Homework 5

Linsu Han (lh2910)

April 14, 2019

1

We are given:

$n = p = 2$
$x_{11} = x_{12}$
$x_{21} = x_{12}$
$y_1 + y_2 = x_{11} + x_{12} = x_{21} + x_{22} = 0$
$\hat{\beta}_0 = 0$

a)

$min[\sum_{i=1}^{n}(y_i - \hat{\beta}_0 - \sum_{j=1}^{p}\hat{\beta}_j x_j)^2 + \lambda \sum_{i=1}^{p}\hat{\beta}_i^2]$

$\Rightarrow min[(y_1 - \hat{\beta}_1 x_{11} - \hat{\beta}_2 x_{12})^2 + (y_2 - \hat{\beta}_1 x_{21} - \hat{\beta}_2 x_{22})^2 + \lambda(\hat{\beta}_1^2 + \hat{\beta}_2^2)]$

$\Rightarrow min[(y_1 - \hat{\beta}_1 x_1 - \hat{\beta}_2 x_1)^2 + (y_2 - \hat{\beta}_1 x_2 - \hat{\beta}_2 x_2)^2 + \lambda(\hat{\beta}_1^2 + \hat{\beta}_2^2)]$

b)

Take derivative of above, and set them equal to 0.

For $\hat{\beta}_1$,

$-2(y_1 - \hat{\beta}_1 x_1 - \hat{\beta}_2 x_1)x_1 - 2(y_2 - \hat{\beta}_1 x_2 - \hat{\beta}_2 x_2)x_2 + 2\lambda\hat{\beta}_1 = 0$

$\lambda\beta_1 + \beta_1(x_1^2 + x_2^2) - x_1 y_1 - x_2 y_2 + \beta_2(x_1^2 + x_2^2) = 0$

$\hat{\beta}_1 = \frac{x_1 y_1 + x_2 y_2 - \beta_2(x_1^2 + x_2^2)}{\lambda + (x_1^2 + x_2^2)}$

Similary for $\hat{\beta}_2$,

$\hat{\beta}_2 = \frac{x_1 y_1 + x_2 y_2 - \beta_1(x_1^2 + x_2^2)}{\lambda + (x_1^2 + x_2^2)}$

Thus,

$\hat{\beta}_1 = \hat{\beta}_2$

c)

$min[(y_1 - \hat{\beta}_1 x_{11} - \hat{\beta}_2 x_{12})^2 + (y_2 - \hat{\beta}_1 x_{21} - \hat{\beta}_2 x_{22})^2 + \lambda(|\hat{\beta}_1| + |\hat{\beta}_2|)]$

d)

The above minimization can be written as:

$min[(y_1 - \hat{\beta}_1 x_{11} - \hat{\beta}_2 x_{12})^2 + (y_2 - \hat{\beta}_1 x_{21} - \hat{\beta}_2 x_{22})^2]$

under the constraint: $|\hat{\beta}_1| + |\hat{\beta}_2| < c$

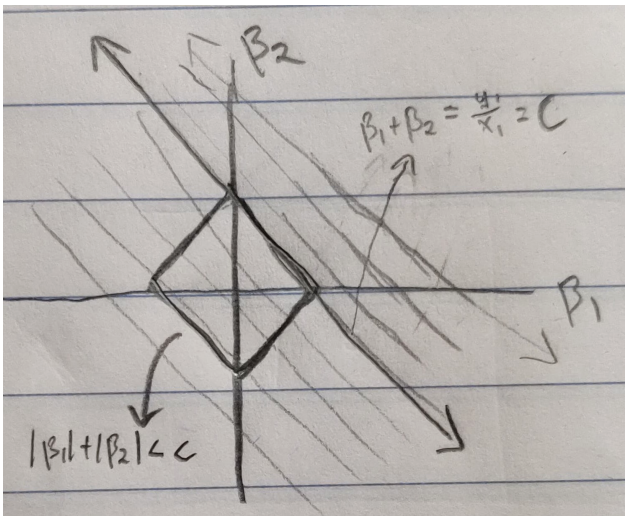
This is a lagrange optimization function.

Without the constraint the optimization would give:

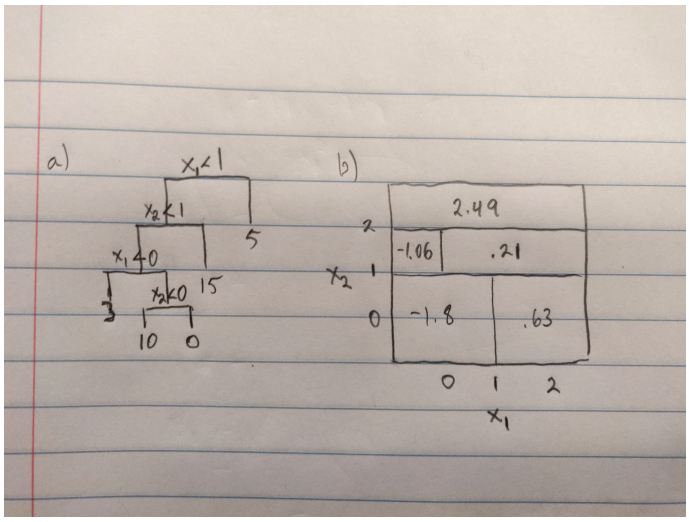
$\hat{\beta}_1 + \hat{\beta}_2 = y_1/x_1$

which are countour straight lines over the space β_1 β_2

Given the constraints, observe the contour lines have multiple points of intersections to the constraint (since they are parallel). Thus there are many possible solutions.


