Machine Learning, 45, 301–329, 2001

Relative Loss Bounds for Multidimensional Regression Problems*

J. KIVINEN† jyrki.kivinen@faceng.anu.edu.au
Department of Engineering, Australian National University, Canberra, ACT 0200, Australia

M. K. WARMUTH** manfred@cse.ucsc.edu
Department of Computer Science, University of California, Santa Cruz, CA 95064, USA

Editor: Peter Bartlett

Abstract. We study on-line generalized linear regression with multidimensional outputs, i.e., neural networks with multiple output nodes but no hidden nodes. We allow at the final layer transfer functions such as the softmax function that need to consider the linear activations to all the output neurons. The weight vectors used to produce the linear activations are represented indirectly by maintaining separate parameter vectors. We get the weight vector by applying a particular parameterization function to the parameter vector. Updating the parameter vectors upon seeing new examples is done additively, as in the usual gradient descent update. However, by using a nonlinear parameterization function between the parameter vectors and the weight vectors, we can make the resulting update of the weight vector quite different from a true gradient descent update. To analyse such updates, we define a notion of a matching loss function and apply it both to the transfer function and to the parameterization function. The loss function that matches the transfer function is used to measure the goodness of the predictions of the algorithm. The loss function that matches the parameterization function can be used both as a measure of divergence between models in motivating the update rule of the algorithm and as a measure of progress in analyzing its relative performance compared to an arbitrary fixed model. As a result, we have a unified treatment that generalizes earlier results for the gradient descent and exponentiated gradient algorithms to multidimensional outputs, including multiclass logistic regression.

Keywords: on-line prediction, relative loss bounds, generalized linear regression, Bregman divergences

1. Introduction

In a regression problem, we have a sequence of n-dimensional real valued *inputs* $\boldsymbol{x}_t \in \mathbf{R}^n$ and for each input $\boldsymbol{x}_t$ a k-dimensional real-valued *desired output* $\boldsymbol{y}_t \in \mathbf{R}^k$. Our goal is to find a mapping $\boldsymbol{f} : \mathbf{R}^n \to \mathbf{R}^k$ that at least approximately models the dependency between $\boldsymbol{x}_t$ and $\boldsymbol{y}_t$. In other words, if we write $\hat{\boldsymbol{y}}_t = \boldsymbol{f}(\boldsymbol{x}_t)$ for the *prediction* our model $\boldsymbol{f}$ makes given the input $\boldsymbol{x}_t$, we want $\hat{\boldsymbol{y}}_t \approx \boldsymbol{y}_t$ for all t. The most basic class of functions $\boldsymbol{f}$ to consider is the

*A preliminary version appeared in Advances in Neural Information Processing Systems 10, pp. 287–293, MIT Press, Cambridge, MA, 1998.

†Supported by University of Helsinki, the Academy of Finland, and the European Commission under Working Group NeuroCOLT2, No. EP27150.
**Supported by NSF grants CCR 9700201 and 9821087.

class of linear functions, which in the case of one-dimensional outputs ($k = 1$) means our problem is to find a suitable weight vector $\boldsymbol{\omega} \in \mathbf{R}^n$ and then predict with $\hat{y}_t = \boldsymbol{\omega} \cdot \boldsymbol{x}_t$. In the case of multidimensional outputs, we actually have a matrix $\Omega \in \mathbf{R}^{k \times n}$ of parameters and $\hat{\boldsymbol{y}}_t = \Omega \boldsymbol{x}_t$. The goodness of the prediction $\hat{\boldsymbol{y}}$ is quantitatively measured in terms of a *loss function*. The *square loss*, given by $\sum_{t,j} (y_{t,j} - \hat{y}_{t,j})^2/2$, is a popular choice that is suitable in many situations.

In *generalized linear regression* (McCullagh & Nelder, 1989) we fix a *transfer function* ϕ and apply it on top of a linear model. Thus, with one-dimensional outputs we would have $\hat{y}_t = \phi(\boldsymbol{\omega} \cdot \boldsymbol{x}_t)$. Here ϕ is usually a continuous increasing function from $\mathbf{R}$ to $\mathbf{R}$, such as the logistic function that maps z to $1/(1 + e^{-z})$. What we call here transfer function is the inverse of what is usually called the link function (McCullagh & Nelder, 1989; Warmuth & Jagota, 1997). Considering transfer functions instead of link functions simplifies technicalities with multidimensional outputs, when we need to consider noninvertible transfer functions.

It is still possible to use the square loss, but this can lead to problems. In particular, when we apply the logistic transfer function and try to find a weight vector $\boldsymbol{\omega}$ that minimizes the square loss over ℓn examples $(\boldsymbol{x}_t, y_t)$, we may have up to ℓ^n local minima (Auer, Herbster, & Warmuth, 1995; Budinich, 1993). Hence, some other choice of loss function might be more useful. With one-dimensional outputs and a strictly increasing continuous ϕ, it is possible to define a *matching loss function* L_ϕ which has the property that the empirical loss

$$\sum_{t=1}^{\ell} L_\phi(y_t, \phi(\boldsymbol{\omega} \cdot \boldsymbol{x}_t)) \tag{1}$$

is a convex function of the weight vector $\boldsymbol{\omega}$ and thus, in particular, has one single minimum (Auer, Herbster, & Warmuth, 1995; Helmbold, Kivinen, & Warmuth, 1999). For example, the matching loss function for the logistic transfer function is the *relative entropy* (a generalization of the logarithmic loss for continuous-valued outcomes).

The main theme of this paper is the generalization of the notion of the matching loss function for multidimensional outputs. With k-dimensional outputs, the n-dimensional input vector $\boldsymbol{x}_t$ is first multiplied by a $k \times n$ *weight matrix* Ω, which gives us the k-dimensional *linear activation* $\hat{\boldsymbol{a}}_t = \Omega \boldsymbol{x}_t$. We then pass the linear activation through a transfer function ϕ, which now is a mapping from $\mathbf{R}^k$ to $\mathbf{R}^k$. This gives the prediction $\hat{\boldsymbol{y}}_t = \phi(\hat{\boldsymbol{a}}_t)$ (see figure 1).

We now wish to define the matching loss for the multidimensional transfer function $\phi\colon \mathbf{R}^k \to \mathbf{R}^k$. Recall that in the one-dimensional case, we assumed the transfer function to be differentiable and strictly increasing. In the multidimensional case, we assume that ϕ is continuously differentiable, has a potential function P_ϕ (i.e. we can write $\phi = \nabla P_\phi$), and this potential function P_ϕ is strictly convex. Notice that in the one-dimensional case, the potential function (i.e., integral function) of any strictly increasing function ϕ is strictly convex, so this naturally generalizes our assumptions from the one-dimensional case. We shall later consider the case in which P_ϕ is convex but not strictly convex, which requires some additional technicalities.

Since ϕ is the gradient of a strictly convex function, it is one-to-one. Hence, for any desired output $\boldsymbol{y}$, there is a unique desired linear activation $\boldsymbol{a}$ such that $\boldsymbol{y} = \phi(\boldsymbol{a})$. Similarly,

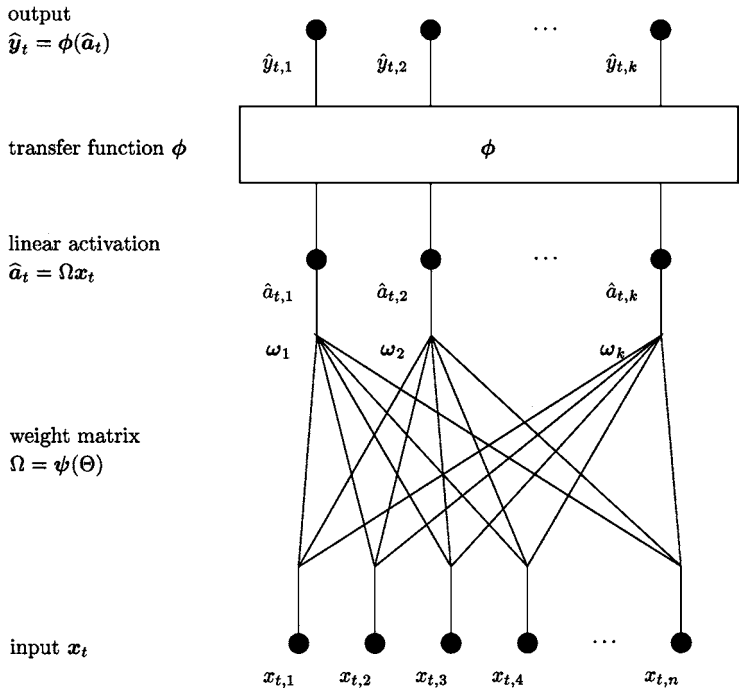

Figure 1. Obtaining the prediction $\hat{\boldsymbol{y}}_t$ from the input $\boldsymbol{x}_t$ via the weights Ω and transfer function ϕ.

given a prediction $\hat{\boldsymbol{y}}$, we can uniquely determine the linear activation $\hat{\boldsymbol{a}}$ such that $\hat{\boldsymbol{y}} = \phi(\hat{\boldsymbol{a}})$. We now define the matching loss L_ϕ for ϕ by

$$L_\phi(\boldsymbol{y}, \hat{\boldsymbol{y}}) = P_\phi(\hat{\boldsymbol{a}}) - P_\phi(\boldsymbol{a}) - (\hat{\boldsymbol{a}} - \boldsymbol{a}) \cdot \phi(\boldsymbol{a}). \tag{2}$$

These matching loss functions are also known as Bregman divergences (Bregman, 1967). From (2) we directly get the basic property

$$\nabla_{\hat{a}} L_\phi(\phi(\boldsymbol{a}), \phi(\hat{\boldsymbol{a}})) = \phi(\hat{\boldsymbol{a}}) - \phi(\boldsymbol{a}). \tag{3}$$

Our assumptions about ϕ imply that L_ϕ is a reasonable loss function in the sense that $L_\phi(\boldsymbol{y}, \boldsymbol{y}) = 0$ for $\boldsymbol{y}$ in the range of ϕ, and $L_\phi(\boldsymbol{y}, \hat{\boldsymbol{y}}) > 0$ when $\hat{\boldsymbol{y}} \neq \boldsymbol{y}$. However, L_ϕ is in general not symmetrical, and does not satisfy the triangle inequality. Therefore, we prefer the term "Bregman divergence" over "Bregman distance" that is also commonly used in literature.

The simplest example is the matching loss for the identity transfer function. For the identity function $\phi(\boldsymbol{a}) = \boldsymbol{a}$, the potential function is given by $P_\phi(\boldsymbol{a}) = \|\boldsymbol{a}\|_2^2/2$, and the matching loss is the squared Euclidean distance $L_\phi(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \|\boldsymbol{y} - \hat{\boldsymbol{y}}\|_2^2/2$. In this case going to multiple output dimensions does not really add anything new to the problem, as the linear

regression problem with k-dimensional outputs decomposes into k independent problems with one-dimensional output.

The softmax function $\boldsymbol{\sigma}$, given by

$$\sigma_i(\boldsymbol{a}) = \frac{\exp(a_i)}{\sum_{j=1}^{k} \exp(a_j)}, \tag{4}$$

gives a more interesting example of a common transfer function with matching loss. Notice that the range of $\boldsymbol{\sigma}$ is $\{\boldsymbol{y} \in \mathbf{R}^k \mid \sum_{j=1}^{k} y_j = 1 \text{ and } y_j > 0 \text{ for all } j\}$. The softmax function generalizes the logistic function to the multidimensional case. Now each variable a_j also affects outputs $\sigma_i(\boldsymbol{a})$ with $i \neq j$ through the normalization factor in the denominator. The softmax function has a convex potential function given by $P_{\boldsymbol{\sigma}}(\boldsymbol{a}) = \ln(\sum_{j=1}^{k} \exp(a_j))$. However, $P_{\boldsymbol{\sigma}}$ is not strictly convex, and its gradient $\boldsymbol{\sigma}$ is not one-to-one. However, with $\boldsymbol{\phi} = \boldsymbol{\sigma}$ it turns out that for given $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ in the range of $\boldsymbol{\phi}$, the right-had side of (2) has the same value for any choices of $\boldsymbol{a}$ and $\hat{\boldsymbol{a}}$ such that $\boldsymbol{\phi}(\boldsymbol{a}) = \boldsymbol{y}$ and $\boldsymbol{\phi}(\hat{\boldsymbol{a}}) = \hat{\boldsymbol{y}}$. Hence, (2) defines a unique matching loss also for the softmax function. This matching loss is the well-known relative entropy

$$L_{\boldsymbol{\sigma}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \sum_{j=1}^{k} y_j \ln \frac{y_j}{\hat{y}_j}. \tag{5}$$

In general, the loss $L_{\boldsymbol{\phi}}(\boldsymbol{\phi}(\boldsymbol{a}), \boldsymbol{\phi}(\hat{\boldsymbol{a}}))$ can be interpreted as the relative entropy between two distributions from the exponential family with cumulant function $P_{\boldsymbol{\phi}}$ and natural parameters $\boldsymbol{a}$ and $\hat{\boldsymbol{a}}$, respectively (Amari, 1985). Similar loss functions have been also used in other work on generalized linear models (McCullagh & Nelder, 1989; Fahrmeir & Tutz, 1991). These statistical interpretations of the loss function and relative loss bounds are discussed by Azoury and Warmuth (1999); here we need only some very basic properties of the matching loss.

Generalizing the notion of matching loss from one-dimensional to multidimensional outputs allows us to similarly generalize loss bounds shown earlier for one-dimensional generalized linear regression (Helmbold, Kivinen, & Warmuth, 1999). Even more interestingly, it turns out that matching loss functions can be applied not only as a measure of loss among predictions, but also as measures of divergence among parameter vectors of learning algorithms. This provides a unifying framework for earlier research in which the squared Euclidean distance (Cesa-Bianchi, Long, & Warmuth, 1996) and relative entropy (Kivinen & Warmuth, 1997) have been considered separately as divergence measures applicable to different algorithms. To see how matching loss functions are applied as divergence measures, we first need to understand the basic setting of *on-line learning*.

We start with the most basic case, linear regression with one-dimensional outputs. In the most typical learning setting, often called *batch learning*, the learning algorithm is given as input the whole set of examples $(\boldsymbol{x}_t, y_t)$, $t = 1, \ldots, \ell$, and then required to find a weight vector $\boldsymbol{\omega}$ such that the total squared loss $\sum_t (y_t - \boldsymbol{\omega} \cdot \boldsymbol{x}_t)^2/2$ is minimized. This particular batch learning problem is of course well studied, but even in this simple case the situation immediately becomes more interesting when we move from batch to on-line learning. An

on-line learning algorithm starts with some fixed initial weight vector $\boldsymbol{\omega}_1$, say $\boldsymbol{\omega}_1 = \mathbf{0}$. Then, at each time step $t = 1, \ldots, \ell$, it gets as input the single example $(\boldsymbol{x}_t, y_t)$ and uses some simple *update rule* to produce a new weight vector $\boldsymbol{\omega}_{t+1}$ from its old weight vector $\boldsymbol{\omega}_t$ and the recent example $(\boldsymbol{x}_t, y_t)$. On-line learning algorithms are useful if the training examples only become available one at a time and the algorithm needs to be able to improve its weights to take more examples into account, or sometimes simply because considering all the examples at once causes computational problems.

