# A Convergence Rate Analysis for LogitBoost, MART and Their Variant

**Peng Sun**[1] SUNP08@MAILS.TSINGHUA.EDU.CN
**Tong Zhang**[2] TZHANG@STAT.RUTGERS.EDU
**Jie Zhou**[1] JZHOU@TSINGHUA.EDU.CN

[1]Tsinghua National Laboratory for Information Science and Technology(TNList), Department of Automation, Tsinghua University, Beijing 100084, China
[2]Baidu Inc., Beijing, China and Department of Statistics, Rutgers University, NJ, USA

## Abstract

LogitBoost, MART and their variant can be viewed as additive tree regression using logistic loss and boosting style optimization. We analyze their convergence rates based on a new weak learnability formulation. We show that it has $O(\frac{1}{T})$ rate when using gradient descent only, while a linear rate is achieved when using Newton descent. Moreover, introducing Newton descent when growing the trees, as LogitBoost does, leads to a faster linear rate. Empirical results on UCI datasets support our analysis.

## 1. Introduction

Boosting is a successful machine learning algorithm for binary classification. After the introduction of the first practical boosting method – AdaBoost (Freund & Schapire, 1995), many variations have been proposed. It was shown boosting can be regarded as gradient descent in function space (Mason et al., 1999). Along this line, Friedman (2001) equipped boosting with tree base learner and formulated it as Multiple Additive Regression Tree (MART), where at each iteration a tree is added to fit the gradient using least squares. Plugging the logistic loss for binary classification, MART becomes a concrete classification method described in (Friedman, 2001). The procedure differs from the one in (Friedman et al., 2000) where Hessian was employed in the so-called LogitBoost procedure, leading to Newton steps instead of gradient descent.

One natural question is whether it Is beneficial to introduce Hessian instead of using only gradient. In (Friedman et al., 2000; Friedman, 2001), where the Hessian was firstly introduced into the Boosting framework, this question was not answered. Recently, Li (2009b; 2010a); Saberian et al. (2011) presented empirical results showing that using Hessian does lead to lower test errors; furthermore, Li (2009a; 2010b) presented empirical results that LogitBoost can decrease training loss faster than MART. However, a convergence rate analysis was absent in previous studies. In this paper, we attempt to fill this gap by theoretically showing how fast the training loss decreases, with or without the Hessian.

*Table 1.* LogitBoost and its variants

| | Gain for Tree Growth | Leaf Value |
|---|---|---|
| GBoost | gradient (12) | gradient (11) |
| MART | gradient (12) | Newton (15) |
| LogitBoost | Newton (16) | Newton (15) |

Looking inside the tree fitting process, we observe that gradient/Hessian descent can be used either when growing the trees or when fitting the leaf values. The combinations give rise to LogitBoost and its two variants (GBoost and MART) as shown in Table 1. MART and LogitBoost have already seen many successful real world applications (e.g., Web Page Ranking (Burges, 2010), Click-Through Rate estimation for web advertisements (Trofimov et al., 2012), Optical Digits Recognition (Shalev-Shwartz et al., 2011)). GBoost is a different procedure considered in this paper, which can serve as a baseline for the purpose of theoretical analysis.

Our main convergence results are: 1) GBoost only achieves $O(\frac{1}{T})$ rate; 2) Both MART and LogitBoost achieve linear rates; 3) LogitBoost has a faster linear rate than MART.

It is worth mentioning that there are two main difficulties of the analysis. 1) The logistic loss is not strongly convex. 2) The regression tree is hard to deal with. Our analysis begins with a one-instance toy example (Section 3), which could be seen as a technical device to be utilized for the analysis of other Boosting algorithms. We then generalize the simple analysis to the many-instance case by viewing the tree growth process as block coordinate descent. In

*Proceedings of the $31^{st}$ International Conference on Machine Learning*, Beijing, China, 2014. JMLR: W&CP volume 32. 

this work we assume that the weak learnability assumption holds, and introduce a novel weak learnability formulation (*i.e.*, lemma 8) that is more appropriate for the analysis of LogitBoost than previous formulations.

### 1.1. Related Work

Convergence Rate. Due to the special property of exponential loss, it is well known that AdaBoost has linear convergence rate (Schapire & Freund, 2012). For logistic loss, (Bickel et al., 2006) showed a sub-linear rate. Very recently, (Telgarsky, 2012; 2013) proved linear convergence for boosting with logistic loss.

The linear convergence rate has also been established for boosting with strongly convex losses in the literature (Rätsch et al., 2001; Grubb & Bagnell, 2011). Unfortunately logistic loss is not strongly convex. To address this difficulty, Telgarsky (2012) proposed a technique to compare the gradient of the loss and its Hessian. Inspired by this idea, a similar technique tailored to Newton descent is presented in this paper.

Weak Learnability. We employ a weak learnability assumption in this work[1]. In the AdaBoost literature (Schapire & Freund, 2012), the weak learnability is formulated with respect to weighted training error. In gradient boosting (Mason et al., 1999), (Grubb & Bagnell, 2011) connects the weighted error formulation to the gradient least square fitting described in (Friedman et al., 2000; Friedman, 2001). A similar weak learnability formulation for gradient fitting was used in (Telgarsky, 2012; 2013). However, the weak learnability assumptions in earlier work involved only gradient, making them unsuitable for our Hessian based Newton descent analysis.

Results most related to this paper are those of (Telgarsky, 2012; 2013). However, there are important differences: 1) The linear rate in (Telgarsky, 2012; 2013) relies on ad hoc step size selection which excludes the Newton descent; in particular, the analysis there doesn't cover MART and LogitBoost. In this work, we directly study Newton descent that is adopted by MART and LogitBoost. 2) The weak learnability assumption in (Telgarsky, 2012; 2013) is defined on gradient only, which is unsuitable to the analysis of Newton descent in the sense that it cannot reflect the convergence rate observed in our experiments. In contrast, we propose a new weak learnability assumption that is more appropriate for Newton descent analysis, and the assumption is justified both theoretically and empirically.

The rest of this paper is organized as follows. In Section 2 we review the problem setup. In Section 3 we introduce the main techniques via a toy example. Section 4 gives the formal analysis. Finally, in Section 5 we present empirical results on UCI datasets to support our claims.

[1]This assumption seems very strong. However, it is realistic in many real world applications. It is shown that weak learnability is equivalent to linear separability over the base learners (see a recent survey (Shalev-Shwartz & Singer, 2010) for this issue), which indeed covers many classification tasks. Fox example, in Optical Character Recognition the data with different labels cannot overlap(say, an image cannot be both '4' and '9'), thus the raw image data can be linearly separated over trees that are deep enough.

