STATS 315B: Homework 2 solution set

Spring2020

1 Problem 1

Random forests improves on the bagging algorithm, in that because the bagging algorithm bootstraps N samples from the original population of N for each tree, these trees tend to be correlated. The variance of the bagged average tends towards $\rho\sigma^2$ as the number of bagged trees approaches infinity (see ESL page 588), which means that our variance is still very much affected by pairwise correlations ρ between our trees. By randomly selecting for subsets of predictor variables at each split, we reduce this correlation and hence reduce the variance of the estimator, while the intrinsic variance of the trees σ^2 does not increase too much. The bias is also unaffected, and so we see that the random forests algorithm improves prediction accuracy through a reduction in estimator bias.

Random forests also tend to learn faster than boosting algorithms, considering there is no shrinkage factor involved. That being said, boosting outperforms random forests in many settings, even if the model takes more trees to stabilize. The disadvantage of random forests becomes apparent when p grows large while the number of relevant variables is small. In this setting,the probability of selecting the "good" predictors at each split becomes quite low (especially if the fraction of predictors selected is low), and random forests tends to perform badly. Another disadvantage is that because each tree in a random forest is grown to full size (until some minimum node size is reached), this can create variance in the model due to overfitting. However, this is usually not a major, model-breaking problem.

We can also introduce randomization into the model (hence reducing the correlation between bagged trees) in other ways. Instead of choosing a random subset of predictors at each stage, we can choose a random subset of the samples in each node when deciding the next split. Another way to introduce variability is to reduce the bagging fraction to below 1; instead of taking bootstrap samples of size N for each tree, take one of size N/α where $\alpha > 1$. This will ensure that each tree is grown on a different data set, and hence will increase variability.

2 Problem 2: the benefits of regularization

Why is it necessary to use regularization in linear regression when the number of predictor variables is greater than the number of observations in the training sample? Explain. If $p > n$, then from basic linear algebra there will be infinite exact solutions to the equation $Y = X\beta$. This means that there are infinite solutions that will return zero residuals on our training set. However, these solutions will perform very badly on a test set, since they would severely overfit the data (also since there are infinite of them, there will be a huge amount of variance in their performance). This is why we need regularization; not only do regularization techniques like lasso and ridge return unique solutions even when $p > n$, but also decrease variance of the result by a lot at the cost of only a bit of bias. They thus provide a good solution to the overfitting problem.

Are there other situations where regularization might help? Even when $p << n$, there are reasons to use regularization. In general, if we expect some predictor coefficients to be small, then we benefit from using regularization, which forces the coefficients to shrink. If the variables are highly correlated, the associated coefficients will have the tendency to be highly variable. Regularization might as well provide a good way of "fixing" the coefficients. Thus, in many other settings, ridge and lasso can outperform linear regression, even if we have enough observations in the training set. The benefit is maximized when we expect some coefficients are identically zero (i.e. some predictors have no significance in our model). In this case, regularization techniques like lasso perform variable selection by often setting these coefficients to zero. Thus, regularization is also very useful when we expect sparse models.

What is the potential disadvantage of introducing regularization? The disadvantage to regularization is that it artificially biases our model. If the data has real high-frequency or otherwise chaotic features, use of regularization will hide those features in the final model. Regularization operates similarly to Occam's Razor - we artificially pick a simpler model, and when this is not the case the model suffers.

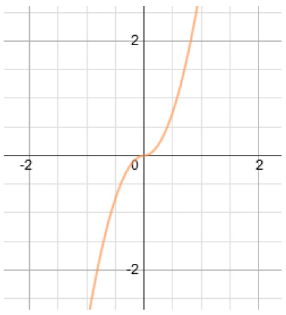
Why is sparsity a reasonable assumption in the boosting context. Is it always? If not, why not? In the case of boosting, we can almost always assume a sparse model. Boosting can be seen as a type of linear regression with basis members the set of all trees possible on our predictor space, and this is huge. The size of the dictionary p is thus (almost always) far greater than the number of datapoints. We cannot expect all trees to have predictive power, and in fact many of them are very correlated (for example, making the split $x_1 < 5$ will be very similar to $x_1 < 5.1$). So we should expect, except in very pathological boundary cases, that boosting operates in a sparse situation. Elements of Statistical Learning describes a very pathological case in which the model is a non-sparse linear combination of a million trees, and where the coefficients are generated by a Gaussian distribution. In this case, the L_2 penalty – rather than the sparse L_1 — would be more appropriate. However, even in this case, there is too little data to estimate the coefficients properly. Hence the L_1 penalty would yield similar performance to L_2, and is thus a better assumption: this is the "bet on sparsity" principle (ESL, 16.2 p611).

