Multi-Class Deep Boosting

Vitaly Kuznetsov
Courant Institute
251 Mercer Street
New York, NY 10012
vitaly@cims.nyu.edu

Mehryar Mohri
Courant Institute & Google Research
251 Mercer Street
New York, NY 10012
mohri@cims.nyu.edu

Umar Syed
Google Research
76 Ninth Avenue
New York, NY 10011
usyed@google.com

Abstract

We present new ensemble learning algorithms for multi-class classification. Our algorithms can use as a base classifier set a family of deep decision trees or other rich or complex families and yet benefit from strong generalization guarantees. We give new data-dependent learning bounds for convex ensembles in the multi-class classification setting 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. These bounds are finer than existing ones both thanks to an improved dependency on the number of classes and, more crucially, by virtue of a more favorable complexity term expressed as an average of the Rademacher complexities based on the ensemble's mixture weights. We introduce and discuss several new multi-class ensemble algorithms benefiting from these guarantees, prove positive results for the H-consistency of several of them, and report the results of experiments showing that their performance compares favorably with that of multi-class versions of AdaBoost and Logistic Regression and their L_1-regularized counterparts.

1 Introduction

Devising ensembles of base predictors is a standard approach in machine learning which often helps improve performance in practice. Ensemble methods include the family of boosting meta-algorithms among which the most notable and widely used one is AdaBoost [Freund and Schapire, 1997], also known as forward stagewise additive modeling [Friedman et al., 1998]. AdaBoost and its other variants learn convex combinations of predictors. They seek to greedily minimize a convex surrogate function upper bounding the misclassification loss by augmenting, at each iteration, the current ensemble, with a new suitably weighted predictor.

One key advantage of AdaBoost is that, since it is based on a stagewise procedure, it can learn an effective ensemble of base predictors chosen from a very large and potentially infinite family, provided that an efficient algorithm is available for selecting a good predictor at each stage. Furthermore, AdaBoost and its L_1-regularized counterpart [Rätsch et al., 2001a] benefit from favorable learning guarantees, in particular theoretical margin bounds [Schapire et al., 1997, Koltchinskii and Panchenko, 2002]. However, those bounds depend not just on the margin and the sample size, but also on the complexity of the base hypothesis set, which suggests a risk of overfitting when using too complex base hypothesis sets. And indeed, overfitting has been reported in practice for AdaBoost in the past [Grove and Schuurmans, 1998, Schapire, 1999, Dietterich, 2000, Rätsch et al., 2001b].

Cortes, Mohri, and Syed [2014] introduced a new ensemble algorithm, *DeepBoost*, which they proved to benefit from finer learning guarantees, including favorable ones even when using as base classifier set relatively rich families, for example a family of very deep decision trees, or other similarly complex families. In DeepBoost, the decisions in each iteration of which classifier to add to the ensemble and which weight to assign to that classifier, depend on the (data-dependent) complexity of the sub-family to which the classifier belongs – one interpretation of DeepBoost is that it applies the principle of structural risk minimization to each iteration of boosting. Cortes, Mohri, and Syed [2014] further showed that empirically DeepBoost achieves a better performance than AdaBoost, Logistic Regression, and their L_1-regularized variants. The main contribution of this paper is an extension of these theoretical, algorithmic, and empirical results to the multi-class setting.

Two distinct approaches have been considered in the past for the definition and the design of boosting algorithms in the multi-class setting. One approach consists of combining base classifiers mapping each example x to an output label y. This includes the SAMME algorithm [Zhu et al., 2009] as well as the algorithm of Mukherjee and Schapire [2013], which is shown to be, in a certain sense, optimal for this approach. An alternative approach, often more flexible and more widely used in applications, consists of combining base classifiers mapping each pair (x, y) formed by an example x and a label y to a real-valued score. This is the approach adopted in this paper, which is also the one used for the design of AdaBoost.MR [Schapire and Singer, 1999] and other variants of that algorithm.

In Section 2, we prove a novel generalization bound for multi-class classification ensembles that depends only on the Rademacher complexity of the hypothesis classes to which the classifiers in the ensemble belong. Our result generalizes the main result of Cortes et al. [2014] to the multi-class setting, and also represents an improvement on the multi-class generalization bound due to Koltchinskii and Panchenko [2002], even if we disregard our finer analysis related to Rademacher complexity. In Section 3, we present several multi-class surrogate losses that are motivated by our generalization bound, and discuss and compare their functional and consistency properties. In particular, we prove that our surrogate losses are *realizable H-consistent*, a hypothesis-set-specific notion of consistency that was recently introduced by Long and Servedio [2013]. Our results generalize those of Long and Servedio [2013] and admit simpler proofs. We also present a family of multi-class DeepBoost learning algorithms based on each of these surrogate losses, and prove general convergence guarantee for them. In Section 4, we report the results of experiments demonstrating that multi-class DeepBoost outperforms AdaBoost.MR and multinomial (additive) logistic regression, as well as their L_1-norm regularized variants, on several datasets.