## 2. Review of LogitBoost and Its Variants

In this section we review the problem setup posed in (Friedman et al., 2000), including three key ingredients: the loss, the function model and the optimization method.

The definition of Logistic loss. Given a set of training data $\{\boldsymbol{x}_i, y_i\}_{i=1}^N$ where the feature $\boldsymbol{x}_i \in \mathcal{X}$ and the label $y_i \in \{\pm 1\}$, a classifier learns a predictor function $F = F(\boldsymbol{x}) \in \mathbb{R}$ by minimizing the total loss over the training dataset $\sum_{i=1}^N \ell(y_i, F(\boldsymbol{x}_i))$. In (Friedman et al., 2000), the instance wise loss $\ell(\cdot,\cdot)$ adopts the so-called logistic loss:

$$\ell(y, F) = r \log\left(\frac{1}{p}\right) + (1-r)\log\left(\frac{1}{1-p}\right), \quad (1)$$

where the $0/1$ response $r = 1$ if $y = +1$, $r = 0$ otherwise; and the probability estimate $p \in [0, 1]$ is a "surrogate" of $F$ via the so-called link function:

$$p = \psi(F) = \frac{e^F}{e^F + e^{-F}}, \quad (2)$$

which is a sigmoid-like function on $F$.

To carry out the numerical optimization, we need the gradient and the Hessian of $\ell(\cdot, F)$ w.r.t. $F$. Simple calculus gives them as:

$$g(y, F) \triangleq \nabla_F \ell(y, F) = 2(p - r), \quad (3)$$

$$h(F) \triangleq \nabla_F^2 \ell(y, F) = 4p(1-p), \quad (4)$$

by noting the derivative of the link function is $\psi'(F) = 2p(1-p)$.

Note that the above formulae are all defined on the implicit variables $r$ and $p$, rather than defined directly on $y$ and $F$. This choice turns out to be very convenient for our later development.

The Additive Tree Model. $F(\boldsymbol{x})$ is expressed as sum of regression trees: $F(\boldsymbol{x}) = \sum_{t=1}^T u_t(\boldsymbol{x})$. Omitting the subscript $t$, a regression tree $u(\boldsymbol{x})$ with $J$ leaves can be written as $u(\boldsymbol{x}) = \sum_{j=1}^J s_j I(\boldsymbol{x} \in R_j)$, where $s_j \in \mathbb{R}$ is the fitted value on the $j$-th leaf, $R_j$ is the region on feature space corresponding to the $j$-th leaf and the indicator function

$I(\cdot) = 1$ if its argument is true and 0 otherwise. In the AdaBoost literature, regression tree is a domain-partition based weak learner with confidence-rated output (Schapire & Singer, 1999).

The Greedy Stage Wise Optimization. At each boosting iteration only one tree is added and the other trees previously learned are fixed. What's special here is that when updating $F = F(\boldsymbol{x})$ a real value $\nu \in (0, 1]$ is multiplied: $F(\boldsymbol{x}) \leftarrow F(\boldsymbol{x}) + \nu u(\boldsymbol{x})$. The fixed step size $\nu$, designated in advance as a tuning parameter, is called shrinkage factor controlling the learning rate (Friedman et al., 2000). In later literature, it is found to have the effect of $L1$ regularization (Rosset et al., 2004). In several up-to-date implementations, this choice leads to satisfactory classification performance, e.g., in (Li, 2010a) a stable and good result is observed by always setting $\nu = 0.1$. The corresponding pseudo-code in given in Algorithm 1.

**Algorithm 1** LogitBoost, MART and Variant.

**input** $\{\boldsymbol{x}_i, y_i\}_{i=1}^N$: the training set; $\nu$: the shrinkage factor; $T$: the maximum number of iterations.
**output** the Additive Tree Model $F = F(\boldsymbol{x})$.
1: $F_i = 0, i = 1, \ldots, N$
2: **for** $t = 1$ to $T$ **do**
3: $p_i = \psi(F_i)$ as in Eq (2) and clap on $p_i$ as in (17), $i = 1, \ldots, N$.
4: Grow a tree with domain partition $\{R_j\}_{j=1}^J$.
5: Fit a value $s_j$ for the $j$-th leaf, $j = 1, \ldots, J$.
6: $F_i \leftarrow F_i + \nu \sum_{j=1}^J s_j I(\boldsymbol{x}_i \in R_j), \quad i = 1, \ldots, N$.
7: **end for**

### 2.1. Tree Growth and Leaf Value Fitting

In Algorithm 1, it needs to specify how to grow a tree (line 4) and how to fit the leaf value (line 5), giving rise to the difference among GBoost, MART and LogitBoost. We provide greater details here. At some iteration, denote by $\boldsymbol{u} = (u_1, \ldots, u_N)^\top \in \mathbb{R}^N$ the fitted leaf values over the $N$ instances. Define the total loss $L(\boldsymbol{u}) \triangleq \sum_{i=1}^N \ell(y_i, F_i + u_i)$. Consider its tractable quadratic approximation at 0: $\widehat{L}(\boldsymbol{u}) = \widehat{L}(0) + \boldsymbol{g}^\top \boldsymbol{u} + \frac{1}{2}\boldsymbol{u}^\top \boldsymbol{H}\boldsymbol{u}$, where $\widehat{L}(0) = L(0) = \sum_{i=1}^N \ell(y_i, F_i)$ and the instance wise gradient and Hessian

$$\boldsymbol{g} = (g_1, \ldots, g_N)^\top \quad (5), \quad \boldsymbol{H} = \mathrm{diag}(h_1, \ldots, h_N) \quad (6)$$

are respectively calculated by (3) and (4) with $p$ replaced by $p_i$ at each instance $i$. Since typically $J < N$, each component of $\boldsymbol{u}$ cannot vary independently: the instances falling into the same leaf must share a common fitted value. We can thus write down $\boldsymbol{u} = \boldsymbol{V}\boldsymbol{s}$, where $\mathbb{R}^J \ni \boldsymbol{s} = (s_1, \ldots, s_j, \ldots, s_J)^\top$ are the fitted values on the $J$ leaves and $\boldsymbol{V} \in \mathbb{R}^{N \times J}$ is a projection matrix:

$$\boldsymbol{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_j, \ldots, \boldsymbol{v}_J], \quad (7)$$

where $\boldsymbol{v}_{j,i} = 1$ if the $j$-th leaf contains the $i$-th instance and 0 otherwise, $j = 1, \ldots, J, i = 1, \ldots, N$. Substituting $\boldsymbol{u} = \boldsymbol{V}\boldsymbol{s}$ back to the quadratic $\widehat{L}(\boldsymbol{u})$ and reload the notation $\widehat{L}(\cdot)$ for $\boldsymbol{s} \in \mathbb{R}^J$ we have

$$\widehat{L}(\boldsymbol{s}) = \widehat{L}(0) + (\boldsymbol{g}^\top \boldsymbol{V})\boldsymbol{s} + \frac{1}{2}\boldsymbol{s}^\top (\boldsymbol{V}^\top \boldsymbol{H}\boldsymbol{V})\boldsymbol{s}. \quad (8)$$

