Proceedings of the Thirteenth Annual Conference on Computational Learning Theory, 2000

Barrier Boosting

Gunnar Rätsch[*], Manfred Warmuth[$], Sebastian Mika[*], Takashi Onoda[†], Steven Lemm[*] and Klaus-Robert Müller[*+]

[*]GMD FIRST, Kekuléstr. 7, 12489 Berlin, Germany
[$]Computer Science Department, UC Santa Cruz, Santa Cruz, CA 95064, USA
[†]CRIEPI, 2-11-1, Iwado Kita, Komae-shi, Tokyo, Japan
[+]University of Potsdam, Am Neuen Palais 10, 14469 Potsdam, Germany
{*raetsch, mika, lemm, klaus*}*@first.gmd.de, manfred@cse.ucsc.edu, onoda@criepi.or.jp*

Abstract

Boosting algorithms like AdaBoost and Arc-GV are iterative strategies to minimize a constrained objective function, equivalent to Barrier algorithms. Based on this new understanding it is shown that convergence of Boosting-type algorithms becomes simpler to prove and we outline directions to develop further Boosting schemes. In particular a new Boosting technique for regression – ε-Boost – is proposed.

1 Introduction

The past years have seen strong interest dedicated to Boosting and ensemble learning algorithms due to their success in practical classification applications (e.g. [13, 26, 28, 43, 2, 11]). Recent research in this field now focuses on the better understanding of these methods and on extensions that are concerned with robustness issues [31, 3, 37, 36, 38] or generalizations of Boosting algorithms to regression [19, 14, 5].

The present work aims to contribute in two respects: (a) we will show an important relation of Boosting to a general class of optimization methods, the so-called barrier optimization – a technique to minimize constrained objective functions [20]. We clarify that Boosting can be seen as a special case of barrier optimization, i.e. as an iterative approximation method to a barrier algorithm that also relates to the Gauss-Southwell method [27] of nonlinear optimization. Furthermore, this understanding allows us to outline possible paths going beyond existing Boosting schemes. For example, convergence theorems from the optimization literature can be applied, simplifying convergence proofs for Boosting type algorithms. We choose a particularly interesting path, giving rise to our second contribution: (b) the definition of a new Boosting algorithm for regression and its convergence proof. Experiments on toy examples follow, that show the proof of concept for our regression algorithm. Finally a brief conclusion is given.

2 Boosting and Convex Programming

In this section we will introduce some terminology and notation conventions. We will mainly consider convex optimization problems for finding hypothesis coefficients that are used for some linear combination of hypotheses from a given finite hypothesis set. Contrarily, in Section 4 we will consider *iterative* algorithms like AdaBoost and Arc-GV.

2.1 Margin, Edge and Linear Programming

We begin by focusing on the following problem. We are given a set of N examples $Z = \{(\mathbf{x}_n, y_n) : 1 \leq n \leq N\} \subseteq \mathcal{X} \times \{\pm 1\}$ and a (finite) set of hypotheses $\mathcal{H} = \{h_j : 1 \leq j \leq J\}$ of the form $\mathcal{X} \to [-1, +1]$.

Our goal is to find a "good" convex combination of the hypotheses, i.e.

$$f_{\boldsymbol{\alpha}}(\mathbf{x}_n) = \sum_{j=1}^{J} \alpha_j h_j(\mathbf{x}_n), \tag{1}$$

where $\boldsymbol{\alpha}$ lies in the J-dimensional probability simplex Γ^J. That is, $\alpha_j \geq 0$ and $\sum_{j=1}^{J} \alpha_j = 1$. If $f_{\boldsymbol{\alpha}}$ is used as a classifier then the classification of instance $\mathbf{x}$ is $\operatorname{sign}(f_{\boldsymbol{\alpha}}(\mathbf{x}))$.[1]

Let us now define a measure of goodness for a single example. The *margin* of an example $(\mathbf{x}_n, y_n)$ with respect to a given weight vector $\boldsymbol{\alpha}$ is defined as $y_n f_{\boldsymbol{\alpha}}(\mathbf{x}_n)$. A positive margin corresponds to a correct classification and the more positive the margin the greater the confidence [39, 40, 49] that the classification is correct. The margin has been frequently used in the context of Support Vector Machines (SVMs) [49, 47] and Boosting (e.g. [39, 40, 36]). However, for the definition of the margin one needs to have a normalization by some norm of $\boldsymbol{\alpha}$, as otherwise one could arbitrarily increase the margin by scaling the weight vector $\boldsymbol{\alpha}$. Different norms are used for SVMs and Boosting. In SVMs the margin is normalized by the ℓ_2-norm of the weight vector in feature space. In Boosting the ℓ_1-norm of the weight vector is used (cf. Footnote 1). Note that for the purpose of classification the normalization of the function $f_{\boldsymbol{\alpha}}$ is immaterial.

For convenience we introduce the matrix $U \in \mathbb{R}^{N \times J}$, where $U_{nj} = y_n h_j(\mathbf{x}_n)$. The n-th example $(\mathbf{x}_n, y_n)$ corresponds to the n-th row and the j-th hypothesis to the j-th column of U. We let U_n denote the n-th row of U. With this notation the margin of the n-th example is $y_n f_{\boldsymbol{\alpha}}(\mathbf{x}_n) = U_n \boldsymbol{\alpha}$. The margin of a function $f_{\boldsymbol{\alpha}}$ is defined as the minimum mar-

[1]Note, that we could use an arbitrary $\boldsymbol{\alpha} \geq \mathbf{0}$, e.g. $\|\boldsymbol{\alpha}\|_1 \neq 1$ and then we would need to normalize the function $f_{\boldsymbol{\alpha}}$. Here, we use the ℓ_1 norm for the normalization.

gin over all N examples, i.e.

$$\rho(\boldsymbol{\alpha}) = \min_{n=1}^{N} U_n \boldsymbol{\alpha}. \tag{2}$$

A reasonable choice [29, 21, 1, 49] for a convex combination is to maximize the minimum margin of the examples, i.e.

$$\text{choose } \boldsymbol{\alpha}^* \in \Gamma^J \text{ such that } \rho(\boldsymbol{\alpha}^*) = \max_{\boldsymbol{\alpha} \in \Gamma^J} \rho(\boldsymbol{\alpha}). \tag{3}$$

Roughly speaking, the larger the margin the better the bounds that can be proven for the generalization error (e.g. [49, 1]). Also SVMs are based on maximizing a minimum margin. They use the ℓ_2-norm to define the margin and the maximum margin hyperplane maximizes the minimum geometric distance of the patterns to the hyperplane. In our case we use the ℓ_1-norm to define the margin. Now the maximum margin hyperplane maximizes the minimum ℓ_∞ distance of the patterns to the hyperplane [30]. We assume for convenience throughout the paper that the hypotheses class is complementation closed ($h \in \mathcal{H}$ implies $-h \in \mathcal{H}$) in order to avoid problems[2] for the case that $\rho(\boldsymbol{\alpha}^*) < 0$.

Boosting algorithms maintain a distribution $\mathbf{d} \in \Gamma^N$ on the examples. What would be a good choice for this distribution for a given set of examples and hypotheses? Assume for a moment that the labels of the hypothesis are binary, i.e. $h_j(\mathbf{x}_n) \in \{\pm 1\}$. Then for a distribution $\mathbf{d}$, the dot product $U_j^\top \mathbf{d} = \sum_{n=1}^N d_n y_n h_j(\mathbf{x}_n)$ is the expectation that h_j predicts the $\{\pm 1\}$ label correctly.[3] We call this the *edge* [8] of the hypothesis h_j. Note that a random hypothesis has an expected edge of zero and since $h_j(\mathbf{x}_n) \in [-1, +1]$, the edge of h_j lies in $[-1, +1]$. We define the *edge of a weight vector* $\mathbf{d}$ as the maximum edge over the set of hypotheses, i.e.

$$\epsilon(\mathbf{d}) = \max_{j=1}^{J} U_j^\top \mathbf{d}. \tag{4}$$

Since $\mathcal{H}$ is complementation closed, the above is equivalent to $\max_{j=1}^J |U_j^\top \mathbf{d}|$. In the case of Boosting, we want to find a distribution on the examples such that the maximum edge of the hypotheses is minimized [8] (a solution always exists):

$$\text{choose } \mathbf{d}^* \in \Gamma^N \text{ such that } \epsilon(\mathbf{d}^*) = \min_{\mathbf{d} \in \Gamma^N} \epsilon(\mathbf{d}). \tag{5}$$

The minimax theorem of linear programming (LP) can be used to make the following connection [8, 21, 17, 3] between the above two optimization problems.