The most familiar update rule for linear regression is the Least Mean Squares (LMS), or Widrow-Hoff, rule

$$\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t - \eta(\boldsymbol{\omega}_t \cdot \boldsymbol{x}_t - y_t)\,\boldsymbol{x}_t \tag{6}$$

where $\eta > 0$ is a learning rate parameter. Note that (6) can be interpreted as a gradient descent step for $\boldsymbol{\omega}$ in minimizing the loss $(y_t - \boldsymbol{\omega} \cdot \boldsymbol{x}_t)^2/2$. This is easily generalized to a nonlinear transfer function ϕ. We see from (3) that the gradient of the matching loss $L_\phi(y, \phi(\boldsymbol{\omega} \cdot \boldsymbol{x}))$ with respect to the weight vector $\boldsymbol{\omega}$ is given by $(\hat{y} - y)\boldsymbol{x}$ where $\hat{y} = \phi(\boldsymbol{\omega} \cdot \boldsymbol{x})$. Hence, the gradient descent update step is given by

$$\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t - \eta(\hat{y}_t - y_t)\,\boldsymbol{x}_t. \tag{7}$$

The update (7) implies a certain linearity: $\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_1 + \sum_{j=1}^{t} c_j \boldsymbol{x}_j$ for some scalar coefficients c_j. In other words, the total change $\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}_1$ in the weight vector is always in the span of the input vectors already seen. A simple corollary to this is that if we use the zero start vector $\boldsymbol{\omega}_1 = \mathbf{0}$, then rotating the input vectors (i.e., replacing each $\boldsymbol{x}_t$ by $A\boldsymbol{x}_t$ where A is a fixed orthonormal matrix) leaves the dot products $\boldsymbol{\omega}_t \cdot \boldsymbol{x}_t$, and hence the actual outputs $\phi(\boldsymbol{\omega}_t \cdot \boldsymbol{x}_t)$, unchanged. Note that the gradient descent algorithm has this property regardless of what loss function is used.

This observation has some interesting implications when the dimensionality n of the input space is very large. Such situations arise for instance with support vector machines (Boser, Guyon, & Vapnik, 1992), when one maps the inputs into a high-dimensional feature space via a nonlinear mapping, thus hoping to reduce a nonlinear problem into a linear one. It is well known that for learning linear functions in n dimensions one needs, in general, at least n examples. For large n this may be impractical, so we need to make some additional restrictions to guarantee learning from a reasonable amount of data. In the context of support vector machines, it has been noted that the actual mappings used to transform the inputs into the high-dimensional feature space can cause the effective dimensionality of the resulting linear learning problem to be significantly lower than the actual dimensionality of the feature space (Guo et al., 1999). Another possible explanation for the fact that learning is possible in quite high-dimensional spaces could be that often a quite large proportion of the input variables are *irrelevant*, i.e., there is a weight vector $\boldsymbol{\omega}$ with most components zero that achieves a small empirical loss. However, the property of having most components zero is obviously not maintained if the input vectors are rotated. Therefore, the preceding argument shows that gradient descent cannot take advantage of this situation. (See Kivinen, Warmuth, & Auer, 1997 for more discussion.) This leads us to consider alternatives in which the weight vector $\boldsymbol{\omega}_t$ has a nonlinear dependence on the input vectors.

As an alternative to the gradient descent update (7), some recent work has considered the *exponentiated gradient* (EG) update (Kivinen & Warmuth, 1997)

$$\omega_{t+1,i} = \omega_{t,i} \exp(-\eta(\hat{y}_t - y_t)x_{t,i})/Z_t \tag{8}$$

where $Z_t = \sum_{i=1}^{n} \omega_{t,i} \exp(-\eta(\hat{y}_t - y_t)x_{t,i})$ is a normalization factor. (Standard reductions can be used to extend this to weights with arbitrary sign and to unnormalized weights (Kivinen & Warmuth, 1997).) Theoretical analysis and experiments on artificial data (Kivinen & Warmuth, 1997; Helmbold, Kivinen, & Warmuth, 1999; Kivinen, Warmuth, & Auer, 1997) suggest that the exponentiated gradient update indeed works well when there is a large number of irrelevant input variables.

In the present paper, we show how the gradient descent and exponentiated gradient algorithms can be seen as special cases of the *general additive algorithm*. To do this, we first introduce a new n-dimensional *parameter vector* $\boldsymbol{\theta}_t$ which we update according to the rule

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta(\hat{y}_t - y_t)\,\boldsymbol{x}_t. \tag{9}$$

Then, we use a *parameterization function* $\psi : \mathbf{R}^n \to \mathbf{R}^n$ to give the actual weight vector as $\boldsymbol{\omega}_t = \psi(\boldsymbol{\theta}_t)$. We now see that the gradient descent update (7) results from (9) simply with the identity function as the parameterization function. The exponentiated gradient update (8) is obtained by using $\psi = \boldsymbol{\sigma}$ where $\boldsymbol{\sigma}$ is the softmax function given in (4).

The general additive algorithm, described above for one-dimensional outputs, generalizes naturally to the case of $k > 1$ output dimensions. Thus, one of the results of this paper is to see how the results of Helmbold, Kivinen, and Warmuth (1999) for matching loss functions in the one-dimensional case generalize to the multidimensional case. With multidimensional outputs, we have k separate parameter vectors $\boldsymbol{\theta}_{t,j}$ $(j = 1, \ldots, k)$ at time t, which constitute the rows of a parameter matrix $\Theta_t \in \mathbf{R}^{k \times n}$. The weight matrix $\Omega_t \in \mathbf{R}^{k \times n}$ is obtained by applying the parameterization function $\psi : \mathbf{R}^n \to \mathbf{R}^n$ to each row of Θ_t separately (see figure 2). We write this as $\Omega_t = \psi(\Theta_t)$. The prediction $\hat{\boldsymbol{y}}_t \in \mathbf{R}^k$ is given by $\hat{\boldsymbol{y}}_t = \phi(\Omega_t \boldsymbol{x}_t)$, where $\phi : \mathbf{R}^k \to \mathbf{R}^k$ is the transfer function (see figure 1). For $j = 1, \ldots, k$, the jth row

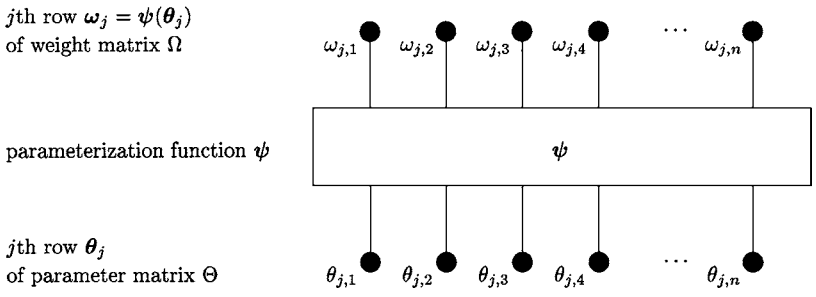

Figure 2. Obtaining the weights Ω from the parameters Θ via the parameterization function ψ.

$\boldsymbol{\theta}_{t,j}$ of the parameter matrix is updated according to the rule

$$\boldsymbol{\theta}_{t+1,j} = \boldsymbol{\theta}_{t,j} - \eta(\hat{y}_{t,j} - y_{t,j})\,\boldsymbol{x}_t \tag{10}$$

where $\hat{y}_{t,j}$ and $y_{t,j}$ are the jth component of the prediction of the algorithm and the desired output.

We can now finally see how matching loss functions relate to the general additive update (9), or more generally (10). Thus, consider substituting $\phi = \psi$ in the formula (2) for the matching loss. Our preceding examples of parameterization functions ψ, the identity function and softmax, were such that the matching loss is well-defined, so in these cases this gives a function L_ψ such that $L_\psi(\boldsymbol{\omega}, \boldsymbol{\omega}) = 0$ and $L_\psi(\boldsymbol{\omega}, \boldsymbol{\omega}') > 0$ for $\boldsymbol{\omega} \neq \boldsymbol{\omega}'$. We can therefore interpret L_ψ as a measure of divergence between weight vectors. It turns out that this measure of divergence is particularly suitable for analysing the general additive update (9). In the analysis, the measure of divergence L_ψ is used for two different purposes. First, it turns out that the update (9) can be motivated as an approximate solution to a minimization problem. In this minimization problem, the basic idea is to choose $\boldsymbol{\omega}_{t+1}$ so as to minimize the loss if the tth example $(\boldsymbol{x}_t, \boldsymbol{y}_t)$ were to occur again at trial $t + 1$. However, we add $L_\psi(\boldsymbol{\omega}_{t+1}, \boldsymbol{\omega}_t)$ to the minimization problem as a regularizer to avoid too drastic updates based on a single example. The second use for L_ψ is as a measure of progress in proving relative loss bounds for the general additive algorithm in an on-line prediction framework. In these bounds, and their proofs, we use the matching loss to measure the progress of the weight vector in each update. The techniques used in the motivation and loss bound proofs are generalizations of the techniques used earlier (Kivinen & Warmuth, 1997; Helmbold, Kivinen, & Warmuth, 1999) in the special cases of the gradient descent algorithm (with the squared Euclidean distance as the divergence function) and the exponentiated gradient algorithm (with the relative entropy as the divergence function). However, the previous work treated only one-dimensional outputs.

In work parallel to this, the general additive update (9) in the context of linear classification, i.e., with a thresholded transfer function, has recently been developed and analyzed by Grove, Littlestone, and Schuurmans (1997) with methods and results very similar to ours; see also Gentile and Littlestone (1999). Gentile and Warmuth (1999) have shown how the notion of matching loss can be generalized to thresholded transfer functions. This relates the Perceptron algorithm to gradient descent and Winnow to the exponentiated gradient algorithm. Cesa-Bianchi (1999) has used somewhat different methods to obtain bounds also in cases in which the loss function does not match the transfer function. Warmuth and Jagota (1997) view (9) as an Euler discretization of a system of partial differential equations and investigate the behavior of the relative loss bounds as the discretization parameter approaches zero.

In Section 2 we review some basic properties of matching loss functions and give some examples. Section 3 discusses the general additive algorithm for on-line prediction in more detail and gives the motivation of the update (10) in terms of a minimization problem. The general nature of relative on-line loss bounds, and the particular results we have for the general additive algorithm, are explained in Section 4.

2. Matching loss functions

Our notion of matching loss is essentially what is known as a Bregman divergence (Bregman, 1967). In statistics, similar loss functions have been used by Amari (1985) and others (McCullagh & Nelder, 1989; Fahrmeir & Tutz, 1991). In this paper we do not make use of the probabilistic interpretations. For the sake of completeness, we derive directly the few basic properties we need. For a discussion of our line of research in the context of statistics see Azoury and Warmuth (1999).

Consider first the one-dimensional case, with a differentiable transfer function $\phi : \mathbf{R} \to \mathbf{R}$ such that $\phi'(a) > 0$ for all a. For this case, the matching loss (Auer, Herbster, & Warmuth, 1995) is defined by

$$L_\phi(y, \hat{y}) = \int_{\phi^{-1}(y)}^{\phi^{-1}(\hat{y})} (\phi(r) - y)\, dr, \tag{11}$$

where we have used the fact that ϕ is invertible. Figure 3 gives a graphical representation.

Since ϕ is continuously differentiable with strictly positive derivative, it has an inverse ϕ^{-1} that also has a strictly positive derivative. We can then use (11) to define another loss function $L_{\phi^{-1}}$. It is instructive here to briefly consider the connection between L_ϕ and $L_{\phi^{-1}}$.

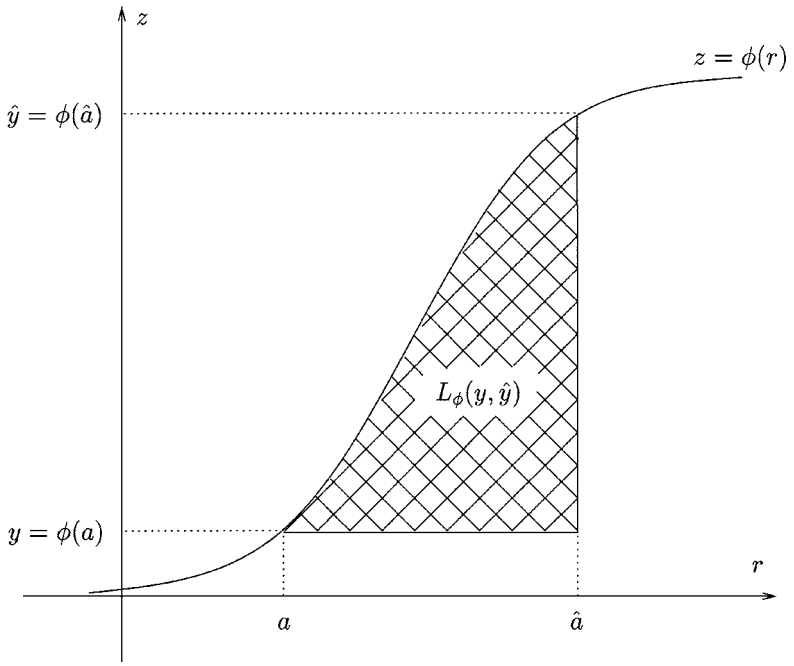

Figure 3. The matching loss given in (11) for one-dimensional output (with $a = \phi^{-1}(y)$ and $\hat{a} = \phi^{-1}(\hat{y})$).

Starting from the definition (11), we get

$$L_\phi(y, \hat{y}) = \int_{\phi^{-1}(y)}^{\phi^{-1}(\hat{y})} (\phi(r) - y)\, dr$$

$$= \int_y^{\hat{y}} (z - y)\, (\phi^{-1})'(z)\, dz \tag{12}$$

$$= \Big|_y^{\hat{y}} (z - y)\phi^{-1}(z) - \int_y^{\hat{y}} \phi^{-1}(z)\, dz \tag{13}$$

$$= \int_{\hat{y}}^{y} (\phi^{-1}(z) - \phi^{-1}(\hat{y}))\, dz, \tag{14}$$

where (12) follows from the variable substitution $r = \phi^{-1}(z)$ and (13) from integration by parts. Comparing this with (11), we notice that the right-hand side of (14) is actually the same as $L_{\phi^{-1}}(\phi^{-1}(\hat{y}), \phi^{-1}(y))$. We have obtained the duality relation

$$L_\phi(y, \hat{y}) = L_{\phi^{-1}}(\hat{a}, a) \tag{15}$$

where $y = \phi(a)$ and $\hat{y} = \phi(\hat{a})$. Notice how the order of the arguments has changed. To visualize (15), compare figure 3 with figure 4, which gives the graphical representation of $L_{\phi^{-1}}(\hat{a}, a)$. Figure 4 is obtained by reflecting figure 3 with respect to the line $z = r$ in the rz plane.

The relation (15) is interesting for us, because it shows that we can measure loss either in terms of the desired and actual outputs y and $\hat{y}$ or in terms of the desired and actual linear activations a and $\hat{a}$, and in both cases we can use the matching loss function. In the multidimensional case it will turn out that we also want to allow noninvertible transfer functions. Therefore, we define

$$\Delta_\phi(\hat{a}, a) = \int_a^{\hat{a}} (\phi(r) - \phi(a))\, dr.$$

The idea is that even if ϕ were not invertible, we could use Δ_ϕ to measure the loss in terms of the linear activations. The notation has been chosen so that if ϕ^{-1} does exist, then Δ_ϕ and $L_{\phi^{-1}}$ are the same function. This follows from (15).

