Smooth Boosting and Learning with Malicious Noise

Rocco A. Servedio

Division of Engineering and Applied Sciences, Harvard University
Cambridge, MA 02138
rocco@deas.harvard.edu

Abstract. We describe a new boosting algorithm which generates only smooth distributions which do not assign too much weight to any single example. We show that this new boosting algorithm can be used to construct efficient PAC learning algorithms which tolerate relatively high rates of malicious noise. In particular, we use the new smooth boosting algorithm to construct malicious noise tolerant versions of the PAC-model p-norm linear threshold learning algorithms described in [23]. The bounds on sample complexity and malicious noise tolerance of these new PAC algorithms closely correspond to known bounds for the online p-norm algorithms of Grove, Littlestone and Schuurmans [14] and Gentile and Littlestone [13]. As special cases of our new algorithms we obtain linear threshold learning algorithms which match the sample complexity and malicious noise tolerance of the online Perceptron and Winnow algorithms. Our analysis reveals an interesting connection between boosting and noise tolerance in the PAC setting.

1 Introduction

Any realistic model of learning from examples must address the issue of noisy data. In 1985 Valiant introduced the notion of PAC learning in the presence of *malicious noise*. This is a worst-case model of errors in which some fraction of the labeled examples given to a learning algorithm may be corrupted by an adversary who can modify both example points and labels in an arbitrary fashion (a detailed description of the model is given in Section 3). The frequency of such corrupted examples is known as the *malicious noise rate*.

Learning in the presence of malicious noise is in general quite difficult. Kearns and Li [16] have shown that for many concept classes it is impossible to learn to accuracy ϵ if the malicious noise rate exceeds $\frac{\epsilon}{1+\epsilon}$. In fact, for many interesting concept classes (such as the class of linear threshold functions), the best efficient algorithms known can only tolerate malicious noise rates significantly lower than this general upper bound. Despite these difficulties, the importance of being able to cope with noisy data has led many researchers to study PAC learning in the presence of malicious noise (see e.g. [1–3, 6, 7, 20]).

In this paper we give a new *smooth boosting* algorithm which can be used to transform a malicious noise tolerant weak learning algorithm into a PAC algorithm which learns successfully in the presence of malicious noise. We use this smooth boosting algorithm to construct a family of PAC algorithms for learning linear threshold functions in the presence of malicious noise. These new algorithms match the sample complexity and noise tolerance of the online p-norm algorithms of Grove, Littlestone and Schuurmans [14] and Gentile and Littlestone [13], which include as special cases the well-known Perceptron and Winnow algorithms.

1.1 Smooth Boosting and Learning with Malicious Noise

Our basic approach is quite simple, as illustrated by the following example. Consider a learning scenario in which we have a weak learning algorithm L which takes as input a finite sample S of m labeled examples. Algorithm L is known to have some tolerance to malicious noise; specifically, L is guaranteed to generate a hypothesis with nonnegligible advantage provided that the frequency of noisy examples in its sample is at most 10%. We would like to learn to high accuracy in the presence of malicious noise at a rate of 1%.

The obvious approach in this setting is to use a boosting algorithm, which will generate some sequence of distributions $\mathcal{D}_1, \mathcal{D}_2, \ldots$ over S. This approach can fail, though, if the boosting algorithm generates distributions which are very skewed from the uniform distribution on S; if distribution $\mathcal{D}_i$ assigns weights as large as $\frac{20}{m}$ to individual points in S, for instance, then the frequency of noisy examples for L in stage i could be as high as 20%. What we need instead is a *smooth* boosting algorithm which only constructs distributions $\mathcal{D}_i$ over S which never assign weight greater than $\frac{10}{m}$ to any single example. By using such a smooth booster we are assured that the weak learner will function successfully at each stage, so the overall boosting process will work correctly.

While the setting described above is artificial, we note that indirect empirical evidence has been given supporting the smooth boosting approach for noisy settings. It is well known [8, 21] that commonly used boosting algorithms such as `AdaBoost` [11] can perform poorly on noisy data. Dietterich [8] has suggested that this poor performance is due to `AdaBoost`'s tendency to generate very skewed distributions which put a great deal of weight on a few noisy examples. This overweighting of noisy examples cannot occur under a smooth boosting regimen.

In Section 2 we give a new boosting algorithm, `SmoothBoost`, which is guaranteed to generate only smooth distributions as described above. We show in Section 5 that the distributions generated by `SmoothBoost` are optimally smooth.

`SmoothBoost` is not the first boosting algorithm which attempts to avoid the skewed distributions of `AdaBoost`; algorithms with similar smoothness guarantees have been given by Domingo and Watanabe [9] and Impagliazzo [15]. Freund [10] has also described a boosting algorithm which uses a more moderate weighting scheme than `AdaBoost`. In Section 2.3 we show that our `SmoothBoost` algorithm has several other desirable properties, such as constructing a large margin final hypothesis, which are essential for the noisy linear threshold learning application of Section 3. We discuss the relationship between `SmoothBoost` and the algorithms of [9, 10, 15] in Section 2.4.

1.2 Learning Linear Threshold Functions with Malicious Noise

We use the `SmoothBoost` algorithm in Section 3 to construct a family of PAC-model malicious noise tolerant algorithms for learning linear threshold functions. A similar family was constructed by Servedio in [23] using `AdaBoost` instead of `SmoothBoost` as the boosting component. It was shown in [23] that for linearly separable data these PAC model algorithms have sample complexity bounds which are essentially identical to those of the online p-norm linear threshold learning algorithms of Grove, Littlestone and Schuurmans [14], which include as special cases ($p = 2$ and $p = \infty$) the well-studied online Perceptron and Winnow algorithms.

Gentile and Littlestone [13] have given mistake bounds for the online p-norm algorithms when run on examples which are not linearly separable, thus generalizing previous bounds on noise tolerance for Perceptron [12] and Winnow [19]. A significant drawback of the `AdaBoost`-based PAC-model p-norm algorithms of [23] is that they do not appear to succeed in the presence of malicious noise. We show in Section 4 that for all values $2 \leq p \leq \infty$, our new PAC algorithms which use `SmoothBoost` match both the sample complexity and the malicious noise tolerance of the online p-norm algorithms. Our construction thus provides malicious noise tolerant PAC analogues of Perceptron and Winnow (and many other algorithms as well).

2 Smooth Boosting

In this section we describe a new boosting algorithm, `SmoothBoost`, which has several useful properties. `SmoothBoost` only constructs smooth distributions which do not put too much weight on any single example; it can be used to generate a large margin final hypothesis; and it can be used with a weak learning algorithm which outputs real-valued hypotheses. All of these properties are essential for the noisy linear threshold learning problem we address in Section 3.

2.1 Preliminaries

We fix some terminology from [15] first. A *measure* on a finite set is a function $M : S \to [0, 1]$. We write $|M|$ to denote $\sum_{x \in S} M(x)$. Given a measure M, there is a natural induced distribution $\mathcal{D}_M$ defined by $\mathcal{D}_M(x) = M(x)/|M|$. This definition yields

Observation 1 $L_\infty(\mathcal{D}_M) \leq \frac{1}{|M|}$.

