

COS511 Term Project

New Results on Boosting with L_1 Regularization

Yongxin (Taylor) XI

Zhen (James) XIANG

Abstract

Boosting algorithms with L_1 regularization on the coefficient of the weak learners, denoted as L_1 -Boost, are the main focus of the paper. We are interested in L_1 -Boost not only because its potential to provide a simpler model (fewer weak learners used), but also its asymptotic margin maximizing behavior. We observe that L_1 -Boost often behave similarly to ε -Boost, a forward stagewise boosting algorithm with tiny step size ε . So we first examine the relationship between the two algorithms. Then we conduct margin analysis of the two algorithms using two popular loss functions: exponential loss and binomial log likelihood loss. We prove a bound on the number of iterations needed by ε -Adaboost algorithm¹ to achieve the maximum margin θ^* up to a given precision. We also show that L_1 -Adaboost achieves margin at least $(\theta^* - r^{-1} \log m)$, where r is the L_1 regularization parameter, and m is the number of training examples. Similar analysis is conducted for ε -Logit and L_1 -Logit. Despite the nice properties, these algorithms are computationally expensive. We therefore try to explore an important property of L_1 -Boost and propose a new algorithm with potential improvement in speed at the end of the paper. The new algorithm, named as L_1 -Boost.turn, solves for the turning points of the solution path of L_1 -Boost where a new weak learner is included.

1 Introduction

Boosting is a methodology used to combine weak learners into one stronger learner, with the objective to maximize the margin of the training data. For example, the most widely known boosting algorithm, Adaboost, has the effect of enlarging the margin even after achieving consistency of the data. Because margin size is directly related with generalization, this explains the good generalization of Adaboost [1]. However, Adaboost has not been proven to converge to a maximizing margin solution. We show that such convergence can be achieved by adding L_1 regularization to the original problem, denoted as L_1 -Adaboost. Let us call boosting with L_1 regularization as L_1 -Boost problem. Recent work of S. Rosset et. al. [2] has shown that when using a certain loss function, L_1 -Boost converges to achieve the “best margin” as the regularization vanishes. This type of loss function is called a “margin maximizing loss function”. It includes the popular exponential loss function and log likelihood loss function. However, Rosset et. al. refer the “best margin” to the best margin achievable under the **particular** loss function, rather than the maximum achievable margin θ^* given the weak hypotheses space; also, the dynamics

¹ ε -Adaboost = ε -Boost algorithm using the exponential loss; L_1 -Adaboost = L_1 -Boost using the exponential loss; ε -Logit = ε -Boost algorithm using the binomial log likelihood loss; L_1 -Logit = L_1 -Boost using the binomial log likelihood loss.

of the convergence is not quantitatively captured in the proof. Therefore, we examine the exponential loss function and the log likelihood loss function in this paper to see if they can achieve θ^* . Also, we provide a bound on the regularization parameter to achieve θ^* up to a given precision.

A variant of L_1 -Boost algorithm is ε -Boost. ε -Boost is a forward stagewise boosting algorithm with small step size ε . The two algorithms often yield similar solution paths, and ε -Boost also has the margin maximizing property. Furthermore, ε -Boost is much easier to implement than L_1 -Boost. We therefore are motivated to examine the close relationship between L_1 -Boost and ε -Boost algorithms. Inspired by T. Hastie’s results for forward stagewise regression and the monotone lasso [3], we believe that the solution paths of L_1 -Boost and ε -Boost are the same under the assumption that the coefficients of all active learners are monotone increasing, but the proof is still under investigation.

Besides margin maximization, it is worth mentioning two other attractive properties of L_1 -Boost and ε -Boost. Firstly, through regularization on coefficients of weak learners, they have control over the complexity of the final hypothesis. The regularization parameter of L_1 -Boost can be adjusted for the desired degree of complexity. Adjusting the number of iterations of ε -Boost has a similar effect. When there is a tradeoff between the model complexity and margin size, cross validation can be used to decide a proper parameter value to use. For data with irrelevant features, such control is especially important. We will illustrate this in the experiment section. The second property is that L_1 -Boost enables “uniform learning” of all the active weak learners with a piecewise smooth coefficient update of these learners. We show that all the active learners have the same gradient along the solution path, i.e., they always contribute equally to minimizing the objective function. Approximately “uniform learning” of every active weak learner can also be found in the totally corrective boosting algorithm [4]. This algorithm can also converge to a margin maximizing solution. Thus, uniform learning might be a crucial aspect for boosting algorithms to achieve the maximum margin.

L_1 -Boost and ε -Boost are computationally expensive. We have to solve L_1 -Boost for many different values of the regularization parameter to find the solution path. As for ε -Boost, it usually iterates tens of thousands of times to find the solution path. To address this problem, we use the uniform learning property of L_1 -Boost to propose a new algorithm L_1 -Boost.turn. Under the assumption of a unique uniform learning curve, we show L_1 -Boost.turn yields the exact solutions of L_1 -Boost at turning points, i.e., points where a new learner is added.

The paper is organized as follows. In Section 2, we formulate the learning problem and describe L_1 -Boost and ε -Boost algorithms. Then we discuss the condition for the two algorithms to yield identical solution paths. In Section 3, we conduct margin analysis of the two boosting algorithms using two loss functions. ε -Adaboost and L_1 -Adaboost are proved to converge to a max-margin solution. We also give bounds for L_1 -Logit and ε -Logit. In section 4, we first show the “uniform learning” property of L_1 -Boost. Following that the new algorithm L_1 -Boost.turn is derived, and we prove that L_1 -Boost.turn yield identical solutions as L_1 -Boost at turning points. Finally in Section 5 we experiment using ε -Adaboost and the original Adaboost on two artificial data sets (with or without irrelevant features). We plot the evolution of margins of all examples for ε -Adaboost for

each data set. Then we compare the prediction performance of the two algorithms for the two data sets. Section 6 concludes with discussion of future work. Detailed proof of theorems and lemmas can be found in the Appendix section.

2 L_1 -Boost and ε -Boost

Our basic assumptions and notations of the boosting problem are very much the same as in class. Suppose we are given m data samples: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2) \dots (\mathbf{x}_m, y_m)$. For each sample (\mathbf{x}_i, y_i) , \mathbf{x}_i is the data vector and $y_i \in \{-1, +1\}$ is the label. We have a weak hypothesis space $\mathcal{H} = \{h_1, h_2 \dots h_{|\mathcal{H}|}\}$. Each hypothesis h_j maps each data point to $+1$ or -1 : $h_j(\mathbf{x}_i) \in \{-1, +1\}$. Similar to what we did in Adaboost, we assume the data is *weak learnable* by θ^* : for any distribution $\{\omega_i\}_{i=1}^m$ ($\omega_i \geq 0, \sum_{i=1}^m \omega_i = 1$), there exists a weak learner $h \in \mathcal{H}$, s.t. $\sum_{i=1}^m \omega_i y_i h(\mathbf{x}_i) \geq \theta^* > 0$. In other words, for any probability distribution $\{\omega_i\}_{i=1}^m$ on data samples, we can always find a weak hypothesis with an error probability less than $\frac{1-\theta^*}{2}$.

Under these assumptions, we hope to find a combined hypothesis $H = \sum_{j=1}^{|\mathcal{H}|} \alpha_j h_j$ that best fits the data. Quantitatively this means the combined hypothesis minimizes the sum of loss functions $L(y_i, H(\mathbf{x}_i))$ on the data samples:

$$\min \sum_{i=1}^m L(y_i, H(\mathbf{x}_i)) \quad (1)$$

in particular, we are interested in the exponential loss function $L_{exp}(y_i, H(\mathbf{x}_i)) = e^{-y_i H(\mathbf{x}_i)}$, and binomial log likelihood loss function $L_{logit}(y_i, H(\mathbf{x}_i)) = \ln(1 + e^{-y_i H(\mathbf{x}_i)})$.