To this extent, the leaf value can be defined as the minimizer $\boldsymbol{s}^* = \arg\min_{\boldsymbol{s}} \widehat{L}(\boldsymbol{s})$, while the total split gain[2] for growing a tree can be defined as the maximum loss reduction $\mathrm{gain}(\boldsymbol{s}^*) = \max_{\boldsymbol{s}} \left(\widehat{L}(0) - \widehat{L}(\boldsymbol{s})\right)$, both boiling down to minimizing (8). There are two methods.

Gradient Descent. Because the Hessian (4) is bounded above by $h \leq 1$, we can replace $\boldsymbol{H}$ with the upper bound $\boldsymbol{I}$, *i.e.*, the identity matrix. Solving the consequent quadratic problem using matrix calculus (Magnus & Neudecker, 2007), we have:

$$\boldsymbol{s}^* = -(\boldsymbol{V}^\top \boldsymbol{V})^{-1} \boldsymbol{V}^\top \boldsymbol{g}, \quad (9)$$

$$\mathrm{gain}(\boldsymbol{s}^*) = \|\boldsymbol{V}\boldsymbol{s}^*\|_2^2 = (\boldsymbol{V}^\top \boldsymbol{g})^\top (\boldsymbol{V}^\top \boldsymbol{V})^{-1} (\boldsymbol{V}^\top \boldsymbol{g}). \quad (10)$$

Note that $\boldsymbol{V}^\top \boldsymbol{V} = \mathrm{diag}(n_1, \ldots, n_j, \ldots, n_J) \in \mathbb{R}^{J \times J}$ where $n_j$ denotes the number of instances falling into the $j$-th leaf. Moreover, (9) and (10) can be rewritten in the "scalar form":

$$s_j^* = -\frac{\overline{g}_j}{n_j} \quad (11) \qquad \mathrm{gain}(\boldsymbol{s}^*) = \sum_{j=1}^J \frac{\overline{g}_j^2}{n_j} \quad (12)$$

where $\overline{g}_j \triangleq \sum_{i \in \mathcal{I}_j} g_i$ should be understood as "node wise gradient". The geometrical meaning is straightforward: in the subspace $\mathrm{span}(\boldsymbol{V})$ we find a vector $\boldsymbol{u} = \boldsymbol{V}\boldsymbol{s} \in \mathbb{R}^N$ that is as close to $-\boldsymbol{g}$ as possible, which in (Friedman, 2001) is described as fitting the negative gradient $\{-g_i\}_{i=1}^N$ using a least squares regression tree.

Newton Descent. Directly solving (8) we have

$$\boldsymbol{s}^* = -(\boldsymbol{V}^\top \boldsymbol{H}\boldsymbol{V})^{-1} \boldsymbol{V}^\top \boldsymbol{g}, \quad (13)$$

$$\mathrm{gain}(\boldsymbol{s}^*) = \|\boldsymbol{V}\boldsymbol{s}^*\|_{\boldsymbol{H}}^2 = (\boldsymbol{V}^\top \boldsymbol{g})^\top (\boldsymbol{V}^\top \boldsymbol{H}\boldsymbol{V})^{-1} (\boldsymbol{V}^\top \boldsymbol{g}), \quad (14)$$

which can be rewritten in "scalar form" as:

$$s_j^* = -\frac{\overline{g}_j}{\hbar_j}, \quad (15) \qquad \mathrm{gain}(\boldsymbol{s}^*) = \sum_{j=1}^J \frac{\overline{g}_j^2}{\hbar_j}, \quad (16)$$

[2] While it is difficult to directly maximize the total gain, (Friedman et al., 2000) suggests to grow a binary tree top-down until $J$ leaves are encountered, maximizing a binary split gain once a time.

where $\hbar_j \triangleq \sum_{i \in \mathcal{I}_j} h_i$ should be understood as "node wise Hessian". The geometrical meaning is similar to the case of gradient descent, but the Euclidean norm should be replaced by matrix norm (the Hessian matrix $\boldsymbol{H}$) when talking about "close". This is described in (Friedman et al., 2000) as fitting the Newton step $\{-g_i/h_i\}_{i=1}^N$ with weights $\{h_i\}_{i=1}^N$ using a weighted least square regression tree.

### 2.2. Clapping on $p_i$

Normally (15) is numerically stable. When $p_i \to r_i$ for all $i \in \mathcal{I}_j$, it can be verified that $s_j \to \frac{1}{2}$ although both $\overline{g}_j \to 0$ and $\hbar_j \to 0$. However, (15) is still occasionally very large when only $\hbar_j \to 0$, causing numerical problems. To tackle this issue, (Friedman et al., 2000) suggests the Newton step (15) to be clapped in a range, say, between $[-5, +5]$. In this work, however, we implement the idea in a slightly different way. Instead of directly clapping the Newton step (15), we clap the $p_i$, $i = 1, \ldots, N$:

$$p_i = \begin{cases} 1 - \rho, & (r_i = 0) \wedge (p_i > 1 - \rho) \\ \rho, & (r_i = 1) \wedge (p_i < \rho) \\ p_i, & \text{otherwise} \end{cases} \tag{17}$$

for some small constant $\rho$ in line 4 of Algorithm 1. From (1), we observe that the clapping (17) is to prevent large penalty of wrong probability estimate (or negative margin in regards of $F \in \mathbb{R}$). In this way, we can show that the Newton step is finite as in the theorem below. The proof is provided in the supplement. Although motivated by making the Newton step numerically stable, we also apply the clapping on $p_i$ in the case of gradient descent.