To see the simplest way of generalizing (11) into multiple dimensions we write it as $L_\phi(\phi(a), \phi(\hat{a})) = P_\phi(\hat{a}) - P_\phi(a) - (\hat{a} - a)\phi(a)$ where P_ϕ is an integral function of ϕ. As $\phi'(a) > 0$ for all a, the integral function P_ϕ is strictly convex. Let now ϕ be a function from $\mathbf{R}^k$ to $\mathbf{R}^k$, and assume that it has a potential function P_ϕ (i.e., $\phi = \nabla P_\phi$ for some $P_\phi : \mathbf{R}^k \to \mathbf{R}$) and that this potential function P_ϕ is convex. Analogous with the one-dimensional case, L_ϕ is now said to be the matching loss function for ϕ if it satisfies

$$L_\phi(\phi(\boldsymbol{a}), \phi(\hat{\boldsymbol{a}})) = P_\phi(\hat{\boldsymbol{a}}) - P_\phi(\boldsymbol{a}) - (\hat{\boldsymbol{a}} - \boldsymbol{a}) \cdot \phi(\boldsymbol{a}) \tag{16}$$

for all $\boldsymbol{a}, \hat{\boldsymbol{a}} \in \mathbf{R}^k$. Thus, the loss $L_\phi(\phi(\boldsymbol{a}), \phi(\hat{\boldsymbol{a}}))$ is the error we make if we approximate $P_\phi(\hat{\boldsymbol{a}})$ by its first-order Taylor polynomial around $\boldsymbol{a}$. Matching loss functions have

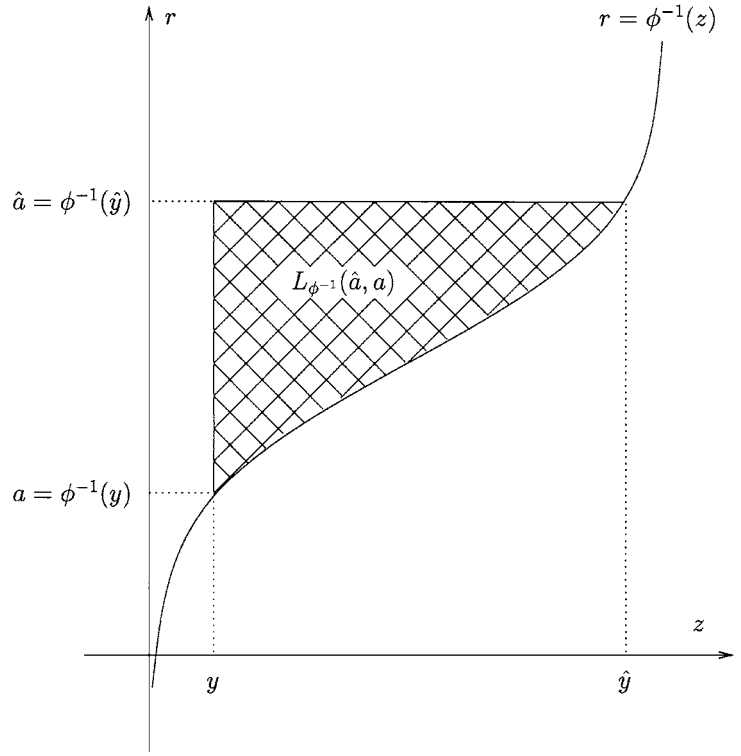

Figure 4. The matching loss for the inverted transfer function.

earlier appeared in optimization literature (Bregman, 1967), where they are known as Bregman divergences. To obtain a formula more obviously analogous with (11) we can write

$$L_\phi(\phi(\boldsymbol{a}), \phi(\hat{\boldsymbol{a}})) = \int_{\boldsymbol{a}}^{\hat{\boldsymbol{a}}} (\phi(\boldsymbol{r}) - \phi(\boldsymbol{a})) \cdot d\boldsymbol{r}$$

where the integral is a path integral the value of which must be independent of the actual path chosen between $\boldsymbol{a}$ and $\hat{\boldsymbol{a}}$.

Before considering when a loss function L_ϕ that satisfies (16) exists, let us establish some basic notation and compute the derivatives of $L_\phi(\phi(\boldsymbol{a}), \phi(\hat{\boldsymbol{a}}))$. For a function $\boldsymbol{f} : \mathbf{R}^k \to \mathbf{R}^k$ we denote by $\mathrm{D}\boldsymbol{f}$ the Jacobian of $\boldsymbol{f}$. That is, $[\mathrm{D}\boldsymbol{f}(\boldsymbol{a})]_{ij} = \partial f_i(\boldsymbol{a})/\partial a_j$. Similarly, for $G : \mathbf{R}^k \to \mathbf{R}$ we use $\mathrm{D}^2 G$ for the Hessian of G, so $[\mathrm{D}^2 G(\boldsymbol{a})]_{ij} = \partial^2 G(\boldsymbol{a})/(\partial a_j \partial a_i)$. Assuming that P_ϕ is twice differentiable, our convexity assumption means that the Hessian $\mathrm{D}^2 P_\phi(\boldsymbol{a})$, i.e., the Jacobian $\mathrm{D}\phi(\boldsymbol{a})$, is positive semidefinite for all $\boldsymbol{a}$. (Remember also that any Hessian, and therefore the Jacobian of any function with a potential function, is symmetrical.)

By differentiating (16) with respect to $\hat{\boldsymbol{a}}$ we see directly that (3) holds for the matching loss function we just defined. Differentiating with respect to $\boldsymbol{a}$ gives us

$$\begin{aligned}\nabla_{\boldsymbol{a}} L_\phi(\phi(\boldsymbol{a}), \phi(\hat{\boldsymbol{a}})) &= -\nabla_{\boldsymbol{a}} P_\phi(\boldsymbol{a}) + (\boldsymbol{a} - \hat{\boldsymbol{a}})^{\mathrm{T}} \mathrm{D}\phi(\boldsymbol{a}) + \phi(\boldsymbol{a}) \\ &= \mathrm{D}\phi(\boldsymbol{a})(\boldsymbol{a} - \hat{\boldsymbol{a}})\end{aligned}$$

since $\phi = \nabla P_\phi$ and $\mathrm{D}\phi$ is symmetrical. In the important special case that ϕ has a differentiable inverse ϕ^{-1}, we can use the chain rule and the fact $\mathrm{D}\phi^{-1}(\boldsymbol{y}) = (\mathrm{D}\phi(\boldsymbol{a}))^{-1}$ for $\boldsymbol{y} = \phi(\boldsymbol{a})$ to further obtain

$$\nabla_{\boldsymbol{y}} L_\phi(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \phi^{-1}(\boldsymbol{y}) - \phi^{-1}(\hat{\boldsymbol{y}}). \tag{17}$$

Comparing (17) with (3) gives a hint about an important relation between the matching losses for ϕ and ϕ^{-1}. Similar to the one-dimensional case (15), we can prove a general duality (Amari, 1985; Azoury & Warmuth, 1999)

$$L_\phi(\boldsymbol{y}, \hat{\boldsymbol{y}}) = L_{\phi^{-1}}(\hat{\boldsymbol{a}}, \boldsymbol{a}) \tag{18}$$

for $\boldsymbol{y} = \phi(\boldsymbol{a})$ and $\hat{\boldsymbol{y}} = \phi(\hat{\boldsymbol{a}})$. Notice that since the Jacobians of ϕ and ϕ^{-1} are assumed to be positive definite, differentiating (3) now shows that $L_{\phi^{-1}}(\hat{\boldsymbol{a}}, \boldsymbol{a})$ is convex in $\hat{\boldsymbol{a}}$ and differentiating (17) shows that $L_\phi(\boldsymbol{y}, \hat{\boldsymbol{y}})$ is convex in $\boldsymbol{y}$. See Azoury and Warmuth (1999) and Warmuth and Jagota (1997) for how this is useful in analyzing on-line algorithms.

We now consider briefly whether (16) does define a unique loss function L_ϕ. Let us define a notation for the right hand side of (16) by

$$\Delta_\phi(\hat{\boldsymbol{a}}, \boldsymbol{a}) = P_\phi(\hat{\boldsymbol{a}}) - P_\phi(\boldsymbol{a}) - (\hat{\boldsymbol{a}} - \boldsymbol{a}) \cdot \phi(\boldsymbol{a}). \tag{19}$$

Notice that the order of the arguments on the left hand side is changed from (16). This notation has the advantage that, by (18), for an invertible ϕ we now have $\Delta_\phi = L_{\phi^{-1}}$. We call Δ_ϕ the *matching divergence function* for ϕ. For the value $L_\phi(\boldsymbol{y}, \hat{\boldsymbol{y}})$ to be uniquely defined by (16) for all $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ in the range of ϕ, we must now have $\Delta_\phi(\hat{\boldsymbol{a}}, \boldsymbol{a}) = \Delta_\phi(\hat{\boldsymbol{a}}', \boldsymbol{a}')$ when $\phi(\hat{\boldsymbol{a}}) = \phi(\hat{\boldsymbol{a}}')$ and $\phi(\boldsymbol{a}) = \phi(\boldsymbol{a}')$. We also need to check that the loss function satisfies $L_\phi(\boldsymbol{y}, \hat{\boldsymbol{y}}) > 0$ for $\boldsymbol{y} \neq \hat{\boldsymbol{y}}$. In practice, it is usually easy to check whether L_ϕ is well defined for whatever ϕ we wish to consider. This should become clear from the following examples. For completeness, we have included in Appendix A a discussion on some sufficient conditions under which L_ϕ is uniquely defined.

Example 1. Let ϕ be a linear function given by $\phi(\boldsymbol{a}) = A\boldsymbol{a}$ where the matrix $A \in \mathbf{R}^{k \times k}$ is symmetrical and positive semidefinite. Because A is symmetrical, we have $\phi = \nabla P_\phi$ where $P_\phi(\boldsymbol{a}) = \boldsymbol{a}^{\mathrm{T}} A\boldsymbol{a}/2$, and because A is positive semidefinite, P_ϕ is convex. Now (16) gives directly $L_\phi(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{2}(\boldsymbol{a} - \hat{\boldsymbol{a}})^{\mathrm{T}} A(\boldsymbol{a} - \hat{\boldsymbol{a}})$ when $\boldsymbol{y} = A\boldsymbol{a}$ and $\hat{\boldsymbol{y}} = A\hat{\boldsymbol{a}}$. If A is invertible, we therefore get $L_\phi(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{2}(\boldsymbol{y} - \hat{\boldsymbol{y}})^{\mathrm{T}} A^{-1}(\boldsymbol{y} - \hat{\boldsymbol{y}})$ for all $\boldsymbol{y}, \hat{\boldsymbol{y}} \in \mathbf{R}^k$. Even if A is not invertible, we know that it has a set of orthogonal eigenvectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ and corresponding nonnegative eigenvalues $\lambda_1, \ldots, \lambda_k$. Now the range of ϕ is the space spanned by the eigenvectors

corresponding to the positive eigenvalues of A. For any $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ in this range V_ϕ we can write $L_\phi(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{2}(\boldsymbol{y} - \hat{\boldsymbol{y}})^{\mathrm{T}} A^* (\boldsymbol{y} - \hat{\boldsymbol{y}})$ where A^* is any matrix such that $A^* \boldsymbol{x}_j = (1/\lambda_j)\boldsymbol{x}_j$ whenever $\lambda_j > 0$.

In certain situations it is useful to extend L_ϕ to the domain $\overline{V_\phi} \times \overline{V_\phi}$ where $\overline{V_\phi}$ is the closure of V_ϕ under the standard topology of $\mathbf{R}^k$. The resulting extended L_ϕ should still be continuous. In the examples we consider such a continuous extension is easily seen to exists and to be unique.

Example 2. Let $\boldsymbol{\sigma} : \mathbf{R}^k \to \mathbf{R}^k$ be the softmax function given by (4). It has a potential function given by $P_{\boldsymbol{\sigma}}(\boldsymbol{a}) = \ln \sum_{j=1}^k \exp(a_j)$. We next see that $P_{\boldsymbol{\sigma}}$ is convex. Its Hessian $\mathrm{D}^2 P_{\boldsymbol{\sigma}}$, i.e., the Jacobian $\mathrm{D}\boldsymbol{\sigma}$, is given by $\mathrm{D}\boldsymbol{\sigma}(\boldsymbol{a})_{ij} = \delta_{ij}\sigma_i(\boldsymbol{a}) - \sigma_i(\boldsymbol{a})\sigma_j(\boldsymbol{a})$ where $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ otherwise. Given a vector $\boldsymbol{x} \in \mathbf{R}^k$, let X be a random variable that has probability $\boldsymbol{\sigma}_i(\boldsymbol{a})$ of taking the value x_i. We have $\boldsymbol{x}^{\mathrm{T}} \mathrm{D}\boldsymbol{\sigma}(\boldsymbol{a})\boldsymbol{x} = \sum_{i=1}^k \sigma_i(\boldsymbol{a}) x_i^2 - \sum_{i=1}^k \sum_{j=1}^k \sigma_i(\boldsymbol{a}) x_i \sigma_j(\boldsymbol{a}) x_j = \mathrm{E}X^2 - (\mathrm{E}X)^2 = \mathrm{Var}X \geq 0$. Therefore, $P_{\boldsymbol{\sigma}}$ is convex.

Substituting the potential $P_{\boldsymbol{\sigma}}$ now into (19) gives

$$\Delta_{\boldsymbol{\sigma}}(\hat{\boldsymbol{a}}, \boldsymbol{a}) = \ln\left(\sum_{i=1}^k e^{\hat{a}_i}\right) - \ln\left(\sum_{i=1}^k e^{a_i}\right) - \sum_{i=1}^k (\hat{a}_i - a_i) \frac{e^{a_i}}{\sum_{j=1}^k e^{a_j}}.$$

From this, we get

$$\begin{aligned}\Delta_{\boldsymbol{\sigma}}(\hat{\boldsymbol{a}}, \boldsymbol{a}) &= \sum_{i=1}^k \frac{e^{a_i}}{\sum_{j=1}^k e^{a_j}} \ln\left(\frac{e^{a_i}}{\sum_{j=1}^k e^{a_j}} \Big/ \frac{e^{\hat{a}_i}}{\sum_{j=1}^k e^{\hat{a}_j}}\right) \\ &= \sum_{j=1}^k \sigma_j(\boldsymbol{a}) \ln(\sigma_j(\boldsymbol{a})/\sigma_j(\hat{\boldsymbol{a}})).\end{aligned}$$

Notice that $\Delta_{\boldsymbol{\sigma}}(\hat{\boldsymbol{a}}, \boldsymbol{a}) = \Delta_{\boldsymbol{\sigma}}(\hat{\boldsymbol{a}}', \boldsymbol{a}')$ whenever $\boldsymbol{\sigma}(\hat{\boldsymbol{a}}) = \boldsymbol{\sigma}(\hat{\boldsymbol{a}}')$ and $\boldsymbol{\sigma}(\boldsymbol{a}) = \boldsymbol{\sigma}(\boldsymbol{a}')$. Therefore, the matching loss is uniquely defined by (16), and we can write $L_{\boldsymbol{\sigma}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \sum_{j=1}^k y_j \ln(y_j/\hat{y}_j)$. The relative entropy is the matching loss function for the softmax function.

To allow $y_j = 0$ or $\hat{y}_j = 0$, we adopt the standard convention that $0 \ln 0 = 0 \ln(0/0) = 0$, and $y \ln(y/0) = \infty$ for $y > 0$. It is well known that for $y_j, \hat{y}_j > 0$ and $\sum_j y_j = \sum_j \hat{y}_j = 1$, the relative entropy $L_{\boldsymbol{\sigma}}(\boldsymbol{y}, \hat{\boldsymbol{y}})$ is nonnegative and zero only for $\boldsymbol{y} = \hat{\boldsymbol{y}}$.

