Worst-case Quadratic Loss Bounds for On-line Prediction of Linear Functions by Gradient Descent

Nicolò Cesa-Bianchi*
Philip M. Long†
Manfred K. Warmuth‡

UCSC-CRL-93-36
October 12, 1993

Board of Studies in Computer and Information Sciences
University of California, Santa Cruz
Santa Cruz, CA 95064

ABSTRACT

In this paper we study the performance of gradient descent when applied to the problem of on-line linear prediction in arbitrary inner product spaces. We show worst-case bounds on the sum of the squared prediction errors under various assumptions concerning the amount of *a priori* information about the sequence to predict. The algorithms we use are variants and extensions of on-line gradient descent. Whereas our algorithms always predict using linear functions as hypotheses, none of our results requires the data to be linearly related. In fact, the bounds proved on the total prediction loss are typically expressed as a function of the total loss of the best fixed linear predictor with bounded norm. All the upper bounds are tight to within constants. Matching lower bounds are provided in some cases. Finally, we apply our results to the problem of on-line prediction for classes of smooth functions.

Keywords: prediction, Widrow-Hoff algorithm, gradient descent, smoothing, inner product spaces, computational learning theory, on-line learning, linear systems.

---

*DSI, Università di Milano, Via Comelico 39, 20135 Milano (ITALY).
Email address: cesabian@dsi.unimi.it.

†Computer Science Department, Duke University, P.O. Box 90129, Durham, NC 27708 USA.
Email address: plong@cs.duke.edu.

‡Computer Science Department, UC Santa Cruz, Santa Cruz, CA 95064 USA.
Email: manfred@cse.ucsc.edu.

1 Introduction

In this paper we analyze algorithms in the on-line prediction model. We assume the prediction process occurs in a sequence of trials. At trial number t the prediction algorithm

- is presented with an *instance* x_t chosen from some domain $\mathcal{X}$,
- is required to return a real number $\hat{y}_t$,
- then receives a real number y_t from the environment which we interpret as the truth.

The total loss of an algorithm over a sequence of m trials is $\sum_{t=1}^{m}(\hat{y}_t - y_t)^2$. A critical aspect of this model is that when the algorithm is making its prediction $\hat{y}_t$ for the tth instance x_t, it has access to pairs (x_s, y_s) *only for* $s < t$.

We adopt a worst-case outlook, following [Daw84, Vov90, LW91, LLW91, FMG92, MF92, CFH+93] and many others, assuming nothing about the environment of the predictor, in particular the pairs $(x_1, y_1), \ldots, (x_m, y_m)$. Our results can be loosely interpreted as having the following message: "To the extent that the environment is friendly, our algorithms have small total loss." Of course, the strength of such results depends on how "friendly" is formalized. For the most general results of this paper (described in Section 4), the domain $\mathcal{X}$ is assumed to be a (real) vector space.[1] To formalize "friendly," we make use of the general notion of an inner product $(\cdot, \cdot)$, which is any function from $\mathcal{X} \times \mathcal{X}$ to $\boldsymbol{R}$ that has certain properties (see Section 3 for a list). The inner product formalization is very general. One of the simplest inner products may be defined as follows in the case that $\mathcal{X} = \boldsymbol{R}^n$ for some n:

$$(\boldsymbol{u}, \boldsymbol{v}) = \sum_{i=1}^{n} u_i v_i = \boldsymbol{u} \cdot \boldsymbol{v}.$$

Notice that for any inner product space $\langle \mathcal{X}, (\cdot, \cdot) \rangle$, for any $\boldsymbol{w} \in \mathcal{X}$, we obtain a linear function $f_{\boldsymbol{w}}$ from $\mathcal{X}$ to $\boldsymbol{R}$ by defining

$$f_{\boldsymbol{w}}(\boldsymbol{x}) := (\boldsymbol{w}, \boldsymbol{x}). \tag{1.1}$$

Typically, we express the bounds on the loss of our algorithms as a function of

$$\inf_{\boldsymbol{w}} \sum_{t} ((\boldsymbol{w}, \boldsymbol{x}_t) - y_t)^2, \tag{1.2}$$

where the infimum is taken over all $\boldsymbol{w}$ whose norm $\sqrt{(\boldsymbol{w}, \boldsymbol{w})}$ is bounded by a parameter. Roughly speaking, this quantity measures the total misfit or noise of the environment with respect to the best "model" in the inner product space. In other words, bounds in terms of (1.2) are strong to the extent that there is a (not too large) $\boldsymbol{w}$ for which $f_{\boldsymbol{w}}$ "approximately" maps $\boldsymbol{x}_t$'s to corresponding y_t's. In many cases we can even bound the additional loss of the algorithm over the above infimum similarly to the additional loss bounds of [CFH+93] obtained in a simpler setting. Our bounds are worst-case in the sense that they hold for all sequences of pairs $(\boldsymbol{x}_t, y_t)$. (In some cases we assume the norm of the $\boldsymbol{x}_t$'s is bounded by a second parameter.)

Faber and Mycielski [FM91] noted that a natural class of smooth functions of a single real variable can be defined using inner products as above. The same class of smooth functions, as well as linear functions in $\boldsymbol{R}^n$, has been heavily studied in Statistics [Har91] (however, with probabilistic assumptions). Thus, general results for learning classes of functions defined by arbitrary inner product spaces can be applied in a variety of circumstances.

[1] The general results will hold for finite and infinite dimensional vector spaces.

Faber and Mycielski proved bounds on $\sum_t (\hat{y}_t - y_t)^2$ under the assumption that there was a $\boldsymbol{w} \in \mathcal{X}$ for which for all t, $y_t = (\boldsymbol{w}, \boldsymbol{x}_t)$, and described some applications of this result for learning classes of smooth functions. Mycielski [Myc88] had already treated the special case of linear functions in $\boldsymbol{R}^n$. The algorithm they analyzed for this "noise-free" case was a generalization of the on-line gradient descent algorithm[2] to arbitrary inner product spaces. We call this algorithm **GD** (defined below). In this paper we analyze the behavior of **GD** in the case in which there isn't necessarily a $\boldsymbol{w}$ for which for all t, $y_t = (\boldsymbol{w}, \boldsymbol{x}_t)$. Faber and Mycielski [FM91] also studied this case, but their algorithms made use of side information which, in this paper, we assume is not available.

Gradient descent is an algorithm design technique which has achieved considerable practical success in more complicated hypothesis spaces, in particular neural networks [Tou89, Tou90, LMT91, MHL92]. Despite this success, there appears not to be a principled method for tuning the learning rate. In this paper, we tune the learning rate in presence of noise with the goal of minimizing the worst-case total squared loss over the best that can be obtained using elements from a given class of linear functions.

The **GD** algorithm maintains an element $\hat{\boldsymbol{w}}$ of $\mathcal{X}$ as its hypothesis which is updated between trials. For each t, let $\hat{\boldsymbol{w}}_t$ be the hypothesis before trial t (the initial hypothesis $\hat{\boldsymbol{w}}_1$ is the zero vector). **GD** predicts with $\hat{y}_t = (\hat{\boldsymbol{w}}_t, \boldsymbol{x}_t)$ and updates the hypothesis following the rule

$$\hat{\boldsymbol{w}}_{t+1} = \hat{\boldsymbol{w}}_t - \eta(\hat{y}_t - y_t)\boldsymbol{x}_t. \tag{1.3}$$

where $\eta > 0$ is the learning rate parameter.

If the real vector space $\mathcal{X}$ has finite dimension, then each element $\boldsymbol{v}$ of $\mathcal{X}$ can be uniquely represented by the real vector $\boldsymbol{c}(\boldsymbol{v})$ of its Fourier coefficients, once a basis is chosen. If the basis is orthonormal, by simple linear algebra facts we have $\hat{y}_t = (\hat{\boldsymbol{w}}_t, \boldsymbol{x}_t) = \boldsymbol{c}(\hat{\boldsymbol{w}}_t) \cdot \boldsymbol{c}(\boldsymbol{x}_t)$. Furthermore, the vector $2(\hat{y}_t - y_t)\boldsymbol{c}(\boldsymbol{x}_t)$ is the gradient, with respect to the vector $\boldsymbol{c}(\hat{\boldsymbol{w}}_t)$, of the squared error $(\hat{y}_t - y_t)^2$ for the pair $(\boldsymbol{x}_t, y_t)$. Hence, in this case, rule (1.3) is indeed an "on-line" version of gradient descent performed over the quadratic loss.

When $\mathcal{X}$ is an arbitrary real vector space, and therefore its elements may not be uniquely represented by finite tuples of reals, the **GD** algorithm is a natural generalization of on-line gradient descent[3] and may viewed as follows [MS91].[4] After each trial t, there is a set S_t of elements $\boldsymbol{w}$ of $\mathcal{X}$ for which $(\boldsymbol{w}, \boldsymbol{x}_t) = y_t$. Intuitively, our hypothesis would like to be more like the elements of S_t, since we are banking on there being a nearly functional relationship $f_{\boldsymbol{w}}$ between the $\boldsymbol{x}_s$'s and the y_s's. It does not want to change *too* much, however, because the example $(\boldsymbol{x}_t, y_t)$ may be misleading. The **GD** algorithm "takes a step" in the direction of the element of S_t which is closest to $\hat{\boldsymbol{w}}_t$ (using the natural notion of the distance between elements of an inner product space).

---

[2] Even though in the neural network community this algorithm is usually credited to Widrow and Hoff [WH60], a similar algorithm for the iterative solution of a system of linear equations was previously developed by Kaczmarz [Kac37].

[3] To be precise, if $\mathcal{X}$ has countably infinite dimension, then **GD** can still be viewed as a mapping performing on-line gradient descent. Such a mapping is clearly noncomputable in general since each step might involve the update of an infinite number of coefficients. However, note that the t-th hypothesis $\hat{\mathbf{w}}_t$ is a linear combination of the first $t-1$ examples $\{\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}\}$ and can thus be represented by $t-1$ real coefficients.

[4] Actually, this interpretation appears to be valid only in the slightly more restricted case that $\langle \mathcal{X}, (\cdot,\cdot) \rangle$ is a Hilbert space.

2 Overview of results

We now give an overview of the bounds obtained in this paper. For any $\boldsymbol{v} \in \mathcal{X}$, $||\boldsymbol{v}|| = \sqrt{(\boldsymbol{v}, \boldsymbol{v})}$ measures the "size" of $\boldsymbol{v}$. We show in Theorem 4.3 that for all sequences $\boldsymbol{s} = \langle (\boldsymbol{x}_t, y_t) \rangle_t \in (\mathcal{X} \times \boldsymbol{R})^*$ and for all positive reals X, W, and E, if $\max_t ||\boldsymbol{x}_t|| \leq X$ and $L_W(\boldsymbol{s}) \leq E$, where

$$L_W(\boldsymbol{s}) = \inf_{||\boldsymbol{w}|| \leq W} \sum_t ((\boldsymbol{w}, \boldsymbol{x}_t) - y_t)^2,$$

then the **GD** algorithm (with learning rate tuned to X,W, and E) achieves the following

$$\sum_t (\hat{y}_t - y_t)^2 \leq L_W(\boldsymbol{s}) + 2(WX)\sqrt{E} + (WX)^2. \tag{2.1}$$

(Notice that $L_W(\boldsymbol{s}) \geq L_{W'}(\boldsymbol{s})$ for all $W' \geq W$.) The above bound is tight in a very strong sense: We show in Theorem 7.1 a lower bound of $L_W(\boldsymbol{s}) + 2(WX)\sqrt{E} + (WX)^2$ that holds for all X, W, and E, also when these parameters are given to the algorithm ahead of time.

We then remove the assumption that a bound E on $L_W(\boldsymbol{s})$ is known for some W. However, we require that y_t's are in a certain range $[-Y, Y]$ for some $Y > 0$. In Theorem 4.4 we show that for all positive reals X and Y and for all sequences $\boldsymbol{s} = \langle (\boldsymbol{x}_t, y_t) \rangle_t \in (\mathcal{X} \times [-Y, Y])^*$ such that $\max_t ||\boldsymbol{x}_t|| \leq X$, the sum of squared errors incurred on $\boldsymbol{s}$ by a variant of the **GD** algorithm (with learning rate tuned to the remaining parameters X and Y) is at most

$$L_{Y/X}(\boldsymbol{s}) + 9.2\left(Y\sqrt{L_{Y/X}(\boldsymbol{s})} + Y^2\right). \tag{2.2}$$

Notice that the above result also holds when $L_{Y/X}(\boldsymbol{s})$ is replaced by $L_W(\boldsymbol{s})$ for any $W \leq Y/X$. Observe that $\sum_t (\hat{y}_t - y_t)^2 - L_{Y/X}(\boldsymbol{s})$ can be interpreted as the excess of the algorithm's total loss over the best that can be obtained using vectors $\boldsymbol{w}$ whose norms are at most Y/X. The above bound is tight within constant factors: We show in Theorem 7.2 that for all prediction algorithms A and all $X, Y, E > 0$, there is a sequence $\boldsymbol{s}$ on $\mathcal{X} \times [-Y, Y]$ such that $\max_t ||\boldsymbol{x}_t|| = X$, $L_{Y/X}(\boldsymbol{s}) = E$, and the total squared loss of A on $\boldsymbol{s}$ is at least $E + 2Y\sqrt{E} + Y^2$. However, the dimension of the inner product space must increase as a function of E. As before, the lower bound holds also if all three parameters are given to the algorithm ahead of time.

We continue by giving the algorithm less information about the sequence. For the case when only a bound X on the norm of any $\boldsymbol{x}_t$ is known, we show in Theorem 4.1 that the **GD** algorithm, tuned to X, achieves the following upper bound on the sum of its squared errors:

$$2.25 \inf_{\boldsymbol{w} \in \mathcal{X}} \left[(\max_t ||\boldsymbol{x}_t||^2)||\boldsymbol{w}||^2 + \sum_t ((\boldsymbol{w}, \boldsymbol{x}_t) - y_t)^2 \right]$$

on any sequence $\boldsymbol{s} = \langle (\boldsymbol{x}_t, y_t) \rangle_t \in (\mathcal{X} \times \boldsymbol{R})^*$ such that $\max_t ||\boldsymbol{x}_t|| \leq X$. Note that this result shows how the **GD** algorithm is able to trade-off between the "size" of a $\boldsymbol{w}$, represented by its norm, and the extent to which $\boldsymbol{w}$ "fits" the data sequence, represented by the sum of squared errors incurred by $f_{\boldsymbol{w}}$.