**Theorem 1** (Bounded Newton Step)**.** *With the value clapping on $p_i$, $i = 1, \ldots, N$ as in (17), the Newton step (15) is bounded such that $\left|\overline{g}_j/\hbar_j\right| \leq 1/(2\rho)$.*

## 3. One-Instance Toy Example

Having introduced the problem setup, we are ready for the analysis. As will be seen shortly in Section 4, the convergence rate heavily depends on how the leave values are fitted, *i.e.*, whether the gradient (11) or the Newton (15) is adopted. In this section we present the main results and the underlying proof techniques via a toy example.

### 3.1. Algorithm 1 with One Instance

Suppose there is only one training instance with label, say, $y = +1$. In this case it is unnecessary to grow a tree in Algorithm 1, and only the leaf value fitting (line 5) matters. Consequently, Algorithm 1 boils down to performing some numerical descent method on the function $-\log(p)$, as in the following pseudo code:

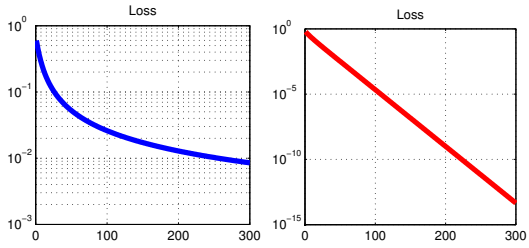


*Figure 1.* How the loss decreases with iteration for the one instance toy example. The vertical axis is in log scale. **Left**: Gradient Descent; **Right**: Newton Descent.

1: $F = 0$, $p = 1/2$.
2: **for** $t = 1$ to $T$ **do**
3: Update: $F \leftarrow F + f$.
4: Compute the implicit variable via (2): $p = \psi(F)$.
5: Compute the loss at iteration $t$: $L_t = -\log(p)$.
6: **end for**

Obviously, the optimal loss value, 0, is achieved when $F = +\infty$. The step $f$ in line 3 can be either gradient descent or Newton descent times a shrinkage factor. Now let's take a closer look.

Gradient Descent. The step $f = -\nu g$, where $g = 2(p - 1)$ as in (3). With $\nu = 0.1$, the result of how $L_t$ decreases with iteration $t$ is shown on the left of Figure 1.

Newton Descent. The step $f = -\nu \frac{g}{h}$, where $g = 2(p - 1)$, $h = 4p(1 - p)$ as in (3) and (4), respectively. With $\nu = 0.1$, the result of how $L_t$ decreases is shown on the right of Figure 1.

From Figure 1, we can clearly see that the gradient descent shows a sub-linear convergence rate consistent with $O(\frac{1}{T})$, while the Newton descent shows a linear rate, *i.e.*, $O(e^{-T})$. Next we will provide a quick theoretic explanation for this phenomenon.

### 3.2. Theoretical Analysis

We follow the standard techniques in convex optimization to derive the desired bound. First, we bound the one step loss reduction; then we derive the recurrence relation from iteration $t - 1$ to iteration $t$; finally we obtain the convergence rate in $t$ from the recurrence relation.

The first part is vital to our analysis. We begin with several related theorems.

**Theorem 2** (Smoothness)**.** *Let $\ell(\cdot)$ be the shorthand of $\ell(y, \cdot)$ and $g$ be the shorthand of gradient (3). Then $\ell(\cdot)$ is 1-smooth and the one step loss reduction satisfies*

$$\ell(F + f) \leq \ell(F) + gf + \frac{1}{2} f^2. \tag{18}$$

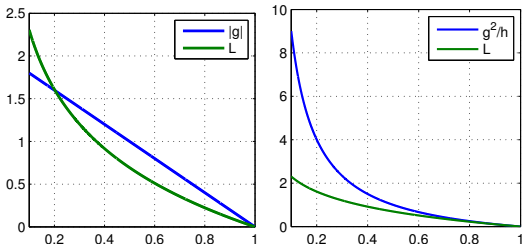


*Figure 2.* **Left**: $|g|$ v.s. $\ell$ as in Theorem 3 when the label $y = +1$; **Right**: $\frac{g^2}{h}$ v.s. $\ell$ as in Theorem 4. In both graphs, the horizontal axis is $p$.

We omit the proof of Theorem 2, which follows from the standard argument in convex optimization textbooks, e.g., (Boyd & Vandenberghe, 2004) and the fact that the Hessian in (4) is upper-bounded by $h \leq 1$.

**Theorem 3** (Comparison I)**.** *Let $g$ and $\ell$ be the shorthand for gradient (3) and loss (1), respectively. When $p \geq \rho$ in the case of $y = +1$ (or $p \leq 1 - \rho$ in the case of $y = -1$), there exists a constant $\alpha = \alpha(\rho) > 0$ such that $|g| \geq \alpha\ell$.*

**Theorem 4** (Comparison II)**.** *Let $g$, $h$ and $\ell$ be the shorthand for gradient (3), Hessian (4) and loss (1), respectively. Then $\forall p$ (and thus $\forall F$), the inequality $\frac{g^2}{h} \geq \ell$ always holds.*

See the supplement for the proofs of the above two theorems. The loss family with the properties $|g| \geq aL$ and $h \leq bL$ for some constants $a$, $b$ was studied in (Telgarsky, 2012). This includes but not restricted to the logistic loss (1). The properties imply the inequality $\frac{g^2}{h} \geq \frac{a^2}{b}L$, which is similar to theorem 4. Obviously, the results in (Telgarsky, 2012) are more general, while Theorem 4 in this work is specific to the logistic loss (1). As will be seen shortly, Theorem 4 is useful in the analysis of Newton step.

In the left of Figure 2 we demonstrate Theorem 3 with $\alpha = 1$ (corresponding to $\rho \approx 0.2$) when $y = +1$; While in the right of Figure 2 we demonstrate Theorem 4 when $y = +1$.

The following two theorems establish convergence rates from the corresponding recurrence relationships.

**Theorem 5** (Recurrence to $O(\frac{1}{T})$ rate)**.** *If a sequence $\epsilon_0 \geq \ldots \geq \epsilon_{t-1} \geq \epsilon_t \geq \ldots \geq \epsilon_T > 0$ has the recurrence relation $\epsilon_t \leq \epsilon_{t-1} - c\epsilon_{t-1}^2$ for a small constant $c > 0$, then we have $O(\frac{1}{T})$ convergence rate: $\epsilon_T \leq \frac{\epsilon_0}{1+c\epsilon_0 T}$.*

**Theorem 6** (Recurrence to linear rate)**.** *If a sequence $\epsilon_0 \geq \ldots \geq \epsilon_{t-1} \geq \epsilon_t \geq \ldots \geq \epsilon_T > 0$ has the recurrence relation $\epsilon_t \leq c\epsilon_{t-1}$ for a small constant $0 < c < 1$, then we have linear convergence: $\epsilon_T \leq \epsilon_0 c^T$.*

See the supplement for the proof of theorem 5. The proof for theorem 6 is obvious and we omit it here. Next we are ready to analyze the toy example.