Although this is just minimizing the error on the training set, from class we know that good generalization capability can be achieved if we use a “simple” hypothesis to fit the data well. One way to make sure our hypothesis is “simple“ is to impose a L_1 constraint on the vector of coefficients $\mathbf{A} = (\alpha_1, \alpha_2, \dots, \alpha_{|\mathcal{H}|}) \in \mathcal{R}^{|\mathcal{H}|}$. In this case the optimization problem becomes:

$$\begin{aligned} \min \quad \mathcal{L}(\mathbf{A}) &= \sum_{i=1}^m L(y_i, \sum_{j=1}^{|\mathcal{H}|} \alpha_j h_j(\mathbf{x}_i)) \\ \text{s.t.} \quad |\mathbf{A}|_1 &= \sum_{j=1}^{|\mathcal{H}|} |\alpha_j| \leq r \end{aligned} \quad (2)$$

For example, if we use exponential loss, $\mathcal{L}(\mathbf{A}) = \sum_{i=1}^m \exp(-y_i \sum_{j=1}^{|\mathcal{H}|} \alpha_j h_j(\mathbf{x}_i))$; if we use binomial log likelihood loss, $\mathcal{L}(\mathbf{A}) = \sum_{i=1}^m \ln(1 + \exp(-y_i \sum_{j=1}^{|\mathcal{H}|} \alpha_j h_j(\mathbf{x}_i)))$. Also, the negative derivative $-\frac{\partial \mathcal{L}}{\partial |\alpha_j|}$ ² can be viewed as learner h_j 's current capability to reduce the loss function.

We call problem (2) the $L_1(r)$ -Boost problem. For every fixed r , we denote the optimal solution of problem (2) as $\mathbf{A}(r)$. As r increases, the points $\mathbf{A}(r)$ form a trajectory in

² $-\frac{\partial \mathcal{L}}{\partial |\alpha_j|}$ is the decreasing rate of loss function L when $|\alpha_j|$ increases. More specifically:

$$-\frac{\partial \mathcal{L}}{\partial |\alpha_j|} = \begin{cases} -\frac{\partial \mathcal{L}}{\partial \alpha_j} & \text{if } \alpha_j \geq 0 \\ \frac{\partial \mathcal{L}}{\partial \alpha_j} & \text{if } \alpha_j < 0 \end{cases}$$

space $\mathcal{R}^{|\mathcal{H}|}$, which is called the solution path of L_1 -Boost. L_1 -Adaboost and L_1 -Logit are specific instances of L_1 -Boost, which use exponential loss and binomial log likelihood loss, respectively.

Although L_1 -Boost minimizes the cost function under L_1 regularization directly, it is hard to solve. In practice, we could use an algorithm called “ ε -Boost”, which is described as follows:

1. select a tiny step size ε
2. set $\mathbf{B}(0) = \mathbf{0}$
3. for $n = 0, 1, 2, 3, \dots$, let $\mathbf{B}((n+1)\varepsilon) = \mathbf{B}(n\varepsilon) + (0, 0, \dots, \varepsilon, \dots, 0)$, where ε is in the j^{th} position and

$$j = \arg \max_j -\frac{\partial \mathcal{L}(\mathbf{B}(n\varepsilon))}{\partial \beta_j} \quad (3)$$

So ε -Boost produces a solution path with step length ε by increasing the weight of learner with the greatest decent by ε in each round. While ε -Boost produces a similar L_1 regularization effect, it does not necessarily generate the same solution path as L_1 -Boost. The reason is that L_1 -Boost is the global optimal solution to problem (2), while ε -Boost is solving the problem using local step search.

An interesting problem is to find conditions under which ε -Boost is equivalent to L_1 -Boost when $\varepsilon \rightarrow 0$. In ε -Boost’s solution path $\mathbf{B}(r) = (\beta_1(r), \beta_2(r), \dots, \beta_{|\mathcal{H}|}(r))$, each $\beta_j(r)$ is non-decreasing w.r.t. r . So if ε -Boost and L_1 -Boost have the same solution path, then one necessary condition is that in $\mathbf{A}(r) = (\alpha_1(r), \alpha_2(r) \dots \alpha_{|\mathcal{H}|}(r))$, $\alpha_j(r)$ is also non-decreasing w.r.t. r . We believe that this is also a sufficient condition: If $\alpha_j(r)$ is non-decreasing, then ε -Boost and L_1 -Boost have the same solution when $\varepsilon \rightarrow 0$. A rigorous proof to this proposition is still under our investigation.

In following sections ε -Adaboost refers to the ε -Boost algorithm using the exponential loss, and ε -Logit refers to the ε -Boost algorithm using the binomial log likelihood loss.

3 Margin Analysis

Theorem 1. *Min-Max-Theorem, von Neumann (1928)*

$$\theta^* := \min_{\mathbf{w}} \max_{j=1, \dots, |\mathcal{H}|} \sum_{i=1}^m \omega_i y_i h_j(\mathbf{x}_i) = \max_{\mathbf{A}} \min_{i=1, \dots, m} y_i \sum_{j=1}^{|\mathcal{H}|} \alpha_j h_j(\mathbf{x}_i) \quad (4)$$

where $\mathbf{w} = (\omega_1, \omega_2 \dots \omega_m) \in \mathcal{P}^m$, $\mathbf{A} = (\alpha_1, \alpha_2 \dots \alpha_{|\mathcal{H}|}) \in \mathcal{P}^{|\mathcal{H}|}$. Here \mathcal{P}^k denotes the k -dimensional probability simplex.

This theorem says the minimum edge θ^* that can be achieved over all possible distributions \mathbf{w} of the training examples is equal to the maximum margin of any linear combination of hypotheses from \mathcal{H} . So, under the assumption that the data is weak learnable by θ^* , there exists a linear combination of hypotheses that attains a margin θ^* .

We now prove the ε -Adaboost algorithm progressively updates this linear combination to achieve the maximum margin θ^* .

Theorem 2. *Assume the maximum margin achievable is θ^* . ε -Adaboost will converge to a solution with margin at least $(\theta^* - \varepsilon - \delta)$ ($\theta^* > \varepsilon > 0, \delta > 0$) on all examples after at most $(\varepsilon\delta)^{-1} \ln m$ iterations.*

To prove Theorem 2, we first introduce a lemma and a definition.

Lemma 2.1. *Let $0 < \varepsilon < 0.5$. Assume the maximum margin achievable is θ^* . After n steps, ε -Adaboost decreases the objective function by at least:*

$$\mathcal{L}(\mathbf{B}(n\varepsilon)) \leq \left(\frac{1 - \theta^*\varepsilon}{1 - \varepsilon^2}\right)^n \mathcal{L}(\mathbf{B}(0)) = \left(\frac{1 - \theta^*\varepsilon}{1 - \varepsilon^2}\right)^n m \quad (5)$$

Proof. Use Taylor Series expansion. See detailed proof in the appendix. \square

Definition 1. *Let $H_n(\mathbf{x}_i) = \sum_{j=1}^{\lfloor n \rfloor} \beta_j(n\varepsilon) h_j(\mathbf{x}_i)$ denote the combined hypothesis at the n^{th} step of the ε -Boost algorithm.*

