Computer lab 3 Block 2

Emil K Svensson

17 December 2016

Assignment 1

1.

```
library(pamr)
library(glmnet)
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.3.2
```

```
library(reshape2)
set.seed(12345)
data <- read.csv2("data.csv", encoding = "latin1")
data <- data[sample(nrow(data)),]
data$Conference <- as.factor(data$Conference)

train  <- data[1:45, ]
test   <- data[46:64, ]

y <- as.factor(train$Conference)
x <- t(train[-which(colnames(data) == "Conference")])

TRAIN <- list(x = x, y = y,geneid = as.character( 1:nrow(x) ), genenames = rownames(x) )

y1 <- as.factor(test$Conference)
x1 <- t(test[-which(colnames(test) == "Conference")])

TEST <- list(x = x1, y = y1,geneid = as.character( 1:nrow(x1) ), genenames = rownames(x1) )

# Cross Validation for the shrunken centroid
 model <- pamr.train(TRAIN, threshold = seq(0,4, 0.1))
```

```
## 1234567891011121314151617181920212223242526272829303132333435363738394041
```

```
cvmodel=pamr.cv(model,TRAIN)
```

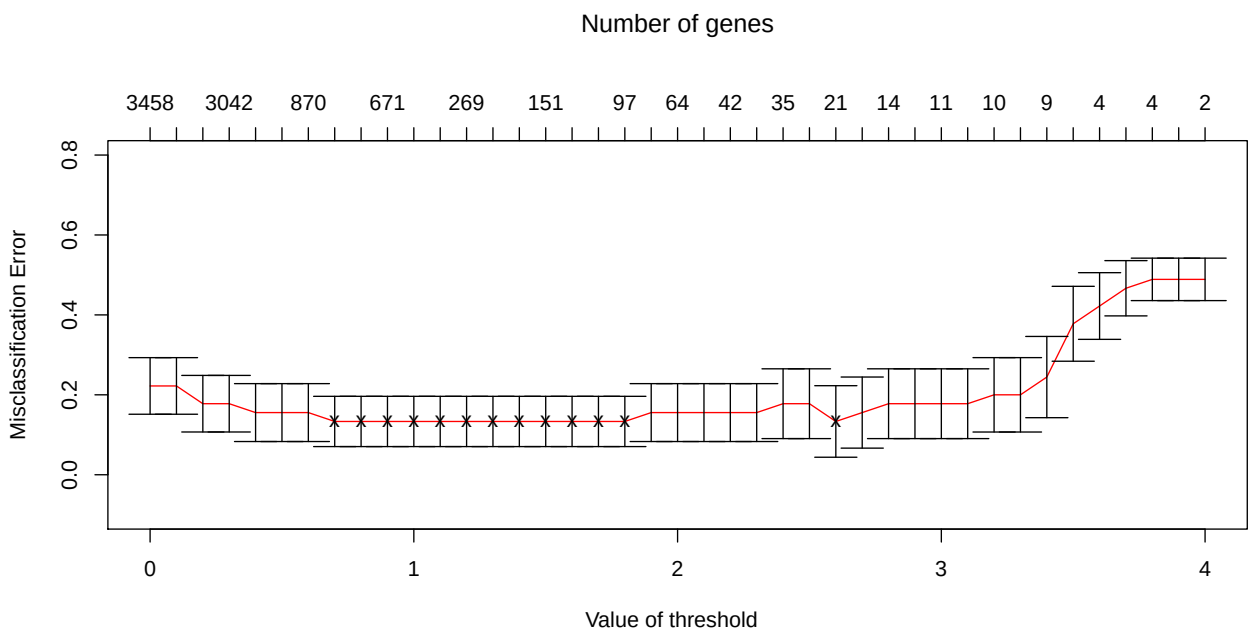
```
## 12Fold 1 :1234567891011121314151617181920212223242526272829303132333435363738394041
## Fold 2 :1234567891011121314151617181920212223242526272829303132333435363738394041
## Fold 3 :1234567891011121314151617181920212223242526272829303132333435363738394041
## Fold 4 :1234567891011121314151617181920212223242526272829303132333435363738394041
## Fold 5 :1234567891011121314151617181920212223242526272829303132333435363738394041
## Fold 6 :1234567891011121314151617181920212223242526272829303132333435363738394041
## Fold 7 :1234567891011121314151617181920212223242526272829303132333435363738394041
## Fold 8 :1234567891011121314151617181920212223242526272829303132333435363738394041
## Fold 9 :1234567891011121314151617181920212223242526272829303132333435363738394041
## Fold 10 :1234567891011121314151617181920212223242526272829303132333435363738394041
```