Finally, with no assumptions on the environment of the learner, a further variant of the **GD** algorithm has the following bound on the sum of squared errors (Theorem 4.6)

$$9 \inf_{\boldsymbol{w} \in \mathcal{X}} \left[(\max_t ||\boldsymbol{x}_t||^2)||\boldsymbol{w}||^2 + \sum_t ((\boldsymbol{w}, \boldsymbol{x}_t) - y_t)^2 \right]$$

that holds on any sequence $\boldsymbol{s} = \langle(\boldsymbol{x}_t, y_t)\rangle_t \in (\mathcal{X} \times \boldsymbol{R})^*$.

We may apply our general bounds to a class of smooth functions of a single real variable, in the manner used by Faber and Mycielski [FM91] in the case that there is a perfect smooth function. The smoothness of a function is measured by the 2-norm of its derivative. Of course, the derivative measures the steepness of a function at a given point, and therefore the 2-norm (or any norm, for that matter) of the derivative measures the tendency of the function to be steep. When normalized appropriately, the 2-norm of a function f's derivative can be seen to be between the average steepness of f and the f's maximum steepness. In Theorem 5.1 we show that if there is an (absolutely continuous) function $f : \boldsymbol{R}_+ \to \boldsymbol{R}$ with $f(0) = 0$ which tends not to be very steep and which tends to approximately map x_t's to the y_t's, and if the x_t's are not very big, then an application of the **GD** algorithm to this case obtains good bounds on the sum of squared errors. More formally, we show that, for example, if the x_t's are taken from $[0, X]$, and if $f : [0, \infty) \to \boldsymbol{R}$ satisfies $||f'||_2 = \sqrt{\int_0^X f'(u)^2\, du} \leq W$, and $\sum_t (f(x_t) - y_t)^2 \leq E$, then the predictions $\hat{y}_t$ of the special case of the general **GD** algorithm applied to this problem satisfy

$$\sum_t (\hat{y}_t - y_t)^2 \leq \inf_{||f'||_2 \leq W} \left[\sum_t (f(x_t) - y_t)^2 \right] + 2W\sqrt{XE} + W^2 X. \tag{2.3}$$

A bound of

$$\sum_t (\hat{y}_t - y_t)^2 \leq W^2 X$$

was proved by [FM91] in the case when $E = 0$. It is surprising that the time required for the algorithm we describe for this problem to make its tth prediction $\hat{y}_t$ is $O(t)$ in the uniform cost model provided that all past examples and predictions are saved. This is because, although the vector space in which we live in this application consists of functions, and therefore the **GD** algorithm requires us to add functions, we can see that the functions that arise are piecewise linear, with the pieces being a simple functions of the past examples and predictions. In the case $E = 0$, however, there is an algorithm with an optimal bound on $\sum_t (\hat{y}_t - y_t)^2$ which computes its tth prediction in $O(\log t)$ time [KL92], raising the hope that there might be a similarly efficient robust algorithm. In Theorem 5.2 we extend our result to apply to classes of smooth functions of $n > 1$ real variables studied by Faber and Mycielski [FM91] in the absence of noise. We further show that upper bound (2.3), even viewed as bound on the excess of the algorithm's total loss over the loss of the best function of "size" at most W, is optimal, constants included.

Littlestone, Long and, Warmuth [LLW91] proved bounds for another algorithm for learning linear functions in $\boldsymbol{R}^n$, in which the $\boldsymbol{x}_t$'s were measured using the infinity norm, and the $\boldsymbol{w}$'s were measured using 1-norm. The bounds for the two algorithms are incomparable because different norms are used to measure the sizes of the $\boldsymbol{x}$'s and the $\boldsymbol{w}$'s. However, the algorithm of [LLW91] does not appear to generalize to arbitrary inner product spaces as did the **GD** algorithm, and therefore those techniques do not appear to be as widely applicable.

One of the main problems with gradient descent is that it motivates a learning rule but does not give any method for choosing the step size. Our results provide a method for setting the learning rate essentially optimally when learning linear functions. An exciting research direction is to investigate to what extent the methods of this paper can be applied to analyze other simple gradient descent learning algorithms.

Our methods can also be applied to the batch setting where the whole sequence of examples is given to the learner at once and the goal of learning is to find the function that minimizes the sum of the squared errors. In the case of linear functions this can be solved directly using the linear least squares method which might be considered to be too computationally expensive. Iterative methods provide an alternative. We prove a total loss bound for a gradient descent algorithm by applying the techniques used in this paper. We then contrast this bound to the standard bound for steepest descent on the squared residual error.

The paper is organized as follows: In Section 3 we recall the notion of inner product space and define the algorithm **GD**. The upper bounds for **GD** and its variants are all proven in Section 4; in this section we also prove bounds for the normalized total loss. These results are applied in Section 5 to derive upper bounds for prediction in classes of smooth functions. The comparison with the standard steepest descent methods is given in Section 6. Corresponding lower bounds for the upper bounds of Sections 4 and 5 are then proven in Section 7. The paper is concluded in Section 8 with some discussion and open problems.

3 Preliminaries

Let $\boldsymbol{N}$ denote the positive integers, $\boldsymbol{R}$ denote the reals. Each prediction of an on-line algorithm is determined by the previous examples and the current instance. In this paper the domain of the instances is always a fixed real vector space $\mathcal{X}$. An on-line prediction algorithm A is a mapping from $(\mathcal{X} \times \boldsymbol{R})^* \times \mathcal{X}$ to $\boldsymbol{R}$. For a finite sequence $\boldsymbol{s} = \langle (x_t, y_t) \rangle_{1 \le t \le m}$ of examples we let $\hat{y}_t$ denote the *prediction* of A on the t-th *trial*, i.e.,

$$\hat{y}_t = A(((x_1, y_1), \ldots, (x_{t-1}, y_{t-1})), x_t).$$

and we call $\hat{y}_1, \ldots, \hat{y}_m$ *the sequence of A's on-line predictions for $\boldsymbol{s}$.*

An *inner product space* (sometimes called a pre-Hilbert space since the imposition of one more assumption yields the definition of a Hilbert space) consists of a real vector space $\mathcal{X}$ and a function $(\cdot, \cdot)$ (called an *inner product*) from $\mathcal{X} \times \mathcal{X}$ to $\boldsymbol{R}$ that satisfies the following for all $\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{x} \in \mathcal{X}$ and $\kappa \in \boldsymbol{R}$:

1. $(\boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{v}, \boldsymbol{u})$;
2. $(\kappa \boldsymbol{u}, \boldsymbol{v}) = \kappa(\boldsymbol{u}, \boldsymbol{v})$;
3. $(\boldsymbol{u} + \boldsymbol{v}, \boldsymbol{x}) = (\boldsymbol{u}, \boldsymbol{x}) + (\boldsymbol{v}, \boldsymbol{x})$;
4. $(\boldsymbol{x}, \boldsymbol{x}) > 0$ whenever $\boldsymbol{x} \neq 0$.

The last requirement can be dropped essentially without affecting the definition (see e.g. [You88, page 25]). For $\boldsymbol{x} \in \mathcal{X}$, the *norm* of $\boldsymbol{x}$, denoted by $||\boldsymbol{x}||$, is defined by

$$||\boldsymbol{x}|| = \sqrt{(\boldsymbol{x}, \boldsymbol{x})}.$$

(These definitions are taken from [You88].)

An example of an inner product is the dot product in $\boldsymbol{R}^n$. For $\boldsymbol{x}, \boldsymbol{y} \in \boldsymbol{R}^n$ for some positive integer n, the dot product of $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined to be

$$\boldsymbol{x} \cdot \boldsymbol{y} = \sum_{i=1}^{n} x_i y_i.$$

Algorithm GD.
Input: $\eta \geq 0$.

- Choose $\mathcal{X}$'s zero vector as initial hypothesis $\hat{\boldsymbol{w}}_1$.
- On each trial t:
  1. Get $\boldsymbol{x}_t \in \mathcal{X}$ from the environment.
  2. Predict with $\hat{y}_t = (\hat{\boldsymbol{w}}_t, \boldsymbol{x}_t)$.
  3. Get $y_t \in \mathcal{X}$ from the environment.
  4. Update the current hypothesis $\hat{\boldsymbol{w}}_t$ according to the rule

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

Figure 4.1: Pseudo-code for algorithm **GD**. (See Theorems 4.1, 4.2, 4.3, and Corollary 4.1.)

The 2-norm (or Euclidian norm) of $\boldsymbol{x} \in \boldsymbol{R}^n$ is then defined to be

$$||\boldsymbol{x}||_2 = \sqrt{\boldsymbol{x} \cdot \boldsymbol{x}} = \sqrt{\sum_{i=1}^{n} x_i^2}.$$

If f is a function from $\boldsymbol{R}$ to $\boldsymbol{R}$, we say that f is *absolutely continuous*[5] iff there exists a (Lebesgue measurable) function $g : \boldsymbol{R} \to \boldsymbol{R}$ such that for all $a, b \in \boldsymbol{R}$, $a \leq b$,

$$f(b) - f(a) = \int_a^b g(x)\, dx.$$

4 Upper bounds for the generalized gradient descent algorithm

In this section, we prove bounds on the worst case sum of squared errors made by the **GD** algorithm (described in Figure 4.1). (Technically, Figure 4.1 describes a different learning algorithm for each initial setting of the "learning rate" η. For a particular η, we will refer to the associated learning algorithm as $\mathbf{GD}_\eta$, and we will use a similar convention throughout the paper).

For the remainder of this section, fix an inner product space $\langle \mathcal{X}, (\cdot, \cdot) \rangle$. In what follows, we will analyze the **GD** algorithm and its variants starting from the case where only a bound on the norm of $\boldsymbol{x}_t$, for all t, is available to the learner ahead of time. We will then show how additional information can be exploited for tuning the learning rate η and obtaining better worst-case bounds. Finally, we will prove a bound for the case where no assumptions are made on the environment of the learner.

4.1 Bounding the size of the instances

In this section we prove that, when given a bound on $\max_t ||\boldsymbol{x}_t||$, the algorithm **GD** can obtain good bounds on the sum of squared errors. We will remove the assumption of this knowledge later through application of standard doubling techniques.

[5] This is shown to be equivalent to a more technical definition in most Calculus texts.

As a first step, we will show the following which might be interpreted as determining the "progress" per trial, that is the amount that $\mathbf{GD}_\eta$ learns from an error. The derivation is based on previous derivations used in the proof of convergence of the on-line gradient descent algorithm (see, e.g. [DH73]).

Lemma 4.1: *Choose* $\boldsymbol{x}, \hat{\boldsymbol{w}}_1, \boldsymbol{w} \in \mathcal{X}, y \in \boldsymbol{R}, \eta > 0$. *Let* $\hat{y} = (\hat{\boldsymbol{w}}_1, \boldsymbol{x})$ *and* $\hat{\boldsymbol{w}}_2 = \hat{\boldsymbol{w}}_1 + \eta(y - \hat{y})\boldsymbol{x}$. *Then*

$$||\hat{\boldsymbol{w}}_1 - \boldsymbol{w}||^2 - ||\hat{\boldsymbol{w}}_2 - \boldsymbol{w}||^2 = (2\eta - \eta^2||\boldsymbol{x}||^2)(\hat{y} - y)^2 - 2\eta(y - \hat{y})(y - (\boldsymbol{w}, \boldsymbol{x})). \tag{4.1}$$

Proof: Let $\alpha = \eta(y - \hat{y})$. Then $\hat{\boldsymbol{w}}_2 = \hat{\boldsymbol{w}}_1 + \alpha\boldsymbol{x}$. Thus

$$\begin{aligned} ||\hat{\boldsymbol{w}}_2 - \boldsymbol{w}||^2 &= ((\hat{\boldsymbol{w}}_2 - \boldsymbol{w}), (\hat{\boldsymbol{w}}_2 - \boldsymbol{w})) \\ &= ((\hat{\boldsymbol{w}}_1 + \alpha\boldsymbol{x} - \boldsymbol{w}), (\hat{\boldsymbol{w}}_1 + \alpha\boldsymbol{x} - \boldsymbol{w})) \\ &= ||\hat{\boldsymbol{w}}_1 - \boldsymbol{w}||^2 + (2\alpha\boldsymbol{x}, (\hat{\boldsymbol{w}}_1 - \boldsymbol{w})) + \alpha^2||\boldsymbol{x}||^2. \end{aligned}$$

This implies

$$\begin{aligned} ||\hat{\boldsymbol{w}}_2 - \boldsymbol{w}||^2 - ||\hat{\boldsymbol{w}}_1 - \boldsymbol{w}||^2 &= 2\alpha(\boldsymbol{x}, (\hat{\boldsymbol{w}}_1 - \boldsymbol{w})) + \alpha^2||\boldsymbol{x}||^2 \\ &= 2\alpha(\hat{y} - (\boldsymbol{w}, \boldsymbol{x})) + \alpha^2||\boldsymbol{x}||^2 \\ &= 2\alpha(\hat{y} - y) + 2\alpha(y - (\boldsymbol{w}, \boldsymbol{x})) + \alpha^2||\boldsymbol{x}||^2. \end{aligned}$$

Expanding our definition of α,

$$\begin{aligned} ||\hat{\boldsymbol{w}}_2 - \boldsymbol{w}||^2 - ||\hat{\boldsymbol{w}}_1 - \boldsymbol{w}||^2 &= -2\eta(\hat{y} - y)^2 + 2\eta(y - \hat{y})(y - (\boldsymbol{w}, \boldsymbol{x})) + \eta^2||\boldsymbol{x}||^2(y - \hat{y})^2 \\ &= -(2\eta - \eta^2||\boldsymbol{x}||^2)(\hat{y} - y)^2 + 2\eta(y - \hat{y})(y - (\boldsymbol{w}, \boldsymbol{x})), \end{aligned}$$

establishing (4.1). □

We need the following simple lemma:

Lemma 4.2: *For all* $q, r, c \in \boldsymbol{R}$ *such that* $c \leq 1$,

$$q^2 - qr \geq cq^2 - \frac{r^2}{4(1-c)}. \tag{4.2}$$

Proof. For $c = 1$ the lemma trivially holds. For $c < 1$ inequality (4.2) is equivalent to $f(q) \geq 0$ where

$$f(q) = (1-c)q^2 - qr + \frac{r^2}{4(1-c)}.$$

By differentiating f we find the unique minimum at $q = \frac{r}{2(1-c)}$ where f is seen to have value 0. □

As a second step, we show a lower bound on the progress per trial. This lower bound will be used to prove the main theorem of this section.

