Boosting with R Programming

A Short Introduction to **gbm** Package

Zhou Yu
Supervised by Trevor Hastie

Jun 12, 2009

Abstract

This is a short introductory tutorial for the **gbm** package for Gradient Boosting Model by Greg Ridgeway. I will use the Spam data as an illustration to show the use of the package. I divide the package functions into five parts and introduce them accordingly in the second section of this paper. Part 1 focuses on the call of gbm function. Part 2 is prediction. Part 3 is about selection of optimal number of iteration of boosting. Part 4 is the relative importance of predictors and part 5 is is the partial dependence between variables.

1 Introduction

In this short tutorial of **gbm** package by Greg Ridgeway in R, I will divide the function in the package into 5 part and illustrate them one by one by a Spam data example. Part 1 is the call of gbm function, which fit a Gradient Boosting Model. There are a lot of parameters in this gbm function and I will introduce them in detail. Part 2 will be the prediction by the object returned by the gbm function in part 1. Part 3 will be the selection of optimal number of iteration, an essential problem in application of Boosting. In the package, we are provided three different methods to make the iteration selection. They are Test Data Selection, Out-Of-Bag Selection and Cross-Validation selection. In part 4 we will discuss the relative importance of predictors in the model. Part 5 will be the partial dependence between variables, which is an interesting application of Lattice Graph in **gbm** package.

2 Illustration with Spam Example

I will use the Spam classification problem as an example to introduce the parameters and functions in the package**gbm**.The data can be downloaded from:
http://archive.ics.uci.edu/ml/machine-learning- databases/spambase/spambase.data.
The spam data has 4601 observations. We will use 80% of the data as training set and the remaining part as test data. The response is in the last column and we will name it as "type".

```
> library(gbm)
> spam<-read.table("http://archive.ics.uci.edu/ml/machine
  -learning-databases/spambase/spambase.data",sep=',')

>  rflabs<-c("make", "address", "all", "3d", "our", "over", "remove",
  "internet","order", "mail", "receive", "will",
  "people", "report", "addresses","free", "business",
  "email", "you", "credit", "your", "font","000","money",
  "hp", "hpl", "george", "650", "lab", "labs",
  "telnet", "857", "data", "415", "85", "technology", "1999",
  "parts","pm", "direct", "cs", "meeting", "original", "project",
  "re","edu", "table", "conference", ";", "(", "[", "!", "$", "#",
  "CAPAVE", "CAPMAX", "CAPTOT","type")
  # Names for predictors and response

> colnames(spam)<-rflabs
```

Here we randomize the order of rows in our observations. This is important when we call the **gbm** function to fit the boosting model and the reason for this will be clear soon.

```
> set.seed(131)
> x<-spam[sample(nrow(spam)),]
```

2.1 Call of gbm function

Now we call the **gbm** function to fit the Gradient Boosting Model.

```
> set.seed(444)# Random for bag.fraction
> gbm0<-gbm(type~.,data=x,train.fraction=0.8,
```

```
interaction.depth=4,shrinkage=.05,
n.trees=2500,bag.fraction=0.5,cv.folds=5,
distribution="bernoulli",verbose=T)
```

We will first introduce the ordinary arguments:

train.fraction
: A number less or equal to 1, indicating the fraction of data used as training data. For example, if train.fraction=0.8, the first 0.8 * nrows(data) will be used as training data while the remaining 20% will be used as test data. So it is recommended to randomize the rows of data before fit the model in case that the responses in the original data are monotonic increasing or all 1 responses are at the top while the remaining 0 are at the bottom. An alternative option is to separate training set and test set at first and put only training set in the formula, setting train.fraction=1, which is the default setting.

interaction.depth
: The number of splits within each tree, which indicating the maximum depth of variable interactions. 1 implies an addictive model, the so called stump tree; 2 implies a model with up to 2-way interaction.

shrinkage
: A shrinking parameter to scale the contribution of each tree in the model.

n.trees
: Number of trees to iterate.

bag.fraction
: The fraction of subsampling from training data. If bag.fraction=0.5, then each time when we fit a new tree we will subsample 50% of the training data randomly to fit the tree. See detail in [4] Friedman(1999), Stochastic Gradient Boosting.

cv.folds
: Parameter for cross validation. If cv.folds=5, the model will perform a 5 fold cross-validation and output. If it is computational feasible, it is always recommended to add this parameter for the use of best number of iteration selection later.

distribution

bernoulli
: Logistic regression for 0-1 outcomes, recommended for classification.

adaboost
The exponential loss for 0-1 outcomes, for classification.