Proof. Theorem [2]: According to Theorem 1, the coefficient vector \mathbf{A} lies on the probability simplex, so its L_1 norm equals 1. So we need to normalize the coefficients to unit L_1 norm to compute the margin. The margin of the i^{th} example after n iterations is then

$$\theta_i = \frac{y_i H_n(\mathbf{x}_i)}{|\mathbf{B}(n\varepsilon)|_1}$$

Now we can express the cost function $\mathcal{L}(\mathbf{B}(n\varepsilon))$ in terms of the margins θ_i :

$$\mathcal{L}(\mathbf{B}(n\varepsilon)) = \sum_{i=1}^m e^{-y_i H_n(\mathbf{x}_i)} = \sum_{i=1}^m e^{-|\mathbf{B}(n\varepsilon)|_1 \theta_i} > e^{-n\varepsilon \theta_{\min}} \quad (6)$$

where θ_{\min} denotes the smallest margin. From Lemma [2.1] and (6),

$$\theta_{\min} > \frac{\ln(1 + \frac{\theta^*\varepsilon - \varepsilon^2}{1 - \theta^*\varepsilon})}{\varepsilon} - \frac{\ln m}{n\varepsilon} \approx \theta^* - \varepsilon - \frac{\ln m}{n\varepsilon} \quad (7)$$

By setting $\delta = \frac{\ln m}{n\varepsilon}$ we immediately get $n = \frac{\ln m}{\varepsilon\delta}$. \square

Corollary 2.1. *The solution of L_1 -Adaboost achieves margin at least $(\theta^* - r^{-1} \ln m)$.*

Proof. Let $r = n\varepsilon$. Obviously $\mathcal{L}(\mathbf{A}(n\varepsilon)) \leq \mathcal{L}(\mathbf{B}(n\varepsilon))$ because L_1 -Adaboost produces the minimizer given the L_1 norm constraint. Denote $\theta_{\min.L1}$ as the margin achieved. Following the proof of Theorem [2], we have $\theta_{\min.L1} > \theta^* - \varepsilon - \frac{\ln m}{n\varepsilon} = \theta^* - \varepsilon - \frac{\ln m}{r}$. Let $\varepsilon \rightarrow 0$ we obtain $\theta_{\min.L1} > \theta^* - \frac{\ln m}{r}$. \square

Theorem 2 and its corollary state that ε -Adaboost and L_1 -Adaboost can both achieve the maximum margin θ^* up to a given precision. And the residual of the achieved margin to θ^* is inversely proportional to the L_1 norm r .

Theorem 3. *Assume the maximum margin achievable is θ^* . ε -Logit will converge to a solution with margin at least $(\frac{N_2}{N_1+N_2} C^*(\theta^* - \varepsilon) - \frac{\ln(m \ln 2)}{(N_1+N_2)\varepsilon})$ ($C^*\theta^* > \varepsilon > 0$) on all examples after at most $(N_1 + N_2)$ iterations, where $C^* = \frac{1}{2 \ln 2} \approx 0.7213$. N_1 is the number of iterations needed to achieve consistency of the combined hypothesis on the data, i.e., a margin larger than 0. N_1 is bounded by $N_1 \leq \frac{2m \ln 2}{\theta^*\varepsilon}$.*

Lemma 3.1. *After at most $\frac{2m \ln 2}{\theta^*\varepsilon}$ steps, the margin achieved by ε -Logit is larger than 0.*

Proof. Use a lower bound of the decrease of the objective function at each iteration. See detailed proof in the appendix. \square

Lemma 3.2. *After N_1 iterations of ε -Logit, n additional steps decreases the objective function by at least:*

$$\mathcal{L}(\mathbf{B}((N_1 + n)\varepsilon)) \leq \left(\frac{1 - C^*\theta^*\varepsilon}{1 - \varepsilon^2}\right)^n \mathcal{L}(\mathbf{B}(N_1\varepsilon)) < \left(\frac{1 - C^*\theta^*\varepsilon}{1 - \varepsilon^2}\right)^n m \ln 2 \quad (8)$$

Proof. The proof is similar to the proof of Lemma [2.1]. The distinction is the introduction of a constant C^* . This is due to the fact that the derivative of the binomial loss function on the right half axis can be lower bounded by C^* times the loss function itself. See the detailed proof in the appendix. \square

Proof. Theorem [3]: By Lemma [3.2] and following a proof similar to Theorem [2], we obtain the lower bound of the margin. \square

Corollary 3.1. *The solution of $L_1(r)$ -Logit achieves margin at least $(1 - \frac{2m \ln 2}{\theta^* r})C^*\theta^* - \frac{\ln(m \ln 2)}{r}$, where $C^* = \frac{1}{2 \ln 2}$.*

Proof. Similar to the argument in Corollary [2.1], since the L_1 norm of ε -Logit after $(N_1 + N_2)$ iterations is $r = (N_1 + N_2)\varepsilon$, the margin θ_ε that ε -Logit achieves is at least $\theta_\varepsilon \geq (1 - \frac{N_1}{N_1 + N_2})C^*(\theta^* - \varepsilon) - \frac{\ln(m \ln 2)}{r} \geq (1 - \frac{2m \ln 2 / (\theta^* \varepsilon)}{r/\varepsilon})C^*(\theta^* - \varepsilon) - \frac{\ln(m \ln 2)}{r} = (1 - \frac{2m \ln 2}{\theta^* r})C^*(\theta^* - \varepsilon) - \frac{\ln(m \ln 2)}{r}$. By letting $\varepsilon \rightarrow 0$ the margin $\theta_{L_1, r}$ that L_1 -Logit achieves is at least $\theta_{L_1, r} \geq \theta_\varepsilon \geq (1 - \frac{2m \ln 2}{\theta^* r})C^*\theta^* - \frac{\ln(m \ln 2)}{r}$. \square

Theorem [3] and its corollary give lower bounds on the margin that ε -Logit and L_1 -Logit achieve. These bounds suggest that ε -Logit and L_1 -Logit can at least achieve margin $C^*\theta^*$ up to a given precision, and in practice they usually achieve better margins than the bound given here.

4 New Algorithm L_1 -Boost.turn

In this section we discuss how to solve L_1 -Boost, and introduce our new efficient algorithm. A common way to solve L_1 -Boost is to solve many convex problems (2) for many sampled r values. The denser the sampling, the more accurate the solution path is, but the more time consuming the computation. But in fact we care most about the turning points of the solution path where a new weak learner is added, because they are the best solutions given fixed budgets of the number of weak learners to use. So we derive a new algorithm called “ L_1 -Boost.turn”, which solves for the turning points of L_1 -Boost.

Before we describe the algorithm, let us first examine an interesting property of L_1 -Boost. At every point of the solution path, all nonzero coefficients have the same derivative:

Theorem 4. *If $\mathbf{A}(r) = (\alpha_1, \alpha_2 \dots \alpha_{|\mathcal{H}|})$ and $\alpha_i, \alpha_j \neq 0$, then*

$$-\frac{\partial \mathcal{L}(\mathbf{A}(r))}{\partial |\alpha_i|} = -\frac{\partial \mathcal{L}(\mathbf{A}(r))}{\partial |\alpha_j|} \quad (9)$$

See appendix for proof.

This property implies that the solution path for L_1 -Boost is a “uniform learning” curve, which means that all active learners (learners with $\alpha \neq 0$) have the same derivative. Readers who are familiar with LASSO (linear regression with L_1 regularization) probably

have already noticed that LASSO has this property too. In the regression literature, a recent algorithm called Least Angle Regression [5] uses this property to solve for the turning points of LASSO where a new regressor is included, and is therefore much more efficient. Inspired by LARS, we use this property to solve for the turning points of L_1 -Boost.

