Stat 760 Homework 10

Will Bliss and Jakob Lovato

4/14/2022

```
#import data
data <- read.csv("/Users/jakoblovato/Desktop/Stat 760/HW 10/data_2.csv")
data1 <- data[which(data[,4] == 1),]
data2 <- data[which(data[,4] == -1),]
```

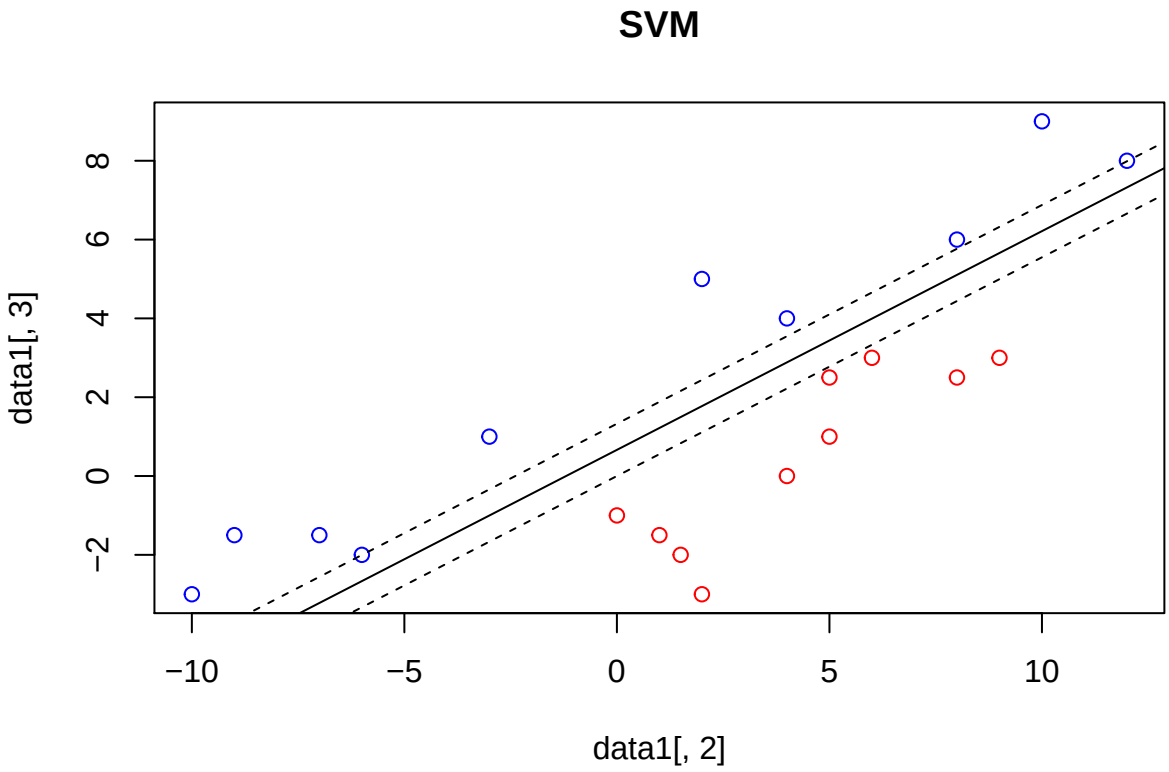
Question 1

This question was skipped as per instructions. However, Dr. Rojas said that since our program was functional using $w_0 = 1$, then we could submit it as is. Hence, this gave a restraint of $y_i(w^T x_i + 1) - 1 \geq 0$.