3 Problem 3

In general, we see that for $\gamma > 1$,

$$\frac{\partial}{\partial a_i}\sum_{j=1}^{n}|a_j|^\gamma = \gamma|a_j|^{\gamma-1}\frac{a_j}{|a_j|} = \gamma a_j|a_j|^{\gamma-2} \tag{1}$$

Because $\gamma > 1$, this is actually continuous across zero, and in fact we see the derivative exists at zero and is equal to zero (this is also easily verified via the direct limit definition of the derivative). Thus, we see that all partial derivatives $\frac{\partial}{\partial a_i}P_\gamma(\mathbf{a})$ are monotonically increasing, continuous, unbounded as $a_j \to \pm\infty$, and also pass through the origin. To illustrate this, we plot $\frac{\partial}{\partial a_i}\hat{P}(a_i)$ for $\gamma = 3$ below:



Similarly, we consider $\frac{\partial}{\partial a_i}\hat{R}(\mathbf{a})$, Because $\hat{R}$ is a concave-up quadratic function of each of the a_i, it is easy to see that this derivative will be a line with nonnegative slope (or a constant function). If we are to search for minima of $R + P$, then setting the partial derivatives to zero, we see that for each predictor i, this problem becomes

$$\frac{\partial}{\partial a_i}\hat{P}(\mathbf{a}) + \frac{\partial}{\partial a_i}\hat{R}(\mathbf{a}) = 0 \Rightarrow \frac{\partial}{\partial a_i}\hat{P}(\mathbf{a}) = -\frac{\partial}{\partial a_i}\hat{R}(\mathbf{a}) \tag{2}$$

Because we have already argued that the term on the left is monotonically increasing, continuous, and passes through the origin as a function of a_i, and the term on the right is a line (now with negative slope due to the minus sign), the only way these two curves can intersect at $a_i = 0$ is if $\hat{R}'(a_i) \propto a_i$, which tells us that $\hat{R}$ as a function of a_i is minimized when $a_i = 0$ (this is where the derivative vanishes). This means that $a_i = 0$ was an ordinary least squares solution. Thus, we have shown that if $a_i = 0$ is not an ordinary least squares solution, there is no reason to expect $a_i = 0$ in this power penalty setting, and hence we have shown for almost all cases (where the OLS solution has all nonzero coefficients), we should expect the power penalty regularized regression to also have nonzero coefficients (note that the other scenario that admits zero coefficients is if the risk function $\hat{R}(a_i)$ is constant, which leads to an equivalently zero $\hat{R}'(a_i) = 0$. However, this is only the case if $x_{ij} = 0, \forall i$, which is clearly not allowed).

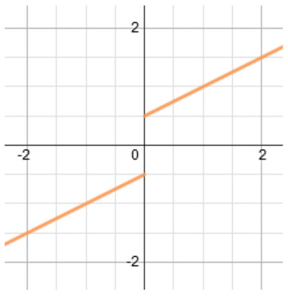
In the Elastic-Net setting, we now have a derivative $\frac{\partial}{\partial a_i}\hat{P}(\mathbf{a})$ that is discontinuous at the origin and also undefined at $a_i = 0$ for any i. We see that

$$\frac{\partial}{\partial a_i}\hat{P}(\mathbf{a}) = (\gamma - 1)a_i + (2-\gamma)\frac{a_i}{|a_i|} \qquad \text{for } a_i \neq 0 \tag{3}$$

This is now discontinuous at the origin. We see easily that

$$\lim_{a_i \to 0^+}\left(\frac{\partial}{\partial a_i}\hat{P}(\mathbf{a})\right) = 2 - \lambda \qquad \lim_{a_i \to 0^-}\left(\frac{\partial}{\partial a_i}\hat{P}(\mathbf{a})\right) = \lambda - 2 \tag{4}$$

We illustrate this by graphing the derivative for $\gamma = 1.5$ below:



As with the power penalty, the risk function has as its derivative with respect to a_i a line with nonnegative slope. Note that the only way it has zero slope is if all the x_{ij} were zero, so we assume the slope is in fact positive. Now, suppose that the negative risk function derivative $-\hat{R}'(a_i)$ has the property that $\lambda - 2 \leq \hat{R}'(0) \leq 2 - \lambda$. To find the minimum, we would like to find an intersection between $-\hat{R}'(a_i)$ and $\hat{P}'(a_i)$, but in this case, because the function $-\hat{R}'(a_i)$ passes through the "gap" in the discontinuity at zero, and is negatively sloped, this intersection will never occur. *The derivative of the total loss will never equal zero.* However, near zero, we see that

$$\lim_{a_i \to 0^+}\left(\hat{P}'(\mathbf{a}) + \hat{R}'(\mathbf{a})\right) \geq (2-\lambda) + (\lambda - 2) = 0 \tag{5}$$

And

$$\lim_{a_i \to 0^-}\left(\hat{P}'(\mathbf{a}) + \hat{R}'(\mathbf{a})\right) \leq (\lambda - 2) + (2-\lambda) = 0 \tag{6}$$

Thus, the total loss derivative is negative to the left of zero, but positive to the right. This tells us that there is indeed a local minimum at zero, and $a_i = 0$ for our Elastic-Net coefficient vector. We have thus shown that as long as $\lambda - 2 \leq \hat{R}'(0) \leq 2 - \lambda$, we will have $a_i = 0$. There is thus a range of cases in which we can expect $a_i = 0$ for any λ, and we have thus shown that the Elastic-Net with nontrivial Lasso component performs variable selection along its path indexed by λ. We are done.

4 Problem 4: Base Learner Correlated with Generalized Residuals

Note that:

$$\begin{aligned}\mathbb{E}[(y-\rho x_j)^2] &= \mathbb{E}[y^2 - 2\rho y\cdot x_j + \rho^2 x_j^2] \\ &= \mathbb{E}[y^2] - 2\rho\mathbb{E}[y\cdot x_j] + \rho^2\mathbb{E}[x^2] \\ &= \mathbb{E}[y^2] - 2\rho\mathbb{E}[y\cdot x_j] + \rho^2 \quad \text{since by assumption } \mathbb{E}[x^2]=1\end{aligned} \tag{1}$$

Now, if we try to minimize this as a function of ρ, we see that, taking the derivative, this occurs where

$$-2\mathbb{E}[y\cdot x_j] + 2\rho = 0 \implies \rho = \mathbb{E}[y\cdot x_j] \tag{2}$$

Thus, we have that

$$\begin{aligned}j^* &= \text{argmin}_j \mathbb{E}[y^2] - 2(\mathbb{E}[y\cdot x_j]^2 + \mathbb{E}[y\cdot x_j]^2 \\ &= \text{argmin}_j \mathbb{E}[y^2] - (\mathbb{E}[y\cdot x_j])^2 \\ &= \text{argmax}_j \big|\mathbb{E}[y\cdot x_j]\big|\end{aligned} \tag{3}$$

5 Problem 5

The partial dependence $F(x)$ on z_l is defined as:

$$\mathbb{E}_{z_{\backslash l}}[F(x)]$$

Let us assume that the variables x have probability density $f(x) = f(z_l, z_{\backslash l})$. We can rewrite the previous expectation as:

$$\mathbb{E}_{z_{\backslash l}}[F(x)] = \int_{z_{\backslash l} \in \Omega_{\backslash l}} F(z_l, z_{\backslash l}) \tilde{f}_{\backslash l}(z_{\backslash l}) dz_{\backslash l}$$

where $\tilde{f}_{\backslash l}(z_{\backslash l}) = \int_{z_l \in \Omega_l} f(z_l, z_{\backslash l}) dz_l$ is the marginal probability of $z_{\backslash l}$ and is independent of z_l.
We have:

$$\begin{aligned} \mathbb{E}_{z_{\backslash l}}[F(x)] &= \mathbb{E}_{z_{\backslash l}}[F_l(z_l)] + \mathbb{E}_{z_{\backslash l}}[F_{\backslash l}(z_{\backslash l})] \quad \text{by assumption and by linearity of the expectation} \\ &= F_l(z_l) + \int F_{\backslash l}(z_{\backslash l}) \tilde{f}_{\backslash l}(z_{\backslash l}) dz_{\backslash l} \quad \text{since } z_l \text{ is considered fixed} \\ &= F_l(z_l) + C \quad \text{since } F_{\backslash l} \text{ does not depend on } z_l \text{, not does the marginal } \tilde{f}_{\backslash l} \end{aligned} \tag{4}$$