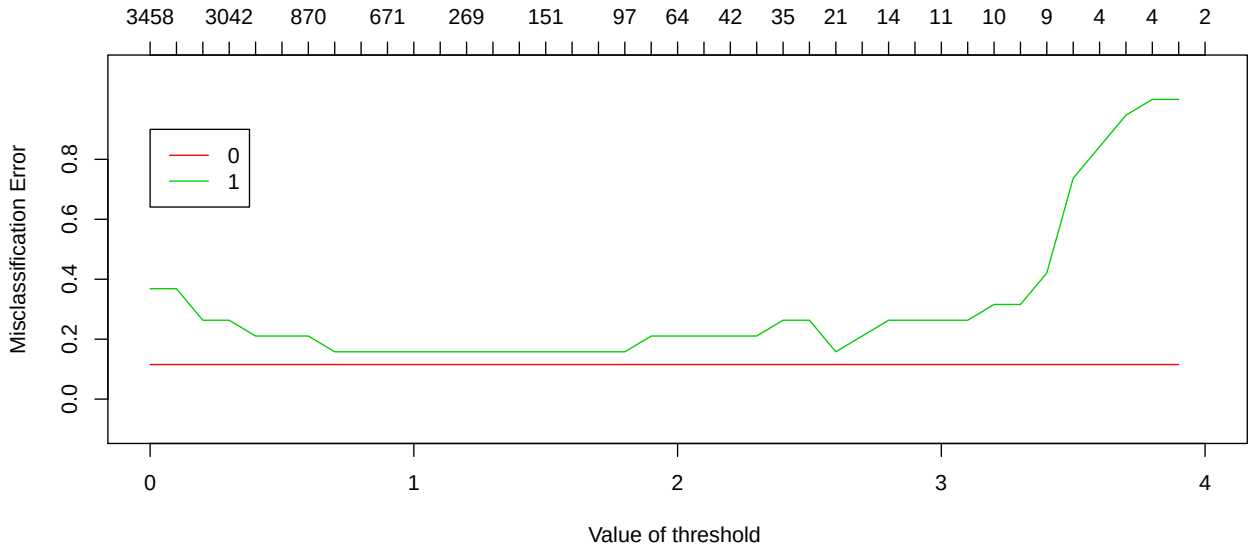
Something something is the best plot since it has the lowest error rate while having the lowest number of features.

```
print(cvmodel) #13 1.5       314    4 ,  mao 314 variabler vill vi ha
```

```
## Call:
## pamr.cv(fit = model, data = TRAIN)
##    threshold nonzero errors
## 1  0.0       3458    10
## 2  0.1       3428    10
## 3  0.2       3110    8
## 4  0.3       3042    8
## 5  0.4       3025    7
## 6  0.5       1977    7
## 7  0.6        870    7
## 8  0.7        850    6
## 9  0.8        673    6
## 10 0.9        671    6
## 11 1.0        295    6
## 12 1.1        290    6
## 13 1.2        269    6
## 14 1.3        234    6
## 15 1.4        154    6
## 16 1.5        151    6
## 17 1.6        128    6
## 18 1.7        100    6
## 19 1.8         97    6
## 20 1.9         73    7
## 21 2.0         64    7
## 22 2.1         57    7
## 23 2.2         42    7
## 24 2.3         37    7
## 25 2.4         35    8
## 26 2.5         23    8
## 27 2.6         21    6
## 28 2.7         20    7
## 29 2.8         14    8
## 30 2.9         12    8
## 31 3.0         11    8
## 32 3.1         10    8
## 33 3.2         10    9
## 34 3.3         10    9
## 35 3.4          9    11
## 36 3.5          4    17
## 37 3.6          4    19
## 38 3.7          4    21
## 39 3.8          4    22
## 40 3.9          3    22
## 41 4.0          2    22
```

```
pamr.plotcv(cvmodel)
```

