

Homework 3

Patricia Hoffman, PhD.

**1) Once again check out wine quality data set described in the web page below:**

**http://archive.ics.uci.edu/ml/machine-learning-databases/wine- quality/winequality.names**

**Remember the Red Wine data set (winequality-red.csv) contains 1599 observations of 11 attributes. The median score of the wine tasters is given in the last column. Note also that the delimiter used in this file is a semi colon and not a comma. This problem is to create a linear model for this data set using the first 1400 observations. Next check the models performance on the last 199 observations. How well did the model predict the results for the last 199 observations? What measure did you use to evaluate how well the model did this prediction? Next use the model to predict the results for the whole data set and measure how well your model worked. (hint: use the r function lm and the regression example from class) Check the coefficients (use the summary function in r).**

**a. How well did the model predict the results for the last 199 observations?**

Correct Predictions = 120

Correct Ration = 0.6030151

**b. What measure did you use to evaluate how well the model did this prediction?**

# Call:

# lm(formula = quality ~ ., data = wine\_Train)

#

# Coefficients:

# (Intercept) fixed.acidity volatile.acidity citric.acid

# 18.709707 0.023299 -1.076205 -0.145282

# residual.sugar chlorides free.sulfur.dioxide total.sulfur.dioxide

# 0.006569 -1.825419 0.003415 -0.003428

# density pH sulphates alcohol

# -14.950446 -0.291896 0.873197 0.277656

**2) The objective of this problem is to see how the number of data points affects over fitting.**

**2a) Start with the full sonar data set (both training and test sets). Use 5 fold cross validation on the whole data set. Average the 5 training errors from each of these runs. Average the 5 test errors from each of these runs. What are these averages?**

Employing the 5-fold cross validation method for linear regression, we get

Average training error = 0.0817

Average test error = 0.255

R code :

rm(list=ls())

setwd("/Users/team\_work/working\_dir")

### Read in the sonar training and test sets

sonar\_train <- read.csv("sonar\_train.csv",header=F)

sonar\_test <- read.csv("sonar\_test.csv",header=F)

### Merge the training and test sets into one set

sonar\_set <- rbind(sonar\_train,sonar\_test)

sonar\_set

### Function to compute the error (training or test error)

compute\_error <- function(Sonar\_Fit,Model\_Sonar,Sonar\_In){

lmFit <- predict(Model\_Sonar,newdata=Sonar\_Fit)

correct <- sum(lmFit\*Sonar\_In$V61 > 0)

Err <- 1 - correct/length(Sonar\_In$V61)

return(Err)

}

##### Function for k-fold cross validation using linear regression

cross\_validate\_linear <- function(df, k){

I <- seq(1:nrow(df))

trainErr <- 0.0

testErr <- 0.0

for(ixval in seq(from = 1, to = k)){

Iout <- which(I%%k == ixval %%k)

SonarIn <- df[-Iout,]

SonarOut <- df[Iout,]

lmSonar <- lm(V61~.,data=SonarIn)

sonarFit <- SonarIn[,-61]

trainErr <- trainErr + compute\_error(sonarFit, lmSonar, SonarIn)

sonarFit <- SonarOut[,-61]

testErr <- testErr + compute\_error(sonarFit, lmSonar, SonarOut)

}

#### Average training and test errors will be the sum divided by k

Err <- c(trainErr/k, testErr/k)

return(Err)

}

### 2a) Compute the average training and test errors using 5-fold cross validation

Error <- cross\_validate\_linear(sonar\_set,5)

print("Average training error = ")

Error[1]

### Average training error = 0.0817

print("Average test error = ")

Error[2]

### Average test error = 0.255

**2b) Next run a series of cross validations on a series of data sets which are decreasing in size. (At a minimum you must have more data points than attributes to use lm.)**

I have chosen 10 random data set sizes (samples from 61 to 208, given that 208 is the total number of rows in the sonar data set) and sorted them in descending order. I pulled random rows from the dataset based on this sample number and generated a three-dimensional matrix whose first column represents data set size, second column represents average training error and the third column represents average test error.