Let $\mathcal{D}$ be a distribution over a set $S = \langle x^1, y_1 \rangle, \ldots, \langle x^m, y_m \rangle$ of labeled examples with each $y_j \in \{-1, 1\}$ and let h be a real-valued function which maps $\{x^1, \ldots, x^m\}$ into $[-1, 1]$. If $\frac{1}{2} \sum_{j=1}^{m} \mathcal{D}(j)|h(x^j) - y_j| \leq \frac{1}{2} - \gamma$ then we say that the *advantage* of h under $\mathcal{D}$ is γ. We say that an algorithm which takes S and $\mathcal{D}$ as input and outputs an h which has advantage at least $\gamma > 0$ is a *weak learning algorithm* (this is somewhat less general than the notion of weak learning which

```
Input:  parameters $0 < \kappa < 1$, $0 \le \theta \le \gamma < \frac{1}{2}$
        sample $S = \langle x^1, y_1 \rangle, \ldots, \langle x^m, y_m \rangle$ where each $y_i \in \{-1, 1\}$
        weak learner WL which takes input $(S, \mathcal{D}_t)$ and outputs
            $h_t : \{x^1, \ldots, x^m\} \to [-1, 1]$

Output: hypothesis $h(x) = \mathrm{sign}(f(x))$

 1. forall $j = 1, \ldots, m$ set $M_1(j) = 1$
 2. forall $j = 1, \ldots, m$ set $N_0(j) = 0$
 3. set $t = 1$
 4. until $|M_t|/m < \kappa$ do
 5.     forall $j = 1, \ldots, m$ set $\mathcal{D}_t(j) = M_t(j)/|M_t|$
 6.     run WL$(S, \mathcal{D}_t)$ to get $h_t$ such that $\frac{1}{2}\sum_{j=1}^{m} \mathcal{D}_t(j)|h_t(x^j) - y_j| \le \frac{1}{2} - \gamma$
 7.     forall $j = 1, \ldots, m$ set $N_t(j) = N_{t-1}(j) + y_j h_t(x^j) - \theta$
 8.     forall $j = 1, \ldots, m$ set $M_{t+1}(j) = \begin{cases} 1 & \text{if } N_t(j) < 0 \\ (1-\gamma)^{N_t(j)/2} & \text{if } N_t(j) \ge 0 \end{cases}$
 9.     set $t = t + 1$
10. set $T = t - 1$
11. return $h = \mathrm{sign}(f(x))$ where $f(x) = \frac{1}{T}\sum_{i=1}^{T} h_i(x)$
```

Fig. 1. The SmoothBoost algorithm.

was originally introduced by Kearns and Valiant in [17] but is sufficient for our purposes). Finally, let $g(x) = \mathrm{sign}(f(x))$ where $f : X \to [-1, 1]$ is a real-valued function. We say that the *margin* of g on a labeled example $\langle x, y \rangle \in X \times \{-1, 1\}$ is $yf(x)$; intuitively, this is the amount by which g predicts y correctly. Note that the margin of g on $\langle x, y \rangle$ is nonnegative if and only if g predicts y correctly.

2.2 The SmoothBoost Algorithm

The SmoothBoost algorithm is given in Figure 1. The parameter κ is the desired error rate of the final hypothesis, the parameter γ is the guaranteed advantage of the hypotheses returned by the weak learner, and θ is the desired margin of the final hypothesis. SmoothBoost runs the weak learning algorithm several times on a sequence of carefully constructed distributions and outputs a thresholded sum of the hypotheses thus generated. The quantity $N_t(j)$ in line 7 may be viewed as the cumulative amount by which the hypotheses $h_1, \ldots, h_t$ beat the desired margin θ on the labeled example $\langle x^j, y_j \rangle$. The measure M_{t+1} assigns more weight to examples where N_t is small and less weight to examples where N_t is large, thus forcing the weak learner to focus in stage $t + 1$ on examples where previous hypotheses have done poorly. Note that since any measure maps into $[0, 1]$ there is a strict bound on the amount of weight which can be assigned to any example.

2.3 Proof of Correctness

Several useful properties of the SmoothBoost algorithm are easy to verify. The algorithm is called SmoothBoost because each distribution it constructs is guaranteed to be "smooth," i.e. no single point receives too much weight:

Lemma 1. *Each $\mathcal{D}_t$ defined in step 5 of* SmoothBoost *has $L_\infty(\mathcal{D}_t) \leq \frac{1}{\kappa m}$.*

Proof. Follows directly from Observation 1 and the condition in line 4. □

Another useful property is that the final hypothesis h has margin at least θ on all but a κ fraction of the points in S :

Theorem 1. *If* SmoothBoost *terminates then f satisfies $\frac{|\{j\ :\ y_j f(x^j) \leq \theta\}|}{m} < \kappa$.*

Proof. Since $N_T(j) = T(y_j f(x^j) - \theta)$, if $y_j f(x^j) \leq \theta$ then $N_T(j) \leq 0$ and hence $M_{T+1}(j) = 1$. Consequently we have

$$\frac{|\{j\ :\ y_j f(x^j) \leq \theta\}|}{m} \leq \frac{\sum_{j=1}^m M_{T+1}(j)}{m} = \frac{|M_{T+1}|}{m} < \kappa$$

by the condition in line 4. □

Note that since $\theta \geq 0$ Theorem 1 implies that the final SmoothBoost hypothesis is correct on all but a κ fraction of S.

Finally we must show that the algorithm terminates in a reasonable amount of time. The following theorem bounds the number of times that SmoothBoost will execute its main loop:

Theorem 2. *If each hypothesis h_t returned by* WL *in line 6 has advantage at least γ under $\mathcal{D}_t$ (i.e. satisfies the condition of line 6) and θ is set to $\frac{\gamma}{2+\gamma}$, then* SmoothBoost *terminates with $T < \frac{2}{\kappa\gamma^2\sqrt{1-\gamma}}$.*

As will be evident from the proof, slightly different bounds on T can be established by choosing different values of θ in the range $[0, \gamma]$. We take $\theta = \frac{\gamma}{2+\gamma}$ in the theorem above both to obtain a margin of $\Omega(\gamma)$ and to obtain a clean bound in the theorem. Theorem 2 follows from the bounds established in the following two lemmas:

Lemma 2. $\sum_{j=1}^m \sum_{t=1}^T M_t(j) y_j h_t(x^j) \geq 2\gamma \sum_{t=1}^T |M_t|$.

Lemma 3. *If $\theta = \frac{\gamma}{2+\gamma}$, then $\sum_{j=1}^m \sum_{t=1}^T M_t(j) y_j h_t(x^j) < \frac{2m}{\gamma\sqrt{1-\gamma}} + \gamma \sum_{t=1}^T |M_t|$.*

Combining these bounds we obtain $\frac{2m}{\gamma\sqrt{1-\gamma}} > \gamma \sum_{t=1}^T |M_t| \geq \gamma\kappa m T$ where the last inequality is because $|M_t| \geq \kappa m$ for $t = 1, \ldots, T$.

Proof of Lemma 2: Since $h_t(x^j) \in [-1, 1]$ and $y_j \in \{-1, 1\}$, we have $y_j h_t(x^j) = 1 - |h_t(x^j) - y_j|$, and thus

$$\sum_{j=1}^m \mathcal{D}_t(j) y_j h_t(x^j) = \sum_{j=1}^m \mathcal{D}_t(j)(1 - |h_t(x^j) - y_j|) \geq 2\gamma.$$

This implies that

$$\sum_{j=1}^{m}\sum_{t=1}^{T} M_t(j) y_j h_t(x^j) = \sum_{t=1}^{T} |M_t| \sum_{j=1}^{m} \mathcal{D}_t(j) y_j h_t(x^j) \geq \sum_{t=1}^{T} 2\gamma |M_t|.$$

□

The proof of Lemma 3 is given in Appendix A.