Example 3. The softmax function $\boldsymbol{\sigma}$ maps $\mathbf{R}^k$ into the $(k-1)$-dimensional set $\{\boldsymbol{y} \in \mathbf{R}^k \mid \sum_i y_i = 1\}$. This results in the softmax not being invertible. Since invertibility simplifies some technicalities, it might be desirable to use the function $\boldsymbol{\rho}$ that maps the $(k-1)$-dimensional space $\mathbf{R}^{k-1}$ into the $(k-1)$ dimensional set $\{\boldsymbol{y} \in [0, 1]^{k-1} \mid \sum_i y_i \leq 1\}$ according to

$$\rho_i(\boldsymbol{a}) = \frac{e^{a_i}}{1 + \sum_{j=1}^{k-1} e^{a_j}}.$$

This function has a strictly convex potential function $P_{\boldsymbol{\rho}}$ given by $P_{\boldsymbol{\rho}}(\boldsymbol{a}) = \ln(1+\sum_{j=1}^{k-1} e^{a_j})$ and an inverse given by $\boldsymbol{\rho}_i^{-1}(\boldsymbol{y}) = \ln y_i - \ln(1 - \sum_{j=1}^{k-1} y_j)$.

To explicitly see the connection to softmax, assume $\boldsymbol{y} = \boldsymbol{\sigma}(\boldsymbol{a})$. Then also $\boldsymbol{y} = \boldsymbol{\sigma}(\boldsymbol{a}')$ where $a'_i = a_i - a_k$. On the other hand, if we write $\boldsymbol{y}' = \boldsymbol{\rho}(a'_1, \ldots, a'_{k-1})$, we have $y_i = y'_i$ for $1 \le i \le k-1$, and $y_k = 1 - \sum_{j=1}^{k-1} y'_j$. Thus, by adding some simple transformations we can replace $\boldsymbol{\sigma}$ by $\boldsymbol{\rho}$.

Example 4. Consider the mapping ϕ that normalizes its argument with respect to the Euclidean norm: $\phi(\boldsymbol{a}) = \boldsymbol{a}/\|\boldsymbol{a}\|_2$. The mapping is not well defined at the origin, but otherwise it can be written as a gradient $\phi(\boldsymbol{a}) = \nabla P_\phi(\boldsymbol{a})$ where the potential function $P_\phi(\boldsymbol{a}) = \|\boldsymbol{a}\|_2$ is convex. However, P_ϕ is not strictly convex. Moreover, we have

$$\Delta_\phi(\hat{\boldsymbol{a}}, \boldsymbol{a}) = \|\hat{\boldsymbol{a}}\|_2 - \|\boldsymbol{a}\|_2 + \frac{\boldsymbol{a}}{\|\boldsymbol{a}\|_2} \cdot (\boldsymbol{a} - \hat{\boldsymbol{a}}) = \|\hat{\boldsymbol{a}}\|_2 - \frac{\boldsymbol{a}}{\|\boldsymbol{a}\|_2} \cdot \hat{\boldsymbol{a}},$$

so $\Delta_\phi(c\hat{\boldsymbol{a}}, \boldsymbol{a}) = c\Delta_\phi(\hat{\boldsymbol{a}}, \boldsymbol{a}) \neq \Delta_\phi(\hat{\boldsymbol{a}}, \boldsymbol{a})$ for $c \neq 1$ although $\phi(\hat{\boldsymbol{a}}) = \phi(c\hat{\boldsymbol{a}})$ for all $c > 0$. Hence, $L_\phi(\boldsymbol{y}, \hat{\boldsymbol{y}})$ is not well defined by (16).

Consider now a situation in which the algorithm predicts with $\phi(\boldsymbol{a}_1)$, then sees the desired output $\phi(\boldsymbol{a}_3)$, and updates its hypothesis so that its prediction with the same input would now be $\phi(\boldsymbol{a}_2)$. We can think that the update causes the loss to decrease by the amount $\Delta_\phi(\boldsymbol{a}_1, \boldsymbol{a}_3) - \Delta_\phi(\boldsymbol{a}_2, \boldsymbol{a}_3)$. The following basic property (Warmuth & Jagota, 1997), which follows directly from the formulation (19), relates the decrease of loss to the amount $\Delta_\phi(\boldsymbol{a}_1, \boldsymbol{a}_2)$ moved by the prediction. This property of matching divergence is essential in our proofs.

Proposition 1. *For any ϕ with Δ_ϕ as its matching divergence function we have*

$$\Delta_\phi(\boldsymbol{a}_1, \boldsymbol{a}_3) = \Delta_\phi(\boldsymbol{a}_1, \boldsymbol{a}_2) + \Delta_\phi(\boldsymbol{a}_2, \boldsymbol{a}_3) + (\boldsymbol{a}_1 - \boldsymbol{a}_2) \cdot (\phi(\boldsymbol{a}_2) - \phi(\boldsymbol{a}_3))$$

for all $\boldsymbol{a}_1$, $\boldsymbol{a}_2$, and $\boldsymbol{a}_3$.

Proposition 1 can be seen as a kind of generalized triangle inequality, but there is a correction term $(\boldsymbol{a}_1 - \boldsymbol{a}_2) \cdot (\phi(\boldsymbol{a}_2) - \phi(\boldsymbol{a}_3))$, which can be positive or negative. Hence, matching divergence functions do not in general satisfy the triangle inequality.

In our proofs we also need to estimate the amount moved by the parameter vectors in one update step. The following results shows one way of doing this in terms of the matching divergence function Δ_ϕ.

Proposition 2. *For arbitrary $\boldsymbol{\theta} \in \mathbf{R}^k$ and $\boldsymbol{x} \in \mathbf{R}^k$ there is a value $0 \le s \le 1$ such that for $\boldsymbol{\theta}' = \boldsymbol{\theta} + s\boldsymbol{x}$ we have*

$$\Delta_\phi(\boldsymbol{\theta} + \boldsymbol{x}, \boldsymbol{\theta}) = \frac{1}{2}\boldsymbol{x}^{\mathrm{T}} \mathrm{D}\phi(\boldsymbol{\theta}')\, \boldsymbol{x}.$$

Proof: Recall that $\Delta_\phi(\boldsymbol{\theta}+\boldsymbol{x}, \boldsymbol{\theta}) = P_\phi(\boldsymbol{\theta}+\boldsymbol{x}) - P_\phi(\boldsymbol{\theta}) - \nabla P_\phi(\boldsymbol{\theta}) \cdot \boldsymbol{x}$ is the error in the first-order Taylor approximation for $P_\phi(\boldsymbol{\theta}+\boldsymbol{x})$ around $\boldsymbol{x} = 0$. Hence, by Taylor's Theorem we can write $\Delta_\phi(\boldsymbol{\theta}+\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{x}^{\mathrm{T}} \mathrm{D}^2 P_\phi(\boldsymbol{\theta}')\boldsymbol{x}/2$ for some $\boldsymbol{\theta}' = \boldsymbol{\theta} + s\boldsymbol{x}$ with $0 \le s \le 1$. Since the Hessian $\mathrm{D}^2 P_\phi$ is the same as the Jacobian $\mathrm{D}\boldsymbol{\phi}$, the claim follows. □

Note that Proposition 2 always gives the bound $\Delta_\phi(\boldsymbol{\theta}+\boldsymbol{x}, \boldsymbol{\theta}) \le \lambda_{\max}\|\boldsymbol{x}\|_2^2/2$ where $\lambda_{\max}$ is the largest eigenvalue of $\mathrm{D}\boldsymbol{\phi}(\boldsymbol{r})$ for any $\boldsymbol{r}$, but in special cases it may be possible to obtain significantly sharper bounds.

3. The general additive algorithm

We now introduce and motivate the general additive algorithm in an on-line prediction framework. We assume that at each *trial* t, for $t = 1, \ldots, \ell$, the learning algorithm is given an input $\boldsymbol{x}_t \in \mathbf{R}^n$, then makes its prediction $\hat{\boldsymbol{y}}_t \in \mathbf{R}^k$, and finally receives as feedback the desired output $\boldsymbol{y}_t \in \mathbf{R}^k$. We call the sequence $S = ((\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_\ell, \boldsymbol{y}_\ell))$ a *trial sequence*. The goal of the algorithm is to minimize its *total loss* for a loss function L_ϕ which we assume to be the matching loss function for some $\phi : \mathbf{R}^k \to \mathbf{R}^k$. The total loss of an algorithm A on the trial sequence S is then $\mathrm{Loss}_\phi(A, S) = \sum_{t=1}^{\ell} L_\phi(\boldsymbol{y}_t, \hat{\boldsymbol{y}}_t)$.

Assume now we are given a parameterization function $\psi : \mathbf{R}^n \to \mathbf{R}^n$ and a transfer function $\phi : \mathbf{R}^k \to \mathbf{R}^k$ that have matching loss functions L_ψ and L_ϕ, respectively. Our general additive (GA) algorithm maintains kn real-valued parameters. We denote by Θ_t the $k \times n$ matrix of the values of these parameters immediately before trial t. Further, we denote by $\boldsymbol{\theta}_{t,j}$ the jth row of Θ_t, and by $\psi(\Theta_t)$ the matrix with $\psi(\boldsymbol{\theta}_{t,j})$ as its jth row. The algorithm is given the initial parameter values $\Theta \in \mathbf{R}^{k \times n}$, and also a learning rate $\eta > 0$. Figure 5 now gives the general additive algorithm which we denote by $\mathrm{GA}(\psi, \phi, \Theta, \eta)$.

As we noticed in the introduction, in the special case that ψ is the identity function, i.e., $\Theta_t = \Omega_t$, the update of the GA algorithm is just the usual gradient descent update: Consider for simplicity the case $k = 1$, so the parameter and weight matrices Θ_t and Ω_t become simply vectors $\boldsymbol{\theta}_t$ and $\boldsymbol{\omega}_t$, respectively. Since (3) implies $\nabla_{\boldsymbol{\theta}} L_\phi(y_t, \phi(\boldsymbol{\theta} \cdot \boldsymbol{x}_t)) = (\phi(\boldsymbol{\theta} \cdot \boldsymbol{x}_t) - y_t)\boldsymbol{x}_t$,

Initialize the parameter matrix as $\Theta_1 = \Theta$.

Repeat for $t = 1, \ldots, \ell$:

- Get the input $\boldsymbol{x}_t$.
- Compute the weight matrix $\Omega_t = \psi(\Theta_t)$, i.e., $\boldsymbol{\omega}_{t,j} = \psi(\boldsymbol{\theta}_{t,j})$ where $\boldsymbol{\omega}_{t,j}$ and $\boldsymbol{\theta}_{t,j}$ are the jth row of Ω_t and Θ_t, respectively.
- Compute the linear activation $\hat{\boldsymbol{a}}_t = \Omega_t \boldsymbol{x}_t$.
- Output the prediction $\hat{\boldsymbol{y}}_t = \phi(\hat{\boldsymbol{a}}_t)$.
- For $j = 1, \ldots, k$, update the kth row of the parameter matrix by

$$\boldsymbol{\theta}_{t+1,j} = \boldsymbol{\theta}_{t,j} - \eta(\hat{y}_{t,j} - y_{t,j})\boldsymbol{x}_t. \tag{20}$$

Figure 5. The general additive algorithm $\mathrm{GA}(\psi, \phi, \Theta, \eta)$.

we can in this special case write the update (20) as

$$\begin{aligned}\boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t - \eta(\phi(\boldsymbol{\theta}_t \cdot \boldsymbol{x}_t) - y_t)\,\boldsymbol{x}_t \\ &= \boldsymbol{\theta}_t - \eta(\nabla_{\boldsymbol{\theta}} L_\phi(y_t, \phi(\boldsymbol{\theta} \cdot \boldsymbol{x}_t)))_{\boldsymbol{\theta}=\boldsymbol{\theta}_t}.\end{aligned}$$

For other parameterization functions ψ, we need to replace $\boldsymbol{\theta}_t \cdot \boldsymbol{x}_t$ in the prediction by $\boldsymbol{\omega}_t \cdot \boldsymbol{x}_t$, where $\boldsymbol{\omega}_t = \psi(\boldsymbol{\theta}_t)$. Like before, can apply (3) to rewrite the update (20) as

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta(\nabla_{\boldsymbol{\omega}} L_\phi(y_t, \phi(\boldsymbol{\omega} \cdot \boldsymbol{x}_t)))_{\boldsymbol{\omega}=\boldsymbol{\omega}_t}.$$

Notice that we now take the gradient with respect to the weights $\boldsymbol{\omega}$, but use it to update the parameters $\boldsymbol{\theta}$. Hence, this is not a usual gradient descent update any more. However, if ψ has a matching loss function, the update can be motivated by an optimization problem given in terms of this loss function. In the rest of this section, we explain this motivation.

Consider a situation in which the algorithm has seen an input $\boldsymbol{x}$, made its prediction $\hat{\boldsymbol{y}} = \phi(\psi(\Theta)\boldsymbol{x})$ based on a parameter matrix Θ, and upon receiving the desired outcome $\boldsymbol{y}$ wishes to use this new information to update its parameter matrix into $\tilde{\Theta}$. Thus, $\tilde{\Theta}$ corresponds to Θ_{t+1} in the notation of figure 5, but we do not yet wish to fix $\tilde{\Theta}$ to that particular choice of Θ_{t+1}. A natural goal in this situation would be to minimize the loss $L_\phi(\boldsymbol{y}, \phi(\psi(\tilde{\Theta})\boldsymbol{x}))$ that would result if the same example were encountered again after the update. However, there are other considerations as well. The example may have been noisy, and in any case the algorithm must somehow avoid losing too much of the information it has gained during the previous trials and stored in form of the old parameter matrix Θ. Hence, the algorithm should not move its parameters too much based on a single example. Here we choose to measure this movement by the divergence function that matches the parameterization function. This way of motivating on-line learning algorithms was introduced by Kivinen and Warmuth (1997).

To first see the idea in its simplest form, assume that both the parameterization function ψ and transfer function ϕ are invertible. Write $\Omega = \psi(\Theta)$ and $\tilde{\Omega} = \psi(\tilde{\Theta})$. To generalize the divergence function Δ_ψ originally defined for vectors in $\mathbf{R}^n$ to matrices in $\mathbf{R}^{k \times n}$ we define $\Delta_\psi(\Theta, \tilde{\Theta}) = \sum_{j=1}^{k} \Delta_\psi(\boldsymbol{\theta}_j, \tilde{\boldsymbol{\theta}}_j)$ where $\boldsymbol{\theta}_j$ and $\tilde{\boldsymbol{\theta}}_j$ are the jth row of Θ and $\tilde{\Theta}$, respectively. We measure the amount moved by the weight matrix by $L_\psi(\tilde{\Omega}, \Omega)$, and the loss of using weights $\tilde{\Omega}$ on the example $(\boldsymbol{x}, \boldsymbol{y})$ by $L_\phi(\boldsymbol{y}, \phi(\tilde{\Omega}\boldsymbol{x}))$. Recall that for invertible ϕ we can write $L_\phi = \Delta_{\phi^{-1}}$, and similarly for ψ. We thus set as the algorithm's goal to minimize the sum

$$U(\tilde{\Omega}) = \Delta_{\psi^{-1}}(\tilde{\Omega}, \Omega) + \eta\, \Delta_{\phi^{-1}}(\boldsymbol{y}, \phi(\tilde{\Omega}\boldsymbol{x})) \tag{21}$$

where $\eta > 0$ is a learning rate parameter that regulates how fast the algorithm will move its weight vector.

