Introduction to Machine Learning

Lab 3 Block 2

Rasmus Holm

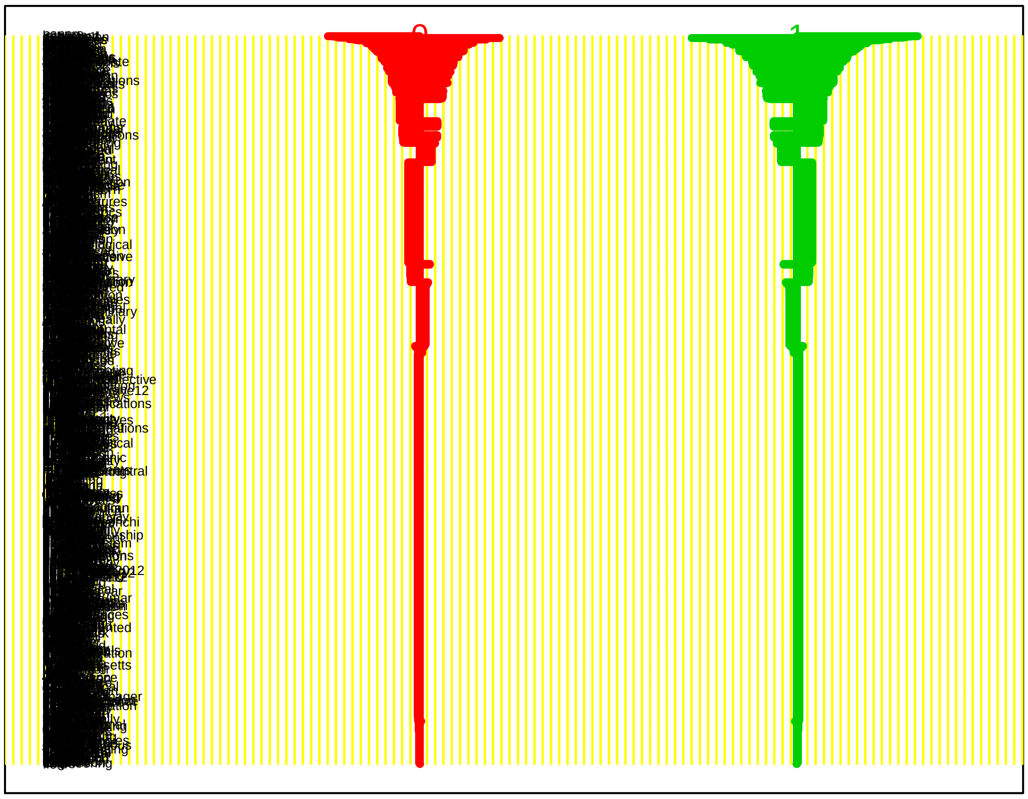
2016-12-14

Contents

Assignment 1

1

```
#> Threshold: 0.5
#> Size: 1979
#> Classification Error: 0.05
#> Top 10 features
#> papers
#> important
#> submission
#> due
#> published
#> position
#> call
#> conference
#> dates
#> candidates
```