2.4 Comparison with Other Boosting Algorithms

The `SmoothBoost` algorithm was inspired by an algorithm given by Impagliazzo in the context of hard-core set constructions in complexity theory [15]. Klivans and Servedio [18] observed that Impagliazzo's algorithm can be reinterpreted as a boosting algorithm which generates distributions $\mathcal{D}_t$ which, like the distributions generated by `SmoothBoost`, satisfy $L_\infty(\mathcal{D}_t) \leq \frac{1}{\kappa m}$. However, our `SmoothBoost` algorithm differs from Impagliazzo's algorithm in several important ways. The algorithm in [15] uses additive rather than multiplicative updates for $M_t(j)$, and the bound on T which is given for the algorithm in [15] is $O(\frac{1}{\kappa^2\gamma^2})$ which is worse than our bound by essentially a factor of $\frac{1}{\kappa}$. Another important difference is that the algorithm in [15] has no θ parameter and does not appear to output a large margin final hypothesis. Finally, the analysis in [15] only covers the case where the weak hypotheses are binary-valued rather than real-valued.

Freund and Schapire's well-known boosting algorithm `AdaBoost` is somewhat faster than `SmoothBoost`, requiring only $T = O(\frac{\log(1/\kappa)}{\gamma^2})$ stages [11]. Like `SmoothBoost`, `AdaBoost` can be used with real-valued weak hypotheses and can be used to output a large margin final hypothesis [22]. However, `AdaBoost` is not guaranteed to generate only smooth distributions, and thus does not appear to be useful in a malicious noise context.

Freund has recently introduced and studied a sophisticated boosting algorithm called `BrownBoost` [10] which uses a gentler weighting scheme than `AdaBoost`. Freund suggests that `BrownBoost` should be well suited for dealing with noisy data; however it is not clear from the analysis in [10] whether `BrownBoost`-generated distributions satisfy a smoothness property such as the $L_\infty(\mathcal{D}_t) \leq \frac{1}{\kappa m}$ property of `SmoothBoost`, or whether `BrownBoost` can be used to generate a large margin final hypothesis. We note that the `BrownBoost` algorithm is much more complicated to run than `SmoothBoost`, as it involves solving a differential equation at each stage of boosting.

`SmoothBoost` is perhaps most similar to the modified AdaBoost algorithm `MadaBoost` which was recently defined and analyzed by Domingo and Watanabe [9]. Like `SmoothBoost`, `MadaBoost` uses multiplicative updates on weights and never allows weights to exceed 1 in value. Domingo and Watanabe proved that `MadaBoost` takes at most $T \leq \frac{2}{\kappa\gamma^2}$ stages, which is quite similar to our bound in Theorem 2. (If we set $\theta = 0$ in `SmoothBoost`, a slight modification of the proof of Theorem 2 gives a bound of roughly $\frac{4}{3\kappa\gamma^2}$, which improves the `Madaboost` bound by a constant factor.) However, the analysis for `MadaBoost` given in [9] only covers only the case of binary-valued weak hypotheses, and does not establish that `MadaBoost` generates a large margin final hypothesis. We also note that our proof technique of simultaneously upper and lower bounding $\sum_{j=1}^{m} \sum_{t=1}^{T} M_t(j) y_j h_t(x^j)$ is different from the approach used in [9].

3 Learning Linear Threshold Functions with Malicious Noise

In this section we show how the `SmoothBoost` algorithm can be used in conjunction with a simple noise tolerant weak learning algorithm to obtain a PAC learning algorithm for learning linear threshold functions with malicious noise.

3.1 Geometric Preliminaries

For $\overline{x} = (x_1, \ldots, x_n) \in \Re^n$ and $p \geq 1$ we write $\|\overline{x}\|_p$ to denote the p-norm of $\overline{x}$, namely $\|\overline{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}$. The ∞-norm of $\overline{x}$ is $\|\overline{x}\|_\infty = \max_{i=1,\ldots,n} |x_i|$. We write $B_p(R)$ to denote the p-norm ball of radius R, i.e. $B_p(R) = \{\overline{x} \in \Re^n : \|\overline{x}\|_p \leq R\}$.

For $p, q \geq 1$ the q-norm is *dual* to the p-norm if $\frac{1}{p} + \frac{1}{q} = 1$; so the 1-norm and the ∞-norm are dual to each other and the 2-norm is dual to itself. For the rest of the paper p and q always denote dual norms. The following facts (see e.g. [25] pp. 203-204) will be useful:

Hölder Inequality: $|\overline{u} \cdot \overline{v}| \leq \|\overline{u}\|_p \|\overline{v}\|_q$ for all $\overline{u}, \overline{v} \in \Re^n$ and $1 \leq p \leq \infty$.

Minkowski Inequality: $\|\overline{u} + \overline{v}\|_p \leq \|\overline{u}\|_p + \|\overline{v}\|_p$ for all $\overline{u}, \overline{v} \in \Re^n$ and $1 \leq p \leq \infty$.

Finally, recall that a *linear threshold function* is a function $f : \Re^n \to \{-1, 1\}$ such that $f(\overline{x}) = \text{sign}(\overline{u} \cdot \overline{x})$ for some $\overline{u} \in \Re^n$.

3.2 PAC Learning with Malicious Noise

Let $EX_{MAL}^{\eta}(\overline{u}, \mathcal{D})$ be a *malicious example oracle with noise rate* η that behaves as follows when invoked: with probability $1 - \eta$ the oracle returns a *clean* example $\langle \overline{x}, \text{sign}(\overline{u} \cdot \overline{x}) \rangle$ where $\overline{x}$ is drawn from the probability distribution $\mathcal{D}$ over $B_p(R)$. With probability η, though, $EX_{MAL}^{\eta}(\overline{u}, \mathcal{D})$ returns a *dirty* example $\langle \overline{x}, y \rangle \in B_p(R) \times \{-1, 1\}$ about which nothing can be assumed. Such a malicious example $\langle \overline{x}, y \rangle$ may be chosen by a computationally unbounded adversary which has complete knowledge of $\overline{u}$, $\mathcal{D}$, and the state of the learning algorithm when the oracle is invoked.

The goal of a learning algorithm in this model is to construct an approximation to the target concept $\text{sign}(\overline{u} \cdot \overline{x})$. More formally, we say that a Boolean function $h : \Re^n \to \{-1, 1\}$ is an *ϵ-approximator for $\overline{u}$ under $\mathcal{D}$* if $\Pr_{\overline{x} \in \mathcal{D}}[h(\overline{x}) \neq \text{sign}(\overline{u} \cdot \overline{x})] \leq \epsilon$. The learning algorithm is given an accuracy parameter ϵ and a confidence parameter δ, has access to $EX_{MAL}^{\eta}(\overline{u}, \mathcal{D})$, and must output a hypothesis h which, with probability at least $1 - \delta$, is an ϵ-approximator for $\overline{u}$ under

$\mathcal{D}$. The *sample complexity* of a learning algorithm in this model is the number of times it queries the malicious example oracle.

(A slightly stronger model of PAC learning with malicious noise has also been proposed [1,6]. In this model first a clean sample of the desired size is drawn from a noise-free oracle; then each point in the sample is independently selected with probability η; then an adversary replaces each selected point with a dirty example of its choice; and finally the corrupted sample is provided to the learning algorithm. This model is stronger than the original malicious noise model since each dirty example is chosen by the adversary with full knowledge of the entire sample. All of our results also hold in this stronger model.)

A final note: like the Perceptron algorithm, the learning algorithms which we consider will require that the quantity $\overline{u} \cdot \overline{x}$ be bounded away from zero (at least most of the time). We thus say that a distribution $\mathcal{D}$ is *ξ-good for $\overline{u}$* if $|\overline{u} \cdot \overline{x}| \geq \xi$ for all $\overline{x}$ which have nonzero probability under $\mathcal{D}$, and we restrict our attention to learning under ξ-good distributions. (Of course, dirty examples drawn from $EX^{\eta}_{MAL}(\overline{u}, \mathcal{D})$ need not satisfy $|\overline{u} \cdot \overline{x}| \geq \xi$.)

