Gradient Descent Boosting: Convergence and Algorithm

Haohua Wan*

May 12, 2017

Abstract

Boosting algorithm has seen many applications in recent years. An early and effective boosting algorithm was developed in Freund and Schapire (1995) called Adaboost. Since then many different boosting algorithms have been developed and gradient descent boosting algorithms might be the most popular among them. This report mainly focuses on three early works about gradient descent boosting. Mason and Frean (1999) provides some quantitative analysis of the convergence property of Anyboost it develops, while Friedman (2001) and Friedman (2002) focus more on the development of the algorithm itself, where they propose the so called greedy gradient boosting algorithm. In the final part of this report, we give some numerical study about the gradient descent boosting algorithm in Friedman (2001) in regression and classification and compare it with several other popular learning models.

Keywords: boosting, gradient descent

1 Introduction

Boosting is a learning method that has been rapidly developing in the last thirty years. The motivation and basic idea of boosting is to combine weak learners, which are only sightly better than random guessing, to reduce the prediction or classification errors. To be more specific, boosting is to repeatedly apply weak learning algorithms to the data to obtain a sequence of weaker learners and then combine them by some weight. A classic boosting algorithm called AdaBoost was developed by Freund and Schapire (1995) and shown to perform significantly and uniformly better than bagging with simple classifiers in Freund and Schapire (1996). This paper also discusses about possible reasons about better performance of boosting method: the first has to do with reducing hypothesis errors by combing many hypotheses; and the second has to do with variance reduction. A lot of boosting algorithms have been developed since then and gradient descent technique is widely used in many of them: gradient descent boosting. In this report, I will review three early works in this field: one is Mason and Frean (1999), which proves convergence of functional-gradient-descent algorithms (Anyboost); the other two are Friedman (2001) and Friedman (2002), which develops a very popular class of gradient boosting models (GBM) with deterministic gradient boosting and stochastic gradient boosting. The reason that I choose these three papers is that: Mason and Frean (1999) is more of theoretical interest, as it is an early work about some analytical property of gradient boosting

*Industrial and Enterprise Systems Engineering Department, University of Illinois at Urbana-Champaign, Urbana, IL 61801; email: hwan3@illinois.edu.

method, while the algorithm it develops is not as popular as some other boosting algorithms; Friedman (2001) and Friedman (2002) is more of practical value, as they develop a much more popular and much more "greedy" gradient boosting algorithm that is widely used in practice.

This report is organized as follows: Section 2 summarizes the convergence results of Anyboost algorithm developed in Mason and Frean (1999) and discusses some limitation of this paper; Section 3 summarizes the "greedy" gradient descent boosting algorithm developed in Friedman (2001) and Friedman (2002) and discusses their limitations; Section 4 describes my numerical study in which greedy descent boosting algorithm developed in Friedman (2001) is applied in two cases: regression and classification; Section 5 discusses some possible directions for future research.

2 Convergence of Anyboost (Mason and Frean (1999))

Mason and Frean (1999) developed a general class of gradient descent boosting algorithms which combine weak learners in an inner product space to minimize the loss function. The model setting is very similar to the one we discussed in our class. Let $X \times Y$ be the sample space where X is the feature space (typically $\mathbb{R}^n$) and Y is the space of labels (typically $\{-1, 1\}$). $\mathcal{P} = \mathcal{P}(X \times Y)$ is the set of all probability measures on $X \times Y$. $\mathcal{F}$ is a family of classifier f (weak learners) such that $f : X \to Y$. Let $lin(\mathcal{F})$ be the set of all linear combinations of elements in $\mathcal{F}$ with an inner product $\langle , \rangle$. The loss function is denoted by $L : lin(\mathcal{F}) \to \mathbb{R}$. And the objective is to find $F \in lin(\mathcal{F})$ such that $L(F)$ is minimized, i.e., solve $F^* = \arg\min_{F \in lin(\mathcal{F})} L(F)$.