By (18), we have $\Delta_{\phi^{-1}}(\boldsymbol{y}, \phi(\tilde{\Omega}\boldsymbol{x})) = \Delta_\phi(\tilde{\Omega}\boldsymbol{x}, \phi^{-1}(\boldsymbol{y}))$. Hence,

$$U(\tilde{\Omega}) = \Delta_{\psi^{-1}}(\tilde{\Omega}, \Omega) + \eta\, \Delta_\phi(\tilde{\Omega}\boldsymbol{x}, \phi^{-1}(\boldsymbol{y})).$$

Notice that since the matching divergence Δ_ϕ for any ϕ is convex in its first argument, U is convex and we can minimize it by setting its derivatives to zero. We have

$$\begin{aligned}\nabla_{\tilde{\omega}_j} U(\tilde{\Omega}) &= \nabla_{\tilde{\omega}_j} \Delta_{\psi^{-1}}(\tilde{\boldsymbol{\omega}}_j, \boldsymbol{\omega}_j) + \eta \left(\frac{\partial \Delta_\phi(\tilde{\boldsymbol{a}}, \phi^{-1}(\boldsymbol{y}))}{\partial \tilde{a}_j} \right)_{\tilde{\boldsymbol{a}}=\tilde{\Omega}\boldsymbol{x}} \boldsymbol{x} \\ &= \psi^{-1}(\tilde{\boldsymbol{\omega}}_j) - \psi^{-1}(\boldsymbol{\omega}_j) + \eta(\phi_j(\tilde{\Omega}\boldsymbol{x}) - y_j)\,\boldsymbol{x}.\end{aligned}$$

Hence, the derivatives of $U(\tilde{\Omega})$ are zero when

$$\tilde{\boldsymbol{\theta}}_j = \boldsymbol{\theta}_j - \eta(\phi_j(\tilde{\Omega}\boldsymbol{x}) - y_j)\,\boldsymbol{x}. \tag{22}$$

Unfortunately, this does not quite give us a closed form for $\tilde{\Theta}$, since $\tilde{\Theta}$ appears on the right-hand side through $\tilde{\Omega} = \psi(\tilde{\Theta})$. However, it is reasonable to expect that a single update does not move the weights too much and hence $\tilde{\Omega} \approx \Omega$. Thus, as an approximate solution to the minimization problem we get

$$\tilde{\boldsymbol{\theta}}_j = \boldsymbol{\theta}_j - \eta(\hat{y}_j - y_j)\,\boldsymbol{x}. \tag{23}$$

This is the update used by the general additive algorithm. For our present purposes, considering the justification of the approximation made in going from (22) to (23) is not central, as we can directly justify the update rule (23) by the relative loss bounds of Section 4. However, analyzing the update (22) remains an interesting subject for further study. Another related subject is updates obtained by taking into the function U to be minimized not only the loss $L_\phi(\boldsymbol{y}_t, \psi(\tilde{\Theta})\boldsymbol{x}_t)$ at the present example but the total loss $\sum_{j=1}^{t} L_\phi(\boldsymbol{y}_j, \psi(\tilde{\Theta})\boldsymbol{x}_j)$ over all previous examples. In the basic case of linear regression, this results in the on-line linear least squares method. Relative loss bounds for such updates have been obtained by Foster (1991), Vovk (1998), Azoury and Warmuth (1999), and Forster (1999).

Consider now the more general case in which the transfer and parameterization function need not be invertible. The most interesting example of this situation is the softmax function. Often a simple reduction gets one back to the invertible case. For example, consider using the softmax function as the parameterization function. As we noticed in Example 3, the softmax function is closely related to an invertible function in a lower dimensional space. If we were to replace the inputs $\boldsymbol{x} \in \mathbf{R}^n$ by lower dimensional inputs $\boldsymbol{x}' = (x_1 - x_n, \ldots, x_{n-1} - x_n)$ and use the invertible function $\boldsymbol{\rho}$ of Example 3 as the parameterization function, then we would get exactly the same predictions we would get with the softmax parameterization function. A similar trick would also work for softmax as the transfer function.

Instead of dimensionality reduction, we could also keep all the original dimensions and introduce an explicit constraint into the minimization problem (21). For example, consider again the softmax function as the parameterization function. We can leave out the normalization and obtain an invertible function ψ with $\psi_i(\boldsymbol{\theta}) = e^{\theta_i}$. It turns out that the matching loss function L_ψ for this invertible ψ is given by $L_\psi(\tilde{\boldsymbol{\omega}}, \boldsymbol{\omega}) = \sum_{j=1}^{n} (\tilde{\omega}_j \ln(\tilde{\omega}_j/\omega_j) + \omega_j - \tilde{\omega}_j)$ for $\tilde{\omega}_j, \omega_j \geq 0$. Restricted to $\sum_{j=1}^{n} \tilde{\omega}_j = \sum_{j=1}^{n} \omega_j = 1$ this simplifies to the relative entropy. The relative entropy with the explicit constraint was originally used to derive the exponentiated gradient update (Kivinen & Warmuth, 1997).

Thus, there are various methods that allow us to avoid noninvertible transfer and parameterization functions if we so wish. However, instead of going into the details of these methods, we conclude this section by redoing our derivation of the general additive algorithm (figure 5) without assuming invertibility of ϕ and ψ. We thus set as the algorithm's goal to minimize the sum

$$U(\tilde{\Theta}) = \Delta_\psi(\Theta, \tilde{\Theta}) + \eta L_\phi(\boldsymbol{y}, \phi(\psi(\tilde{\Theta})\boldsymbol{x})) \tag{24}$$

where $\eta > 0$ is again a learning rate. Let $\tilde{\boldsymbol{\theta}}_j$ be the jth row of $\tilde{\Theta}$ and $\tilde{\theta}_{ji}$ the ith element of $\tilde{\boldsymbol{\theta}}_j$. Write $\tilde{\boldsymbol{a}} = \psi(\tilde{\Theta})\boldsymbol{x}$. We now have

$$\begin{aligned}
\frac{\partial L_\phi(\boldsymbol{y}, \phi(\tilde{\boldsymbol{a}}))}{\partial \tilde{\theta}_{ji}} &= \sum_{p=1}^{k} \frac{\partial \tilde{a}_p}{\partial \tilde{\theta}_{ji}} \frac{\partial L_\phi(\boldsymbol{y}, \phi(\tilde{\boldsymbol{a}}))}{\partial \tilde{a}_p} \\
&= \sum_{p=1}^{k} \frac{\partial(\psi(\tilde{\boldsymbol{\theta}}_p) \cdot \boldsymbol{x})}{\partial \tilde{\theta}_{ji}} (\phi_p(\tilde{\boldsymbol{a}}) - y_p) \\
&= \frac{\partial(\psi(\tilde{\boldsymbol{\theta}}_j) \cdot \boldsymbol{x})}{\partial \tilde{\theta}_{ji}} (\phi_j(\tilde{\boldsymbol{a}}) - y_j) \\
&= \sum_{q=1}^{n} x_q \frac{\partial \psi_q(\tilde{\boldsymbol{\theta}}_j)}{\partial \tilde{\theta}_{ji}} (\phi_j(\tilde{\boldsymbol{a}}) - y_j) \\
&= [\mathrm{D}\psi(\tilde{\boldsymbol{\theta}}_j)\boldsymbol{x}]_i (\phi_j(\tilde{\boldsymbol{a}}) - y_j).
\end{aligned}$$

(Recall that the existence of a matching divergence for ψ implies that this Jacobian $\mathrm{D}\psi(\boldsymbol{\theta})$ is symmetrical.) We also have

$$\frac{\partial \Delta_\psi(\Theta, \tilde{\Theta})}{\partial \tilde{\theta}_{ji}} = \frac{\partial \Delta_\psi(\boldsymbol{\theta}_j, \tilde{\boldsymbol{\theta}}_j)}{\partial \tilde{\theta}_{ji}} = [\mathrm{D}\psi(\tilde{\boldsymbol{\theta}}_j)(\tilde{\boldsymbol{\theta}}_j - \boldsymbol{\theta}_j)]_i.$$

Hence, the derivative of $U(\tilde{\Theta})$ with respect to the jth row vector of $\tilde{\Theta}$ is given by

$$\nabla_{\tilde{\boldsymbol{\theta}}_j} U(\tilde{\Theta}) = \mathrm{D}\psi(\tilde{\boldsymbol{\theta}}_j)(\tilde{\boldsymbol{\theta}}_j - \boldsymbol{\theta}_j + \eta(\phi_j(\tilde{\boldsymbol{a}}) - y_j)\boldsymbol{x}).$$

Thus, for $\tilde{\Theta}$ such that all the partial derivatives $\partial U(\tilde{\Theta})/\partial \tilde{\theta}_{ji}$ are zero we have

$$\tilde{\boldsymbol{\theta}}_j = \boldsymbol{\theta}_j - \eta(\phi_j(\tilde{\boldsymbol{a}}) - y_j)\boldsymbol{x} + \boldsymbol{r}_j \tag{25}$$

where the residual $\boldsymbol{r}_j$ is an arbitrary vector with the property $\mathrm{D}\psi(\tilde{\boldsymbol{\theta}}_j)\boldsymbol{r}_j = 0$. For the parameterization functions ψ we have used in our examples, it is always the case that if $\mathrm{D}\psi(\boldsymbol{\theta})\boldsymbol{r} = 0$ holds for some $\boldsymbol{\theta}$ then it holds for all $\boldsymbol{\theta}$, in other words, the zero space of the Jacobian is the same everywhere. (See discussion in the Appendix.) In this case $\mathrm{D}\psi(\boldsymbol{\theta})\boldsymbol{r} = 0$ implies $\psi(\boldsymbol{\theta}) = \psi(\boldsymbol{\theta} + \boldsymbol{r})$ for all $\boldsymbol{\theta}$, so in (25) we can just set $\boldsymbol{r}_j$ to zero (or

any other convenient value) without affecting the weights $\psi(\tilde{\boldsymbol{\theta}}_j)$ that result. Again, solving (25) is not straightforward, as $\tilde{\boldsymbol{\theta}}_j$ also appears on the right-hand side via the dependence $\tilde{\boldsymbol{a}} = \psi(\tilde{\Theta})\boldsymbol{x}$, but assuming that a single update does not move the weights too much, we can approximate $\tilde{\boldsymbol{a}}$ by $\hat{\boldsymbol{a}} = \psi(\Theta)$. Therefore, we replace (25) by

$$\tilde{\boldsymbol{\theta}}_j = \boldsymbol{\theta}_j - \eta(\phi_j(\hat{\boldsymbol{a}}) - y_j)\,\boldsymbol{x},$$

which gives a good approximation assuming $\tilde{\boldsymbol{\theta}}_j$ is reasonably close to $\boldsymbol{\theta}_j$. (Here we have also chosen $\boldsymbol{r}_j = 0$.)

4. Relative loss bounds

Our goal is to provide for the algorithm's loss upper bounds that hold for arbitrary trial sequences $S = ((\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_\ell, \boldsymbol{y}_\ell))$. In particular, we wish to avoid any statistical assumptions about the examples $(\boldsymbol{x}_t, \boldsymbol{y}_t)$. To obtain interesting bounds with minimal assumptions, we turn to relative loss bounds. We begin this section with a high-level view of what kind of bounds we are after, proceed to give the actual formal results in Theorem 3, and then discuss in more detail the interpretation of the parameters that appear in the bounds. The proof of Theorem 3 concludes the section.

Let us define $\mathrm{Loss}_\phi(\Omega, S) = \sum_{t=1}^{\ell} L_\phi(\boldsymbol{y}_t, \phi(\Omega\boldsymbol{x}_t))$ as the loss of the algorithm that keeps a constant weight matrix Ω. Then $\inf_\Omega \mathrm{Loss}_\phi(\Omega, S)$ is the least loss obtainable by a fixed predictor from the model class we use, and hence a reasonable comparison value to use in measuring the performance of the general additive algorithm. Thus, we require that the algorithm's loss should be small if there is a fixed predictor Ω that has a small loss, but if all fixed predictors have a large loss we also allow the algorithm to have a large loss.

We can prove relative loss bounds of two slightly different kinds. First, it is possible to show that with a suitable learning rate, for all trial sequences S and all fixed weight matrices Ω the general additive algorithm achieves the loss bound

$$\mathrm{Loss}_\phi(\mathrm{GA}, S) \le p\mathrm{Loss}_\phi(\Omega, S) + q \tag{26}$$

for some positive p and q. Here typically p can be taken to be a small numerical constant (say, $p = 2$) but q depends on some norms of the inputs $\boldsymbol{x}_t$ and the rows of the weight matrix Ω, as well as the transfer function ϕ and the parameterization function ψ. The role of the term q will be discussed in detail later. For now, we just notice that the bound (26) implies that on the limit of large $\mathrm{Loss}_\phi(\Omega)$, the general additive algorithm is within a small constant coefficient of the fixed predictor Ω.

We can also prove bounds of the form

$$\mathrm{Loss}_\phi(\mathrm{GA}, S) \le \mathrm{Loss}_\phi(\Omega, S) + q_1\sqrt{\mathrm{Loss}_\phi(\Omega, S)} + q_2 \tag{27}$$

where q_1 and q_2 are related to the value q in (26). Thus, we see that the loss of the general additive algorithm is asymptotically the same as the loss of any fixed predictor Ω if we ignore lower-order terms. The drawback of this stricter bound is that to achieve it, the

learning rate must be chosen very carefully, and the optimal learning rate actually depends on $\mathrm{Loss}_\phi(\Omega, S)$. Hence, (27) gives an indication of the performance of the algorithm when it is tuned well, but this bound is less robust against changes in tuning than (26) that is achieved by a learning rate that does not depend on $\mathrm{Loss}_\phi(\Omega, S)$.

The bounds (26) and (27) show us the asymptotic performance of the general additive algorithm when $\mathrm{Loss}_\phi(\Omega)$ is large for all Ω. Typically, we would expect $\mathrm{Loss}_\phi(\Omega, S)$ to be roughly linear in the length ℓ of the trial sequence S. Hence, for long sequences the lower-order terms are relatively less important. However, for short sequences the lower-order terms can be quite significant. In particular, they contain the dependence of the loss on the dimensionality n of the problem. Thus, unless the number of examples is significantly larger than the dimensionality, we need to analyze the lower-order terms carefully.

The lower-order terms in the bounds (26) and (27) will be given in terms of a product of three factors. The first factor is the divergence $\Delta_\psi(\Theta_1, \Theta)$ where Θ_1 is the algorithm's initial parameter matrix and Θ is such that $\Omega = \psi(\Theta)$. The second factor, which we define by

$$b_{\mathcal{X},\psi} = \sup\{\boldsymbol{x}^{\mathrm{T}} \mathrm{D}\psi(\boldsymbol{\theta})\boldsymbol{x} \mid \boldsymbol{\theta} \in \mathbf{R}^n, \boldsymbol{x} \in \mathcal{X}\} \tag{28}$$

where $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\ell\}$ is the set of input vectors, can be interpreted as the maximum norm of any input vector in a norm defined by the parameterization function ψ. In Example 5 below we show how $b_{\mathcal{X},\psi}$ can easily be evaluated when ψ is a linear function or the softmax function. The third factor c_ϕ, defined as

$$c_\phi = \sup\left\{ \frac{\|\boldsymbol{y} - \hat{\boldsymbol{y}}\|_2^2}{2L_\phi(\boldsymbol{y}, \hat{\boldsymbol{y}})} \,\middle|\, \boldsymbol{y}, \hat{\boldsymbol{y}} \in \overline{V_\phi} \right\}, \tag{29}$$

relates the matching loss function for the transfer function ϕ to the square loss. In Example 6 we evaluate this constant for linear functions, the softmax function, and the one-dimensional case.