2 Multi-class data-dependent learning guarantee for convex ensembles

In this section, we present a data-dependent learning bound in the multi-class setting for convex ensembles based on multiple base hypothesis sets. Let $\mathcal{X}$ denote the input space. We denote by $\mathcal{Y} = \{1, \ldots, c\}$ a set of $c \geq 2$ classes. The label associated by a hypothesis $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ to $x \in \mathcal{X}$ is given by $\operatorname{argmax}_{y \in \mathcal{Y}} f(x, y)$. The margin $\rho_f(x, y)$ of the function f for a labeled example $(x, y) \in \mathcal{X} \times \mathcal{Y}$ is defined by

$$\rho_f(x, y) = f(x, y) - \max_{y' \neq y} f(x, y'). \tag{1}$$

Thus, f misclassifies (x, y) iff $\rho_f(x, y) \leq 0$. We consider p families $H_1, \ldots, H_p$ of functions mapping from $\mathcal{X} \times \mathcal{Y}$ to $[0, 1]$ and the ensemble family $\mathcal{F} = \operatorname{conv}(\bigcup_{k=1}^{p} H_k)$, 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 $k_t \in [1, p]$. We assume that training and test points are drawn i.i.d. according to some distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$ and denote by $S = ((x_1, y_1), \ldots, (x_m, y_m))$ a training sample of size m drawn according to $\mathcal{D}^m$. For any $\rho > 0$, the generalization error $R(f)$, its ρ-margin error $R_\rho(f)$ and its empirical margin error are defined as follows:

$$R(f) = \mathop{\mathrm{E}}_{(x,y)\sim\mathcal{D}}[1_{\rho_f(x,y)\leq 0}], \quad R_\rho(f) = \mathop{\mathrm{E}}_{(x,y)\sim\mathcal{D}}[1_{\rho_f(x,y)\leq \rho}], \text{ and } \widehat{R}_{S,\rho}(f) = \mathop{\mathrm{E}}_{(x,y)\sim S}[1_{\rho_f(x,y)\leq \rho}], \tag{2}$$

where the notation $(x, y) \sim S$ indicates that (x, y) is drawn according to the empirical distribution defined by S. For any family of hypotheses G mapping $\mathcal{X} \times \mathcal{Y}$ to $\mathbb{R}$, we define $\Pi_1(G)$ by

$$\Pi_1(G) = \{x \mapsto h(x, y) : y \in \mathcal{Y}, h \in G\}. \tag{3}$$

The following theorem gives a margin-based Rademacher complexity bound for learning with ensembles of base classifiers with multiple hypothesis sets. As with other Rademacher complexity learning guarantees, our bound is data-dependent, which is an important and favorable characteristic of our results.

Theorem 1. *Assume $p > 1$ and let $H_1, \ldots, H_p$ be p families of functions mapping from $\mathcal{X} \times \mathcal{Y}$ to $[0, 1]$. 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 $f = \sum_{t=1}^{T} \alpha_t h_t \in \mathcal{F}$:*

$$R(f) \leq \widehat{R}_{S,\rho}(f) + \frac{8c}{\rho} \sum_{t=1}^{T} \alpha_t \mathfrak{R}_m(\Pi_1(H_{k_t})) + \frac{2}{c\rho} \sqrt{\frac{\log p}{m}} + \sqrt{\left\lceil \tfrac{4}{\rho^2} \log\left(\tfrac{c^2 \rho^2 m}{4 \log p}\right) \right\rceil \frac{\log p}{m} + \frac{\log \frac{2}{\delta}}{2m}},$$

Thus, $R(f) \leq \widehat{R}_{S,\rho}(f) + \frac{8c}{\rho} \sum_{t=1}^{T} \alpha_t \mathfrak{R}_m(H_{k_t}) + O\left(\sqrt{\frac{\log p}{\rho^2 m} \log\left[\frac{\rho^2 c^2 m}{4 \log p}\right]}\right)$.

The full proof of theorem 3 is given in Appendix B. Even for $p = 1$, that is for the special case of a single hypothesis set, our analysis improves upon the multi-class margin bound of Koltchinskii and Panchenko [2002] since our bound admits only a linear dependency on the number of classes c instead of a quadratic one. However, the main remarkable benefit of this learning bound is that its complexity term 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 negatively affect generalization if the corresponding total mixture weight (sum of α_ts corresponding to that hypothesis set) is relatively small. Using such potentially complex families could help achieve a better margin on the training sample.

The theorem cannot be proven via the standard Rademacher complexity analysis of Koltchinskii and Panchenko [2002] since the complexity term of the bound would then be $\mathfrak{R}_m(\mathrm{conv}(\bigcup_{k=1}^{p} H_k)) = \mathfrak{R}_m(\bigcup_{k=1}^{p} H_k)$ which does not admit an explicit dependency on the mixture weights and is lower bounded by $\sum_{t=1}^{T} \alpha_t \mathfrak{R}_m(H_{k_t})$. Thus, the theorem provides a finer learning bound than the one obtained via a standard Rademacher complexity analysis.