The basic idea of Anyboost is that given $F \in lin(\mathcal{F})$, we want to find a $f \in \mathcal{F}$ such that $L(F + \epsilon f) - L(F)$ is negative, i.e., $L(F + \epsilon f)$ decreases. Therefore, a natural choice of f would be $f = -\nabla(L(F))$, but in general it is very likely that $-\nabla L(F) \notin \mathcal{F}$. By Taylor's expansion, we have $L(F+\epsilon f) = L(F)+\epsilon\langle \nabla L(F), f\rangle + o(\epsilon)$, and thus in order to minimize $L(F+\epsilon f)-L(F)$, a good choice of f would be the one that maximizes $-\langle \nabla L(F), f\rangle$. And these ideas motivate Anyboost algorithm, which is described in Table 1.

Algorithm 1: Anyboost

Input: a labeled training data set, a routine that $R(F)$ that returns $f \in \mathcal{F}$ that gives a large $-\langle \nabla L(F), f\rangle$
Initialization: $F_0 = 0, \boldsymbol{f}_1 = 0$
For $i = 1, 2, , \cdots, T$:
Let $f_{t+1} = R(F_t)$
if $-\langle \nabla L(F_t), f_{t+1}\rangle \leq 0$, return F_t
Choose w_{t+1} and let $F_{t+1} = F_t + w_{t+1} f_{t+1}$
End For
Return F_{T+1}

Table 1: Anyboost Algorithm

Note that in Table 1 the inner product $\langle , \rangle$ and the step size w_t are not specified, and thus they can be adapted for specific application. And the algorithm terminates when $-\langle \nabla L(F_t), f_{t+1}\rangle \leq 0$, which means that combining more weak learners no longer reduces the loss function. Mason and Frean (1999) also points out that many boosting algorithms, such as Adaboost (Freund and Schapire (1995)) and Logitboost(Friedman and Tibshirani (2000)), can be viewed as a special case of Anyboost algorithm.
An import result in Mason and Frean (1999)) is the following theorem, which states that Anyboost

algorithm either stops after finite iterations or converges to some finite value.

THEOREM 1. *Let $L : lin(\mathbf{F}) \to \mathbb{R}$ be any lower bounded, $\beta-$smooth loss function, i.e., for any $F, F' \in lin(\mathbf{F})$, $\|\nabla L(F) - \nabla L(F')\| \leq \beta \|F - F'\|$. Let $F_0, F_1, \cdots$ be the sequence of classifiers generated by Anyboost algorithm with step size*

$$w_{t+1} = -\frac{\langle \nabla L(F_t), f_{t+1} \rangle}{\beta \|f_{t+1}\|^2}, \tag{1}$$

then Anyboost algorithm either stops at iteration T with $-\langle \nabla L(F_T), f_{T+1} \rangle \leq 0$ or $L(F_t)$ converges to some finite value C^* with $\lim_{t\to\infty} -\langle \nabla L(F_t), f_{t+1} \rangle = 0$.

Another important result in Mason and Frean (1999)) states that any accumulation point of the sequence F_t generated by Anyboost algorithm can be a global minimum of the loss function if we can always find the best f_t.

THEOREM 2. *Let $L : lin(\mathbf{F}) \to \mathbb{R}$ be a convex loss function with properties in Theorem 1, and let $F_0, F_1, \cdots$ be the sequence of classifiers generated by Anyboost algorithm with step size* (1)*. Assume the class of weak learners $\mathcal{F}$ is negation closed (if $f \in \mathcal{F}$, then $-f \in \mathcal{F}$) and that on each iteration of Anyboost algorithm we can always find f_{t+1} maximizing $-\langle \nabla L(F_t), f_{t+1} \rangle$. Then any accumulation point F of the sequence F_t generated by Anyboost algorithm satisfies*

$$\sup_{f\in\mathcal{F}} -\langle \nabla L(F), f \rangle = 0, \quad \textit{and} \quad L(F) = \inf_{F' \in lin(\mathbf{F})} L(F')$$

Although Theorem 1 and 2 give convergence results for a large set of boosting algorithms, which is very useful in the algorithms' performance analysis, it also has several limitations. First, the routine $R(F)$ that returns $f \in \mathcal{F}$ with a large value of $-\langle \nabla L(F), f \rangle$ in Anyboost algorithm may be hard to obtain. In addition, at each iteration, it does not extract as much information as many other boosting algorithms do, like the Greedy Gradient Descent Boosting in Friedman (2001). And this might be why it is not used so widely in practice. Second, in Theorem 1, it requires the loss function to be $\beta-$smooth, which may be too restricted in many applications. Third, the convergence result in Theorem 1 is only an existence result, which does not specify the exact value of C^*. Moreover, from Theorem 1, we do not know the consistency of Anyboost algorithm, i.e., what is the difference between C^* and the lower bound of the loss function. And in many applications, we will be most intereted in this difference.