Question 2

```
x <- data[,2:3]
y <- data[,4]
w0 <- 1

#function to check if wieght vector w is within feasible region
inRegion <- function(w){
  result <- c()
  for(i in 1:nrow(x)){
    #store vector of true/false. If all true, the weight vector w is inside feasible region
    result <- c(result, (y[i] * (w %*% t(x[i,]) + w0) - 1 >= 0))
  }
  if(sum(result) < 20){
    return(FALSE)
  }
  else{
    return(TRUE)
  }
}

#function to return quadratic length
quadLen <- function(w){
  return(w %*% t(t(w)))
}

#initialize quadratic length to be minimized
minQuadLen <- 1000

#pick initial point inside feasible region
```

```r
point <- c(0, 0)
while(TRUE){
  temp1 <- runif(1, -5, 5)
  temp2 <- runif(1, -5, 5)
  w <- c(temp1, temp2)
  if(inRegion(w)){
    point <- w
    minQuadLen <- quadLen(point)
    break
  }
}

#minimize quadratic length within feasible region
for(i in 1:500){
  #pick new point and see if quadratic length is reduced
  newPoint <- c(point[1] + runif(1, -0.1, 0.1), point[2] + runif(1, -0.1, 0.1))
  if(inRegion(newPoint) && quadLen(newPoint) < minQuadLen){
    point <- newPoint
    minQuadLen <- quadLen(point)
  }
}

w <- point
#plot SVM
plot(data1[,2], data1[,3], col = "red", xlim = c(min(data[,2]), max(data[,2])),
     ylim = c(min(data[,3]), max(data[,3])), main = "SVM")
points(data2[,2], data2[,3], col = "blue")
abline(w0 / -w[2], w[1] / -w[2])
abline((w0 - 1) / -w[2], w[1] / -w[2], lty = 2)
abline((w0 + 1) / -w[2], w[1] / -w[2], lty = 2)
```




```
#classify data
classifications <- as.numeric(20)
preds <- as.matrix(x) %*% as.matrix(w)
for(i in 1:nrow(preds)){
  if(preds[i] < 0){
    classifications[i] <- -1
  }
  else{
    classifications [i] <- 1
  }
}

error <- sum(y != classifications)
error
```

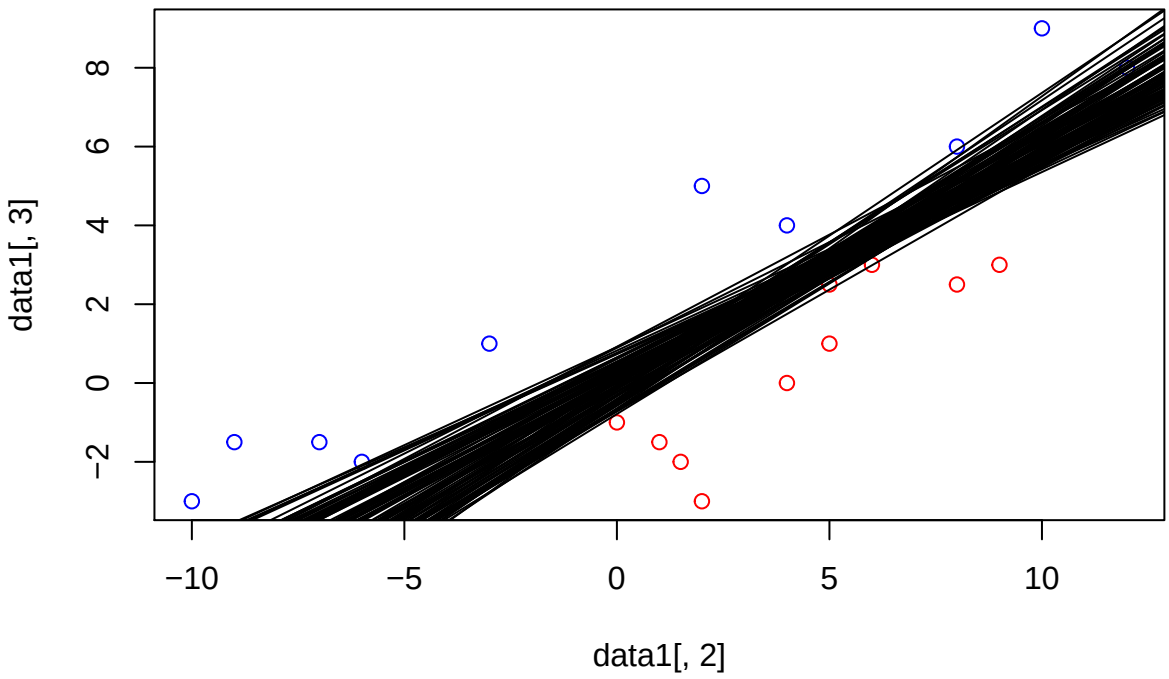
```
## [1] 0
```

The misclassification rate is 0, so all the data has been correctly classified.