Example 5. Consider bounding the value $\boldsymbol{x}^{\mathrm{T}} \mathrm{D}\boldsymbol{\sigma}(\theta)\boldsymbol{x}$ where $\boldsymbol{\sigma}$ is the softmax function (4). As we saw in Example 2, this value is a variance of a random variable with the range $\{x_1, \ldots, x_n\}$. Hence, we have $b_{\mathcal{X},\boldsymbol{\sigma}} \leq \max_{\boldsymbol{x}\in\mathcal{X}}(\max_i x_i - \min_i x_i)^2/4 \leq \max_{\boldsymbol{x}\in\mathcal{X}} \|\boldsymbol{x}\|_\infty^2$ where $\|\boldsymbol{x}\|_\infty = \max_i |x_i|$.

If ψ is a linear function with $\psi(\theta) = A\theta$ for a symmetrical positive semidefinite A, we clearly have $b_{\mathcal{X},\psi} \leq \lambda_{\max} \max_{\boldsymbol{x}\in\mathcal{X}} \|\boldsymbol{x}\|_2^2$ where $\lambda_{\max}$ is the largest eigenvalue of A.

Example 6. For the softmax function $\boldsymbol{\sigma}$ given in (4), the matching loss function $L_{\boldsymbol{\sigma}}$ is the relative entropy, for which it is well known that $L_{\boldsymbol{\sigma}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) \geq 2\|\boldsymbol{y} - \hat{\boldsymbol{y}}\|_2^2$. Hence, we have $c_\phi \leq 1/4$.

If ϕ is a linear function with $\phi(\boldsymbol{\theta}) = A\boldsymbol{\theta}$, where A is a k by k symmetrical positive semidefinite matrix, the matching loss function L is given by $L_\phi(y, \hat{y}) = \frac{1}{2}(\boldsymbol{y} - \hat{\boldsymbol{y}})^{\mathrm{T}} A^*(\boldsymbol{y} - \hat{\boldsymbol{y}})$, as explained in Example 1. From this we see that c_ϕ is the largest eigenvalue of A. (Note the restriction $\boldsymbol{y}, \hat{\boldsymbol{y}} \in V_\phi$.)

Finally, consider the special case $k = 1$, with $\phi : \mathbf{R} \to \mathbf{R}$ differentiable and strictly increasing. Let Z be a bound such that $0 < \phi'(z) \leq Z$ holds for all z. Then ϕ has a differentiable inverse ϕ^{-1}, and by doing a variable change $r = \phi^{-1}(z)$ in the integration of (11) we get

$$L_\phi(\phi(a), \phi(\hat{a})) = \int_a^{\hat{a}} (\phi(r) - \phi(a)) dr = \int_{\phi(a)}^{\phi(\hat{a})} (z - \phi(a))(\phi^{-1})'(z)\, dz.$$

Our assumptions imply $(\phi^{-1})'(z) \geq 1/Z$ for all z. If $a \leq \hat{a}$, then $z - \phi(a) \geq 0$ for $\phi(a) \leq z \leq \phi(\hat{a})$, and we can estimate

$$L_\phi(\phi(a), \phi(\hat{a})) \geq \frac{1}{Z} \int_{\phi(a)}^{\phi(\hat{a})} (z - \phi(a))\, dz = \frac{1}{2Z} (\phi(\hat{a}) - \phi(a))^2.$$

If $\hat{a} < a$, then $z - \phi(a) \leq 0$ in the range of the integration, and we get

$$L_\phi(\phi(a), \phi(\hat{a})) = -\int_{\phi(\hat{a})}^{\phi(a)} (z - \phi(a))(\phi^{-1})'(z)\, dz \geq \frac{1}{2Z} (\phi(\hat{a}) - \phi(a))^2.$$

Hence, we have $c_\phi \leq Z$.

The actual loss bounds can now be stated as follows.

Theorem 3. *Let $\psi : \mathbf{R}^n \to \mathbf{R}^n$ and $\phi : \mathbf{R}^k \to \mathbf{R}^k$ have the matching divergence functions Δ_ψ and Δ_ϕ, respectively. Consider a trial sequence $S = ((\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_\ell, \boldsymbol{y}_\ell))$ with $x_t \in \mathcal{X} \subset \mathbf{R}^n$ and $y_t \in \overline{V_\phi}$ for $t = 1, \ldots, \ell$. Let $b \geq b_{\mathcal{X},\psi}$ and $c \geq c_\phi$ where $b_{\mathcal{X},\psi}$ is as in* (28) *and c_ϕ as in* (29). *For an arbitrary initial parameter matrix $\Theta_1 \in \mathbf{R}^{k \times n}$ and the learning rate $\eta = 1/(2bc)$ we have*

$$\mathrm{Loss}_\phi(\mathrm{GA}(\psi, \phi, \Theta_1, \eta), S) \leq 2\mathrm{Loss}_\phi(\psi(\Theta^*), S) + 4bc\Delta_\psi(\Theta_1, \Theta^*). \tag{30}$$

for any parameter matrix $\Theta^* \in \mathbf{R}^{k \times n}$. Further, assume we have a parameter matrix $\Theta^* \in \mathbf{R}^{k \times n}$ such that $\mathrm{Loss}_\phi(\psi(\Theta^*), S) \leq K$ and $\Delta_\psi(\Theta_1, \Theta^*) \leq R$ hold for given values $K > 0$ and $R > 0$. For the learning rate

$$\eta = \frac{\sqrt{(bcR)^2 + Kbc R} - bcR}{Kbc} \tag{31}$$

we now have

$$\mathrm{Loss}_\phi(\mathrm{GA}(\psi, \phi, \Theta_1, \eta), S) \leq \mathrm{Loss}_\phi(\psi(\Theta^*), S) + 2\sqrt{KbcR} + 4bcR. \tag{32}$$

To understand the bounds of Theorem 3, notice first that the bound (30) is of the form (26). The bound (32) is of the form (27), assuming the parameter K is chosen such that

$K = O(\mathrm{Loss}_\phi(\psi(\Theta^*), S))$. Next we see how the actual constants in the bounds come out in some example cases.

To keep the discussion simple, let us assume that $\mathrm{Loss}_\phi(\psi(\Theta^*), S) = 0$ for some Θ^*, i.e., some weight matrix $\Omega^* = \psi(\Theta^*)$ satisfies $\boldsymbol{y}_t = \Omega^* \boldsymbol{x}_t$ for all t. In bound (32) we can then take $K = 0$, and the bound becomes $L_\phi(\mathrm{GA}, S) \leq 4bcR$. We consider first the constant c, which can be chosen based only on the transfer function ϕ. As we saw in Example 6, we can choose $c = 1$ if the transfer function is the identity function, and $c = 1/4$ if it is the softmax function. The final case we considered in Example 6 was a strictly increasing one-dimensional transfer function. In this case our choice for c is the maximum value of the derivative of the transfer function, which can be quite high for some transfer functions.

Consider now the constants b and R. The simplest case is the identity parameterization function with one-dimensional outputs, so there is just one parameter vector $\boldsymbol{\theta}_t$, and the matching divergence function is half the squared Euclidean distance. If we use the zero initial vector $\boldsymbol{\theta}_1 = \mathbf{0}$, we can take $R = \|\boldsymbol{\theta}^*\|_2^2/2$. By Example 5, the value b can be chosen as $b = \max_t \|\boldsymbol{x}_t\|_2^2$. Assuming for concreteness the identity transfer function and $c = 1$, the final bound now becomes $L_\phi(\mathrm{GA}, S) \leq 2\|\boldsymbol{\theta}^*\|_2^2 \max_t \|\boldsymbol{x}_t\|_2^2$. Thus, the bound is essentially the product of two squared norms, the norm of the correct weight vector $\boldsymbol{\theta}^*$ and the maximum norm of the instances. In the multidimensional case, the bound will have the sum of the squared norms of all the rows of the correct weight matrix Θ^*.

As another example, consider the softmax parameterization function $\psi = \boldsymbol{\sigma}$. We need to assume that the correct weight matrix Ω^* is in the range of $\boldsymbol{\sigma}$, i.e., all its entries are positive and each row sums to 1. We start again with the one-dimensional case. By Example 5, we can take $b = \max_t \|\boldsymbol{x}_t\|_\infty^2$. The matching divergence Δ_ψ is now the relative entropy. To see a crude upper bound for it, choose again zero initial parameters $\boldsymbol{\theta}_1 = \mathbf{0}$, so $\boldsymbol{\omega}_1 = (1/n, \ldots, 1/n)$. For any parameter vector $\boldsymbol{\theta}^* \in \mathbf{R}^n$ and the corresponding weight vector $\boldsymbol{\omega} = \boldsymbol{\sigma}(\boldsymbol{\theta})$, we then have

$$\begin{aligned}\Delta_\psi(\boldsymbol{\theta}_1, \boldsymbol{\theta}) &= \sum_{i=1}^n \omega_i \ln \frac{\omega_i}{\omega_{1,i}} \\ &= \sum_{i=1}^n \omega_i \ln \omega_i + \ln n \\ &\leq \ln n.\end{aligned}$$

In this case we can thus take $R = \ln n$, so the final bound (for identity transfer function with $c = 1$) becomes $L_\phi(\mathrm{GA}, S) \leq 4 \max_t \|\boldsymbol{x}_t\|_\infty^2 \ln n$. For the multidimensional case we get an additional factor of k, since we need to replace the relative entropy for one weight vector with the sum of relative entropies for all the rows in the weight matrix.

The assumption that Ω^* is in the range of $\boldsymbol{\sigma}$ is of course quite restricting. This restriction can be circumvented by a standard reduction. We replace each $\boldsymbol{x}_t = (x_{t,1}, \ldots, x_{t,n}) \in \mathbf{R}^n$ by $\boldsymbol{x}'_t = (Ux_{t,1}, \ldots, Ux_{t,n}, -Ux_{t,1}, \ldots, -Ux_{t,n}) \in \mathbf{R}^{2n}$ for a suitable constant $U > 0$. Then for any vector $\boldsymbol{\omega} \in \mathbf{R}^n$ with $\|\boldsymbol{\omega}\|_1 \leq U$ there is a vector $\boldsymbol{\omega}' \in [0, 1]^n$ such that we have $\boldsymbol{\omega} \cdot \boldsymbol{x}_t = \boldsymbol{\omega}' \cdot \boldsymbol{x}'_t$ for all $\boldsymbol{x}_t$ and additionally $\sum_{i=1}^n \omega'_i = 1$. Assuming now that we know an upper bound $U \geq \|\boldsymbol{\omega}^*\|_1$ for the 1-norm of the correct weight vector $\boldsymbol{\omega}^*$, it is easy to

see that an algorithm GA$'$ that works as GA but transforms each inputs $\boldsymbol{x}_t$ into $\boldsymbol{x}'_t$ as above achieves the loss bound $L_\phi(\mathrm{GA}', S) \leq 4U^2 \max_t \|\boldsymbol{x}_t\|_\infty^2 \ln(2n)$. Hence, again the bound has the squared product of two norms.

As we have seen, the GA algorithm with the identity parameterization function corresponds to the gradient descent algorithm. The GA algorithm with the softmax parameterization function corresponds to the exponentiated gradient algorithm (Kivinen & Warmuth, 1997). Earlier work (Kivinen & Warmuth, 1997; Helmbold, Kivinen, & Warmuth, 1999) has considered the gradient descent and exponentiated gradient algorithms for one-dimensional outputs and shown loss bounds with the same products of dual norms that appear here. See those earlier papers for a discussion of the implications of such bounds and some simulation results.

Loss bounds with products of norms also appear in classification with linear threshold functions, which can be related to regression with a matching loss function via the hinge loss (Gentile & Warmuth, 1999). The Perceptron algorithm for classification is analogous to the gradient descent algorithm for regression: the loss bound depends on the product of the 2-norm of the instances and the 2-norm of the correct weight vector. The normalized Winnow algorithm (Littlestone, 1989) is analogous to the exponentiated gradient algorithm: the loss bound depends on the product of the ∞-norm of the instances and the 1-norm of the correct weight vector. Grove, Littlestone, and Schuurmans (1997) have generalized this by giving for each $p > 2$ a classification algorithm that has a loss bound in terms of on the p-norm of the instances and the q-norm of the correct weight vector, where $1/p + 1/q = 1$. This result for general norm pairs has also been extended to linear regression (Gentile & Littlestone, 1999).

We now turn to proving Theorem 3. The following key lemma is a straightforward generalization of a result of Warmuth and Jagota (1997).

Lemma 4. *Let Θ_{t+1} denote the parameter matrix of* $\mathrm{GA}(\psi, \phi, \Theta_1, \eta)$ *after trial t on a trial sequence S. For an arbitrary parameter matrix $\Theta^* \in \mathbf{R}^{k \times n}$ we now have*

$$\begin{aligned}\mathrm{Loss}_\phi(\mathrm{GA}(\psi, \phi, \Theta_1, \eta), S) \leq\ & \mathrm{Loss}_\phi(\psi(\Theta^*), S) \\ & + \frac{1}{\eta}(\Delta_\psi(\Theta_1, \Theta^*) - \Delta_\psi(\Theta_{\ell+1}, \Theta^*)) \\ & + \frac{1}{\eta}\sum_{t=1}^{\ell} \Delta_\psi(\Theta_{t+1}, \Theta_t). \end{aligned} \tag{33}$$

Proof: We consider a trial sequence $S = ((\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_\ell, \boldsymbol{y}_\ell))$ with $\boldsymbol{y}_t \in V_\phi$ for $t = 1, \ldots, \ell$. For the more general case $\boldsymbol{y}_t \in \overline{V_\phi}$ the claim follows by continuity.

Let now $\boldsymbol{\theta}^*_j$ and $\boldsymbol{\theta}_{t,j}$ be the jth row of Θ^* and Θ_t, respectively. By Proposition 1, we can write

$$\begin{aligned}&\Delta_\psi(\Theta_t, \Theta^*) - \Delta_\psi(\Theta_{t+1}, \Theta^*) \\ &\quad = \sum_{j=1}^{k} (\Delta_\psi(\boldsymbol{\theta}_{t,j}, \boldsymbol{\theta}^*_j) - \Delta_\psi(\boldsymbol{\theta}_{t+1,j}, \boldsymbol{\theta}^*_j))\end{aligned}$$

$$= -\sum_{j=1}^{k}(\Delta_\psi(\boldsymbol{\theta}_{t+1,j}, \boldsymbol{\theta}_{t,j}) + (\psi(\boldsymbol{\theta}_{t,j}) - \psi(\boldsymbol{\theta}_j^*)) \cdot (\boldsymbol{\theta}_{t+1,j} - \boldsymbol{\theta}_{t,j}))$$

$$= -\sum_{j=1}^{k}(\Delta_\psi(\boldsymbol{\theta}_{t+1,j}, \boldsymbol{\theta}_{t,j}) + \eta(y_{t,j} - \hat{y}_{t,j})(\psi(\boldsymbol{\theta}_{t,j}) \cdot \boldsymbol{x}_t - \psi(\boldsymbol{\theta}_j^*) \cdot \boldsymbol{x}_t))$$

$$= -(\Delta_\psi(\Theta_{t+1}, \Theta_t) + \eta(\boldsymbol{y}_t - \hat{\boldsymbol{y}}_t) \cdot (\hat{\boldsymbol{a}}_t - \boldsymbol{a}_t^*)),$$

where $\hat{\boldsymbol{a}}_t = \psi(\Theta_t)\boldsymbol{x}_t$ is the vector of the linear activations at time t, and similarly $\boldsymbol{a}_t^* = \psi(\Theta^*)\boldsymbol{x}_t$.