3 Algorithms

In this section, we will use the learning guarantees just described to derive several new ensemble algorithms for multi-class classification.

3.1 Optimization problem

Let $H_1, \ldots, H_p$ be p disjoint families of functions taking values in $[0, 1]$ with increasing Rademacher complexities $\mathfrak{R}_m(H_k)$, $k \in [1, p]$. 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 3 holds uniformly for all $\rho > 0$ and functions $f \in \mathrm{conv}(\bigcup_{k=1}^{p} H_k)$. Since the last term of the bound does not depend on $\boldsymbol{\alpha}$, it suggests selecting $\boldsymbol{\alpha}$ that would minimize:

$$G(\boldsymbol{\alpha}) = \frac{1}{m} \sum_{i=1}^{m} 1_{\rho_f(x_i, y_i) \leq \rho} + \frac{8c}{\rho} \sum_{t=1}^{T} \alpha_t r_t,$$

where $r_t = \mathfrak{R}_m(H_{d(h_t)})$ and $\boldsymbol{\alpha} \in \Delta$.[1] 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

$$\min_{\boldsymbol{\alpha} \geq 0} \frac{1}{m} \sum_{i=1}^{m} 1_{\rho_f(x_i, y_i) \leq 1} + 8c \sum_{t=1}^{T} \alpha_t r_t \quad \text{s.t.} \sum_{t=1}^{T} \alpha_t \leq \frac{1}{\rho}. \tag{4}$$

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 $u \mapsto 1_{u \leq 0}$ over $\mathbb{R}$. Φ may be selected to be

[1] The condition $\sum_{t=1}^{T} \alpha_t = 1$ of Theorem 3 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).

for example the exponential function as in AdaBoost [Freund and Schapire, 1997] or the logistic function. Using such an upper bound, we obtain the following convex optimization problem:

$$\min_{\boldsymbol{\alpha} \geq 0} \frac{1}{m} \sum_{i=1}^{m} \Phi\Big(1 - \rho_f(x_i, y_i)\Big) + \lambda \sum_{t=1}^{T} \alpha_t r_t \quad \text{s.t.} \sum_{t=1}^{T} \alpha_t \leq \frac{1}{\rho}, \tag{5}$$

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.[2] Introducing a Lagrange variable $\beta \geq 0$ associated to the constraint in (5), the problem can be equivalently written as

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

Here, β is a parameter that can be freely selected by the algorithm since any choice of its value is equivalent to a choice of ρ in (5). Since Φ is a non-decreasing function, the problem can be equivalently written as

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

Let $\{h_1, \ldots, h_N\}$ be the set of distinct base functions, and let $F_{\max}$ be the objective function based on that expression:

$$F_{\max}(\boldsymbol{\alpha}) = \frac{1}{m} \sum_{i=1}^{m} \max_{y \neq y_i} \Phi\Big(1 - \sum_{j=1}^{N} \alpha_j \mathsf{h}_j(x_i, y_i, y)\Big) + \sum_{j=1}^{N} \Lambda_j \alpha_j, \tag{6}$$

with $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_N) \in \mathbb{R}^N$, $\mathsf{h}_j(x_i, y_i, y) = h_j(x_i, y_i) - h_j(x_i, y)$, and $\Lambda_j = \lambda r_j + \beta$ for all $j \in [1, N]$. Then, our optimization problem can be rewritten as $\min_{\boldsymbol{\alpha} \geq 0} F_{\max}(\boldsymbol{\alpha})$. This defines a convex optimization problem since the domain $\{\boldsymbol{\alpha} \geq 0\}$ is a convex set and since $F_{\max}$ is convex: each term of the sum in its definition is convex as a pointwise maximum of convex functions (composition of the convex function Φ with an affine function) and the second term is a linear function of $\boldsymbol{\alpha}$. In general, $F_{\max}$ is not differentiable even when Φ is, but, since it is convex, it admits a sub-differential at every point. Additionally, along each direction, $F_{\max}$ admits left and right derivatives both non-increasing and a differential everywhere except for a set that is at most countable.

3.2 Alternative objective functions

We now consider the following three natural upper bounds on $F_{\max}$ which admit useful properties that we will discuss later, the third one valid when Φ can be written as the composition of two function Φ_1 and Φ_2 with Φ_1 a non-increasing function:

$$F_{\text{sum}}(\boldsymbol{\alpha}) = \frac{1}{m} \sum_{i=1}^{m} \sum_{y \neq y_i} \Phi\Big(1 - \sum_{j=1}^{N} \alpha_j \mathsf{h}_j(x_i, y_i, y)\Big) + \sum_{j=1}^{N} \Lambda_j \alpha_j \tag{7}$$

$$F_{\text{maxsum}}(\boldsymbol{\alpha}) = \frac{1}{m} \sum_{i=1}^{m} \Phi\Big(1 - \sum_{j=1}^{N} \alpha_j \rho_{h_j}(x_i, y_i)\Big) + \sum_{j=1}^{N} \Lambda_j \alpha_j \tag{8}$$

$$F_{\text{compsum}}(\boldsymbol{\alpha}) = \frac{1}{m} \sum_{i=1}^{m} \Phi_1\Big(\sum_{y \neq y_i} \Phi_2\Big(1 - \sum_{j=1}^{N} \alpha_j \mathsf{h}_j(x_i, y_i, y)\Big)\Big) + \sum_{j=1}^{N} \Lambda_j \alpha_j. \tag{9}$$