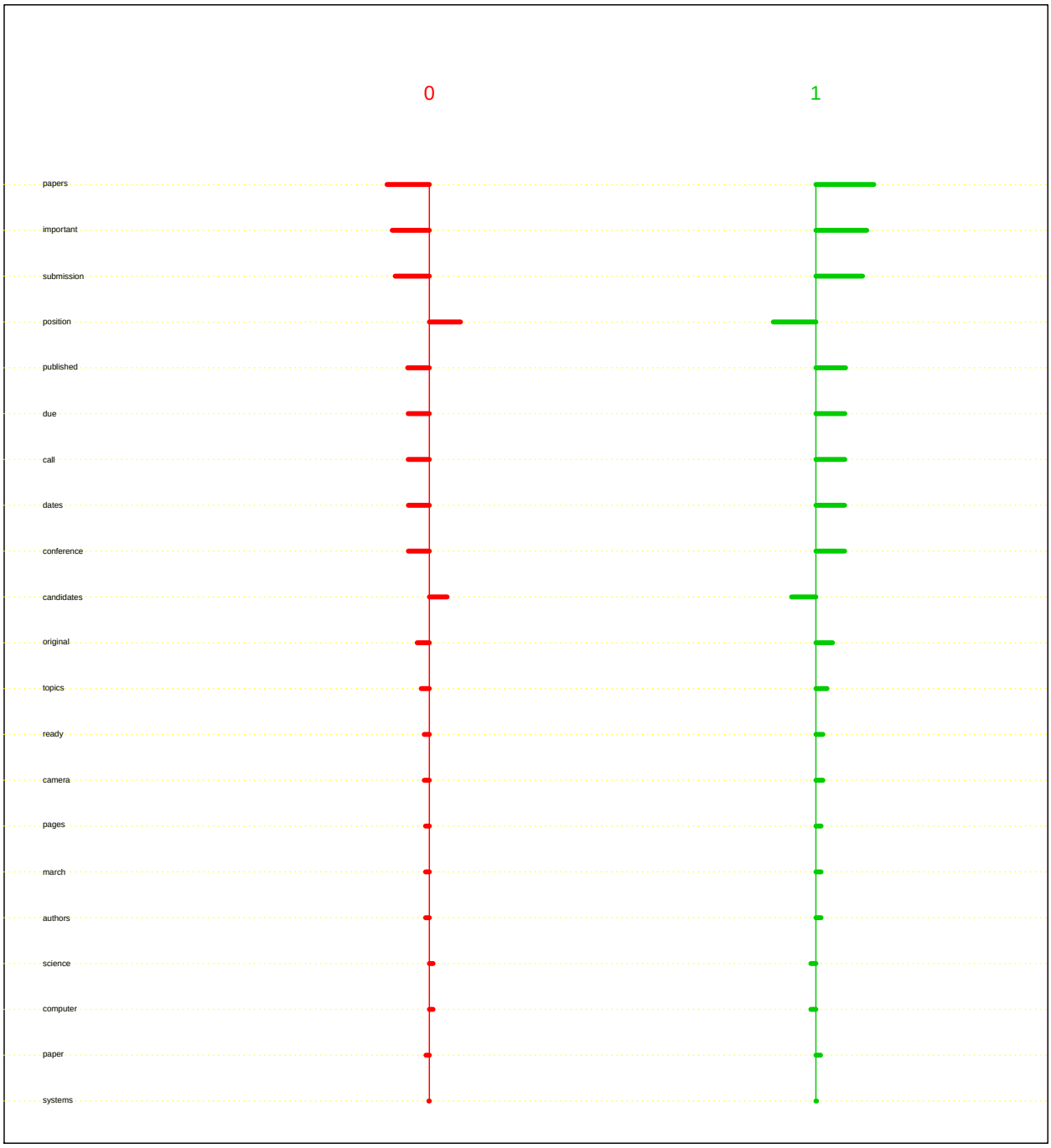





```
# Training a model with the best threshold from the cross validation
modcv <- pamr.train(TRAIN, threshold = 2.6)
```

```
## 1
```

```
pamr.plotcen(modcv, TRAIN, threshold = 2.6)
```