R code :

###### 2b) Pick 10 random data set sizes between 61 and 208 and arrange these in decreasing order

sample\_size <- sort(sample(61:208,10),decreasing=T)

### Set up a matrix with 10 rows and 3 columns. First column represents the chosen data set size, second column represents

### training error and the third column represents test error

Err\_Matrix <- matrix(nrow = 10, ncol = 3)

colnames(Err\_Matrix) <- c("Dataset\_Size","Training\_Error","Test\_Error")

for(i in seq(1:10)){

Err\_Matrix[i,1] <- sample\_size[i]

### Perform 10-fold cross validation on the data set with size sample\_size[i]

Error <- cross\_validate\_linear(sonar\_set[sample(nrow(sonar\_set),sample\_size[1]),],10)

Err\_Matrix[i,2] <- Error[1]

Err\_Matrix[i,3] <- Error[2]

}

Err\_Matrix

### Dataset\_Size Training\_Error Test\_Error

### [1,] 201 0.08457643 0.2435714

### [2,] 199 0.09231737 0.2485714

### [3,] 198 0.08069982 0.2438095

### [4,] 196 0.08015040 0.2671429

### [5,] 177 0.08347759 0.2533333

### [6,] 156 0.07961326 0.2590476

### [7,] 149 0.08125230 0.2592857

### [8,] 126 0.09397483 0.2740476

### [9,] 114 0.07904236 0.2388095

### [10,] 69 0.08126765 0.2685714

cex = .5

pch = 21

matplot(Err\_Matrix[,1],Err\_Matrix[,2],xlim=c(50,250),ylim=c(0,1),main="Data set size versus Error",

xlab="Number of observations",

ylab="Training & Test Error",

col='red',

cex = cex,

pch = pch)

matpoints(Err\_Matrix[,1],Err\_Matrix[,3],col='blue',pch=pch)

legend(50, 1.0, c('Training Error', 'Testing Error'), cex=.8, col=c('red', 'blue'), pch=c(pch, pch))

**Plot both the training error and test error verses data set size on the same graph. The horizontal axis should be the number of observations in the data set and the vertical axis should be error rate.**



**3a) From problem 2, pick a data set size that is clearly over fit. Try to improve the result with an ensemble method. Use the small sonar data set that you have chosen as the training set and put the rest of the data into the hold out set. Generate 10 linear models using your training data set. Each of these models will incorporate a different random subset of the attributes.**

**To generate one of these linear models:**

**A) Fix n to be a number between 5 and 30. Now, choose n attributes randomly.**

**For example if you fixed n to be 11 then choose 11 attributes randomly from the 60 available sonar attributes.**

**B) Fit the linear model to the training set using only these n attributes.**

**C) Use this model to make predictions on both the training set and the hold out set.**

**D) Record the training error and test error.**

**E) Retain the predictions for both the training set and the test set (This will become an attribute for problem 3b)**

**F) Rank this model. (You will have 10 models to rank. Give the model with the lowest test error the highest rank)**

Below is the ranked list of errors for each of the 10 random models. They are sorted from smallest to largest so rank 10 is on top (i.e. see the row names for the rank number). It can be seen that the minimum “Test” error is about 26%.