2

Elastic Net

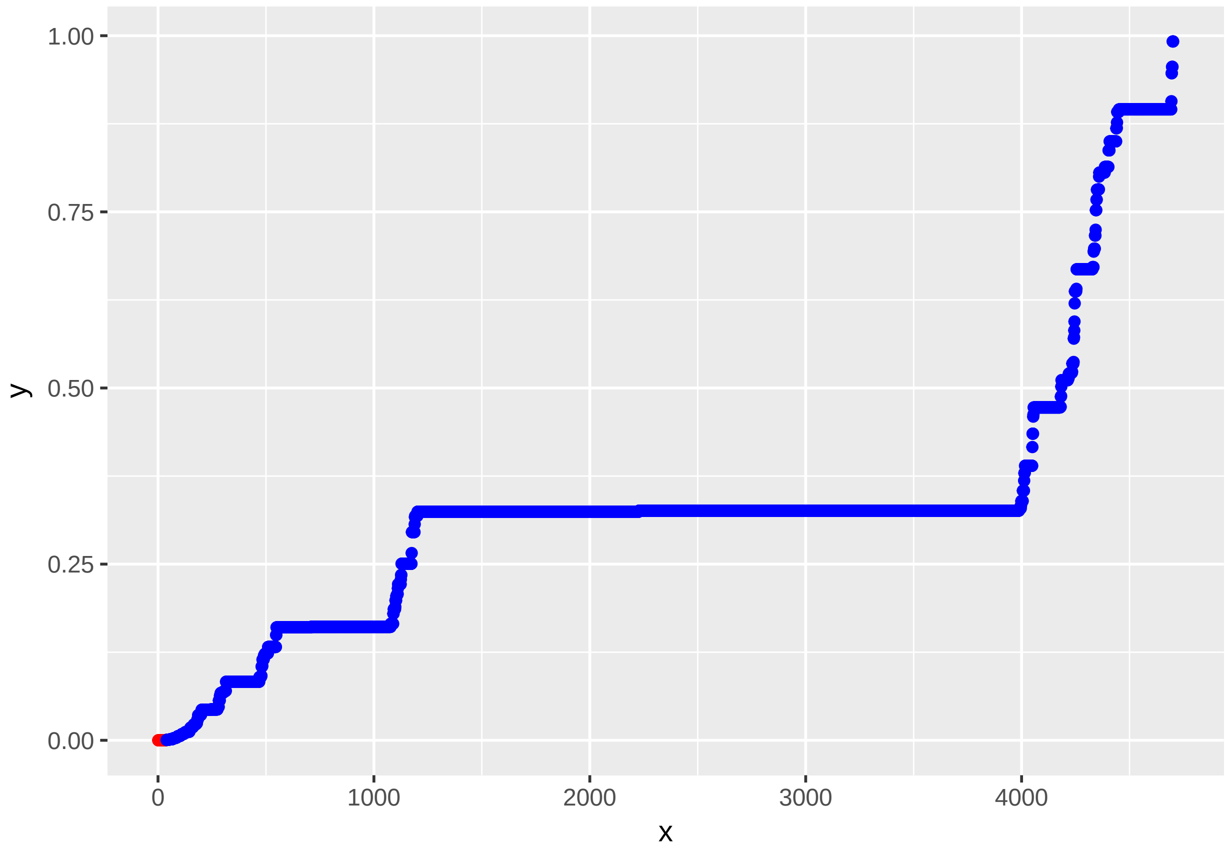
```
#> Penalty Deviance
#> Lambda: 0.131162838159315
#> Size: 38
#> Classification Error: 0.15
```

Support Vector Machine

```
#> Size: 43
#> Classification Error: 0.05
```

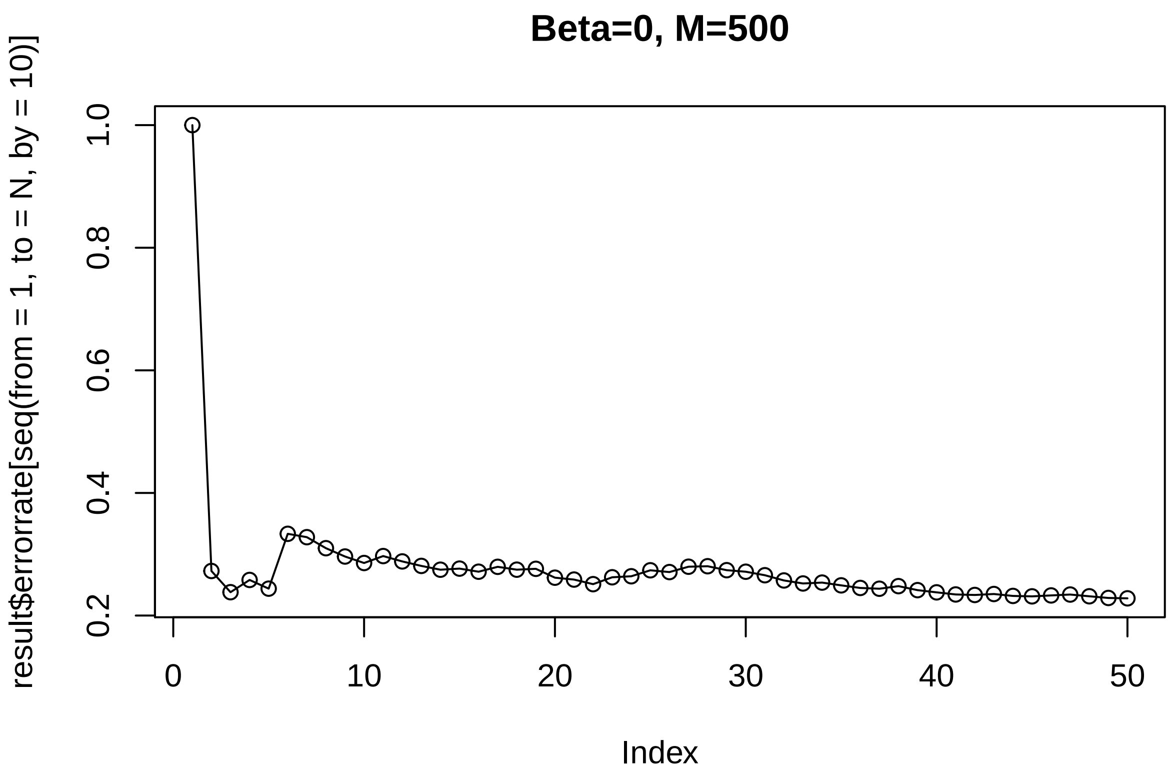
3

```
#> Top 10 features
#> papers
#> submission
#> position
#> published
#> important
#> call
#> conference
#> candidates
#> dates
#> paper
```

