HW5 SML1

Jie Li jl5246

April 17, 2019

Problem 1

Ridge Regression

$$\min_{\beta} \|y-\hat{y}\|^2 + \lambda\|\beta^2\| \implies \min_{\beta} \sum_{i}^{n}(y_i - \sum_{j}^{p}\beta_j * x_{ij})^2 + \lambda\sum_{j}^{p}\beta_j^2$$

$$\hat{\beta}_{ridge} = (X^TX + \lambda I)^{-1}X^TY$$

The designed matrix in our setting: $\lambda = 0.01$

$$\mathbf{X} = \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}$$

$$\mathbf{Y} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

$$\mathbf{X^TX} = \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}$$

$$\hat{\beta} = (\begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix} + \begin{bmatrix} 1.01 & 0 \\ 0 & 1.01 \end{bmatrix})^{-1}\begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}^T\begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 0.4988 \\ 0.4988 \end{bmatrix}$$

Lasso Regression

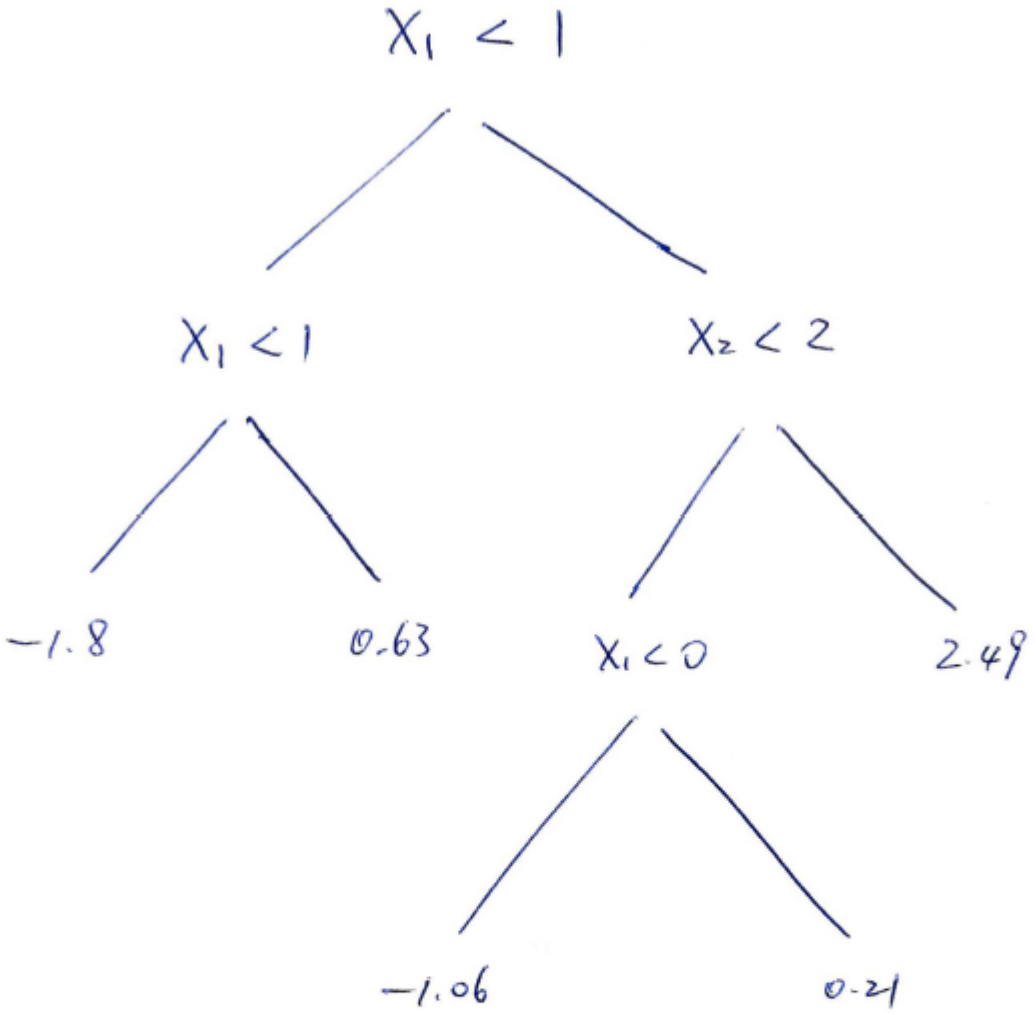
$$\min_{\beta} \|y-\hat{y}\|^2 + \lambda\|\beta\| \implies \min_{\beta} \sum_{i}^{n}(y_i - \sum_{j}^{p}\beta_j * x_{ij})^2 + \lambda\sum_{j}^{p}|\beta_j|$$

$$\frac{\partial L(\beta, \lambda)}{\partial \beta} = 0 \pm \lambda = 0$$

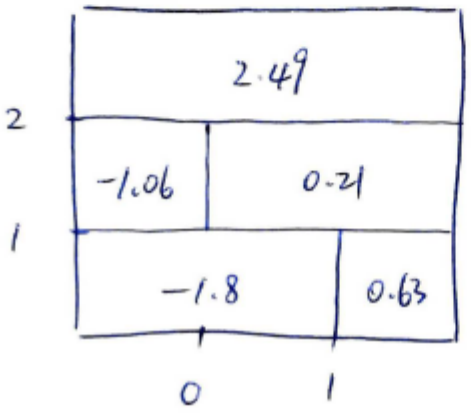
Therefore, since the function does not depended on β, the lasso coeficients $\hat{\beta}_1$ and $\hat{\beta}_2$ are not unique.