Theorem 1.

$$\min_{\mathbf{d} \in \Gamma^N} \epsilon(\mathbf{d}) = \max_{\boldsymbol{\alpha} \in \Gamma^J} \rho(\boldsymbol{\alpha}). \tag{6}$$

The theorem is proven by considering both sides of (6) as linear programming problems (Here $\mathbf{0}$ and $\mathbf{1}$ are vectors or all zeros and ones, respectively, where the dimension is understood from the context):

$$\begin{array}{lll} & \max_{\rho, \boldsymbol{\alpha}} \ \rho & \min_{\epsilon, \mathbf{d}} \ \epsilon \\ \text{s.t.} & \boldsymbol{\alpha} \geq \mathbf{0}, \boldsymbol{\alpha}^\top \mathbf{1} = 1 & \mathbf{d} \geq \mathbf{0}, \mathbf{d}\mathbf{1} = 1 \\ & U\boldsymbol{\alpha} \geq \rho \mathbf{1} & U^\top \mathbf{d} \leq \epsilon \mathbf{1} \\ & \text{Margin-LP Problem} & \text{Edge-LP Problem} \end{array} \tag{7}$$

[2] Alternatively, one can use two non-negative weights per hypothesis h_j, one for h_j and one for $-h_j$.

[3] Here correct means $+1$ and incorrect -1. If correct is encoded as $+1$ and incorrect as 0, then it would become $\frac{1}{2}(U_j^\top \mathbf{d} + 1)$.

Both problems are dual to each other and thus the equality of the theorem follows from the fact that the primal and the dual objective have the same value. Since our hypothesis class is complementation closed, this value is always non-negative. The Margin-LP Problem was introduced in [29] and was first used for Boosting in [8, 21].

2.2 Boosting and Relative Entropy Minimization

We will now use the Edge-LP problem to make a connection to a class of Boosting algorithms that use a relative entropy in the objective function [22, 25, 9]. In the *Totally Corrective Algorithm* of [22] and in a related algorithm by [9] the edge was forced to be zero for all hypotheses. This essentially corresponds to the Edge-LP Problem with ϵ fixed at 0. Furthermore a relative entropy to the uniform distribution was used as the objective function:

$$\begin{array}{ll} \min_{\mathbf{d}} & \sum_{n=1}^{N} d_n \ln \frac{d_n}{1/N} \\ \text{s.t.} & \mathbf{d} \geq \mathbf{0}, \mathbf{d}^\top \mathbf{1} = 1 \\ & U^\top \mathbf{d} = \mathbf{0} \end{array} \tag{8}$$

Totally Corrective Algorithm

Note, the well-known AdaBoost algorithm [16, 40] can be motivated as minimizing a relative entropy subject to the constraint that the edge of only the last hypothesis is zero [16, 22, 25, 9].

However, how should one choose $\mathbf{d}$ when there is no distribution $\mathbf{d}$ for which the edges of all hypotheses are zero? Theorem 1 implies that if the margin $\rho(\boldsymbol{\alpha}^*) > 0$, then such distribution $\mathbf{d}$ does not exist. In this case the minimal edge is positive. This question can be answered by adding a relative entropy to the objective function of the Edge-LP Problem:

$$\begin{array}{ll} \min_{\epsilon, \mathbf{d}} & \epsilon + \beta \sum_{n=1}^{N} d_n \ln \frac{d_n}{1/N} \\ \text{s.t.} & \mathbf{d} \geq \mathbf{0}, \mathbf{d}^\top \mathbf{1} = 1 \\ & U^\top \mathbf{d} \leq \epsilon \mathbf{1}, \end{array} \tag{9}$$

Note that we introduced a constant parameter β, which controls the trade-off between keeping the edge ϵ versus the relative entropy of $\mathbf{d}$ minimal. For $\beta \to 0$ we recover the Edge-LP Problem. Also note that if $\epsilon = 0$ is enforced in the above problem then we arrive at the optimization problem of the Totally Corrective Algorithm. We believe that above problem with the trade-off parameter β is the natural choice for the case when $\rho(\boldsymbol{\alpha}^*) > 0$.

Before we continue we change to a more convenient variant of the above problem (9). Using the new problem (called the *Edge-Entropy Problem*) will simplify the notation in the sequel of the paper. Note that constraints of both problems are the same, but the objective function of the new problem differs by $\beta \mathbf{d}^\top \mathbf{1}(\ln N + 1)$. Since we have the constraint $\mathbf{d}^\top \mathbf{1} = 1$, it is a constant and both optimization problems are equivalent:

$$\begin{array}{ll} \min_{\epsilon, \mathbf{d}} & \epsilon + \beta \sum_{n=1}^{N} d_n \ln d_n - d_n \\ \text{s.t.} & \mathbf{d} \geq \mathbf{0}, \mathbf{d}^\top \mathbf{1} = 1 \\ & U^\top \mathbf{d} \leq \epsilon \mathbf{1}, \end{array} \tag{10}$$

Edge-Entropy Problem

The dual of the above is

$$\begin{array}{ll}\max\limits_{\rho_\beta,\boldsymbol{\alpha}} & \rho_\beta - \beta \sum\limits_{n=1}^{N} \exp\left(\frac{\rho_\beta - U_n\boldsymbol{\alpha}}{\beta}\right) \\ \text{s.t.} & \boldsymbol{\alpha} \geq \mathbf{0}, \boldsymbol{\alpha}^\top \mathbf{1} = 1 \end{array} \tag{11}$$

Margin-Exp Problem

In Section 4.2 we will present how this problem is related to the Arc-GV algorithm [8]. Let $\rho_\beta(\boldsymbol{\alpha})$ be the solution of (11) for a fixed $\boldsymbol{\alpha}$. We have:

$$\rho_\beta(\boldsymbol{\alpha}) = -\beta \log \left[\sum_{n=1}^{N} \exp\left(-\frac{U_n\boldsymbol{\alpha}}{\beta}\right)\right]. \tag{12}$$

Analyzing the behavior of $\rho_\beta(\boldsymbol{\alpha})$ for $\beta \to 0$ yields:

$$\lim_{\beta\to 0} \rho_\beta(\boldsymbol{\alpha}) = \rho(\boldsymbol{\alpha}), \tag{13}$$

where the convergence in terms of β is linear in the worst case. We can get rid of the variable ρ_β in the Margin-Exp Problem by plugging in the optimal $\rho_\beta(\boldsymbol{\alpha})$ given in (12). This results in an equivalent optimization problem with one less variable to optimize:

$$\begin{array}{ll}\min\limits_{\boldsymbol{\alpha}} & \beta \left[\log \sum\limits_{n=1}^{N} \exp\left(-\frac{U_n\boldsymbol{\alpha}}{\beta}\right) + 1\right] \\ \text{s.t.} & \boldsymbol{\alpha} \geq \mathbf{0}, \boldsymbol{\alpha}^\top \mathbf{1} = 1 \end{array} \tag{14}$$

Note that except for the constraint $\boldsymbol{\alpha}^\top \mathbf{1} = 1$ (and some constants) the above problem is dual to the optimization problem (8) of the Totally Corrective Algorithm. Also AdaBoost can be motivated as optimizing a log of a sum of exponentials, where the constraint $\boldsymbol{\alpha}^\top \mathbf{1} = 1$ is absent.[4] Note also that the solution $\boldsymbol{\alpha}^*_\beta$ of the Margin-Exp Problem converges (for $\beta \to 0$) to a global solution $\boldsymbol{\alpha}^*$ of the Margin-LP Problem.[5]

When the examples are noisy then the solution found by (3) and (11) for $\beta \to 0$ tends to over-fit the data, as all patterns are classified with some non-negative margin [21, 36]. In Section 4.5 we will consider a regularization approach, where the entropy is used as regularization by keeping $\beta > 0$. Then, the parameter β specifies the trade-off between minimizing the edge and keeping the relative entropy small.

Note that the constraints $U\boldsymbol{\alpha} \geq \rho\mathbf{1}$ of the Margin-LP problem are absent from the Margin-Exp problem. However,

$$\lim_{\beta\to 0} -\beta \exp\left(\beta^{-1}(\rho - U_n\boldsymbol{\alpha})\right) = -\infty \text{ iff } U_n\boldsymbol{\alpha} \leq \rho$$

and thus the additional term (which involves a sum of exponentials) in the objective function of the Margin-Exp Problem enforces the constraints $U_n\alpha \geq \rho$ for small enough β. This technique for solving a constraint optimization problem is known as the *exponential barrier method* (e.g. [10]). Also the constraints $\mathbf{d} \geq \mathbf{0}$ can be removed when going from the Edge-LP Problem to the Edge-Entropy problem. Here, the entropy term works as a barrier for the non-negativity constraints. This barrier function is called the *entropic barrier* [6].

Before we show some more connections to Boosting in Section 4, we will introduce the numerical method of barrier optimization in the next section.