#### 3.2.1. Gradient Descent

Denote the loss at current iteration by $L$ and that at next iteration by $L^+$. From Theorem 2, we have

$$L^+ \leq L + gf + \frac{1}{2}f^2. \tag{19}$$

Substituting the gradient descent times a shrinkage factor $f = -\nu g$ in the right hand side, we have:

$$L^+ \leq L - \nu(1 - \frac{1}{2}\nu)g^2. \tag{20}$$

Noting the initial value $F = 0$, we apply Theorem 3 with $\alpha = 1$ and get:

$$L^+ \leq L - \nu(1 - \frac{1}{2}\nu)L^2. \tag{21}$$

This leads to a recurrence relationship from iteration $t-1$ to $t$. Setting a proper $\nu$ such that $\nu(1 - \frac{1}{2}\nu) > 0$, (21) implies a $O(\frac{1}{T})$ convergence rate due to Theorem 5.

Remark. If the loss is $m$-strongly convex, we can show $g^2 \geq (2m)L$, leading to a linear convergence rate, which well known in convex optimization, e.g., (Boyd & Vandenberghe, 2004; Nesterov, 2004). However, this argument cannot be applied to logistic loss because the minimum of the Hessian (4) is 0, implying that the strong convexity doesn't hold.

#### 3.2.2. Newton Descent

For $f \geq 0$, we use the Mean Value Theorem in calculus and have

$$L^+ = L + gf + \frac{1}{2}h_\xi f^2, \tag{22}$$

$\xi \in [0, f]$ and $h_\xi$ denotes the Hessian at $\xi$. Substituting the Newton Step times a shrinkage factor $f = -\nu\frac{g}{h} \geq 0$, we have

$$L^+ = L - \nu\frac{g^2}{h} + \frac{1}{2}\nu^2\frac{h_\xi}{h}\frac{g^2}{h}. \tag{23}$$

Since the initial value $F = 0$ and $f \geq \xi \geq 0$, it holds that

$$\frac{h_\xi}{h} \leq 1 \tag{24}$$

by checking the form of the Hessian (4) and the form of the link (2). Noticing that $\frac{g^2}{h} \geq 0$, the inequality simplifies to

$$L^+ \leq L - \nu(1 - \frac{1}{2}\nu)\frac{g^2}{h}. \tag{25}$$

By applying Theorem 4, we obtain the recurrence relation

$$\begin{aligned} L^+ &\leq L - \nu(1 - \frac{1}{2}\nu)L \\ &= (1 - \nu(1 - \frac{1}{2}\nu))L. \end{aligned} \tag{26}$$

By setting a proper value for $\nu$ such that $0 < (1 - \nu(1 - \frac{1}{2}\nu)) < 1$, we get the linear convergence rate according to Theorem 6.

Remark. If the loss is strong convex and the Hessian is strictly Lipschitz continuous, we can show that the Newton Step leads to quadratic convergence rate, which is again a standard result in convex optimization textbook, e.g., (Boyd & Vandenberghe, 2004; Nesterov, 2004).

We make the following remarks concerning the one-instance example of this section. 1) The toy example becomes realistic if we allow a very large $J$ such that at each leaf there is only one training instance. In general, when $J < N$ we intuitively cannot expect a faster convergence rate than what is established here. 2) The proof for the toy example can be adapted to handle the general case, as we shall illustrate in the next section.

## 4. The General Analysis

In this section we provide the full analysis for $N$ instances and $J$ leaves. Basically, we extend the one-instance analysis to the one-leaf analysis and sum up for the totally $J$ leaves using matrix notation. The analysis in Section 4.1 covers GBoost, while that in Section 4.2 covers MART and LogitBoost.

### 4.1. Gradient Descent

In Section 3.2.1 we have established convergence rates for the toy example, and the key step is the application of Theorem 3 to obtain (21) from (20). To do the same in the $J$-leaf case, we need to convert "node wise gradient" to "instance wise gradient". This idea is captured by Definition 1 and Lemma 7.

**Definition 1** (Weak Learnability)**.** For a set of $N$ instances with labels $\{\pm 1\}^N$ and non-negative weights $\{w_1, \ldots, w_N\}$, there exists a $J$-leaf classification tree (*i.e.*, outputting $\pm 1$ at each leaf) such that the weighted error rate at each leaf is strictly less than $\frac{1}{2}$ by at least $\delta > 0$.

This definition is the weak learnability assumption in the AdaBoost literature (Schapire & Freund, 2012). The $J$-leaf tree can be seen as a so-called domain-partition weak learner (Schapire & Singer, 1999), where each leaf, an atomic domain with output value $\pm 1$, can be regarded as a weak classifier. Definition 1 assumes that each leaf classifier is better than random guessing by $\delta > 0$.

**Lemma 7** (Weak Learnability I)**.** *Assume that the weak learnability assumption as in Definition 1 holds, then for any gradient $\boldsymbol{g} \in \mathbb{R}^N$ as defined in (5) and $g_i \neq 0$, $i = 1, \ldots, N$, there exists a $J$-leaf regression tree whose projection matrix $\boldsymbol{V} \in \mathbb{R}^{N \times J}$ satisfies*

$$(\boldsymbol{V}^\top \boldsymbol{g})^\top (\boldsymbol{V}^\top \boldsymbol{V})^{-1} (\boldsymbol{V}^\top \boldsymbol{g}) \geq \gamma^2 \boldsymbol{g}^\top \boldsymbol{g} \tag{27}$$

*for some constant $\gamma > 0$.*

See the supplement for its proof. Recall the gain (10) and its geometric meaning, Lemma 7 says that we can always find a subspace $\mathrm{span}(\boldsymbol{V})$ such that the projected $\boldsymbol{g}$ has big enough norm compared with the original $\boldsymbol{g}$. The proof of Lemma 7 was due to (Grubb & Bagnell, 2011) which contains a very similar result. In the following we will use the second part of Lemma 7 to derive our bounds.