To bound the loss $L_\phi(\boldsymbol{y}_t, \hat{\boldsymbol{y}}_t)$, where now $\hat{\boldsymbol{y}}_t = \phi(\hat{\boldsymbol{a}}_t)$, in terms of $L_\phi(\boldsymbol{y}_t, \phi(\boldsymbol{a}_t^*))$, we note that $L_\phi(\boldsymbol{y}, \phi(\boldsymbol{a}))$ is convex in $\boldsymbol{a}$ and hence bounded from below by its first order Taylor expansion. Hence, by applying (3) to write the Taylor expansion we get

$$L_\phi(\boldsymbol{y}_t, \phi(\boldsymbol{a}_t^*)) \geq L_\phi(\boldsymbol{y}_t, \phi(\hat{\boldsymbol{a}}_t)) + (\phi(\hat{\boldsymbol{a}}_t) - \boldsymbol{y}_t) \cdot (\boldsymbol{a}_t^* - \hat{\boldsymbol{a}}_t).$$

Combining this with the expression given above for $\Delta_\psi(\Theta_t, \Theta^*) - \Delta_\psi(\Theta_{t+1}, \Theta^*)$ yields

$$L_\phi(\boldsymbol{y}_t, \phi(\hat{\boldsymbol{a}}_t)) \leq L_\phi(\boldsymbol{y}_t, \phi(\boldsymbol{a}_t^*)) + \frac{1}{\eta}(\Delta_\psi(\Theta_t, \Theta^*) - \Delta_\psi(\Theta_{t+1}, \Theta^*) + \Delta_\psi(\Theta_{t+1}, \Theta_t)),$$

from which (33) follows directly by a summation over t. □

Suitable bounds applied to the divergences $\Delta_\psi(\Theta_{t+1}, \Theta_t)$ in (33) give the following bound.

Lemma 5. *Let $\psi : \mathbf{R}^n \to \mathbf{R}^n$ and $\phi : \mathbf{R}^k \to \mathbf{R}^k$ have the matching divergence functions Δ_ψ and Δ_ϕ, respectively. Consider a trial sequence $S = ((\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_\ell, \boldsymbol{y}_\ell))$ with $\boldsymbol{x}_t \in \mathcal{X} \subset \mathbf{R}^n$ and $\boldsymbol{y}_t \in \overline{V_\phi}$ for $t = 1, \ldots, \ell$. Let $b \geq b_{\mathcal{X},\psi}$ and $c \geq c_\phi$ where $b_{\mathcal{X},\psi}$ is as in* (28) *and c_ϕ as in* (29). *For any initial parameter matrix $\Theta_1 \in \mathbf{R}^{k \times n}$ and learning rate η we have*

$$\mathrm{Loss}_\phi(\mathrm{GA}(\psi, \phi, \Theta_1, \eta), S) \leq \frac{\mathrm{Loss}_\phi(\psi(\Theta^*), S)}{1 - bc\eta} + \frac{\Delta_\psi(\Theta_1, \Theta^*)}{\eta - bc\eta^2} \tag{34}$$

for all parameter matrices $\Theta^* \in \mathbf{R}^{k \times n}$.

Proof: For $j = 1, \ldots, k$, write $R_j = \{(1-s)\boldsymbol{\theta}_{t,j} + s\boldsymbol{\theta}_{t+1,j} \mid 0 \leq s \leq 1\}$. By Proposition 2 and our assumptions about the functions ψ and ϕ we have

$$\Delta_\psi(\Theta_{t+1}, \Theta_t) = \sum_{j=1}^{k} \Delta_\psi(\boldsymbol{\theta}_{t+1,j}, \boldsymbol{\theta}_{t,j})$$

$$\leq \sum_{j=1}^{k} \frac{1}{2} \max_{\boldsymbol{\theta} \in \mathrm{R_j}} ((\boldsymbol{\theta}_{t+1,j} - \boldsymbol{\theta}_{t,j})^{\mathrm{T}} \mathrm{D}\psi(\boldsymbol{\theta})(\boldsymbol{\theta}_{t+1,j} - \boldsymbol{\theta}_{t,j}))$$

$$\leq \frac{1}{2} \sup_{\boldsymbol{\theta} \in \mathbf{R}^N} \left(\boldsymbol{x}_t^{\mathrm{T}} \mathrm{D}\boldsymbol{\psi}(\boldsymbol{\theta}) \boldsymbol{x}_t \right) \eta^2 \sum_{j=1}^{k} \| y_{t,j} - \hat{y}_{t,j} \|_2^2$$

$$\leq \frac{b_{\mathcal{X},\boldsymbol{\psi}}}{2} \eta^2 (\mathbf{y}_t - \hat{\mathbf{y}}_t)^2$$

$$\leq b_{\mathcal{X},\boldsymbol{\psi}}\, c_{\boldsymbol{\phi}}\, \eta^2 L_{\boldsymbol{\phi}}(\mathbf{y}_t, \hat{\mathbf{y}}_t).$$

Now (34) follows directly from (33) by omitting the negative term $-\Delta_{\boldsymbol{\psi}}(\Theta_{\ell+1}, \Theta^*)$. □

We are now ready to prove our main theorem.

Proof of Theorem 3. Substituting $\eta = 1/(2bc)$ into (34) gives (30). To obtain (32) we first notice that for $K \geq \mathrm{Loss}_{\boldsymbol{\phi}}(\boldsymbol{\psi}(\Theta^*), S)$ and $R \geq \Delta_{\boldsymbol{\psi}}(\Theta_1, \Theta^*)$ (34) implies

$$\mathrm{Loss}_{\boldsymbol{\phi}}(\mathrm{GA}(\boldsymbol{\psi}, \boldsymbol{\phi}, \Theta_1, \eta), S) \leq \mathrm{Loss}_{\boldsymbol{\phi}}(\boldsymbol{\psi}(\Theta^*), S) + K h(bc\eta, bcR/K)$$

where $h(r, z) = r/(1 - r) + z/(r - r^2)$. A simple but tedious calculation shows that $h(\sqrt{z^2 + z} - z, z) \leq 2\sqrt{z} + 4z$, so (31) implies (32). □

We close the technical part of the paper by briefly considering alternative methods for bounding the term $\sum_{t=1}^{\ell} \Delta_{\boldsymbol{\psi}}(\Theta_{t+1}, \Theta_t)$ in (33). Here we consider only the case $k = 1$ with one-dimensional outputs; some of these ideas also generalize to the multi-dimensional case. First we see how the methods used by Cesa-Bianchi (1999) to deal with non-matching loss functions relate to our present framework. Fix now a strictly increasing differentiable transfer function $\boldsymbol{\phi} : \mathbf{R} \to \mathbf{R}$, and let L be a twice differentiable function on $V_{\boldsymbol{\phi}} \times V_{\boldsymbol{\phi}}$ such that $L(y, \boldsymbol{\phi}(a))$ is convex in a. In the case of linear regression, with the identity function as the transfer function $\boldsymbol{\phi}$, this property holds for any convex loss function, and Cesa-Bianchi (1999) has shown that it also is holds for the Hellinger loss with the logistic function as the transfer function.

Note that if $L = L_{\boldsymbol{\phi}}$, the change $\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t = \eta(y_t - \hat{y}_t)\boldsymbol{x}_t$ in the parameter vector of the GA algorithm at trial t can by (3) be written as $\eta \nabla_{\boldsymbol{\omega}} L(y_t, \boldsymbol{\phi}(\boldsymbol{\omega} \cdot \boldsymbol{x}_t))$ with the gradient evaluated at $\boldsymbol{\omega} = \boldsymbol{\psi}(\boldsymbol{\theta}_t)$. Thus, with the matching loss $L = L_{\boldsymbol{\phi}}$ we obtain the update of the GA algorithm as a special case of the update

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \left(\nabla_{\boldsymbol{\omega}} L(y_t, \boldsymbol{\phi}(\boldsymbol{\omega} \cdot \boldsymbol{x}_t)) \right)_{\boldsymbol{\omega} = \boldsymbol{\psi}(\boldsymbol{\theta}_t)}. \tag{35}$$

We denote the algorithm with the more general update (35) by GA_L. Notice that if $\boldsymbol{\psi}$ is the identity function, we get the gradient descent, and for $\boldsymbol{\psi}$ the softmax, we get the exponentiated gradient algorithm.

We also define $\mathrm{Loss}_{L,\boldsymbol{\phi}}(\boldsymbol{\omega}, S) = \sum_{t=1}^{\ell} L(y_t, \boldsymbol{\phi}(\boldsymbol{\omega} \cdot \boldsymbol{x}_t))$, and $\mathrm{Loss}_{L,\boldsymbol{\phi}}(A, S)$ for a prediction algorithm A similarly. The only special property of the loss function $L_{\boldsymbol{\phi}}$ needed in the proof of Lemma 4 was that fact that $L_{\boldsymbol{\phi}}(y, \boldsymbol{\phi}(a))$ is convex in a. Hence, Lemma 4 remains true if we replace GA by GA_L and $\mathrm{Loss}_{\boldsymbol{\phi}}$ by $\mathrm{Loss}_{L,\boldsymbol{\phi}}$ where L satisfies the convexity condition given above.

To see how bounds such as those given by Cesa-Bianchi (1999) can be obtained from Lemma 4, we derive for the terms $\Delta_\psi(\boldsymbol{\theta}_{t+1}, \boldsymbol{\theta}_t)$ bounds somewhat different from those of the proof of Lemma 5. As $\nabla_\omega L(y_t, \phi(\boldsymbol{\omega} \cdot \boldsymbol{x}_t)) = (\partial L(y_t, \phi(a))/\partial a)_{a=\boldsymbol{\omega}\cdot\boldsymbol{x}_t} \boldsymbol{x}_t$, applying Proposition 2 to the update (35) yields

$$\Delta_\psi(\boldsymbol{\theta}_{t+1}, \boldsymbol{\theta}_t) = \frac{1}{2}\eta^2 \left(\frac{\partial L(y_t, \phi(a)}{\partial a} \right)^2_{a=\psi(\boldsymbol{\theta}_t)\cdot\boldsymbol{x}_t} \boldsymbol{x}_t^{\mathrm{T}} \mathrm{D}\boldsymbol{\psi}(\boldsymbol{\theta}) \boldsymbol{x}_t$$

for some $\boldsymbol{\theta}$. If we now assume a bound $|\partial L(y_t, \phi(a))/\partial a| \leq Z$ for the derivative of the loss function combined with the transfer function, we get $\Delta_\phi(\boldsymbol{\theta}_{t+1}, \boldsymbol{\theta}_t) \leq \eta^2 Z^2 b_{\mathcal{X},\psi}/2$, so Lemma 4 implies

$$\begin{aligned} &\mathrm{Loss}_{L,\phi}(\mathrm{GA}_L(\boldsymbol{\psi}, \boldsymbol{\phi}, \Theta_1, \eta), S) \\ &\quad \leq \mathrm{Loss}_{L,\phi}(\boldsymbol{\psi}(\theta^*), S) + \frac{1}{\eta}\Delta_\psi(\theta_1, \theta^*) + \frac{1}{2}\ell\eta Z^2 b_{\mathcal{X},\psi}. \end{aligned}$$

Choosing $\eta = \sqrt{2\Delta_\psi(\theta_1, \theta^*)/\ell b_{\mathcal{X},\psi}}/Z$ gives

$$\begin{aligned} &\mathrm{Loss}_{L,\phi}(\mathrm{GA}_L(\boldsymbol{\psi}, \boldsymbol{\phi}, \Theta_1, \eta), S) \\ &\quad \leq \mathrm{Loss}_{L,\phi}(\boldsymbol{\psi}(\theta^*), S) + Z\sqrt{2\ell\Delta_\psi(\theta_1, \theta^*) b_{\mathcal{X},\psi}}. \end{aligned}$$

If we now bound $b_{\mathcal{X},\psi}$ as shown in Example 5 for the identity and softmax functions, we get essentially the the basic bound given by Cesa-Bianchi (1999) for the gradient descent and exponentiated gradient algorithms. Note that here the linear dependence on $\sqrt{K}$ of (32), where K is an estimated loss for the best fixed predictor, is replaced by a linear dependence on $\sqrt{\ell}$, where ℓ is the length of the sequence. Thus, if the best fixed predictor incurs on the average a constant loss per trial, the two bounds give the same asymptotic order for the additional loss $\mathrm{Loss}(\mathrm{GA}, S) - \mathrm{Loss}(\Omega^*, S)$. Cesa-Bianchi also shows how the $\sqrt{\ell}$ dependence can be replaced by a $\sqrt{K}$ dependence if the loss function satisfies somewhat different assumptions.

As another alternative method, assume that $(y_t - \hat{y}_t)^2 \leq 4Y^2$ for some $Y > 0$. This is the case if, for example, we always have $|y_t| \leq Y$ and $|\hat{y}_t| \leq Y$. Even if we assume a bounded range for the desired outcomes y_t, which is reasonable in many situations, this does not immediately guarantee a similar bound for the predictions $\hat{y}_t$ of the GA algorithm. However, consider modifying the algorithm by defining $\tilde{y}_t = \phi(\boldsymbol{\psi}(\boldsymbol{\theta}_t) \cdot \boldsymbol{x}_t)$ and then predicting with

$$\hat{y}_t = \begin{cases} -Y & \text{if } \tilde{y}_t < -Y \\ \tilde{y}_t & \text{if } -Y \leq \tilde{y}_t \leq Y \\ Y & \text{if } Y < \tilde{y}_t. \end{cases}$$

This modified prediction is also used in the update, so we still have $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \eta(y_t - \hat{y}_t)\boldsymbol{x}_t$. Let us denote the modified algorithm by GA^Y. Then following the proof of Lemma 5, we

can show for the GA^Y algorithm the bound

$$\Delta_\psi(\boldsymbol{\theta}_{t+1}, \boldsymbol{\theta}_t) \le 2b_{\mathcal{X},\psi}\eta^2 Y^2,$$

assuming $|y_t| \le Y$. Since the proof of Lemma 4 is also valid for the modified algorithm GA^Y, we obtain

$$\begin{aligned}&\mathrm{Loss}_\phi(\mathrm{GA}^Y(\psi, \phi, \Theta_1, \eta), S)\\ &\quad\le \mathrm{Loss}_\phi(\psi(\theta^*), S) + \frac{1}{\eta}\Delta_\psi(\theta_1, \theta^*) + 2\ell\eta Y^2 b_{\mathcal{X},\psi}.\end{aligned}$$

Choosing $\eta = \sqrt{\Delta_\psi(\theta_1, \theta^*)/2\ell b_{\mathcal{X},\psi}}/Y$ gives

$$\begin{aligned}&\mathrm{Loss}_{L,\phi}(\mathrm{GA}^Y(\psi, \phi, \Theta_1, \eta), S)\\ &\quad\le \mathrm{Loss}_{L,\phi}(\psi(\theta^*), S) + 2Y\sqrt{2\ell\Delta_\psi(\theta_1, \theta^*)b_{\mathcal{X},\psi}},\end{aligned}$$

again with a linear dependence on $\sqrt{\ell}$ for Y a constant.