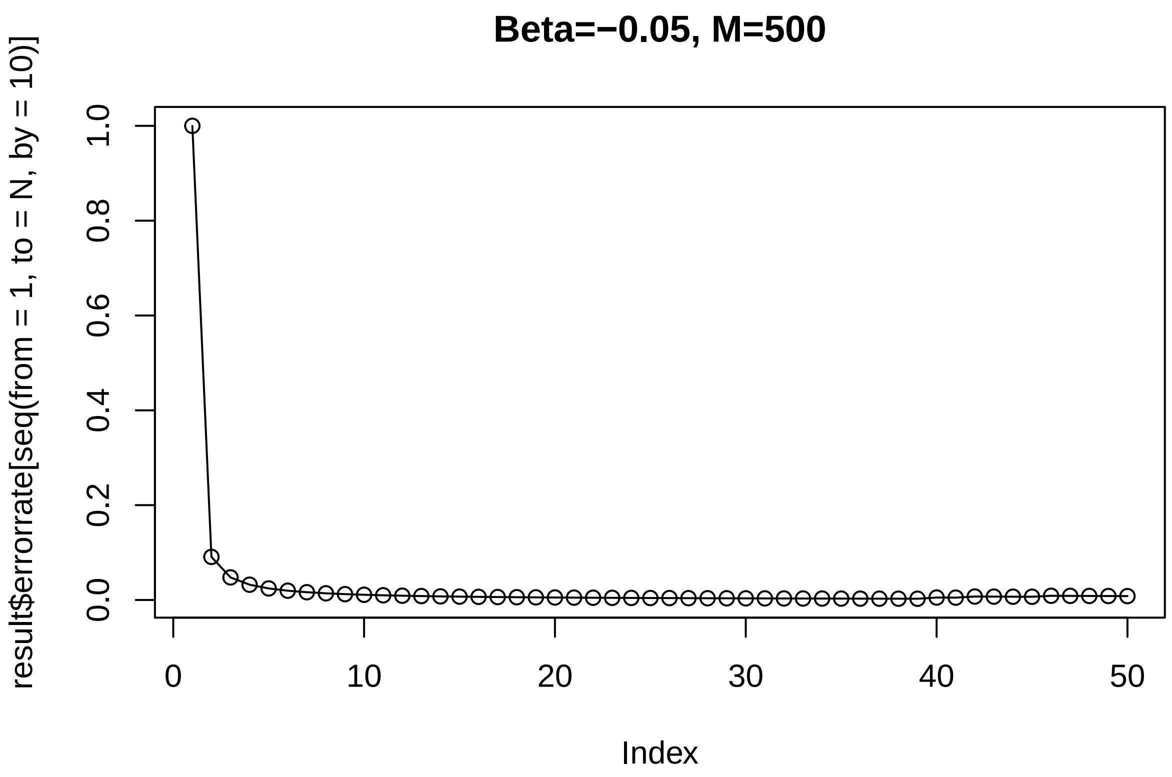


Assignment 2

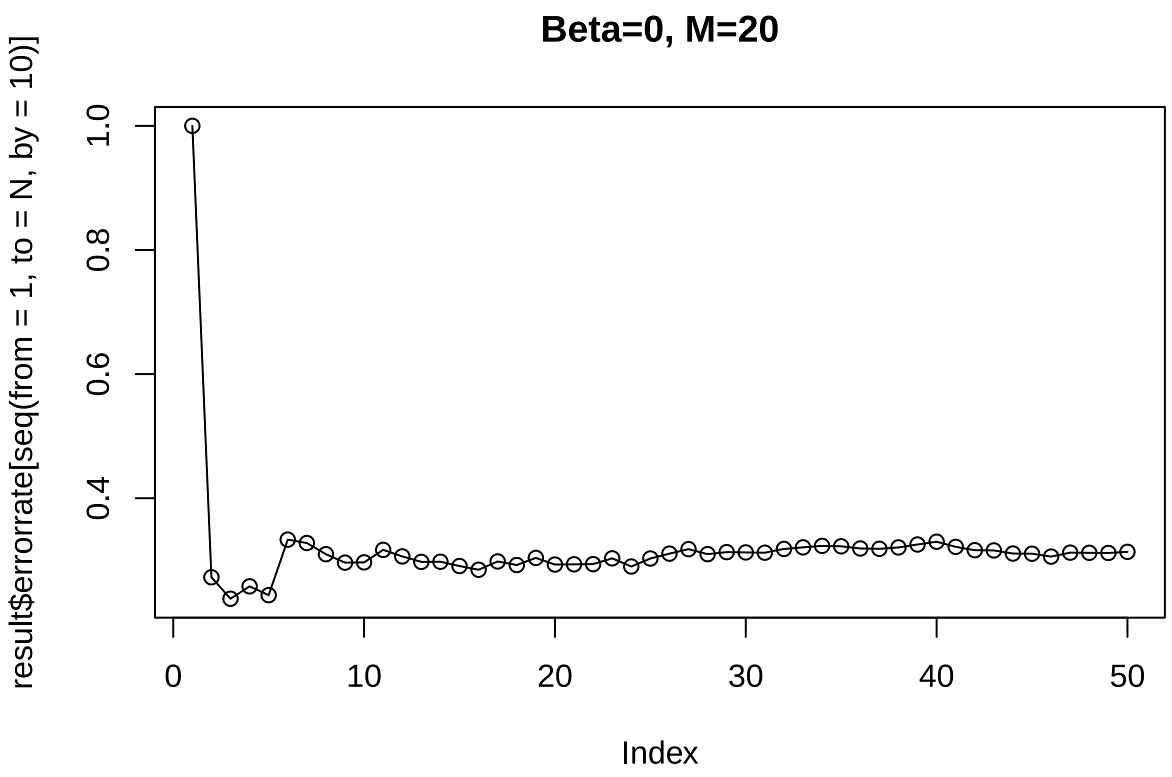
```
#> Number of Suppoer Vectors: 112
```