3 Gradient Boosted Models (Friedman (2001), Friedman (2002))

Gradient Boosted Models (GBM), which was developed in Friedman (2001), is now widely used in practice. The model setting is very similar to that in Section 2. Let $X \times Y$ be the sample space where X is the feature space (typically $\mathbb{R}^n$) and Y is the space of labels (typically $\{-1, 1\}$). And let $\{\boldsymbol{x}_i, y_i\}_1^N$ be the training sample. Let $\mathcal{P} = \mathcal{P}(X \times Y)$ be the set of all probability measures on $X \times Y$. The classifier F has an additive expansion of the form

$$F(\boldsymbol{x}; \{\beta_m, \boldsymbol{a}_m\}_1^M) = \sum_{m=1}^{M} \beta_m h(\boldsymbol{x}; \boldsymbol{a}_m), \tag{2}$$

where $h(\boldsymbol{x};\boldsymbol{a}_m)$ is usually a parameterized weak learner. And the loss function is denoted by $L(y,F)$. The objective is to solve the following problem

$$\{\beta_m,\boldsymbol{a}_m\}_1^M = \arg min_{\{\beta'_m,\boldsymbol{a}'_m\}_1^M} \sum_{i=1}^N L\left(y_i, \sum_{m=1}^M \beta'_m h(\boldsymbol{x}_i;\boldsymbol{a}'_m)\right) \tag{3}$$

which is very much like the ERM algorithm. And if (3) is infeasible, we can try a greedy stagewise approach (this is why this model is also called "greedy gradient descent algorithm" and in the following discussion, we will simple denote it by gradient descent model, or GBM). For $m = 1, 2, \cdots, M$, solve

$$(\beta_m,\boldsymbol{a}_m) = \arg min_{\beta,\boldsymbol{a}} \sum_{i=1}^N L(y_i, F_{m-1}(\boldsymbol{x}_i) + \beta h(\boldsymbol{x}_i;\boldsymbol{a})) \tag{4}$$

and then update

$$F_m(\boldsymbol{x}) = F_{m-1}(\boldsymbol{x}) + \beta_m h(\boldsymbol{x};\boldsymbol{a}_m). \tag{5}$$

If (4) is still hard to solve, we can implement a steepest descent routine by first getting the data based gradient

$$-g_m(\boldsymbol{x}_i) = -\left[\frac{\partial L(y_i, F(\boldsymbol{x}_i))}{\partial F(\boldsymbol{x}_i)}\right]_{F(\boldsymbol{x})=F_{m-1}(\boldsymbol{x})}. \tag{6}$$

(6) is defined only at the data points $\{\boldsymbol{x}_i\}_1^N$, and in order to generalize it to other $\boldsymbol{x}-$values, we can choose $h(\boldsymbol{x};\boldsymbol{a})$ that is parallel to or has a high correlation with $-\boldsymbol{g}_m$, i.e., solve

$$\boldsymbol{a}_m = \arg min_{\boldsymbol{a},\beta} \sum_{i=1}^N \left(-g_m(\boldsymbol{x}_i) - \beta h(\boldsymbol{x}_i;\boldsymbol{a})\right)^2. \tag{7}$$

And then we can update

$$F_m(\boldsymbol{x}) = F_{m-1}(\boldsymbol{x}) + \rho_m h(\boldsymbol{x};\boldsymbol{a}), \tag{8}$$

where ρ_m is obtained through line search

$$\rho_m = \arg min_\rho \sum_{i=1}^N L(y_i, F_{m-1}(\boldsymbol{x}) + \rho h(\boldsymbol{x}). \tag{9}$$

By (7), if we have a least-squares algorithm, then we can replace the much more difficult minimization problem (4) by a least-squares minimization problem (7) and a single variable minimization problem (9). And this gives rise to the following steepest-descent algorithm in Table 2.