However, for $\mathbb{E}[F(x) \big| z_l]$, we have:

$$\begin{aligned} \mathbb{E}[F(x) \big| z_l] &= \mathbb{E}[F_l(z_l) \big| z_l] + \mathbb{E}[F_{\backslash l}(z_{\backslash l}) \big| z_l] \quad \text{by assumption and by linearity of the expectation} \\ &= F_l(z_l) + \int F_{\backslash l}(z_{\backslash l}) \frac{f(z_l, z_{\backslash l})}{\tilde{f}_l(z_l)} dz_{\backslash l} \\ &= F_l(z_l) + \int F_{\backslash l}(z_{\backslash l}) \frac{f(z_l, z_{\backslash l})}{\tilde{f}_l(z_l)} dz_{\backslash l} \end{aligned} \tag{5}$$

The conditional expectation $\mathbb{E}[F_{\backslash l}(z_{\backslash l}) \big| z_l]$ is thus a function of z_l. A sufficient condition for the two expectations to be the same is to have z_l and $z_{\backslash l}$ independent.

6 Problem 6

1. The following are the results we obtained:

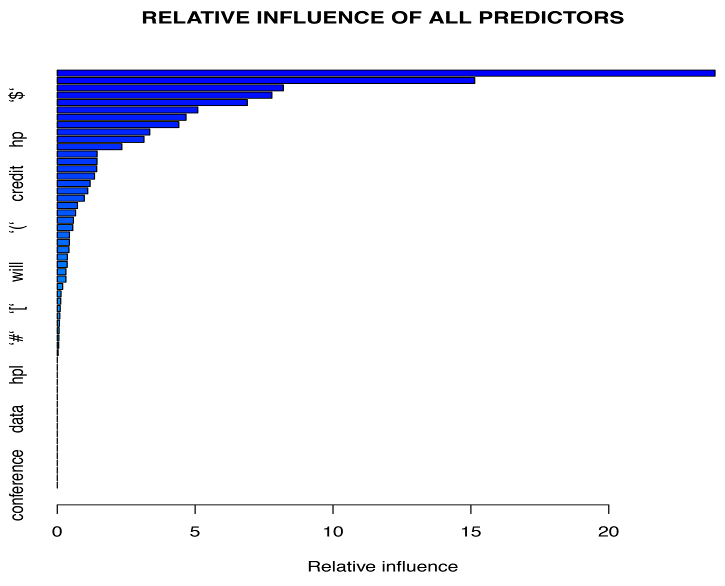
   Misclassification Rate = 3.9%
   Spam Misclassification Rate = 5.7%
   Non-spam Misclassification Rate = 2.7%

2. In order to penalize misclassifying "good" emails as "spam" more than misclassifying "spam" as "good", we modified the weights w_i given to the gbm model. The weights affect the cost function of the tree as follows:

$$err = \frac{\sum_{i=1}^{N} w_i 1(y_i \neq G(x_i)}{\sum_{i=1}^{N} w_i}$$

   Essentially, we assign a weight w_i to each training observation i and penalize its mis-classification using that weight. Therefore, to achieve a higher penalization for misclassifying non-spam emails compared to spam, we just associated greater weights to the non-spam than to the spam observations. Specifically, we associated a weight of 0.134 to the non-spam and 0.01 to the spam emails. The misclassification results are as follows:

   Misclassification Rate = 10.10 %
   Spam Misclassification Rate = 24.75 %
   Non-spam Misclassification Rate = 0.21 %
   The dependence plot is given below. Some of the important attributes are "remove", "!", "money", "000", etc.



```
##Problem 6 part a
library(gbm)
spam_train<-read.table("Spam_Train.txt", sep=',')
spam_test<-read.table("Spam_Test.txt",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")
colnames(spam_train)<-rflabs
colnames(spam_test)<-rflabs

gbm0<-gbm(type~.,data=spam_train,train.fraction=1,interaction.depth=4,
shrinkage=.05,n.trees=200,bag.fraction=0.5,cv.folds=5,
distribution="bernoulli",verbose=T);

gbm0.predict<-predict(gbm0,spam_test,type="response")
gbm0.predict_bin<-ifelse(gbm0.predict<0.5, 0, 1)
result<-ifelse(gbm0.predict_bin == spam_test$type,1,0)
err_rate = 1-mean(result)

#print misclassification rate
print(err_rate*100)
temp1<-ifelse((spam_test$type==1) & (gbm0.predict_bin!=spam_test$type),1,0)
temp2<-ifelse(spam_test$type==1,1,0)
spam_err=sum(temp1)/sum(temp2)
print ('spam misclassification rate')
print(spam_err*100)
temp1<-ifelse((spam_test$type==0) & (gbm0.predict_bin!=spam_test$type),1,0)
temp2<-ifelse(spam_test$type==0,1,0)
non_spam_err=sum(temp1)/sum(temp2)
print ('non-spam misclassification rate')
print(non_spam_err*100)

##Problem 6 part b
w<-ifelse(spam_train$type==0,0.134,0.01)
gbm1<-gbm(type~.,data=spam_train,weights = w, train.fraction=1,
interaction.depth=4,shrinkage=.05, n.trees=200,bag.fraction=0.5,cv.folds=5,
distribution="bernoulli",verbose=T);
```

7 Problem 7

The code is provided below.

Here, we can see the most interesting information. We refer to Figure 1. With income, we see a very expected increase in house prices as neighborhoods that make more will cost more. Similarly, we see a good peak at about 2-4 for occupancy. We would expect larger houses meant for families to have a greater cost. However, as the occupancy of a neighborhood increases, this could mean lower income levels, which would then make the house prices drop. Finally, for latitude, we see that the norther you are, the cheaper your house will become. This is probably not the full story. Thus, we can look into this a bit further to find some really interesting things out. In Figure 2, basically plot 6 different price plots of latitude against longitude for 6 different median incomes. We see something very interesting. Each of the 6 plot points basically represents the space of California, since we're plotting latitude against longitude. Thus, the left side of the plot will correspond to the coast, and as we go further right, we'll be going more inland. Same for the up/down, we'll be going from north to south. Thus, we can see that as we increase our income, the expensive housing prices shift more inland. This makes sense. Initially, we only see expensive house prices for low income neighborhoods in the south coast (basically, San Diego). However, as we get into more and more high income earning neighborhoods, we find expensive houses more in land and north. This basically means we're including neighborhoods in Mountain View and Palo Alto.

```
library(gbm)
#We read the data
cal <- read.table('California_Data.txt',sep=",",header = FALSE);
names(cal) <- c("MedianValue","MedianIncome","HouseMedianAge",
"AvgNoRoom","AvgNoBedrooms","Population","AvgOcc","Latitude","Longitude");
set.seed(121)
x<-cal[sample(nrow(cal)),]


#WE fit the GBM Model
gbm0 <- gbm(MedianValue ~., data=x, train.fraction=0.8, interaction.depth=4,
shrinkage=.01,n.trees=5000,bag.fraction=0.5,cv.folds=5,
distribution="gaussian",verbose=T);
best.iter0 <- gbm.perf(gbm0,method="cv");

train.error <- min(gbm0$train.error);
test.error <- min(gbm0$valid.error);
cv.error <- min(gbm0$cv.error);

print(c(train.error,test.error,cv.error))
# 0.1785954  0.4679111 0.2227002


summary.gbm(gbm0)
# var   rel.inf
# MedianIncome     MedianIncome 50.816394
# Longitude           Longitude 13.776515
# AvgOcc                 AvgOcc 13.735860
# Latitude             Latitude 11.952022
# HouseMedianAge HouseMedianAge  4.315428
# AvgNoRoom           AvgNoRoom  2.497519
# Population         Population  1.533650
# AvgNoBedrooms   AvgNoBedrooms  1.372612
```