Let us define the N^{th} turning point to be the place where the $(N + 1)^{\text{th}}$ learner is just added. Suppose we already have $(N - 1)$ turning points and the identities of the N active learners. Denote the $(N - 1)^{\text{th}}$ turning point as $\mathbf{A}_{N-1} = (\alpha_1, \alpha_2, \dots, \alpha_{N-1}, 0, \dots, 0) \in \mathbf{R}^{|\mathcal{H}|}$. Now we want to solve for the N^{th} point, where the N learners have the same derivative as the $(N + 1)^{\text{th}}$ learner to be added. Therefore N equations (usually nonlinear) can be formulated to solve for the updates for the coefficients of the N learners: $\mathbf{A}_N = (\alpha_1 + \Delta\alpha_1, \alpha_2 + \Delta\alpha_2, \dots, \alpha_{N-1} + \Delta\alpha_{N-1}, \Delta\alpha_N, 0, \dots, 0)$. We should then check if any coefficient reverses sign. Because if the coefficient of a learner passes 0, the constraint in Theorem 4 no longer has to hold. Our strategy for this is to backtrack to the point where the coefficient is 0, exclude that learner, then start over again.

L_1 -Boost.turn can be described as follows:

1. Select a budget M of the number of weak learners to be used.
2. Set $\mathbf{A} = \mathbf{0}$. Set $I = \{1, 2, \dots, |\mathcal{H}|\}$ to be the indices of inactive learners.
3. Find the current best learner $j^* = \arg \max_j \{-\frac{\partial \mathcal{L}(\mathbf{A})}{\partial \alpha_j}\}$. Include j^* into the active learner set: $A = \{j^*\}$ and delete it from the inactive learner set: $I = I - \{j^*\}$.
4. For $N = 1, 2, \dots, M$

- For each learner j from I

Solve the N equations for the N unknowns $\Delta\alpha_1(j), \Delta\alpha_2(j), \dots, \Delta\alpha_N(j)$

$$\begin{aligned} & \frac{\partial \mathcal{L}(\mathbf{A} + [\Delta\alpha_1(j), \Delta\alpha_2(j), \dots, \Delta\alpha_N(j), 0, \dots, 0])}{\partial \alpha_j} \\ &= \frac{\partial \mathcal{L}(\mathbf{A} + [\Delta\alpha_1(j), \Delta\alpha_2(j), \dots, \Delta\alpha_N(j), 0, \dots, 0])}{\partial \alpha_k}, \forall k \in A \end{aligned}$$

- Select learner j^* which creates the minimum positive update of the L_1 norm of \mathbf{A} , i.e. $j^* = \arg \min_j \{\sum_{i=1}^N \Delta\alpha_i(j)\}_+$.
- Check if any entry of $\mathbf{A} + [\Delta\alpha_1(j^*), \Delta\alpha_2(j^*), \dots, \Delta\alpha_N(j^*), 0, \dots, 0]$ changes sign.

If no sign has been changed

$$A = A + \{j^*\}; I = I - \{j^*\};$$

$$\mathbf{A} = \mathbf{A} + [\Delta\alpha_1(j^*), \Delta\alpha_2(j^*), \dots, \Delta\alpha_N(j^*), 0, \dots, 0].$$

Else

Solve for the point $\tilde{\mathbf{A}}$ where coefficient of learner d first touches 0.

$N = N - 1$; Remove the learner d from A ; Add d into I .

$$\mathbf{A} = \tilde{\mathbf{A}}.$$

End For

Theorem 5. *If there exist a unique uniform learning curve in the parameter space (curve along which active learners have the same gradient), then the solutions of L_1 -Boost.turn algorithm are identical to L_1 -Boost at the turning points of its solution path.*

Proof. See appendix for proof. □

The speed of this algorithm depends mainly on the efficiency of the nonlinear equation solver. We implement the algorithm on a toy example using the MATLAB universal nonlinear equation solver, and find that its speed and solution curve are both similar to ε -Boost. However, the results are not shown in the paper because we would like to compare it with L_1 -Boost, rather than ε -Boost. Also, we will examine the condition for the assumption made in the Theorem 5 to hold. This will be done in future work.

5 Experiment

In this section we experiment on a toy example to: first, show the margins achieved by ε -Adaboost; second, compare the generalization performance of ε -Adaboost and the original Adaboost with or without irrelevant noisy features.

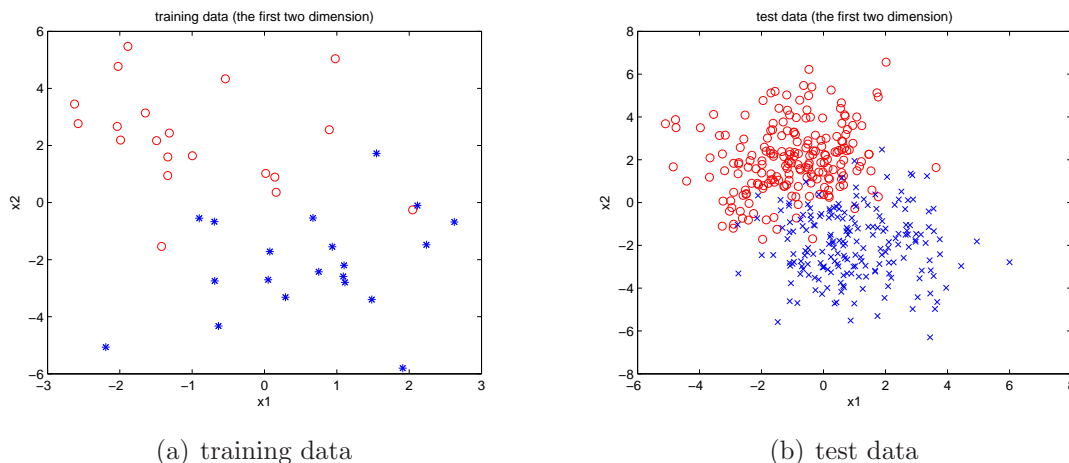


Figure 1: The first two discriminative features of training data and test data

We create two sets of artificial data sets. There are 40 examples in each data set, 20 examples from each class. For the first data sets, there are two features x_1 and x_2 . Examples in each class are drawn from two dimensional Gaussian distributions. As shown in Fig.1(a): positive examples are drawn from distribution $\mathcal{N}([-1, 2]^T, \sigma^2 \mathbf{I})$, whereas negative examples are drawn from distribution $\mathcal{N}([1, -2]^T, \sigma^2 \mathbf{I})$. (We set the variance σ^2 to 1.5). In order to test the effect of different algorithms on irrelevant features, we construct the second data set by adding 3 irrelevant features to the first data set. These irrelevant features are independent gaussian noises with zero mean and variance $\sigma^2 = 1.5$. The testing data sets containing 400 examples are generated in the same way, with the first two dimensions shown in Fig.1(b).