The length of the bars represent the weights for the predictions. So for predicting a 1 i.e. a mail regarding conferences large positive weights are given to words as papers, submissions and important. A large negative weight is given to the word position when trying to predict a mail.

```
crossed<-pamr.listgenes(modcv,TRAIN,threshold=2.6)
```

```
##       id   0-score 1-score
##  [1,] 3036 -0.219  0.2997
##  [2,] 2049 -0.1922 0.263
##  [3,] 4060 -0.1765 0.2415
##  [4,] 3187 0.1609  -0.2202
```

```
##  [5,] 3364 -0.1122 0.1535
##  [6,] 596  -0.1094 0.1498
##  [7,] 1262 -0.1094 0.1498
##  [8,] 869  -0.1085 0.1485
##  [9,] 1045 -0.1085 0.1485
## [10,] 607  0.0919  -0.1258
## [11,] 2990 -0.0629 0.0861
## [12,] 4282 -0.0421 0.0577
## [13,] 599  -0.0265 0.0362
## [14,] 3433 -0.0265 0.0362
## [15,] 389  -0.0195 0.0267
## [16,] 2588 -0.0195 0.0267
## [17,] 3022 -0.0195 0.0267
## [18,] 850  0.0191  -0.0261
## [19,] 3725 0.0191  -0.0261
## [20,] 3035 -0.0175 0.0239
## [21,] 4129 -0.0026 0.0035
```

```
cat( paste( colnames(data)[as.numeric(crossed[,1])], collapse='\n' ) )[1:10]
```

```
## papers
## important
## submission
## position
## published
## call
## due
## conference
## dates
## candidates
## original
## topics
## camera
## ready
## authors
## march
## pages
## computer
## science
## paper
## systems
```

```
## NULL
```

These are the variables chosen by the model. Submission, papers, conference and publish are among these variables, seems like resonable words in a email from a conference.

```
table(pamr.predict(modcv,TEST$x,threshold = 2.6),TEST$y)
```

```
##
##    0 1
##  0 9 2
##  1 0 8
```

Seems like a low error rate ca 10 % considering the number of observations the model used.

2.

a)

```
elasticNet<- cv.glmnet(x = as.matrix(train[,-which(colnames(train) == "Conference")]), y = train$Confer
       family = "binomial", alpha = 0.5)
mycoeffs <- coefficients(elasticNet)[-1,]
cat(paste0(names(mycoeffs[mycoeffs != 0]),"\n"))
```

```
## call
##  candidates
##  conference
##  dates
##  due
##  important
##  papers
##  position
##  published
##  submission
```

These are the coefficients chosen by the elastic net, similar to the previous model but fewer number of variables. Submission, position and papers are still here for example.

```
cat(elasticNet$name)
```

```
## Binomial Deviance
```

This is the penalty factor that the cross validation chooses.

```
elsNet<- predict(elasticNet, s = elasticNet$lambda.min, newx = as.matrix(test[,-which(colnames(test) ==
table(ifelse(elsNet > 0, 1,0),test$Conference)
```

```
##
##     0 1
##   0 9 2
##   1 0 8
```

One observation more is missclassified here.

b)

```
library(kernlab)
filter <- ksvm(Conference~.,data=train,kernel="vanilladot")
```

```
##  Setting default kernel parameters
```

```
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.
```