At some iteration, consider the stepsize vector $\boldsymbol{f} = (f_1, \ldots, f_N)^\top \in \mathbb{R}^N$, and denote the total current loss by $L \triangleq \sum_{i=1}^N \ell(y_i, F_i)$ and the total loss at next iteration by $L^+ \triangleq \sum_{i=1}^N \ell(y_i, F_i + f_i)$. Using Theorem 2, we can bound the loss reduction as

$$L^+ \leq L + \boldsymbol{g}^\top \boldsymbol{f} + \frac{1}{2} \boldsymbol{f}^\top \boldsymbol{f}. \tag{28}$$

Replacing $\boldsymbol{f}$ by the gradient descent in (9) times the shrinkage factor $\nu$:

$$\boldsymbol{f} = -\nu \boldsymbol{V} \boldsymbol{s} = -\nu \boldsymbol{V} (\boldsymbol{V}^\top \boldsymbol{V})^{-1} \boldsymbol{g}, \tag{29}$$

we have:

$$\begin{aligned} L^+ \leq\ & L - \nu (\boldsymbol{V}^\top \boldsymbol{g})^\top (\boldsymbol{V}^\top \boldsymbol{V})^{-1} (\boldsymbol{V}^\top \boldsymbol{g}) \\ & + \frac{1}{2} \nu^2 (\boldsymbol{V}^\top \boldsymbol{g})^\top (\boldsymbol{V}^\top \boldsymbol{V})^{-1} (\boldsymbol{V}^\top \boldsymbol{V}) (\boldsymbol{V}^\top \boldsymbol{V})^{-1} (\boldsymbol{V}^\top \boldsymbol{g}). \end{aligned} \tag{30}$$

Noting that $\boldsymbol{V}^\top \boldsymbol{V}$ is invertible and eliminating $(\boldsymbol{V}^\top \boldsymbol{V})^\top (\boldsymbol{V}^\top \boldsymbol{V})^{-1}$, we obtain

$$L^+ \leq L - \nu (1 - \frac{1}{2} \nu) (\boldsymbol{V}^\top \boldsymbol{g})^\top (\boldsymbol{V}^\top \boldsymbol{V})^{-1} (\boldsymbol{V}^\top \boldsymbol{g}). \tag{31}$$

With the help of Lemma 7, it yields

$$L^+ \leq L - \nu (1 - \frac{1}{2} \nu) \gamma^2 \|\boldsymbol{g}\|_2^2. \tag{32}$$

Recall the norm inequality $\|\boldsymbol{g}\|_2^2 \geq \frac{1}{N} \|\boldsymbol{g}\|_1^2$ and multiply both sides by $\frac{1}{N}$, we obtain

$$\frac{L^+}{N} \leq \frac{L}{N} - \nu (1 - \frac{1}{2} \nu) \gamma^2 \left( \frac{\|\boldsymbol{g}\|_1}{N} \right)^2. \tag{33}$$

Recall the clapping operation (17) and Theorem 3, we can compare $\|\boldsymbol{g}\|_1$ to $L$ as

$$\|\boldsymbol{g}\|_1 \geq \alpha L \tag{34}$$

for some $\alpha = \alpha(\rho)$. Substituting back, we obtain the recurrence relation:

$$\frac{L^+}{N} \leq \frac{L}{N} - \nu (1 - \frac{1}{2} \nu) \gamma^2 \alpha^2 \left( \frac{L}{N} \right)^2. \tag{35}$$

Viewing $\frac{L^+}{N}$ as $\epsilon_t$, $\frac{L}{N}$ as $\epsilon_{t-1}$ and setting proper $\nu$ so that $\nu(1 - \frac{1}{2}\nu)\gamma^2 \alpha^2 > 0$, we establish the $O(\frac{1}{T})$ convergence rate with the help of Theorem 5.

### 4.2. Newton Descent

In Section 3.2.2 we have established convergence rate for the toy example by using the following two ideas:

- The application of the comparison theorem 4 so as to transit from (25) to (26). To do the same in the $J$-leaf $N$-instance case, we need to convert the "node wise Newton Decrement" to "instance wise Newton Decrement". This idea is captured by Lemma 8 below.
- The upper bound of the term $\frac{h_\xi}{h}$ given by (24). In the general $J$-leaf $N$-instance case, a similar upper bound is available as in Lemma 9 and Corollary 10 below.

We are now ready to present lemmas needed for our subsequent analysis.

**Lemma 8** (Weak Learnability II)**.** *Let $\boldsymbol{g} \in \mathbb{R}^N$ ($g_i \neq 0$, $i = 1, \ldots, N$) be defined in (5) and $\boldsymbol{H} \in \mathbb{R}^{N\times N}$ be defined in (6), where $p_i$ is clapped as in (17). If the weak learnability assumption 1 holds, then there exists a $J$-leaf regression tree whose projection matrix $\boldsymbol{V} \in \mathbb{R}^{N\times J}$ satisfies*

$$(\boldsymbol{V}^\top \boldsymbol{g})^\top (\boldsymbol{V}^\top \boldsymbol{H}\boldsymbol{V})^{-1}(\boldsymbol{V}^\top \boldsymbol{g}) \geq \gamma_*^2 \boldsymbol{g}^\top \boldsymbol{H}^{-1}\boldsymbol{g}, \quad (36)$$

*for some constant $\gamma_* > 0$.*

See the supplement for its proof. Lemma 8 is a new weak learnability result that can be used to analyze Newton descent. It says that with the same weak learnability assumption in Lemma 7, we can find a subspace $span(\boldsymbol{V})$ such that the projection of $\boldsymbol{g}$ in the subspace has a sufficiently large matrix norm induced by the Hessian matrix $\boldsymbol{H}$. The geometric meaning is given in (14).

**Lemma 9** (Change of Hessian)**.** *For the current $F \in \mathbb{R}$ and a step $f \in \mathbb{R}$, the change of the Hessian (4) can be bounded in terms of step size $|f|$:*

$$\frac{h(F+f)}{h(F)} \leq e^{2|f|}. \quad (37)$$

See the supplement for its proof. The role of Lemma 9 is similar to that of equation (24) in the toy example of Section 3.2.2.