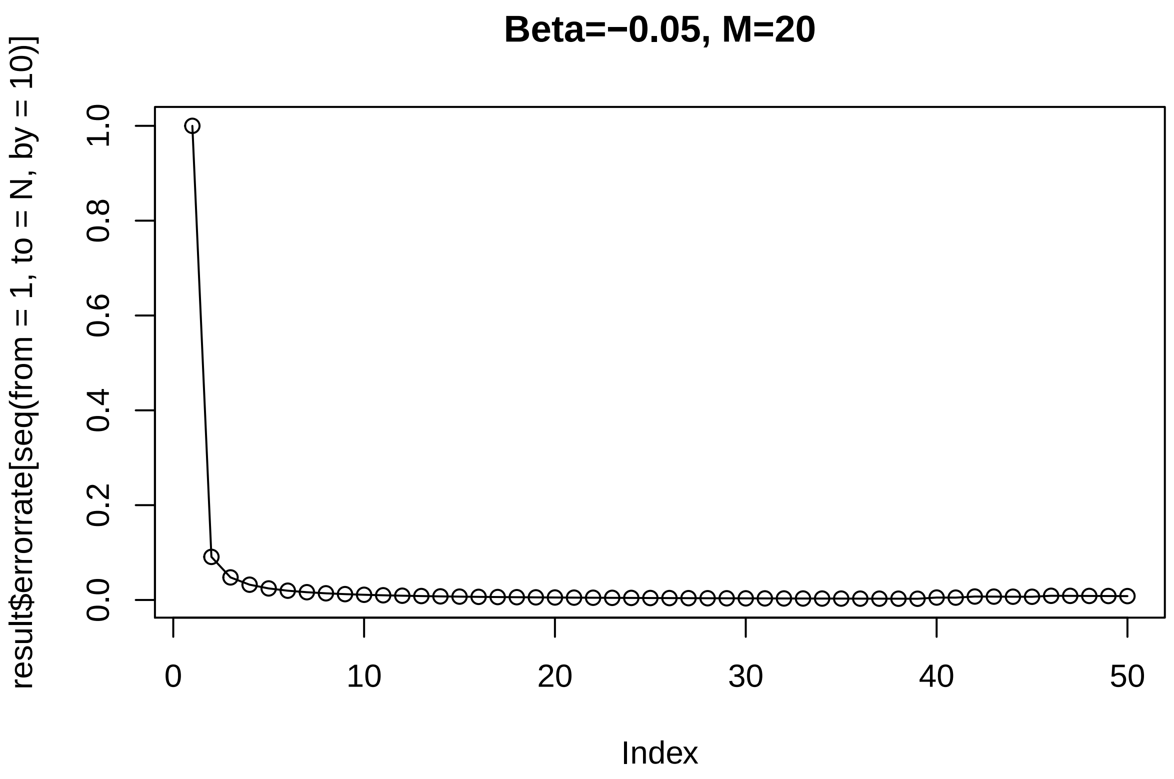
```
#> Number of Suppoer Vectors: 4
```



```
#> Number of Suppoer Vectors: 20
```



```
#> Number of Suppoer Vectors: 4
```


Beta=−0.05, M=20
result$errorrate[seq(from = 1, to = N, by = 10)]
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40
50
Index

Appendix

Code for Assignment 1

```r
library(pamr)
library(glmnet)
library(kernlab)
library(ggplot2)

data <- read.csv("../data/data.csv", sep=";", header=TRUE,
                 stringsAsFactors=FALSE, encoding="latin1")
rownames(data) <- 1:nrow(data)

set.seed(12345)
train_idx <- sample(nrow(data), size=floor(nrow(data) * 7 / 10))
train <- data[train_idx,]
test <- data[-train_idx,]

x <- t(train[, -ncol(data)])
y <- train[, ncol(data)]

x_test <- t(test[, -ncol(data)])
y_test <- test[, ncol(data)]

set.seed(12345)

nsc_data <- list(x=x, y=as.factor(y),
                 geneid=as.character(1:nrow(x)),
                 genenames=rownames(x))
model <- pamr.train(nsc_data, threshold=seq(0,4, 0.1))
cvmodel <- pamr.cv(model, nsc_data)

optimal_threshold <- cvmodel$threshold[which.min(cvmodel$error)]
optimal_size <- cvmodel$size[which.min(cvmodel$error)]

class_error <- 1 - (sum(pamr.predict(model, x_test,
                                     threshold=optimal_threshold) == y_test) /
                      length(y_test))
genes <- pamr.listgenes(model, nsc_data, threshold=optimal_threshold)
cat(paste("Threshold:", optimal_threshold))
cat(paste("Size:", optimal_size))
cat(paste("Classification Error:", class_error))
cat("Top 10 features")
cat(paste(colnames(data)[as.numeric(genes[,1])][1:10], collapse='\n' ) )
pamr.plotcen(model, nsc_data, threshold=optimal_threshold)
## pamr.plotcv(cvmodel)

set.seed(12345)

