Deep Boosting

Corinna Cortes CORINNA@GOOGLE.COM
Google Research, 111 8th Avenue, New York, NY 10011

Mehryar Mohri MOHRI@CIMS.NYU.EDU
Courant Institute and Google Research, 251 Mercer Street, New York, NY 10012

Umar Syed USYED@GOOGLE.COM
Google Research, 111 8th Avenue, New York, NY 10011

Abstract

We present a new ensemble learning algorithm, *DeepBoost*, which can use as base classifiers a hypothesis set containing deep decision trees, or members of other rich or complex families, and succeed in achieving high accuracy without overfitting the data. The key to the success of the algorithm is a *capacity-conscious* criterion for the selection of the hypotheses. We give new data-dependent learning bounds for convex ensembles expressed in terms of the Rademacher complexities of the sub-families composing the base classifier set, and the mixture weight assigned to each sub-family. Our algorithm directly benefits from these guarantees since it seeks to minimize the corresponding learning bound. We give a full description of our algorithm, including the details of its derivation, and report the results of several experiments showing that its performance compares favorably to that of AdaBoost and Logistic Regression and their L_1-regularized variants.

1. Introduction

Ensemble methods are general techniques in machine learning for combining several predictors or experts to create a more accurate one. In the batch learning setting, techniques such as bagging, boosting, stacking, error-correction techniques, Bayesian averaging, or other averaging schemes are prominent instances of these methods (Breiman, 1996; Freund & Schapire, 1997; Smyth & Wolpert, 1999; MacKay, 1991; Freund et al., 2004). Ensemble methods often significantly improve performance in practice (Quinlan, 1996; Bauer & Kohavi, 1999; Caruana et al., 2004; Dietterich, 2000; Schapire, 2003) and benefit from favorable learning guarantees. In particular, AdaBoost and its variants are based on a rich theoretical analysis, with performance guarantees in terms of the margins of the training samples (Schapire et al., 1997; Koltchinskii & Panchenko, 2002).

Standard ensemble algorithms such as AdaBoost combine functions selected from a base classifier hypothesis set H. In many successful applications of AdaBoost, H is reduced to the so-called *boosting stumps*, that is decision trees of depth one. For some difficult tasks in speech or image processing, simple boosting stumps are not sufficient to achieve a high level of accuracy. It is tempting then to use a more complex hypothesis set, for example the set of all decision trees with depth bounded by some relatively large number. But, existing learning guarantees for AdaBoost depend not only on the margin and the number of the training examples, but also on the complexity of H measured in terms of its VC-dimension or its Rademacher complexity (Schapire et al., 1997; Koltchinskii & Panchenko, 2002). These learning bounds become looser when using too complex base classifier sets H. They suggest a risk of overfitting which indeed can be observed in some experiments with AdaBoost (Grove & Schuurmans, 1998; Schapire, 1999; Dietterich, 2000; Rätsch et al., 2001b).

This paper explores the design of alternative ensemble algorithms using as base classifiers a hypothesis set H that may contain very deep decision trees, or members of some other very rich or complex families, and that can yet succeed in achieving a higher performance level. Assume that the set of base classifiers H can be decomposed as the union of p disjoint families $H_1, \ldots, H_p$ ordered by increasing complexity, where H_k, $k \in [1, p]$, could be for example the set of decision trees of depth k, or a set of functions based on monomials of degree k. Figure 1 shows a pictorial illustration. Of course, if we strictly confine ourselves to using hypotheses belonging only to families H_k with small k, then we are effectively using a smaller base classifier set H with favorable guarantees. But, to succeed in some chal-

Proceedings of the 31[st] *International Conference on Machine Learning*, Beijing, China, 2014. JMLR: W&CP volume 32.

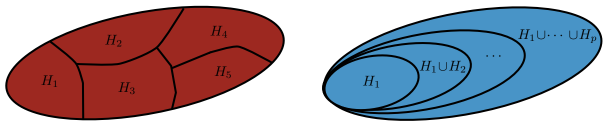


Figure 1. Base classifier set H decomposed in terms of sub-families $H_1, \ldots, H_p$ or their unions.

lenging tasks, the use of a few more complex hypotheses could be needed. The main idea behind the design of our algorithms is that an ensemble based on hypotheses drawn from $H_1, \ldots, H_p$ can achieve a higher accuracy by making use of hypotheses drawn from H_ks with large k if it allocates more weights to hypotheses drawn from H_ks with a small k. But, can we determine quantitatively the amounts of mixture weights apportioned to different families? Can we provide learning guarantees for such algorithms?

Note that our objective is somewhat reminiscent of that of model selection, in particular Structural Risk Minimization (SRM) (Vapnik, 1998), but it differs from that in that we do not wish to limit our base classifier set to some *optimal* $\mathcal{H}_q = \bigcup_{k=1}^{q} H_k$. Rather, we seek the freedom of using as base hypotheses even relatively deep trees from rich H_ks, with the promise of doing so infrequently, or that of reserving them a somewhat small weight contribution. This provides the flexibility of learning with deep hypotheses.

We present a new algorithm, *DeepBoost*, whose design is precisely guided by the ideas just discussed. Our algorithm is grounded in a solid theoretical analysis that we present in Section 2. We give new data-dependent learning bounds for convex ensembles. These guarantees are expressed in terms of the Rademacher complexities of the sub-families H_k and the mixture weight assigned to each H_k, in addition to the familiar margin terms and sample size. Our *capacity-conscious* algorithm is derived via the application of a coordinate descent technique seeking to minimize such learning bounds. We give a full description of our algorithm, including the details of its derivation and its pseudocode (Section 3) and discuss its connection with previous boosting-style algorithms. We also report the results of several experiments (Section 4) demonstrating that its performance compares favorably to that of AdaBoost, which is known to be one of the most competitive binary classification algorithms.

2. Data-dependent learning guarantees for convex ensembles with multiple hypothesis sets

Non-negative linear combination ensembles such as boosting or bagging typically assume that base functions are selected from the same hypothesis set H. Margin-based generalization bounds were given for ensembles of base functions taking values in $\{-1, +1\}$ by Schapire et al. (1997) in terms of the VC-dimension of H. Tighter margin bounds with simpler proofs were later given by Koltchinskii & Panchenko (2002), see also (Bartlett & Mendelson, 2002), for the more general case of a family H taking arbitrary real values, in terms of the Rademacher complexity of H.

Here, we also consider base hypotheses taking arbitrary real values but assume that they can be selected from *several* distinct hypothesis sets $H_1, \ldots, H_p$ with $p \geq 1$ and present margin-based learning in terms of the Rademacher complexity of these sets. Remarkably, the complexity term of these bounds admits an explicit dependency in terms of the mixture coefficients defining the ensembles. Thus, the ensemble family we consider is $\mathcal{F} = \operatorname{conv}(\bigcup_{k=1}^{p} H_k)$, that is the family of functions f of the form $f = \sum_{t=1}^{T} \alpha_t h_t$, where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_T)$ is in the simplex Δ and where, for each $t \in [1, T]$, h_t is in H_{k_t} for some $k_t \in [1, p]$.

Let $\mathcal{X}$ denote the input space. $H_1, \ldots, H_p$ are thus families of functions mapping from $\mathcal{X}$ to $\mathbb{R}$. We consider the familiar supervised learning scenario and assume that training and test points are drawn i.i.d. according to some distribution $\mathcal{D}$ over $\mathcal{X} \times \{-1, +1\}$ and denote by $S = ((x_1, y_1), \ldots, (x_m, y_m))$ a training sample of size m drawn according to $\mathcal{D}^m$.

Let $\rho > 0$. For a function f taking values in $\mathbb{R}$, we denote by $R(f)$ its binary classification error, by $R_\rho(f)$ its ρ-margin error, and by $\widehat{R}_{S,\rho}(f)$ its empirical margin error:

$$R(f) = \operatorname*{E}_{(x,y)\sim\mathcal{D}}[1_{yf(x)\leq 0}], \quad R_\rho(f) = \operatorname*{E}_{(x,y)\sim\mathcal{D}}[1_{yf(x)\leq \rho}],$$
$$\widehat{R}_\rho(f) = \operatorname*{E}_{(x,y)\sim S}[1_{yf(x)\leq \rho}],$$

where the notation $(x, y) \sim S$ indicates that (x, y) is drawn according to the empirical distribution defined by S.

The following theorem gives a margin-based Rademacher complexity bound for learning with such functions in the binary classification case. As with other Rademacher complexity learning guarantees, our bound is data-dependent, which is an important and favorable characteristic of our results. For $p = 1$, that is for the special case of a single hypothesis set, the analysis coincides with that of the standard ensemble margin bounds (Koltchinskii & Panchenko, 2002).

Theorem 1. *Assume $p > 1$. Fix $\rho > 0$. Then, for any $\delta > 0$, with probability at least $1 - \delta$ over the choice of a sample S of size m drawn i.i.d. according to $\mathcal{D}^m$, the following inequality holds for all $f = \sum_{t=1}^{T} \alpha_t h_t \in \mathcal{F}$:*

$$R(f) \leq \widehat{R}_{S,\rho}(f) + \frac{4}{\rho} \sum_{t=1}^{T} \alpha_t \mathfrak{R}_m(H_{k_t})$$
$$+ \frac{2}{\rho}\sqrt{\frac{\log p}{m}} + \sqrt{\left\lceil \frac{4}{\rho^2} \log\left[\frac{\rho^2 m}{\log p}\right]\right\rceil \frac{\log p}{m} + \frac{\log \frac{2}{\delta}}{2m}}.$$

Thus, $R(f) \leq \widehat{R}_{S,\rho}(f) + \frac{4}{\rho} \sum_{t=1}^{T} \alpha_t \mathfrak{R}_m(H_{k_t}) + C(m,p)$ *with* $C(m,p) = O\left(\sqrt{\frac{\log p}{\rho^2 m} \log\left[\frac{\rho^2 m}{\log p}\right]}\right)$.

This result is remarkable since the complexity term in the right-hand side of the bound admits an explicit dependency on the mixture coefficients α_t. It is a weighted average of Rademacher complexities with mixture weights α_t, $t \in [1,T]$. Thus, the second term of the bound suggests that, while some hypothesis sets H_k used for learning could have a large Rademacher complexity, this may not be detrimental to generalization if the corresponding total mixture weight (sum of α_ts corresponding to that hypothesis set) is relatively small. Such complex families offer the potential of achieving a better margin on the training sample.