Problem 2

(1)



(2)



Caption for the picture.

Problem 3

```
##     G     G     G     G     R     R     R     R     R     R
## 0.10  0.15  0.20  0.20  0.55  0.60  0.60  0.65  0.70  0.75
```

For majority vote approach, `Red` has 6 votes, but `Green` has only 4, so it will be classified as `Red`

For average probability approach, P(Red|X) = 0.45, so it will be classified as `Green`

HW5 SML

Jie Li

April 14, 2019

```
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 3.5.3
```

```
library(purrr)
library(reshape2)
library(ggplot2)
setwd("C:/Users/jay48/OneDrive/Documents/work/Statistical ML/HW5")
```

implementing `Training`

```
train = function(x, w, y)
{
  # Assume: Decision Stump is finding the best feature split, y = {-1, 1}
  w = as.matrix(w)

  cost.mat = matrix(NA, 1200, 256)
  cost.opt = 10

  for(j in 1:dim(x)[2])
  {
    if(!is.factor(x[,j]))
    {
      for(i in 1:dim(x)[1])
      {
        yhat = 2*(x[,j] > x[i,j]) - 1
        cost = t(w) %*% (y != yhat) / sum(w)
        #cost.mat[i,j] = cost
        if(cost < cost.opt)
        {
          cost.opt = cost
          j.opt = j
          theta.opt = x[i,j]
        }
      }
    }
    else stop("Input design matrix is valid")
  }
  return(list(j = j.opt, theta = theta.opt, m = 1))
}
```

implementing `Classify`

```
classify = function(x, par)
{
  yhat = 2*(x[, par$j] > par$theta) - 1
  return(yhat)
}
```

implementing AdaBoost

```
AdaBoost = function(x, y, T)
{
  par.all = matrix(list())
  alpha = c()

  n = dim(x)[1]
  w = rep(1/n, n)

  for(i in 1:T)
  {
    par = train(x,w,y)
    label = classify(x, par)
    error = w %*% (y != label) / sum(w)
    alpha[i] = as.numeric(1/2 * log((1-error)/error))
    w = w*exp(alpha[i] * (y != label))
    par.all[[i]] = par
  }
  return(list(alpha = alpha, allPars = par.all))
}
```

implementing Agg_class

```
agg_class = function(x, model, T)
{
  alpha = model$alpha
  par.all = model$allPars

  yhat = 0
  for(i in 1:T)
  {
    yhat = yhat + alpha[i] * classify(x, par.all[[i]])
  }
  return(ifelse(yhat >0, 1, -1))
}
```

Loading training and testing data

```
# Read all images corresponding to digit "3"
zip.3 = read.table("./train_3.txt", header = FALSE, sep = ",")
zip.3 = cbind(zip.3, y=rep(-1, 658)) %>% data.frame()

# Read all images corresponding to digit "8"
zip.8 = read.table("./train_8.txt", header = FALSE, sep = ",")
zip.8 = as.matrix(cbind(zip.8, y=rep(1, 542)))

# Combine training set
data = data.frame(rbind(zip.3, zip.8))
data_x = data[, -257]
data_y = data[, 257]

# Read testing dataset
test = read.table("./zip_test.txt", header = F, sep = " ")
test = test[test[,1] %in% c(3,8), 1:257]
test_x = test[, 2:257]
test_y = ifelse(test[,1] == "3", -1, 1)
```

implementing Cross Validation process to tune num of trees

```
Sys.time()
```

```
## [1] "2019-04-16 23:29:34 EDT"
```

```
T = 20
index = sample(1:5, dim(data)[1], TRUE)
cv.error = matrix(NA, 5, T)

for(i in 1:5)
{
  training_x = data[index != i, -257]
  training_y = data[index != i, 257]

  validation_x = data[index == i, -257]
  validation_y = data[index == i, 257]

  n = dim(training_x)[1]
  w = rep(1/n, n)

  for(t in 1:T)
  {
    model = AdaBoost(training_x, training_y, t)
    result = agg_class(validation_x, model, t)
    cv.error[i,t] = sum(validation_y != result)/length(validation_y)
  }
}

avg.cv.error = apply(cv.error, 2, mean)
avg.cv.error
```

```
##  [1] 0.11790114 0.11790114 0.08615543 0.09983454 0.07988689 0.08461436
##  [7] 0.07755751 0.07953082 0.07936995 0.06962462 0.06957058 0.06733279
## [13] 0.06806794 0.06790065 0.06923184 0.06641725 0.06470342 0.06563848
## [19] 0.06459112 0.06137613
```

```
Sys.time()
```

```
## [1] "2019-04-17 02:14:57 EDT"
```