alpha <- 0.5
fit <- cv.glmnet(x=t(x), y=y, alpha=alpha, family="binomial")

optimal_lambda <- fit$lambda[which.min(fit$cvm)]
```

```
optimal_size <- fit$nzero[which.min(fit$cvm)]
penalty <- strsplit(fit$name, " ")[[1]][2]

class_error <- 1 - (sum(predict(fit, t(x_test), type="class") == y_test) /
                    length(y_test))
cat(paste("Penalty", penalty))
cat(paste("Lambda:", optimal_lambda))
cat(paste("Size:", optimal_size))
cat(paste("Classification Error:", class_error))

set.seed(12345)

fit <- ksvm(x=t(x), y=y, kernel="vanilladot",
            type="C-svc", cross=10, scale=FALSE)

optimal_size <- fit@nSV
class_error <- 1 - (sum(predict(fit, t(x_test)) == y_test) / length(y_test))
cat(paste("Size:", optimal_size))
cat(paste("Classification Error:", class_error))

benjamini_hochberg <- function(x, y, alpha) {
    pvalues <- apply(x, 2, function(feature) {
        t.test(feature ~ y, alternative="two.sided")$p.value
    })
    m <- length(pvalues)

    sorted <- sort(pvalues)
    values <- 1:m * alpha / m

    L <- which.min(sorted < values) - 1
    mask <- sorted <= sorted[L]
    list(mask=mask, pvalues=sorted, features=colnames(x)[order(pvalues)][mask])
}

result <- benjamini_hochberg(x=data[,-ncol(data)], y=data[, ncol(data)], alpha=0.05)
cat("Top 10 features")
cat(paste(result$features[1:10], collapse='\n' ) )
ggplot() +
    geom_point(data=data.frame(x=1:length(result$features),
                               y=result$pvalues[result$mask]),
               aes(x=x, y=y), col="red") +
    geom_point(data=data.frame(x=((length(result$features) + 1):(ncol(data) -1)),
                               y=result$pvalues[!result$mask]),
               aes(x=x, y=y), col="blue")
```