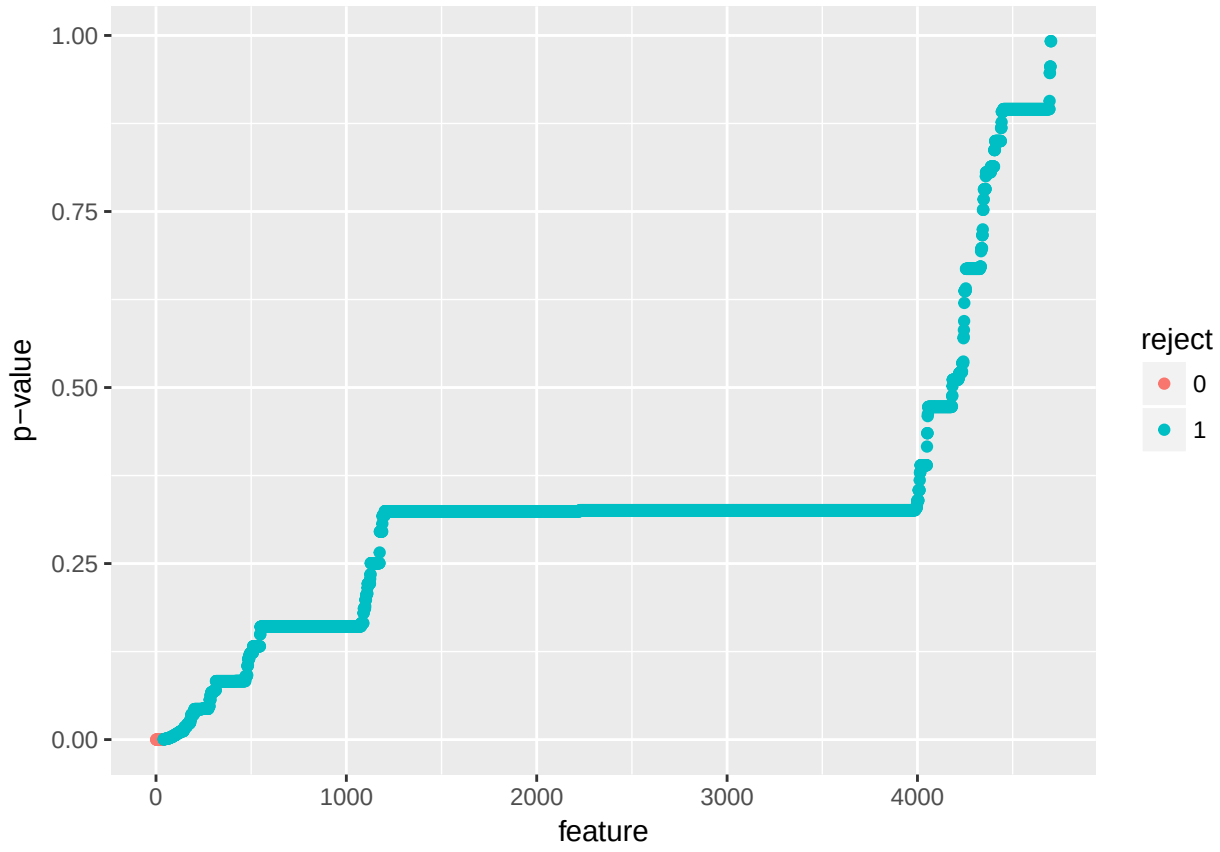
```
table(     predict(filter,test[,-ncol(test)])    ,test$Conference)
```

```
##
##     0 1
##   0 9 1
##   1 0 9
```

1/19 as missclassification error is the best result so far. The number of support vectors choosen were 44, but the number of features are 4702, since svm only uses the most important vectors it uses all features avaliable in the model.

3

```r
#extracts the y-variable and remove the factors
Conference.t.test<- as.numeric(as.character(data$Conference))

#remove all factors and put them in a data.frame since the sapply transposes.
data.t.test <- t(apply(data,1,FUN = function(x) as.numeric(as.character(x))))
data.t.test <- as.data.frame(data.t.test)
colnames(data.t.test) <- colnames(data)

#calculates the t-test for all features vs Conference and extracts the p-value
# and puts the feature name and its p-value in a matrix
pvalues<-matrix(ncol = 2, nrow = (ncol(data)-1))
for (i in 1:(ncol(data)-1) ){
  pvalues[i,]<- c(colnames(data.t.test)[i] ,
                  t.test(data.t.test[,i]~Conference.t.test ,alternative = "two.sided" )$p.value)
}

#tidying the data up a bit and transforms it in to a data.frame
pvalues<-as.data.frame(pvalues)
colnames(pvalues) <- c("feature","pvalue")
pvalues$pvalue <- as.numeric(as.character(pvalues$pvalue))


#setting a alpha
alph <- 0.05
pvalues <- pvalues[order(pvalues$pvalue),]
pvalues$reject <- 1:nrow(pvalues)

# for (i in 1:nrow(pvalues)){
#   pvalues$reject[i]<-(ifelse(alph*(i/nrow(pvalues)) > pvalues$pvalue[i] , 0,1))
#
# }
pvalues$reject<-(ifelse(alph*(1:nrow(pvalues)/nrow(pvalues)) > pvalues$pvalue, 0,1))


#Making a plot
pvalues$feature <- as.factor(pvalues$feature)
pvalues$reject <- as.factor(pvalues$reject)

ggplot(data = pvalues[1:4702,], aes(x = 1:4702,y=pvalue, col = reject)) + geom_point() + labs(x="featur
```




```
cat(paste("Number of features kept:",nrow(pvalues[pvalues$reject == 0,]),"\n The features kept were:"))
```

```
## Number of features kept: 39
##  The features kept were:
```

```
cat(paste0(pvalues$feature[pvalues$reject == 0],"\n"))
```

```
## papers
##  submission
##  position
##  published
##  important
##  call
##  conference
##  candidates
##  dates
##  paper
##  topics
##  limited
##  candidate
##  camera
##  ready
##  authors
##  phd
##  projects
##  org
##  chairs
##  due
```

```
##  original
##  notification
##  salary
##  record
##  skills
##  held
##  team
##  pages
##  workshop
##  committee
##  proceedings
##  apply
##  strong
##  international
##  degree
##  excellent
##  post
##  presented
```