F_{sum} is obtained from $F_{\max}$ simply by replacing in the definition of $F_{\max}$ the $\max$ operator by a sum. Clearly, function F_{sum} is convex and inherits the differentiability properties of Φ. A drawback of F_{sum} is that for problems with very large c as in structured prediction, the computation of the sum

[2] Note that this is a standard practice in the field of optimization. The optimization problem in (4) is equivalent to a vector optimization problem, where $(\sum_{i=1}^{m} 1_{\rho_f(x_i, y_i) \leq 1}, \sum_{t=1}^{T} \alpha_t r_t)$ is minimized over $\boldsymbol{\alpha}$. The latter problem can be scalarized leading to the introduction of a parameter λ in (5).

may require resorting to approximations. F_{maxsum} is obtained from $F_{\max}$ by noticing that, by the sub-additivity of the $\max$ operator, the following inequality holds:

$$\max_{y \neq y_i} \sum_{j=1}^{N} -\alpha_j \mathsf{h}_j(x_i, y_i, y) \leq \sum_{j=1}^{N} \max_{y \neq y_i} -\alpha_j \mathsf{h}_j(x_i, y_i, y) = \sum_{j=1}^{N} \alpha_j \rho_{h_j}(x_i, y_i).$$

As with F_{sum}, function F_{maxsum} is convex and admits the same differentiability properties as Φ. Unlike F_{sum}, F_{maxsum} does not require computing a sum over the classes. Furthermore, note that the expressions $\rho_{h_j}(x_i, y_i)$, $i \in [1, m]$, can be pre-computed prior to the application of any optimization algorithm. Finally, for $\Phi = \Phi_1 \circ \Phi_2$ with Φ_1 non-increasing, the $\max$ operator can be replaced by a sum before applying ϕ_1, as follows:

$$\max_{y \neq y_i} \Phi\Big(1 - \mathsf{f}(x_i, y_i, y)\Big) = \Phi_1\Big(\max_{y \neq y_i} \Phi_2\big(1 - \mathsf{f}(x_i, y_i, y)\big)\Big) \leq \Phi_1\Big(\sum_{y \neq y_i} \Phi_2\big(1 - \mathsf{f}(x_i, y_i, y)\big)\Big),$$

where $\mathsf{f}(x_i, y_i, y) = \sum_{j=1}^{N} \alpha_j \mathsf{h}_j(x_i, y_i, y)$. This leads to the definition of F_{compsum}.

In Appendix C, we discuss the consistency properties of the loss functions just introduced. In particular, we prove that the loss functions associated to $F_{\max}$ and F_{sum} are *realizable H-consistent* (see Long and Servedio [2013]) in the common cases where the exponential or logistic losses are used and that, similarly, in the common case where $\Phi_1(u) = \log(1 + u)$ and $\Phi_2(u) = \exp(u + 1)$, the loss function associated to F_{compsum} is H-consistent.

Furthermore, in Appendix D, we show that, under some mild assumptions, the objective functions we just discussed are essentially within a constant factor of each other. Moreover, in the case of binary classification all of these objectives coincide.

3.3 Multi-class DeepBoost algorithms

In this section, we discuss in detail a family of multi-class DeepBoost algorithms, which are derived by application of coordinate descent to the objective functions discussed in the previous paragraphs. We will assume that Φ is differentiable over $\mathbb{R}$ and that $\Phi'(u) \neq 0$ for all u. This condition is not necessary, in particular, our presentation can be extended to non-differentiable functions such as the hinge loss, but it simplifies the presentation. In the case of the objective function F_{maxsum}, we will assume that both Φ_1 and Φ_2, where $\Phi = \Phi_1 \circ \Phi_2$, are differentiable. Under these assumptions, F_{sum}, F_{maxsum}, and F_{compsum} are differentiable. $F_{\max}$ is not differentiable due to the presence of the $\max$ operators in its definition, but it admits a sub-differential at every point.

For convenience, 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 $\mathbb{R}^N$, $k \in [1, N]$. For a differentiable objective F, we denote by $F'(\boldsymbol{\alpha}, \mathbf{e}_j)$ the directional derivative of F along the direction $\mathbf{e}_j$ at $\boldsymbol{\alpha}$. Our coordinate descent algorithm consists of first determining the direction of maximal descent, that is $k = \operatorname{argmax}_{j \in [1,N]} |F'(\boldsymbol{\alpha}_{t-1}, \mathbf{e}_j)|$, next of determining the best step η along that direction that preserves non-negativity of α, $\eta = \operatorname{argmin}_{\boldsymbol{\alpha}_{t-1} + \eta \mathbf{e}_k \geq 0} F(\boldsymbol{\alpha}_{t-1} + \eta \mathbf{e}_k)$, and updating $\boldsymbol{\alpha}_{t-1}$ to $\boldsymbol{\alpha}_t = \boldsymbol{\alpha}_{t-1} + \eta \mathbf{e}_k$. We will refer to this method as *projected coordinate descent*. The following theorem provides a convergence guarantee for our algorithms in that case.