3 Barrier optimization

3.1 Problem definition

Let us shortly review some basic statements and formulations about barrier optimization techniques. For details, the reader is referred to e.g. [20, 6, 27]. Consider the convex optimization problem

$$\begin{array}{ll}\min & g(\boldsymbol{\theta}) \\ \text{with} & c_n(\boldsymbol{\theta}) \geq 0, \quad \forall n = 1 \ldots N \end{array} \tag{15}$$

We call $\mathcal{C}$ the convex set of feasible solutions described by

$$\mathcal{C} = \{\boldsymbol{\theta} \,:\, c_n(\boldsymbol{\theta}) \geq 0, \quad \forall\, n = 1, \ldots, N\}. \tag{16}$$

We assume $\mathcal{C}$ is non-empty, i.e. there exists a feasible solution. If the function $g(\boldsymbol{\theta})$ is strictly convex, then the solution of problem (15) is unique. If $g(\boldsymbol{\theta})$ is convex only, there exists a set of global solutions $\boldsymbol{\Theta}^*$. Note that from the convexity of $g(\boldsymbol{\theta})$ and $\mathcal{C}$ follows that any local minimum is a global minimum as well.

3.2 Barrier functions

Problem (15) can be solved by finding a sequence of (unconstraint) minimizers of the so called barrier (or penalty) error function

$$E_\beta(\boldsymbol{\theta}) = g(\boldsymbol{\theta}) + \sum_{n=1}^{N} \kappa_\beta(c_n(\boldsymbol{\theta})), \tag{17}$$

where, κ_β is a barrier function and $\beta > 0$ is a penalty parameter. Common choices for κ_β can be found in Table 1. Barrier algorithms use a suitably chosen sequence of β's that goes to zero. For each β, using the $\boldsymbol{\theta}$ found in the previous iteration as a starting value, (17) is minimized.

For the Log-Barrier and the Entropic Barrier the initial $\boldsymbol{\theta}$ has to be a feasible solutions. The barrier function assures that the sequence of $\boldsymbol{\theta}$'s corresponding to the decreasing sequence of β values remain feasible. Thus the final $\boldsymbol{\theta}^*$ is approached from the "interior" of the feasible region. Methods based on the Log-Barrier and the Entropic Barrier are therefore called interior point methods. Such methods are often used for finding solutions for SVMs [7].

Table 1: Common barrier functions used in convex optimization

$\kappa_\beta(t)$	Name
$-\beta \log t$	Log-Barrier [6, 20, 27]
$-\beta t \log t$	Entropic Barrier [6, 27]
$\beta \exp(-t/\beta)$	Exp-Barrier [10, 24, 12, 33]

In the case of the Exp-Barrier, the initial solution does not have to be feasible [10]. For the Exp-Barrier the minimizers

[4] However, the scaling can easily be done by setting $\beta = 1/(\mathbf{1}^\top \boldsymbol{\alpha}^u)$, where $\boldsymbol{\alpha}^u$ is now the unnormalized version of $\boldsymbol{\alpha}$. This will be worked out in detail in Section 4.2.

[5] For the case that the solution of the Margin-LP Problem is not unique, (14) prefers solutions which have a small relative entropy in the dual domain.

of $E_\beta(\boldsymbol{\theta})$ will become feasible automatically, when β becomes small enough. If a constraint is violated then this will lead to an exponential growth in the barrier objective (17). So the barrier has the effect that it keeps $\boldsymbol{\theta}$ in the feasible region or pulls it closer to a feasible solution. Note that Entropic Barrier is the dual of the Exp-Barrier (cf. Section 2.2).

The Exp-Barrier can also be used to solve the feasibility problem for convex programming:

$$\begin{array}{ll} \text{find} & \boldsymbol{\theta} \\ \text{with} & c_n(\boldsymbol{\theta}) \geq 0 \quad \forall n = 1 \ldots N \end{array} \tag{18}$$

In fact, we will see in Section 4.3 that AdaBoost exploits that property. Solving the feasibility problem with interior point methods is more involved [48]. In the rest of the paper we concentrate on the Exp-Barrier only and focus on its connection to Boosting methods. Most of the work in Section 5 can be extended to other barrier functions.

3.3 Convergence

Besides an intuitive reasoning for the barrier function converging to an optimal solution of (15) the following presents some more formal aspects. Let us define

$$\boldsymbol{\theta}_\beta := \operatorname*{argmin}_{\boldsymbol{\theta}} E_\beta(\boldsymbol{\theta}) \tag{19}$$

as the optimal solution for some fixed β. Moreover, let $\boldsymbol{\Theta}^*$ be the set of global solutions of (15). Then for any barrier function and any sequence $\beta_t \to 0$ holds:

$$\lim_{t\to\infty} \boldsymbol{\theta}_{\beta_t} \in \boldsymbol{\Theta}^* \tag{20}$$

However, for the Exp-Barrier[6] it it turns out to be unnecessary to *exactly* minimize E_{β_t} for each β_t. The following proposition shows how close an estimate $\boldsymbol{\theta}^t$ to $\boldsymbol{\theta}_{\beta_t}$ has to be, in order to obtain the desired convergence:

Proposition 2 (along the lines of [10]). *Assume g and c_n are differentiable and convex functions. Let $\boldsymbol{\theta}^t$ be an $\delta_t \geq 0$ minimizer of*

$$E_{\beta_t}(\boldsymbol{\theta}) = g(\boldsymbol{\theta}) + \beta_t \sum_{n=1}^{N} \exp(-c_n(\boldsymbol{\theta})/\beta_t), \tag{21}$$

i.e. $\|\nabla_{\boldsymbol{\theta}} E_{\beta_t}(\boldsymbol{\theta}^t)\| \leq \delta_t$. Then, for $\delta_t, \beta_t \overset{t\to\infty}{\longrightarrow} 0$ every limit point of $\{\boldsymbol{\theta}^t\}_{t\in\mathbb{N}}$ is a global solution to (15)*.*

In the sequel we will often use a simpler version of Proposition 2, where we use $\delta_t = \beta_t$ and require $\|\nabla_{\boldsymbol{\theta}} E_{\beta_t}(\boldsymbol{\theta}^t)\| \leq \beta_t$.

4 Boosting as Barrier Algorithms

In Section 2 we have considered convex optimization problems for finding the hypothesis coefficients $\boldsymbol{\alpha}$ for a given finite hypothesis set $\mathcal{H}$. Now, we would like to consider iterative algorithms that have to solve two problems: In each iteration one has to select a hypothesis from $\mathcal{H}$ and then one assigns a weight to the selected hypothesis. We will show how AdaBoost [16] and Arc-GV [8] can be understood as particular implementations of a barrier optimization approach, asymptotically solving convex optimization problems of the type stated in Section 2. It is shown, that Boosting is basically a Gauss-Southwell method minimizing a barrier function.

Throughout the paper, we will think of dealing in each iteration t of Boosting with a full hypothesis weight vector $\boldsymbol{\alpha}^t$ of size J. However, usually we will change only one entry j at a time. We denote by I_t the index of the hypothesis that has been chosen in the t-th iteration and by α_{I_t} the hypothesis coefficient that has been updated. Thus, I is a mapping from the hypotheses that have been chosen so far to $\mathcal{H}$, i.e. $I : \mathbb{N} \to \mathcal{H}$.

4.1 The underlying Optimization Problem

Let $f_{\boldsymbol{\alpha}}$ be a convex combination (cf. Section 2) of hypotheses $h \in \mathcal{H}$ as defined in Section 2

$$f_{\boldsymbol{\alpha}}(\mathbf{x}) := \sum_{j=1}^{J} \frac{\alpha_j}{\|\boldsymbol{\alpha}\|_1} h_j(\mathbf{x}), \tag{22}$$

where we enforce the hypothesis coefficients to sum to one by dividing by $\|\boldsymbol{\alpha}\|_1$. For simplicity, we will consider the unnormalized version of $\boldsymbol{\alpha}$ throughout the rest of the paper and normalize, if it is needed.