Code for Assignment 2

```
set.seed(1234567890)
spam <- read.csv2("../data/spambase.csv")

ind <- sample(1:nrow(spam))
spam <- spam[ind,c(1:48,58)]
```

```r
spam$Spam <- 2 * spam$Spam - 1

gaussian_k <- function(x, h) {
    exp(-(x / h)^2)
}

euclidean_d <- function(x, xi) {
    x <- t(as.matrix(x))
    xi <- as.numeric(xi)
    sqrt(colSums((x - xi)^2))
    ## d <- dist(rbind(x, xi))
    ## d[(length(d) - nrow(x) + 1):length(d)]
}

SVM <- function(sv, i) {
    h <- 1
    b <- 0
    x <- sv[, -ncol(sv)]
    t <- sv[, ncol(sv)]

    predicted <- sum(t * gaussian_k(euclidean_d(x, i), h)) + b
    predicted
}

sv.least_important <- function(sv) {
    which.max(lapply(sv, function(m) {
        obs <- spam[m,]
        x <- obs[, -ncol(obs)]
        t <- obs$Spam
        y <- SVM(spam[sv,], x)
        h <- 1
        k <- gaussian_k(euclidean_d(x, x), h)
        t * (y - t * k)
    }))
}

run_BOSVM <- function(data, beta, M, N) {
    errors <- 1
    errorrate <- vector(length = N)
    errorrate[1] <- 1
    sv <- c(1)

    for(i in 2:N) {
        predicted <- SVM(data[sv,], data[i, -ncol(data)])

        if (data[i, "Spam"] * predicted < beta) {
            sv <- c(sv, i)
            errors <- errors + 1

            if (length(sv) > M) {
                sv <- sv[-sv.least_important(sv)]
            }
        }
        errorrate[i] <- errors / i
    }
    list(errorrate=errorrate, sv=sv)
}

N <- 500
result <- run_BOSVM(data=spam, beta=0, M=500, N=N)
cat(paste("Number of Suppoer Vectors:", length(result$sv)))
plot(result$errorrate[seq(from=1, to=N, by=10)],
     type="o", main="Beta=0, M=500")
result <- run_BOSVM(data=spam, beta=-0.05, M=500, N=N)
cat(paste("Number of Suppoer Vectors:", length(result$sv)))
plot(result$errorrate[seq(from=1, to=N, by=10)],
     type="o", main="Beta=-0.05, M=500")
result <- run_BOSVM(data=spam, beta=0, M=20, N=N)
cat(paste("Number of Suppoer Vectors:", length(result$sv)))
plot(result$errorrate[seq(from=1, to=N, by=10)],
     type="o", main="Beta=0, M=20")
result <- run_BOSVM(data=spam, beta=-0.05, M=20, N=N)
cat(paste("Number of Suppoer Vectors:", length(result$sv)))
plot(result$errorrate[seq(from=1, to=N, by=10)],
     type="o", main="Beta=-0.05, M=20")
```