Theorem 2. *Assume that Φ is twice differentiable and that $\Phi''(u) > 0$ for all $u \in \mathbb{R}$. Then, the projected coordinate descent algorithm applied to F converges to the solution $\boldsymbol{\alpha}^*$ of the optimization $\max_{\boldsymbol{\alpha} \geq 0} F(\boldsymbol{\alpha})$ for $F = F_{\text{sum}}$, $F = F_{\text{maxsum}}$, or $F = F_{\text{compsum}}$. If additionally Φ is strongly convex over the path of the iterates $\boldsymbol{\alpha}_t$, then there exists $\tau > 0$ and $\gamma > 0$ such that for all $t > \tau$,*

$$F(\boldsymbol{\alpha}_{t+1}) - F(\boldsymbol{\alpha}^*) \leq (1 - \tfrac{1}{\gamma})(F(\boldsymbol{\alpha}_t) - F(\boldsymbol{\alpha}^*)). \tag{10}$$

The proof is given in Appendix I and is based on the results of Luo and Tseng [1992]. The theorem can in fact be extended to the case where instead of the best direction, the derivative for the direction selected at each round is within a constant threshold of the best [Luo and Tseng, 1992]. The conditions of Theorem 2 hold for many cases in practice, in particular in the case of the exponential loss ($\Phi = \exp$) or the logistic loss ($\Phi(-x) = \log_2(1 + e^{-x})$). In particular, linear convergence is guaranteed in those cases since both the exponential and logistic losses are strongly convex over a compact set containing the converging sequence of $\boldsymbol{\alpha}_t$s.

```
MDeepBoostSum($S = ((x_1, y_1), \ldots, (x_m, y_m))$)
 1  for $i \leftarrow 1$ to $m$ do
 2      for $y \in \mathcal{Y} - \{y_i\}$ do
 3          $\mathcal{D}_1(i, y) \leftarrow \frac{1}{m(c-1)}$
 4  for $t \leftarrow 1$ to $T$ do
 5      $k \leftarrow \underset{j \in [1,N]}{\text{argmin}}\ \epsilon_{t,j} + \frac{\Lambda_j m}{2S_t}$
 6      if $\big((1 - \epsilon_{t,k})e^{\alpha_{t-1,k}} - \epsilon_{t,k}e^{-\alpha_{t-1,k}} < \frac{\Lambda_k m}{S_t}\big)$ then
 7          $\eta_t \leftarrow -\alpha_{t-1,k}$
 8      else $\eta_t \leftarrow \log\Big[-\frac{\Lambda_k m}{2\epsilon_t S_t} + \sqrt{\big[\frac{\Lambda_k m}{2\epsilon_t S_t}\big]^2 + \frac{1-\epsilon_t}{\epsilon_t}}\Big]$
 9      $\boldsymbol{\alpha}_t \leftarrow \boldsymbol{\alpha}_{t-1} + \eta_t \mathbf{e}_k$
10      $S_{t+1} \leftarrow \sum_{i=1}^m \sum_{y \neq y_i} \Phi'\big(1 - \sum_{j=1}^N \alpha_{t,j}\mathsf{h}_j(x_i, y_i, y)\big)$
11      for $i \leftarrow 1$ to $m$ do
12          for $y \in \mathcal{Y} - \{y_i\}$ do
13              $\mathcal{D}_{t+1}(i, y) \leftarrow \frac{\Phi'\big(1 - \sum_{j=1}^N \alpha_{t,j}\mathsf{h}_j(x_i, y_i, y)\big)}{S_{t+1}}$
14  $f \leftarrow \sum_{j=1}^N \alpha_{t,j} h_j$
15  return $f$
```

Figure 1: Pseudocode of the MDeepBoostSum algorithm for both the exponential loss and the logistic loss. The expression of the weighted error $\epsilon_{t,j}$ is given in (12).

We will refer to the algorithm defined by projected coordinate descent applied to F_{sum} by MDeepBoostSum, to F_{maxsum} by MDeepBoostMaxSum, to F_{compsum} by MDeepBoostCompSum, and to $F_{\max}$ by MDeepBoostMax. In the following, we briefly describe MDeepBoostSum, including its pseudocode. We give a detailed description of all of these algorithms in the supplementary material: MDeepBoostSum (Appendix E), MDeepBoostMaxSum (Appendix F), MDeepBoostCompSum (Appendix G), MDeepBoostMax (Appendix H).