Suppose one would like to solve the Margin-LP Problem (7), where we can omit the $\boldsymbol{\alpha}^\top \mathbf{1} = 1$ constraint due to of the normalization introduced in (22):

$$\begin{array}{ll} \max & \rho \\ \text{with} & y_n f_{\boldsymbol{\alpha}}(\mathbf{x}_n) \geq \rho, \\ & \boldsymbol{\alpha} \geq 0 \end{array} \tag{23}$$

Note that the constraint $\alpha_j \geq 0$ can always be enforced by selecting the h_j with the appropriate sign (we have assumed complementation closedness of $\mathcal{H}$). For convenience, we will in the sequel also often write $\boldsymbol{\alpha}^\top \mathbf{1}$ instead of $\|\boldsymbol{\alpha}\|_1$. Using (17) and the definition of U_{nt} we minimize

$$E_\beta(\boldsymbol{\alpha}, \rho) = -\rho + \beta \sum_{n=1}^{N} \exp\left(\frac{\rho\|\boldsymbol{\alpha}\|_1 - U_n\boldsymbol{\alpha}}{\beta\|\boldsymbol{\alpha}\|_1}\right) \tag{24}$$

with respect to ρ and $\boldsymbol{\alpha}$ with $\boldsymbol{\alpha} \geq \mathbf{0}$. Let

$$[\boldsymbol{\alpha}_\beta, \rho_\beta] \quad = \quad \operatorname*{argmin}_{\boldsymbol{\alpha}\geq 0, \rho} E_\beta(\boldsymbol{\alpha}, \rho) \tag{25}$$

and let $[\boldsymbol{\alpha}^*, \rho^*]$ be a global solution to (23). Note that $\rho_\beta = \rho_\beta(\boldsymbol{\alpha})$ introduced in Section 2.2. Using (20) we conclude, that

$$\lim_{\beta\to 0} [\boldsymbol{\alpha}_\beta, \rho_\beta] = [\boldsymbol{\alpha}^*, \rho^*] \tag{26}$$

4.2 Arc-GV – A Greedy Algorithm

Let us now consider one particular iterative strategy for minimizing (24) which will turn out to be exactly Arc-GV: We start with $\boldsymbol{\alpha}^0 = \mathbf{0}$ and $f_{\boldsymbol{\alpha}^0} \equiv 0$. In each iteration t we (i) approximate ρ_β and (ii) find a hypothesis $h_j \in \mathcal{H}$ (i.e. an index j) and update α_j to get the new hypothesis weight vector $\boldsymbol{\alpha}^t$.

Now the details: First we set $\beta_t \equiv \|\boldsymbol{\alpha}^t\|^{-1}$ which is a reasonable choice, as (assuming complementation closed hypothesis sets) for AdaBoost and Arc-GV $\|\boldsymbol{\alpha}\| \to \infty$ holds. In step (i) one approximates the minimum margin ρ_β by (cf. Eq. (13))

$$\overline{\rho} = \rho(\boldsymbol{\alpha}^{t-1}) = \min_n \; y_n f_{\boldsymbol{\alpha}^{t-1}}(\mathbf{x}_n).$$

[6] For other barrier functions there exist similar results.

In step (ii) one updates the hypothesis weighting α_{I_t} by

$$\overline{\alpha}_{I_t} = \operatorname*{argmin}_{\alpha_{I_t}>0} \sum_{n=1}^{N} \exp\left(\overline{\rho}\|\boldsymbol{\alpha}\|_1 - U_n\boldsymbol{\alpha}\right), \tag{27}$$

where one chooses the index I_t such that the sum in (27) is minimal. That is, one finds the hypothesis such that some weighted training error ϵ is minimized [16]. Note that for the case where we use $\{\pm 1\}$-valued weak learners one can compute a closed form solution [8] for (27):

$$\overline{\alpha}_{I_t} = \frac{1}{2}\log\left(\frac{1-\epsilon}{\epsilon}\frac{\overline{\rho}}{1-\overline{\rho}}\right) \tag{28}$$

Moreover, note that $\overline{\rho}$ is the same as `top` used in [8].

Proposition 2 serves as a useful tool for proving the convergence of such kind of algorithms: From the fast convergence of $\overline{\rho}$ to ρ_β we can conclude that $\nabla_\rho E_\beta(\boldsymbol{\alpha}, \overline{\rho})$ vanishes fast. Consider the update for the hypothesis weight $\overline{\alpha}_{I_t}$ given by (27) and (28), respectively. Under rather mild assumptions, one can show that in the limit for $\|\boldsymbol{\alpha}\|_1 \to \infty$ holds:

$$\operatorname*{argmin}_{\alpha_{I_t}\geq 0} E_\beta(\boldsymbol{\alpha}, \overline{\rho}) = \overline{\alpha}_{I_t}, \tag{29}$$

where $\boldsymbol{\alpha}$ is the vector of the last iteration $\boldsymbol{\alpha}^{t-1}$ changed in the I_t-th component only. This is because the factor $\beta = \|\boldsymbol{\alpha}\|_1^{-1}$ in front of the sum in (24) loses its influence on the minimization of α_{I_t} as $\|\boldsymbol{\alpha}\|_1^{t-1}$ becomes large.

However, (29) does not imply that the gradient with respect to $\boldsymbol{\alpha}$ becomes $\mathbf{0}$ yet, as needed to apply Proposition 2. It needs some more argumentation to show that

$$\nabla_{\boldsymbol{\alpha}} E_\beta(\overline{\boldsymbol{\alpha}}, \overline{\rho}) \to \mathbf{0}, \tag{30}$$

where one exploits that always the best hypothesis h_{I_t} is found. Essentially, a proof can be done in the style of Theorem 4 in Section 5.5, where one uses the convergence properties of the Gauss-Southwell method (gradient-descent in coordinate directions with maximal gradient) [27]. One has to show, that $\|\boldsymbol{\alpha}\|_1$ grows slow enough while the gradients become smaller. Similar techniques have been used in [15, 14] and (30) has already been shown in [8], so we will not go into further details. Using the property (30), one can finally apply Proposition 2 which shows the convergence.

Based on the argumentation above, we would like to view Arc-GV as a barrier algorithm using a particular strategy (related to the Gauss-Southwell method) for minimizing E_β and choosing β.

4.3 AdaBoost – Finding a Separation

We can argue as before and find that AdaBoost can be interpreted as a barrier algorithm for finding a separation of some training set with margin at least φ. For this, we fix ρ to φ and get the barrier function from Eq. (24)) as

$$E_\beta(\boldsymbol{\alpha}) = -\varphi + \beta\sum_{n=1}^{N}\exp\left(\frac{\|\boldsymbol{\alpha}\|_1\varphi - U_n\boldsymbol{\alpha}}{\beta\|\boldsymbol{\alpha}\|_1}\right). \tag{31}$$

Originally, the β in front of the sum in (31) down-weights the constraint penalties against the objective. But in the case of AdaBoost and Arcing we have $\varphi \equiv 0$ and $\varphi \equiv$ const., respectively, and our goal is to simply obtain a feasible solution ($y_n f_{\boldsymbol{\alpha}}(\mathbf{x}_n) \geq \varphi$, cf. Eq. (18)). Thus, we can omit the β here and drop the φ in front. What remains is only the sum of (31). By using the same simplified optimization strategy (coordinate–wise descent) and setting $\beta \equiv \|\boldsymbol{\alpha}\|_1^{-1}$ in (31) as before, we obtain the original AdaBoost error function [8, 18, 31, 36]. Thus, we will get a solution with margin at least φ, if $\|\boldsymbol{\alpha}\|_1 \to \infty$. Note that $\|\boldsymbol{\alpha}\|_1$ will stay bounded if and only if there is no solution [16, 36].

4.4 Iterative Boosting Algorithms that always update all previous coefficients

There are two basic approaches for designing boosting algorithms. The first one is to update the coefficient of a single (possibly new) hypothesis in each iteration. In this approach we want to do little work per iteration. Typical examples are AdaBoost and Arc-GV. A second approach is to ignore optimization issues and use the optimization problems considered in Section 2.1 to always find the optimal weights of all past hypotheses. The Totally Corrective Algorithm of [22] and the Column Generation Algorithm of [3] are particular examples using this approach.[7]

Assume at trial t we already have a subset $\mathcal{H}^{t-1}$ of $t-1$ hypotheses from the base set $\mathcal{H}$. We also have a vector $\boldsymbol{\alpha}^{t-1}$ of $t-1$ weights for combining the hypotheses of $\mathcal{H}^{t-1}$ in some optimal way. This vector was found by solving some convex optimization problem (e.g. the Margin-Exp Problem) when applied to the set $\mathcal{H}^{t-1}$. Note that via the dual relationships we always have a corresponding distribution $\mathbf{d}^{t-1}$ on the examples. During trial t we add one more hypothesis h_t from the base set $\mathcal{H}$ to $\mathcal{H}^{t-1}$ to form $\mathcal{H}^t$ and then find t new weights $\boldsymbol{\alpha}^t$ *for all* hypotheses of $\mathcal{H}^t$.

We have not specified how to choose the new hypothesis h_t at trial t. A natural greedy heuristic is choose a hypothesis such that the value of optimization problem for $\mathcal{H}_{t-1} \cup \{h_t\}$ is optimized or approximately optimized.

Basically, any optimization algorithm for finding a linear combination for a fixed set of hypotheses (or by the duality relationship a distribution based on these hypotheses) immediately leads to a boosting algorithm via the above scheme.

