.$$

Putting together as in (A.1) proves the lemma. ∎

ACKNOWLEDGMENTS

We are grateful to the anonymous reviewers, for providing a proof of Lemma 2.1 much more elegant than the one we had before. The authors were partially supported by ESPRIT Working Group EP 27150, Neural and Computational Learning II (NeuroColt II). Claudio Gentile also acknowledges partial support of MURST Cofi 99 ("Stochastic Processes with Spatial Structure").

REFERENCES

1. D. Angluin, Queries and concept learning, *Mach. Learning* **2** (1988), 319–342.
2. P. Auer and M. K. Warmuth, Tracking the best disjunction, *Mach. Learning* **32** (1998), 127–150.
3. K. Azoury and M. K. Warmuth, Relative loss bounds for on-line density estimation with the exponential family of distributions, *Mach. Learning*, to appear.
4. H. D. Block, The perceptron: A model for brain functioning, *Rev. Modern Phys.* **34** (1962), 123–135.
5. L. M. Bregman, The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming, *USSR Comput. Math. Phys.* **7** (1967), 200–217.
6. T. Bylander, The binary exponentiated gradient algorithm for learning linear functions, *in* "Proceedings, 8th Annu. Conf. on Comput. Learning Theory," pp. 184–192, Assoc. Comput. Mach., New York, 1997.
7. N. Cesa-Bianchi, Y. Freund, D. Haussler, D. Helmbold, R. Schapire, and M. K. Warmuth, How to use expert advice, *J. Assoc. Comput. Mach.* **44** (1997), 427–485.
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 Trans. Neural Networks* **7** (1996), 604–619.
9. N. Cesa-Bianchi, Analysis of two gradient-based algorithms for on-line regression, *J. Comput. System Sci.* **59** (1999), 329–411.
10. Y. Censor and A. Lent, An iterative row-action method for interval convex programming, *J. Optim. Theory Appl.* **34** (1981), 321–353.
11. Y. Censor and S. A. Zenios, "Parallel Optimization, Theory, Algorithms, and Applications," Oxford Univ. Press, New York, 1997.
12. C. Cortes and V. Vapnik, Support-vector networks, *Mach. Learning* **20** (1995), 273–297.
13. C. Gentile and M. K. Warmuth, Linear hinge loss and average margin, *in* "Proceedings, Advances in Neural Information Processing Systems 11" (M. S. Kearns, S. A. Solla, and D. A. Cohn, Eds.), pp. 225–231, MIT Press, Cambridge, MA, 1998.
14. C. Gentile and N. Littlestone, The robustness of the *p*-norm algorithms, *in* "Proceedings, 12th Annu. Conf. on Comput. Learning Theory, 1999," pp. 1–11.
15. C. Gentile, A new approximate maximal margin classification algorithm, *in* "Proceedings, Advances in Neural Information Processing Systems 13," to appear.
16. G. J. Gordon, Regret bounds for prediction problems, *in* "Proceedings, 12th Annu. Conf. on Comput. Learning Theory," pp. 29–40, Assoc. Comput. Mach., New York, 1999.
17. A. J. Grove, N. Littlestone, and D. Schuurmans, General convergence results for linear discriminant updates, *in* "Proceedings, 10th Annu. Conf. on Comput. Learning Theory," pp. 171–183, Assoc. Comput. Mach., New York, 1997.
18. D. Helmbold, J. Kivinen, and M. K. Warmuth, Worst-case loss bounds for sigmoided linear neurons, *IEEE Trans. Neural Networks* **10** (1999), 1291–1304.
19. M. Herbster and M. K. Warmuth, Tracking the best expert, *Mach. Learning* **32** (1998), 151–178.
20. M. Herbster and M. K. Warmuth, Tracking the best regressor, *in* "Proceedings, 11th Annu. Conf. on Comput. Learning Theory," pp. 24–31, Assoc. Comput. Mach., New York, 1998.
21. A. Jagota and M. K. Warmuth, Continuous and discrete time nonlinear gradient descent: Relative loss bounds and convergence, *in* "Electronic Proceedings of Fifth International Symposium on Artificial Intelligence and Mathematics," (R. Greiner and E. Boros, Eds.), 1998; rutcor.rutgers.edu/~amai.
22. J. Kivinen and M. K. Warmuth, Additive versus exponentiated gradient updates for linear prediction, *Inform. Comput.* **132** (1997), 1–64.
23. J. Kivinen and M. K. Warmuth, Relative loss bounds for multidimensional regression problems, *in* "Proceedings, Advances in Neural Information Processing Systems 10" (M. I. Jordan, M. J. Kearns, and S. A. Solla, Eds.) pp. 287–293, MIT Press, Cambridge, MA, 1997; *J. Mach. Learning*, to appear.

24. J. Kivinen and M. K. Warmuth, Averaging expert predictions, *in* "Proceedings, 4th European Conference on Comput. Learning Theory," Lecture Notes in Computer Science, Vol. 1572, pp. 153–167, Springer-Verlag, New York/Berlin, 1999.
25. N. Littlestone, Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm, *Mach. Learning* **2** (1988), 285–318.
26. N. Littlestone, "Mistake Bounds and Logarithmic Linear-Threshold Learning Algorithms," Ph.D. thesis, TR UCSC-CRL-89–11, University of California Santa Cruz, 1989.
27. N. Littlestone, Redundant noisy attributes, attribute errors, and linear threshold learning using Winnow, *in* "Proceedings, 4th Annu. Workshop on Comput. Learning Theory," pp. 147–156, Morgan Kaufmann, San Mateo, CA, 1991.
28. N. Littlestone, P. M. Long, and M. K. Warmuth, On-line learning of linear functions, *Comput. Complexity* **5** (1995), 1–23.
29. N. Littlestone and M. K. Warmuth, The weighted majority algorithm, *Inform. Comput.* **108** (1994), 212–261.
30. A. B. J. Novikov, On convergence proofs of perceptrons, *in* "Proceedings, Symposium on the Mathematical Theory of Automata," Vol. XII, pp. 615–622, 1962.
31. F. Rosenblatt, "Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms," Spartan Books, Washington, DC, 1962.
32. V. Vovk, Aggregating strategies, *in* "Proceedings, 3rd Annu. Workshop on Comput. Learning Theory," pp. 371–383, Morgan Kaufmann, San Mateo, CA, 1990.
33. V. Vovk, Competitive on-line linear regression, TR, Royal Holloway, University of London, CSD-TR-97–13, preliminary version, *in* "Proceedings, Advances in Neural Information Processing Systems 10" (M. I. Jordan, M. J. Kearns, and S. A. Solla, Eds.), pp. 364–370, MIT Press, Cambridge, MA, 1997.
34. V. Vovk, A game of prediction with expert advice, *J. Comput. System. Sci.* **56** (1998), 153–173.
35. V. Vovk, Derandomizing stochastic prediction strategies, *Mach. Learning* **35** (1999), 247–282.
36. B. Widrow and M. E. Hoff, Adaptive switching circuits, *in* "1960 IRE WESCON Conv. Recond, Part 4," pp. 96–104.
37. K. Yamanishi, A decision-theoretic extension of stochastic complexity and its applications to learning, *IEEE Inform. Theory* **44** (1998), 1424–1439.