To reduce overfitting of Algorithm 2, it is an good idea to apply regularization method. Simulation results in Friedman (2001) indicate that that adding a shrinkage term is often better than reducing the number of components, i.e., (8) becomes

$$F_m(\boldsymbol{x}) = F_{m-1}(\boldsymbol{x}) + \nu \cdot \rho_m h(\boldsymbol{x};\boldsymbol{a}), \tag{10}$$

where ν is the shrinkage rate.

Algorithm 2: GBM

Input: a labeled training data set $\{\boldsymbol{x}_i, y_i\}_1^N$

Initialization: $F_0(\boldsymbol{x}) = \arg min_\rho \sum_{i=1}^N L(y_i, \rho)$

For $m = 1, 2, , \cdots, M$:

$$\tilde{y}_i = -\left[\frac{\partial L(y_i, F(\boldsymbol{x}_i))}{\partial F(\boldsymbol{x}_i)}\right]_{F(\boldsymbol{x})=F_{m-1}(\boldsymbol{x})}, i = 1, \cdots, N$$

$$\boldsymbol{a}_m = \arg min_{\boldsymbol{a},\beta} \sum_{i=1}^N (\tilde{y}_i - \beta h(\boldsymbol{x}_i; \boldsymbol{a}))^2$$

$$\rho_m = \arg min_\rho \sum_{i=1}^N L(y_i, F_{m-1}(\boldsymbol{x}_i) + \rho h(\boldsymbol{x}_i)$$

$$F_m(\boldsymbol{x}) = F_{m-1}(\boldsymbol{x}) + \rho_m h(\boldsymbol{x}; \boldsymbol{a})$$

End For

Table 2: Gradient Boost Algorithm

Algorithm 3: Stochastic GBM

Input: a labeled training data set $\{\boldsymbol{x}_i, y_i\}_1^N$

Initialization: $F_0(\boldsymbol{x}) = \arg min_\rho \sum_{i=1}^N L(y_i, \rho)$

For $m = 1, 2, , \cdots, M$:

$$\{\pi(i)\}_1^N = \{i\}^N$$

$$\tilde{y}_{\pi(i)} = -\left[\frac{\partial L(y_i, F(\boldsymbol{x}_{\pi(i)}))}{\partial F(\boldsymbol{x}_{\pi(i)})}\right]_{F(\boldsymbol{x})=F_{m-1}(\boldsymbol{x})}, i = 1, \cdots, \tilde{N}$$

$$\boldsymbol{a}_m = \arg min_{\boldsymbol{a},\beta} \sum_{i=1}^{\tilde{N}} \left(\tilde{y}_{\pi(i)} - \beta h(\boldsymbol{x}_{\pi(i)}; \boldsymbol{a})\right)^2$$

$$\rho_m = \arg min_\rho \sum_{i=1}^{\tilde{N}} L(y_{\pi(i)}, F_{m-1}(\boldsymbol{x}_{\pi(i)}) + \rho h(\boldsymbol{x}_{\pi(i)})$$

$$F_m(\boldsymbol{x}) = F_{m-1}(\boldsymbol{x}) + \rho_m h(\boldsymbol{x}; \boldsymbol{a})$$

End For

Table 3: Stochastic Gradient Boost Algorithm

A stochastic version of Algorithm 2 has been developed in Friedman (2002), in which at each iteration, only a subsample of the training data is used to fit the weak learner and compute the model update. To be more specific, let $\{\pi(i)\}_1^N$ be a random permutation of $\{1, \cdots, N\}$, and thus a random subsample of size $\tilde{N} < N$ is given by $\{y_{\pi(i)}, \boldsymbol{x}_{\pi(i)}\}_1^{\tilde{N}}$. The detailed description of stochastic gradient boosting algorithm is provided in Table 3.

The simulation results in Friedman (2002) shows that stochastic gradient boost algorithm can be much faster and more accurate than deterministic gradient boost algorithm, although the exact reason remains unknown. One possible explanation has to do with variance reduction as stochastic gradient boost algorithm is shown to be most effective with small samples with high capacity weak learners.