4.5 A regularized Version of Boosting

The question is which optimization problem should we use to construct a boosting algorithm (via the scheme of the previous subsection) in the case when the examples is noisy. Before we address this we give some background. It has been shown that solutions found by AdaBoost, Arc-GV and also for the Margin-Exp problem for $\beta \to 0$, tend to over-fit the data, as all patterns are classified with some non-negative margin [21, 36]. Several approaches have been proposed to deal with this situation. For SVMs slack variables have been frequently used [4, 49] to allow for some *soft margin*, i.e. violations of the margin constraints similar to (7). In the dual domain the introduction of slack variables leads to an ℓ_∞-norm constraint on the dual variables. Therefore, the (dual) variables $\mathbf{d}$ of the Edge-LP are restricted to intersection of the probability simplex Γ^N and some hypercube only, where the size of the hypercube is controlled by some regularization constant. Basically, the distribution $\mathbf{d}$ is kept near to the

[7] The boosting algorithms of [9] update the weights of all past hypotheses in parallel in an attempt to find a close approximation of the optimal weights of all the past hypotheses.

center of the simplex Γ^N. The same idea has been proposed for Boosting in [37].

In view of this discussion, it is natural to keep the β parameter in the Edge-Entropy problem (10) fixed or lower bounded. Now the relative entropy term does not vanish and is traded off with the maximum edge ϵ. The parameter β should be tuned via cross-validation. The more noise in the data, the larger β should be. Large β give the relative entropy to the uniform distribution high importance and the solution is kept close to the center of the simplex Γ^N (as in SVMs). It can be shown that this regularization approach is equivalent to the PBVM algorithm proposed in [44].

Note that the trade-off between a relative entropy and some loss has long been used to derive on-line learning algorithms (see the derivation of the Exponentiated Gradient Algorithm in [23]).

5 Boosting for Regression

So far we were mainly concerned with understanding Boosting algorithms for classification from the barrier optimization point of view and pointed out potential extensions. In this section we will now consider one natural generalization in depth: Boosting for regression.

5.1 Problem definition

Let $\mathcal{H}$ be a finite class of base hypotheses $\mathcal{H} := \{h_j : \mathcal{X} \to \mathbb{R} : j = 1, \dots, J\}$. The regression problem is to find some function[8] $f_{\boldsymbol{\alpha}} \in \operatorname{lin}(\mathcal{H}), f_{\boldsymbol{\alpha}} : \mathcal{X} \to \mathbb{R}$:

$$f_{\boldsymbol{\alpha}}(\mathbf{x}) = \sum_{j=1}^{J} \alpha_j h_j(\mathbf{x}) \quad \text{with} \quad \boldsymbol{\alpha} \in \mathbb{R}^J, \mathbf{x} \in \mathcal{X}, \tag{32}$$

based on iid (training) data $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N) \in \mathcal{X} \times \mathbb{R}$. The goal of the learning process is to find a function f with a small risk $R[f] = \int_{\mathcal{X}\times\mathbb{R}} l(y - f(\mathbf{x}))\, dP(\mathbf{x}, y)$, where P is the probability measure which is assumed to be responsible for the generation of the observed data, and l is a loss function, e.g. $l(y - f(\mathbf{x})) = (y - f(\mathbf{x}))^2$, depending on the specific regression estimation problem at hand. Since we do not know the probability density P and our hypothesis class might be large, one has to take care that the function f does not overfit the data. Thus, one has to introduce *capacity control*, i.e. one has to bound the *complexity* of f. One way to obtain a small(er) risk is to minimize a regularized risk functional

$$R_{\text{reg}}[f] := C\mathbf{P}[f] + R_{\text{emp}}[f] \tag{33}$$

where $R_{\text{emp}}[f] := \frac{1}{N}\sum_{n=1}^{N} l(y_n - f(\mathbf{x}_n))$ is the empirical risk, $\mathbf{P}$ is a regularizer penalizing (and therefore limiting) the model complexity. The parameter C defines the trade-off between complexity and empirical risk. Under rather mild assumptions it can be shown [46] that there exists a constant C' such that minimizing (33) produces the same function as solving the problem

$$\min_{\mathbf{P}[f] \le C'} R_{\text{emp}}[f]. \tag{34}$$

This is shown by the following lemma:

[8] In some cases it might be useful also to have a bias b. This extension is made in Appendix A.1.

Lemma 3 (Along the lines of [46]). *Let $l(y - f_{\boldsymbol{\alpha}}(\mathbf{x}))$ and $\mathbf{P}[f_{\boldsymbol{\alpha}}]$ be convex in $\boldsymbol{\alpha}$. If $\mathbf{P}$ defines a structure of nested subsets of $\operatorname{lin}(\mathcal{H})$, then for any sample $Z \subset \mathcal{X} \times \mathbb{R}$ there exists a monotonically decreasing function $C'(C)$, such that the problems* (33) *and* (34) *using C and $C'(C)$, respectively, have the same solution sets.*

5.2 The Convex Program

Let us assume that l and $\mathbf{P}$ are chosen such that $R_{\text{reg}}[f_{\boldsymbol{\alpha}}]$ is convex in $\boldsymbol{\alpha}$. Then we can solve the following convex program for minimizing $R_{\text{reg}}[f_{\boldsymbol{\alpha}}]$ with respect to $\boldsymbol{\alpha}$:

$$\begin{aligned} \min \quad & C\mathbf{P}[f_{\boldsymbol{\alpha}}] + \tfrac{1}{N}\sum_{n=1}^{N} l(\delta_n) \\ \text{with} \quad & \delta_n = y_n - f_{\boldsymbol{\alpha}}(\mathbf{x}_n) \\ & n = 1, \dots, N, \boldsymbol{\delta} \in \mathbb{R}^N, \boldsymbol{\alpha} \in \mathbb{R}^J \end{aligned} \tag{35}$$

Consider the following optimization problem which is equivalent to (35) in the spirit of Lemma 3.

$$\begin{aligned} \min \quad & \tfrac{1}{N}\sum_{n=1}^{N} l(\delta_n) \\ \text{with} \quad & \mathbf{P}[f_{\boldsymbol{\alpha}}] \le C' \\ & \delta_n = y_n - f_{\boldsymbol{\alpha}}(\mathbf{x}_n) \\ & n = 1, \dots, N, \boldsymbol{\delta} \in \mathbb{R}^N, \boldsymbol{\alpha} \in \mathbb{R}^J \end{aligned} \tag{36}$$

5.3 A Barrier Algorithm

Before we derive the barrier objective, we substitute δ_n by two non-negative variables $\xi_n, \xi_n^* \ge 0$:

$$\delta_n = \xi_n - \xi_n^*.$$

Then each equality constraint in (35) can be replace by two inequality constraints:

$$\begin{aligned} y_n - \sum_{j=1}^{J} \alpha_j h_j(\mathbf{x}_n) &\le \xi_n \\ -y_n + \sum_{j=1}^{J} \alpha_j h_j(\mathbf{x}_n) &\le \xi_n^* \end{aligned}$$

Thus, the barrier minimization objective for the problem (35) using the exponential penalty can be written as:

$$\begin{aligned} E_\beta(\boldsymbol{\alpha}, \boldsymbol{\xi}, \boldsymbol{\xi}^*) := C\mathbf{P}[f_{\boldsymbol{\alpha}}] + \frac{1}{N}\sum l(\xi_n - \xi_n^*) + \\ + \beta \sum_{n=1}^{N} \left(\mathrm{e}^{-\xi_n/\beta} + \mathrm{e}^{-\xi_n^*/\beta} \right) + \\ + \beta \sum_{n=1}^{N} \left(\mathrm{e}^{(\delta_n - \xi_n)/\beta} + \mathrm{e}^{(-\delta_n - \xi_n^*)/\beta} \right) \end{aligned} \tag{37}$$

where $\delta_n := y_n - \sum_{j=1}^{J} \alpha_j h_j(x_n)$. To show how to minimize (37) using a Boosting-type algorithm, we need to specify $\mathbf{P}$ and l. For convenience we use the L_1-norm of the hypothesis weight vector $\boldsymbol{\alpha}$ as a measure for the complexity of $f_{\boldsymbol{\alpha}}$ [45]

$$\mathbf{P}_1[f_{\boldsymbol{\alpha}}] := \|\boldsymbol{\alpha}\|_1. \tag{38}$$

Note that it fulfills the assumptions made in Lemma 3. Furthermore, as frequently and successfully used for regression with SVMs [49, 41], we consider the ε-insensitive loss:

$$l_\varepsilon(f(\mathbf{x}) - y) := \max(0, \|f(\mathbf{x}) - y_n\|_1 - \varepsilon). \tag{39}$$