Lemma 4.3: *Choose* $\boldsymbol{x}, \hat{\boldsymbol{w}}_1, \boldsymbol{w} \in \mathcal{X}, y \in \boldsymbol{R}$. *Choose* $X, \beta, c \in \boldsymbol{R}$ *such that* $X \geq ||\boldsymbol{x}||$, $0 < \beta < 2$ *and* $c \leq 1$. *Let*

$$\hat{y} = (\hat{\boldsymbol{w}}_1, \boldsymbol{x}) \qquad \text{and} \qquad \hat{\boldsymbol{w}}_2 = \hat{\boldsymbol{w}}_1 + \frac{\beta}{X^2}(y - \hat{y})\boldsymbol{x}.$$

Then

$$||\hat{\boldsymbol{w}}_1 - \boldsymbol{w}||^2 - ||\hat{\boldsymbol{w}}_2 - \boldsymbol{w}||^2 \geq \frac{2\beta - \beta^2}{X^2}\left[c(\hat{y} - y)^2 - \frac{\beta^2}{(2\beta - \beta^2)^2(1-c)}(y - (\boldsymbol{w}, \boldsymbol{x}))^2\right].$$

Proof. Applying Lemma 4.1 with $\eta = \frac{\beta}{X^2}$, we get

$$\begin{aligned}
||\hat{\boldsymbol{w}}_1 - \boldsymbol{w}||^2 - ||\hat{\boldsymbol{w}}_2 - \boldsymbol{w}||^2 &= \left[\left(\frac{2\beta}{X^2} - \frac{\beta^2||\boldsymbol{x}||^2}{X^4}\right)(\hat{y}-y)^2 - \frac{2\beta}{X^2}(y-\hat{y})(y-(\boldsymbol{w},\boldsymbol{x}))\right] \\
&\geq \left[\left(\frac{2\beta}{X^2} - \frac{\beta^2}{X^2}\right)(\hat{y}-y)^2 - \frac{2\beta}{X^2}(y-\hat{y})(y-(\boldsymbol{w},\boldsymbol{x}))\right] \qquad (4.3) \\
&\geq \frac{2\beta-\beta^2}{X^2}\left[(\hat{y}-y)^2 - \frac{2\beta}{2\beta-\beta^2}|y-\hat{y}|\,|y-(\boldsymbol{w},\boldsymbol{x})|\right] \qquad (4.4) \\
&\geq \frac{2\beta-\beta^2}{X^2}\left[c(\hat{y}-y)^2 - \frac{\beta^2}{(2\beta-\beta^2)^2(1-c)}(y-(\boldsymbol{w},\boldsymbol{x}))^2\right] \qquad (4.5)
\end{aligned}$$

where Inequality (4.3) holds because $X \geq ||\boldsymbol{x}||$ and Inequality (4.5) is an application of Lemma 4.2. □

The next theorem shows that the performance of the **GD** algorithm degrades gracefully as the relationship to be modelled moves away from being $(\boldsymbol{w},\cdot)$ from some $\boldsymbol{w} \in \mathcal{X}$. Throughout the paper, for all sequences $\boldsymbol{s} = \langle(\boldsymbol{x}_t, y_t)\rangle_t \in (\mathcal{X}\times\boldsymbol{R})^*$ and all $\boldsymbol{w} \in \mathcal{X}$, let

$$L_{\boldsymbol{w}}(\boldsymbol{s}) = \sum_t ((\boldsymbol{w},\boldsymbol{x}_t) - y_t)^2,$$

and for all $W > 0$ let

$$L_W(\boldsymbol{s}) = \inf_{||\boldsymbol{w}||\leq W} L_{\boldsymbol{w}}(\boldsymbol{s}).$$

Theorem 4.1: *Choose $0 < \beta < 2$, $0 < c \leq 1$, $m \in \boldsymbol{N}$, and $\boldsymbol{s} = \langle(\boldsymbol{x}_t, y_t)\rangle_{t\leq m} \in (\mathcal{X}\times\boldsymbol{R})^m$. Let $X \geq \max_t ||\boldsymbol{x}_t||$, and let $\hat{y}_1, \ldots, \hat{y}_m$ be the sequence of $\mathbf{GD}_{\beta/X^2}$'s on-line predictions for $\boldsymbol{s}$. Then,*

$$\sum_{t=1}^m (\hat{y}_t - y_t)^2 \leq \inf_{\boldsymbol{w}\in\mathcal{X}}\left[\frac{X^2||\boldsymbol{w}||^2}{(2\beta-\beta^2)c} + \frac{L_{\boldsymbol{w}}(\boldsymbol{s})}{(2-\beta)^2 c(1-c)}\right]. \qquad (4.6)$$

In particular, if $\beta = 2/3$ and $c = 1/2$,

$$\sum_{t=1}^m (\hat{y}_t - y_t)^2 \leq 2.25 \inf_{\boldsymbol{w}\in\mathcal{X}}\left[X^2||\boldsymbol{w}||^2 + L_{\boldsymbol{w}}(\boldsymbol{s})\right]. \qquad (4.7)$$

Notice that, by setting $c = 1/2$ and by letting $\beta \to 0$, the constant on the $L_{\boldsymbol{w}}(\boldsymbol{s})$ term can be brought arbitrarily close to 1 at the expense of increasing the constant on the other term.

Proof: Choose $\boldsymbol{w} \in \mathcal{X}$. If $\hat{\boldsymbol{w}}_1, \hat{\boldsymbol{w}}_2, \ldots, \hat{\boldsymbol{w}}_{m+1}$ is the sequence of $\mathbf{GD}_{\beta/X^2}$'s hypotheses, we get

$$\begin{aligned}
&\sum_{t=1}^m \frac{2\beta-\beta^2}{X^2}\left[c(\hat{y}_t - y_t)^2 - \frac{\beta^2}{(2\beta-\beta^2)^2(1-c)}(y_t - (\boldsymbol{w},\boldsymbol{x}_t))^2\right] \\
\leq\ &\sum_{t=1}^m (||\hat{\boldsymbol{w}}_t - \boldsymbol{w}||^2 - ||\hat{\boldsymbol{w}}_{t+1} - \boldsymbol{w}||^2) \qquad \text{by Lemma 4.3} \\
=\ &||\hat{\boldsymbol{w}}_1 - \boldsymbol{w}||^2 - ||\hat{\boldsymbol{w}}_{m+1} - \boldsymbol{w}||^2 \\
\leq\ &||\boldsymbol{w}||^2 \qquad \text{since } \hat{\boldsymbol{w}}_1 = \vec{0} \text{ and } ||\cdot|| \text{ is nonnegative.}
\end{aligned}$$

Thus

$$\sum_{t=1}^{m}\left[c(\hat{y}_t - y_t)^2 - \frac{\beta^2}{(2\beta-\beta^2)^2(1-c)}(y_t - (\boldsymbol{w},\boldsymbol{x}_t))^2\right] \leq \frac{X^2||\boldsymbol{w}||^2}{2\beta-\beta^2}$$

Solving for $\sum_t(\hat{y}_t - y_t)^2$ yields

$$\sum_{t=1}^{m}(\hat{y}_t - y_t)^2 \quad \leq \quad \frac{X^2||\boldsymbol{w}||^2}{(2\beta-\beta^2)c} + \frac{\beta^2}{(2\beta-\beta^2)^2c(1-c)}L_{\boldsymbol{w}}(\boldsymbol{s})$$

establishing (4.6). Formula (4.7) then follows immediately. □

Observe that the assumption $\hat{\boldsymbol{w}}_1 = \vec{0}$ is chosen merely for convenience. If $\hat{\boldsymbol{w}}_1 \neq \vec{0}$, then the factor $||\boldsymbol{w}||^2$ in (4.6) is replaced by $||\boldsymbol{w} - \hat{\boldsymbol{w}}_1||^2$. Thus, in this more general form, the bound of Theorem 4.1 depends on the squared distance between the starting vector $\hat{\boldsymbol{w}}_1$ and the "target" $\boldsymbol{w}$.

Normalized loss

If we run algorithm **GD** with learning rate η set in each trial t to $\frac{\beta}{||\boldsymbol{x}_t||^2}$, we can then prove a variant of Theorem 4.1 for a different notion of loss (previously studied by Faber and Mycielski [FM91]) which we call *normalized loss*. The normalized loss incurred by an algorithm predicting $\hat{y}_t$ on a trial $(\boldsymbol{x}_t, y_t)$ is defined by $\frac{(\hat{y}_t - y_t)^2}{||\boldsymbol{x}_t||^2}$. We begin by proving the following result via a straightforward variant of the proof of Lemma 4.3.

Lemma 4.4: *Choose* $\boldsymbol{x}, \hat{\boldsymbol{w}}_1, \boldsymbol{w} \in \mathcal{X}, y \in \boldsymbol{R}$, $0 < \beta < 2$, *and* $0 < c \leq 1$. *Let*

$$\hat{y} = (\hat{\boldsymbol{w}}_1, \boldsymbol{x}) \qquad \text{and} \qquad \hat{\boldsymbol{w}}_2 = \hat{\boldsymbol{w}}_1 + \frac{\beta}{||\boldsymbol{x}||^2}(y - \hat{y})\boldsymbol{x}.$$

Then

$$||\hat{\boldsymbol{w}}_1 - \boldsymbol{w}||^2 - ||\hat{\boldsymbol{w}}_2 - \boldsymbol{w}||^2 \geq \frac{2\beta-\beta^2}{||\boldsymbol{x}||^2}\left[(\hat{y}-y)^2c - \frac{\beta^2}{(2\beta-\beta^2)^2(1-c)}(y - (\boldsymbol{w},\boldsymbol{x}))^2\right].$$

We now extend Theorem 4.1 to the normalized loss. Let

$$L'_{\boldsymbol{w}}(\boldsymbol{s}) = \sum_{t=1}^{m}\frac{(f_{\boldsymbol{w}}(\boldsymbol{x}_t) - y_t)^2}{||\boldsymbol{x}_t||^2}.$$

Theorem 4.2: *Choose* $0 < \beta < 2$, $m \in \boldsymbol{N}$, *and* $\boldsymbol{s} = \langle(\boldsymbol{x}_t, y_t)\rangle_{t\leq m} \in (\mathcal{X} \times \boldsymbol{R})^m$. *Let* $\hat{y}_1, \ldots, \hat{y}_m$ *be the sequence of* $\mathbf{GD}_{\beta/||\boldsymbol{x}_t||^2}$*'s on-line predictions for* $\boldsymbol{s}$. *Then,*

$$\sum_{t=1}^{m}\frac{(\hat{y}_t - y_t)^2}{||\boldsymbol{x}_t||^2} \leq \inf_{\boldsymbol{w}\in\mathcal{X}}\left[\frac{||\boldsymbol{w}||^2}{\beta(2-\beta)c} + \frac{L'_{\boldsymbol{w}}(\boldsymbol{s})}{(2-\beta)^2c(1-c)}\right]$$

for all $0 < c \leq 1$. *In particular, if* $\beta = 2/3$ *and* $c = \frac{1}{2}$,

$$\sum_{t=1}^{m}\frac{(\hat{y}_t - y_t)^2}{||\boldsymbol{x}_t||^2} \leq 2.25 \inf_{\boldsymbol{w}\in\mathcal{X}}\left[||\boldsymbol{w}||^2 + L'_{\boldsymbol{w}}(\boldsymbol{s})\right].$$

The above theorem shows that the knowledge of a bound on $||\boldsymbol{x}_t||$, for all t, is not necessary when the normalized loss is used. This raises the question of whether the setting $\eta = \frac{\beta}{||\boldsymbol{x}_t||^2}$ (for some fixed β not depending on $||\boldsymbol{x}_t||$) can be successfully used when the goal is to minimize the total unnormalized loss and no bound on $||\boldsymbol{x}_t||$ is available beforehand. On the other hand, suppose $\mathcal{X} = \boldsymbol{R}$, and the inner product is just the ordinary product on the reals. Suppose further that for $\epsilon > 0$, $x_1 = \epsilon$, and $y_1 = 1$, whereas for all $t > 1$, $x_t = 1$ and $y_t = 0$. Then for smaller and smaller ϵ, the total (unnormalized) quadratic loss of the **GD** with the above setting of η in this case is unbounded, whereas there is a w such that $\sum_t (wx_t - y_t)^2 = 1$, namely 0. (This example is due to Ethan Bernstein.)

4.2 Tuning β

The next result shows that, if certain parameters are known in advance, optimal performance can be obtained by tuning β. We need a technical lemma first. Define the function $G : \boldsymbol{R}_+^3 \to (0, 1]$ by

$$G(E, W, X) = \frac{WX}{\sqrt{E} + WX}.$$

Lemma 4.5: *For all $E, W, X > 0$*

$$\frac{(WX)^2}{\beta(2-\beta)c} + \frac{E}{(2-\beta)^2 c(1-c)} = E + (WX)^2 + 2WX\sqrt{E} \tag{4.8}$$

whenever $\beta = G(E, W, X)$ and $c = \frac{\sqrt{E}+WX}{2\sqrt{E}+WX}$.