Choose the optimal num of trees, re-train the model with whole training data, and predict in test set

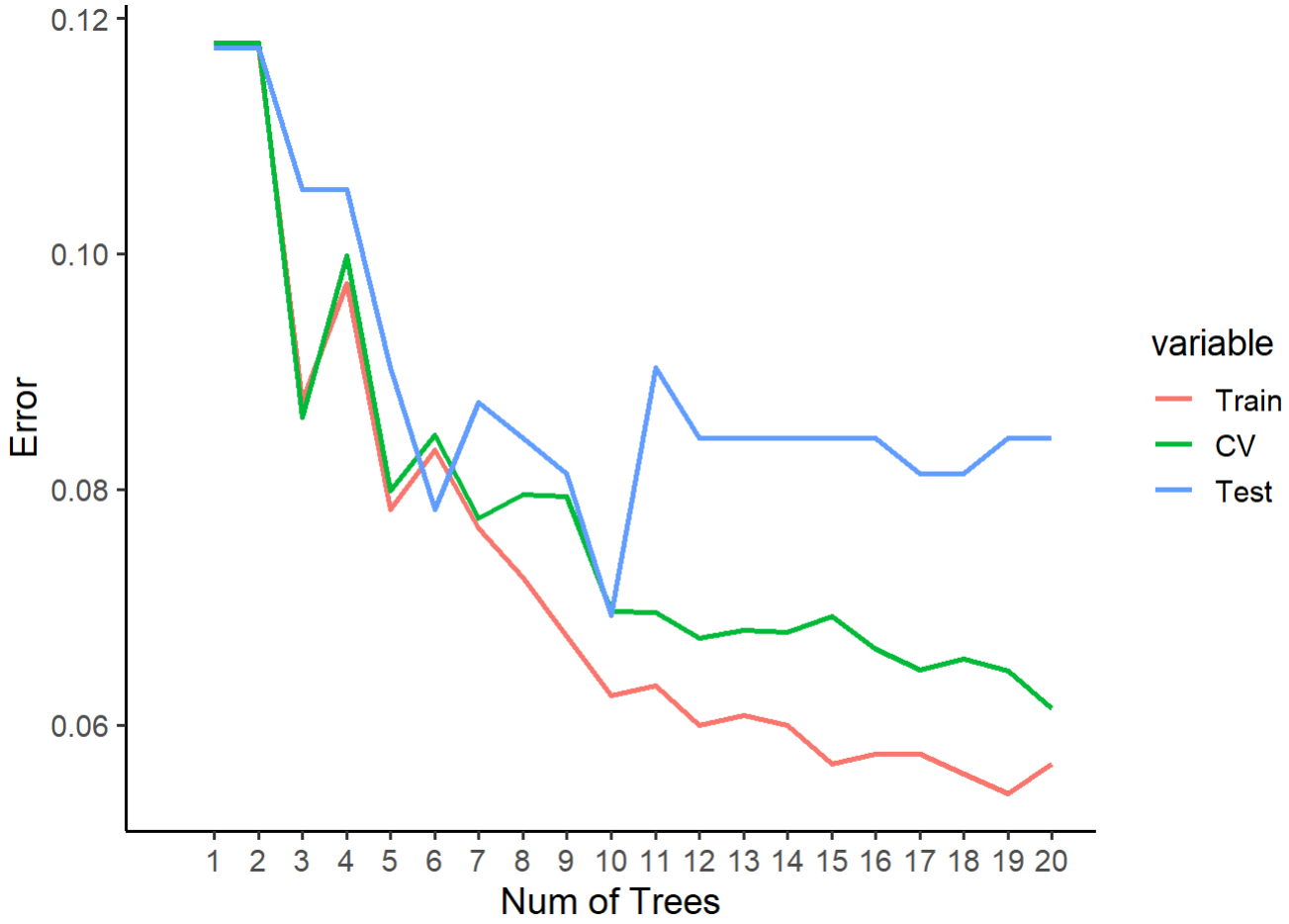
```
train.error = c()
test.error = c()

T.opt = which.min(avg.cv.error)

for(t in 1:T.opt)
{
  model = AdaBoost(data_x, data_y, t)
  result = agg_class(data_x, model, t)
  train.error[t] = sum(data_y != result)/length(data_y)

  pred = agg_class(test_x, model, t)
  test.error[t] = sum(test_y != pred)/length(test_y)
}

data.frame(index = 1:20, Train = train.error, CV = avg.cv.error, Test = test.error) %>% melt("index") %>%
  ggplot()+
  geom_line(aes(index, value, col = variable), size = 1)+
  scale_x_continuous("Num of Trees", breaks=1:20, limits=c(0,20))+
  ylab("Error")+
  theme_classic(14)
```


0.12
0.10
0.08
0.06
Error
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Num of Trees
variable
Train
CV
Test