The theorem cannot be proven via a standard Rademacher complexity analysis such as that of Koltchinskii & Panchenko (2002) since the complexity term of the bound would then be the Rademacher complexity of the family of hypotheses $\mathcal{F} = \operatorname{conv}(\bigcup_{k=1}^{p} H_k)$ and would not depend on the specific weights α_t defining a given function f. Furthermore, the complexity term of a standard Rademacher complexity analysis is always lower bounded by the complexity term appearing in our bound. Indeed, since $\mathfrak{R}_m(\operatorname{conv}(\cup_{k=1}^{p} H_k)) = \mathfrak{R}_m(\cup_{k=1}^{p} H_k)$, the following lower bound holds for any choice of the non-negative mixtures weights α_t summing to one:

$$\mathfrak{R}_m(\mathcal{F}) \geq \max_{k=1}^{m} \mathfrak{R}_m(H_k) \geq \sum_{t=1}^{T} \alpha_t \mathfrak{R}_m(H_{k_t}). \tag{1}$$

Thus, Theorem 1 provides a finer learning bound than the one obtained via a standard Rademacher complexity analysis. The full proof of the theorem is given in Appendix A. Our proof technique exploits standard tools used to derive Rademacher complexity learning bounds (Koltchinskii & Panchenko, 2002) as well as a technique used by Schapire, Freund, Bartlett, and Lee (1997) to derive early VC-dimension margin bounds. Using other standard techniques as in (Koltchinskii & Panchenko, 2002; Mohri et al., 2012), Theorem 1 can be straightforwardly generalized to hold uniformly for all $\rho > 0$ at the price of an additional term that is in $O\left(\sqrt{\frac{\log \log \frac{2}{\rho}}{m}}\right)$.

3. Algorithm

In this section, we will use the learning guarantees of Section 2 to derive a *capacity-conscious* ensemble algorithm for binary classification.

3.1. Optimization problem

Let $H_1, \ldots, H_p$ be p disjoint families of functions taking values in $[-1,+1]$ with increasing Rademacher complexities $\mathfrak{R}_m(H_k)$, $k \in [1,p]$. We will assume that the hypothesis sets H_k are symmetric, that is, for any $h \in H_k$, we also have $(-h) \in H_k$, which holds for most hypothesis sets typically considered in practice. This assumption is not necessary but it helps simplifying the presentation of our algorithm. For any hypothesis $h \in \cup_{k=1}^{p} H_k$, we denote by $d(h)$ the index of the hypothesis set it belongs to, that is $h \in H_{d(h)}$. The bound of Theorem 1 holds uniformly for all $\rho > 0$ and functions $f \in \operatorname{conv}(\bigcup_{k=1}^{p} H_k)$.[1] Since the last term of the bound does not depend on $\boldsymbol{\alpha}$, it suggests selecting $\boldsymbol{\alpha}$ to minimize

$$G(\boldsymbol{\alpha}) = \frac{1}{m} \sum_{i=1}^{m} 1_{y_i \sum_{t=1}^{T} \alpha_t h_t(x_i) \leq \rho} + \frac{4}{\rho} \sum_{t=1}^{T} \alpha_t r_t,$$

where $r_t = \mathfrak{R}_m(H_{d(h_t)})$. Since for any $\rho > 0$, f and f/ρ admit the same generalization error, we can instead search for $\alpha \geq 0$ with $\sum_{t=1}^{T} \alpha_t \leq 1/\rho$ which leads to

$$\min_{\boldsymbol{\alpha} \geq 0} \frac{1}{m} \sum_{i=1}^{m} 1_{y_i \sum_{t=1}^{T} \alpha_t h_t(x_i) \leq 1} + 4 \sum_{t=1}^{T} \alpha_t r_t \ \text{ s.t.} \sum_{t=1}^{T} \alpha_t \leq \frac{1}{\rho}.$$

The first term of the objective is not a convex function of $\boldsymbol{\alpha}$ and its minimization is known to be computationally hard. Thus, we will consider instead a convex upper bound. Let $u \mapsto \Phi(-u)$ be a non-increasing convex function upper bounding $u \mapsto 1_{u \leq 0}$ with Φ differentiable over $\mathbb{R}$ and $\Phi'(u) \neq 0$ for all u. Φ may be selected to be for example the exponential function as in AdaBoost (Freund & Schapire, 1997) or the logistic function. Using such an upper bound, we obtain the following convex optimization problem:

$$\min_{\boldsymbol{\alpha} \geq 0} \frac{1}{m} \sum_{i=1}^{m} \Phi\Big(1 - y_i \sum_{t=1}^{T} \alpha_t h_t(x_i)\Big) + \lambda \sum_{t=1}^{T} \alpha_t r_t \tag{2}$$
$$\text{s.t.} \sum_{t=1}^{T} \alpha_t \leq \frac{1}{\rho},$$

where we introduced a parameter $\lambda \geq 0$ controlling the balance between the magnitude of the values taken by function Φ and the second term. Introducing a Lagrange variable $\beta \geq 0$ associated to the constraint in (2), the problem can be equivalently written as

$$\min_{\boldsymbol{\alpha} \geq 0} \frac{1}{m} \sum_{i=1}^{m} \Phi\Big(1 - y_i \sum_{t=1}^{T} \alpha_t h_t(x_i)\Big) + \sum_{t=1}^{T} (\lambda r_t + \beta) \alpha_t.$$

Here, β is a parameter that can be freely selected by the algorithm since any choice of its value is equivalent to a

[1] The condition $\sum_{t=1}^{T} \alpha_t = 1$ of Theorem 1 can be relaxed to $\sum_{t=1}^{T} \alpha_t \leq 1$. To see this, use for example a null hypothesis ($h_t = 0$ for some t).

choice of ρ in (2). Let $\{h_1, \ldots, h_N\}$ be the set of distinct base functions, and let G be the objective function based on that collection:

$$G(\boldsymbol{\alpha}) = \frac{1}{m}\sum_{i=1}^{m}\Phi\Big(1 - y_i\sum_{j=1}^{N}\alpha_j h_j(x_i)\Big) + \sum_{t=1}^{N}(\lambda r_j + \beta)\alpha_j,$$

with $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_N) \in \mathbb{R}^N$. Note that we can drop the requirement $\boldsymbol{\alpha} \geq 0$ since the hypothesis sets are symmetric and $\alpha_t h_t = (-\alpha_t)(-h_t)$. For each hypothesis h, we keep either h or $-h$ in $\{h_1, \ldots, h_N\}$. Using the notation

$$\Lambda_j = \lambda r_j + \beta, \tag{3}$$

for all $j \in [1, N]$, our optimization problem can then be rewritten as $\min_{\boldsymbol{\alpha}} F(\boldsymbol{\alpha})$ with

$$F(\boldsymbol{\alpha}) = \frac{1}{m}\sum_{i=1}^{m}\Phi\Big(1 - y_i\sum_{j=1}^{N}\alpha_j h_j(x_i)\Big) + \sum_{t=1}^{N}\Lambda_j|\alpha_j|, \tag{4}$$

with no non-negativity constraint on $\boldsymbol{\alpha}$. The function F is convex as a sum of convex functions and admits a sub-differential at all $\boldsymbol{\alpha} \in \mathbb{R}$. We can design a boosting-style algorithm by applying coordinate descent to $F(\boldsymbol{\alpha})$.

Let $\boldsymbol{\alpha}_t = (\alpha_{t,1}, \ldots, \alpha_{t,N})^\top$ denote the vector obtained after $t \geq 1$ iterations and let $\boldsymbol{\alpha}_0 = \mathbf{0}$. Let $\mathbf{e}_k$ denote the kth unit vector in $\mathbb{R}^N$, $k \in [1, N]$. The direction $\mathbf{e}_k$ and the step η selected at the tth round are those minimizing $F(\boldsymbol{\alpha}_{t-1} + \eta\mathbf{e}_k)$, that is

$$\begin{aligned} F(\boldsymbol{\alpha}_{t-1} + \eta\mathbf{e}_k) = \frac{1}{m}\sum_{i=1}^{m}\Phi\Big(1 - y_i f_{t-1}(x_i) - \eta y_i h_k(x_i)\Big) \\ + \sum_{j \neq k}\Lambda_j|\alpha_{t-1,j}| + \Lambda_k|\alpha_{t-1,k} + \eta|, \end{aligned}$$

where $f_{t-1} = \sum_{j=1}^{N}\alpha_{t-1,j}h_j$. For any $t \in [1, T]$, we denote by $\mathcal{D}_t$ the distribution defined by

$$\mathcal{D}_t(i) = \frac{\Phi'\big(1 - y_i f_{t-1}(x_i)\big)}{S_t}, \tag{5}$$

where S_t is a normalization factor, $S_t = \sum_{i=1}^{m}\Phi'(1 - y_i f_{t-1}(x_i))$. For any $s \in [1, T]$ and $j \in [1, N]$, we denote by $\epsilon_{s,j}$ the weighted error of hypothesis h_j for the distribution $\mathcal{D}_s$, for $s \in [1, T]$:

$$\epsilon_{s,j} = \frac{1}{2}\Big[1 - \mathop{\mathrm{E}}_{i \sim \mathcal{D}_s}[y_i h_j(x_i)]\Big]. \tag{6}$$

3.2. DeepBoost

Figure 2 shows the pseudocode of the algorithm *DeepBoost* derived by applying coordinate descent to the objective function (4). The details of the derivation of the expression are given in Appendix B. In the special cases of the exponential loss ($\Phi(-u) = \exp(-u)$) or the logistic loss ($\Phi(-u) = \log_2(1 + \exp(-u))$), a closed-form expression is given for the step size (lines 12-16), which is the same in both cases (see Sections B.4 and B.5). In the generic case, the step size η_t can be found using a line search or other numerical methods. Note that when the condition of line 12 is satisfied, the step taken by the algorithm cancels out the coordinate along the direction k, thereby leading to a sparser result. This is consistent with the fact that the objective function contains a second term based on (weighted) L_1-norm, which is favoring sparsity.

Our algorithm is related to several other boosting-type algorithms devised in the past. For $\lambda = 0$ and $\beta = 0$ and using the exponential surrogate loss, it coincides with AdaBoost (Freund & Schapire, 1997) with precisely the same direction and same step $\log\Big[\sqrt{\frac{1-\epsilon_t}{\epsilon_t}}\Big]$ using $\mathcal{H} = \bigcup_{k=1}^{p} H_k$ as the hypothesis set for base learners. This corresponds to