3.3 A Noise Tolerant Weak Learning Algorithm

As shown in Figure 2, our weak learning algorithm for linear threshold functions, called `WLA`, takes as input a data set S and a distribution $\mathcal{D}$ over S. The algorithm computes the vector $\overline{z}$ which is the average location of the (label-normalized) points in S under $\mathcal{D}$, transforms $\overline{z}$ to obtain a vector $\overline{w}$, and predicts using the linear functional defined by $\overline{w}$. As motivation for the algorithm, note that if every example pair $\langle \overline{x}, y \rangle$ satisfies $y = \text{sign}(\overline{u} \cdot \overline{x})$ for some $\overline{u}$, then each point $y\overline{x}$ would lie on the same side of the hyperplane defined by $\overline{u}$ as $\overline{u}$ itself, and hence the average vector $\overline{z}$ defined in Step 1 of the algorithm intuitively should point in roughly the same direction as $\overline{u}$.

In [23] it is shown that the `WLA` algorithm is a weak learning algorithm for linear threshold functions in a noise-free setting. The following theorem shows that if a small fraction of the examples in S are affected by malicious noise, `WLA` will still generates a hypothesis with nonnegligible advantage provided that the input distribution $\mathcal{D}$ is sufficiently smooth.

Theorem 3. *Fix $2 \leq p \leq \infty$ and let $S = \langle \overline{x}^1, y_1 \rangle, \dots, \langle \overline{x}^m, y_m \rangle$ be a set of labeled examples with each $\overline{x}^j \in B_p(R)$. Let $\mathcal{D}$ be a distribution over S such that $L_\infty(\mathcal{D}) \leq \frac{1}{\kappa m}$. Suppose that $\xi > 0$ and $\overline{u} \in \Re^n$ are such that $\xi \leq R\|\overline{u}\|_q$ and at most $\eta' m$ examples in S do not satisfy $y_j(\overline{u} \cdot \overline{x}^j) \geq \xi$, where $\eta' \leq \frac{\kappa \xi}{4R\|\overline{u}\|_q}$. Then `WLA`$(p, S, \mathcal{D})$ returns a hypothesis $h : B_p(R) \to [-1, 1]$ which has advantage at least $\frac{\xi}{4R\|\overline{u}\|_q}$ under $\mathcal{D}$.*

Proof: By Hölder's inequality, for any $\overline{x} \in B_p(R)$ we have

$$|h(\overline{x})| = \frac{|\overline{w} \cdot \overline{x}|}{\|\overline{w}\|_q R} \leq \frac{\|\overline{w}\|_q \|\overline{x}\|_p}{\|\overline{w}\|_q R} \leq 1,$$

and thus h indeed maps $B_p(R)$ into $[-1, 1]$.

```
Input:   parameter $p \geq 2$
         sample $S = \langle \overline{x}^1, y_1 \rangle, \ldots, \langle \overline{x}^m, y_m \rangle$ where each $y_i \in \{-1, 1\}$
         distribution $\mathcal{D}$ over $S$
         upper bound $R$ on $\|\overline{x}\|_p$

Output:  hypothesis $h(\overline{x})$

  1. set $\overline{z} = \sum_{j=1}^m \mathcal{D}(j) y_j \overline{x}^j$
  2. for all $i = 1, \ldots, n$ set $w_i = \mathrm{sign}(z_i)|z_i|^{p-1}$
  3. return hypothesis $h(\overline{x}) \equiv \overline{v} \cdot \overline{x}$ where $\overline{v} = \frac{\overline{w}}{\|\overline{w}\|_q R}$
```

Fig. 2. The p-norm weak learning algorithm `WLA`.

Now we show that h has the desired advantage. Since $h_t(\overline{x}^j) \in [-1, 1]$ and $y_j \in \{-1, 1\}$, we have $|h(\overline{x}^j) - y_j| = 1 - y_j h(\overline{x}^j)$, so

$$\frac{1}{2}\sum_{j=1}^m \mathcal{D}(j)|h(\overline{x}^j) - y_j| = \frac{1}{2}\sum_{j=1}^m \mathcal{D}(j)(1 - y_j h(\overline{x}^j)) = \frac{1}{2} - \left(\frac{\sum_{j=1}^m \mathcal{D}(j) y_j (\overline{w} \cdot \overline{x}^j)}{2\|\overline{w}\|_q R} \right).$$

To prove the theorem it thus suffices to show that $\frac{\sum_{j=1}^m \mathcal{D}(j) y_j (\overline{w} \cdot \overline{x}^j)}{\|\overline{w}\|_q} \geq \frac{\xi}{2\|\overline{u}\|_q}$. The numerator of the left side is $\overline{w} \cdot \left(\sum_{j=1}^m \mathcal{D}(j) y_j \overline{x}^j \right) = \overline{w} \cdot \overline{z} = \sum_{i=1}^n |z_i|^p = \|\overline{z}\|_p^p$. Using the fact that $(p-1)q = p$, the denominator is

$$\|\overline{w}\|_q = \left(\sum_{i=1}^n \left(|z_i|^{p-1} \right)^q \right)^{1/q} = \left(\sum_{i=1}^n |z_i|^p \right)^{1/q} = \|\overline{z}\|_p^{p/q}.$$

We can therefore rewrite the left side as $\|\overline{z}\|_p^p / \|\overline{z}\|_p^{p/q} = \|\overline{z}\|_p$, and thus our goal is to show that $\|\overline{z}\|_p \geq \frac{\xi}{2\|\overline{u}\|_q}$. By Hölder's inequality it suffices to show that $\overline{z} \cdot \overline{u} \geq \frac{\xi}{2}$, which we now prove.

Let $S_1 = \{\langle \overline{x}^j, y_j \rangle \in S : y_j(\overline{u} \cdot \overline{x}^j) \geq \xi\}$ and let $S_2 = S \setminus S_1$. The definition of S_1 immediately yields $\sum_{j \in S_1} \mathcal{D}(j) y_j (\overline{u} \cdot \overline{x}^j) \geq \mathcal{D}(S_1)\xi$. Moreover, since each $\|\overline{x}^j\|_p \leq R$, by Hölder's inequality we have $y_j(\overline{u} \cdot \overline{x}^j) \geq -\|\overline{x}^j\|_p \cdot \|\overline{u}\|_q \geq -R\|\overline{u}\|_q$ for each $\langle \overline{x}^j, y_j \rangle \in S_2$. Since each example in S_2 has weight at most $\frac{1}{\kappa m}$ under $\mathcal{D}$, we have $\mathcal{D}(S_2) \leq \frac{\eta'}{\kappa}$, and hence

$$\begin{aligned}
\overline{z} \cdot \overline{u} &= \sum_{j=1}^m \mathcal{D}(j) y_j (\overline{u} \cdot \overline{x}^j) = \sum_{j \in S_1} \mathcal{D}(j) y_j (\overline{u} \cdot \overline{x}^j) + \sum_{j \in S_2} \mathcal{D}(j) y_j (\overline{u} \cdot \overline{x}^j) \\
&\geq \mathcal{D}(S_1)\xi - \mathcal{D}(S_2) R \|\overline{u}\|_q \geq \left(1 - \frac{\eta'}{\kappa}\right)\xi - \frac{\eta' R \|\overline{u}\|_q}{\kappa} \\
&\geq \frac{3\xi}{4} - \frac{\xi}{4} = \frac{\xi}{2},
\end{aligned}$$

where the inequality $(1 - \frac{\eta'}{\kappa}) \geq \frac{3}{4}$ follows from the bound on η' and the fact that $\xi \leq R\|\overline{u}\|_q$. □

3.4 Putting it All Together

The algorithm for learning $\text{sign}(\overline{u} \cdot \overline{x})$ with respect to a ξ-good distribution $\mathcal{D}$ over $B_p(R)$ is as follows:

- Draw from $EX^{\eta}_{MAL}(\overline{u}, \mathcal{D})$ a sample $S = \langle \overline{x}^1, y_1 \rangle, \ldots, \langle \overline{x}^m, y_m \rangle$ of m labeled examples.
- Run `SmoothBoost` on S with parameters $\kappa = \frac{\epsilon}{4}$, $\gamma = \frac{\xi}{4R\|\overline{u}\|_q}$, $\theta = \frac{\gamma}{2+\gamma}$ using `WLA` as the weak learning algorithm.