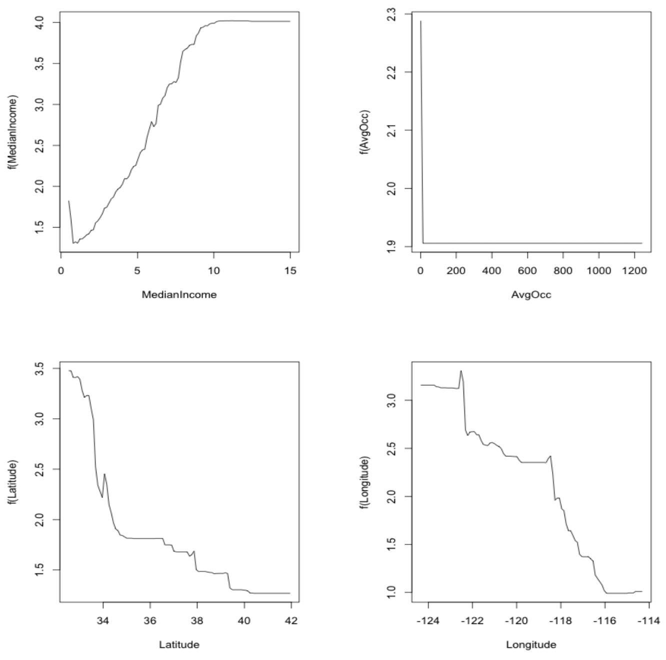


Figure 1: Partial Dependence

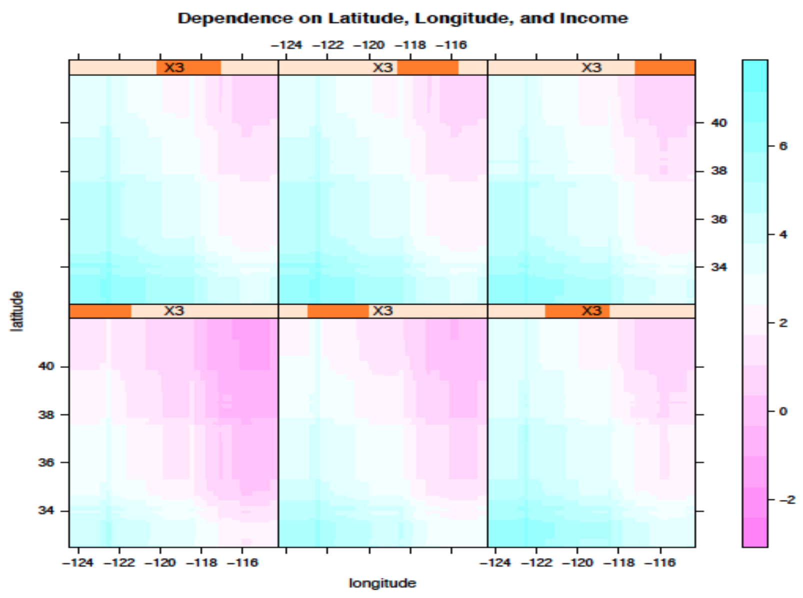


Figure 2: Dependence on Latitude, Longitude and Income

8 Problem 8

1. We can fit a gbm model to the data. We set as factors anything that is not an ordered variable. We also note that gbm automatically uses surrogates, so we do not need to do anything with imputation or data deletion for missing values. We get a testing error of 3.85. The mean squared error we got on homework 1 was 4.16. This is indeed an improvement.

2. The most important variables are occupation,household, marital status and ethnicity. It is true that sex is one of the least important variables in this analysis. But our analysis is correct by looking at the dependence structure between sex and occupation. It can seen both for male and female the dependence on occupation is identical, which is one of the most influential factors. And because of this hidden effect of similar occupation of either sex, it does not come as one of the most influential variable and thus the result is consistent. See Figure 3.