Proof. First notice that, when β and c are chosen as in the lemma's hypothesis, $0 < \beta \leq 1$ and $\frac{1}{2} \leq c < 1$ for all $E, W, X \geq 0$. Second, observe that (4.8) can be rewritten as

$$\frac{1}{\beta(2-\beta)c} + \frac{y^2}{(2-\beta)^2 c(1-c)} = (y+1)^2 \tag{4.9}$$

where $y = \frac{\sqrt{E}}{WX}$. Now let

$$\beta = G(E, W, X) = \frac{1}{y+1} \qquad \text{and} \qquad c = \frac{y+1}{2y+1}.$$

Then

$$(2-\beta)c = 1 \qquad \text{and} \qquad (2-\beta)^2 c(1-c) = 1 - \beta = \frac{y}{y+1}.$$

By making these substitutions in (4.9) we obtain $y + 1 + y(y+1) = (y+1)^2$. □

Theorem 4.3: *For each $E, X, W \geq 0$, the algorithm $\mathbf{GD}_{G(E,W,X)/X^2}$ has the following properties.*
Choose $m \in \boldsymbol{N}$, $\boldsymbol{s} = \langle (\boldsymbol{x}_t, y_t) \rangle_{t \leq m} \in (\mathcal{X} \times \boldsymbol{R})^m$, such that $\max_t ||\boldsymbol{x}_t|| \leq X$, and $L_W(\boldsymbol{s}) \leq E$. Let $\hat{y}_1, \ldots, \hat{y}_m$ be the sequence of $\mathbf{GD}_{G(E,W,X)/X^2}$'s on-line predictions for $\boldsymbol{s}$. Then,

$$\sum_{t=1}^{m} (\hat{y}_t - y_t)^2 \leq L_W(\boldsymbol{s}) + 2WX\sqrt{E} + (WX)^2.$$

Proof. Choose $m \in \boldsymbol{N}$, $\boldsymbol{s} = \langle(\boldsymbol{x}_t, y_t)\rangle_{t \le m} \in (\mathcal{X} \times \boldsymbol{R})^m$ for which $L_W(\boldsymbol{s}) \le E$ and $\max_t ||\boldsymbol{x}_t||^2 \le X$. By Theorem 4.1, for all β and c such that $0 < \beta < 2$ and $0 < c \le 1$, we have

$$\begin{aligned}
\sum_{t=1}^{m} (\hat{y}_t - y_t)^2 &\le \inf_{\boldsymbol{w} \in \mathcal{X}} \left[\frac{X^2 \|\boldsymbol{w}\|^2}{\beta(2-\beta)c} + \frac{L_{\boldsymbol{w}}(\boldsymbol{s})}{(2-\beta)^2 c(1-c)} \right] \\
&\le \frac{(WX)^2}{\beta(2-\beta)c} + \frac{L_W(\boldsymbol{s})}{(2-\beta)^2 c(1-c)} \\
&= \frac{(WX)^2}{\beta(2-\beta)c} + \frac{L_W(\boldsymbol{s})}{(2-\beta)^2 c(1-c)} - L_W(\boldsymbol{s}) + L_W(\boldsymbol{s}) \\
&\le \frac{(WX)^2}{\beta(2-\beta)c} + \frac{E}{(2-\beta)^2 c(1-c)} - E + L_W(\boldsymbol{s})
\end{aligned}$$

since $L_W(\boldsymbol{s}) \le E$ and $0 < (2-\beta)^2 c(1-c) \le 1$ for the given ranges of c and β. Applying Lemma 4.5 for $\beta = G(E, W, X)$ and $c = \frac{\sqrt{E}+WX}{2\sqrt{E}+WX}$, we then conclude

$$\begin{aligned}
\sum_{t=1}^{m} (\hat{y}_t - y_t)^2 &\le E + 2WX\sqrt{E} + (WX)^2 - E + L_W(\boldsymbol{s}) \\
&= L_W(\boldsymbol{s}) + 2WX\sqrt{E} + (WX)^2
\end{aligned}$$

as desired. □

A corollary to Theorem 4.3 can be obtained for the normalized loss.

Corollary 4.1: *For any $E, W \ge 0$ and for any $m \in \boldsymbol{N}$. Choose $\boldsymbol{s} = \langle(\boldsymbol{x}_t, y_t)\rangle_{t \le m} \in (\mathcal{X} \times \boldsymbol{R})^m$, such that $L'_W(\boldsymbol{s}) \le E$. Let $\hat{y}_1, \ldots, \hat{y}_m$ be the sequence of $\mathbf{GD}_{G(E,W,1)/||\boldsymbol{x}_t||^2}$'s on-line predictions for $\boldsymbol{s}$. Then,*

$$\sum_{t=1}^{m} \frac{(\hat{y}_t - y_t)^2}{||\boldsymbol{x}_t||^2} \le L'_W(\boldsymbol{s}) + 2W\sqrt{E} + W^2.$$

Proof. Lemma 4.5 can be applied to Theorem 4.2 with $X = 1$ and $c = \frac{\sqrt{E}+W}{2\sqrt{E}+W}$. The derivation then closely resembles that of Theorem 4.3. □

The following corollary shows an application of Theorem 4.3 to classes of linear functions in $\boldsymbol{R}^n$. For each W, n, let $\mathrm{LIN}_{W,n}$ be the set of all functions f from $\boldsymbol{R}^n$ to $\boldsymbol{R}$ for which there exists $\boldsymbol{w} \in \boldsymbol{R}^n$, $||\boldsymbol{w}||_2 \le W$, such that for all $\boldsymbol{x} \in \boldsymbol{R}^n$, $f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x}$.

Corollary 4.2: *For each $E, X, W \ge 0$, the* **GD** *algorithm has the following properties. Choose $m, n \in \boldsymbol{N}$, $\langle(\boldsymbol{x}_t, y_t)\rangle_{t \le m} \in (\boldsymbol{R}^n \times \boldsymbol{R})^m$, such that $\max_t ||\boldsymbol{x}_t||_2 \le X$, and there is an $f \in \mathrm{LIN}_{W,n}$ for which $\sum_{t=1}^{m} (f(\boldsymbol{x}_t) - y_t)^2 \le E$. Let $\hat{y}_1, \ldots, \hat{y}_m$ be the sequence of $\mathbf{GD}_{G(E,W,X)/X^2}$'s on-line predictions for $\boldsymbol{s}$, when* **GD** *is applied to the inner product space $\mathrm{LIN}_{W,n}$. Then,*

$$\sum_{t=1}^{m} (\hat{y}_t - y_t)^2 \le \inf_{f \in \mathrm{LIN}_{W,n}} \left[\sum_{t=1}^{m} (f(\boldsymbol{x}_t) - y_t)^2 \right] + 2WX\sqrt{E} + W^2X^2.$$

In the next section, we show that techniques from [CFH+93] may also be applied to obtain a $L_{Y/X}(\boldsymbol{s}) + O(Y\sqrt{E} + Y^2)$ bound on the total loss (unnormalized) when bounds X on $||\boldsymbol{x}_t||$ and Y on $|y_t|$ are known for all t. However, the delicate interplay between L_W and W (loosely speaking, increasing W decreases L_W) has so far prevented us from obtaining such a result without knowledge of any of the three parameters W, X, and E.

Algorithm G1.
Input $X, Y \geq 0$.

- For each $i = 0, 1, \ldots$
  - Let $k_i = z^i(aY)^2$.
  - Repeat
    1. Give $\boldsymbol{x}_t$ to $\mathbf{GD}_{G(k_i,Y/X,X)/X^2}$.
    2. Get $\mathbf{GD}_{G(k_i,Y/X,X)/X^2}$'s prediction h_t.
    3. Predict with
$$\hat{y}_t = \begin{cases} -Y & \text{if } h_t < -Y \\ h_t & \text{if } |h_t| \leq Y \\ Y & \text{otherwise.} \end{cases}$$
    4. Pass y_t to $\mathbf{GD}_{G(k_i,Y/X,X)/X^2}$.

  until the total loss in this loop exceeds
$$k_i + 2Y\sqrt{k_i} + Y^2.$$

Figure 4.2: Pseudo-code for the algorithm **G1**. (See Theorem 4.4.) Here **GD** is used as a subroutine and its learning rate is set using the function G defined in Section 4.2. Optimized values for the parameters are $z = 2.618$ and $a = 2.0979$.

4.3 Bounding the range of the y_t's

We now introduce an algorithm **G1** for the case where a bound X on the norm of $\boldsymbol{x}_t$ and a bound Y on $|y_t|$, for all t, are known ahead of time. The algorithm is sketched in Figure 4.2. In the following theorem we show a bound on the difference between the total loss of **G1** and the loss of the best linear predictor $\boldsymbol{w}$ whose norm is bounded by Y/X, where X bounds the norm of the $\boldsymbol{x}_t$'s and Y bounds the norm of the y_t's. The bound Y/X on the norm of the best linear predictor comes from an application of Theorem 4.3 and is the largest value for which we can prove the result.

Theorem 4.4: *For each $X, Y \geq 0$, the algorithm* **G1** *has the following properties. Choose $m \in \boldsymbol{N}$, $\boldsymbol{s} = \langle(\boldsymbol{x}_t, y_t)\rangle_{t \leq m} \in (\mathcal{X} \times [-Y, Y])^m$ such that $\max_t \|\boldsymbol{x}_t\| \leq X$. Let $\hat{y}_1, \ldots, \hat{y}_m$ be the sequence of $\mathbf{G1}_{X,Y}$'s on-line predictions for $\boldsymbol{s}$. Then*

$$\sum_{t=1}^{m} (\hat{y}_t - y_t)^2 \leq L_{Y/X}(\boldsymbol{s}) + 9.2\left(Y\sqrt{L_{Y/X}(\boldsymbol{s})} + Y^2\right).$$

Before proving the theorem we need some preliminary lemmas.

Lemma 4.6: *The total loss of* **G1** *incurred in each loop i is at most $k_i + (2az^{i/2} + 5)Y^2$.*

Proof. By construction of **G1**, the total loss incurred in each loop i is at most $k_i + (2az^{i/2} + 1)Y^2$ plus the possible additional loss on the trial causing the exit from the loop. To upper bound this additional loss observe that **G1** always predicts with a value $\hat{y}_t$ in the range $[-Y, Y]$. By hypothesis, $y_t \in [-Y, Y]$ for all t. Hence the loss of **G1** on a single trial t is at most $4Y^2$. □

In what follows $W = Y/X$. Let $\boldsymbol{s}_i$ be the subsequence of $\boldsymbol{s}$ fed to **G1** during loop i.

Lemma 4.7: *If* **G1** *exits loop i, then $L_W(\boldsymbol{s}_i) > k_i$.*

Proof. By construction of **G1**, if **G1** exits loop i, then the total loss incurred on subsequence $\boldsymbol{s}_i$ is bigger than

$$k_i + 2Y\sqrt{k_i} + Y^2.$$

Since $|y_t| \leq Y$ and since **G1** predicts on each trial of loop i by "clipping" the prediction of $\mathbf{GD}_{G(k_i,W,X)/X^2}$ to make it fit in the range $[-Y, Y]$, we conclude that the total loss incurred by $\mathbf{GD}_{G(k_i,W,X)/X^2}$ on loop i is bigger than $k_i + 2Y\sqrt{k_i} + Y^2$ as well. Hence by Theorem 4.3 $L_W(\boldsymbol{s}_i) > k_i$ must hold. □

Lemma 4.8: *Let ℓ be the index of the last loop entered by* **G1**. *Then*

$$\ell \leq \log_z \left(1 + \frac{(z-1)L_W(\boldsymbol{s})}{(aY)^2}\right).$$

Proof.

$$\begin{aligned}
L_W(\boldsymbol{s}) &= \inf_{\|\boldsymbol{w}\| \leq W} L_{\boldsymbol{w}}(\boldsymbol{s}) \\
&= \inf_{\|\boldsymbol{w}\| \leq W} \left[\sum_{i=0}^{\ell} L_{\boldsymbol{w}}(\boldsymbol{s}_i)\right] \\
&\geq \sum_{i=0}^{\ell} \left[\inf_{\|\boldsymbol{w}\| \leq W} L_{\boldsymbol{w}}(\boldsymbol{s}_i)\right] \\
&= \sum_{i=0}^{\ell} L_W(\boldsymbol{s}_i) \\
&\geq \sum_{i=0}^{\ell-1} k_i + L_W(\boldsymbol{s}_\ell) \qquad \text{by Lemma 4.7} \\
&> (aY)^2 \sum_{i=0}^{\ell-1} z^i \\
&= (aY)^2 \frac{z^\ell - 1}{z - 1}.
\end{aligned}$$

Solving for ℓ finally yields the lemma. □

Lemma 4.9: *The total loss on* **G1** *on the last loop ℓ entered is at most*

$$L_W(\boldsymbol{s}_\ell) + (2az^{\ell/2} + 5)Y^2.$$

Proof. By construction of **G1**, the total loss L_ℓ of **G1** on loop ℓ is the total loss of $\mathbf{GD}_{G(k_\ell,W,X)/X^2}$ on $\boldsymbol{s}_\ell$. If $L_W(\boldsymbol{s}_\ell) \leq k_\ell$, then by Theorem 4.3

$$\begin{aligned}
L_\ell &\leq L_W(\boldsymbol{s}_\ell) + 2WX\sqrt{k_\ell} + (WX)^2 \\
&\leq L_W(\boldsymbol{s}_\ell) + 2Y\sqrt{k_\ell} + Y^2 \qquad \text{since } Y = WX \\
&= L_W(\boldsymbol{s}_\ell) + (2az^{\ell/2} + 1)Y^2 \\
&< L_W(\boldsymbol{s}_\ell) + (2az^{\ell/2} + 5)Y^2.
\end{aligned}$$

On the other hand, if $L_W(\boldsymbol{s}_\ell) > k_\ell$, then by Lemma 4.6

$$\begin{aligned}
L_\ell &\leq k_\ell + (2az^{\ell/2} + 5)Y^2 \\
&< L_W(\boldsymbol{s}_\ell) + (2az^{\ell/2} + 5)Y^2
\end{aligned}$$

and the proof is concluded. □

Lemma 4.10: *For all* $x \geq 0$, $\frac{\ln(1+x)}{\ln(2.618)} \leq 0.8362\sqrt{x}$.