We use both ε -Boost ($\varepsilon = 0.01$) and Adaboost to train data set 1 and data set 2 respectively. The weak learners are simply the stumps of each feature. Obviously a weak learner will be redundant if there exists another learner that yields the same dichotomies on all examples. So for each feature, we first sort the training examples with respect to this feature, then define the thresholds of the stumps to be the average feature value of

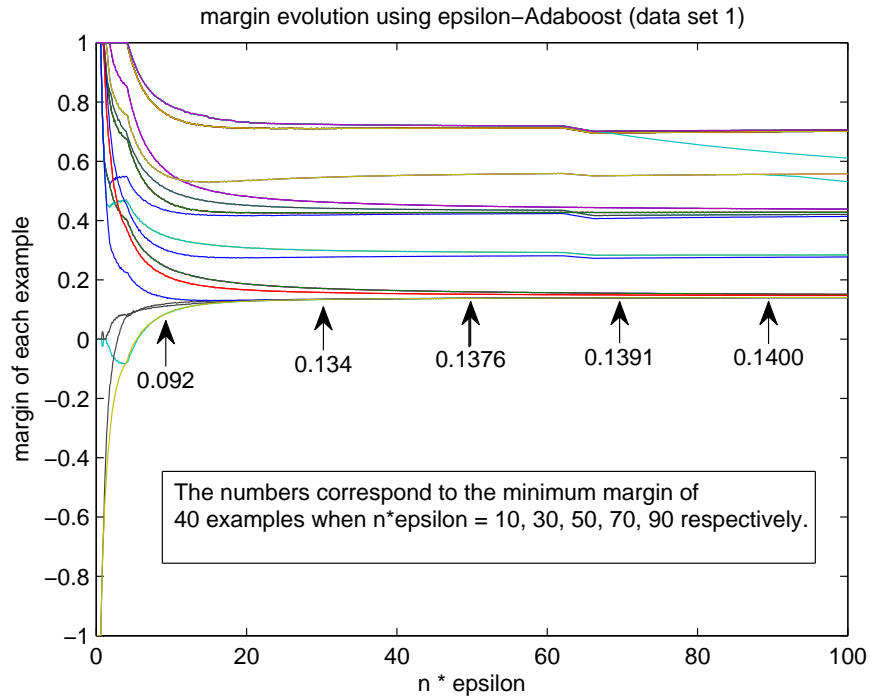


Figure 2: Data set 1: margins of 40 examples during training using ϵ -Adaboost

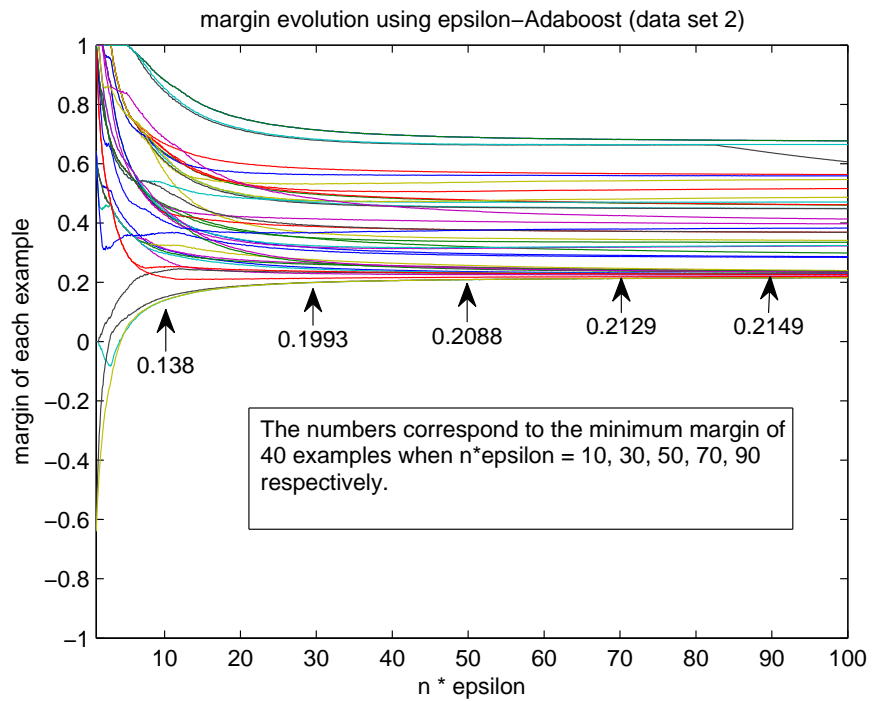


Figure 3: Data set 2: margins of 40 examples during training using ϵ -Adaboost

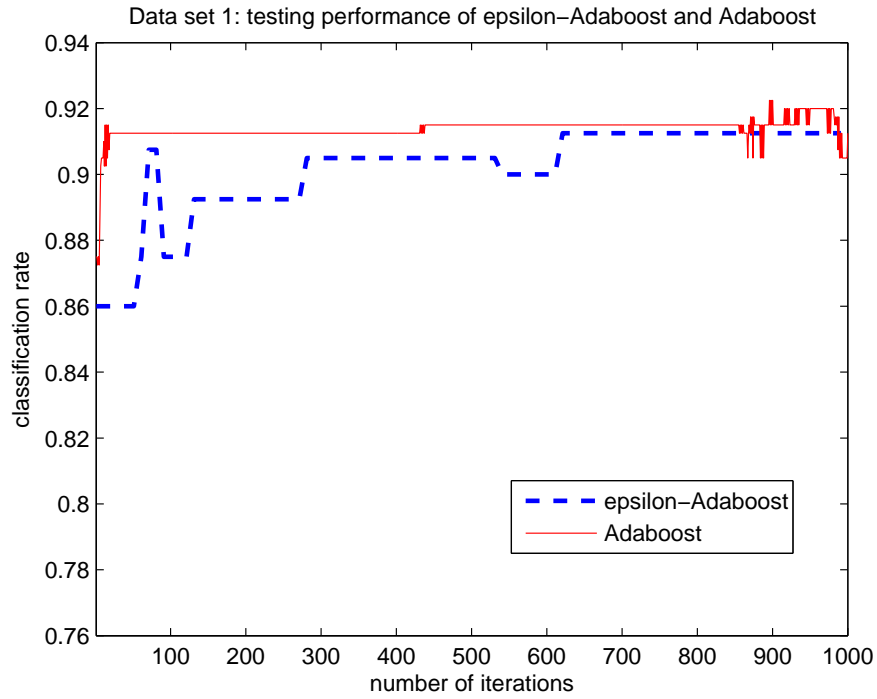


Figure 4: Data set 1: classification accuracy using epsilon-Adaboost and Adaboost

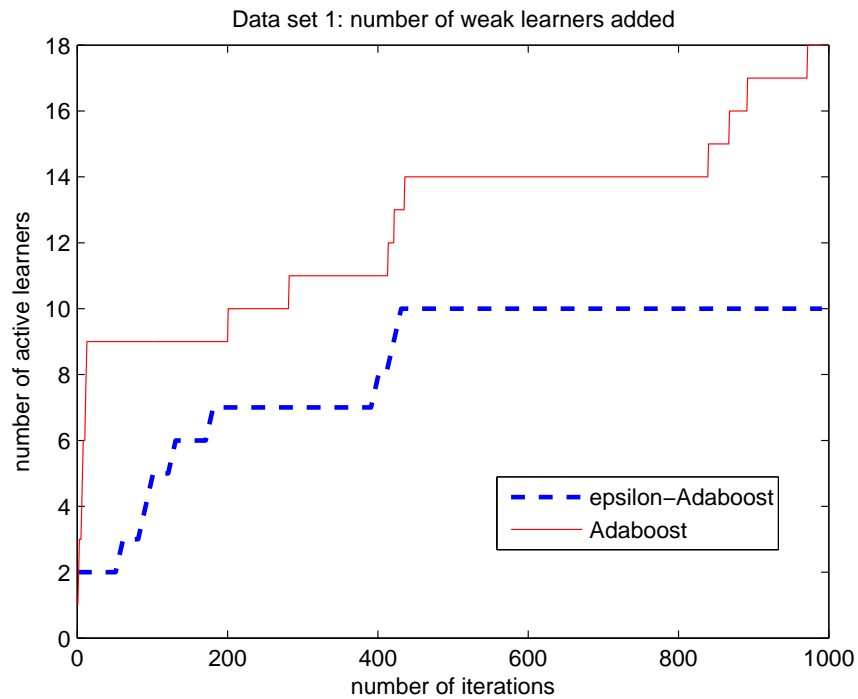


Figure 5: Data set 1: number of weak learners used by epsilon-Adaboost and Adaboost