We are now ready to derive our general results similar to what we have shown in Section 3.2.2. We define $L^+$, $L$, and $\boldsymbol{f}$ with the same meanings of the corresponding quantities in Section 4.1. Using second order Taylor expansion, we obtain

$$L^+ = L + \boldsymbol{g}^\top \boldsymbol{f} + \frac{1}{2}\boldsymbol{f}^\top \boldsymbol{H}_\xi \boldsymbol{f}, \quad (38)$$

where $\xi \in \mathbb{R}^N$ such that each of its component $\xi_i$ lies within $F_i$ and $F_i + f_i$ ($i = 1, \ldots, N$), and $\boldsymbol{H}_\xi$ is a shorthand of the Hessian matrix at $\xi$. Replacing $\boldsymbol{f}$ with the Newton step (13) times a shrinkage factor $\nu$:

$$\boldsymbol{f} = -\nu \boldsymbol{V}\boldsymbol{s} = -\nu \boldsymbol{V}(\boldsymbol{V}^\top \boldsymbol{H}\boldsymbol{V})^{-1}\boldsymbol{V}^\top \boldsymbol{g}, \quad (39)$$

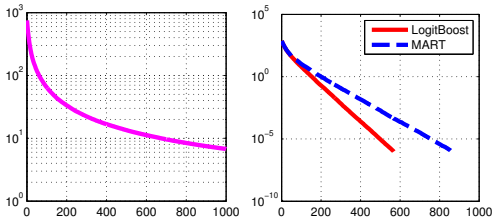


*Figure 3.* A typical convergence pattern on UCI dataset #3. The horizontal axis indicates iterations, and the vertical axis indicates logistic loss on training data in log scale. **Left**: GBoost; **Right**: MART and LogitBoost.

we have:

$$L^+ = L - \nu(\boldsymbol{V}^\top \boldsymbol{g})^\top (\boldsymbol{V}^\top \boldsymbol{H}\boldsymbol{V})^{-1}\boldsymbol{V}^\top \boldsymbol{g} + \frac{1}{2}\nu^2 (\boldsymbol{V}^\top \boldsymbol{g})^\top (\boldsymbol{V}^\top \boldsymbol{H}\boldsymbol{V})^{-1}(\boldsymbol{V}^\top \boldsymbol{H}_\xi \boldsymbol{V})(\boldsymbol{V}^\top \boldsymbol{H}\boldsymbol{V})^{-1}\boldsymbol{V}^\top \boldsymbol{g}, \quad (40)$$

where we can bound the term $(\boldsymbol{V}^\top \boldsymbol{H}_\xi \boldsymbol{V})(\boldsymbol{V}^\top \boldsymbol{H}\boldsymbol{V})^{-1}$ using Theorem 9 as follows.

**Corollary 10** (Node wise Hessian)**.** *Assume that the Newton step (39) is used with the value clapping (17). Then $(\boldsymbol{V}^\top \boldsymbol{H}_\xi \boldsymbol{V})(\boldsymbol{V}^\top \boldsymbol{H}\boldsymbol{V})^{-1} \leq \mu \boldsymbol{I}$, where $\mu = \exp(\nu/\rho)$.*

The proof is given in the supplement. For now we can apply Corollary 10 to obtain the inequality

$$L^+ \leq L - \nu(1 - \frac{1}{2}\nu\mu)(\boldsymbol{V}^\top \boldsymbol{g})^\top (\boldsymbol{V}^\top \boldsymbol{H}\boldsymbol{V})(\boldsymbol{V}^\top \boldsymbol{g}). \quad (41)$$

Using Lemma 8, we convert the node wise Newton decrement to instance wise Newton decrement in the sense that

$$(\boldsymbol{V}^\top \boldsymbol{g})^\top (\boldsymbol{V}^\top \boldsymbol{H}\boldsymbol{V})(\boldsymbol{V}^\top \boldsymbol{g}) \leq \gamma_*^2 \boldsymbol{g}^\top \boldsymbol{H}^{-1}\boldsymbol{g}. \quad (42)$$

Substituting back, the inequality simplifies as:

$$L^+ \leq L - \nu(1 - \frac{1}{2}\nu\mu)\gamma_*^2 \boldsymbol{g}^\top \boldsymbol{H}^{-1}\boldsymbol{g}. \quad (43)$$

Finally, we connect Newton objective reduction and loss reduction using Theorem 4 as follows

$$\boldsymbol{g}^\top \boldsymbol{H}^{-1}\boldsymbol{g} \geq L, \quad (44)$$

which, together with (43), yields the recurrence relation:

$$L^+ \leq (1 - \nu(1 - \frac{1}{2}\nu\mu)\gamma_*^2)L. \quad (45)$$

With proper step size $\nu$ ensuring $0 < \nu(1 - \frac{1}{2}\nu\mu)\gamma_*^2 < 1$, we can establish the desired linear convergence rate from Theorem 6.

Remark 1. Our analysis for Newton descent is based on the new weak learnability result in Lemma 8. From (45), the

quality of our bound relies on the constant $\gamma_*$ defined in (36): the bigger the $\gamma_*$, the faster the convergence. It is worth mentioning that the bound in the related work (Telgarsky, 2012) is similar to (27), a choice that is unsuitable for Newton descent. We confirm this assertion by experiments in Section 5, where we show that large $\gamma$ doesn't lead to faster convergence while large $\gamma_*$ does.

Remark 2. According to the above discussion, we should prefer LogitBoost to MART in order to achieve faster convergence. This is because when growing a tree, MART does not search for good $\gamma_*$.

## 5. Experiments

In this section we present empirical results on five binary datasets: #1: optdigits05, #2: pendigits49, #3: zipcode38, #4: letter01, #5: mnist10k05. These are synthesized from the corresponding UCI datasets; e.g., "optdigits05" means we pick class 0 and class 5 from the multiclass datasets "optdigits". In all of the following experiments we set $\nu = 0.1$, $J = 8$ and the clapping $\rho = 0.05$.

Convergence rate We note that the empirical results of (Li, 2009a; 2010b) have already demonstrated that the training loss of LogitBoost decreases faster than that of MART, e.g., Figure 2 in (Li, 2010b). For completeness, we perform a similar experiment here, focusing on binary classification and adding GBoost in our comparisons. In the supplement, we plot the convergence for GBoost, MART and LogitBoost on datasets #1 to #5. Figure 3 shows the typical convergence pattern. As in our analysis, GBoost (*i.e.*, the gradient descent) shows a sub-linear convergence rate of $O(\frac{1}{T})$, while MART and LogitBoost (*i.e.*, the Newton descent) show linear rates.