Assignment 2

```
set.seed(1234567890)
spam <- read.csv2("spambase.csv")
ind <- sample(1:nrow(spam))
spam <- spam[ind,c(1:48,58)]
h <- 1
betai <- -0.5  # Your value here
  Mi <- 20 # Your value here
  N <- 500 # number of training points

  if(all(levels(factor(spam$Spam)) == c(-1,1)) | all(levels(factor(spam$Spam)) == c(1,-1)) ){

  }else{
    spam$Spam[spam$Spam == 0] <- (-1)
  }

gaussian_k <- function(x, h = 1) { # Gaussian kernel
  return(exp(-(x^2/2*h^2)))
}


#a_n = C- m_n



SVM <- function(sv,i,M = Mi, beta = betai){




   step4 <- function(dataindex){
```

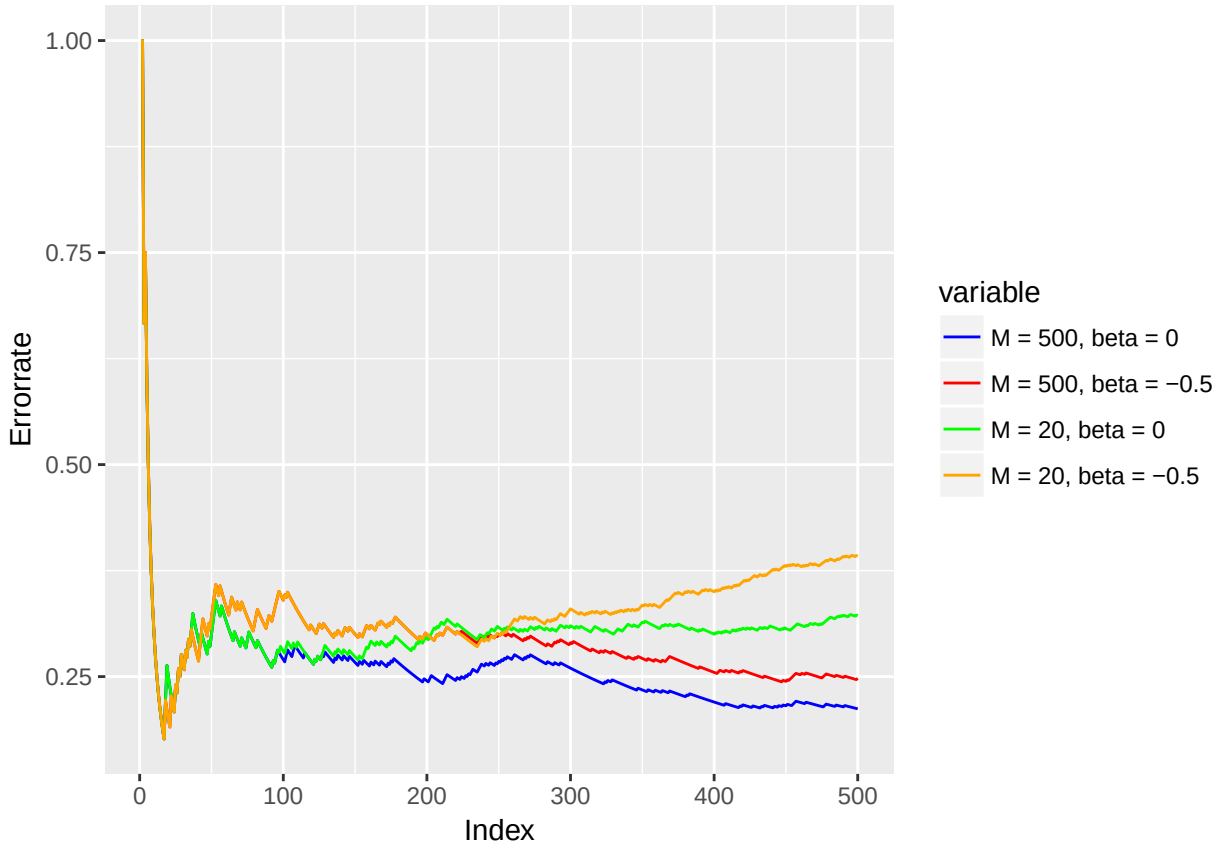
```
    b<-0
    distelement<-as.matrix(dist(rbind(dataindex,spam[sv,-ncol(spam)])))[-1,1]

    return(sum(spam[sv,"Spam"]* gaussian_k(distelement)) + b )
  }

  step8 <- function(SV){
    yxm <- c()
    res<-c()
    for (m in SV) {
      distelement <- as.matrix(dist(rbind(spam[m,-ncol(spam)],spam[m,-ncol(spam)])))
      yxm <-  sum(spam[m,"Spam"]* gaussian_k(distelement[-1,1]))
      res <-  c(res,spam[m,"Spam"]*(step4(dataindex = spam[m,-ncol(spam)]) - yxm))
    }
    return(which.max(res))
  }

 b <- 0
 errors <- 1
 errorrate <- vector(length = N)
 errorrate[1] <- 1
 sv <- c(1)
 s4<-c()

 for(i in 2:N) {

   s4<-step4(dataindex = spam[i,-ncol(spam)])

   if(spam[i,"Spam"]*s4 < 0){
     errors <- errors + 1
   }

   if(spam[i,"Spam"]*s4 < beta){
     sv[length(sv)+1] <- i
   }

   if (length(sv) > M  ){
       sv <- sv[-step8(SV = sv)]
     }
   errorrate[i] <- errors / i
   }