Gaussian
For minimizing squared Error, for regression.

laplace
For minimizing absolute error, for regression.

poisson
Count outcomes.

coxph
Censored survival outcomes.

verbose
Logical variable. If TRUE, gbm will print out the progress and performance of indicators.

keep.data
Logical variable. If TRUE, gbm will keep the data and an index of the data stored with the returning gbm object. Keeping the data will make the subsequent call of gbm.more faster though at the cost of storing an extra copy of the data sets.

If one wants to fit more iteration on the existing gbm.object, one can use gbm.more function.

```
> gbm1<-gbm.more(gbm0,n.new.tree=100,verbose=F)
```

For the technical details of the parameters and distributions, one can type in vignette("gbm ") to see the detail written by the author of **gbm**.

```
> vignette("gbm")
```

One can also use the following to know more about the return value in the object of gbm.

```
> ?gbm.object
```

2.2 Prediction

We will first discuss the fitted value in a bit more detail here, stored in gbm0$fit.

```
> gbm0$fit
```

This is a vector containing the fitted values on the scale of the loss function. Here in our case it returns the log-odds scale since we our using distribution as "bernoulli". Under the "bernoulli "distribution, we can use the predict function to return the probability of responses by specifying the type as response.

```
> gbm0.predict<-predict(gbm0,x_train,type="response",n.trees=300)
```

Another option for type is "link", which return the same value as gbm0$fit. For all distribution except "bernoulli"and "poisson ", the return value for both"respond"and "link"are the same. The predict function doesn't have an direct return of 0, 1 value. So it will be the user's responsibility and choice to set the threshold of probability to determine class.

2.3 Optimal Number of Iteration

Now we want to determine the optimal number of iteration. We have three different choices. According to the author of **gbm** package, the cross-validation usually have the best performance over the other two, with no waste of any observation of training data, though it would have far more computational cost.

Cross-Validation
: We use cross-validation to determine the optimal number of iteration. You need to have cv.folds>1 in the call of gbm.

OOB Error
: Because we have set the bag.fraction parameter smaller than 1, we have those Out-Of-Bag observations to estimate the optimal iteration by finding the lowest Out-Of-Bag error. This is equivalent to N fold cross-validation.

Test Error
: By setting the train.fraction smaller than 1, we have the test data set to decide the optimal iteration. You need have train.error¡1 in the call of gbm.

In the **gbm**, we use the gbm.perf to plot the curve for error rate and number of iterations. perf is short for performance. This function

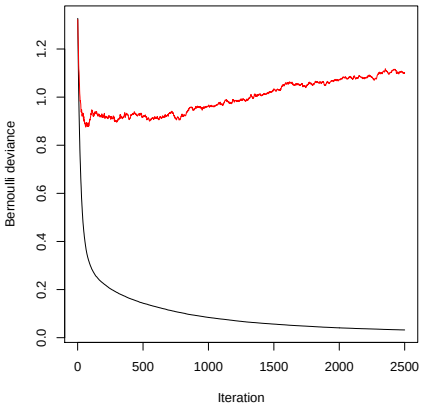

Figure 1: The plot produced by gbm.perf(gbm=0,method="test"). The monotonic decreasing black line is for training error and the red line is for the test error.

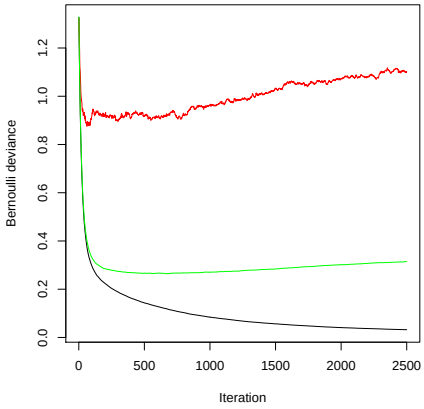

Figure 2: The plot produced by gbm.perf(gbm=0,method="cv "). The monotonic decreasing black line is for training error, the red line is for the test error and the green line is the cross-validation error.