DeepBoost$(S = ((x_1, y_1), \ldots, (x_m, y_m)))$

1 **for** $i \leftarrow 1$ **to** m **do**
2 $\quad D_1(i) \leftarrow \frac{1}{m}$
3 **for** $t \leftarrow 1$ **to** T **do**
4 $\quad$ **for** $j \leftarrow 1$ **to** N **do**
5 $\quad\quad$ **if** $(\alpha_{t-1,j} \neq 0)$ **then**
6 $\quad\quad\quad d_j \leftarrow \big(\epsilon_{t,j} - \frac{1}{2}\big) + \mathrm{sgn}(\alpha_{t-1,j})\frac{\Lambda_j m}{2S_t}$
7 $\quad\quad$ **elseif** $\big(\big|\epsilon_{t,j} - \frac{1}{2}\big| \leq \frac{\Lambda_j m}{2S_t}\big)$ **then**
8 $\quad\quad\quad d_j \leftarrow 0$
9 $\quad\quad$ **else** $d_j \leftarrow \big(\epsilon_{t,j} - \frac{1}{2}\big) - \mathrm{sgn}(\epsilon_{t,j} - \frac{1}{2})\frac{\Lambda_j m}{2S_t}$
10 $\quad k \leftarrow \mathop{\mathrm{argmax}}_{j \in [1,N]} |d_j|$
11 $\quad \epsilon_t \leftarrow \epsilon_{t,k}$
12 $\quad$ **if** $\big(\big|(1 - \epsilon_t)e^{\alpha_{t-1,k}} - \epsilon_t e^{-\alpha_{t-1,k}}\big| \leq \frac{\Lambda_k m}{S_t}\big)$ **then**
13 $\quad\quad \eta_t \leftarrow -\alpha_{t-1,k}$
14 $\quad$ **elseif** $\big((1 - \epsilon_t)e^{\alpha_{t-1,k}} - \epsilon_t e^{-\alpha_{t-1,k}} > \frac{\Lambda_k m}{S_t}\big)$ **then**
15 $\quad\quad \eta_t \leftarrow \log\Big[-\frac{\Lambda_k m}{2\epsilon_t S_t} + \sqrt{\big[\frac{\Lambda_k m}{2\epsilon_t S_t}\big]^2 + \frac{1-\epsilon_t}{\epsilon_t}}\Big]$
16 $\quad$ **else** $\eta_t \leftarrow \log\Big[+\frac{\Lambda_k m}{2\epsilon_t S_t} + \sqrt{\big[\frac{\Lambda_k m}{2\epsilon_t S_t}\big]^2 + \frac{1-\epsilon_t}{\epsilon_t}}\Big]$
17 $\quad \boldsymbol{\alpha}_t \leftarrow \boldsymbol{\alpha}_{t-1} + \eta_t\mathbf{e}_k$
18 $\quad S_{t+1} \leftarrow \sum_{i=1}^{m}\Phi'\big(1 - y_i\sum_{j=1}^{N}\alpha_{t,j}h_j(x_i)\big)$
19 $\quad$ **for** $i \leftarrow 1$ **to** m **do**
20 $\quad\quad D_{t+1}(i) \leftarrow \frac{\Phi'\big(1 - y_i\sum_{j=1}^{N}\alpha_{t,j}h_j(x_i)\big)}{S_{t+1}}$
21 $f \leftarrow \sum_{j=1}^{N}\alpha_{T,j}h_j$
22 **return** f

Figure 2. Pseudocode of the DeepBoost algorithm for both the exponential loss and the logistic loss. The expression of the weighted error $\epsilon_{t,j}$ is given in (6). In the generic case of a surrogate loss Φ different from the exponential or logistic losses, η_t is found instead via a line search or other numerical methods from $\eta_t = \mathrm{argmax}_\eta F(\boldsymbol{\alpha}_{t-1} + \eta\mathbf{e}_k)$.

ignoring the complexity term of our bound as well as the control of the sum of the mixture weights via β. For $\lambda = 0$ and $\beta = 0$ and using the logistic surrogate loss, our algorithm also coincides with additive logistic loss (Friedman et al., 1998).

In the special case where $\lambda = 0$ and $\beta \neq 0$ and for the exponential surrogate loss, our algorithm matches the L_1-norm regularized AdaBoost (e.g., see (Rätsch et al., 2001a)). For the same choice of the parameters and for the logistic surrogate loss, our algorithm matches the L_1-norm regularized additive Logistic Regression studied by Duchi & Singer (2009) using the base learner hypothesis set $\mathcal{H} = \bigcup_{k=1}^{p} H_k$. $\mathcal{H}$ may in general be very rich. The key foundation of our algorithm and analysis is instead to take into account the relative complexity of the sub-families H_k. Also, note that L_1-norm regularized AdaBoost and Logistic Regression can be viewed as algorithms minimizing the learning bound obtained via the standard Rademacher complexity analysis (Koltchinskii & Panchenko, 2002), using the exponential or logistic surrogate losses. Instead, the objective function minimized by our algorithm is based on the generalization bound of Theorem 1, which as discussed earlier is a finer bound (see (1)). For $\lambda = 0$ but $\beta \neq 0$, our algorithm is also close to the so-called *unnormalized Arcing* (Breiman, 1999) or AdaBoost$_\rho$ (Rätsch & Warmuth, 2002) using $\mathcal{H}$ as a hypothesis set. AdaBoost$_\rho$ coincides with AdaBoost modulo the step size, which is more conservative than that of AdaBoost and depends on ρ. Rätsch & Warmuth (2005) give another variant of the algorithm that does not require knowing the best ρ, see also the related work of Kivinen & Warmuth (1999); Warmuth et al. (2006).

Our algorithm directly benefits from the learning guarantees given in Section 2 since it seeks to minimize the bound of Theorem 1. In the next section, we report the results of our experiments with DeepBoost. Let us mention that we have also designed an alternative deep boosting algorithm that we briefly describe and discuss in Appendix C.

4. Experiments

An additional benefit of the learning bounds presented in Section 2 is that they are data-dependent. They are based on the Rademacher complexity of the base hypothesis sets H_k, which in some cases can be well estimated from the training sample. The algorithm DeepBoost directly inherits this advantage. For example, if the hypothesis set H_k is based on a positive definite kernel with sample matrix $\mathbf{K}_k$, it is known that its empirical Rademacher complexity can be upper bounded by $\frac{\sqrt{\mathrm{Tr}[\mathbf{K}_k]}}{m}$ and lower bounded by $\frac{1}{\sqrt{2}} \frac{\sqrt{\mathrm{Tr}[\mathbf{K}_k]}}{m}$. In other cases, when H_k is a family of functions taking binary values, we can use an upper bound on the Rademacher complexity in terms of the growth function of H_k, $\Pi_{H_k}(m)$: $\mathfrak{R}_m(H_k) \leq \sqrt{\frac{2 \log \Pi_{H_k}(m)}{m}}$. Thus, for the family H_1^{stumps} of boosting stumps in dimension d, $\Pi_{H_1^{\text{stumps}}}(m) \leq 2md$, since there are $2m$ distinct threshold functions for each dimension with m points. Thus, the following inequality holds:

$$\mathfrak{R}_m(H_1^{\text{stumps}}) \leq \sqrt{\frac{2 \log(2md)}{m}}. \tag{7}$$

Similarly, we consider the family of decision trees H_2^{stumps} of depth 2 with the same question at the internal nodes of depth 1. We have $\Pi_{H_2^{\text{stumps}}}(m) \leq (2m)^2 \frac{d(d-1)}{2}$ since there are $d(d-1)/2$ distinct trees of this type and since each induces at most $(2m)^2$ labelings. Thus, we can write

$$\mathfrak{R}_m(H_2^{\text{stumps}}) \leq \sqrt{\frac{2 \log(2m^2 d(d-1))}{m}}. \tag{8}$$

More generally, we also consider the family of all binary decision trees H_k^{trees} of depth k. For this family it is known that $\text{VC-dim}(H_k^{\text{trees}}) \leq (2^k + 1) \log_2(d+1)$ (Mansour, 1997). More generally, the VC-dimension of $\mathcal{T}_n$, the family of decision trees with n nodes in dimension d can be bounded by $(2n+1)\log_2(d+2)$ (see for example (Mohri et al., 2012)). Since $\mathfrak{R}_m(H) \leq \sqrt{\frac{2\,\text{VC-dim}(H) \log(m+1)}{m}}$, for any hypothesis class H we have

$$\mathfrak{R}_m(\mathcal{T}_n) \leq \sqrt{\frac{(4n+2)\log_2(d+2)\log(m+1)}{m}}. \tag{9}$$

The experiments with DeepBoost described below use either $\mathcal{H}^{\text{stumps}} = H_1^{\text{stumps}} \cup H_2^{\text{stumps}}$ or $\mathcal{H}_K^{\text{trees}} = \bigcup_{k=1}^{K} H_k^{\text{trees}}$, for some $K > 0$, as the base hypothesis sets. For any hypothesis in these sets, DeepBoost will use the upper bounds given above as a proxy for the Rademacher complexity of the set to which it belongs. We leave it to the future to experiment with finer data-dependent estimates or upper bounds on the Rademacher complexity, which could further improve the performance of our algorithm. Recall that each iteration of DeepBoost searches for the base hypothesis that is optimal with respect to a certain criterion (see lines 5-10 of Figure 2). While an exhaustive search is feasible for H_1^{stumps}, it would be far too expensive to visit all trees in $\mathcal{H}_K^{\text{trees}}$ when K is large. Therefore, when using $\mathcal{H}_K^{\text{trees}}$ and also H_2^{stumps} as the base hypotheses we use the following heuristic search procedure in each iteration t: First, the optimal tree $h_1^* \in H_1^{\text{trees}}$ is found via exhaustive search. Next, for all $1 < k \leq K$, a locally optimal tree $h_k^* \in H_k^{\text{trees}}$ is found by considering only trees that can be obtained by adding a single layer of leaves to h_{k-1}^*. Finally, we select the best hypotheses in the set $\{h_1^*, \ldots, h_K^*, h_1, \ldots, h_{t-1}\}$, where $h_1, \ldots, h_{t-1}$ are the hypotheses selected in previous iterations.

Table 1. Results for boosted decision stumps and the exponential loss function.

breastcancer	AdaBoost H_1^{stumps}	AdaBoost H_2^{stumps}	AdaBoost-L1	DeepBoost
Error	0.0429	0.0437	0.0408	**0.0373**
(std dev)	(0.0248)	(0.0214)	(0.0223)	**(0.0225)**
Avg tree size	1	2	1.436	1.215
Avg no. of trees	100	100	43.6	21.6

ocr17	AdaBoost H_1^{stumps}	AdaBoost H_2^{stumps}	AdaBoost-L1	DeepBoost
Error	0.0085	0.008	0.0075	**0.0070**
(std dev)	0.0072	0.0054	0.0068	**(0.0048)**
Avg tree size	1	2	1.086	1.369
Avg no. of trees	100	100	37.8	36.9

ionosphere	AdaBoost H_1^{stumps}	AdaBoost H_2^{stumps}	AdaBoost-L1	DeepBoost
Error	0.1014	0.075	0.0708	**0.0638**
(std dev)	(0.0414)	(0.0413)	(0.0331)	**(0.0394)**
Avg tree size	1	2	1.392	1.168
Avg no. of trees	100	100	39.35	17.45

ocr49	AdaBoost H_1^{stumps}	AdaBoost H_2^{stumps}	AdaBoost-L1	DeepBoost
Error	0.0555	0.032	0.03	**0.0275**
(std dev)	0.0167	0.0114	0.0122	**(0.0095)**
Avg tree size	1	2	1.99	1.96
Avg no. of trees	100	100	99.3	96

german	AdaBoost H_1^{stumps}	AdaBoost H_2^{stumps}	AdaBoost-L1	DeepBoost
Error	0.243	0.2505	0.2455	**0.2395**
(std dev)	(0.0445)	(0.0487)	(0.0438)	**(0.0462)**
Avg tree size	1	2	1.54	1.76
Avg no. of trees	100	100	54.1	76.5

ocr17-mnist	AdaBoost H_1^{stumps}	AdaBoost H_2^{stumps}	AdaBoost-L1	DeepBoost
Error	0.0056	0.0048	0.0046	**0.0040**
(std dev)	0.0017	0.0014	0.0013	**(0.0014)**
Avg tree size	1	2	2	1.99
Avg no. of trees	100	100	100	100

diabetes	AdaBoost H_1^{stumps}	AdaBoost H_2^{stumps}	AdaBoost-L1	DeepBoost
Error	0.253	0.260	0.254	**0.253**
(std dev)	(0.0330)	(0.0518)	(0.04868)	**(0.0510)**
Avg tree size	1	2	1.9975	1.9975
Avg no. of trees	100	100	100	100

ocr49-mnist	AdaBoost H_1^{stumps}	AdaBoost H_2^{stumps}	AdaBoost-L1	DeepBoost
Error	0.0414	0.0209	0.0200	**0.0177**
(std dev)	0.00539	0.00521	0.00408	**(0.00438)**
Avg tree size	1	2	1.9975	1.9975
Avg no. of trees	100	100	100	100

Breiman (1999) and Reyzin & Schapire (2006) extensively investigated the relationship between the complexity of decision trees in an ensemble learned by AdaBoost and the generalization error of the ensemble. We tested DeepBoost on the same UCI datasets used by these authors, `http://archive.ics.uci.edu/ml/datasets.html`, specifically `breastcancer`, `ionosphere`, `german (numeric)` and `diabetes`. We also experimented with two optical character recognition datasets used by Reyzin & Schapire (2006), `ocr17` and `ocr49`, which contain the handwritten digits 1 and 7, and 4 and 9 (respectively). Finally, because these OCR datasets are fairly small, we also constructed the analogous datasets from all of MNIST, `http://yann.lecun.com/exdb/mnist/`, which we call `ocr17-mnist` and `ocr49-mnist`. More details on all the datasets are given in Table 4, Appendix D.1.