Weak Learnability and Faster Linear Rate In this work, our analysis for Newton descent relies on the constant $\gamma_*$ defined in (36): a bigger $\gamma_*$ indicates faster convergence. To study this issue empirically, we look at the ratio $\frac{(\boldsymbol{V}^\top \boldsymbol{g})^\top (\boldsymbol{V}^\top \boldsymbol{H} \boldsymbol{V})^{-1} (\boldsymbol{V}^\top \boldsymbol{g})}{\boldsymbol{g}^\top \boldsymbol{H}^{-1} \boldsymbol{g}}$ over all iterations and pick the minimum value as $\gamma_*$. Similarly, we can set $\gamma$ defined in (36) to be the minimum value of $\frac{(\boldsymbol{V}^\top \boldsymbol{g})^\top (\boldsymbol{V}^\top \boldsymbol{V})^{-1} (\boldsymbol{V}^\top \boldsymbol{g})}{\boldsymbol{g}^\top \boldsymbol{g}}$ — this choice is employed by (Telgarsky, 2012) where the resulting bound contains a quantity being similar to $\gamma$. Table 2 lists the values $\gamma_*$ and $\gamma$ for the two algorithms LogitBoost and MART, and it gives the number of iterations $T'$ needed to achieve $10^{-6}$ accuracy for logistic loss.

As can be seen in Table 2 and Figure 3, LogitBoost converges faster than MART. Checking Table 1, we also find that LogitBoost has bigger $\gamma_*$ than MART. However, the faster convergence of LogitBoost cannot be captured by $\gamma$. For example, on dataset #3 LogitBoost has a smaller $\gamma$ value than that of MART while still converges faster (the third row in Table 2 and the right graph in Figure 3). This shows that the analysis of Newton descent should depend on $\gamma_*$ instead of $\gamma$.

*Table 2.* The weak learnability for LogitBoost and MART

| | MART | | | LogitBoost | | |
|---|---|---|---|---|---|---|
| | $\gamma_*$ | $\gamma$ | $T'$ | $\gamma_*$ | $\gamma$ | $T'$ |
| #1 | 0.565 | 0.317 | 217 | 0.817 | 0.400 | 206 |
| #2 | 0.157 | 0.047 | 500 | 0.466 | 0.034 | 269 |
| #3 | 0.056 | 0.047 | 865 | 0.219 | 0.039 | 568 |
| #4 | 0.078 | 0.029 | 518 | 0.263 | 0.027 | 345 |
| #5 | 0.042 | 0.037 | 1043 | 0.203 | 0.021 | 582 |

**Acknowledgments** The authors would like to thank the reviewers for their helpful comments that improved the paper. This work is supported by the National Natural Science Foundation of China under Grants 61225008 and 61020106004, the National Basic Research Program of China under Grant 2014CB349304, the Ministry of Education of China under Grant 20120002110033, and the Tsinghua University Initiative Scientific Research Program.

## References

Bickel, Peter J, Ritov, Ya'acov, and Zakai, Alon. Some theory for generalized boosting algorithms. *The Journal of Machine Learning Research*, 7:705–732, 2006.

Boyd, Stephen Poythress and Vandenberghe, Lieven. *Convex optimization*. Cambridge university press, 2004.

Burges, Chris. From ranknet to lambdarank to lambdamart: An overview. *Microsoft Research Technical Report MSR-TR-2010-82*, 2010.

Freund, Y. and Schapire, R. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory*, pp. 23–37. Springer, 1995.

Friedman, Jerome, Hastie, Trevor, and Tibshirani, Robert. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2):337–407, 2000.

Friedman, Jerome H. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pp. 1189–1232, 2001.

Grubb, Alexander and Bagnell, Drew. Generalized boosting algorithms for convex optimization. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 1209–1216, 2011.

Li, Ping. Abc-boost: Adaptive base class boost for multi-class classification. In *Proceedings of the 26th Annu-*

*al International Conference on Machine Learning*, pp. 625–632. ACM, 2009a.

Li, Ping. Abc-logitboost for multi-class classification. *arXiv preprint arXiv:0908.4144*, 2009b.

Li, Ping. Robust logitboost and adaptive base class (abc) logitboost. In *Uncertainty in Artificial Intelligence*, 2010a.

Li, Ping. An empirical evaluation of four algorithms for multi-class classification: Mart, abc-mart, robust logitboost, and abc-logitboost. *arXiv preprint arXiv:1001.1020*, 2010b.

Magnus, Jan R and Neudecker, Heinz. *Matrix differential calculus with applications in statistics and econometrics*. John Wiley & Sons, 3rd edition, 2007.

Mason, Llew, Baxter, Jonathan, Bartlett, Peter, and Frean, Marcus. Boosting algorithms as gradient descent in function space. In *Advances in Neural Information Processing Systems*. NIPS, 1999.

Nesterov, Yurii. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer, 2004.

Rätsch, Gunnar, Mika, Sebastian, and Warmuth, Manfred K. On the convergence of leveraging. In *Advances in Neural Information Processing Systems*, pp. 487–494, 2001.

Rosset, Saharon, Zhu, Ji, and Hastie, Trevor. Boosting as a regularized path to a maximum margin classifier. *The Journal of Machine Learning Research*, 5:941–973, 2004.

Saberian, Mohammad J, Masnadi-Shirazi, Hamed, and Vasconcelos, Nuno. Taylorboost: First and second-order boosting algorithms with explicit margin control. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pp. 2929–2934. IEEE, 2011.

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

Schapire, Robert E and Freund, Yoav. *Boosting: Foundations and Algorithms*. The MIT Press, 2012.

Shalev-Shwartz, Shai and Singer, Yoram. On the equivalence of weak learnability and linear separability: New relaxations and efficient boosting algorithms. *Machine learning*, 80(2-3):141–163, 2010.

Shalev-Shwartz, Shai, Wexler, Yonatan, and Shashua, Amnon. Shareboost: Efficient multiclass learning with feature sharing. In *NIPS*, 2011.

Telgarsky, Matus. A primal-dual convergence analysis of boosting. *The Journal of Machine Learning Research*, 13:561–606, 2012.

Telgarsky, Matus. Margins, shrinkage, and boosting. In *Proceedings of the 30th International Conference on Machine Learning (ICML2013)*, pp. 307–315, 2013.

Trofimov, Ilya, Kornetova, Anna, and Topinskiy, Valery. Using boosted trees for click-through rate prediction for sponsored search. In *Proceedings of the Sixth International Workshop on Data Mining for Online Advertising and Internet Economy*, pp. 2. ACM, 2012.