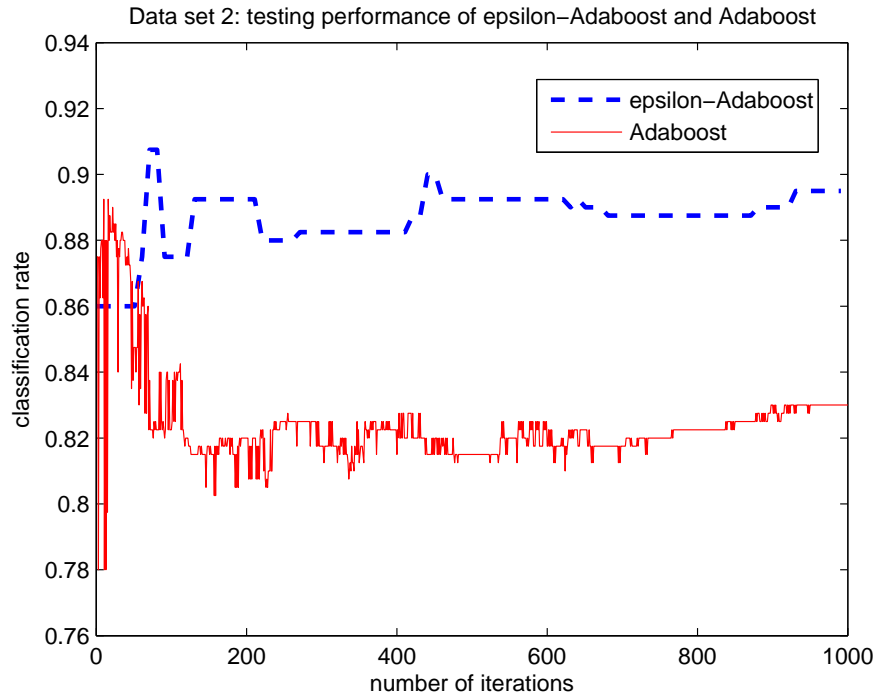


Figure 6: Data set 2: classification accuracy using epsilon-Adaboost and Adaboost

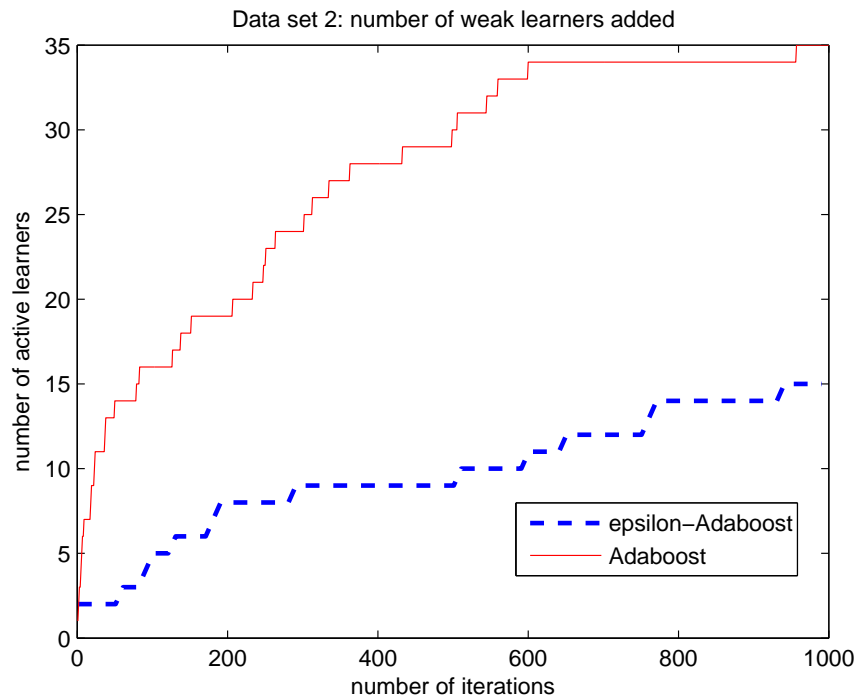


Figure 7: Data set 2: number of weak learners used by epsilon-Adaboost and Adaboost

adjacent examples. This produces $2 \times (40 - 1) = 78$ weak learners for a single feature (by reversing the sign of stump we double). Two learners which always classify positive or negative are added as well.

The margin of each example is recorded for every 100 iterations for ε -Boost. The margins within 10,000 iterations are shown in Fig.2 and Fig.3. In both cases, the minimum margin appears to be increasing similarly to the bound in Theorem 2. Notice that the achievable margin for data set 2 is larger than data set 1 because the weak learner space is richer than the first due to more features.

Then we examine the generalization performances of ε -Adaboost and Adaboost on both data sets. For data set 1, we expect both algorithms to perform well. For data set 2, since the data contains irrelevant noisy features, the algorithm has to be more careful when selecting weak learners. So we expect ε -Adaboost to outperform Adaboost because of its potential to provide a sparser model. As shown in Fig.4 , Adaboost does a fairly good job for data set 1. The prediction rate is greatly increased during the first few steps, and is resistant to overfitting for a long duration, although the solutions at the end become quite unstable. ε -Adaboost, on the other hand, improves more gently in the beginning, but the solution it achieves later is comparable to Adaboost. We then plot the number of different weak learners added by the two algorithms in Fig.5. We observe that ε -Adaboost uses fewer weak learners than Adaboost when achieving similar generalization performance. This might be explained by the similarity of ε -Adaboost to L_1 -Adaboost, which puts L_1 regularization on the coefficients of the learners.

For data set 2, Adaboost is quite unstable in the beginning and overfits after about 100 iterations, as shown in Fig.6. This might be explained by its rapid addition of new weak learners, including those of irrelevant features. ε -Adaboost seems to be more robust to irrelevant features. The number of weak learners it uses, as shown in Fig.7, is only slightly increased from data set 1. Although the prediction performance drops a little bit, it is still more reliable comparing with Adaboost.

6 Conclusion

In this paper we mainly discuss about L_1 -Boost and ε -Boost algorithms for binary classification. The major contributions are: first, we have proved that both ε -Adaboost and L_1 -Adaboost are margin maximizers, and give a quantitative bound on how they achieve the maximum margin θ^* ; second, we have derived a new algorithm called L_1 -Boost.turn, which could potentially speed up L_1 -Boost; third, we have experimentally shown that because the sparser solution it provides, ε -Boost and L_1 -Boost may be better boosting algorithms to use if the data contains irrelevant features. Future work includes: 1) a closer investigation of the relationship between L_1 -Boost and ε -Boost; 2) implementation of L_1 -Boost to provide ground truth for L_1 -Boost.turn algorithm; 3) examine the “unique uniform learning curve” assumption made in the proof of L_1 -Boost.turn; 4) experiments using different algorithms on real data sets.