Those gradient boosting algorithms developed in Friedman (2001) and Friedman (2002) have become very popular in practice, and many statistical softwares and packages have included them. However, they also have some limitations. First, both algorithms, even the stochastic gradient descent algorithm, contains a large amount of gradient computation, which could be a major concern if the running time becomes an issue. Second, Friedman (2001) and Friedman (2002) provide little analytical result about the performance of these two algorithms, although they indeed implement a lot of simulations. Due to the lack of quantitative analysis, it is hard to analyze the properties of these algorithms under statistical learning framework. For example, we do not know whether these two algorithms are stable or not. And we do not know these two algorithms are consistent or not, either.

4 Numerical Study

In this part, we provide two numerical studies to illustrate the application of Gradient Boost Models in Friedman (2001). The first example is regression, and the second example is classification. In these two examples, I will compare the prediction accuracy and running time of deterministic gradient descent algorithm in Friedman (2001) with other popular learning models.

4.1 Regression Study

In this study, we use housing data collected on residential properties sold in Ames, Iowa between 2006 and 2010. This data set contains 1460 samples, and each sample has 80 features, such as the building class, lot size in square feet, type of utilities available and so on. Some of these features are numerical variables, and others are categorical variables. And we want to build a regression model to predict the sale price.

We will use regression trees based gradient boosted model in Friedman (2001) to do the prediction. In other words, in (2), we let the weak learners be J-node regression tree model, which itself has the additive form

$$h(\boldsymbol{x}; \{b_j, R_j\}_1^J) = \sum_{j=1}^{J} b_j \mathbf{1}(\boldsymbol{x} \in R_j), \tag{11}$$

where $\{R_j\}_1^J$ is the terminal nodes of this regression tree. Hence for a regression tree, (8) becomes

$$F_m(\boldsymbol{x}) = F_{m-1}(\boldsymbol{x}) + \rho_m \sum_{j=1}^{J} b_{jm} \mathbf{1}(\boldsymbol{x} \in R_{jm}), \tag{12}$$

where $\{R_{jm}\}_1^J$ are the terminal nodes at the $m-$th iteration that are obtained by least square and b_{jm} is the corresponding least square coefficients. To be more specific, in Table 2, we solve

$$\arg min_{R'_j, b'_j} \sum_{i=1}^{N} \left(\tilde{y}_i - \sum_{j=1}^{J} b'_j \mathbf{1}(\boldsymbol{x} \in R'_j)) \right)^2 \tag{13}$$

to obtain R_j and b_j at each iteration. And to do regression, we choose the loss function to be $L(y, F) = (y - F)^2/2$. The algorithm is summarized in Table 4.

Algorithm 4: Regression Tree GBM
Input: a labeled training data set $\{\boldsymbol{x}_i, y_i\}_1^N$
Initialization: $F_0(\boldsymbol{x}) = \bar{y}$
For $m = 1, 2, , \cdots, M$:
$\tilde{y}_i = y_i - F_{m-1}(\boldsymbol{x}_i), i = 1, \cdots, N$
$\{R_{jm}\}_1^J = J$-terminal node tree$(\{\tilde{y}_i, \boldsymbol{x}_i\}_1^N)$
$\gamma_{jm} = \text{mean}_{\boldsymbol{x}_i \in R_{jm}}\{y_i - F_{m-1}(\boldsymbol{x}_i)\}, j = 1, \cdots, J$
$F_m(\boldsymbol{x}) = F_{m-1}(\boldsymbol{x}) + \sum_{j=1}^{J} \gamma_{jm} \mathbf{1}(\boldsymbol{x} \in R_{jm})$
End For

Table 4: Regression Tree Gradient Boost Algorithm

The prediction accuracy in this case study is represented by Root-Mean-Squared-Error (RMSE) between the logarithm of the predicted value and the logarithm of the observed sales price. The reason that we are taking logs is that errors in predicting expensive houses will be comparable to errors in predicting cheap houses.

In Table 5, we provide the average time/accuracy, as well as the corresponding standard error of prediction RMSE by running the simulation 5 times on different randomly splitted data (75% training data and 25% test data), by GBM and four other models. Random Forest is an ensemble tree model that is often used to provide a benchmark in prediction or classification. Ridge and Lasso are two basic regularized regression models that are easy to implement. And MLR is the simple Multiple Linear Regression. As for the GBM model, we set shrinkage parameter as 0.05, and number of trees as 300.