Define $\mathsf{f}_{t-1} = \sum_{j=1}^N \alpha_{t-1,j}\mathsf{h}_j$. Then, $F_{\text{sum}}(\boldsymbol{\alpha}_{t-1})$ can be rewritten as follows:

$$F_{\text{sum}}(\boldsymbol{\alpha}_{t-1}) = \frac{1}{m}\sum_{i=1}^m \sum_{y \neq y_i} \Phi\Big(1 - \mathsf{f}_{t-1}(x_i, y_i, y)\Big) + \sum_{j=1}^N \Lambda_j \alpha_{t-1,j}.$$

For any $t \in [1, T]$, we denote by $\mathcal{D}_t$ the distribution over $[1, m] \times [1, c]$ defined for all $i \in [1, m]$ and $y \in \mathcal{Y} - \{y_i\}$ by

$$\mathcal{D}_t(i, y) = \frac{\Phi'\big(1 - \mathsf{f}_{t-1}(x_i, y_i, y)\big)}{S_t}, \tag{11}$$

where S_t is a normalization factor, $S_t = \sum_{i=1}^m \sum_{y \neq y_i} \Phi'(1 - \mathsf{f}_{t-1}(x_i, y_i, y))$. For any $j \in [1, N]$ and $s \in [1, T]$, we also define the weighted error $\epsilon_{s,j}$ as follows:

$$\epsilon_{s,j} = \frac{1}{2}\Big[1 - \underset{(i,y) \sim \mathcal{D}_s}{\mathrm{E}}\big[\mathsf{h}_j(x_i, y_i, y)\big]\Big]. \tag{12}$$

Figure 1 gives the pseudocode of the MDeepBoostSum algorithm. The details of the derivation of the expressions are given in Appendix E. 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 6-8), which is the same in both cases (see Sections E.2.1 and E.2.2). In the generic case, the step size can be found using a line search or other numerical methods.

The algorithms presented above have several connections with other boosting algorithms, particularly in the absence of regularization. We discuss these connections in detail in Appendix K.

4 Experiments

The algorithms presented in the previous sections can be used with a variety of different base classifier sets. For our experiments, we used multi-class binary decision trees. A multi-class binary decision tree in dimension d can be defined by a pair $(\mathtt{t}, \mathbf{h})$, where $\mathtt{t}$ is a binary tree with a variable-threshold question at each internal node, e.g., $X_j \leq \theta$, $j \in [1, d]$, and $\mathbf{h} = (h_l)_{l \in \text{Leaves}(\mathtt{t})}$ a vector of distributions over the leaves Leaves($\mathtt{t}$) of $\mathtt{t}$. At any leaf $l \in \text{Leaves}(\mathtt{t})$, $h_l(y) \in [0, 1]$ for all $y \in \mathcal{Y}$ and $\sum_{y \in \mathcal{Y}} h_l(y) = 1$. For convenience, we will denote by $\mathtt{t}(x)$ the leaf $l \in \text{Leaves}(\mathtt{t})$ associated to x by $\mathtt{t}$. Thus, the score associated by $(\mathtt{t}, \mathbf{h})$ to a pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$ is $h_l(y)$ where $l = \mathtt{t}(x)$.

Let $\mathcal{T}_n$ denote the family of all multi-class decision trees with n internal nodes in dimension d. In Appendix J, we derive the following upper bound on the Rademacher complexity of $\mathcal{T}_n$:

$$\mathfrak{R}(\Pi_1(\mathcal{T}_n)) \leq \sqrt{\frac{(4n+2)\log_2(d+2)\log(m+1)}{m}}. \tag{13}$$

All of the experiments in this section use $\mathcal{T}_n$ as the family of base hypothesis sets (parametrized by n). Since $\mathcal{T}_n$ is a very large hypothesis set when n is large, for the sake of computational efficiency we make a few approximations. First, although our MDeepBoost algorithms were derived in terms of Rademacher complexity, we use the upper bound in Eq. (13) in place of the Rademacher complexity (thus, in Algorithm 1 we let $\Lambda_n = \lambda B_n + \beta$, where B_n is the bound given in Eq. (13)). Secondly, instead of exhaustively searching for the best decision tree in $\mathcal{T}_n$ for each possible size n, we use the following greedy procedure: Given the best decision tree of size n (starting with $n = 1$), we find the best decision tree of size $n+1$ that can be obtained by splitting one leaf, and continue this procedure until some maximum depth K. Decision trees are commonly learned in this manner, and so in this context our Rademacher-complexity-based bounds can be viewed as a novel stopping criterion for decision tree learning. Let H_K^* be the set of trees found by the greedy algorithm just described. In each iteration t of MDeepBoost, we select the best tree in the set $H_K^* \cup \{h_1, \ldots, h_{t-1}\}$, where $h_1, \ldots, h_{t-1}$ are the trees selected in previous iterations.