A Appendix: Proof

A.1 Lemma [2.1]

Proof. Assume the j^{th} hypothesis is selected at the $(n+1)^{\text{th}}$ update, then

$$\begin{aligned}\mathbf{B}((n+1)\varepsilon) &= \mathbf{B}(n\varepsilon) + (0, 0, \dots, \varepsilon, \dots, 0) \\ &= (\beta_1(n\varepsilon), \beta_2(n\varepsilon), \dots, \beta_j(n\varepsilon) + \varepsilon, \dots, \beta_{|\mathcal{H}|}(n\varepsilon))\end{aligned}$$

$\mathcal{L}(\mathbf{B}(n\varepsilon)) = \sum_{i=1}^m e^{-y_i H_n(\mathbf{x}_i)}$ is the loss at the n^{th} step. Using Taylor Series Expansion, we obtain:

$$\begin{aligned}\mathcal{L}(\mathbf{B}((n+1)\varepsilon)) &= \mathcal{L}(\mathbf{B}(n\varepsilon)) + \varepsilon \frac{\partial \mathcal{L}(\mathbf{B}(n\varepsilon))}{\partial \beta_j} + \frac{\varepsilon^2}{2} \frac{\partial^2 \mathcal{L}(\mathbf{B}(n\varepsilon))}{\partial \beta_j^2} + \dots \\ &= \mathcal{L}(\mathbf{B}(n\varepsilon)) + \sum_{k=1}^{\infty} \frac{C(k)}{k!} \varepsilon^k\end{aligned}$$

where

$$\begin{aligned}C(k) &= \frac{\partial^k \mathcal{L}(\mathbf{B}(n\varepsilon))}{\partial \beta_j^k} \\ &= \sum_{i=1}^m (-y_i h_j(\mathbf{x}_i))^k e^{-y_i H_n(\mathbf{x}_i)} \\ &= \begin{cases} C(1) \leq -\theta^* \mathcal{L}(\mathbf{B}(n\varepsilon)) & \text{if } k \text{ is odd} \\ \mathcal{L}(\mathbf{B}(n\varepsilon)) & \text{if } k \text{ is even} \end{cases}\end{aligned}$$

Because

$$\begin{aligned}C(1) &= \sum_{i=1}^m -y_i h_j(\mathbf{x}_i) e^{-y_i H_n(\mathbf{x}_i)} \\ &= -\sum_{i=1}^m y_i h_j(\mathbf{x}_i) \frac{e^{-y_i H_n(\mathbf{x}_i)}}{\sum_{i=1}^m e^{-y_i H_n(\mathbf{x}_i)}} \sum_{i=1}^m e^{-y_i H_n(\mathbf{x}_i)} \\ &= -\left(\sum_{i=1}^m y_i h_j(\mathbf{x}_i) \omega_i\right) \mathcal{L}(\mathbf{B}(n\varepsilon)) \\ &\leq -\theta^* \mathcal{L}(\mathbf{B}(n\varepsilon))\end{aligned}$$

Therefore,

$$\begin{aligned}\mathcal{L}(\mathbf{B}((n+1)\varepsilon)) &\leq \mathcal{L}(\mathbf{B}(n\varepsilon)) \left(1 - \theta^* \varepsilon + \frac{\varepsilon^2}{2!} - \frac{\theta^* \varepsilon^3}{3!} + \frac{\varepsilon^4}{4!} - \frac{\theta^* \varepsilon^5}{5!} + \frac{\varepsilon^6}{6!} - \dots\right) \\ &= \mathcal{L}(\mathbf{B}(n\varepsilon)) \left(1 - \theta^* \varepsilon + \varepsilon^2 - \theta^* \varepsilon^3 + \varepsilon^4 - \theta^* \varepsilon^5 + \varepsilon^6 - \dots\right) \\ &\quad - \mathcal{L}(\mathbf{B}(n\varepsilon)) \left(\frac{\varepsilon^2}{2} - \frac{5\theta^* \varepsilon^3}{6} + \frac{23\varepsilon^4}{24} - \frac{119\theta^* \varepsilon^5}{120} + \frac{719\varepsilon^6}{720} - \dots\right) \\ (\text{assume } \varepsilon < 0.5) &\leq \mathcal{L}(\mathbf{B}(n\varepsilon)) \left(1 - \theta^* \varepsilon + \varepsilon^2 - \theta^* \varepsilon^3 + \varepsilon^4 - \theta^* \varepsilon^5 + \varepsilon^6 - \dots\right) \\ &= \mathcal{L}(\mathbf{B}(n\varepsilon)) \left(1 - \theta^* (\varepsilon + \varepsilon^3 + \varepsilon^5 + \dots) + (\varepsilon^2 + \varepsilon^4 + \varepsilon^6 + \dots)\right) \\ &= \mathcal{L}(\mathbf{B}(n\varepsilon)) \frac{1 - \theta^* \varepsilon}{1 - \varepsilon^2}\end{aligned}$$

Because $\mathcal{L}(\mathbf{B}(0)) = m$,

$$\mathcal{L}(\mathbf{B}(n\varepsilon)) \leq \left(\frac{1 - \theta^* \varepsilon}{1 - \varepsilon^2}\right)^n \mathcal{L}(\mathbf{B}(0)) = \left(\frac{1 - \theta^* \varepsilon}{1 - \varepsilon^2}\right)^n m \quad (10)$$

□

A.2 Lemma [3.1]

Proof. Suppose that at the $(N_1 + 1)^{th}$ step of ε -Logit, the minimum of $y_i H_{B_{N_1}(\mathbf{x}_i)}$ of the m examples is at the first time larger than 0. That is, $\min_{i=1}^m y_i H_{B_n}(\mathbf{x}_i) < 0$, for $n = 1, 2, \dots, N_1$, and $\min_{i=1}^m y_i H_{B_{N_1+1}}(\mathbf{x}_i) > 0$. In the proof we show that each of the N_1 steps reduces the objective function by at least $\frac{1}{2}\theta^*\varepsilon + \mathcal{O}(\varepsilon^2)$, then it follows naturally that $N_1 \leq \frac{m \ln 2 - \ln 2}{\frac{1}{2}\theta^*\varepsilon} < \frac{2m \ln 2}{\theta^*\varepsilon}$. Therefore, ε -Logit achieves margin at least 0 after $\frac{2m \ln 2}{\theta^*\varepsilon}$ steps.

What is left is to show that each of the N_1 steps reduces the objective function by at least $\frac{1}{2}\theta^*\varepsilon + \mathcal{O}(\varepsilon^2)$. At the n^{th} step ($n = 1, 2, \dots, N_1$)

$$\begin{aligned} \frac{\partial \mathcal{L}(\mathbf{B}(n\varepsilon))}{\partial \beta_j} &= \sum_{i=1}^m -y_i h_j(\mathbf{x}_i) \frac{e^{-y_i H_n(\mathbf{x}_i)}}{1 + e^{-y_i H_n(\mathbf{x}_i)}} \\ &= - \sum_{i=1}^m y_i h_j(\mathbf{x}_i) \frac{\frac{e^{-y_i H_n(\mathbf{x}_i)}}{1 + e^{-y_i H_n(\mathbf{x}_i)}}}{\sum_{i=1}^m \frac{e^{-y_i H_n(\mathbf{x}_i)}}{1 + e^{-y_i H_n(\mathbf{x}_i)}}} \sum_{i=1}^m \frac{e^{-y_i H_n(\mathbf{x}_i)}}{1 + e^{-y_i H_n(\mathbf{x}_i)}} \\ &= - \left(\sum_{i=1}^m y_i h_j(\mathbf{x}_i) \omega_i \right) \sum_{i=1}^m \frac{e^{-y_i H_n(\mathbf{x}_i)}}{1 + e^{-y_i H_n(\mathbf{x}_i)}} \end{aligned}$$

Because $\frac{e^{-y_i H_n(\mathbf{x}_i)}}{1 + e^{-y_i H_n(\mathbf{x}_i)}} > \frac{1}{2}$ for any inconsistent point, and $\sum_{i=1}^m y_i h_j(\mathbf{x}_i) \omega_i \geq \theta^*$, we have $\sum_{i=1}^m \frac{e^{-y_i H_n(\mathbf{x}_i)}}{1 + e^{-y_i H_n(\mathbf{x}_i)}} > \frac{1}{2}$. So $\frac{\partial \mathcal{L}(\mathbf{B}(n\varepsilon))}{\partial \beta_j} < -\frac{1}{2}\theta^*$. Thus $\mathcal{L}(\mathbf{B}((n+1)\varepsilon)) = \mathcal{L}(\mathbf{B}(n\varepsilon)) + \frac{\partial \mathcal{L}(\mathbf{B}(n\varepsilon))}{\partial \beta_j} \varepsilon + \mathcal{O}(\varepsilon^2) \leq \mathcal{L}(\mathbf{B}(n\varepsilon)) - \frac{1}{2}\theta^*\varepsilon + \mathcal{O}(\varepsilon^2)$. □