As we discussed in Section 3.2, by fixing the parameters β and λ to certain values, we recover some known algorithms as special cases of DeepBoost. Our experiments compared DeepBoost to AdaBoost ($\beta = \lambda = 0$ with exponential loss), to Logistic Regression ($\beta = \lambda = 0$ with logistic loss), which we abbreviate as LogReg, to L_1-norm regularized AdaBoost (e.g., see (Rätsch et al., 2001a)) abbreviated as AdaBoost-L1, and also to the L_1-norm regularized additive Logistic Regression algorithm studied by (Duchi & Singer, 2009) ($\beta > 0, \lambda = 0$) abbreviated as LogReg-L1.

In the first set of experiments reported in Table 1, we compared AdaBoost, AdaBoost-L1, and DeepBoost with the exponential loss ($\Phi(-u) = \exp(-u)$) and base hypotheses $\mathcal{H}^{\text{stumps}}$. We tested standard AdaBoost with base hypotheses H_1^{stumps} and H_2^{stumps}. For AdaBoost-L1, we optimized over $\beta \in \{2^{-i} : i = 6, \ldots, 0\}$ and for DeepBoost, we optimized over β in the same range and $\lambda \in \{0.0001, 0.005, 0.01, 0.05, 0.1, 0.5\}$. The exact parameter optimization procedure is described below.

In the second set of experiments reported in Table 2, we used base hypotheses $\mathcal{H}_K^{\text{trees}}$ instead of $\mathcal{H}^{\text{stumps}}$, where the maximum tree depth K was an additional parameter to be optimized. Specifically, for AdaBoost we optimized over $K \in \{1, \ldots, 6\}$, for AdaBoost-L1 we optimized over those same values for K and $\beta \in \{10^{-i} : i = 3, \ldots, 7\}$, and for DeepBoost we optimized over those same values for K, β and $\lambda \in \{10^{-i} : i = 3, \ldots, 7\}$.

The last set of experiments, reported in Table 3, are identical to the experiments reported in Table 2, except we used the logistic loss $\Phi(-u) = \log_2(1 + \exp(-u))$.

We used the following parameter optimization procedure in all experiments: Each dataset was randomly partitioned into 10 folds, and each algorithm was run 10 times, with a different assignment of folds to the training set, validation set and test set for each run. Specifically, for each run $i \in \{0, \ldots, 9\}$, fold i was used for testing, fold $i + 1 \pmod{10}$ was used for validation, and the remaining folds were used for training. For each run, we selected the parameters that had the lowest error on the validation set and then measured the error of those parameters on the test set. The average error and the standard deviation of the error over all 10 runs is reported in Tables 1, 2 and 3, as is the average number of trees and the average size of the trees in the ensembles.

In all of our experiments, the number of iterations was set to 100. We also experimented with running each algorithm

Table 2. Results for boosted decision trees and the exponential loss function.

breastcancer	AdaBoost	AdaBoost-L1	DeepBoost
Error	0.0267	0.0264	**0.0243**
(std dev)	(0.00841)	(0.0098)	**(0.00797)**
Avg tree size	29.1	28.9	20.9
Avg no. of trees	67.1	51.7	55.9

ocr17	AdaBoost	AdaBoost-L1	DeepBoost
Error	0.004	0.003	**0.002**
(std dev)	(0.00316)	(0.00100)	**(0.00100)**
Avg tree size	15.0	30.4	26.0
Avg no. of trees	88.3	65.3	61.8

ionosphere	AdaBoost	AdaBoost-L1	DeepBoost
Error	0.0661	0.0657	**0.0501**
(std dev)	(0.0315)	(0.0257)	**(0.0316)**
Avg tree size	29.8	31.4	26.1
Avg no. of trees	75.0	69.4	50.0

ocr49	AdaBoost	AdaBoost-L1	DeepBoost
Error	0.0180	**0.0175**	**0.0175**
(std dev)	(0.00555)	(0.00357)	**(0.00510)**
Avg tree size	30.9	62.1	30.2
Avg no. of trees	92.4	89.0	83.0

german	AdaBoost	AdaBoost-L1	DeepBoost
Error	0.239	0.239	**0.234**
(std dev)	(0.0165)	(0.0201)	**(0.0148)**
Avg tree size	3	7	16.0
Avg no. of trees	91.3	87.5	14.1

ocr17-mnist	AdaBoost	AdaBoost-L1	DeepBoost
Error	0.00471	0.00471	**0.00409**
(std dev)	(0.0022)	(0.0021)	**(0.0021)**
Avg tree size	15	33.4	22.1
Avg no. of trees	88.7	66.8	59.2

diabetes	AdaBoost	AdaBoost-L1	DeepBoost
Error	0.249	0.240	**0.230**
(std dev)	(0.0272)	(0.0313)	**(0.0399)**
Avg tree size	3	3	5.37
Avg no. of trees	45.2	28.0	19.0

ocr49-mnist	AdaBoost	AdaBoost-L1	DeepBoost
Error	0.0198	0.0197	**0.0182**
(std dev)	(0.00500)	(0.00512)	**(0.00551)**
Avg tree size	29.9	66.3	30.1
Avg no. of trees	82.4	81.1	80.9

Table 3. Results for boosted decision trees and the logistic loss function.

breastcancer	LogReg	LogReg-L1	DeepBoost
Error	0.0351	0.0264	**0.0264**
(std dev)	(0.0101)	(0.0120)	**(0.00876)**
Avg tree size	15	59.9	14.0
Avg no. of trees	65.3	16.0	23.8

ocr17	LogReg	LogReg-L1	DeepBoost
Error	0.00300	0.00400	**0.00250**
(std dev)	(0.00100)	(0.00141)	**(0.000866)**
Avg tree size	15.0	7	22.1
Avg no. of trees	75.3	53.8	25.8

ionosphere	LogReg	LogReg-L1	DeepBoost
Error	0.074	0.060	**0.043**
(std dev)	(0.0236)	(0.0219)	**(0.0188)**
Avg tree size	7	30.0	18.4
Avg no. of trees	44.7	25.3	29.5

ocr49	LogReg	LogReg-L1	DeepBoost
Error	0.0205	0.0200	**0.0170**
(std dev)	(0.00654)	(0.00245)	**(0.00361)**
Avg tree size	31.0	31.0	63.2
Avg no. of trees	63.5	54.0	37.0

german	LogReg	LogReg-L1	DeepBoost
Error	0.233	0.232	**0.225**
(std dev)	(0.0114)	(0.0123)	**(0.0103)**
Avg tree size	7	7	14.4
Avg no. of trees	72.8	66.8	67.8

ocr17-mnist	LogReg	LogReg-L1	DeepBoost
Error	0.00422	0.00417	**0.00399**
(std dev)	(0.00191)	(0.00188)	**(0.00211)**
Avg tree size	15	15	25.9
Avg no. of trees	71.4	55.6	27.6

diabetes	LogReg	LogReg-L1	DeepBoost
Error	0.250	**0.246**	**0.246**
(std dev)	(0.0374)	**(0.0356)**	**(0.0356)**
Avg tree size	3	3	3
Avg no. of trees	46.0	45.5	45.5

ocr49-mnist	LogReg	LogReg-L1	DeepBoost
Error	0.0211	**0.0201**	**0.0201**
(std dev)	(0.00412)	**(0.00433)**	**(0.00411)**
Avg tree size	28.7	33.5	72.8
Avg no. of trees	79.3	61.7	41.9

for up to 1,000 iterations, but observed that the test errors did not change significantly, and more importantly the ordering of the algorithms by their test errors was unchanged from 100 iterations to 1,000 iterations.

Observe that with the exponential loss, DeepBoost has a smaller test error than AdaBoost and AdaBoost-L1 on every dataset and for every set of base hypotheses, except for the `ocr49-mnist` dataset with decision trees where its performance matches that of AdaBoost-L1. Similarly, with the logistic loss, DeepBoost performs always at least as well as LogReg or LogReg-L1. For the small-sized UCI datasets it is difficult to obtain statistically significant results, but, for the larger `ocrXX-mnist` datasets, our results with DeepBoost are statistically significantly better at the 2% level using one-sided paired t-tests in all three sets of experiments (three tables), except for `ocr49-mnist` in Table 3, where this holds only for the comparison with LogReg.

This across-the-board improvement is the result of DeepBoost's complexity-conscious ability to dynamically tune the sizes of the decision trees selected in each boosting round, trading off between training error and hypothesis class complexity. The selected tree sizes should depend on properties of the training set, and this is borne out by our experiments: For some datasets, such as `breastcancer`, DeepBoost selects trees that are smaller on average than the trees selected by AdaBoost-L1 or LogReg-L1, while, for other datasets, such as `german`, the average tree size is larger. Note that AdaBoost and AdaBoost-L1 produce ensembles of trees that have a constant depth since neither algorithm penalizes tree size except for imposing a maximum tree depth K, while for DeepBoost the trees in one ensemble typically vary in size. Figure 3 plots the distri-

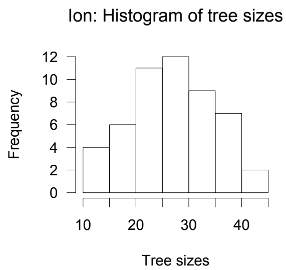


Figure 3. Distribution of tree sizes when DeepBoost is run on the `ionosphere` dataset.

bution of tree sizes for one run of DeepBoost. It should be noted that the columns for AdaBoost in Table 1 simply list the number of stumps to be the same as the number of boosting rounds; a careful examination of the ensembles for 100 rounds of boosting typically reveals a 5% duplication of stumps in the ensembles.

Theorem 1 is a margin-based generalization guarantee, and is also the basis for the derivation of DeepBoost, so we should expect DeepBoost to induce large margins on the training set. Figure 4 shows the margin distributions for AdaBoost, AdaBoost-L1 and DeepBoost on the same subset of the `ionosphere` dataset.