The ε-insensitive loss has appealing properties, as it will usually lead to sparse solutions of (35) and the ν-trick [42, 45] can be applied to even optimize ε automatically (cf. Appendix A.2). Note that most of the following derivations for this particular choice of $\mathbf{P}$ and l generalize easily to other regularizers and cost functions (e.g. for some other norm of $\boldsymbol{\alpha}$ and for the squared loss using the log-barrier function).
Plugging in our definitions of $\mathbf{P}[\cdot]$ and $l(\cdot)$ to (37) using Lemma 3 yields:

$$E_\beta(\boldsymbol{\alpha}, \boldsymbol{\xi}, \boldsymbol{\xi}^*) := \frac{1}{N}\|\boldsymbol{\xi} - \boldsymbol{\xi}^*\|_1 + \\ +\beta \sum_{n=1}^{N} \left(\mathrm{e}^{-\xi_n/\beta} + \mathrm{e}^{-\xi_n^*/\beta}\right) + \\ +\beta \sum_{n=1}^{N} \left(\mathrm{e}^{(\delta_n - \varepsilon - \xi_n)/\beta} + \mathrm{e}^{(-\delta_n - \varepsilon - \xi_n^*)/\beta}\right) \quad (40)$$

with $\|\boldsymbol{\alpha}\|_1 \leq C'$. Now we can find the optimal slack variables by minimizing (40) for given β and $\boldsymbol{\alpha}$ by setting $\nabla_{\boldsymbol{\xi}} E_\beta = \mathbf{0}$ and solving for $\boldsymbol{\xi}, \boldsymbol{\xi}^*$. We get:

$$\xi_n(\beta) = \beta \log(1 + \mathrm{e}^{-(\varepsilon - \delta_n)/\beta}) \quad (41)$$

$$\xi_n^*(\beta) = \beta \log(1 + \mathrm{e}^{-(\varepsilon + \delta_n)/\beta}). \quad (42)$$

As expected, $\lim_{\beta \to 0} \xi_n(\beta) = \max(0, \varepsilon - \delta_n)$ and $\lim_{\beta \to 0} \xi_n^*(\beta) = \max(0, \varepsilon + \delta_n)$ hold.

5.4 How to Optimize in Practice

Usually, in a barrier algorithm one would optimize all $n+2N$ parameters directly (up to a certain precision) for some β and then decrease β until the desired precision is reached. But then we would need to know all hypothesis in $\mathcal{H}$ in advance in order to optimize their weights (like in SVMs). Thus, we consider a Boosting-type algorithm that finds one new hypothesis h_{I_t} and its weight α_{I_t} in each iteration. Then there is only one parameter, $\alpha_{I_t}(\beta)$, to be determined in each iteration.[9]
In the described setting we can fulfill the constraint $\|\boldsymbol{\alpha}\|_1 \leq C'$ only by stopping the algorithm when the constraint is violated, because the weights of the previous iterations are already fixed. One way to keep going is to redefine $\boldsymbol{\alpha}$:

$$\alpha_j := \gamma_j \min(1, C'\|\boldsymbol{\gamma}\|_1^{-1}) \quad (43)$$

and to optimize in terms of $\boldsymbol{\gamma}$. Then the constraint is always fulfilled and is active, if and only if $\|\boldsymbol{\gamma}\|_1 \geq C'$.[10]

5.5 Convergence

The critical point in proving the convergence of the algorithm is how the base learner selects the next hypothesis, i.e. which index I_t must be chosen, such that E_β is reduced reasonably – without knowing the whole hypothesis class $\mathcal{H}$ in advance. The idea is as follows: The gradient of E_β with respect to each α_j can be computed as

$$\nabla_{\alpha_j} E_\beta(\boldsymbol{\alpha}, \boldsymbol{\xi}, \boldsymbol{\xi}^*) = \sum_{n=1}^{N} w_n h_j(\mathbf{x}_n),$$

where $w_n = \mathrm{e}^{(\delta_n - \boldsymbol{\xi}_n - \varepsilon)/\beta} - \mathrm{e}^{(-\delta_n - \boldsymbol{\xi}_n^* - \varepsilon)/\beta}$. To reduce E_β iteratively (for some fixed β) one may choose the hypothesis h_{I_t} such that

$$I_t = \operatorname*{argmax}_{j=1,\ldots,J} \left|\nabla_{\alpha_j} E_\beta\right|. \quad (44)$$

This corresponds to a coordinate descent method, the so-called Gauss-Southwell method, which finally converges [27] to the global solution of E_β – for some fixed β. This choice of I_t ensures, that the L_∞-norm of $\nabla_{\boldsymbol{\alpha}} E_\beta$, and therefore any norm of $\nabla_{\boldsymbol{\alpha}} E_\beta$ (if $\mathcal{H}$ is finite), will converge to zero for some fixed β, because of the convergence properties of the Gauss-Southwell method.

Theorem 4. *Assume $\mathcal{H}$ is finite. Let the base learner be*

$$L(X, \mathbf{w}) := \operatorname*{argmax}_{h \in \mathcal{H}} |h(X)^\top \mathbf{w}|. \quad (45)$$

Algorithm 1 The ε-Boost algorithm

argument: Sample $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}, \mathbf{y} = \{y_1, \ldots, y_N\}$
Number of iterations T
Regularization constant C'
Tube size $\varepsilon \geq 0$
constants: $\beta_{start} \in (0,1)$, $p > 1$.
returns: Linear combination from $\mathcal{H}$.
function ε-Boost$(X, \mathbf{y}, T, C', \varepsilon)$
  Set $\beta = \beta_{start}$
  for $n = 1, \ldots, N$ **do**
    $w_n = \exp((-y_n - \varepsilon)/\beta) - \exp((y_n - \varepsilon)/\beta)$
  endfor
  for $t = 1, \ldots, T$
    $h_t := L(X, \mathbf{w})$
    $\gamma_t := \operatorname*{argmax}_{\gamma_t, b \in \mathbb{R}} E_\beta(\boldsymbol{\gamma}^t)$
    Compute $\boldsymbol{\alpha}$ by (43)
    Set $\delta_n := y_n - \sum_q^j \alpha_q h_q(x_n)$
    for $n = 1, \ldots, N$ **do**
      $w_n := \mathrm{e}^{(\delta_n - \boldsymbol{\xi}_n - \varepsilon)/\beta} - \mathrm{e}^{(-\delta_n - \boldsymbol{\xi}_n^* - \varepsilon)/\beta}$
    endfor
* **if** $|h_t(X)^\top \mathbf{w}| < \beta$, **do**
      $\beta := \beta^p$
    endif
  endfor
  return $f = b + \sum_{t=1}^{J} \alpha_t h_t$
end
function $E_\beta(\boldsymbol{\gamma})$
  Compute $\boldsymbol{\alpha}$ by (43)
  Set $\delta_n := y_n - \sum_{q=1}^{t} \alpha_q h_q(x_n)$
  Compute $\boldsymbol{\xi}, \boldsymbol{\xi}^*$ by (41) and (42)
  Set $\mathbf{r} := [\mathbf{d}^\top + \boldsymbol{\xi}^\top + \varepsilon\mathbf{1}, -\mathbf{d}^\top + \boldsymbol{\xi}^{*\top} + \varepsilon\mathbf{1}, \boldsymbol{\xi}^\top, \boldsymbol{\xi}^{*\top}]$
  return $\frac{1}{N}\|\boldsymbol{\xi} - \boldsymbol{\xi}^*\|_1 + \beta \sum_q \exp(-r_q/\beta)$
end

[9] To minimize (37) with respect to α_{I_t} for a given β, one may employ standard techniques like Brent's line search method which work quite fast.

[10] There are also other ways, like introducing a scaling variable, to deal with this problem.

Suppose we run ε-Boost (see pseudocode) with $C' > 0$ as regularization constant and $\varepsilon \geq 0$ as tube size. Then for $T \to \infty$ the output of the algorithm converges to a global solution of (36) *using $l = l_\varepsilon$ and $\mathbf{P}[f_{\boldsymbol{\alpha}}] = \|\boldsymbol{\alpha}\|_1$.*

Proof. Assume we would fix $\beta > 0$, then the sequence of $\boldsymbol{\alpha}^t$ generated in each iteration t converges to the global minimum of E_β exploiting the convexity of E_β and the convergence properties of the Gauss-Southwell method. Thus, after a finite number of steps we have

$$J^{-1/2}\|\nabla_{\boldsymbol{\alpha}} E_\beta(\boldsymbol{\alpha})\|_2 \leq \beta.$$

Moreover, let h_{I_t} be the hypothesis found in the t-th iteration by the base learner $L(X, \mathbf{w}^t)$. Then

$$\begin{aligned} J^{-1/2}\|\nabla_{\boldsymbol{\alpha}} E_\beta(\boldsymbol{\alpha}^t)\|_2 &\leq \|\nabla_{\boldsymbol{\alpha}} E_\beta(\boldsymbol{\alpha}^t)\|_\infty \\ &= h(X)^\top \mathbf{w}^t. \end{aligned}$$

Thus, line $*$ in Algorithm 1 ensures that we decrease β only if $\|\nabla_{\boldsymbol{\alpha}} E_\beta(\boldsymbol{\alpha}^t)\|_2 \leq \beta_t\sqrt{J}$. Hence, the algorithm generates sequences of $\boldsymbol{\alpha}^t$ and $(\beta_t, \beta_t\sqrt{J})$ fulfilling the conditions in Proposition 2. This proves the theorem. □

Let us analyze (45) as a particular way of finding the next hypothesis: Assume $\|h_j(X)\|_2 = \text{const}$ for all hypothesis. Then selecting the hypothesis which minimizes the mean squared error (MSE)

$$h = \operatorname*{argmin}_{h \in \mathcal{H}} \frac{1}{N} \sum_{n=1}^{N} (w_n - h(\mathbf{x}_n))^2,$$

will result in the same hypothesis as in (44) and (45).