A.3 Lemma [3.2]

Proof. After achieving consistency of the data, i.e., $y_i H_n(\mathbf{x}_i) > 0$ for $i = 1, \dots, m$, the cost function and its negative derivative is related by: $\frac{e^{-y_i H_n(\mathbf{x}_i)}}{1 + e^{-y_i H_n(\mathbf{x}_i)}} \geq C^* \ln(1 + e^{-y_i H_n(\mathbf{x}_i)})$, where $C^* = \frac{\frac{1}{2}}{\ln 2} = \frac{1}{2 \ln 2}$.

$$\begin{aligned} \frac{\partial \mathcal{L}(\mathbf{B}(n\varepsilon))}{\partial \beta_j} &= - \left(\sum_{i=1}^m y_i h_j(\mathbf{x}_i) \omega_i \right) \sum_{i=1}^m \frac{e^{-y_i H_n(\mathbf{x}_i)}}{1 + e^{-y_i H_n(\mathbf{x}_i)}} \\ &\leq - \left(\sum_{i=1}^m y_i h_j(\mathbf{x}_i) \omega_i \right) C^* \sum_{i=1}^m \ln(1 + e^{-y_i H_n(\mathbf{x}_i)}) \\ &\leq -C^* \theta^* \mathcal{L}(\mathbf{B}(n\varepsilon)) \end{aligned}$$

Then, we can conduct proof similar to the one in Lemma [2.1]. □

A.4 Theorem 4

Proof. Prove by contradiction, suppose $\mathbf{A}(r) = (\alpha_1, \alpha_2 \dots \alpha_{|\mathcal{H}|})$ is solving problem (2) and we have i, j such that $i \neq j$, $\alpha_i, \alpha_j \neq 0$ and

$$-\frac{\partial \mathcal{L}(\mathbf{A}(r))}{\partial \alpha_i} \neq -\frac{\partial \mathcal{L}(\mathbf{A}(r))}{\partial \alpha_j} \quad (11)$$

without loss of generality, we can assume

$$\frac{\partial \mathcal{L}(\mathbf{A}(r))}{\partial \alpha_i} < \frac{\partial \mathcal{L}(\mathbf{A}(r))}{\partial \alpha_j} \quad (12)$$

The idea to jiggle α_i and α_j a little bit, so that they still satisfy the same L1 constraint but the objective function becomes smaller. To do this, let $\alpha'_i = \alpha_i + \varepsilon$, $\alpha'_j = \alpha_j - \varepsilon$. As long as $0 < \varepsilon < \min\{|\alpha_i|, |\alpha_j|\}$ (that's where the condition $\alpha_i, \alpha_j \neq 0$ come into play), such jiggling will preserve the L1 norm: $|\alpha'_i| + |\alpha'_j| = |\alpha_i| + |\alpha_j|$. But after such transformation, let's compare the value of loss function on $\mathbf{A}'(r) = (\dots, \alpha'_i, \dots, \alpha'_j, \dots)$ and $\mathbf{A}(r) = (\dots, \alpha_i, \dots, \alpha_j, \dots)$:

$$\begin{aligned} \mathcal{L}(\mathbf{A}'(r)) - \mathcal{L}(\mathbf{A}(r)) &= \mathcal{L}((\dots, \alpha'_i, \dots, \alpha'_j, \dots)) - \mathcal{L}((\dots, \alpha_i, \dots, \alpha_j, \dots)) \\ &= \mathcal{L}((\dots, \alpha_i + \varepsilon, \dots, \alpha_j - \varepsilon, \dots)) - \mathcal{L}((\dots, \alpha_i, \dots, \alpha_j, \dots)) \\ &= \varepsilon \frac{\partial \mathcal{L}(\mathbf{A}(r))}{\partial \alpha_i} + (-\varepsilon) \frac{\partial \mathcal{L}(\mathbf{A}(r))}{\partial \alpha_j} + O(\varepsilon^2) \\ &= \left(\frac{\partial \mathcal{L}(\mathbf{A}(r))}{\partial \alpha_i} - \frac{\partial \mathcal{L}(\mathbf{A}(r))}{\partial \alpha_j} \right) \varepsilon + O(\varepsilon^2) \end{aligned} \quad (13)$$

since $\frac{\partial \mathcal{L}(\mathbf{A}(r))}{\partial \alpha_i} - \frac{\partial \mathcal{L}(\mathbf{A}(r))}{\partial \alpha_j} < 0$, we can always find a small $\varepsilon > 0$ such that the above formula is negative, this contradicts with the optimality of $\mathbf{A}(r)$. Therefore the assumption of existing two non-zero α satisfying (11) is invalid. \square

A.5 Theorem 5

Proof. Under the assumption of the uniqueness of the uniform learning curve, the solution path of L_1 -Boost is this curve. For L_1 -Boost, say at the N^{th} stage, there are N active learners with nonzero coefficients. 1) Assuming the N learners stay active, their coefficients will travel along the uniform learning curve in the N -dimensional parameter space, until a new learner is the **first** among all inactive learners to have the same gradient with the N learners. This learner then becomes the $(N + 1)^{\text{th}}$ learner. In the algorithm L_1 -Boost.turn, for **each** inactive learner, we solve for the points on the curve where the N active learners have the same gradient with it. Then we select the point which yield **least** increase of the L_1 norm of the coefficient vector. Since the L_1 norm of the coefficient vector is monotone increasing along the curve, the selection criterion guarantees the selected learner to be the first along the curve. Therefore it is identical to L_1 -Boost solution at this point. 2) If any of the N learners becomes inactive (coefficient becomes 0) before the $(N + 1)^{\text{th}}$ learner is found. Then L_1 -Boost will treat this learner as other inactive learners as soon as its coefficient is 0, and instead travel along the uniform learning curve in the rest $(N - 1)$ -dimensional space. The back-tracking procedure in the L_1 -Boost.turn algorithm ensures this. Therefore we prove the theorem. \square

B Reference

1. R. E. Schapire, Y. Freund, P. L. Bartlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651-1686, October 1998.
2. S. Rosset, J. Zhu, and T. Hastie. Margin maximizing loss functions. In *Advances in Neural Information Processing Systems (NIPS) 15*. MIT Press, 2003.
3. T. Hastie, J. Taylor, R. Tibshirani, and G. Walther. Forward stagewise regression and the monotone lasso. *Electronic Journal of Statistics* 2007, Vol. 1, 1-29; doi:10.1214/07-EJS004.
4. M. K Warmuth, J. Liao, and G. Rasch. Totally corrective boosting algorithms that maximize the margin. *Proceedings of the 23rd international conference on Machine learning*, pages: 1001 - 1008, 2006 ISBN:1-59593-383-2.
5. B. Efron, T. Hastie, I. Johnstone and R. Tibshirani. Least angle regression. *Annual Statistics*, Volume 32, Number 2 (2004), 407-499.
6. L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithms as gradient descent. In *Neural Information Processing Systems*, pages 512-518. MIT Press, 2000.
7. S. Rosset, J. Zhu, T. Hastie, Boosting as a Regularized Path to a Maximum Margin Classifier, *Journal of Machine Learning Research* 5 (2004) 941-973.
8. J. H. Friedman, *Greedy Function Approximation: A Gradient Boosting Machine*, 1999.
9. J. Kivinen, M. K. Warmuth, *Boosting as Entropy Projection*, Annual Workshop on Computational Learning Theory, 1999.