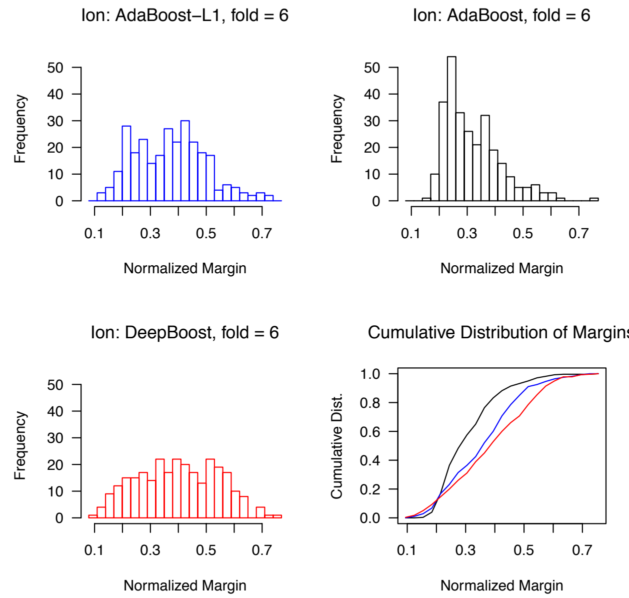


Figure 4. Distribution of normalized margins for AdaBoost (upper right), AdaBoost-L1 (upper left) and DeepBoost (lower left) on the same subset of `ionosphere`. The cumulative margin distributions (lower right) illustrate that DeepBoost (red) induces larger margins on the training set than either AdaBoost (black) or AdaBoost-L1 (blue).

5. Conclusion

We presented a theoretical analysis of learning with a base hypothesis set composed of increasingly complex subfamilies, including very deep or complex ones, and derived an algorithm, DeepBoost, which is precisely based on those guarantees. We also reported the results of experiments with this algorithm and compared its performance with that of AdaBoost and additive Logistic Regression, and their L_1-norm regularized counterparts in several tasks. We have derived similar theoretical guarantees in the multi-class setting and used them to derive a family of new multi-class deep boosting algorithms that we will present and discuss elsewhere. Our theoretical analysis and algorithmic design could also be extended to ranking and to a broad class of loss functions. This should also lead to the generalization of several existing algorithms and their use with a richer hypothesis set structured as a union of families with different Rademacher complexity. In particular, the broad family of maximum entropy models and conditional maximum entropy models and their many variants, which includes the already discussed logistic regression, could all be extended in a similar way. The resulting *DeepMaxent* models (or their conditional versions) may admit an alternative theoretical justification that we will discuss elsewhere. Our algorithm can also be extended by considering non-differentiable convex surrogate losses such as the hinge loss. When used with kernel base classifiers, this leads to an algorithm we have named *DeepSVM*. The theory we developed could perhaps be further generalized to encompass the analysis of other learning techniques such as multi-layer neural networks.

Our analysis and algorithm also shed some new light on some remaining questions left about the theory underlying AdaBoost. The primary theoretical justification for AdaBoost is a margin guarantee (Schapire et al., 1997; Koltchinskii & Panchenko, 2002). However, AdaBoost does not precisely maximize the minimum margin, while other algorithms such as arc-gv (Breiman, 1996) that are designed to do so tend not to outperform AdaBoost (Reyzin & Schapire, 2006). Two main reasons are suspected for this observation: (1) in order to achieve a better margin, algorithms such as arc-gv may tend to select deeper decision trees or in general more complex hypotheses, which may then affect their generalization; (2) while those algorithms achieve a better margin, they do not achieve a better margin distribution. Our theory may help better understand and evaluate the effect of factor (1) since our learning bounds explicitly depend on the mixture weights and the contribution of each hypothesis set H_k to the definition of the ensemble function. However, our guarantees also suggest a better algorithm, DeepBoost.

Acknowledgments

We thank Vitaly Kuznetsov for his comments on an earlier draft of this paper. The work of M. Mohri was partly funded by the NSF award IIS-1117591.

References

Bartlett, Peter L. and Mendelson, Shahar. Rademacher and Gaussian complexities: Risk bounds and structural results. *JMLR*, 3, 2002.

Bauer, Eric and Kohavi, Ron. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36(1-2):105–139, 1999.

Breiman, Leo. Bagging predictors. *Machine Learning*, 24 (2):123–140, 1996.

Breiman, Leo. Prediction games and arcing algorithms. *Neural Computation*, 11(7):1493–1517, 1999.

Caruana, Rich, Niculescu-Mizil, Alexandru, Crew, Geoff, and Ksikes, Alex. Ensemble selection from libraries of models. In *ICML*, 2004.

Dietterich, Thomas G. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, 2000.

Duchi, John C. and Singer, Yoram. Boosting with structural sparsity. In *ICML*, pp. 38, 2009.

Freund, Yoav and Schapire, Robert E. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer System Sciences*, 55(1): 119–139, 1997.

Freund, Yoav, Mansour, Yishay, and Schapire, Robert E. Generalization bounds for averaged classifiers. *The Annals of Statistics*, 32:1698–1722, 2004.

Friedman, Jerome, Hastie, Trevor, and Tibshirani, Robert. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28:2000, 1998.

Grove, Adam J and Schuurmans, Dale. Boosting in the limit: Maximizing the margin of learned ensembles. In *AAAI/IAAI*, pp. 692–699, 1998.

Kivinen, Jyrki and Warmuth, Manfred K. Boosting as entropy projection. In *COLT*, pp. 134–144, 1999.

Koltchinskii, Vladmir and Panchenko, Dmitry. Empirical margin distributions and bounding the generalization error of combined classifiers. *Annals of Statistics*, 30, 2002.

MacKay, David J. C. *Bayesian methods for adaptive models*. PhD thesis, California Institute of Technology, 1991.

Mansour, Yishay. Pessimistic decision tree pruning based on tree size. In *Proceedings of ICML*, pp. 195–201, 1997.

Mohri, Mehryar, Rostamizadeh, Afshin, and Talwalkar, Ameet. *Foundations of Machine Learning*. The MIT Press, 2012.

Quinlan, J. Ross. Bagging, boosting, and C4.5. In *AAAI/IAAI, Vol. 1*, pp. 725–730, 1996.

Rätsch, Gunnar and Warmuth, Manfred K. Maximizing the margin with boosting. In *COLT*, pp. 334–350, 2002.

Rätsch, Gunnar and Warmuth, Manfred K. Efficient margin maximizing with boosting. *Journal of Machine Learning Research*, 6:2131–2152, 2005.

Rätsch, Gunnar, Mika, Sebastian, and Warmuth, Manfred K. On the convergence of leveraging. In *NIPS*, pp. 487–494, 2001a.

Rätsch, Gunnar, Onoda, Takashi, and Müller, Klaus-Robert. Soft margins for AdaBoost. *Machine Learning*, 42(3):287–320, 2001b.

Reyzin, Lev and Schapire, Robert E. How boosting the margin can also boost classifier complexity. In *ICML*, pp. 753–760, 2006.

Schapire, Robert E. Theoretical views of boosting and applications. In *Proceedings of ALT 1999*, volume 1720 of *Lecture Notes in Computer Science*, pp. 13–25. Springer, 1999.

Schapire, Robert E. The boosting approach to machine learning: An overview. In *Nonlinear Estimation and Classification*, pp. 149–172. Springer, 2003.

Schapire, Robert E., Freund, Yoav, Bartlett, Peter, and Lee, Wee Sun. Boosting the margin: A new explanation for the effectiveness of voting methods. In *ICML*, pp. 322–330, 1997.

Smyth, Padhraic and Wolpert, David. Linearly combining density estimators via stacking. *Machine Learning*, 36: 59–83, July 1999.

Vapnik, Vladimir N. *Statistical Learning Theory*. Wiley-Interscience, 1998.

Warmuth, Manfred K., Liao, Jun, and Rätsch, Gunnar. Totally corrective boosting algorithms that maximize the margin. In *ICML*, pp. 1001–1008, 2006.

A. Proof of Theorem 1

Theorem 1. *Assume $p > 1$. Fix $\rho > 0$. Then, for any $\delta > 0$, with probability at least $1 - \delta$ over the choice of a sample S of size m drawn i.i.d. according to $\mathcal{D}$, the following inequality holds for all $f = \sum_{t=1}^{T} \alpha_t h_t$:*

$$R(f) \leq \widehat{R}_{S,\rho}(f) + \frac{4}{\rho} \sum_{t=1}^{T} \alpha_t \mathfrak{R}_m(H_{k_t}) + \frac{2}{\rho}\sqrt{\frac{\log p}{m}} + \sqrt{\left\lceil \frac{4}{\rho^2} \log\left[\frac{\rho^2 m}{\log p}\right]\right\rceil \frac{\log p}{m} + \frac{\log \frac{2}{\delta}}{2m}}.$$

Thus, $R(f) \leq \widehat{R}_{S,\rho}(f) + \frac{4}{\rho}\sum_{t=1}^{T} \alpha_t \mathfrak{R}_m(H_{k_t}) + C(m,p)$ with $C(m,p) = O\left(\sqrt{\frac{\log p}{\rho^2 m} \log\left[\frac{\rho^2 m}{\log p}\right]}\right)$.

Proof. For a fixed $\mathbf{h} = (h_1, \ldots, h_T)$, any $\boldsymbol{\alpha} \in \Delta$ defines a distribution over $\{h_1, \ldots, h_T\}$. Sampling from $\{h_1, \ldots, h_T\}$ according to $\boldsymbol{\alpha}$ and averaging leads to functions g of the form $g = \frac{1}{n}\sum_{i=1}^{T} n_t h_t$ for some $\mathbf{n} = (n_1, \ldots, n_T)$, with $\sum_{t=1}^{T} n_t = n$, and $h_t \in H_{k_t}$.