Proof. The inequality in the statement of the lemma is equivalent to

$$\frac{\ln(1+x)}{\sqrt{x}} - 0.8362\ln(2.618) \leq 0.$$

The function $\frac{\ln(1+x)}{\sqrt{x}}$ has a unique maximum at $x \cong 3.921$. At this value of x the above inequality is seen to hold. □

Proof of Theorem 4.4. By Lemmas 4.6 and 4.9,

$$\begin{aligned}
& \sum_{t=1}^{m}(\hat{y}_t - y_t)^2 \\
\leq\ & \sum_{i=0}^{\ell-1}\left[k_i + (2az^{i/2}+5)Y^2\right] + L_W(\boldsymbol{s}_\ell) + (2az^{\ell/2}+5)Y^2 \\
\leq\ & \sum_{i=0}^{\ell-1} k_i + 2aY^2\sum_{i=0}^{\ell} z^{i/2} + L_W(\boldsymbol{s}_\ell) + 5(\ell+1)Y^2 \\
<\ & \sum_{i=0}^{\ell-1} L_W(\boldsymbol{s}_i) + L_W(\boldsymbol{s}_\ell) + 2aY^2\frac{z^{(\ell+1)/2}-1}{\sqrt{z}-1} + 5(\ell+1)Y^2 \quad \text{by Lemma 4.7} \\
\leq\ & L_W(\boldsymbol{s}) + 2aY^2\frac{z^{(\ell+1)/2}}{\sqrt{z}-1} - \frac{2aY^2}{\sqrt{z}-1} + 5(\ell+1)Y^2 \\
\leq\ & L_W(\boldsymbol{s}) + 2aY^2\frac{\sqrt{z\left(1+\frac{(z-1)L_W(\boldsymbol{s})}{(aY)^2}\right)}}{\sqrt{z}-1} - \frac{2aY^2}{\sqrt{z}-1} + 5(\ell+1)Y^2 \quad \text{by Lemma 4.8} \\
=\ & L_W(\boldsymbol{s}) + 2aY^2\frac{\sqrt{z}}{\sqrt{z}-1}\sqrt{1+\frac{(z-1)L_W(\boldsymbol{s})}{(aY)^2}} - \frac{2aY^2}{\sqrt{z}-1} + 5(\ell+1)Y^2 \\
<\ & L_W(\boldsymbol{s}) + 2aY^2\frac{\sqrt{z}}{\sqrt{z}-1}\left(1+\sqrt{\frac{(z-1)L_W(\boldsymbol{s})}{(aY)^2}}\right) - \frac{2aY^2}{\sqrt{z}-1} + 5(\ell+1)Y^2 \\
=\ & L_W(\boldsymbol{s}) + 2aY^2\frac{\sqrt{z}}{\sqrt{z}-1} + 2Y\frac{\sqrt{z}}{\sqrt{z}-1}\sqrt{(z-1)L_W(\boldsymbol{s})} - \frac{2aY^2}{\sqrt{z}-1} + 5(\ell+1)Y^2 \\
\leq\ & L_W(\boldsymbol{s}) + 2Y\frac{\sqrt{z}}{\sqrt{z}-1}\sqrt{(z-1)L_W(\boldsymbol{s})} + 2aY^2 \\
+\ & 5\left[\log_z\left(1+\frac{(z-1)L_W(\boldsymbol{s})}{(aY)^2}\right)+1\right]Y^2 \quad \text{by Lemma 4.8.}
\end{aligned}$$

The factor $\frac{\sqrt{z}}{\sqrt{z}-1}\sqrt{z-1}$ is minimized at $z \cong 2.618$. Plugging back in this value and using Lemma 4.10 we get

$$\begin{aligned}
& \sum_{t=1}^{m}(\hat{y}_t - y_t)^2 \\
\leq\ & L_W(\boldsymbol{s}) + 6.6604Y\sqrt{L_W(\boldsymbol{s})} + 2aY^2 + 5\left[0.8362\sqrt{\frac{1.618L_W(\boldsymbol{s})}{(aY)^2}}+1\right]Y^2 \\
\leq\ & L_W(\boldsymbol{s}) + \left(6.6604 + \frac{5.319}{a}\right)Y\sqrt{L_W(\boldsymbol{s})} + (2a+5)Y^2.
\end{aligned}$$

Algorithm G1-norm.
Input $Y \geq 0$.

- For each $i = 0, 1, \ldots$
  - Let $k_i = z^i(aY)^2$.
  - Repeat
    1. Give $\boldsymbol{x}_t$ to $\mathbf{GD}_{G(k_i,Y,1)/||\boldsymbol{x}_t||^2}$.
    2. Get $\mathbf{GD}_{G(k_i,Y,1)/||\boldsymbol{x}_t||^2}$'s prediction h_t.
    3. Predict with
$$\hat{y}_t = \begin{cases} -Y & \text{if } h_t < -Y \\ h_t & \text{if } |h_t| \leq Y \\ Y & \text{otherwise.} \end{cases}$$
    4. Pass y_t to $\mathbf{GD}_{G(k_i,Y,1)/||\boldsymbol{x}_t||^2}$.

  until the total loss in this loop exceeds
$$k_i + 2Y\sqrt{k_i} + Y^2.$$

Figure 4.3: Pseudo-code for the algorithm **G1-norm**. (See Theorem 4.5.)

Finally, by letting $a = 2.0979$ to trade-off between the last two terms we obtain

$$\sum_{t=1}^{m}(\hat{y}_t - y_t)^2 \leq L_W(\boldsymbol{s}) + 9.2\left(Y\sqrt{L_W(\boldsymbol{s})} + Y^2\right).$$

□

Our next result is a corollary to Theorem 4.4 for the normalized loss. The algorithm **G1-norm** used in the proof is sketched in Figure 4.3. Since the normalized loss is used, there is no need to know a bound X on the norm of the $\boldsymbol{x}_t$'s.

Theorem 4.5: *For all $Y \geq 0$, the algorithm* **G1-norm** *has the following properties. Choose $m \in \boldsymbol{N}$, $\boldsymbol{s} = \langle(\boldsymbol{x}_t, y_t)\rangle_{t\leq m} \in (\mathcal{X} \times [-Y, Y])^m$. Let $\hat{y}_1, \ldots, \hat{y}_m$ be the sequence of* **G1-norm**$_Y$*'s on-line predictions for $\boldsymbol{s}$. Then*

$$\sum_{t=1}^{m}\frac{(\hat{y}_t - y_t)^2}{||\boldsymbol{x}_t||^2} \leq L'_Y(\boldsymbol{s}) + 9.2\left(Y\sqrt{L'_Y(\boldsymbol{s})} + Y^2\right).$$

Proof. Given Corollary 4.1, the proof follows from a straightforward adaptation to the normalized loss of the proof of Theorem 4.4. □

4.4 Predicting with no a priori information

In this section we remove all assumptions that the learner has prior knowledge. We introduce a new variant of the **GD** algorithm which we call **G2**. This new variant is described in Figure 4.4. A bound on **G2**'s total error follows quite straightforwardly from Theorem 4.1 via the application of standard doubling techniques.

Theorem 4.6: *For any $0 < \beta < 2$, the algorithm* $\mathbf{G2}_\beta$ *has the following properties. Choose $m \in \boldsymbol{N}, \boldsymbol{s} = \langle(\boldsymbol{x}_t, y_t)\rangle_{t\leq m} \in (\mathcal{X} \times \boldsymbol{R})^m$. Let $\hat{y}_1, \ldots, \hat{y}_m$ be the sequence of* $\mathbf{G2}_\beta$*'s on-line predictions for $\boldsymbol{s}$. Then,*

$$\sum_{t=1}^{m}(\hat{y}_t - y_t)^2 \leq \inf_{\boldsymbol{w}\in\mathcal{X}}\left[\frac{4(\max_t||\boldsymbol{x}_t||^2)||\boldsymbol{w}||^2}{\beta(2-\beta)c} + \frac{L_{\boldsymbol{w}}(\boldsymbol{s})}{(2-\beta)^2 c(1-c)}\right]$$

Algorithm G2.
Input $0 < \beta < 2$.

- Let $j = 0$.
- Let $X_1 = ||\boldsymbol{x}_1||$.
- On each trial t:
  1. Let $j \leftarrow \max\left\{j, \left\lceil \log_{\sqrt{2}} \frac{||\boldsymbol{x}_t||}{X_1} \right\rceil\right\}$.
  2. Give $\boldsymbol{x}_t$ to $\mathbf{GD}_{\beta/(2^{j/2}X_1)^2}$.
  3. Use $\mathbf{GD}_{\beta/(2^{j/2}X_1)^2}$'s prediction $\hat{y}_t$.
  4. Pass y_t to $\mathbf{GD}_{\beta/(2^{j/2}X_1)^2}$.

Figure 4.4: Pseudo-code for the algorithm **G2** that uses **GD** as a subroutine. (See Theorem 4.6.) The learning rate of **GD** is dynamically set depending on the relative sizes of the $\boldsymbol{x}_t$'s.

for all $0 < c \leq 1$. *In particular, if* $\beta = 4/3$ *and* $c = 1/2$,

$$\sum_{t=1}^{m} (\hat{y}_t - y_t)^2 \leq 9 \inf_{\boldsymbol{w} \in \mathcal{X}} \left[(\max_t ||\boldsymbol{x}_t||^2) ||\boldsymbol{w}||^2 + L_{\boldsymbol{w}}(\boldsymbol{s}) \right].$$

Proof: Choose $0 < \beta < 2$ and $0 < c \leq 1$. Notice that, in addition to a vector of hypothesized weights, **G2** maintains an integer j between trials. Before learning takes place, j is set to 0. After **G2** receives $\boldsymbol{x}_1$, it sets $X_1 = ||\boldsymbol{x}_1||$ and starts as a subroutine $\mathbf{GD}_{\beta/(X_1)^2}$. Thereafter, at each trial t, after **G2** receives $\boldsymbol{x}_t$, it sets

$$j \leftarrow \max\left\{j, \left\lceil \log_{\sqrt{2}} \frac{||\boldsymbol{x}_t||}{X_1} \right\rceil\right\}.$$

Then **G2** uses $\mathbf{GD}_{\beta/(2^{j/2}X_1)^2}$ for prediction on that trial.

Thus **G2** uses $\mathbf{GD}_{\beta/(X_1)^2}$ as long as the $\boldsymbol{x}_t$'s are smaller than X_1, at which time it switches over to $\mathbf{GD}_{\beta/(\sqrt{2}X_1)^2}$, which it uses as long as the $\boldsymbol{x}_t$'s are no bigger than $\sqrt{2}X_1$ (possibly for 0 trials), and continues in this manner, successively multiplying its assumed upper bound on the 2-norm of the $\boldsymbol{x}_t$'s by $\sqrt{2}$. Let $X = \max_t ||\boldsymbol{x}_t||$. It follows immediately from Theorem 4.1 that

$$\begin{aligned}
\sum_{t=1}^{m} (\hat{y}_t - y_t)^2 &\leq \left(\sum_{i=0}^{\lceil \log_{\sqrt{2}}(X/X_1) \rceil} \frac{||\boldsymbol{w}||^2 (2^{i/2}X_1)^2}{\beta(2-\beta)c} \right) + \frac{L_{\boldsymbol{w}}(\boldsymbol{s})}{(2-\beta)^2 c(1-c)} \\
&= \frac{||\boldsymbol{w}||^2 X_1^2}{\beta(2-\beta)c} \left(\sum_{i=0}^{\lceil \log_{\sqrt{2}}(X/X_1) \rceil} 2^i \right) + \frac{L_{\boldsymbol{w}}(\boldsymbol{s})}{(2-\beta)^2 c(1-c)} \\
&< \frac{||\boldsymbol{w}||^2 X_1^2}{\beta(2-\beta)c} 2^{2+2\log_2(X/X_1)} + \frac{L_{\boldsymbol{w}}(\boldsymbol{s})}{(2-\beta)^2 c(1-c)} \\
&= \frac{4||\boldsymbol{w}||^2 X^2}{\beta(2-\beta)c} + \frac{L_{\boldsymbol{w}}(\boldsymbol{s})}{(2-\beta)^2 c(1-c)}.
\end{aligned}$$

Plugging in $\beta = 4/3$ and $c = 1/2$ completes the proof. □

5 Application to classes of smooth functions

In this section, we describe applications of the inner product results of the previous section to arbitrary classes of smooth functions. While we will focus on applications of Theorem 4.3, we note that analogs of the other results of Section 4 can be obtained in a similar manner.

5.1 Smooth functions of a single variable

We begin with a class of smooth functions of a single real variable that was studied by Faber and Mycielski [FM91] in a similar context, except using the assumption that there was a function f in the class such that $y_t = f(x_t)$ for all t. Their methodology was to prove general results like those of the previous section under that assumption that there was a $\boldsymbol{w}$ with $f_{\boldsymbol{w}}(\boldsymbol{x}_t) = y_t$ for all t, then to reduce the smooth function learning problem to the more general problem as we do below. Similar function classes have also often been studied in nonparametric statistics (see, e.g. [Har91]) using probabilistic assumptions on the generation of the x_t's.

Let $\boldsymbol{R}_+$ be the set of nonnegative reals. We define the set SMO_W to be all absolutely continuous $f : \boldsymbol{R}_+ \to \boldsymbol{R}$ for which

1. $f(0) = 0$
2. $\sqrt{\int_0^\infty f'(z)^2\, dz} \leq W$.

The assumption that $f(0) = 0$ will be satisfied by many natural functions of interest. Examples include distance traveled as a function of time and return as a function of investment. We will prove the following result about SMO_W.

Theorem 5.1: *For each $E, X, W \geq 0$, there is a prediction algorithm $\mathbf{A}_{\mathrm{SMO}}$ with the following properties.*
Choose $m \in \boldsymbol{N}$, $\boldsymbol{s} = \langle (x_t, y_t) \rangle_{t \leq m} \in ([0, X] \times \boldsymbol{R})^m$, such that there is an $f \in \mathrm{SMO}_W$ for which $\sum_{t=1}^m (f(x_t) - y_t)^2 \leq E$. Let $\hat{y}_1, \ldots, \hat{y}_m$ be the sequence of $\mathbf{A}_{\mathrm{SMO}}$'s on-line predictions for $\boldsymbol{s}$. Then,

$$\sum_{t=1}^m (\hat{y}_t - y_t)^2 \leq \inf_{f \in \mathrm{SMO}_W} \left[\sum_{t=1}^m (f(x_t) - y_t)^2 \right] + 2W\sqrt{XE} + W^2 X.$$

Proof: For now, let us ignore computational issues. We'll treat them again after the proof.