While we described many objective functions that can be used as the basis of a multi-class deep boosting algorithm, the experiments in this section focus on algorithms derived from F_{sum}. We also refer the reader to Table 3 in Appendix A for results of experiments with F_{compsum} objective functions. The F_{sum} and F_{compsum} objectives combine several advantages that suggest they will perform well empirically. F_{sum} is consistent and both F_{sum} and F_{compsum} are (by Theorem 4) H-consistent. Also, unlike $F_{\max}$ both of these objectives are differentiable, and therefore the convergence guarantee in Theorem 2 applies. Our preliminary findings also indicate that algorithms based on F_{sum} and F_{compsum} objectives perform better than those derived from $F_{\max}$ and F_{maxsum}. All of our objective functions require a choice for Φ, the loss function. Since Cortes et al. [2014] reported comparable results for exponential and logistic loss for the binary version of DeepBoost, we let Φ be the exponential loss in all of our experiments with MDeepBoostSum. For MDeepBoostCompSum we select $\Phi_1(u) = \log_2(1+u)$ and $\Phi_2(-u) = \exp(-u)$.

In our experiments, we used 8 UCI data sets: `abalone`, `handwritten`, `letters`, `pageblocks`, `pendigits`, `satimage`, `statlog` and `yeast` – see more details on these datasets in Table 4, Appendix L. In Appendix K, we explain that when $\lambda = \beta = 0$ then MDeepBoostSum is equivalent to AdaBoost.MR. Also, if we set $\lambda = 0$ and $\beta \neq 0$ then the resulting algorithm is an L_1-norm regularized variant of AdaBoost.MR. We compared MDeepBoostSum to these two algorithms, with the results also reported in Table 1 and Table 2 in Appendix A. Likewise, we compared MDeepBoost-CompSum with multinomial (additive) logistic regression, LogReg, and its L_1-regularized version LogReg-L1, which, as discussed in Appendix K, are equivalent to MDeepBoostCompSum when $\lambda = \beta = 0$ and $\lambda = 0, \beta \geq 0$ respectively. Finally, we remark that it can be argued that the parameter optimization procedure (described below) significantly extends AdaBoost.MR since it effectively implements structural risk minimization: for each tree depth, the empirical error is minimized and we choose the depth to achieve the best generalization error.

All of these algorithms use maximum tree depth K as a parameter. L_1-norm regularized versions admit two parameters: K and $\beta \geq 0$. Deep boosting algorithms have a third parameter, $\lambda \geq 0$. To set these parameters, we used the following parameter optimization procedure: we randomly partitioned each dataset into 4 folds and, for each tuple (λ, β, K) in the set of possible parameters (described below), we ran MDeepBoostSum, with a different assignment of folds to the training

Table 1: Empirical results for MDeepBoostSum, $\Phi = \exp$. AB stands for AdaBoost.

abalone	AB.MR	AB.MR-L1	MDeepBoost
Error	0.739	0.737	0.735
(std dev)	(0.0016)	(0.0065)	(0.0045)

handwritten	AB.MR	AB.MR-L1	MDeepBoost
Error	0.024	0.025	0.021
(std dev)	(0.0011)	(0.0018)	(0.0015)

letters	AB.MR	AB.MR-L1	MDeepBoost
Error	0.065	0.059	0.058
(std dev)	(0.0018)	(0.0059)	(0.0039)

pageblocks	AB.MR	AB.MR-L1	MDeepBoost
Error	0.035	0.035	0.033
(std dev)	(0.0045)	(0.0031)	(0.0014)

pendigits	AB.MR	AB.MR-L1	MDeepBoost
Error	0.014	0.014	0.012
(std dev)	(0.0025)	(0.0013)	(0.0011)

satimage	AB.MR	AB.MR-L1	MDeepBoost
Error	0.112	0.117	0.117
(std dev)	(0.0123)	(0.0096)	(0.0087)

statlog	AB.MR	AB.MR-L1	MDeepBoost
Error	0.029	0.026	0.024
(std dev)	(0.0026)	(0.0071)	(0.0008)

yeast	AB.MR	AB.MR-L1	MDeepBoost
Error	0.415	0.410	0.407
(std dev)	(0.0353)	(0.0324)	(0.0282)

set, validation set and test set for each run. Specifically, for each run $i \in \{0, 1, 2, 3\}$, fold i was used for testing, fold $i + 1 \pmod 4$ 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 test error and the standard deviation of the test error over all 4 runs is reported in Table 1. Note that an alternative procedure to compare algorithms that is adopted in a number of previous studies of boosting [Li, 2009a,b, Sun et al., 2012] is to simply record the average test error of the best parameter tuples over all runs. While it is of course possible to overestimate the performance of a learning algorithm by optimizing hyperparameters on the test set, this concern is less valid when the size of the test set is large relative to the "complexity" of the hyperparameter space. We report results for this alternative procedure in Table 2 and Table 3, Appendix A.

For each dataset, the set of possible values for λ and β was initialized to $\{10^{-5}, 10^{-6}, \ldots, 10^{-10}\}$, and to $\{1, 2, 3, 4, 5\}$ for the maximum tree depth K. However, if we found an optimal parameter value to be at the end point of these ranges, we extended the interval in that direction (by an order of magnitude for λ and β, and by 1 for the maximum tree depth K) and re-ran the experiments. We have also experimented with 200 and 500 iterations but we have observed that the errors do not change significantly and the ranking of the algorithms remains the same.