For any $\mathbf{N} = (N_1, \ldots, N_p)$ with $|\mathbf{N}| = n$, we consider the family of functions

$$G_{\mathcal{F},\mathbf{N}} = \left\{ \frac{1}{n} \sum_{k=1}^{p} \sum_{j=1}^{N_k} h_{k,j} \,|\, \forall (k,j) \in [p] \times [N_k], h_{k,j} \in H_k \right\},$$

and the union of all such families $G_{\mathcal{F},n} = \bigcup_{|\mathbf{N}|=n} G_{\mathcal{F},\mathbf{N}}$. Fix $\rho > 0$. For a fixed $\mathbf{N}$, the Rademacher complexity of $G_{\mathcal{F},\mathbf{N}}$ can be bounded as follows for any $m \geq 1$: $\mathfrak{R}_m(G_{\mathcal{F},\mathbf{N}}) \leq \frac{1}{n}\sum_{k=1}^{p} N_k \,\mathfrak{R}_m(H_k)$. Thus, the following standard margin-based Rademacher complexity bound holds (Koltchinskii & Panchenko, 2002). For any $\delta > 0$, with probability at least $1 - \delta$, for all $g \in G_{\mathcal{F},\mathbf{N}}$,

$$R_\rho(g) - \widehat{R}_{S,\rho}(g) \leq \frac{2}{\rho}\frac{1}{n}\sum_{k=1}^{p} N_k \,\mathfrak{R}_m(H_k) + \sqrt{\frac{\log \frac{1}{\delta}}{2m}}.$$

Since there are at most p^n possible p-tuples $\mathbf{N}$ with $|\mathbf{N}| = n$, by the union bound, for any $\delta > 0$, with probability at least $1 - \delta$, for all $g \in G_{\mathcal{F},n}$, we can write

$$R_\rho(g) - \widehat{R}_{S,\rho}(g) \leq \frac{2}{\rho}\frac{1}{n}\sum_{k=1}^{p} N_k \,\mathfrak{R}_m(H_k) + \sqrt{\frac{\log \frac{p^n}{\delta}}{2m}}.$$

Thus, with probability at least $1 - \delta$, for all functions $g = \frac{1}{n}\sum_{i=1}^{T} n_t h_t$ with $h_t \in H_{k_t}$, the following inequality holds

$$R_\rho(g) - \widehat{R}_{S,\rho}(g) \leq \frac{2}{\rho}\frac{1}{n}\sum_{k=1}^{p} \sum_{t: k_t = k} n_t \,\mathfrak{R}_m(H_{k_t}) + \sqrt{\frac{\log \frac{p^n}{\delta}}{2m}}.$$

Taking the expectation with respect to $\boldsymbol{\alpha}$ and using $\mathrm{E}_{\boldsymbol{\alpha}}[n_t/n] = \alpha_t$, we obtain that for any $\delta > 0$, with probability at least $1 - \delta$, for all $\mathbf{h}$, we can write

$$\mathop{\mathrm{E}}_{\boldsymbol{\alpha}}[R_\rho(g) - \widehat{R}_{S,\rho}(g)] \leq \frac{2}{\rho}\sum_{t=1}^{T} \alpha_t \mathfrak{R}_m(H_{k_t}) + \sqrt{\frac{\log \frac{p^n}{\delta}}{2m}}.$$

Fix $n \geq 1$. Then, for any $\delta_n > 0$, with probability at least $1 - \delta_n$,

$$\mathop{\mathrm{E}}_{\boldsymbol{\alpha}}[R_{\rho/2}(g) - \widehat{R}_{S,\rho/2}(g)] \leq \frac{4}{\rho}\sum_{t=1}^{T} \alpha_t \mathfrak{R}_m(H_{k_t}) + \sqrt{\frac{\log \frac{p^n}{\delta_n}}{2m}}.$$

Choose $\delta_n = \frac{\delta}{2p^{n-1}}$ for some $\delta > 0$, then for $p \geq 2$, $\sum_{n \geq 1} \delta_n = \frac{\delta}{2(1-1/p)} \leq \delta$. Thus, for any $\delta > 0$ and any $n \geq 1$, with probability at least $1 - \delta$, the following holds for all $\mathbf{h}$:

$$\mathop{\mathrm{E}}_{\boldsymbol{\alpha}}[R_{\rho/2}(g) - \widehat{R}_{S,\rho/2}(g)] \leq \frac{4}{\rho}\sum_{t=1}^{T} \alpha_t \mathfrak{R}_m(H_{k_t}) + \sqrt{\frac{\log \frac{2p^{2n-1}}{\delta}}{2m}}. \quad (10)$$

Now, for any $f = \sum_{t=1}^{T} \alpha_t h_t \in \mathcal{F}$ and any $g = \frac{1}{n}\sum_{i=1}^{T} n_t h_t$, we can upper bound $R(f) = \Pr_{(x,y)\sim\mathcal{D}}[yf(x) \leq 0]$, the generalization error of f, as follows:

$$\begin{aligned} R(f) &= \Pr_{(x,y)\sim\mathcal{D}}[yf(x) - yg(x) + yg(x) \leq 0] \\ &\leq \Pr[yf(x) - yg(x) < -\rho/2] + \Pr[yg(x) \leq \rho/2] \\ &= \Pr[yf(x) - yg(x) < -\rho/2] + R_{\rho/2}(g). \end{aligned}$$

We can also write

$$\begin{aligned} \widehat{R}_{\rho/2}(g) &= \widehat{R}_{S,\rho/2}(g - f + f) \\ &\leq \widehat{\Pr}[yg(x) - yf(x) < -\rho/2] + \widehat{R}_{S,\rho}(f). \end{aligned}$$

Combining these inequalities yields

$$\begin{aligned} &\Pr_{(x,y)\sim\mathcal{D}}[yf(x) \leq 0] - \widehat{R}_{S,\rho}(f) \\ &\leq \Pr[yf(x) - yg(x) < -\rho/2] \\ &+ \widehat{\Pr}[yg(x) - yf(x) < -\rho/2] + R_{\rho/2}(g) - \widehat{R}_{S,\rho/2}(g). \end{aligned}$$

Taking the expectation with respect to $\boldsymbol{\alpha}$ yields

$$\begin{aligned} R(f) - \widehat{R}_{S,\rho}(f) \leq{}& \mathop{\mathrm{E}}_{x\sim\mathcal{D},\boldsymbol{\alpha}}[1_{yf(x)-yg(x)<-\rho/2}] + \\ &\mathop{\mathrm{E}}_{x\sim\mathcal{D},\boldsymbol{\alpha}}[1_{yg(x)-yf(x)<-\rho/2}] + \mathop{\mathrm{E}}_{\boldsymbol{\alpha}}[R_{\rho/2}(g) - \widehat{R}_{S,\rho/2}(g)]. \end{aligned}$$

Since $f = \mathrm{E}_{\boldsymbol{\alpha}}[g]$, by Hoeffding's inequality, for any x,

$$\begin{aligned} \mathop{\mathrm{E}}_{\boldsymbol{\alpha}}[1_{yf(x)-yg(x)<-\rho/2}] &= \Pr_{\boldsymbol{\alpha}}[yf(x) - yg(x) < -\rho/2] \leq e^{-\frac{n\rho^2}{8}} \\ \mathop{\mathrm{E}}_{\boldsymbol{\alpha}}[1_{yg(x)-yf(x)<-\rho/2}] &= \Pr_{\boldsymbol{\alpha}}[yg(x) - yf(x) < -\rho/2] \leq e^{-\frac{n\rho^2}{8}}. \end{aligned}$$

Thus, for any fixed $f \in \mathcal{F}$, we can write

$$R(f) - \widehat{R}_{S,\rho}(f) \leq 2e^{-n\rho^2/8} + \mathop{\mathrm{E}}_{\boldsymbol{\alpha}}[R_{\rho/2}(g) - \widehat{R}_{S,\rho/2}(g)].$$

Thus, the following inequality holds:

$$\sup_{f\in\mathcal{F}} R(f) - \widehat{R}_{S,\rho}(f) \leq 2e^{-n\rho^2/8} + \sup_{\mathbf{h}} \mathop{\mathrm{E}}_{\boldsymbol{\alpha}}[R_{\rho/2}(g) - \widehat{R}_{S,\rho/2}(g)].$$

Therefore, in view of (10), for any $\delta > 0$ and any $n \geq 1$, with probability at least $1-\delta$, the following holds for all $f \in \mathcal{F}$:

$$\begin{aligned} &R(f) - \widehat{R}_{S,\rho}(f) \\ &\leq \frac{4}{\rho}\sum_{t=1}^{T}\alpha_t \mathfrak{R}_m(H_{k_t}) + 2e^{-n\rho^2/8} + \sqrt{\frac{\log\frac{2p^{2n-1}}{\delta}}{2m}} \\ &= \frac{4}{\rho}\sum_{t=1}^{T}\alpha_t \mathfrak{R}_m(H_{k_t}) + 2e^{-n\rho^2/8} + \sqrt{\frac{(2n-1)\log p + \log\frac{2}{\delta}}{2m}}. \end{aligned}$$

To select n, we seek to minimize

$$f\colon n \mapsto 2e^{-n\rho^2/8} + \sqrt{\frac{n\log p}{m}} = 2e^{-nu} + \sqrt{nv},$$

with $u = \rho^2/8$ and $v = (\log p)/m$. f is differentiable and for all n, $f'(n) = -2ue^{-nu} + \frac{\sqrt{v}}{2\sqrt{n}}$. The minimum of f is thus for n such that

$$\begin{aligned} f'(n) = 0 \Leftrightarrow 2ue^{-nu} = \frac{\sqrt{v}}{2\sqrt{n}} &\Leftrightarrow -2une^{-2un} = -\frac{v}{8u} \\ &\Leftrightarrow n = \frac{-1}{2u}W_{-1}\Big(\frac{-v}{8u}\Big), \end{aligned}$$

where W_{-1} is the second branch of the Lambert function (inverse of $x \mapsto xe^x$. It is not hard to verify that the following inequalities hold for all $x \in (0, 1/e]$:

$$-\log(x) \leq -W_{-1}(-x) \leq 2\log(x).$$

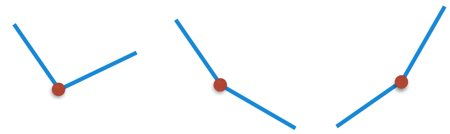
Bounding $-W_{-1}$ using the lower bound leads to the following choice for n:

$$n = \Big\lceil \frac{-1}{2u}\log\Big(\frac{v}{8u}\Big)\Big\rceil = \Big\lceil \frac{4}{\rho^2}\log\Big(\frac{\rho^2 m}{\log p}\Big)\Big\rceil.$$

Plugging in this value of n yields the following bound:

$$\begin{aligned} R(f) - \widehat{R}_{S,\rho}(f) \leq \frac{4}{\rho}\sum_{t=1}^{T}\alpha_t\mathfrak{R}_m(H_{k_t}) + \frac{2}{\rho}\sqrt{\frac{\log p}{m}} \\ + \sqrt{\Big\lceil \frac{4}{\rho^2}\log\Big[\frac{\rho^2 m}{\log p}\Big]\Big\rceil \frac{\log p}{m} + \frac{\log\frac{2}{\delta}}{2m}}, \end{aligned}$$

which concludes the proof. $\square$



(a) (b) (c)

Figure 5. Illustration of the directional derivatives in the three cases of definition (11).

B. Coordinate descent

B.1. Maximum descent coordinate

For a differentiable convex function, the definition of coordinate descent along the direction with maximal descent is standard: the direction selected is the one maximizing the absolute value of the directional derivative. Here, we clarify the definition of the maximal descent strategy for a non-differentiable convex function.

For any function $Q\colon \mathbb{R}^N \to \mathbb{R}$, we denote by $Q'_+(\boldsymbol{\alpha}, \mathbf{e})$ the right directional derivative of Q at $\boldsymbol{\alpha} \in \mathbb{R}^N$ and by $Q'_-(\boldsymbol{\alpha}, \mathbf{e})$ its left directional derivative at $\boldsymbol{\alpha} \in \mathbb{R}^N$ along the direction $\mathbf{e} \in \mathbb{R}^N$, $\|\mathbf{e}\| = 1$, when they exist:

$$\begin{aligned} Q'_+(\boldsymbol{\alpha}, \mathbf{e}) &= \lim_{\eta\to 0^+} \frac{Q(\boldsymbol{\alpha} + \eta\mathbf{e}) - Q(\boldsymbol{\alpha})}{\eta} \\ Q'_-(\boldsymbol{\alpha}, \mathbf{e}) &= \lim_{\eta\to 0^-} \frac{Q(\boldsymbol{\alpha} + \eta\mathbf{e}) - Q(\boldsymbol{\alpha})}{\eta}. \end{aligned}$$

For the remaining of this section, we will assume that Q is a convex function. It is known that in that case these quantities always exist and that $Q'_-(\boldsymbol{\alpha}, \mathbf{e}) \leq Q'_+(\boldsymbol{\alpha}, \mathbf{e})$ for all $\boldsymbol{\alpha}$ and $\mathbf{e}$. The left and right directional derivatives coincide with the directional derivative $Q'(\boldsymbol{\alpha}, \mathbf{e})$ of Q along the direction $\mathbf{e}$ when Q is differentiable at $\boldsymbol{\alpha}$ along the direction $\mathbf{e}$: $Q'(\boldsymbol{\alpha}, \mathbf{e}) = Q'_+(\boldsymbol{\alpha}, \mathbf{e}) = Q'_-(\boldsymbol{\alpha}, \mathbf{e})$.