Fix $E, X, W \geq 0$. The algorithm $\mathbf{A}_{\mathrm{SMO}}$ operates by reducing the problem of learning SMO_W to a more general problem of the type treated in the previous section.

Let $L^2(\boldsymbol{R}_+)$ be the space of (measurable) functions g from $\boldsymbol{R}_+$ to $\boldsymbol{R}$ for which $\int_0^\infty g(u)^2\, du$ is finite. $L^2(\boldsymbol{R}_+)$ is well known to be an inner product space (see, e.g. [You88]), with the inner product defined by

$$(g_1, g_2) = \int_0^\infty g_1(u) g_2(u)\, du.$$

Further, we define $g_3 = g_2 + g_1$ by

$$(\forall x) \quad g_3(x) = g_2(x) + g_1(x),$$

and $g_3 = \kappa g_1$ by

$$(\forall x) \quad g_3(x) = \kappa g_1(x).$$

Algorithm $\mathbf{A}_{\mathrm{SMO}}$.
Input: $E, W, X \geq 0$.

- On each trial t:
  1. Get $x_t \in [0, X]$ from the environment.
  2. Give $\chi_{\leq x_t} \in L^2(\boldsymbol{R}_+)$ to $\mathbf{GD}_{G(E,W,X)/X^2}$.
  3. Use $\mathbf{GD}_{G(E,W,X)/X^2}$'s prediction $\hat{y}_t$.
  4. Pass y_t to $\mathbf{GD}_{G(E,W,X)/X^2}$.

Figure 5.1: Pseudo-code for algorithm $\mathbf{A}_{\mathrm{SMO}}$. (See Theorem 5.1.) Algorithm **GD** (here used as a subroutine) is applied to the inner product space $\mathcal{X} = L^2(\boldsymbol{R}_+)$. The function G, used to set **GD**'s learning rate, is defined in Section 4.2.

Now apply algorithm **GD** to this particular inner product space, $L^2(\boldsymbol{R}_+)$, with learning rate η set to $G(E, W, X)$, where the function G is defined in Section 4.2. For any $x \geq 0$, define $\chi_{\leq x} : \boldsymbol{R}_+ \to \boldsymbol{R}$ by

$$\chi_{\leq x}(u) = \begin{cases} 1 & \text{if } u \leq x \\ 0 & \text{otherwise.} \end{cases}$$

Note that for any $x \leq X$

$$||\chi_{\leq x}|| = \sqrt{\int_0^\infty \chi_{\leq x}(u)^2 \, du} = \sqrt{x} \leq \sqrt{X}, \tag{5.1}$$

and therefore $\chi_{\leq x} \in L^2(\boldsymbol{R}_+)$.

In Figure 5.1, we give a short description of the algorithm $\mathbf{A}_{\mathrm{SMO}}$. Note that for any $f \in \mathrm{SMO}_W$,

$$||f'|| = \sqrt{\int_0^\infty f'(u)^2 \, du} \leq W. \tag{5.2}$$

Finally, note that since $f(0) = 0$,

$$(f', \chi_{\leq x}) = \int_0^\infty f'(u)\chi_{\leq x}(u) \, du = \int_0^x f'(u) \, du = f(x) - f(0) = f(x). \tag{5.3}$$

Thus, if there is an $f \in \mathrm{SMO}_W$ for which $\sum_{t=1}^m (f(x_t) - y_t)^2 \leq E$, then $f' \in L^2(\boldsymbol{R}_+)$ has $||f'|| \leq W$ and satisfies

$$\sum_{t=1}^m ((f', \chi_{\leq x_t}) - y_t)^2 \leq E.$$

Combining this with (5.1) and Theorem 4.3, we can see that **GD**'s predictions satisfy

$$\sum_{t=1}^m (\hat{y}_t - y_t)^2 \leq \inf_{||f'|| \leq W} \left[\sum_{t=1}^m ((f', \chi_{\leq x_t}) - y_t)^2 \right] + 2W\sqrt{XE} + W^2 X.$$

The result then follows from the fact that $\mathbf{A}_{\mathrm{SMO}}$ just makes the same predictions as **GD**. □

By closely examining the predictions of the algorithm $\mathbf{A}_{\mathrm{SMO}}$ of Theorem 5.1, we can see that it can be implemented in time polynomial in t. The algorithm **GD** maintains a function $\hat{\boldsymbol{w}} \in L^2(\boldsymbol{R}_+)$ which it updates between trials. As before, let $\hat{\boldsymbol{w}}_t$ be the tth hypothesis of

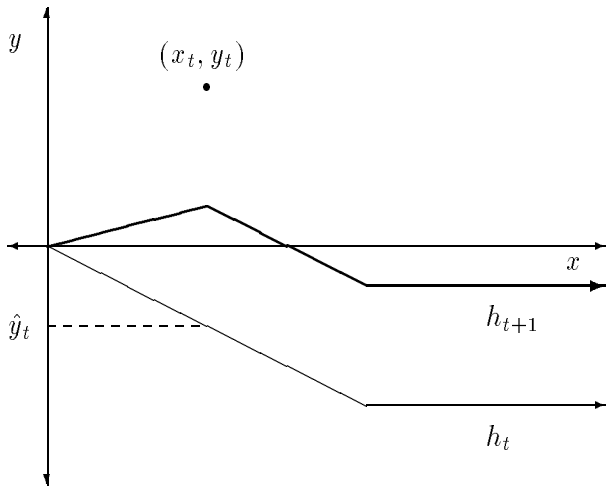

Figure 5.2: An example of the update of the application of the **GD** algorithm to smoothing in the single-variable case. The derivative of the hypothesis is modified by a constant in the appropriate direction to the left of x_t, and left unchanged to the right.

GD. We can see that $\hat{\boldsymbol{w}}_t$ can be interpreted as the derivative of $\mathbf{A}_{\mathrm{SMO}}$'s tth hypothesis. This is because **GD**'s tth prediction, and therefore $\mathbf{A}_{\mathrm{SMO}}$'s tth prediction, is

$$(\hat{\boldsymbol{w}}_t, \chi_{\leq x_t}) = \int_0^\infty \hat{\boldsymbol{w}}_t(u)\chi_{\leq x_t}(u)\,du = \int_0^{x_t} \hat{\boldsymbol{w}}_t(u)\,du.$$

Hence $\mathbf{A}_{\mathrm{SMO}}$'s tth hypothesis h_t satisfies $h_t' = \hat{\boldsymbol{w}}_t$.

GD sets $\hat{\boldsymbol{w}}_1$ to be the constant 0 function, and its update is

$$\hat{\boldsymbol{w}}_{t+1} = \hat{\boldsymbol{w}}_t + \eta(y_t - \hat{y}_t)\chi_{\leq x_t},$$

where η doesn't depend on t (see the proof of Theorem 4.3). Integrating yields the following expression for $\mathbf{A}_{\mathrm{SMO}}$'s $t+1$st hypothesis:

$$h_{t+1}(x) = \begin{cases} h_t(x) + \eta(y_t - \hat{y}_t)x & \text{if } x \leq x_t \\ h_t(x) + \eta(y_t - \hat{y}_t)x_t & \text{otherwise} \end{cases}$$

and therefore

$$h_{t+1}(x) = h_t(x) + \eta(y_t - \hat{y}_t)\min\{x_t, x\}.$$

By induction, we have

$$h_{t+1}(x) = \eta\sum_{s\leq t}(y_s - \hat{y}_s)\min\{x_s, x\},$$

trivially computable in $O(t)$ time if the previous $\hat{y}_s$'s are saved. This algorithm is illustrated in Figure 5.2.

5.2 Smooth functions of several variables

Theorem 5.1 can be generalized to higher dimensions as follows. The analogous generalization in the absence of noise was carried out in [FM91]. The domain X is $\boldsymbol{R}_+^n$. We define the set $\mathrm{SMO}_{W,n}$ to be all functions $f : \boldsymbol{R}_+^n \to \boldsymbol{R}$ for which there is a function $\tilde{f}$ such that

1. $\forall \boldsymbol{x} \in \boldsymbol{R}^n \; f(\boldsymbol{x}) = \int_0^{x_1} \ldots \int_0^{x_n} \tilde{f}(u_1, \ldots, u_n)\, du_n \ldots du_1$
2. $\sqrt{\int_0^{\infty} \ldots \int_0^{\infty} (\tilde{f}(u_1, \ldots, u_n))^2\, du_n \ldots du_1} \leq W$.

It is easily verified that when $\tilde{f}$ exists, it is defined by

$$\tilde{f}(u_1, \ldots, u_n) = \frac{\partial^n f(u_1, \ldots, u_n)}{\partial u_1 \ldots \partial u_n}.$$

We can establish the following generalization of Theorem 5.1.

Theorem 5.2: *For each $E, X, W \geq 0$ and $n \in \boldsymbol{N}$, there is a prediction algorithm $\mathbf{A}_{\mathrm{SMOn}}$ with the following properties.*
Choose $m \in \boldsymbol{N}$, $\boldsymbol{s} = \langle (\boldsymbol{x}_t, y_t) \rangle_{t \leq m} \in ([0, X]^n \times \boldsymbol{R})^m$, such that there is an $f \in \mathrm{SMO}_{W,n}$ for which $\sum_{t=1}^m (f(\boldsymbol{x}_t) - y_t)^2 \leq E$. Let $\hat{y}_1, \ldots, \hat{y}_m$ be the sequence of $\mathbf{A}_{\mathrm{SMOn}}$'s on-line predictions for $\boldsymbol{s}$. Then,

$$\sum_{t=1}^m (\hat{y}_t - y_t)^2 \leq \inf_{f \in \mathrm{SMO}_{W,n}} \left[\sum_{t=1}^m (f(\boldsymbol{x}_t) - y_t)^2 \right] + 2WX^{n/2}\sqrt{E} + W^2 X^n.$$

Proof. Fix $E, X, W, n \geq 0$. The algorithm $\mathbf{A}_{\mathrm{SMOn}}$ operates by reducing the problem of learning $\mathrm{SMO}_{W,n}$ to a more general problem of the type treated in the previous section.

Let $L^2(\boldsymbol{R}_+^n)$ be the space of (measurable) functions g from $\boldsymbol{R}_+^n$ to $\boldsymbol{R}$ for which

$$\int_0^{\infty} \ldots \int_0^{\infty} g(\boldsymbol{x})^2\, dx_n \ldots dx_1$$

is finite. Again, it is well known (see e.g. [You88]), that $L^2(\boldsymbol{R}_+^n)$ has an inner product defined by

$$(g_1, g_2) = \int_0^{\infty} \ldots \int_0^{\infty} g_1(\boldsymbol{x}) g_2(\boldsymbol{x})\, dx_n \ldots dx_1.$$

Now apply algorithm **GD** to this particular inner product space, $L^2(\boldsymbol{R}_+^n)$, with learning rate η set to $G(E, W, X)$, where the function G is defined in Section 4.2. For any $\boldsymbol{x} \in \boldsymbol{R}_+^n$, define $\chi_{\leq \boldsymbol{x}} : \boldsymbol{R}_+^n \to \boldsymbol{R}$ as the indicator function of the rectangle $[0, x_1] \times \ldots \times [0, x_n]$. Note that for any $\boldsymbol{x} \in [0, X]^n$

$$||\chi_{\leq \boldsymbol{x}}|| = \sqrt{\int_0^{\infty} \ldots \int_0^{\infty} \chi_{\leq \boldsymbol{x}}(\boldsymbol{u})^2\, du_n \ldots du_1} = \sqrt{\prod_{i=1}^n x_i} \leq X^{n/2}. \tag{5.4}$$

and therefore $\chi_{\leq \boldsymbol{x}} \in L^2(\boldsymbol{R}_+^n)$.

The algorithm $\mathbf{A}_{\mathrm{SMOn}}$ is sketched in Figure 5.3. Note that for any $f \in \mathrm{SMO}_{W,n}$, there is a function $\tilde{f}$ such that

$$(\tilde{f}, \chi_{\leq \boldsymbol{x}_t}) = \int_0^{\infty} \int_0^{\infty} \tilde{f}(x_1, \ldots, x_n) \chi_{\leq \boldsymbol{x}_t}(x_1, \ldots, x_n)\, dx_n \ldots dx_1 = f(\boldsymbol{x}_t).$$

Thus, if there is an $f \in \mathrm{SMO}_{W,n}$ for which $\sum_{t=1}^m (f(\boldsymbol{x}) - y_t)^2 \leq E$, then the corresponding $\tilde{f} \in L^2(\boldsymbol{R}_+)$, which has $||\tilde{f}|| \leq W$, satisfies $\sum_{t=1}^m ((\tilde{f}, \chi_{\leq \boldsymbol{x}_t}) - y_t)^2 \leq E$. Combining this with (5.4) and Theorem 4.3, we can see that **GD**'s predictions satisfy

$$\sum_{t=1}^m (\hat{y}_t - y_t)^2 \leq \inf_{||\tilde{f}|| \leq W} \left[\sum_{t=1}^m ((\tilde{f}, \chi_{\leq \boldsymbol{x}_t}) - y_t)^2 \right] + 2WX^{n/2}\sqrt{E} + W^2 X^n.$$

The result then follows from the fact that $\mathbf{A}_{\mathrm{SMOn}}$ just makes the same predictions as **GD**. □

It is easy to see, by extending the discussion following Theorem 5.1, that the predictions of Theorem 5.2 can be computed in $O(tn)$ time, if previous predictions are saved.

Algorithm $\mathbf{A}_{\mathrm{SMOn}}$.
Input: $E, W, X \geq 0$.

- On each trial t:
  1. Get $\boldsymbol{x}_t \in [0, X]^n$ from the environment.
  2. Give $\chi_{\leq \boldsymbol{x}_t} \in L^2(\boldsymbol{R}_+^n)$ to $\mathbf{GD}_{G(E,W,X)/X^2}$.
  3. Use $\mathbf{GD}_{G(E,W,X)/X^2}$'s prediction $\hat{y}_t$.
  4. Pass y_t to $\mathbf{GD}_{G(E,W,X)/X^2}$.