Appendix A: Existence of matching loss

We consider briefly some sufficient conditions under which (16) defines a unique loss function L_ϕ. For the value $L_\phi(\boldsymbol{y}, \hat{\boldsymbol{y}})$ to be uniquely defined by (16) for all $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ in the range of ϕ, we must have $\Delta_\phi(\hat{\boldsymbol{a}}, \boldsymbol{a}) = \Delta_\phi(\hat{\boldsymbol{a}}', \boldsymbol{a}')$ when $\phi(\hat{\boldsymbol{a}}) = \phi(\hat{\boldsymbol{a}}')$ and $\phi(\boldsymbol{a}) = \phi(\boldsymbol{a}')$. We also need to check that the loss function satisfies $L_\phi(\boldsymbol{y}, \hat{\boldsymbol{y}}) > 0$ for $\boldsymbol{y} \ne \hat{\boldsymbol{y}}$.

Consider first the case in which the potential function P_ϕ is *strictly* convex, i.e., for all $\boldsymbol{a}$ and $\hat{\boldsymbol{a}}$ we have $P_\phi((1-s)\boldsymbol{a} + s\hat{\boldsymbol{a}}) < (1-s)P_\phi(\boldsymbol{a}) + sP_\phi(\hat{\boldsymbol{a}})$ for $0 < s < 1$. Then the gradient ϕ is one-to-one, so the value $L_\phi(\boldsymbol{y}, \hat{\boldsymbol{y}})$ is uniquely defined by (16) for all $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$. Further, from the characterization of the matching loss as the error in the first-order Taylor approximation for P_ϕ, it is clear that the loss $L_\phi(\boldsymbol{y}, \hat{\boldsymbol{y}})$ is strictly positive for $\boldsymbol{y} \ne \hat{\boldsymbol{y}}$.

We now consider the possibility that P_ϕ is convex but not strictly convex. We also make one additional assumption. Let $\mathrm{Z}_\phi(\boldsymbol{a})$ be the zero space of the Hessian $\mathrm{D}^2 P_\phi(\boldsymbol{a})$, i.e.,

$$\mathrm{Z}_\phi(\boldsymbol{a}) = \{\boldsymbol{x} \mid \mathrm{D}^2 P_\phi(\boldsymbol{a})\boldsymbol{x} = \boldsymbol{0}\}.$$

We are next going to show that for a loss satisfying (16) to exists, it is sufficient to make the additional assumption that the zero space $\mathrm{Z}_\phi(\boldsymbol{a})$ of the Hessian is the same for all $\boldsymbol{a}$. In other words, our assumption is that if for a given $\boldsymbol{x}$ we have $\mathrm{D}^2 P_\phi(\boldsymbol{a})\boldsymbol{x} = \boldsymbol{0}$ for some $\boldsymbol{a}$ then actually $\mathrm{D}^2 P_\phi(\boldsymbol{a}')\boldsymbol{x} = \boldsymbol{0}$ for *all* $\boldsymbol{a}'$. Intuitively, this means that there is a fixed set of directions along which P_ϕ is linear, and along any other direction P_ϕ is strictly convex. We are going to prove that under this assumption, $\phi(\hat{\boldsymbol{a}}) = \phi(\hat{\boldsymbol{a}}')$ and $\phi(\boldsymbol{a}) = \phi(\boldsymbol{a}')$ implies $\Delta_\phi(\hat{\boldsymbol{a}}, \boldsymbol{a}) = \Delta_\phi(\hat{\boldsymbol{a}}', \boldsymbol{a}')$.

Consider first vectors $\boldsymbol{a}$ and $\boldsymbol{a}'$ for which $\phi(\boldsymbol{a}) = \phi(\boldsymbol{a}')$ holds. Hence, the gradient of P_ϕ is the same at $\boldsymbol{a}$ and $\boldsymbol{a}'$, so by convexity, P_ϕ must be linear on the line between $\boldsymbol{a}$ and $\boldsymbol{a}'$. In

particular, $P_\phi(\boldsymbol{a}') - P_\phi(\boldsymbol{a}) = \phi(\boldsymbol{a}) \cdot (\boldsymbol{a}' - \boldsymbol{a})$. By direct substitution into (19) we now see that $\Delta_\phi(\hat{\boldsymbol{a}}, \boldsymbol{a}) = \Delta_\phi(\hat{\boldsymbol{a}}, \boldsymbol{a}')$ holds for all $\hat{\boldsymbol{a}}$.

Assume now $\phi(\hat{\boldsymbol{a}}) = \phi(\hat{\boldsymbol{a}}')$. As above, this implies that the first order Taylor approximation for P_ϕ around $\hat{\boldsymbol{a}}$ gives the the value $P_\phi(\hat{\boldsymbol{a}}')$ exactly, so $\Delta_\phi(\hat{\boldsymbol{a}}', \hat{\boldsymbol{a}}) = 0$. This also means that the error term, given by $(\hat{\boldsymbol{a}}' - \hat{\boldsymbol{a}})^{\mathrm{T}} \mathrm{D}^2 P_\phi((1-s)\hat{\boldsymbol{a}} + s\hat{\boldsymbol{a}}')(\hat{\boldsymbol{a}}' - \hat{\boldsymbol{a}})$ for some $0 \leq s \leq 1$, must be zero. Recall that the Eigenvectors of any symmetrical $k \times k$ real matrix span $\mathbf{R}^k$, and Eigenvectors corresponding to different Eigenvalues are orthogonal. Take now an arbitrary vector $\boldsymbol{x}$ and write it as a linear combination of Eigenvectors of $\mathrm{D}^2 P_\phi(\boldsymbol{a})$. Since we assume all the corresponding Eigenvalues to be nonnegative, using this representation shows us that $\boldsymbol{x}^{\mathrm{T}} \mathrm{D}^2 P_\phi(\boldsymbol{a}')\boldsymbol{x} = 0$ implies $\mathrm{D}^2 P_\phi(\boldsymbol{a}')\boldsymbol{x} = 0$. Thus, we see that $\hat{\boldsymbol{a}}' - \hat{\boldsymbol{a}} \in \mathrm{Z}_\phi((1-s)\hat{\boldsymbol{a}} + s\hat{\boldsymbol{a}}')$ and, by our additional assumption, $\hat{\boldsymbol{a}}' - \hat{\boldsymbol{a}} \in \mathrm{Z}_\phi(\boldsymbol{a})$ for all $\boldsymbol{a}$. This implies that the component of the gradient $\phi(\boldsymbol{a})$ in the direction of $\hat{\boldsymbol{a}}' - \hat{\boldsymbol{a}}$ is constant with respect to $\boldsymbol{a}$:

$$\nabla_{\boldsymbol{a}}((\hat{\boldsymbol{a}}' - \hat{\boldsymbol{a}}) \cdot \phi(\boldsymbol{a})) = (\hat{\boldsymbol{a}}' - \hat{\boldsymbol{a}})^{\mathrm{T}} \mathrm{D}\phi(\boldsymbol{a}) = 0.$$

Hence, we get

$$\begin{aligned} \Delta_\phi(\hat{\boldsymbol{a}}', \boldsymbol{a}) - \Delta_\phi(\hat{\boldsymbol{a}}, \boldsymbol{a}) &= P_\phi(\hat{\boldsymbol{a}}') - P_\phi(\hat{\boldsymbol{a}}) + \phi(\boldsymbol{a}) \cdot (\hat{\boldsymbol{a}} - \hat{\boldsymbol{a}}') \\ &= P_\phi(\hat{\boldsymbol{a}}') - P_\phi(\hat{\boldsymbol{a}}) + \phi(\hat{\boldsymbol{a}}) \cdot (\hat{\boldsymbol{a}} - \hat{\boldsymbol{a}}') \\ &= \Delta_\phi(\hat{\boldsymbol{a}}', \hat{\boldsymbol{a}}) = 0. \end{aligned}$$

Thus, we have proved that if the zero space $\mathrm{Z}_\phi(\boldsymbol{a})$ of the Hessian is the same for all $\boldsymbol{a}$, then $\phi(\hat{\boldsymbol{a}}) = \phi(\hat{\boldsymbol{a}}')$ and $\phi(\boldsymbol{a}) = \phi(\boldsymbol{a}')$ implies $\Delta_\phi(\hat{\boldsymbol{a}}, \boldsymbol{a}) = \Delta_\phi(\hat{\boldsymbol{a}}', \boldsymbol{a}')$. Hence, (16) defines a unique value $L_\phi(\boldsymbol{y}, \hat{\boldsymbol{y}})$ for all $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ in the range of ϕ. Since P_ϕ is convex, $L_\phi(\boldsymbol{y}, \hat{\boldsymbol{y}}) \geq 0$ always holds. We still wish to show that $L_\phi(\boldsymbol{y}, \hat{\boldsymbol{y}}) = 0$ implies $\boldsymbol{y} = \hat{\boldsymbol{y}}$. Thus, assume $\Delta_\phi(\hat{\boldsymbol{a}}', \hat{\boldsymbol{a}}) = 0$. We have argued above that now the dot product $(\hat{\boldsymbol{a}}' - \hat{\boldsymbol{a}}) \cdot \phi(\boldsymbol{a})$ has the same value for all $\boldsymbol{a}$. For any $\boldsymbol{x} \in \mathbf{R}^k$, we have $P_\phi(\boldsymbol{x} + \hat{\boldsymbol{a}} - \hat{\boldsymbol{a}}') = P_\phi(\boldsymbol{x}) + (\hat{\boldsymbol{a}} - \hat{\boldsymbol{a}}') \cdot \phi(\boldsymbol{a})$ for some $\boldsymbol{a} = \boldsymbol{x} + s(\hat{\boldsymbol{a}} - \hat{\boldsymbol{a}}')$, $0 \leq s \leq 1$. Since we assume the dot product $(\hat{\boldsymbol{a}}' - \hat{\boldsymbol{a}}) \cdot \phi(\boldsymbol{a})$ to be constant with respect to $\boldsymbol{a}$, we can simply write $P_\phi(\boldsymbol{x} + \hat{\boldsymbol{a}} - \hat{\boldsymbol{a}}') = P_\phi(\boldsymbol{x}) + c(\hat{\boldsymbol{a}}, \hat{\boldsymbol{a}}')$ for some c that does not depend on $\boldsymbol{x}$. By considering $\boldsymbol{x}$ in the neighborhood of $\hat{\boldsymbol{a}}'$ we easily see that $\nabla P_\phi(\hat{\boldsymbol{a}}) = \nabla P_\phi(\hat{\boldsymbol{a}}')$, i.e., $\phi(\hat{\boldsymbol{a}}) = \phi(\hat{\boldsymbol{a}}')$.

Example 7. Consider the softmax function from Example 2. As we saw there, the Hessian $\mathrm{D}^2 P_{\boldsymbol{\sigma}}$ has the property that for any $\boldsymbol{x} \in \mathbf{R}^k$ the value $\boldsymbol{x}^{\mathrm{T}} \mathrm{D}^2 P_{\boldsymbol{\sigma}}(\boldsymbol{a})\boldsymbol{x}$ is the variance of a random variable X with range $\{x_1, \ldots, x_k\}$. Further, X takes each value x_i with a nonzero probability. Therefore, $\boldsymbol{x}^{\mathrm{T}} \mathrm{D}^2 P_{\boldsymbol{\sigma}}(\boldsymbol{a})\boldsymbol{x} = 0$ if and only if $\boldsymbol{x} = (c, \ldots, c)$ for some constant c. As we have argued above, this implies that $\mathrm{D}^2 P_{\boldsymbol{\sigma}}(\boldsymbol{a})\boldsymbol{x}$ is zero if and only if $\boldsymbol{x} = (c, \ldots, c)$ for some constant c. Hence, the zero space $\mathrm{Z}_\phi(\boldsymbol{a})$ of the Hessian is the same for all $\boldsymbol{a}$, so the matching loss is uniquely defined.

Example 8. Consider the normalization mapping $\phi(\boldsymbol{a}) = \boldsymbol{a}/\|\boldsymbol{a}\|_2$ from Example 4. The potential is given by $P_\phi(\boldsymbol{a}) = \|\boldsymbol{a}\|_2$. The zero space of the Hessian is given by

$$\mathrm{Z}_\phi(\boldsymbol{a}) = \{s\boldsymbol{a} \mid s \in \mathbf{R}\}$$

and is not constant with respect to $\boldsymbol{a}$. This is consistent with our finding in Example 4 that the matching loss is not well-defined.

Acknowledgments

The authors thank Albert Atserias, Katy Azoury, Chris Bishop, Nicolò Cesa-Bianchi, David Helmbold, and Nick Littlestone for helpful discussions.

References

Amari, S. (1985). *Differential Geometrical Methods in Statistics*. Berlin: Springer.

Auer, P., Herbster, M., & Warmuth, M. K. (1995). Exponentially many local minima for single neurons. *Advances in Neural Information Processing Systems* (vol. 8, pp. 316–317). Cambridge, MA: MIT Press.

Azoury, K. & Warmuth, M. K. (1999). Relative loss bounds for on-line density estimation with the exponential family of distributions. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence* (pp. 31–40). San Francisco, CA: Morgan Kaufmann.

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory* (pp. 144–152). New York: ACM.

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

Budinich, M. (1993). Some notes on perceptron learning. *Journal of Physics A: Mathematical and General, 26*, 4237–4247.

Cesa-Bianchi, N. (1999). Analysis of two gradient-based algorithms for on-line regression. *Journal of Computer and System Sciences, 59*, 392–411.

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

Fahrmeir, L. & Tutz, G. (1991). *Multivariate Statistical Modelling Based on Generalized Linear Models*. New York: Springer.

Forster, J. (1999). On relative loss bounds in generalized linear regression. In *Proceedings of the 12th International Symposium on Fundamentals of Computation Theory* (pp. 269–280). Berlin: Springer.

Foster, D. P. (1991). Prediction in the worst case. *The Annals of Statistics, 19*, 1084–1090.

Gentile, C. & Littlestone, N. (1999). The robustness of the p-norm algorithms. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory* (pp. 1–11). New York: ACM.

Gentile, C. & Warmuth, M. K. (1999). Linear hinge loss and average margin. *Advances in Neural Information Processing Systems* (vol. 11, pp. 225–231). Cambridge, MA: MIT Press.

Grove, A. J., Littlestone, N., & Schuurmans, D. (1997). General convergence results for linear discriminant updates. In *Proceedings of the Tenth Annual Conference on Computational Learning Theory* (pp. 171–183). New York: ACM.

Guo, Y., Bartlett, P. L., Shawe-Taylor, J., & Williamson, R. C. (1999). Covering numbers for support vector machines. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory* (pp. 267–277). New York: ACM.

Helmbold, D. P., Kivinen, J., & Warmuth, M. K. (1999). Relative loss bounds for single neurons. *IEEE Transactions on Neural Networks, 10*, 1291–1304.

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

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

Littlestone, N. (1989). Mistake bounds and logarithmic linear-threshold learning algorithms. Ph.D. Thesis, Technical Report UCSC-CRL-89-11, University of California, Santa Cruz.

McCullagh, P. & Nelder, J. A. (1989). *Generalized Linear Models*. New York: Chapman & Hall.

Vovk, V. (1998). Competitive on-line linear regression. *Advances in Neural Information Processing Systems* (vol. 10, pp. 364–370). Cambridge, MA: MIT Press.

Warmuth, M. K. & Jagota, A. K. (1997). Continuous versus discrete-time nonlinear gradient descent: Relative loss bounds and convergence. Unpublished manuscript.

Received November 1, 1999
Revised —
Accepted October 1, 2000
Final manuscript October 5, 2000