For any $j \in [1, N]$, let $\mathbf{e}_j$ denote the jth unit vector in $\mathbb{R}^N$. For any $\boldsymbol{\alpha} \in \mathbb{R}^N$ and $j \in [1, N]$, we define the *descent gradient* $\delta Q(\boldsymbol{\alpha}, \mathbf{e}_j)$ *of* Q *along the direction* $\mathbf{e}_j$ as follows:

$$\delta Q(\boldsymbol{\alpha}, \mathbf{e}_j) = \begin{cases} 0 & \text{if } Q'_-(\boldsymbol{\alpha}, \mathbf{e}_j) \leq 0 \leq Q'_+(\boldsymbol{\alpha}, \mathbf{e}_j) \\ Q'_+(\boldsymbol{\alpha}, \mathbf{e}_j) & \text{if } Q'_-(\boldsymbol{\alpha}, \mathbf{e}_j) \leq Q'_+(\boldsymbol{\alpha}, \mathbf{e}_j) \leq 0 \\ Q'_-(\boldsymbol{\alpha}, \mathbf{e}_j) & \text{if } 0 \leq Q'_-(\boldsymbol{\alpha}, \mathbf{e}_j) \leq Q'_+(\boldsymbol{\alpha}, \mathbf{e}_j). \end{cases} \tag{11}$$

$\delta Q(\boldsymbol{\alpha}, \mathbf{e}_j)$ is the element of the subgradient along $\mathbf{e}_j$ that is the closest to 0. Figure 5 illustrates the three cases in that definition. Note that when Q is differentiable along $\mathbf{e}_j$, then $Q'_+(\boldsymbol{\alpha}, \mathbf{e}_j) = Q'_-(\boldsymbol{\alpha}, \mathbf{e}_j)$ and $\delta Q(\boldsymbol{\alpha}, \mathbf{e}_j) = Q'(\boldsymbol{\alpha}, \mathbf{e}_j)$. The maximum descent coordinate can then be defined by

$$k = \operatorname*{argmax}_{j\in[1,N]} |\delta Q(\boldsymbol{\alpha}, \mathbf{e}_j)| \tag{12}$$

This coincides with the standard definition when Q is convex and differentiable.

B.2. Direction

In view of (12), at each iteration $t \geq 1$, the direction $\mathbf{e}_k$ selected by coordinate descent with maximum descent is $k = \operatorname{argmax}_{j\in[1,N]} |\delta Q(\boldsymbol{\alpha}_{t-1}, \mathbf{e}_j)|$. To determine k, we compute $\delta Q(\boldsymbol{\alpha}_{t-1}, \mathbf{e}_j)$ for all $j \in [1, N]$ by distinguishing two cases: $\alpha_{t-1,j} \neq 0$ and $\alpha_{t-1,j} = 0$.

Assume first that $\alpha_{t-1,j} \neq 0$ and let s denote the sign of $\alpha_{t-1,j}$. For η sufficiently small, $\alpha_{t-1,j} + \eta$ has the sign of $\alpha_{t-1,j}$, that is s and

$$F(\boldsymbol{\alpha}_{t-1} + \eta \mathbf{e}_j) = \frac{1}{m}\sum_{i=1}^{m} \Phi\Big(1 - y_i f_{t-1}(x_i) - \eta y_i h_j(x_i)\Big) + \sum_{p\neq j} \Lambda_j |\alpha_{t-1,p}| + s\Lambda_j(\alpha_{t-1,j} + \eta).$$

Thus, when $\alpha_{t-1,j} \neq 0$, F admits a directional derivative along $\mathbf{e}_j$ given by

$$\begin{aligned} F'(\boldsymbol{\alpha}_{t-1}, \mathbf{e}_j) &= -\frac{1}{m}\sum_{i=1}^{m} y_i h_j(x_i)\Phi'\big(1 - y_i f_{t-1}(x_i)\big) + s\Lambda_j \\ &= -\frac{1}{m}\sum_{i=1}^{m} y_i h_j(x_i)\mathcal{D}_t(i) S_t + s\Lambda_j \\ &= (2\epsilon_{t,j} - 1)\frac{S_t}{m} + s\Lambda_j, \end{aligned}$$

and $\delta F(\boldsymbol{\alpha}_{t-1}, \mathbf{e}_j) = (2\epsilon_{t,j} - 1)\frac{S_t}{m} + \operatorname{sgn}(\alpha_{t-1,j})\Lambda_j$. When $\alpha_{t-1,j} = 0$, we find similarly that

$$\begin{aligned} F'_+(\boldsymbol{\alpha}_{t-1}, \mathbf{e}_j) &= (2\epsilon_{t,j} - 1)\frac{S_t}{m} + \Lambda_j \\ F'_-(\boldsymbol{\alpha}_{t-1}, \mathbf{e}_j) &= (2\epsilon_{t,j} - 1)\frac{S_t}{m} - \Lambda_j. \end{aligned}$$

The condition $(F'_-(\boldsymbol{\alpha}, \mathbf{e}_j) \leq 0 \leq F'_+(\boldsymbol{\alpha}, \mathbf{e}_j))$ is equivalent to

$$\Big(-\Lambda_j \leq (2\epsilon_{t,j} - 1)\frac{S_t}{m} \leq \Lambda_j\Big) \Leftrightarrow \left|\epsilon_{t,j} - \frac{1}{2}\right| \leq \frac{\Lambda_j m}{2S_t}.$$

Thus, in summary, we can write, for all $j \in [1, N]$,

$$\delta F(\boldsymbol{\alpha}_{t-1}, \mathbf{e}_j) = \begin{cases} (2\epsilon_{t,j} - 1)\frac{S_t}{m} + \operatorname{sgn}(\alpha_{t-1,j})\Lambda_j & \text{if } (\alpha_{t-1,j} \neq 0) \\ 0 & \text{else if } \left|\epsilon_{t,j} - \frac{1}{2}\right| \leq \frac{\Lambda_j m}{2S_t} \\ (2\epsilon_{t,j} - 1)\frac{S_t}{m} + \Lambda_j & \text{else if } \epsilon_{t,j} - \frac{1}{2} \leq -\frac{\Lambda_j m}{2S_t} \\ (2\epsilon_{t,j} - 1)\frac{S_t}{m} - \Lambda_j & \text{otherwise.} \end{cases}$$

This can be simplified by unifying the last two cases and observing that the sign of $(\epsilon_{t,j} - \frac{1}{2})$ suffices to distinguish between the last two cases:

$$\delta F(\boldsymbol{\alpha}_{t-1}, \mathbf{e}_j) = \begin{cases} (2\epsilon_{t,j} - 1)\frac{S_t}{m} + \operatorname{sgn}(\alpha_{t-1,j})\Lambda_j & \text{if } (\alpha_{t-1,j} \neq 0) \\ 0 & \text{else if } \left|\epsilon_{t,j} - \frac{1}{2}\right| \leq \frac{\Lambda_j m}{2S_t} \\ (2\epsilon_{t,j} - 1)\frac{S_t}{m} - \operatorname{sgn}(\epsilon_{t,j} - \frac{1}{2})\Lambda_j & \text{otherwise.} \end{cases}$$

B.3. Step

Given the direction $\mathbf{e}_k$, the optimal step value η is given by $\operatorname{argmin}_\eta F(\boldsymbol{\alpha}_{t-1} + \eta\, \mathbf{e}_k)$. In the most general case, η can be found via a line search or other numerical methods. In some special cases, we can derive a closed-form solution for the step by minimizing an upper bound on $F(\boldsymbol{\alpha}_{t-1} + \eta\, \mathbf{e}_k)$. For convenience, in what follows, we use the shorthand ϵ_t for $\epsilon_{t,k}$.

Since $y_i h_k(x_i) = \frac{1 + y_i h_k(x_i)}{2}\cdot(1) + \frac{1 - y_i h_k(x_i)}{2}\cdot(-1)$, by the convexity of $u \mapsto \Phi(1 - \eta u)$, the following holds for all $\eta \in \mathbb{R}$:

$$\begin{aligned} \Phi\Big(1 - y_i f_{t-1}(x_i) - \eta y_i h_k(x_i)\Big) &\leq \frac{1 + y_i h_k(x_i)}{2}\Phi\Big(1 - y_i f_{t-1}(x_i)) - \eta\Big) \\ &\quad + \frac{1 - y_i h_k(x_i)}{2}\Phi\Big(1 - y_i f_{t-1}(x_i)) + \eta\Big). \end{aligned} \tag{13}$$

Thus, we can write

$$\begin{aligned} &F(\boldsymbol{\alpha}_{t-1} + \eta \mathbf{e}_k) - \sum_{j\neq k} \Lambda_j |\alpha_{t-1,j}| \\ &\leq \frac{1}{m}\sum_{i=1}^{m} \frac{1 + y_i h_k(x_i)}{2}\Phi\Big(1 - y_i f_{t-1}(x_i)) - \eta\Big) \\ &\quad + \frac{1}{m}\sum_{i=1}^{m} \frac{1 - y_i h_k(x_i)}{2}\Phi\Big(1 - y_i f_{t-1}(x_i)) + \eta\Big) \\ &\quad + \Lambda_k |\alpha_{t-1,k} + \eta|. \end{aligned}$$

Let $J(\eta)$ denote that upper bound. We can select η to minimize $J(\eta)$. J is convex and admits a subdifferential at all points. Thus, η^* is a minimizer of $J(\eta)$ iff $0 \in \partial J(\eta^*)$, where $\partial J(\eta^*)$ denotes the subdifferential of J at η^*.

B.4. Exponential loss

In the case $\Phi = \exp$, $J(\eta)$ can be expressed as follows

$$\begin{aligned} J(\eta) &= \frac{1}{m}\sum_{i=1}^{m} \frac{1 + y_i h_k(x_i)}{2} e^{1 - y_i f_{t-1}(x_i)} e^{-\eta} \\ &\quad + \frac{1}{m}\sum_{i=1}^{m} \frac{1 - y_i h_k(x_i)}{2} e^{1 - y_i f_{t-1}(x_i)} e^{\eta} \\ &\quad + \Lambda_k |\alpha_{t-1,k} + \eta|, \end{aligned}$$

and $e^{1 - y_i f_{t-1}(x_i)} = \Phi'(1 - y_i f_{t-1}(x_i)) = S_t \mathcal{D}_t(i)$. Thus, J can be rewritten as follows:[2]

$$J(\eta) = (1 - \epsilon_t)\frac{S_t}{m} e^{-\eta} + \epsilon_t \frac{S_t}{m} e^{\eta} + \Lambda_k |\alpha_{t-1,k} + \eta|,$$

[2] Note that when the functions in H take values in $\{-1, +1\}$, (13) is in fact an equality and $J(\eta)$ coincides with $F(\boldsymbol{\alpha}_{t-1} + \eta \mathbf{e}_t) - \sum_{j\neq k} \Lambda_j |\alpha_{t-1,j}|$.

