STAT W 4400 HW2-2 Perceptron

Yanjin Li

February 22, 2016

In this problem, I implement perceptron on a two-class set of lienarly separable data. For the purpose of testing the algorithm, I will randomly generate both training and test datasets. Then, I will implement perceptron algorithm with learning rate 1/k into R code.

```
#Inputs
#w:  w[1:d] is the normal vector of a hyperplane,
#    w[d+1] = -c is the negative offset parameter.
#n: sample size

#Outputs
#S: n by (d+1) sample matrix with last col 1
#y: vector of the associated class labels
```

0. Data generating Randomly generate two sets of linearly separable datasets with binary classes labelled as +1 and -1.

1. Perceptron solution evaluation function returning with class labels

```
classify <- function(S,z){
  n <- nrow(S[[1]])
  y <- matrix(ncol = n, nrow = 1)
  for (i in 1:n){
    x <- S[[1]][i,]
    if(t(z) %*% x >0){
      y[i] <- 1
    }else{
      y[i] <- -1
    }
  }
  y <- c(y)
  return(y)
}
```

2. Perceptron algorithm inplement
   Implement the Batch Perceptron with learning rate 1/k where k is the number of the current iteration.

```
perceptrain <- function(S, y){
  n <- nrow(S[[1]])
  m <- ncol(S[[1]])
  z <- rnorm(m)
  Z_hist <- rbind(matrix(ncol = m, nrow = 0), z)
  for (k in 1: 1e+05){
    Cp <- 0
    gdtCp <- 0
    for (i in 1:n){
      x <- S[[1]][i,]
      if (sign(y[i]) != sign(z %*% x)){
        Cp <- Cp + abs(z %*% x)
        gdtCp <- gdtCp + (-y[i]) %*% x
      }
    }
    if (Cp == 0){
      Z_hist <- rbind(Z_hist, z)
      return(list(z, Z_hist))
    }else{
      z <- z - (1/k) * gdtCp
      Z_hist <- rbind(Z_hist, z)
    }
  }
}
```

3. Training and testing processes Generate a 3D random vector z, run functions and train my Perceptron on traing and test datasets.

```
require(mvtnorm)
```

```
## Loading required package: mvtnorm
```

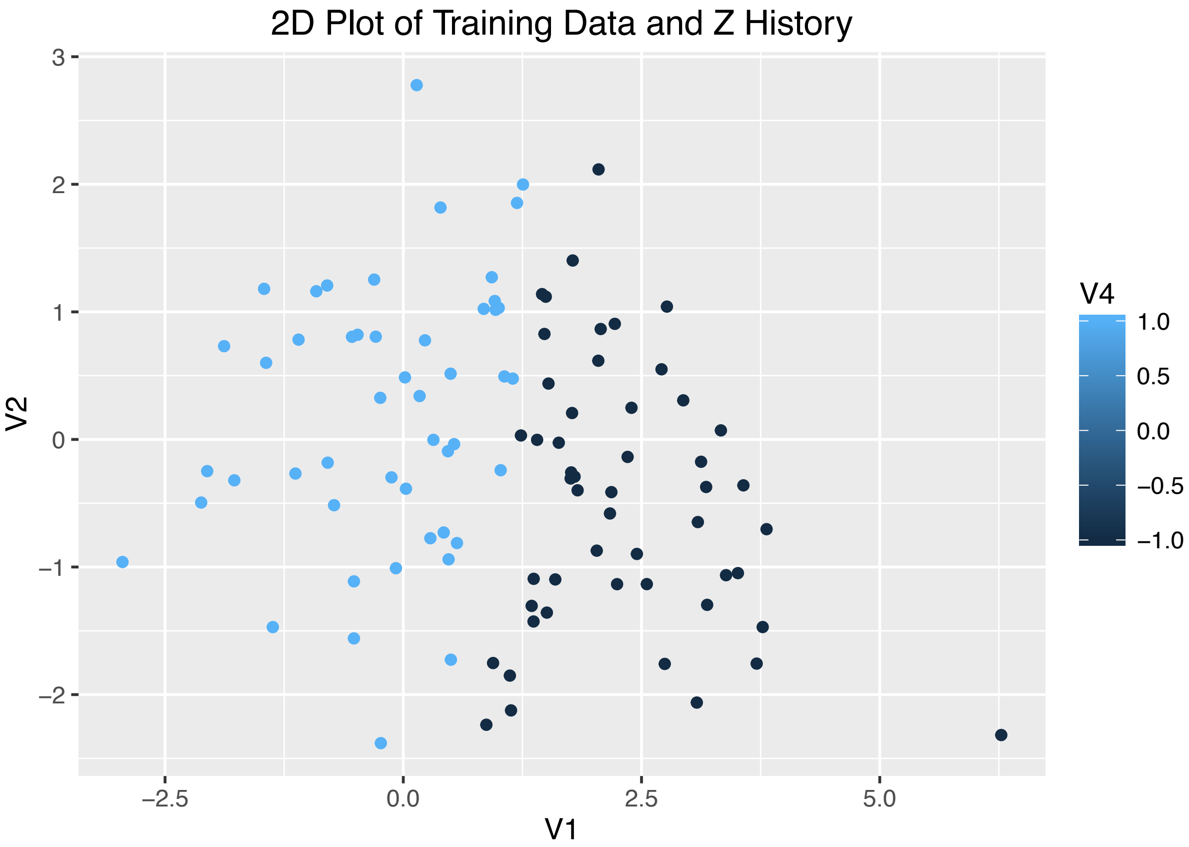
```
z <- rnorm(3)
training <- fakedata(z, 100)
```

```
## Loading required package: MASS
```