will also return the value of the optimal number of iteration according to the criterion specified by the parameter "method ". The value of method can be test, OOB or cv. The OOB result is relatively conservative, which means it generally underestimate the optimal number of iteration.

```
> best.iter_test<-gbm.perf(gbm0,method="test")
> best.iter_test
[1] 62
> best.iter_OOB<- gbm.perf(gbm0,method="OOB")
> best.iter_test
[1] 175
```

The results are shown in Figure 1 and Figure 2. Notice that for both test and OOB we will produce the same plot, with one line for training data and one line for test data since we have set train.fraction< 1 in the call of gbm. In Figure 1, the black lines is for the training error, which is monotonic decreasing in boosting. The red line is for the test error. If by default the train.fraction=1, then we will only have the black line for the test error.

If we use cross validation option, we will get a third line in the plot. In figure 2, the additional green line is for the cross-validation error rate.Here the optimal iteration number given by cross-validation is much bigger compared to using test data and OOB data.

```
> best.iter<-gbm.perf(gbm0,method="cv")
> best.iter
[1] 671
```

2.4 Importance of Predictors

Different from other packages we usually use, the **gbm** package use the function summary to compute the relative importance of each variable.

```
> summary(gbm0,main="RELATIVE INFLUENCE OF ALL PREDICTORS")
          var      rel.inf
1         '!' 2.610369e+01
2          hp 1.402512e+01
3         '$' 9.927094e+00
4      remove 6.745069e+00
5      george 6.094248e+00
6        your 5.962285e+00
7        free 5.645130e+00
8      CAPMAX 4.868798e+00
9      CAPTOT 4.288199e+00
10     CAPAVE 3.179463e+00
11        you 1.769283e+00
  ..........................
```

We have a parameter "method"to specify the different method of computing the relative importance predictors. The default is method="relative.influence", which is the usual way suggested by Friedman(2001), as we shown in Figure 3. Another one is to randomly permutes one variable while keeping others fixed and computes the associated reduction in predictive performance. The more the reduction, the more importance the variable is. To use this method, we set method="permutation.test.gbm". One can use the help of summary.gbm to see more in detail. In Figure 4, we can see that importance of different predictors are more uniformly distributed for this criterion. The detail description of those two methods of evaluating predictor importance can be found in page 593, Chapter 15.3.2, of Elements of Statistical Learning, 2nd Edition.

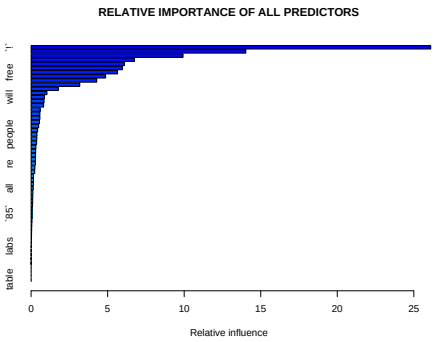

Figure 3: Variable importance by the default method of relative.influence

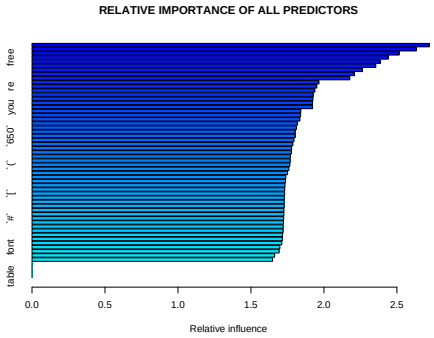

Figure 4: Variable importance by the method permutation.test.gbm.

```
> ?summary.gbm
```

2.5 Partial Dependence

The partial dependence function in package **gbm** incorporate package **lattice** and show the interaction between two or more variable by using lattice graph. Let's first see its arguments.

x
: A gbm.object returned by gbm.

i.var
: A vector of indices of the variables to plot. The indices are of the same order as the data sets we used for the parameter of gbm.

n.trees
: The number of trees used to generate the plot. Only the first n.trees will be used. By default, all tress in object x will be used to plot the graph.