Figure 5.3: Pseudo-code for algorithm $\mathbf{A}_{\mathrm{SMOn}}$. (See Theorem 5.2.) Algorithm **GD** (here used as a subroutine) is applied to the inner product space $\mathcal{X} = L^2(\boldsymbol{R}_+^n)$. The function G, used to set **GD**'s learning rate, is defined in Section 4.2.

6 A comparison to standard gradient descent methods

The goal of this section is to compare the total square loss bounds obtained via our analysis to the bounds obtained via the standard analysis of gradient descent methods. Standard methods only deal with the case when all the pairs (x_t, y_t) are given at once (batch case) rather than in an on-line fashion. Thus we consider the problem of finding the solution $\boldsymbol{x} \in \boldsymbol{R}^n$ of a system of linear equations

$$\begin{aligned} a_{1,1}x_1 + a_{1,2}x_2 + \quad \ldots \quad + a_{1,n}x_n &= b_1 \\ &\vdots \\ a_{m,1}x_1 + a_{m,2}x_2 + \quad \ldots \quad + a_{m,n}x_n &= b_m \end{aligned}$$

where $a_{i,j}, b_i \in \boldsymbol{R}$. The above system can be given the more compact representation $A\boldsymbol{x} = \boldsymbol{b}$, where $\boldsymbol{b} = (b_1, \ldots, b_m)$ and A is a $m \times n$ matrix with entries $a_{i,j}$. ($A\boldsymbol{x}$ denotes the usual matrix-vector product.) For simplicity, we assume in this section that $A\boldsymbol{x} = \boldsymbol{b}$ has a solution. However, we do not assume that the matrix A has any special property.

A standard iterative approach for solving the problem $A\boldsymbol{x} = \boldsymbol{b}$ is to perform gradient descent over the squared residual error $R(\boldsymbol{x}) = ||A\hat{\boldsymbol{x}} - \boldsymbol{b}||_2^2$, where $\hat{\boldsymbol{x}}$ is a candidate solution. We will prove upper bounds on the sum of $R(\hat{\boldsymbol{x}}_t)$ for the sequence $\hat{\boldsymbol{x}}_1, \hat{\boldsymbol{x}}_2, \ldots$ of candidate solutions generated by the gradient descent method tuned either according to the standard analysis or to our analysis. The bound are expressed in terms of both the norm of the solution $\boldsymbol{x}$ and the eigenvalues of $A^T A$, where A^T denotes the transpose matrix of A.

We define the norm $||A||$ of a matrix A by

$$||A||_2 = \sup_{||\boldsymbol{v}||_2 = 1} ||A\boldsymbol{v}||_2.$$

This is the norm induced by the Euclidean norm for vectors in $\boldsymbol{R}^n$ (see [GL89].) Notice that $||A\boldsymbol{v}||_2 \leq ||A||_2 ||\boldsymbol{v}||_2$ (Cauchy-Schwartz inequality). We will make use of the following well-known facts.

Fact 6.1 ([HJ85]): *For any real matrix A, $||A||_2 = \sqrt{\lambda_{max}}$, where λ_{max} is the largest eigenvalue of $A^T A$.*

Fact 6.2 ([HJ85]): *For any real matrix A,*

$$||A^T||_2 = ||A||_2.$$

Given a candidate solution $\hat{\boldsymbol{x}} \in \boldsymbol{R}^n$ with squared residual error $R(\hat{\boldsymbol{x}})$, the gradient of $R(\hat{\boldsymbol{x}})$ with respect to $\hat{\boldsymbol{x}}$ is $\vec{\nabla} R(\hat{\boldsymbol{x}}) = 2A^T(A\hat{\boldsymbol{x}} - \boldsymbol{b})$. By applying the gradient descent (Kaczmarz) rule for the batch case we derive the update

$$\hat{\boldsymbol{x}}_{t+1} = \hat{\boldsymbol{x}}_t - \eta 2A^T(A\hat{\boldsymbol{x}} - \boldsymbol{b}) \tag{6.1}$$

for some scaling factor $\eta > 0$. Simple manipulation shows that

$$R(\hat{\boldsymbol{x}}_{t+1}) = R(\hat{\boldsymbol{x}}_t) + \eta^2 ||A\vec{\nabla} R(\hat{\boldsymbol{x}}_t)||_2^2 - \eta ||\vec{\nabla} R(\hat{\boldsymbol{x}}_t)||_2^2. \tag{6.2}$$

Following the standard analysis of gradient descent, we find the value of η minimizing the LHS of (6.2) at

$$\eta_1 = \frac{||\vec{\nabla} R(\hat{\boldsymbol{x}}_t)||_2^2}{2||A\vec{\nabla} R(\hat{\boldsymbol{x}}_t)||_2^2}.$$

By plugging this optimal value of η back in (6.2) we get

$$R(\hat{\boldsymbol{x}}_{t+1}) = R(\hat{\boldsymbol{x}}_t) - \frac{||\vec{\nabla} R(\hat{\boldsymbol{x}}_t)||_2^4}{4||A\vec{\nabla} R(\hat{\boldsymbol{x}}_t)||_2^2}.$$

Proposition 6.1: *For all $m, n > 0$, for any $m \times n$ real matrix A and for any vector $\boldsymbol{x} \in \boldsymbol{R}^n$. Let $\boldsymbol{b} = A\boldsymbol{x}$ and let $\lambda_{min}, \lambda_{max}$ be, respectively, the smallest and the largest eigenvalues of A^TA. Then, if $\hat{\boldsymbol{x}}_0 = 0$ and $\hat{\boldsymbol{x}}_{t+1}$ is computed from $\hat{\boldsymbol{x}}_t$ using formula (6.1) with $\eta = \eta_1$,*

$$\sum_{t=0}^{\infty} ||A\hat{\boldsymbol{x}}_t - \boldsymbol{b}||_2^2 \leq \frac{(\lambda_{min} + \lambda_{max})^2}{4\lambda_{min}} ||\boldsymbol{x}||_2^2.$$

Proof. If $\lambda_{min} = 0$, then the bound holds vacuously. Assume then $\lambda_{min} > 0$. Via an application of the Kantorovich inequality to the square matrix A^TA (see e.g. [Lue84]) it can be shown that

$$R(\hat{\boldsymbol{x}}_{t+1}) \leq \left(1 - \frac{4\lambda_{min}\lambda_{max}}{(\lambda_{min} + \lambda_{max})^2}\right) R(\hat{\boldsymbol{x}}_t). \tag{6.3}$$

Therefore, we get

$$\frac{4\lambda_{min}\lambda_{max}}{(\lambda_{min} + \lambda_{max})^2} R(\hat{\boldsymbol{x}}_t) \leq R(\hat{\boldsymbol{x}}_t) - R(\hat{\boldsymbol{x}}_{t+1}).$$

By summing up over all iterations t we obtain

$$\frac{4\lambda_{min}\lambda_{max}}{(\lambda_{min} + \lambda_{max})^2} \sum_{t=0}^{\infty} R(\hat{\boldsymbol{x}}_t) \leq R(\hat{\boldsymbol{x}}_0).$$

Recalling that $\hat{\boldsymbol{x}}_0 = (0, \ldots, 0)$ and making use of Fact 6.1,

$$\begin{aligned}
\sum_{t=0}^{\infty} ||A\hat{\boldsymbol{x}}_t - \boldsymbol{b}||_2^2 &\leq \frac{(\lambda_{min} + \lambda_{max})^2}{4\lambda_{min}\lambda_{max}} R(\hat{\boldsymbol{x}}_0) \\
&\leq \frac{(\lambda_{min} + \lambda_{max})^2}{4\lambda_{min}\lambda_{max}} ||A\boldsymbol{x}||_2^2 \\
&\leq \frac{(\lambda_{min} + \lambda_{max})^2}{4\lambda_{min}\lambda_{max}} ||A||_2^2 ||\boldsymbol{x}||_2^2 \\
&\leq \frac{(\lambda_{min} + \lambda_{max})^2}{4\lambda_{min}\lambda_{max}} \lambda_{max} ||\boldsymbol{x}||_2^2 \\
&= \frac{(\lambda_{min} + \lambda_{max})^2}{4\lambda_{min}} ||\boldsymbol{x}||_2^2
\end{aligned}$$

concluding the proof. □

A different analysis of update (6.1) can be obtained by applying the techniques developed in Section 4. Let $D(\hat{\boldsymbol{x}})$ be the distance $||\hat{\boldsymbol{x}} - \boldsymbol{x}||_2^2$ of $\hat{\boldsymbol{x}}$ to the solution $\boldsymbol{x}$. An easy adaptation of Lemma 4.1 shows that

$$D(\hat{\boldsymbol{x}}_{t+1}) = D(\hat{\boldsymbol{x}}_t) + \eta^2 ||\vec{\nabla} R(\hat{\boldsymbol{x}}_t)||_2^2 - 4\eta R(\hat{\boldsymbol{x}}_t). \tag{6.4}$$

Here, the minimization over η yields the optimimum at

$$\eta_2 = \frac{2R(\hat{\boldsymbol{x}}_t)}{||\vec{\nabla} R(\hat{\boldsymbol{x}}_t)||_2^2}.$$

We then have the following result.

Proposition 6.2: *For all $m, n > 0$, for any $m \times n$ real matrix A and for any vector $\boldsymbol{x} \in \boldsymbol{R}^n$. Let $\boldsymbol{b} = A\boldsymbol{x}$ and let λ_{max} be the largest eigenvalue of $A^T A$. Then, if $\hat{\boldsymbol{x}}_0 = 0$ and $\hat{\boldsymbol{x}}_{t+1}$ is computed from $\hat{\boldsymbol{x}}_t$ using formula (6.1) with $\eta = \eta_2$,*

$$\sum_{t=0}^{\infty} ||A\hat{\boldsymbol{x}}_t - \boldsymbol{b}||_2^2 \leq \lambda_{max} ||\boldsymbol{x}||_2^2.$$

Proof. By plugging η_2 for η in (6.4) we obtain

$$\begin{aligned}
D(\hat{\boldsymbol{x}}_{t+1}) &= D(\hat{\boldsymbol{x}}_t) - \frac{4R(\hat{\boldsymbol{x}}_t)^2}{||\vec{\nabla} R(\hat{\boldsymbol{x}}_t)||_2^2} \\
&= D(\hat{\boldsymbol{x}}_t) - ||A\hat{\boldsymbol{x}}_t - \boldsymbol{b}||_2^2 \frac{||A\hat{\boldsymbol{x}}_t - \boldsymbol{b}||_2^2}{||A^T(A\hat{\boldsymbol{x}}_t - \boldsymbol{b})||_2^2} \\
&\leq D(\hat{\boldsymbol{x}}_t) - \frac{||A\hat{\boldsymbol{x}}_t - \boldsymbol{b}||_2^2}{||A^T||_2^2} \qquad \text{by definition of } ||A^T||_2 \\
&\leq D(\hat{\boldsymbol{x}}_t) - \frac{||A\hat{\boldsymbol{x}}_t - \boldsymbol{b}||_2^2}{||A||_2^2} \qquad \text{by Fact 6.2.}
\end{aligned}$$

Therefore, rearranging the above and summing up over all iterations t,

$$\begin{aligned}
\sum_{t=0}^{\infty} ||A\hat{\boldsymbol{x}}_t - \boldsymbol{b}||_2^2 &\leq ||A||_2^2 D(\hat{\boldsymbol{x}}_0) \\
&= ||A||_2^2 ||\boldsymbol{x}||_2^2
\end{aligned}$$

since $\hat{\boldsymbol{x}}_0 = (0, \ldots, 0)$. By Fact 6.1, this implies

$$\sum_{t=0}^{\infty} ||A\hat{\boldsymbol{x}}_t - \boldsymbol{b}||_2^2 \leq \lambda_{max} ||\boldsymbol{x}||_2^2.$$

□

In summary, we compared two tunings of η for the learning rule (6.1). The first and standard one maximizes the decrease of $||A\hat{\boldsymbol{x}} - \boldsymbol{b}||_2^2$ and the second one maximizes the decrease in $||\hat{\boldsymbol{x}} - \boldsymbol{x}||_2^2$, where $\boldsymbol{x}$ is a solution.

The first method has the advantage that one can show that $||A\hat{\boldsymbol{x}} - \boldsymbol{b}||_2^2$ decreases by a fixed factor in each trial (Inequality (6.3)). (Note that this factor is 1 when $\lambda_{min} = 0$, and this holds when A does not have full rank.) In contrast, matrices A can be constructed where updating with the optimal learning rate η_2 causes an increase in $||A\hat{\boldsymbol{x}} - \boldsymbol{b}||_2^2$.

The second method, however, always leads to better bounds on $\sum_t ||A\hat{\boldsymbol{x}}_t - \boldsymbol{b}||_2^2$ since

$$\lambda_{max} \leq \frac{(\lambda_{min} + \lambda_{max})^2}{4\lambda_{min}}$$

for all $\lambda_{min}, \lambda_{max} \geq 0$. (Notice that the corresponding bound for the first method is vacuous when $\lambda_{min} = 0$, which holds, as we said above, when A does not have full rank.)

7 Lower bounds

In this section, we describe lower bounds which match the upper bounds of Theorems 4.3, 5.1, and 5.2, constants included. In fact, these lower bounds show that even the upper bound on the excess of the algorithm's squared loss above the best fixed element within a given class of functions is optimal.