The results of our experiments show that, for each dataset, deep boosting algorithms outperform the other algorithms evaluated in our experiments. Let us point out that, even though not all of our results are statistically significant, MDeepBoostSum outperforms AdaBoost.MR and AdaBoost.MR-L1 (and, hence, effectively structural risk minimization) on each dataset. More importantly, for each dataset MDeepBoostSum outperforms other algorithms on most of the individual runs. Moreover, results for some datasets presented here (namely `pendigits`) appear to be state-of-the-art. We also refer our reader to experimental results summarized in Table 2 and Table 3 in Appendix A. These results provide further evidence in favor of DeepBoost algorithms. The consistent performance improvement by MDeepBoostSum over AdaBoost.MR or its L1-norm regularized variant shows the benefit of the new complexity-based regularization we introduced.

5 Conclusion

We presented new data-dependent learning guarantees for convex ensembles in the multi-class setting where the base classifier set is composed of increasingly complex sub-families, including very deep or complex ones. These learning bounds generalize to the multi-class setting the guarantees presented by Cortes et al. [2014] in the binary case. We also introduced and discussed several new multi-class ensemble algorithms benefiting from these guarantees and proved positive results for the H-consistency and convergence of several of them. Finally, we reported the results of several experiments with DeepBoost algorithms, and compared their performance with that of AdaBoost.MR and additive multinomial Logistic Regression and their L_1-regularized variants.

Acknowledgments

We thank Andres Muñoz Medina and Scott Yang for discussions and help with the experiments. This work was partly funded by the NSF award IIS-1117591 and supported by a NSERC PGS grant.

References

P. Bühlmann and B. Yu. Boosting with the L2 loss. *J. of the Amer. Stat. Assoc.*, 98(462):324–339, 2003.

M. Collins, R. E. Schapire, and Y. Singer. Logistic regression, Adaboost and Bregman distances. *Machine Learning*, 48:253–285, September 2002.

C. Cortes, M. Mohri, and U. Syed. Deep boosting. In *ICML*, pages 1179 – 1187, 2014.

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

J. C. Duchi and Y. Singer. Boosting with structural sparsity. In *ICML*, page 38, 2009.

N. Duffy and D. P. Helmbold. Potential boosters? In *NIPS*, pages 258–264, 1999.

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

J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

J. H. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28:2000, 1998.

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

J. Kivinen and M. K. Warmuth. Boosting as entropy projection. In *COLT*, pages 134–144, 1999.

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

M. Ledoux and M. Talagrand. *Probability in Banach Spaces: Isoperimetry and Processes*. Springer, 1991.

P. Li. ABC-boost: adaptive base class boost for multi-class classification. In *ICML*, page 79, 2009a.

P. Li. ABC-logitboost for multi-class classification. Technical report, Rutgers University, 2009b.

P. M. Long and R. A. Servedio. Consistency versus realizable H-consistency for multiclass classification. In *ICML (3)*, pages 801–809, 2013.

Z.-Q. Luo and P. Tseng. On the convergence of coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications*, 72(1):7 – 35, 1992.

L. Mason, J. Baxter, P. L. Bartlett, and M. R. Frean. Boosting algorithms as gradient descent. In *NIPS*, 1999.

M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012.

I. Mukherjee and R. E. Schapire. A theory of multiclass boosting. *JMLR*, 14(1):437–497, 2013.

G. Rätsch and M. K. Warmuth. Maximizing the margin with boosting. In *COLT*, pages 334–350, 2002.

G. Rätsch and M. K. Warmuth. Efficient margin maximizing with boosting. *JMLR*, 6:2131–2152, 2005.

G. Rätsch, S. Mika, and M. K. Warmuth. On the convergence of leveraging. In *NIPS*, pages 487–494, 2001a.

G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for AdaBoost. *Machine Learning*, 42(3):287–320, 2001b.

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

R. E. Schapire and Y. Freund. *Boosting: Foundations and Algorithms*. The MIT Press, 2012.

R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. In *ICML*, pages 322–330, 1997.

P. Sun, M. D. Reid, and J. Zhou. Aoso-logitboost: Adaptive one-vs-one logitboost for multi-class problem. In *ICML*, 2012.

A. Tewari and P. L. Bartlett. On the consistency of multiclass classification methods. *JMLR*, 8:1007–1025, 2007.

M. K. Warmuth, J. Liao, and G. Rätsch. Totally corrective boosting algorithms that maximize the margin. In *ICML*, pages 1001–1008, 2006.

T. Zhang. Statistical analysis of some multi-category large margin classification methods. *JMLR*, 5:1225–1251, 2004a.

T. Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *Annals of Statistics*, 32(1):56–85, 2004b.

J. Zhu, H. Zou, S. Rosset, and T. Hastie. Multi-class adaboost. *Statistics and Its Interface*, 2009.

H. Zou, J. Zhu, and T. Hastie. New multicategory boosting algorithms based on multicategory fisher-consistent losses. *Annals of Statistics*, 2(4):1290–1306, 2008.