β_2
$\beta_1 + \beta_2 = \frac{y_i}{x_i} = C$
β_1
$|\beta_1| + |\beta_2| < C$

2


a)
$x_1 < 1$
$x_2 < 1$
5
$x_1 \geq 0$
15
$x_2 < 0$
10
0
b)
2.49
-1.06
.21
-1.8
.63
x_2
2
1
0
0
1
2
x_1

3

```
s = c(.1, .15, .2, .2, .55, .6, .65, .7, .75)
sum(s >= .5) # red voter
```

```
## [1] 5
```

```
sum(s < .5) # green votes
```

```
## [1] 4
```

Majority votes for red

```
mean(s)
```

```
## [1] 0.4333333
```

Average probability votes for green.

4

Data Initialization

```
path = "C:/Users/Linsu Han/Documents/[COLUMBIA]/STAT 5241 - Statistical Machine Learning/Homework/Homework 5"
setwd(path)
data3 = read.csv("train_3.txt", header=FALSE)
data8 = read.csv("train_8.txt", header=FALSE)
data = rbind(cbind(y=1, data3), cbind(y=-1, data8))

X = data[, -1]
y = data[, 1]
```

Trains one stump on data

```
train = function(X, w, y) { # X - data matrix, y - +/- labels, w - weights
  n = dim(X)[1]
  p = dim(X)[2]
  div = numeric(p) # divider
  sign = numeric(p)
  errs =  numeric(p)
  for (j in 1:p) {
    div_ = unique(X[, j])
    errs_ = numeric(length(div_))
    for (k in 1:length(div_)) {
      g = (X[, j] >  div_[k])*2 - 1 # classifies one-way (positive)
      errs_[k] = sum(w*(y != g)) # one-way errors
    }
    asdf = rbind(errs_, 1-errs_) # two-way errors (pos on row 1, neg on row 2)
    index = arrayInd(which.min(asdf), dim(asdf)) # index of smallest error
    div[j] = div_[index[2]]
    sign[j] = ifelse(index[1] == 1, 1, -1) # flips class sign if smallest error on row 2
    errs[j] = min(asdf)
  }

  a = which.min(errs)
  b = div[a]
  c = sign[a]

  return(list(j=a, theta=b, m=c))
}
```

Classifies a set given stump parameters

```
classify = function(X, pars) {
  j = pars$j
  theta = pars$theta
  sign = pars$m

  label = (2*(X[, j] > theta) - 1)*sign

  return(label)
}
```

Here we look at the parameters and error of one stump

```
w = rep(1/1200, 1200)
(pars = train(X, w, y)) # parameters
```

```
## $j
## [1] 167
##
## $theta
## [1] -1
##
## $m
## [1] -1
```

```
label = classify(X, pars)
sum(y != label)/length(label) # error
```

```
## [1] 0.1175
```

AdaBoost

```
adaboost = function(X, y, B) {
  alpha = numeric(B)
  allPars = list()
  n = dim(X)[1]
  w = rep(1/n, n)

  for (b in 1:B) {
    allPars[[b]] = train(X, w, y)
    wrong = (y != classify(X, allPars[[b]]))
    err = sum(w*wrong)
    alpha[b] = log((1-err)/err)
    w0 = w*exp(alpha[b]*wrong)
    w = w0/sum(w0)
  }
  return(list(alpha=alpha, allPars=allPars))
}
```

We find 20 stumps

```
B = 20
adab = adaboost(X, y, B)
```

A peek at the alpha values (stump voting weights)

```
adab$alpha
```

```
##  [1] 2.0163205 1.5003861 1.5767874 1.0596539 1.0899234 0.8583109 0.9778857
##  [8] 0.7670998 0.9206825 0.7812381 0.7457988 0.8846705 0.8407783 0.8282838
## [15] 0.7298701 0.8573321 0.7580157 0.7710288 0.6858910 0.8427834
```

Aggregate classifier on the alpha-weighted stumps

```
agg_class = function(X, alpha, allPars) {
  n = dim(X)[1]
  B = length(alpha)
  G = matrix(0, nrow=n, ncol=B)

  for (b in 1:B) {
    G[, b] = classify(X, allPars[[b]])
  }

  G = G %*% alpha
  vote = sign(G)

  return(vote)
}
```

Missclassification error of our aggregate classifier

```
labs = agg_class(X, adab$alpha, adab$allPars)
sum(y != labs)/length(y)
```