Theorem 7.1: *Fix an inner product space $\mathcal{X}$ for which an orthonormal basis can be found.*[6] *For all $E, X, W \geq 0$ and all prediction algorithm $\mathbf{A}$, there exists $n \in \boldsymbol{N}$ and a pair $(\boldsymbol{x}, y) \in \mathcal{X} \times \boldsymbol{R}$, such that $||\boldsymbol{x}|| \leq X$ and the following hold: There is a $\boldsymbol{w} \in \mathcal{X}$ for which $||\boldsymbol{w}|| = W$ and $((\boldsymbol{w}, \boldsymbol{x}) - y)^2 = E$. Furthermore, if $\hat{y} = A(\boldsymbol{x}_t)$ then*

$$(\hat{y} - y)^2 \geq E + 2WX\sqrt{E} + (WX)^2.$$

Proof. Choose an orthonormal basis for $\mathcal{X}$. Set $\boldsymbol{x} = (X, 0, \ldots)$, $y = \mathrm{sgn}(-\hat{y})(WX + \sqrt{E})$, and $\boldsymbol{w} = (\mathrm{sgn}(-\hat{y})W, 0, \ldots)$. The result then follows easily. □

To establish the upper bound of Theorem 4.4, in which general bounds were obtained without any knowledge of an upper bound on $L_W(\boldsymbol{s})$, we required the assumption that the y_t's were in a known range $[-Y, Y]$ and compared the total loss of the **GD** algorithm on $\boldsymbol{s}$ against $L_W(\boldsymbol{s})$, where $W = Y/(\max_t ||\boldsymbol{x}_t||)$. Therefore, the above lower bound does not say anything about the optimality of those results. The following lower bound shows that Theorem 4.4 cannot be significantly improved in general. It further has obvious consequences concerning the finite dimension case when the "noise level" E is not too large relative to the number n of variables as well as X and Y.

Theorem 7.2: *Let $\langle \mathcal{X}_d \rangle_{d \in \boldsymbol{N}}$ be any sequence of inner product spaces such that $\mathcal{X}_d$ is a d-dimensional vector space. Choose $X, Y, E > 0$. Let n be any integer such that*

$$n \geq \left(1 + \frac{\sqrt{E}}{Y}\right)^2. \tag{7.1}$$

Then for any prediction algorithm A there is a sequence $\langle (\boldsymbol{x}_1, y_1) \rangle_{t \leq n} \in (\mathcal{X}_n \times [-Y, Y])^n$ such that

1. *For all $1 \leq t \leq n$, $\|\boldsymbol{x}_t\| = X$.*
2. *If for each t, $\hat{y}_t = A(((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_{t-1}, y_{t-1})), \boldsymbol{x}_t)$, then*

$$\sum_{t=1}^{n} (y_t - \hat{y}_t)^2 \geq (Y + \sqrt{E})^2 = E + 2Y\sqrt{E} + Y^2.$$

[6] An orthonormal basis can be found under quite general conditions. See e.g. [You88] for details.

3. There exists $\boldsymbol{w} \in \boldsymbol{R}^n$ *such that* $\|\boldsymbol{w}\| = Y/X$ *and*

$$\sum_{t=1}^{n} (y_t - (\boldsymbol{w}, \boldsymbol{x}_t))^2 = E.$$

Proof. Choose $X, Y, E > 0$ and choose $n \in \boldsymbol{N}$ so that (7.1) is satisfied. Let $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n$ be an orthonormal basis of $\mathcal{X}_n$ (since $\mathcal{X}_n$ is a finite-dimensional inner product space, such an orthonormal basis can always be found). Let $\boldsymbol{x}_i = X\boldsymbol{e}_i$, for $i = 1, \ldots, n$. Since the basis is orthonormal, $\|\boldsymbol{x}_i\| = X$ for all i, fulfilling part 1. Consider the adversary which at each step $t = 1, \ldots, n$ feeds the algorithm with vector $\boldsymbol{x}_t$ and, upon algorithm's prediction $\hat{y}_t$, responds with

$$y_t := \operatorname{sgn}(-\hat{y}_t)\frac{Y + \sqrt{E}}{\sqrt{n}}.$$

This implies

$$(y_t - \hat{y}_t)^2 \geq \left(\frac{Y + \sqrt{E}}{\sqrt{n}}\right)^2$$

for all $t = 1, 2, \ldots, n$. This proves part 2. Now let $\boldsymbol{w}$ be the vector of $\mathcal{X}_n$ with coordinates

$$\left(\operatorname{sgn}(-\hat{y}_1)\frac{Y/X}{\sqrt{n}}, \ldots, \operatorname{sgn}(-\hat{y}_n)\frac{Y/X}{\sqrt{n}}\right)$$

with respect to the basis $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n$. To prove part 3, first notice that $\|\boldsymbol{w}\| = Y/X$. Second, for each $t = 1, \ldots, n$ we have

$$\begin{aligned}
(y_t - (\boldsymbol{x}_t, \boldsymbol{w}))^2 &= \left[\operatorname{sgn}(-\hat{y}_t)\frac{Y + \sqrt{E}}{\sqrt{n}} - (\boldsymbol{x}_t, \boldsymbol{w})\right]^2 \\
&= \left[\operatorname{sgn}(-\hat{y}_t)\frac{Y + \sqrt{E}}{\sqrt{n}} - X(\boldsymbol{e}_t, \boldsymbol{w})\right]^2 \\
&= \left[\operatorname{sgn}(-\hat{y}_t)\frac{Y + \sqrt{E}}{\sqrt{n}} - X\left(\operatorname{sgn}(-\hat{y}_t)\frac{Y/X}{\sqrt{n}}\right)\right]^2 \\
&= \left(\frac{Y + \sqrt{E}}{\sqrt{n}} - \frac{Y}{\sqrt{n}}\right)^2 \\
&= \frac{E}{n}.
\end{aligned}$$

This concludes the proof of part 3. Finally, notice that (7.1) implies that for all $t = 1, 2, \ldots, n$

$$|y_t| = \frac{Y + \sqrt{E}}{\sqrt{n}} \leq Y.$$

The proof is complete. □

We conclude with a lower bound for smooth functions.

Theorem 7.3: *Choose* $E, X, W \geq 0$, $n \in \boldsymbol{N}$, *and a prediction algorithm* $\mathbf{A}$. *Then there exists* $m \in \boldsymbol{N}$, $\boldsymbol{s} = \langle(\boldsymbol{x}_t, y_t)\rangle_{t \leq m} \in ([0, X]^n \times \boldsymbol{R})^m$, *such that the following hold: There is a function* $f \in \mathrm{SMO}_{W,n}$ *for which* $\sum_{t=1}^{m}(f(\boldsymbol{x}_t) - y_t)^2 \leq E$. *If for each* t,

$$\hat{y}_t = A(((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_{t-1}, y_{t-1})), \boldsymbol{x}_t),$$

then

$$\sum_{t=1}^{m}(\hat{y}_t - y_t)^2 \geq E + 2WX^{n/2}\sqrt{E} + 2W^2X^n$$

Proof. In fact $m = 1$ suffices in this case. Let $\boldsymbol{x}_1 = (X, \ldots, X)$. Suppose the first prediction $\hat{y}_1$ of $\mathbf{A}$ is nonpositive. Let

$$y_1 = WX^{n/2} + \sqrt{E}$$

and let the function $f : \boldsymbol{R}_+^n \to \boldsymbol{R}$ be defined by

$$f(\boldsymbol{x}) = \frac{W}{X^{n/2}} \prod_{i=1}^{n} x_i,$$

if $\boldsymbol{x} \in [0, X]^n$, and $f(\boldsymbol{x}) = 0$ otherwise. Then, for any $\boldsymbol{x} \in [0, X]^n$,

$$f(\boldsymbol{x}) = \int_0^{x_1} \ldots \int_0^{x_n} \tilde{f}(u_1, \ldots, u_n)\, du_n \ldots du_1,$$

where $\tilde{f} \equiv \frac{W}{X^{n/2}}$. The following are then easily verified

1. $f(0) = 0$
2. $(f(x_1) - y_1)^2 = (WX^{n/2} - (WX^{n/2} + \sqrt{E}))^2 = E$
3. $\sqrt{\int_0^\infty \ldots \int_0^\infty \tilde{f}(\boldsymbol{u})^2\, du_n \ldots du_1} = \sqrt{X^n(c/X^{n/2})^2} = W$
4. $(\hat{y}_1 - y_1)^2 \geq (WX^{n/2} + \sqrt{E})^2 = E + 2WX^{n/2}\sqrt{E} + W^2X^n$

since $\hat{y}_1 \leq 0$. The case in which $\hat{y}_1 > 0$ can be handled symmetrically. □

8 Discussion and conclusions

In this paper we have investigated the performance of the gradient descent rule applied to the problem of on-line prediction in arbitrary inner product spaces. Through a reduction, we then applied our results to natural classes of smooth functions.

One of the most interesting contributions of this work is perhaps the derivation of the optimal "learning rate" for gradient descent methods when the goal is to minimize the worst-case sum of squared errors. Our tuning of the learning rate is based on *a priori* information that can be guessed on-line with an increase in the total loss of constant factors only. In the case of iterative solution of systems of linear equations, we also showed that, with respect to the sum of squared errors, the tuning provided by our analysis compares favorably against the tuning obtained via the standard gradient descent analysis.

It is an open problem whether, instead of using adversarial arguments as we do here, our lower bounds can already be obtained when the examples are randomly and independently drawn from a natural distribution. For more simple functions this was done in [CFH+93]: the lower bounds there are with respect to uniform distributions and the upper bounds which essentially meet the lower bounds are proven for the worst-case as done in this paper.

An interesting open problem is whether a variant of the $\mathbf{GD}_{X,Y}$ algorithm (see Figure 4.2) exists such that, for all sequences $\boldsymbol{s} = \langle(\boldsymbol{x}_t, y_t)\rangle_{t \leq m}$ satisfying $||\boldsymbol{x}_t|| \leq X$ and $|y_t| \leq Y$ for all t, the additional total loss of the algorithm on $\boldsymbol{s}$ over and above $\inf_{\boldsymbol{w} \in \mathcal{X}} L_{\boldsymbol{w}}(\boldsymbol{s})$ is bounded by a function of X, Y only. Notice that this does not contradict Theorem 7.2.

The most challenging research direction is to prove worst case loss bounds for other gradient descent applications (by tuning the learning rate) as we have done in this paper for linear functions and the square loss. For example, are there useful worst case loss bounds for learning linear functions with other loss functions than the square loss. Another interesting case would be worst case loss bounds for learning the class of linear functions passed through a fixed transfer function (such as tanh or the sigmoid function) for any reasonable loss function.

Acknowledgements

We thank Ethan Bernstein for helpful conversations, and for pointing out an error in an earlier version of this paper. Thanks to Jyrki Kivinen for simplying the proof of Theorem 7.2 and to Peter Auer, Gianpiero Cattaneo, and Shuxian Lou for their comments. We are also grateful to Jan Mycielski for telling us about the paper by Kaczmarz.

Part of this research was done while Nicolò Cesa-Bianchi was visiting UC Santa Cruz partially supported by the "Progetto finalizzato sistemi informatici e calcolo parallelo" of CNR under grant 91.00884.69.115.09672 (Italy), and Phil Long was at Technische Universitaet Graz supported by a Lise Meitner Fellowship from the Fonds zur Förderung der wissenschaftlichen Forschung (Austria), and at UC Santa Cruz supported by a UCSC Chancellor's dissertation-year fellowship. Phil Long was also supported by AFOSR grant F49620-92-J-051. Manfred Warmuth was supported by by ONR grant NO0014-91-J-1162 and NSF grant IRI-9123692.

References

[CFH+93] N. Cesa-Bianchi, Y. Freund, D.P. Helmbold, D. Haussler, R.E. Schapire, and M.K. Warmuth. How to use expert advice. *Proceedings of the 25th ACM Symposium on the Theory of Computation*, 1993.

[Daw84] A. Dawid. Statistical theory: The prequential approach. *Journal of the Royal Statistical Society (Series A)*, pages 278–292, 1984.

[DH73] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.

[FM91] V. Faber and J. Mycielski. Applications of learning theorems. *Fundamenta Informaticae*, 15(2):145–167, 1991.

[FMG92] M. Feder, N. Merhav, and M. Gutman. Universal prediction of individual sequences. *IEEE transactions of information theory*, 38:1258–1270, 1992.

[GL89] G.H. Golub and C.F. Van Loan. *Matrix Computations*. The John Hopkins UP, 1989.

[Har91] W. Hardle. *Smoothing Techniques*. Springer Verlag, 1991.

[HJ85] R.A. Horn and C.R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

[Kac37] S. Kaczmarz. Angenaherte Auflösung von systemen linearer gleichungen. *Bull. Acad. Polon. Sci. Lett. A*, 35:355–357, 1937.

[KL92] D. Kimber and P.M. Long. The learning complexity of smooth functions of a single variable. *The 1992 Workshop on Computational Learning Theory*, pages 153–159, 1992.

[LLW91] N. Littlestone, P.M. Long, and M.K. Warmuth. On-line learning of linear functions. *Proceedings of the 23rd ACM Symposium on the Theory of Computation*, pages 465–475, 1991.

[LMT91] R.P. Lippman, J.E. Moody, and D.S. Touretsky. *Advances in Neural Information Processing Systems*, volume 3. Morgan Kaufmann, 1991.

[Lue84] D.G. Luenberger. *Linear and Nonlinear Programming.* Addison-Wesley, 1984.

[LW91] N. Littlestone and M.K. Warmuth. The weighted majority algorithm. Technical Report UCSC-CRL-91-28, UC Santa Cruz, October 1991. A preliminary version appeared in *the Proceedings of the 30th Annual IEEE Symposium on the Foundations of Computer Science*, October 89, pages 256-261.

[MF92] N. Merhav and M. Feder. Universal sequential learning and decision from individual data sequences. *The 1992 Workshop on Computational Learning Theory*, pages 413–427, 1992.

[MHL92] J.E. Moody, S.J. Hanson, and R.P. Lippman. *Advances in Neural Information Processing Systems*, volume 4. Morgan Kaufmann, 1992.

[MS91] J. Mycielski and S. Swierczkowski. General learning theorems. Unpublished, 1991.

[Myc88] J. Mycielski. A learning algorithm for linear operators. *Proceedings of the American Mathematical Society*, 103(2):547–550, 1988.

[Tou89] David S. Touretsky. *Advances in Neural Information Processing Systems*, volume 1. Morgan Kaufmann, 1989.

[Tou90] David S. Touretsky. *Advances in Neural Information Processing Systems*, volume 2. Morgan Kaufmann, 1990.

[Vov90] V. Vovk. Aggregating strategies. In *Proceedings of the 3nd Workshop on Computational Learning Theory*, pages 371–383. Morgan Kaufmann, 1990.

[WH60] B. Widrow and M.E. Hoff. Adaptive switching circuits. *1960 IRE WESCON Conv. Record*, pages 96–104, 1960.

[You88] N. Young. *An introduction to Hilbert space.* Cambridge University Press, 1988.