Question 3

```
regression <- function(data){
  betaZero <- matrix(1, ncol = 1, nrow(data))
  X <- as.matrix(cbind(betaZero, data[,2]))
  XTXInverse <- solve(t(X) %*% X)
  betas <- XTXInverse %*% t(X) %*% data[,3]
  rownames(betas) <- c("intercept", "slope")
  return(as.data.frame(t(betas)))
}
```

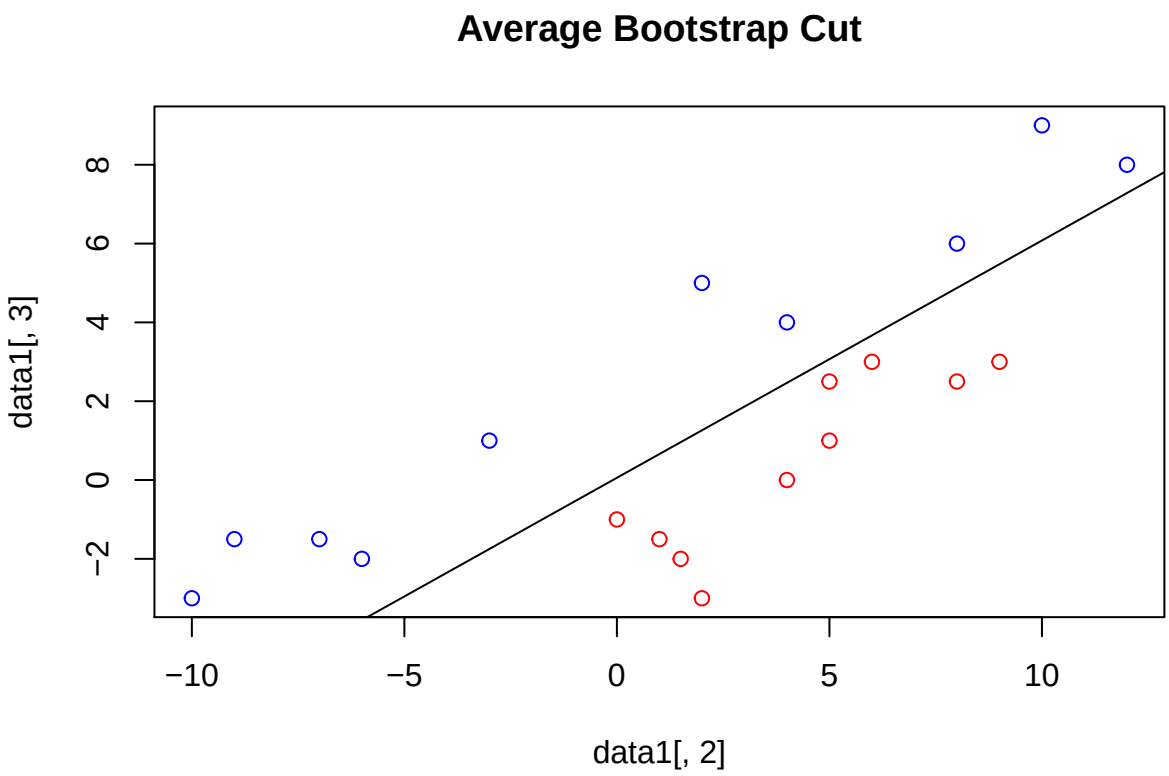
```r
#bootstrapping
n <- nrow(data)
slopes <- c()
intercepts <- c()

plot(data1[,2], data1[,3], col = "red", xlim = c(min(data[,2]), max(data[,2])),
     ylim = c(min(data[,3]), max(data[,3])), main = "All Bootstrap Cuts")
points(data2[,2], data2[,3], col = "blue")

for(i in 1:100){
  indices <- sample(c(1:n), n, replace = TRUE)
  bootData <- data[indices, ]
  bootData1 <- bootData[which(bootData[,4] == 1),]
  bootData2 <- bootData[which(bootData[,4] == -1),]

  fit1 <- regression(bootData1)
  fit2 <- regression(bootData2)

  intercept1 <- fit1$intercept
  slope1 <- fit1$slope
  intercept2 <- fit2$intercept
  slope2 <- fit2$slope

  bootIntercept <- mean(c(intercept1, intercept2))
  bootSlope <- mean(c(slope1, slope2))

  intercepts <- c(intercepts, bootIntercept)
  slopes <- c(slopes, bootSlope)

  abline(c(bootIntercept, bootSlope))
}
```




```
bootstrap <- c(mean(intercepts), mean(slopes))
bootstrap
```

```
## [1] 0.05651164 0.60213486
```

```
plot(data1[,2], data1[,3], col = "red", xlim = c(min(data[,2]), max(data[,2])),
     ylim = c(min(data[,3]), max(data[,3])), main = "Average Bootstrap Cut")
points(data2[,2], data2[,3], col = "blue")
abline(bootstrap)
```



The average cut from the bootstrap has y-intercept 0.0565116 and slope 0.6021349.