 #plot(errorrate)
 length(sv)
 errorrate[N]
 return(errorrate)


}


#system.time()

 system.time(svm1<-SVM(M = 500, beta = 0))
```

```
##    user  system elapsed
##   2.115   0.032   2.189
```

```
svm2<-SVM(M = 500, beta = -0.05)
svm3<-SVM(M = 20, beta = 0)
svm4<-SVM(M = 20, beta = -0.05)
erry<-melt(data.frame(svm1=svm1,svm2=svm2,svm3=svm3,svm4=svm4))
```

```
## No id variables; using all as measure variables
```

```
erry$index <- rep(1:length(svm1),4)
ggplot(data = erry, aes(x=index, y = value, color = variable)) + geom_line() +
  labs(x="Index",y = "Errorrate") + scale_color_manual(
    labels=c("M = 500, beta = 0","M = 500, beta = -0.5",
             "M = 20, beta = 0","M = 20, beta = -0.5"),
              values = c("blue", "red","green","orange"))
```



The M = 500 with beta = 0 is better than the one with -0.5 since we don't accept any errors with beta = 0. But since we don't pass the max number of support vectors and have to replace them it only will accept more errors and not improve the model..

The M = 20 with beta = 0 is the slowest since it takes in and evaluates the support vectors and their contributions the most times since it is set to don't accept any errors. This compared to the M = 20, beta = -0.5 that accepts some errors and dont have to evaluate that many support vectors contribution and instead accepts some errors. This seems to be a better approach this time since the M = 20, beta = 0 doesn't add worse support vectors as the one with beta = -0.5 does.