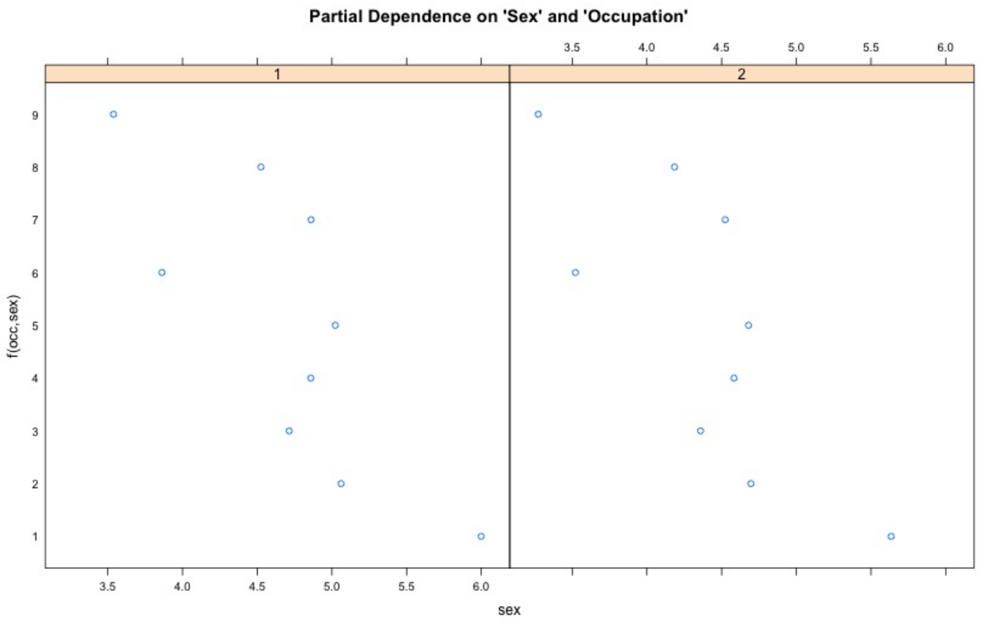


Figure 3: Partial Dependence on ‘Sex’ and ‘Occupation’

```
library(gbm)
#We read the data

income <- read.table('Income_Data.txt',sep=",",header = FALSE);
names(income) <- c("income", "sex", "marital", "age", "edu", "occ", "living",
                 "dual.in", "n.peop", "n.und.18", "household", "h.type",
                 "ethnic", "language")

for (i in c(2,3,6,8,11,12,13,14)){
  income[,i] = as.factor(income[,i])
}
for (i in c(1,4,5,7,9,10)){
  income[,i] = as.ordered(income[,i])
}

#We split the data into training and test set
test.set <- sample(c(1:8993),2000);

#We fit gbm model
gbm0 <- gbm(income ~., data=income[-test.set,], interaction.depth=4,
shrinkage=.1, n.trees=5000,bag.fraction=0.5,cv.folds=5,
distribution="gaussian",verbose=T);

best.iter0 <- gbm.perf(gbm0,method="cv");
yhat <- predict.gbm(gbm0,income[test.set,],type="response",n.trees = best.iter0);
mean((as.numeric(income[test.set,1]) - yhat)^2)
#3.72

#Error of CART from hw1
mean((predict(tree_opt,income[test.set,]) - as.numeric(income[test.set,1]))^2)
# 4.16

# GBM gives better abs test error over CART.
#So we might conclude that MART gets slightly better accuracy than CART.

summary.gbm(gbm0)
#var    rel.inf
#occ            occ 28.5777984
#household household 13.5866849
#marital     marital 12.5364529
#ethnic       ethnic 11.2578068
#n.peop       n.peop  6.8883616
#edu             edu  6.4684603
#age             age  6.2478105
#h.type       h.type  6.1172914
#living       living  3.3673484
#language   language  2.1371208
#n.und.18   n.und.18  1.3959783
#dual.in     dual.in  0.8354575
#sex             sex  0.5834282
```

9 Problem 9

(a) In this problem, we used the first 80% of shuffled data set as the training set, and the rest as the test set. Also the maximum number of trees was set to 500, and using the 5-fold cross-validation, we got the best number of trees to be 138(see the left panel of Figure 1).

The overall misclassification error was 42.83%, and the misclassification errors for classes were as listed on the Table 1.

Class	1	2	3	4	5	6	7	8	9
Misclass. Error(%)	19.67	96.10	67.42	76.32	37.41	21.70	65.52	21.48	82.81

Table 1: Misclassification errors for each class

Among all occupation classes, our GBM model could predict Professional/Managerial(1), Student, HS or College(6), Retired(8) classes with ease, while it had difficulty predicting the classes Sales Worker(2), Clerical/Service Worker(4), Unemployed(9). In particular, prediction for Sales Worker(2) was worse than random guess (which gives misclassification probability 8/9, or 88.89%).