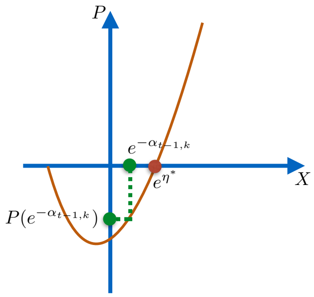


Figure 6. Plot of the polynomial function P.

where we used the shorthand $\epsilon_t = \epsilon_{t,k}$ where k is the index of the direction $\mathbf{e}_k$ selected. If $\alpha_{t-1,k} + \eta^* = 0$, then the subdifferential of $|\alpha_{t-1,k} + \eta|$ at η^* is the set $\{\nu \colon \nu \in [-1,+1]\}$. Thus, $\partial J(\eta^*)$ contains 0 iff there exists $\nu \in [-1,+1]$ such that

$$\begin{aligned}
&-(1-\epsilon_t)\frac{S_t}{m}e^{-\eta^*} + \epsilon_t\frac{S_t}{m}e^{\eta^*} + \Lambda_k\nu = 0\\
\Leftrightarrow &-(1-\epsilon_t)e^{\alpha_{t-1,k}} + \epsilon_t e^{-\alpha_{t-1,k}} + \frac{\Lambda_k m}{S_t}\nu = 0.
\end{aligned}$$

This is equivalent to the condition

$$\left|(1-\epsilon_t)e^{\alpha_{t-1,k}} - \epsilon_t e^{-\alpha_{t-1,k}}\right| \leq \frac{\Lambda_k m}{S_t}. \tag{14}$$

If $\alpha_{t-1,k} + \eta^* > 0$, then the subdifferential of $|\alpha_{t-1,k} + \eta|$ at η^* is reduced to $\{1\}$ and $\partial J(\eta^*)$ contains 0 iff

$$\begin{aligned}
&-(1-\epsilon_t)e^{-\eta^*} + \epsilon_t e^{\eta^*} + \frac{\Lambda_k m}{S_t} = 0\\
\Leftrightarrow\ &\epsilon_t e^{2\eta^*} + \frac{\Lambda_k m}{S_t}e^{\eta^*} - (1-\epsilon_t) = 0. \qquad (15)
\end{aligned}$$

Solving the resulting second-degree equation in e^{η^*} gives

$$e^{\eta^*} = -\frac{\Lambda_k m}{2\epsilon_t S_t} + \sqrt{\left(\frac{\Lambda_k m}{2\epsilon_t S_t}\right)^2 + \frac{1-\epsilon_t}{\epsilon_t}},$$

that is

$$\eta^* = \log\left[-\frac{\Lambda_k m}{2\epsilon_t S_t} + \sqrt{\left(\frac{\Lambda_k m}{2\epsilon_t S_t}\right)^2 + \frac{1-\epsilon_t}{\epsilon_t}}\right].$$

Let P be the second-degree polynomial of (15) whose solution is e^{η^*}. P is convex, has one negative solution, one positive solution, and the positive solution is e^{η^*}. Since $e^{-\alpha_{t-1,k}}$ is positive, the condition $\alpha_{t-1,k} + \eta^* > 0$ or $-\alpha_{t-1,k} < \eta^*$ is then equivalent to $P(e^{-\alpha_{t-1,k}}) < 0$ (see Figure 6), that is

$$\begin{aligned}
&\epsilon_t e^{-2\alpha_{t-1,k}} + \frac{\Lambda_k m}{S_t}e^{-\alpha_{t-1,k}} - (1-\epsilon_t) < 0\\
\Leftrightarrow &(1-\epsilon_t)e^{\alpha_{t-1,k}} - \epsilon_t e^{-\alpha_{t-1,k}} > \frac{\Lambda_k m}{S_t}. \qquad (16)
\end{aligned}$$

Note that $\eta^* \leq \eta^0$, where $\eta^0 = \log\left[\sqrt{\frac{1-\epsilon_t}{\epsilon_t}}\right]$ is the step size used is AdaBoost.

The case $\alpha_{t-1,k} + \eta^* < 0$ can be treated similarly. It is equivalent to the condition

$$(1-\epsilon_t)e^{\alpha_{t-1,k}} - \epsilon_t e^{-\alpha_{t-1,k}} < -\frac{\Lambda_k m}{S_t}, \tag{17}$$

and leads to the step size

$$\eta^* = \log\left[\frac{\Lambda_k m}{2\epsilon_t S_t} + \sqrt{\left(\frac{\Lambda_k m}{2\epsilon_t S_t}\right)^2 + \frac{1-\epsilon_t}{\epsilon_t}}\right].$$

B.5. Logistic loss

In the case of logistic loss, for any $u \in \mathbb{R}$, $\Phi(-u) = \log_2(1 + e^{-u})$ and $\Phi'(-u) = \frac{1}{\log 2}\frac{1}{(1+e^u)}$. To determine the step size, we use the following general upper bound:

$$\begin{aligned}
\Phi(-u-v) - \Phi(-u) &= \log_2\left[\frac{1+e^{-u-v}}{1+e^{-u}}\right]\\
&= \log_2\left[\frac{1+e^{-u}+e^{-u-v}-e^{-u}}{1+e^{-u}}\right]\\
&= \log_2\left[1 + \frac{e^{-v}-1}{1+e^{u}}\right]\\
&\leq \frac{e^{-v}-1}{(\log 2)(1+e^u)}\\
&= \Phi'(-u)(e^{-v}-1).
\end{aligned}$$

Thus, we can write

$$\begin{aligned}
&F(\boldsymbol{\alpha}_{t-1} + \eta\mathbf{e}_t) - F(\boldsymbol{\alpha}_{t-1})\\
&\leq \frac{1}{m}\sum_{i=1}^m \Phi'(1 - y_i f_{t-1}(x_i))(e^{-\eta y_i h_k(x_i)} - 1)\\
&\quad + \Lambda_k(|\alpha_{t-1,k} + \eta| - |\alpha_{t-1,k}|)\\
&= \frac{1}{m}\sum_{i=1}^m \mathcal{D}_t(i)S_t(e^{-\eta y_i h_k(x_i)} - 1)\\
&\quad + \Lambda_k(|\alpha_{t-1,k} + \eta| - |\alpha_{t-1,k}|).
\end{aligned}$$

To determine η, we can minimize this upper bound, or equivalently the following

$$\frac{1}{m}\sum_{i=1}^m \mathcal{D}_t(i)S_t\, e^{-\eta y_i h_k(x_i)} + \Lambda_k|\alpha_{t-1,k} + \eta|.$$

This expression is syntactically the same as the one considered in the case of the exponential loss with only the distribution weights $\mathcal{D}_t(i)$ and S_t being different. Indeed, in the case of the exponential loss ($\Phi = \exp$), we can write

$$F(\boldsymbol{\alpha}_{t-1} + \eta \mathbf{e}_k) - \sum_{j \neq k} \Lambda_j |\alpha_{t-1,j}|$$

$$= \frac{1}{m} \sum_{i=1}^{m} \Phi(1 - y_i f_{t-1}(x_i) - \eta y_i h_k(x_i)) + \Lambda_k |\alpha_{t-1,k} + \eta|,$$

$$= \frac{1}{m} \sum_{i=1}^{m} \Phi(1 - y_i f_{t-1}(x_i))\, e^{-\eta y_i h_k(x_i)} + \Lambda_k |\alpha_{t-1,k} + \eta|,$$

$$= \frac{1}{m} \sum_{i=1}^{m} \Phi'(1 - y_i f_{t-1}(x_i))\, e^{-\eta y_i h_k(x_i)} + \Lambda_k |\alpha_{t-1,k} + \eta|,$$

$$= \frac{1}{m} \sum_{i=1}^{m} D_t(i) S_t\, e^{-\eta y_i h_k(x_i)} + \Lambda_k |\alpha_{t-1,k} + \eta|.$$

Thus, we obtain immediately the same expressions for the step size in the case of the logistic loss with the same three cases, but with $S_t = \sum_{i=1}^{m} \frac{1}{1+e^{-1+y_i f_{t-1}(x_i)}}$ and $\mathcal{D}_t(i) = \frac{1}{S_t} \frac{1}{1+e^{-1+y_i f_{t-1}(x_i)}}$.

C. Alternative DeepBoost$_\gamma$ algorithm

We also devised and implemented an alternative algorithm, DeepBoost$_\gamma$, which is inspired by the learning bound of Theorem 1 but does not seek to minimize it. The algorithm admits a parameter $\gamma > 0$ representing the edge value demanded at each boosting round. This is the amount by which we require the error ϵ_t of the base hypothesis h_t selected at round t to be better than $\frac{1}{2}$: $\epsilon_t - \frac{1}{2} > \gamma$. We assume given p distinct hypothesis sets with increasing degrees of complexity $H_1, \ldots, H_p$. DeepBoost$_\gamma$ proceeds as if we were running AdaBoost using only as base hypothesis set H_1. But, at each round, if the edge achieved by the best hypothesis found in H_1 is not sufficient, that is if it is not larger than the demanded edge γ, then it selects instead the hypothesis in H_2 with the smallest error on the sample weighted by D_t. If the edge of that hypothesis is also not sufficient, it proceeds with the next hypothesis set and so forth. If the edge is insufficient even with the best hypothesis in H_p, then it just uses the best hypothesis found in $H = \bigcup_{k=1}^{p} H_k$. The edge parameter γ is determined via cross-validation.

DeepBoost$_\gamma$ is inspired by the bound of Theorem 1 since it seeks to use as much as possible hypotheses from H_1 or lower complexity families and only when necessary functions from more complex families. Since it tends to choose rarely hypotheses from more complex H_ks, the complexity term of the bound of Theorem 1 remains close to the one using only H_1. On the other hand, DeepBoost$_\gamma$ can achieve a smaller empirical margin loss (first term of the bound) by selecting, when needed, more powerful hypotheses than those accessible using H_1 alone.

We carried out some early experiments on several datasets with DeepBoost$_\gamma$ using boosting stumps, in which the performance of the algorithm was found to be superior to that of AdaBoost. A more extensive study of the theoretical and empirical properties of this algorithm are left to the future.

Table 4. Dataset statistics. `german` refers more specifically to the `german (numeric)` dataset.

	`breastcancer`	`ionosphere`	`german`
Examples	699	351	1000
Attributes	9	34	24

	`diabetes`	`ocr17`	`ocr49`
Examples	768	2000	2000
Attributes	8	196	196

	`ocr17-mnist`	`ocr49-mnist`
Examples	15170	13782
Attributes	400	400

D. Additional empirical information

D.1. Dataset sizes and attributes

The size and the number of attributes for the datasets used in our experiments are indicated in Table 4.