The definition of the base learner (45) seems to be very restricting. The next corollary uses a weaker condition on L and is a direct implication of Theorem 4:

Corollary 5. *Assume the conditions as in Theorem 4 and a base learner L which for some weighting $\mathbf{w}$ always returns a hypothesis h, such that for some $K < \infty$*

$$\max_{h \in \mathcal{H}} |h(X)^\top \mathbf{w}| < K |h(X)^\top \mathbf{w}|. \tag{46}$$

Suppose we run ε-Boost. Then for $T \to \infty$ the output of the algorithm converges to a global solution of (34)*.*

Proof. Using the same argumentation as in the proof of Theorem 4, after a finite number of steps t we have a gradient with respect to a sub-vector $\overline{\boldsymbol{\alpha}}$ of $\boldsymbol{\alpha}$ satisfying:

$$J^{-1/2}\|\nabla_{\overline{\boldsymbol{\alpha}}} E_\beta(\boldsymbol{\alpha}^t)\|_2 \leq \beta.$$

Moreover, let h_{I_t} be the hypothesis found in the t-th iteration by a base learner $L(X, \mathbf{w}^t)$ fulfilling condition (46). Suppose there would exist a hypothesis h_j such that

$$|\nabla_{\alpha_j} E_\beta(\boldsymbol{\alpha})| > K\beta.$$

Then the base learner is enforced to return h_j before β is decreased. Thus, the coordinate descent iterations will focus on directions with large gradients. Thus after a finite number of iterations the gradient is reduced for such hypothesis as well, until

$$\|\nabla_{\boldsymbol{\alpha}} E_\beta(\boldsymbol{\alpha}^t)\|_\infty \leq K\|\nabla_{\overline{\boldsymbol{\alpha}}} E_\beta(\boldsymbol{\alpha}^t)\|_\infty$$

Hence,

$$\begin{aligned} J^{-1/2}\|\nabla_{\boldsymbol{\alpha}} E_\beta(\boldsymbol{\alpha}^t)\|_2 &\leq \|\nabla_{\boldsymbol{\alpha}} E_\beta(\boldsymbol{\alpha}^t)\|_\infty \\ &\leq K h(X)^\top \mathbf{w}^t. \end{aligned}$$

Thus, line $*$ in Algorithm 1 ensures that we decrease β only if $\|\nabla_{\boldsymbol{\alpha}} E_\beta(\boldsymbol{\alpha}^t)\|_2 \leq K\beta_t\sqrt{J}$. Hence, the algorithm generates sequences of $\boldsymbol{\alpha}^t$ and $(\beta_t, K\beta_t\sqrt{J})$ fulfilling the conditions in Proposition 2. This proves the theorem. □

NB: One is now looking for a hypothesis which is not too bad compared to the best hypothesis in $\mathcal{H}$. If $K = \infty$, then one would allow that the base learner never returns some of the needed hypotheses. Thus, we could remove them from $\mathcal{H}$ and would get $K < \infty$.

5.6 An Experiment on toy data

To illustrate (i) that the proposed regression algorithm converges to the optimal (i.e. zero error) solution and (ii) is capable of finding a good fit to noisy data we applied it to a toy example whose results are shown in Figure 1. For simplicity we used radial basis function kernels as base hypothesis, i.e. $\mathcal{H} = \{h_n(\mathbf{x}) = \exp(-2\|\mathbf{x} - \mathbf{x}_n\|^2) \mid n = 1, \ldots, N\}$.

6 Conclusion

Barrier optimization is a general framework for minimizing a constrained objective function. We have proposed to understand AdaBoost or Arc-GV as special iterative strategies for barrier optimization. This new view suggests that we can use the tool kit of optimization theory for barrier methods quite convenient in the context of Boosting and it allows e.g. simpler convergence proofs and more general Boosting algorithms for classification and regression. The proposed new ε-Boost algorithm exemplifies this general claim, as it defines a very natural Boosting scheme for regression.
So far the strength of our contribution is to be seen on the theoretical and conceptual side. On the practical side this paper has only shown toy examples, to give the proof of concept. Large scale simulation studies need to follow. Future theoretical research will be dedicated to further exploiting the link between barrier methods and Boosting, in order to obtain extensions of Boosting algorithms that are eventually faster, better, more general and easier to understand.

Acknowledgments: We thank Alex Smola, Bob Williamson and Bernhard Schölkopf for valuable discussions. This work was partially funded by DFG under contract JA 379/91, JA 379/71 and by EU in the NeuroColt2 project. Manfred Warmuth and a visit of Gunnar Rätsch to UC Santa Cruz were partially funded by the NSF grant CCR-9821087. Gunnar Rätsch would like to thank CRIEPI, ANU and UC Santa Cruz for warm hospitality.

A Extensions

A.1 Regression using a Bias

In some cases it might be useful to have a bias term. One way to achieve this is using a hypothesis class that includes the constant function. But then – depending on the definition of the regularizer – the bias is penalized, too. Therefore, it

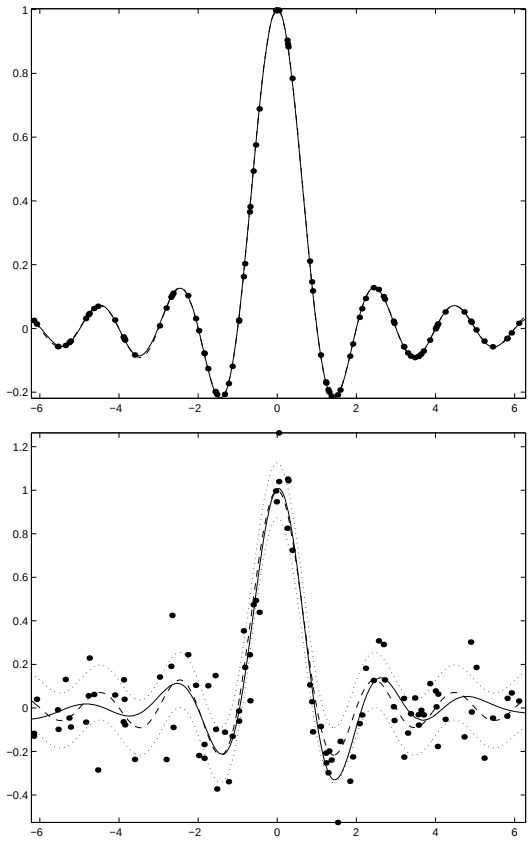

Figure 1: Toy example: The left panel shows the fit of the sinc function without noise (training samples: dots, fit: line) after a large number of iterations without any regularization. It is almost perfect, i.e. empirically the algorithm converges to the optimal solution of the unregularized linear program. The right panel shows a fit of a noisy sinc function with regularization parameter $C' = 4.0$ (training samples: dots, fit: solid, ε-tube: dotted, true function: dashed).

is worth investigating how to explicitely introduce it to ε-Boost.

In regression with bias b one has to find some function $f_{\boldsymbol{\alpha},b} \in \mathrm{aff}(\mathcal{H}), f_{\boldsymbol{\alpha},b} : \mathcal{X} \to \mathbb{R}$:

$$f_{\boldsymbol{\alpha},b}(\mathbf{x}) = \sum_{j=1}^{J} \alpha_j h_j(\mathbf{x}) + b$$
$$\text{with} \qquad \boldsymbol{\alpha} \in \mathbb{R}^J, \mathbf{x} \in \mathcal{X}, b \in \mathbb{R},$$

Thus, we only need to change (32) and all referring equations. In particular one needs to change the definition of δ_n:

$$\delta_n := y_n - b - \sum_{j=1}^{J} \alpha_j h_j(\mathbf{x}_n)$$

However, the problem is how to find the minimizing b for (40) in each iteration. One way is to optimize b separately in each iteration, e.g. after finding α_{I_t} (cf. Algorithm 1). Another approach, likely to be more efficient in the number of boosting iterations, would be to find b and α_{I_t} simultaneously. Certainly, optimizing two variables instead of one is much more computationally expensive than for one. But standard methods like the Newton algorithm or Conjugate Gradient algorithms work quite fast for two variables. Note that for both approaches Theorem 4 and Corrolary 5 can be easily extended.

A.2 Adaptive ε-Boost

One problem of the ε-insensitive loss used in ε-Boost is that it has a free parameter ε specifying the tube size. Tuning ε needs some knowledge about the problem at hand. In [42] a way for automatically tuning ε for SVMs was proposed: given a constant $\nu \in (0, 1)$, the tube-size is automatically chosen such that approximately a fraction of ν patterns lie outside the ε-tube.