Simulation No.	Random Forest	Ridge	Lasso	GBM	MLR
1	0.1387	0.1202	0.1165	0.12990	0.1309
2	0.1568	0.1398	0.1373	0.1448	0.1530
3	0.1414	0.1379	0.1349	0.1281	0.1473
4	0.1373	0.1324	0.1291	0.1206	0.1424
5	0.1323	0.1189	0.1134	0.1185	0.1286
RMSE Mean	0.1413	0.12984	0.12624	0.12838	0.14044
RMSE std	0.0041	0.00440	0.00480	0.00460	0.00470
Running Time/s(mean)	12.4400	0.49000	0.35800	5.10200	0.01600

Table 5: Model Prediction of Five Simulations

From Table 5, it can be seen that:
(1) In terms of prediction error, Lasso and GBM perform best, and from our other simulation results, we believe that GBM is slightly better than Lasso, and also GBM has smaller standard error. Ridge is slightly worse but usually better than Random Forest. As a result, from these simulation results, it can be seen that GBM is indeed better than many other popular learning models, although not very obvious.
(2) In terms of running time, MLR is the fastest, as it is the simplest model; and GBM is the slowest. As it was said before, GBM involves a lot of gradient computation. Moreover, please note that the running time of GBM is about 10 times as that of Ridge and Lasso, and therefore when taking running time into consideration, GBM might not be the best model.

4.2 Classification Study

In this study, we use loan data for all loans issued through the 2007-2015 in Lending Club, which include the current loan status (Current, Late, Fully Paid, etc.) and latest payment information. There are 887383 observations, and each of them contains 75 features. We want to build a binary classification model, and use it to predict the default status of all loans issued in 2016.

We will use classification tree based gradient boosted model in Friedman (2001) to do the classification. First define the loss function to be

$$L(\{y_k, F_k(\boldsymbol{x})\}_1^2) = -\sum_{k=1}^{2} y_k \log p_k(\boldsymbol{x}) \tag{14}$$

where $y_k = \mathbf{1}(class = k)$, and $p_k(\boldsymbol{x}) = \mathbf{P}(y_k = 1|\boldsymbol{x})$. Use symmetric multiple logistic transform

$$F_k(\boldsymbol{x}) = \log p_k(\boldsymbol{x}) - \frac{1}{2}\sum_{l=1}^{2} \log p_l(\boldsymbol{x}) \tag{15}$$

or

$$p_k(\boldsymbol{x}) = \exp(F_k(\boldsymbol{x}))/\sum_{l=1}^{2} \exp(F_l(\boldsymbol{x})). \tag{16}$$

Similar to the development of regression tree based gradient boosting algorithm, the 2-class logistic gradient boosting algorithm is developed and described in Table 6. Note that this 2-class algorithm can be extended to $K-$ class algorithm, but for our illustration purpose, 2-class is enough.

Algorithm 5: 2-class Logistic GBM

Input: a labeled training data set $\{\boldsymbol{x}_i, y_i\}_1^N$
Initialization: $F_{k0}(\boldsymbol{x}) = 0, k = 1, 2$
For $m = 1, 2, , \cdots, M$:
  $p_k(\boldsymbol{x}) = \exp(F_k(\boldsymbol{x}))/\sum_{l=1}^{2} \exp(F_l(\boldsymbol{x})), k = 1, 2$
  For $k = 1$ to 2
    $\hat{y}_{ik} = y_{ik} - p_k(\boldsymbol{x}_i), i = 1, \cdots, N$
    $\{R_{jkm}\}_{j=1}^{J} = J -$ terminal node tree $(\{\tilde{y}_{ik}, \boldsymbol{x}_i\}_1^N)$
    $\gamma_{jkm} = 1/2\frac{\sum_{\boldsymbol{x}\in R_{jkm}} \hat{y}_{ik}}{\sum_{\boldsymbol{x}\in R_{jkm}} |\hat{y}_{ik}|(1-|\hat{y}_{ik}|)}$
    $F_{km}(\boldsymbol{x}) = F_{k,m-1}(\boldsymbol{x}) + \sum_{j=1}^{J} \gamma_{jkm}\mathbf{1}(\boldsymbol{x} \in R_{jkm})$
  End For
End For