> ErrNSorted

Train Test

10 0.2384615 0.2564103

9 0.1923077 0.2564103

8 0.2000000 0.2820513

7 0.2076923 0.3076923

6 0.2230769 0.3076923

5 0.2076923 0.3205128

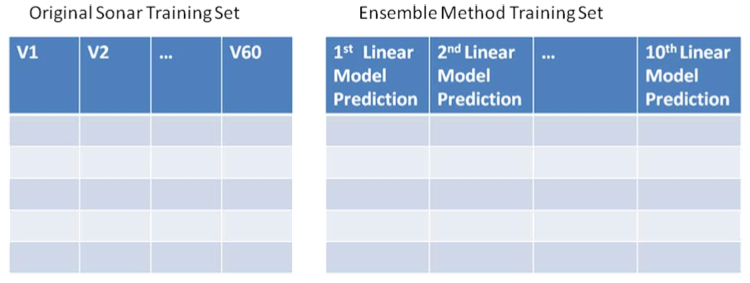
4 0.2000000 0.3205128

3 0.2230769 0.3333333

2 0.2461538 0.3589744

1 0.2615385 0.3717949

**3b) In this step, use linear regression to create an ensemble model. Treat the output of the 10 linear models (from step E above) as inputs to a new regression to create the ensemble model. (See the figure below.) You now have 10 new attributes for each observation (one from each of the predictions you made in step E above.) The next step is to perform the linear regression over the ensemble training set which only has the 10 new attributes. (Ignore the original 60 attributes.) Compare the performance of the ensemble model on the training set with the performance of the ensemble model on the hold out set. What are the coefficients for the 10 new attributes? Compare these coefficients with the ranks given to the models in part F) above.**



It can be seen below that the training and test error is much lower for the ensemble method. The training error went from ~24% to ~15% and the test error went from ~26% to ~15%. The ensemble method improved results significantly.

> trainErrEnsemble

[1] 0.1461538

> testErrEnsemble

[1] 0.1538462

Below are the coefficients for the ensemble model.

> as.matrix(coef(lmETrain))

[,1]

(Intercept) 0.006635544

V1 0.086151794

V2 0.071808311

V3 0.227308924

V4 0.167952079

V5 0.419067269

V6 -0.008488708

V7 0.050047936

V8 -0.203900113

V9 0.352880597

V10 0.268482278

**3c) Repeat Homework problem 3b with various values for n (the number of randomly chosen attributes). In part 3aA of the example above, n was fixed to be 11. Now, put n in a loop. That is in pseudo code: for (n in seq(5,30,by=5){... create ensemble model}. Plot the training error and test error as a function of n. How well did the new model do in comparison to the original model created by problem 2?**

Below is a dump of the Training and Test errors for each attribute count value used in building each of the models. The data was dumped for both the non-ensemble method (ErrVSNSorted) and the ensemble method (testErrEnsembleDFSorted). Below that is a plot of the error for all 4 items vs. N (train & test for both ensemble and non-ensemble). The ensemble method made a significant improvement over the non-ensemble method. The “test” error dropped from ~30% to ~18%.

> ErrVSNSorted

Train Test AttribCount

3 0.2030769 0.3038462 15

2 0.2500000 0.3141026 10

6 0.1407692 0.3153846 30

4 0.1969231 0.3269231 20

5 0.1653846 0.3384615 25

1 0.2807692 0.3538462 5

> testErrEnsembleDFSorted

Err N

4 0.1794872 20

5 0.1794872 25

2 0.2179487 10

6 0.2179487 30

3 0.2307692 15

1 0.2692308 5



**4) Perform a ridge regression on the wine quality data set from problem 1. Compare the coefficients resulting from the ridge regression with the coefficients that were obtained in problem 1. What conclusions can you make from this comparison?**

As in problem 1, this method also has a high error rate for both training and testing data. The minimum test error was “0.553” for a lambda value of “25.1”.

**Ridge Regression Linear Regression(Problem1)**

**Coefficients Coefficients**

(Intercept) 24.257154088 18.709707

fixed.acidity 0.027426843 0.023299

volatile.acidity -1.050428877 -1.076205

citric.acid -0.110732308 -0.145282

residual.sugar 0.008985090 0.006569

chlorides -1.796104656 -1.825419

free.sulfur.dioxide 0.003155438 0.003415

total.sulfur.dioxide -0.003354383 -0.003428

density -20.625243849 -14.950446

pH -0.245217194 -0.291896

sulphates 0.863509097 0.873197

alcohol 0.267599624 0.277656

The relative magnitudes are close but the values do differ some.  Density is a little higher for RR and so is the Intercept. They seem to agree on which have the biggest effects.