This idea was extended in [45] for SVM regression using linear programming. It turns out to be a straightforward extension of (36):

$$\begin{array}{ll} \min & \nu\varepsilon + \frac{1}{N}\sum_{n=1}^{N} \xi_n + \xi_n^*, \varepsilon \geq 0 \\ \text{with} & \boldsymbol{\xi}, \boldsymbol{\xi}^* \in \mathbb{R}_+^N, \boldsymbol{\alpha} \in \mathbb{R}^J \\ & y_n - f_{\boldsymbol{\alpha},b}(\mathbf{x}_n) \leq \varepsilon + \xi_n, \quad n = 1, \ldots, N \\ & -y_n + f_{\boldsymbol{\alpha},b}(\mathbf{x}_n) \leq \varepsilon + \xi_n^*, \quad n = 1, \ldots, N \\ & \|\boldsymbol{\alpha}\| \leq C' \end{array} \tag{47}$$

Hence the difference between (36) and (47) lies in the fact that ε has become a positively constrained variable of the optimization problem itself. The ν property for (47), stated below (see [45, 37]) can be shown straight forward.

Proposition 6 ([45]). *Assume $\varepsilon > 0$. Suppose we run* (47). *The following statements hold:*

(i) ν is an upper bound on the fraction of errors (i.e. points outside the ε-tube).

(ii) ν is a lower bound on the fraction of points not inside (i.e. outside or on the edge of) the ε tube.

References

[1] M. Anthony and P. Bartlett. *Neural Network Learning: Theoretical Foundations.* Cambridge University Press, 1999.

[2] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithm: Bagging, boosting and variants. *Machine Learning*, pages 105–142, 1999.

[3] K.P. Bennett, A. Demiriz, and J. Shawe-Taylor. A column generation algorithm for boosting. In *Prooceedings, 17th ICML*, 2000.

[4] K.P. Bennett and O.L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1:23–34, 1992.

[5] A. Bertoni, P. Campadelli, and M. Parodi. A boosting algorithm for regression. In W.Gerstner, A.Germond, M.Hasler, and J.-D. Nicoud, editors, *Proceedings ICANN'97, Int. Conf. on Artificial Neural Networks*, volume V of *LNCS*, pages 343–348, Berlin, 1997. Springer.

[6] D.P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 1995.

[7] B.E. Boser, I.M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, PA, 1992.

[8] L. Breiman. Prediction games and arcing algorithms. Technical Report 504, Statistics Department, University of California, 1997.

[9] M. Collins, R.E. Schapire, and Y. Singer. Adaboost and logistic regression unified in the context of information geometry. In *Colt'00*, 2000.

[10] R. Cominetti and J.-P. Dussault. A stable exponential penalty algorithm with superlinear convergence. *J.O.T.A.*, 83(2), 1994.

[11] T.G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 1999.

[12] M. Doljansky and M. Teboulle. An interior proximal algorithm and the exponential multiplier method for semidefinite programming. *SIAM J. Optim.*, 9(1):1–13, 1998.

[13] H. Drucker, R. Schapire, and P. Simard. Boosting performance in neural networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 7:705 – 719, 1993.

[14] N. Duffy and D. Helmbold. Leveraging for regression. In *Colt'00*, 2000.

[15] N. Duffy and D. Helmbold. Potential boosters? In S.A. Solla, T.K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 258–264. MIT Press, 2000.

[16] Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *EuroCOLT: European Conference on Computational Learning Theory*. LNCS, 1994.

[17] Y. Freund and R.E. Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 2000. to appear.

[18] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. Technical report, Department of Statistics, Sequoia Hall, Stanford Univerity, 1998.

[19] J.H. Friedman. Greedy function approximation. Technical report, Department of Statistics, Stanford University, 1999.

[20] K.R. Frisch. The logarithmic potential method of convex programming. Memorandum, University Institute of Economics, Oslo, May 13 1955.

[21] A.J. Grove and D. Schuurmans. Boosting in the limit: Maximizing the margin of learned ensembles. In *Proceedings of the Fifteenth National Conference on Artifical Intelligence*, 1998.

[22] J. Kivinen and M. Warmuth. Boosting as entropy projection. In *Colt'99*, 1999.

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

[24] B.W. Kort and D.P. Bertsekas. Multiplier methods for convex programming. In *Proc 1073 IEEE Conf. Decision Control, San-Diego, Calif.*, pages 428–432, 1973.

[25] J. Lafferty. Additive models, boosting, and inference for generalized divergences. In *Proc. 12th Annu. Conf. on Comput. Learning Theory*, pages 125–133, New York, NY, 1999. ACM Press.

[26] Y. LeCun, L. D. Jackel, L. Bottou, A. Brunot, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. A. Müller, E. Säckinger, P. Simard, and V. Vapnik. Comparison of learning algorithms for handwritten digit recognition. In F. Fogelman-Soulié and P. Gallinari, editors, *Proceedings ICANN'95 — International Conference on Artificial Neural Networks*, volume II, pages 53 – 60, Nanterre, France, 1995. EC2.

[27] D.G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley Publishing Co., Reading, second edition, May 1984. Reprinted with corrections in May, 1989.

[28] R. Maclin and D. Opitz. An empirical evaluation of bagging and boosting. In *Proc. of AAAI*, 1997.

[29] O.L. Mangasarian. Linear and nonlinear separation of patterns by linear programming. *Operations Research*, 13:444–452, 1965.

[30] O.L. Mangasarian. Mathematical programming in data mining. *Data Mining and Knowledge Discovery*, 42(1):183–201, 1997.

[31] L. Mason, P. Bartlett, and J. Baxter. Improved generalization through explicit optimization of margins. Technical report, Deparment of Systems Engineering, Australian National University, 1998.

[32] L. Mason, J. Baxter, P.L. Bartlett, and M. Frean. Functional gradient techniques for combining hypotheses. In A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 221–247. MIT Press, Cambridge, MA, 1999.

[33] L. Mosheyev and M. Zibulevsky. Penalty/barrier multiplier algorithm for semidefinite programming. *Optimization Methods and Software*, 1999. Accepted.

[34] T. Onoda, G. Rätsch, and K.-R. Müller. An asymptotical analysis and improvement of adaboost in the binary classification case. *Journal of Japanese Society for AI*, 15(2):287–296, 2000. In japanese.

[35] G. Rätsch, T. Onoda, and K.-R. Müller. Regularizing AdaBoost. In M.S. Kearns, S.A. Solla, and D.A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 564–570. MIT Press, 1999.

[36] G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for AdaBoost. *Machine Learning*, 2000. In press.

[37] G. Rätsch, B. Schökopf, A. Smola, S. Mika, T. Onoda, and K.-R. Müller. Robust ensemble learning. In A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 207–219. MIT Press, Cambridge, MA, 1999.

[38] G. Rätsch, B. Schölkopf, S. Mika, and K.-R. Müller. Svm and boosting: One class. Submitted to NIPS'00, May 2000.

[39] R.E. Schapire, Y. Freund, P. Bartlett, and W.S. Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. In *Proc. 14th International Conference on Machine Learning*, pages 322–330. Morgan Kaufmann, 1997.

[40] R.E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. In *Colt'98*, pages 80–91, 1998.

[41] B. Schölkopf, C.J.C. Burges, and A.J. Smola, editors. *Advances in Kernel Methods – Support Vector Learning*. MIT Press, 1999.

[42] B. Schölkopf, A. Smola, R. C. Williamson, and P.L. Bartlett. New support vector algorithms. *Neural Computation*, 12:1083 – 1121, 2000.

[43] H. Schwenk and Y. Bengio. AdaBoosting neural networks. In W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud, editors, *Proc. of the Int. Conf. on Artificial Neural Networks (ICANN'97)*, volume 1327 of *LNCS*, pages 967–972, Berlin, 1997. Springer.

[44] Y. Singer. Leveraged vector machines. In S.A. Solla, T.K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 610–616. MIT Press, 2000.

[45] A. Smola, B. Schölkopf, and G. Rätsch. Linear programs for automatic accuracy control in regression. In *Proceedings ICANN'99, Int. Conf. on Artificial Neural Networks*, Berlin, 1999. Springer.

[46] A. J. Smola. *Learning with Kernels*. PhD thesis, Technische Universität Berlin, 1998.

[47] A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, editors. *Advances in Large Margin Classifiers*. MIT Press, 2000.

[48] R.J. Vanderbei and D.F. Shanno. An interior point algorithm for nonconvex nonlinear porgramming. Technical Report SOR-97-21, Statistics and Operations Research,, Princeton University, 1997.

[49] V.N. Vapnik. *The nature of statistical learning theory*. Springer Verlag, New York, 1995.