We now determine constraints on the sample size m and the malicious noise rate η under which this is a successful and efficient learning algorithm.

We first note that since $\mathcal{D}$ is ξ-good for $\overline{u}$, we have that $\xi \leq R\|\overline{u}\|_q$. Furthermore, since $\kappa = \frac{\epsilon}{4}$, Lemma 1 implies that each distribution $\mathcal{D}_t$ which is given to `WLA` by `SmoothBoost` has $L_\infty(\mathcal{D}_t) \leq \frac{4}{\epsilon m}$. Let $S_C \subseteq S$ be the clean examples and $S_D = S \setminus S_C$ the dirty examples in S. If $\eta \leq \frac{\epsilon\xi}{32R\|\overline{u}\|_q}$ and $m \geq \frac{96R\|\overline{u}\|_q}{\epsilon\xi} \log \frac{2}{\delta}$, then a simple Chernoff bound implies that with probability at least $1 - \frac{\delta}{2}$ we have $|S_D| \leq \frac{\epsilon\xi}{16R\|\overline{u}\|_q} m$. Thus, we can apply Theorem 3 with $\eta' = \frac{\epsilon\xi}{16R\|\overline{u}\|_q}$; so each weak hypothesis $h_t(\overline{x}) = \overline{v}^t \cdot \overline{x}$ generated by `WLA` has advantage $\frac{\xi}{4R\|\overline{u}\|_q}$ under $\mathcal{D}_t$. Consequently, by Theorems 1 and 2, `SmoothBoost` efficiently outputs a final hypothesis $h(\overline{x}) = \text{sign} f(\overline{x})$ which has margin less than θ on at most an $\frac{\epsilon}{4}$ fraction of S. Since $|S_C|$ is easily seen to be at least $\frac{m}{2}$, we have that the margin of h is less than θ on at most an $\frac{\epsilon}{2}$ fraction of S_C. This means that we can apply powerful methods from the theory of data-dependent structural risk minimization [5, 24] to bound the error of h under $\mathcal{D}$.

Recall that the final `SmoothBoost` hypothesis is $h(\overline{x}) = \text{sign}(f(\overline{x}))$ where $f(\overline{x}) = \overline{v} \cdot \overline{x}$ is a convex combination of hypotheses $h_t(\overline{x}) = \overline{v}^t \cdot \overline{x}$. Since each vector $\overline{v}^t$ satisfies $\|\overline{v}^t\|_q \leq \frac{1}{R}$, by Minkowski's inequality we have that $\|\overline{v}\|_q \leq \frac{1}{R}$ as well. The following theorem is proved in [23]:

Theorem 4. *Fix any value* $2 \leq p \leq \infty$ *and let* $\mathcal{F}$ *be the class of functions* $\{\overline{x} \mapsto \overline{v} \cdot \overline{x} : \|\overline{v}\|_q \leq \frac{1}{R}, \overline{x} \in B_p(R)\}$. *Then* $fat_{\mathcal{F}}(\mu) \leq \frac{2 \log 4n}{\mu^2}$, *where* $fat_{\mathcal{F}}(\mu)$ *is the* fat-shattering dimension of $\mathcal{F}$ at scale μ *as defined in, e.g., [4, 5, 24].*

The following theorem is from [5]:

Theorem 5. *Let* $\mathcal{F}$ *be a collection of real-valued functions over some domain* X, *let* $\mathcal{D}$ *be a distribution over* $X \times \{-1, 1\}$, *let* $S = \langle \overline{x}^1, y_1 \rangle, \ldots, \langle \overline{x}^m, y_m \rangle$ *be a sequence of labeled examples drawn from* $\mathcal{D}$, *and let* $h(\overline{x}) = sign(f(\overline{x}))$ *for some* $f \in \mathcal{F}$. *If* h *has margin less than* θ *on at most* k *examples in* S, *then with probability at least* $1 - \delta$ *we have that* $\Pr_{\langle \overline{x}, y \rangle \in \mathcal{D}}[h(\overline{x}) \neq y]$ *is at most*

$$\frac{k}{m} + \sqrt{\frac{2}{m}(d \ln(34e/m) \log(578m) + \ln(4/\delta))}, \tag{1}$$

where $d = fat_{\mathcal{F}}(\theta/16)$.

We have that h has margin less than θ on at most an $\frac{\epsilon}{2}$ fraction of the clean examples S_C, so we may take k/m to be $\frac{\epsilon}{2}$ in the above theorem. Now if we apply Theorem 4 and solve for m the inequality obtained by setting (1) to be at most ϵ, we obtain

Theorem 6. *Fix $2 \leq p \leq \infty$ and let $\mathcal{D}$ be a distribution over $B_p(R)$ which is ξ-good for $\overline{u}$. The algorithm described above uses $m = \tilde{O}\left(\left(\frac{R\|\overline{u}\|_q}{\xi\epsilon}\right)^2\right)$ examples and outputs an ϵ-approximator for $\overline{u}$ under $\mathcal{D}$ with probability $1-\delta$ in the presence of malicious noise at a rate $\eta = \Omega\left(\epsilon \cdot \frac{\xi}{R\|\overline{u}\|_q}\right)$.*

4 Comparison with Online Algorithms

The bounds given by Theorem 6 on sample complexity and malicious noise tolerance of our algorithms based on `SmoothBoost` are remarkably similar to the bounds which can be obtained through a natural PAC conversion of the online p-norm algorithms introduced by Grove, Littlestone and Schuurmans [14] and studied by Gentile and Littlestone [13]. Grove, Littlestone and Schuurmans (Theorem 6.1) proved that the online p-norm algorithm makes at most $O\left(\left(\frac{R\|\overline{u}\|_q}{\xi}\right)^2\right)$ mistakes on linearly separable data. Subsequently Gentile and Littlestone [13] extended the analysis from [14] and considered a setting in which the examples are not linearly separable. Their analysis (Theorem 6) shows that if an example sequence containing K malicious errors is provided to the online p-norm algorithm, then the algorithm will make at most

$$O\left(\left(\frac{R\|\overline{u}\|_q}{\xi}\right)^2 + K \cdot \frac{R\|\overline{u}\|_q}{\xi}\right)$$

mistakes. To obtain PAC-model bounds on the online p-norm algorithms in the presence of malicious noise, we use the following theorem due to Auer and Cesa-Bianchi [3] (Theorem 6.2):

Theorem 7. *Fix a hypothesis class $\mathcal{H}$ of Vapnik-Chervonenkis dimension d. Let A be an online learning algorithm with the following properties: (1) A only uses hypotheses which belong to $\mathcal{H}$, (2) if A is given a noise-free example sequence then A makes at most m_0 mistakes, and (3) if A is given an example sequence with K malicious errors then A makes at most $m_0 + BK$ mistakes. Then there is a PAC algorithm A' which learns to accuracy ϵ and confidence δ, uses $\tilde{O}(\frac{B^2}{\epsilon^2} + \frac{m_0}{\epsilon} + \frac{d}{\epsilon})$ examples, and can tolerate a malicious noise rate $\eta = \frac{\epsilon}{2B}$.*

Applying this theorem, we find that these PAC conversions of the online p-norm algorithms have sample complexity and malicious noise tolerance bounds which are essentially identical to the bounds given for our `SmoothBoost`-based algorithm.