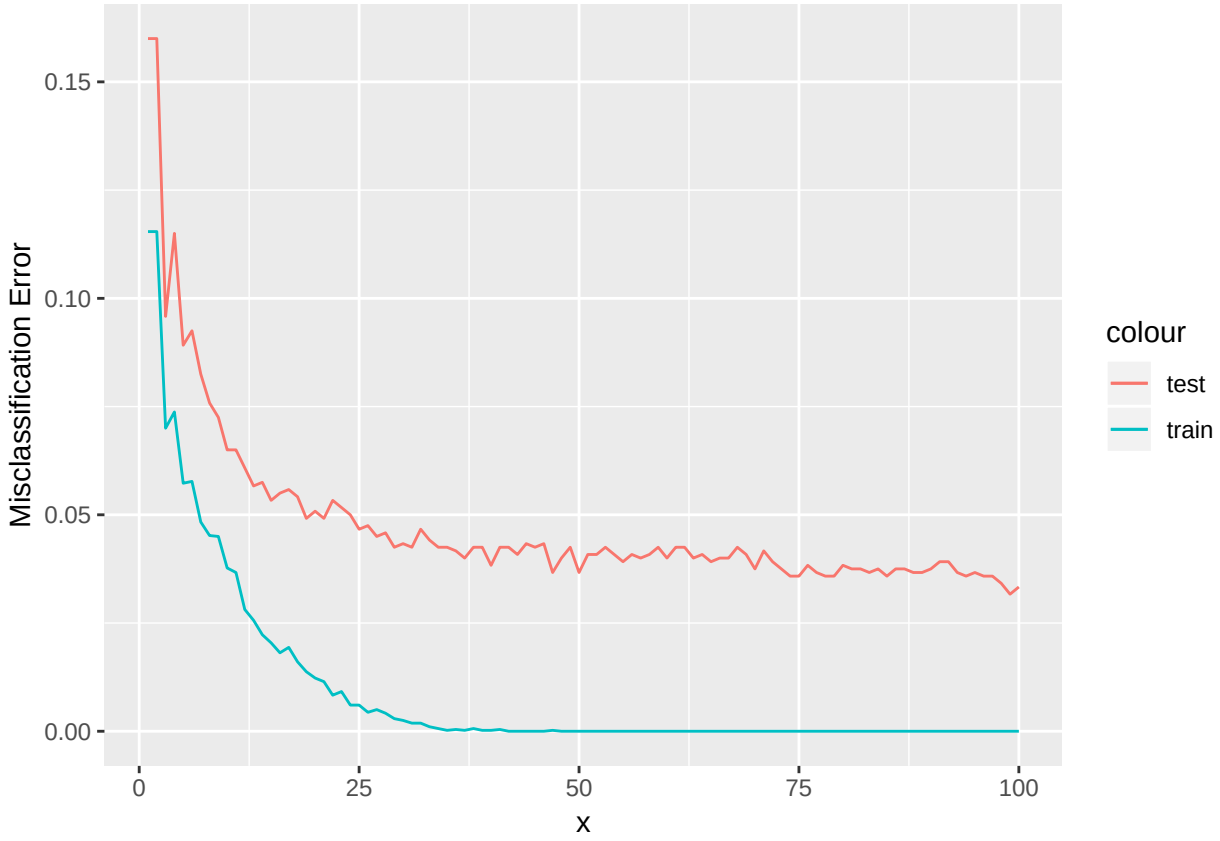
```
## [1] 0.015
```

Cross Validation

```
n = dim(X)[1]
shuffle = sample(n)
num_folds = 5
folds = list()

B = 100

train_errs_over_B = matrix(0, nrow=num_folds, ncol=B)
test_errs_over_B = matrix(0, nrow=num_folds, ncol=B)

for (k in 1:num_folds) {
  a = (k-1)*n/num_folds + 1
  b = k*n/num_folds
  k_slice = a:b

  X_train = X[-k_slice, ]
  y_train = y[-k_slice]

  X_test = X[k_slice, ]
  y_test = y[k_slice]

  adab = adaboost(X_train, y_train, B)

  train_errs = numeric(B)
  test_errs = numeric(B)

  for (i in 1:B) {
    slice = 1:i
    alpha = adab$alpha[slice]
    allPars = adab$allPars[slice]
    labs_train = agg_class(X_train, alpha, allPars)
    labs_test = agg_class(X_test, alpha, allPars)
    train_errs[i] = sum(y_train != labs_train)/length(y_train)
    test_errs[i] = sum(y_test != labs_test)/length(y_test)
  }
```

```
  train_errs_over_B[k, ] = train_errs
  test_errs_over_B[k, ] = test_errs
}
```

```
library(ggplot2)

x = 1:B
train_err = apply(train_errs_over_B, 2, mean)
test_err = apply(test_errs_over_B, 2, mean)

ggplot(mapping=aes(x=x)) +
  geom_line(mapping=aes(y=train_err, col="train")) +
  geom_line(mapping=aes(y=test_err, col="test")) +
  labs(y="Misclassification Error")
```



Observe that the training error approaches 0 as the number of stumps increase, while the testing error decreases and levels off to a constant value.