(b) The most important variables in our gbm model are AGE, ANNUAL INCOME OF HOUSEHOLD, EDUCATION and HOUSEHOLDER STATUS, whose relative influences were greater than 10, as shown in the bar plot in Figure 1.

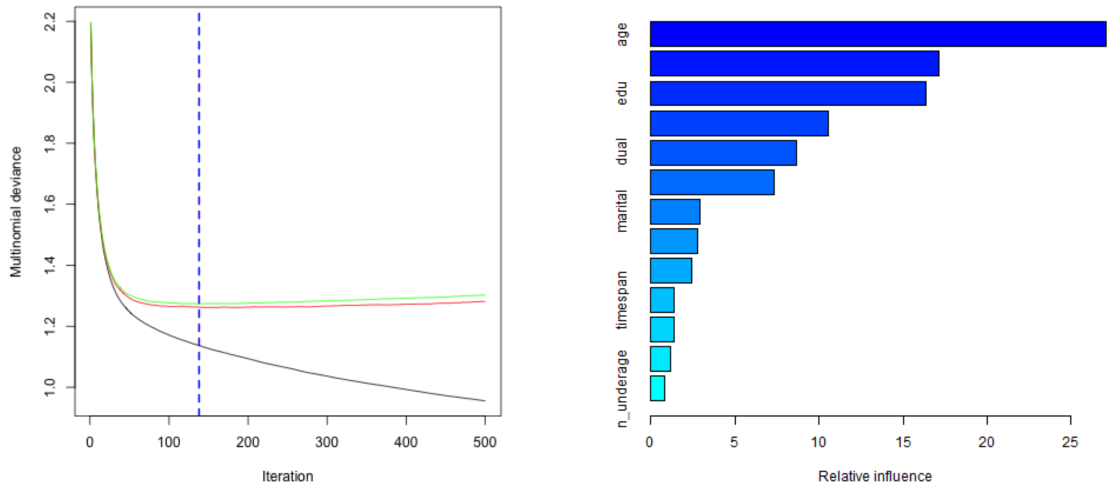


Figure 1: Left: Training(black), test(red) and CV(green) errors vs the number of trees. Right: the relative influence of variables.

The R code for this problem is provided below.

```r
### Stats 315B HW 2 Problem 9 ###
install.packages("gbm")
library(gbm)

## Data preparation ##
```

```
data = read.csv("Occupation_Data.txt",header = FALSE)
colnames(data) = c("occupation", "home_type", "sex", "marital", "age", "edu",
                   "income", "timespan", "dual", "n_person", "n_underage",
                   "household_status", "ethnic", "language")
factor.cols = c(1, 2, 3, 4, 9, 12, 13, 14)
data[,factor.cols] = lapply(data[,factor.cols],as.factor)
# Shuffle data set for dividing it into training and test sets
set.seed(2016)
data = data[sample(nrow(data)),]
attach(data)

## Fitting gbm ##
set.seed(2016)
gbm0 = gbm(occupation~., data = data, train.fraction = .8, interaction.depth = 4,
           shrinkage = .05, n.trees = 500, bag.fraction = 0.5, cv.folds = 5,
           distribution="multinomial", verbose=T)

## Prediction and results ##
test.data = data[(gbm0$nTrain+1):nrow(data),]
# Find the best number of trees by CV
png("p9_cv.png")
best.iter = gbm.perf(gbm0,method="cv")
dev.off()
best.iter # 138

gbm0.pred = predict(gbm0, test.data, type = "response", n.trees = best.iter)
gbm0.pred.class = apply(gbm0.pred,1,which.max)
miscl.error = sum(gbm0.pred.class!=test.data$occupation)/nrow(test.data)
100*round(miscl.error,4) # 42.83

miscl.error.by.class = c()
for (i in levels(test.data$occupation)){
  one.class = (test.data$occupation == i)
  miscl.error.by.class = c(miscl.error.by.class,
                           sum(gbm0.pred.class[one.class]!=i)/sum(one.class))
}
100*round(miscl.error.by.class,4)
# 19.67 96.10 67.42 76.32 37.41 21.70 65.52 21.48 82.81

png("p9_importance.png")
head(summary(gbm0))
dev.off()
```

prob9.R