5 `SmoothBoost` is Optimally Smooth

It is evident from the proof of Theorem 6 that the smoothness of the distributions generated by `SmoothBoost` relates directly to the level of malicious noise which our linear threshold learning algorithm can tolerate. On the other hand, as mentioned in Section 1, Kearns and Li have shown that for a broad range of concept classes no algorithm can learn to accuracy ϵ in the presence of malicious noise at a rate $\eta > \frac{\epsilon}{1+\epsilon}$. Using the Kearns-Li upper bound on malicious noise tolerance, we prove in this section that `SmoothBoost` is optimal up to constant factors in terms of the smoothness of the distributions which it generates. This demonstrates an interesting connection between bounds on noise-tolerant learning and bounds on boosting algorithms.

Recall that if `SmoothBoost` is run on a set of m examples with input parameters κ, γ, θ, then each distribution $\mathcal{D}_t$ which `SmoothBoost` constructs will satisfy $L_\infty(\mathcal{D}_t) \leq \frac{1}{\kappa m}$. The proof is by contradiction; so suppose that there exists a boosting algorithm called `SuperSmoothBoost` which is similar to `SmoothBoost` but which has an even stronger guarantee on its distributions. More precisely we suppose that `SuperSmoothBoost` takes as input parameters κ, γ and a labeled sample S of size m, has access to a weak learning algorithm `WL`, generates a sequence $\mathcal{D}_1, \mathcal{D}_2, \ldots$ of distributions over S, and outputs a Boolean-valued final hypothesis h. As in Section 2.3, we suppose that if the weak learning algorithm `WL` always returns a hypothesis h_t which has advantage γ under $\mathcal{D}_t$, then `SuperSmoothBoost` will eventually halt and the final hypothesis h will agree with at least a $1-\kappa$ fraction of the labeled examples in S. Finally, we suppose that each distribution $\mathcal{D}_t$ is guaranteed to satisfy $L_\infty(\mathcal{D}_t) \leq \frac{1}{64\kappa m}$.

Consider the following severely restricted linear threshold learning problem: the domain is $\{-1,1\}^2 \subset \Re^2$, so any distribution $\mathcal{D}$ can assign weight only to these four points. Moreover, we only allow two possibilities for the target concept $\mathrm{sign}(\overline{u} \cdot \overline{x})$: the vector $\overline{u}$ is either $(1,0)$ or $(0,1)$. The four points in $\{-1,1\}^2$ are classified in all four possible ways by these two concepts, and hence the concept class consisting of these two concepts is a *distinct* concept class as defined by Kearns and Li [16]. It is clear that every example belongs to $B_\infty(1)$ (i.e. $R=1$), that $\|\overline{u}\|_1 = 1$, and that any distribution $\mathcal{D}$ over $\{-1,1\}^2$ is 1-good for $\overline{u}$ (i.e. $\xi = 1$).

Consider the following algorithm for this restricted learning problem:

- Draw from $EX^{\eta}_{MAL}(\overline{u}, \mathcal{D})$ a sample $S = \langle \overline{x}^1, y_1\rangle, \ldots, \langle \overline{x}^m, y_m\rangle$ of m labeled examples.
- Run `SuperSmoothBoost` on S with parameters $\kappa = \frac{\epsilon}{4}$, $\gamma = \frac{\xi}{4R\|\overline{u}\|_q} = \frac{1}{4}$ using `WLA` with $p = \infty$ as the weak learning algorithm.

Suppose that the malicious noise rate η is 2ϵ. As in Section 3.4, a Chernoff bound shows that for $m = O(\frac{1}{\epsilon}\log\frac{1}{\delta})$, with probability at least $1 - \frac{\delta}{2}$ we have that the sample S contains at most $4\epsilon m$ dirty examples. By the `SuperSmoothBoost` smoothness property and our choice of κ, we have that $L_\infty(\mathcal{D}_t) \leq \frac{1}{16\epsilon m}$. Theorem 3 now implies that each `WLA` hypothesis h_t has advantage at least $\frac{\xi}{4R\|\overline{u}\|_q} = \frac{1}{4}$

with respect to $\mathcal{D}_t$. As in Section 3.4, we have that with probability at least $1-\frac{\delta}{2}$ the final hypothesis h output by `SuperSmoothBoost` disagrees with at most an $\frac{\epsilon}{2}$ fraction of the clean examples S_C.

Since the domain is finite (in fact of size four) we can bound generalization error directly. A simple Chernoff bound argument shows that if m is sufficiently large, then with probability at least $1-\delta$ the hypothesis h will be an ϵ-approximator for $\text{sign}(\overline{u}\cdot\overline{x})$ under $\mathcal{D}$. However, Kearns and Li have shown (Theorem 1 of [16]) that no learning algorithm for a distinct concept class can learn to accuracy ϵ with probability $1-\delta$ in the presence of malicious noise at rate $\eta \geq \frac{\epsilon}{1+\epsilon}$. This contradiction proves that the `SuperSmoothBoost` algorithm cannot exist, and hence the distributions generated by `SmoothBoost` are optimal up to constant factors.

6 Conclusions and Further Work

One goal for future work is to improve the `SmoothBoost` algorithm given in Section 2. As noted in Section 5, the smoothness of the generated distributions is already essentially optimal; however it may be possible to improve other aspects of the algorithm such as the number of stages of boosting which are required. Is there an algorithm which matches the smoothness of `SmoothBoost` but, like `AdaBoost`, runs for only $O(\frac{\log(1/\kappa)}{\gamma^2})$ stages? Another possible improvement would be to eliminate the θ (margin) parameter of `SmoothBoost`; a version of the algorithm which automatically chooses an appropriate margin parameter would be useful in practical situations.

7 Acknowledgements

We thank Avrim Blum for a helpful discussion concerning the malicious noise tolerance of the Perceptron algorithm. We also thank Les Valiant for suggesting that the techniques in this paper could be used to prove a lower bound on the smoothness of an arbitrary boosting algorithm.

References

1. J. Aslam and S. Decatur. Specification and simulation of statistical query algorithms for efficiency and noise tolerance, *J. Comput. Syst. Sci.* **56** (1998), 191-208.
2. P. Auer. Learning nested differences in the presence of malicious noise, *Theoretical Computer Science* **185**(1) (1997), 159-175.
3. P. Auer and N. Cesa-Bianchi. On-line learning with malicious noise and the closure algorithm, *Ann. Math. and Artif. Intel.* **23** (1998), 83-99.
4. P. Bartlett, P. Long and R. Williamson. Fat-shattering and the learnability of real-valued functions, *J. Comput. Syst. Sci.,* **52**(3) (1996), 434-452.
5. P. Bartlett and J. Shawe-Taylor. Generalization performance of support vector machines and other pattern classifiers, *in* B. Scholkopf, C.J.C. Burges, and A.J. Smola, eds, *Advances in Kernel Methods – Support Vector Learning,* (1999), 43-54.