Table 6: Classification Tree Gradient Boost Algorithm

In Table 7, we provide our prediction result by GBM and Rpart. (Note that we are using training data in 2007-2015 to predict the default status of each loan in 2016). Rpart is Recursive Partitioning and Regression Trees, which is a variation of tree based model, and it is used as the benchmark in this case. I also tried many other popular models as in the regression study, but due to this huge dataset, many models do not work, such as Logistic Model and Random Forest. Although it may be possible to adapt those models to this huge dataset by using some packages, due to the time limit, I only choose GBM and Rpart models, as well as their average model (use the average of these two models' prediction). In GBM, I set the number of trees to be 30000 and shrinkage to be 0.001. Besides, our code is run on a computer with Processor Intel(R) Core(TM) i7-4790 CPU @3.60GHz 3.60GHz, and RAM 16GB.

From Table 7, it can be seen that GBM performs better than Rpart in an obvious way, and from my experiment, it is highly likely that the number of trees can be increased even further to obtain a better result. However, it is also very obvious that GBM runs very slowly when the number of trees is 30000, and this is also why I do not try even larger number of trees as it will take too much time to run the code. Again, from this classification study, it can be implied that although GBM can be better in many applications, it also requires a lot more computational resources.

Model	Rpart	GBM	Average
Logloss	0.1137995	0.1029014	0.1019476
Running Time	$\approx 15min$	$\approx 13hours$	$\approx 13hours$

Table 7: Model classification Results

5 Future Works

This report mainly summarizes and reviews three papers: Mason and Frean (1999), Friedman (2001) and Friedman (2002), all of which are early works about gradient boosting methods. So it is not surprising that many further works can be done based on those papers, and the following is some direction that I think worth exploring (although some of them might have already be done):

First, in Mason and Frean (1999), as Theorem 1 is only an existence result, it would be good if we can show the convergence rate of Anyboost. Moreover, it would be even better if we can determine the limit value C^* or the difference between C^* and the lower bound of loss function. Besides, we can try to investigate better ways of calculating a large value of $-\langle \nabla L(F), f \rangle$, such as getting the largest value of $-\langle \nabla L(F), f \rangle$ when $f \in \mathcal{F}$.

Second, in Friedman (2001) and Friedman (2002), there is a lack of analytical results. Therefore, many can be done from this analytical perspective. For example, it would be interesting to analyze the stability, generalization ability, and consistency of gradient boosting method (GBM) under statistical learning framework. In addition, the implementation of GBM in Section 4 shows that although GBM performs better than many other popular learning models, its running time is also much longer. Therefore, Friedman (2002) proposed a stochastic version of GBM, which can reduce the running time to some extent (due to the time limit, I did not implement stochastic GBM). And recently, there are many papers about how to further accelerate stochastic gradient method which could be useful in reducing the running time(I remember that some student has presented several very recent papers on this topic in May 10th's presentation).

Finally, in order to reduce overfitting, the gradient descent models often include some regularization terms, such as the number of trees and shrinkage in my implementation of tree based GBM. In my implementation, it takes a lot of time to tune these parameters, and therefore it would be extremely useful if someone can develop some guiding principle about how to tune these parameters.

References

Freund, Y. and Schapire, R. E. (1995). A desicion-theoretic generalization of on-line learning and an application to boosting. *In European conference on computational learning theory*, pages 23–37.

Freund, Y. and Schapire, R. E. (1996). Experiments with a new boosting algorithm. *Machine Learning: Proceedings of the Thirteenth International Conference*.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 29(5):221–246.

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics and Data Analysis*, 38(4):367–378.

Friedman, Jerome, T. H. and Tibshirani, R. (2000). Stochastic gradient boosting. *Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)*, 28(2):337–407.

Mason, Llew, J. B. P. L. B. and Frean, M. (1999). Functional gradient techniques for combining hypotheses. *Advances in Neural Information Processing Systems*, pages 221–246.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386–407.