We first draw a one variable partial dependence plot, using the variable '!', as shown in figure 5.

```
> best.iter=<-gbm.perf(gbm0,method="OOB")#Best iteration by OOB
> names(spam)
 [1] "make"       "address"    "all"        "3d"         "our"
 [6] "over"       "remove"     "internet"   "order"      "mail"
```

```
[11] "receive"    "will"       "people"     "report"     "addresses"
[16] "free"       "business"   "email"      "you"        "credit"
[21] "your"       "font"       "000"        "money"      "hp"
[26] "hpl"        "george"     "650"        "lab"        "labs"
[31] "telnet"     "857"        "data"       "415"        "85"
[36] "technology" "1999"       "parts"      "pm"         "direct"
[41] "cs"         "meeting"    "original"   "project"    "re"
[46] "edu"        "table"      "conference" ";"          "("
[51] "["          "!"          "$"          "#"          "CAPAVE"
[56] "CAPMAX"     "CAPTOT"     "type"
```

```
> plot(x=gbm0,i.var=52,n.trees=best.iter,
  main="Partial Dependence of '!' ")  #Number 52 is variable "!"
```

If we want to see the partial dependence on two variable, we get the 2-dimensional lattice plot in figure 6.

```
> plot(gbm0,c(21,24),best.iter,ylim=c(0,3),main=
  "Partial Dependence on 'your' and 'money' ")
  # 21 is "your", 24 is "money"
```

We can see there is indeed strong interaction between "your "and "money ".

We can even draw 3 variables partial dependence plot by the following syntax, which produces a 3 dimensional lattice plot, as the reader can try yourself. The plot result is not shown here.

```
> plot(gbm0,c(21,24,53),best.iter,ylim=c(0,3),main=
  "Partial Dependence on 'your', 'money' and '!' ")
```

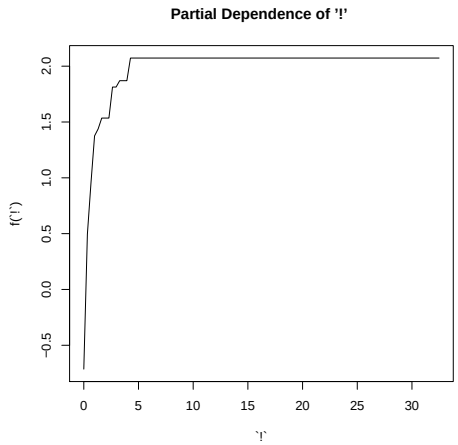

Figure 5: Partial dependence of log-odds of spam data on variable "!".

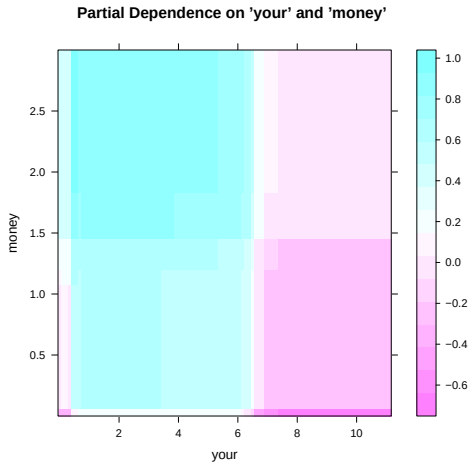

Figure 6: Partial dependence of log-odds of spam data on variable "your" and "money",which is a two dimensional lattice plot.

3 Acknowledgements

I am very grateful to the instructions and discussion with Prof.Trevor Hastie. Without his support and help, I cannot understand all those parameters and results in the whole package of **gbm**.

References

[1] Trevor Hastie, Robert Tibshirani, Jerome Friedman, *The Elements of Statistical Learning Data Mining, Inference, and Prediction.* Springer, 2nd Edition, 2008.

[2] Greg Ridgeway *Package 'gbm '*, Version 1.6-3, April 17, 2009, http://cran.r-project.org/web/packages/gbm/gbm.pdf.

[3] Greg Ridgeway *Generalized Boosted Models: A guide to the gbm package*, August 3, 2007 http://cran.r-project.org/web/packages/gbm/vignettes/gbm.pdf.

[4] Jerome H. Friedman *Stochastic Gradient Boosting*, March, 1999