6. N. Cesa-Bianchi, E. Dichterman, P. Fischer, E. Shamir and H.U. Simon. Sample-efficient strategies for learning in the presence of noise, *J. ACM* **46**(5) (1999), 684-719.
7. S. Decatur. Statistical queries and faulty PAC oracles, *in* "Proc. Sixth Work. on Comp. Learning Theory" (1993), 262-268.
8. T.G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning,* **40**(2) (2000), 139-158.
9. C. Domingo and O. Watanabe. MadaBoost: a modification of AdaBoost. *in* "Proc. 13th Conf. on Comp. Learning Theory" (2000), 180-189.
10. Y. Freund. An adaptive version of the boost by majority algorithm, *in* "Proc. Twelfth Conf. on Comp. Learning Theory" (1999), 102-113.
11. Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting, *J. Comput. Syst. Sci.* **55**(1) (1997), 119-139.
12. Y. Freund and R. Schapire. Large margin classification using the perceptron algorithm, *in* "Proc. 11th Conf. Comp. Learning Theory" (1998), 209-217.
13. C. Gentile and N. Littlestone. The robustness of the p-norm algorithms, *in* "Proc. 12th Ann. Conf. on Comp. Learning Theory" (1999), 1-11.
14. A. Grove, N. Littlestone and D. Schuurmans. General convergence results for linear discriminant updates, *in* "Proc. 10th Ann. Conf. on Comp. Learning Theory" (1997), 171-183.
15. R. Impagliazzo. Hard-core distributions for somewhat hard problems, *in* "Proc. 36th Symp. on Found. of Comp. Sci." (1995), 538-545.
16. M. Kearns and M. Li. Learning in the presence of malicious errors, *SIAM J. Comput.* **22**(4) (1993), 807-837.
17. M. Kearns, L. Valiant. Cryptographic limitations on learning boolean formulae and finite automata, *J. ACM* **41**(1) (1994), 67-95. Also "Proc. 21st Symp. on Theor. of Comp." (1989), 433-444.
18. A. Klivans and R. Servedio. Boosting and hard-core sets, *in* "Proc. 40th Ann. Symp. on Found. of Comp. Sci." (1999), 624-633.
19. N. Littlestone. Redundant noisy attributes, attribute errors, and linear-threshold learning using winnow, *in* "Proc. Fourth Workshop on Computational Learning Theory," (1991), 147-156.
20. Y. Mansour and M. Parnas. Learning conjunctions with noise under product distributions, *Inf. Proc. Let.* **68**(4) (1998), 189-196.
21. R.E. Schapire. Theoretical views of boosting, *in* "Proc. 10th Int. Conf. on Algorithmic Learning Theory" (1999), 12-24.
22. R. Schapire, Y. Freund, P. Bartlett and W.S. Lee. Boosting the margin: a new explanation for the effectiveness of voting methods, *Annals of Statistics* **26**(5) (1998), 1651-1686.
23. R. Servedio. PAC analogues of perceptron and winnow via boosting the margin, *in* "Proc. 13th Conf. on Comp. Learning Theory," 2000.
24. J. Shawe-Taylor, P.L. Bartlett, R.C. Williamson and M. Anthony. Structural risk minimization over data-dependent hierarchies, *IEEE Trans. Inf. Theory,* **44**(5) (1998), 1926-1940.
25. A. Taylor and W. Mann. *Advanced Calculus,* Wiley & Sons, 1972.
26. L.G. Valiant. Learning disjunctions of conjunctions, *in* "Proc. 9th Internat. Joint Conf. on Artif. Intel." (1985), 560-566.

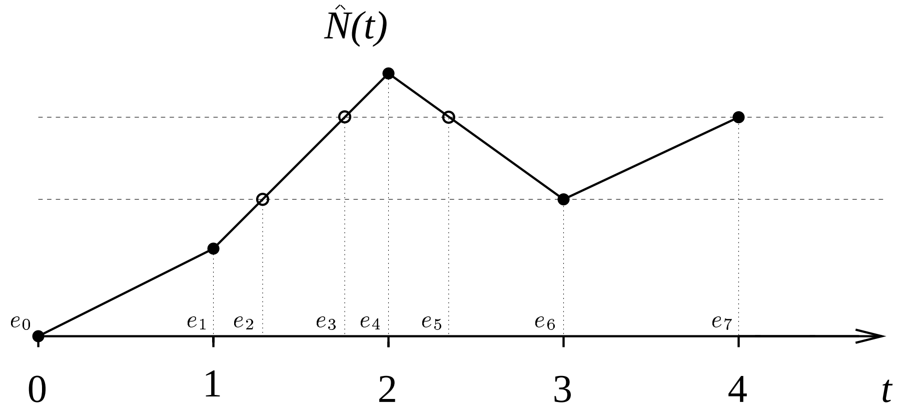

Fig. 3. A plot of $\hat{N}$ with $T = 4$. Note that $\hat{N}$ is piecewise linear with joins at integer values of t. A possible pairing of segments matches $[e_2, e_3]$ with $[e_5, e_6]$ and $[e_3, e_4]$ with $[e_4, e_5]$, leaving $[e_0, e_1]$, $[e_1, e_2]$ and $[e_6, e_7]$ unpaired. In this example $\hat{N}$ is increasing on each unpaired segment.

A Proof of Lemma 3

By the definition of $N_t(j)$, we have

$$\begin{aligned}\sum_{j=1}^{m}\sum_{t=1}^{T} M_t(j) y_j h_t(x^j) &= \sum_{j=1}^{m}\sum_{t=1}^{T} M_t(j)(N_t(j) - N_{t-1}(j) + \theta)\\ &= \theta\sum_{t=1}^{T}|M_t| + \sum_{t=1}^{T}\sum_{j=1}^{m} M_t(j)(N_t(j) - N_{t-1}(j)). \quad (2)\end{aligned}$$

It thus suffices to show that if $\theta = \frac{\gamma}{2+\gamma}$, then for each $j = 1, \ldots, m$ we have

$$\sum_{t=1}^{T} M_t(j)(N_t(j) - N_{t-1}(j)) < \frac{2}{\gamma\sqrt{1-\gamma}} + (\gamma - \theta)\sum_{t=1}^{T} M_t(j) \quad (3)$$

since summing this inequality over $j = 1, \ldots, m$ and substituting into (2) proves the lemma. Fix any $j \in \{1, \ldots, m\}$; for ease of notation we write N_t and M_t in place of $N_t(j)$ and $M_t(j)$ for the rest of the proof.

If $N_t = N_{t-1}$ for some integer t then the term $M_t(N_t - N_{t-1})$ contributes 0 to the sum in (3), so without loss of generality we assume that $N_t \neq N_{t-1}$ for all integers t. We extend the sequence $(N_0, N_1, \ldots, N_T)$ to a continuous piecewise linear function $\hat{N}$ on $[0, T]$ in the obvious way, i.e. for t an integer and $\epsilon \in [0, 1]$ we have $\hat{N}(t + \epsilon) = N_t + \epsilon(N_{t+1} - N_t)$. Let

$$E = \{e \in [0, T] : \hat{N}(e) = N_t \text{ for some integer } t = 0, 1, \ldots, T\}.$$

The set E is finite so we have $0 = e_0 < e_1 \cdots < e_r = T$ with $E = \{e_0, \ldots, e_r\}$ (see Figure 3). Since for each integer $t \geq 1$ the interval $(t-1, t]$ must contain some e_i, we can reexpress the sum $\sum_{t=1}^{T} M_t(N_t - N_{t-1})$ as

$$\sum_{i=1}^{r} M_{\lceil e_i \rceil} \left(\hat{N}(e_i) - \hat{N}(e_{i-1})\right). \tag{4}$$

We say that two segments $[e_{a-1}, e_a]$ and $[e_{b-1}, e_b]$ *match* if $\hat{N}(e_{a-1}) = \hat{N}(e_b)$ and $\hat{N}(e_{b-1}) = \hat{N}(e_a)$. For example, in Figure 3 the segment $[e_2, e_3]$ matches $[e_5, e_6]$ but does not match $[e_6, e_7]$. We pair up matching segments until no more pairs can be formed. Note that if any unpaired segments remain, it must be the case that either $\hat{N}$ is increasing on each unpaired segment (if $N_T > 0$) or $\hat{N}$ is decreasing on each unpaired segment (if $N_T < 0$). Now we separate the sum (4) into two pieces, i.e. $\sum_{i=1}^{r} M_{\lceil e_i \rceil}(\hat{N}(e_i) - \hat{N}(e_{i-1})) = P + U$, where P is the sum over all paired segments and U is the sum over all unpaired segments. We will show that $P < (\gamma - \theta)\sum_{t=1}^{T} M_t$ and $U < \frac{2}{\gamma\sqrt{1-\gamma}}$, thus proving the lemma.

First we bound P. Let $[e_{a-1}, e_a]$ and $[e_{b-1}, e_b]$ be a pair of matching segments where $\hat{N}$ is increasing on $[e_{a-1}, e_a]$ and decreasing on $[e_{b-1}, e_b]$. The contribution of these two segments to P is

$$\begin{aligned} M_{\lceil e_a \rceil} \left(\hat{N}(e_a) - \hat{N}(e_{a-1})\right) &+ M_{\lceil e_b \rceil} \left(\hat{N}(e_b) - \hat{N}(e_{b-1})\right) \\ &= (M_{\lceil e_a \rceil} - M_{\lceil e_b \rceil}) \left(\hat{N}(e_a) - \hat{N}(e_{a-1})\right). \end{aligned} \tag{5}$$

Since each segment $[e_{a-1}, e_a]$ is contained in $[t-1, t]$ for some integer t, we have that $\lceil e_a \rceil - 1 \leq e_{a-1} < e_a \leq \lceil e_a \rceil$. The linearity of $\hat{N}$ on $[\lceil e_a \rceil - 1, \lceil e_a \rceil]$ implies that

$$N_{\lceil e_a \rceil - 1} \leq \hat{N}(e_{a-1}) < \hat{N}(e_a) \leq N_{\lceil e_a \rceil} \leq N_{\lceil e_a \rceil - 1} + 1 - \theta \tag{6}$$

where the last inequality is because $y_j h_t(x^j) \leq 1$ in line 7 of `SmoothBoost`. Similarly, we have that $\lceil e_b \rceil - 1 \leq e_{b-1} < e_b \leq \lceil e_b \rceil$, and hence

$$N_{\lceil e_b \rceil - 1} \geq \hat{N}(e_{b-1}) > \hat{N}(e_b) \geq N_{\lceil e_b \rceil} \geq N_{\lceil e_b \rceil - 1} - 1 - \theta. \tag{7}$$

Since $\hat{N}(e_a) = \hat{N}(e_{b-1})$ inequalities (6) and (7) imply that $N_{\lceil e_a \rceil - 1} \geq N_{\lceil e_b \rceil - 1} - 2$. The definition of M now implies that $M_{\lceil e_b \rceil} \geq (1-\gamma) M_{\lceil e_a \rceil}$. Since $\hat{N}(e_a) - \hat{N}(e_{a-1}) > 0$, we thus have that (5) is at most

$$\gamma M_{\lceil e_a \rceil} \left(\hat{N}(e_a) - \hat{N}(e_{a-1})\right) \leq \gamma(1-\theta) M_{\lceil e_a \rceil}(e_a - e_{a-1}) \tag{8}$$

where the inequality follows from (6) and the linearity of $\hat{N}$ on $[e_{a-1}, e_a]$. Since $\hat{N}(e_a) - \hat{N}(e_{a-1}) = \hat{N}(e_{b-1}) - \hat{N}(e_b)$, we similarly have that (5) is at most

$$\begin{aligned} \gamma M_{\lceil e_a \rceil} \left(\hat{N}(e_{b-1}) - \hat{N}(e_b)\right) &\leq \frac{\gamma}{1-\gamma} M_{\lceil e_b \rceil} \left(\hat{N}(e_{b-1}) - \hat{N}(e_b)\right) \\ &\leq \frac{\gamma}{1-\gamma}(1+\theta) M_{\lceil e_b \rceil}(e_{b-1} - e_b). \end{aligned} \tag{9}$$

Using the fact that $\theta = \frac{\gamma}{2+\gamma}$ and some algebra, inequalities (8) and (9) imply that (5) is at most

$$\frac{\gamma(1+\gamma)}{2+\gamma}\left(M_{\lceil e_a \rceil}(e_a - e_{a-1}) + M_{\lceil e_b \rceil}(e_{b-1} - e_b)\right). \tag{10}$$

If we sum (10) over all pairs of matching segments the resulting quantity is an upper bound on P. In this sum, for each value of $t = 1, \ldots, T$, the coefficient of M_t will be at most $\frac{\gamma(1+\gamma)}{2+\gamma} = \gamma - \theta$. (This bound on the coefficient of M_t holds because for each t, the total length of all paired segments in $[t-1, t]$ is at most 1). Consequently we have $P < (\gamma - \theta)\sum_{t=1}^{T} M_t$ as desired.

Now we show that U, the sum over unpaired segments, is at most $\frac{2}{\gamma\sqrt{1-\gamma}}$. If $\hat{N}$ is decreasing on each unpaired segment then clearly $U < 0$, so we suppose that $\hat{N}$ is increasing on each unpaired segment. Let $[e_{c_1-1}, e_{c_1}], \ldots, [e_{c_d-1}, e_{c_d}]$ be all the unpaired segments. As in Figure 2 it must be the case that the intervals $[\hat{N}(e_{c_i-1}), \hat{N}(e_{c_i}))$ are all disjoint and their union is $[0, N_T)$. By the definition of M, we have $U = \sum_{i=1}^{d}(1-\gamma)^{(N_{\lceil e_{c_i} \rceil - 1})/2}\left(\hat{N}(e_{c_i}) - \hat{N}(e_{c_i-1})\right)$. As in the bound for P, we have

$$N_{\lceil e_{c_i} \rceil - 1} \le \hat{N}(e_{c_i-1}) < \hat{N}(e_{c_i}) \le N_{\lceil e_{c_i} \rceil} \le N_{\lceil e_{c_i} \rceil - 1} + 1 - \theta < N_{\lceil e_{c_i} \rceil - 1} + 1$$

and hence

$$\begin{aligned} U &\le \sum_{i=1}^{d}(1-\gamma)^{(\hat{N}(e_{c_i})-1)/2}\left(\hat{N}(e_{c_i}) - \hat{N}(e_{c_i-1})\right) \\ &= (1-\gamma)^{-1/2}\sum_{i=1}^{d}(1-\gamma)^{\hat{N}(e_{c_i})/2}\left(\hat{N}(e_{c_i}) - \hat{N}(e_{c_i-1})\right). \end{aligned}$$

Since $\hat{N}$ is increasing, for each i we have

$$(1-\gamma)^{\hat{N}(e_{c_i})/2}\left(\hat{N}(e_{c_i}) - \hat{N}(e_{c_i-1})\right) < \int_{z=\hat{N}(e_{c_i-1})}^{\hat{N}(e_{c_i})}(1-\gamma)^{z/2}dz.$$

Since the disjoint intervals $[\hat{N}(e_{c_i-1}), \hat{N}(e_{c_i}))$ cover $[0, N_T)$ we thus have

$$\begin{aligned} U &< (1-\gamma)^{-1/2}\int_{z=0}^{N_T}(1-\gamma)^{z/2}dz \\ &< (1-\gamma)^{-1/2}\int_{z=0}^{\infty}(1-\gamma)^{z/2}dz \\ &= \frac{-2}{\sqrt{1-\gamma}\ln(1-\gamma)} \quad < \quad \frac{2}{\gamma\sqrt{1-\gamma}} \qquad \text{for } 0 < \gamma < 1/2. \end{aligned}$$

(Lemma 3) ■