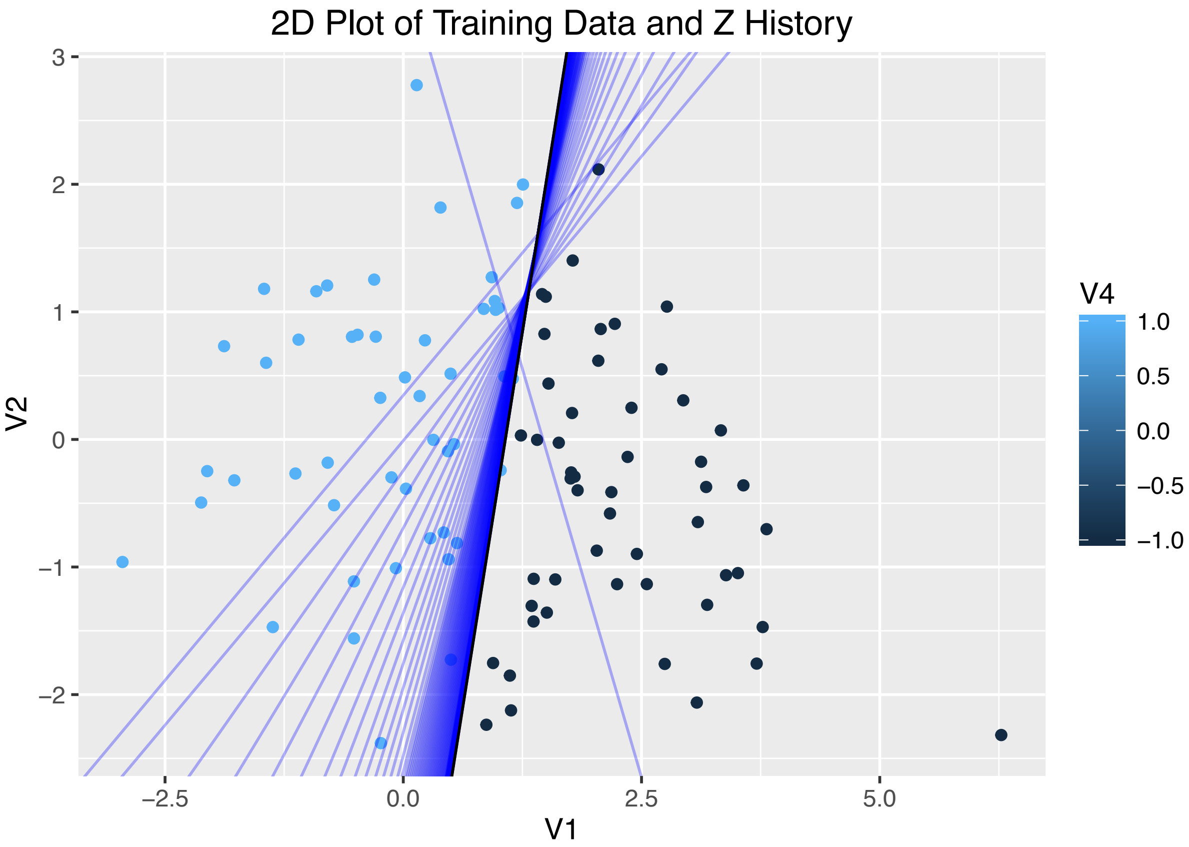
```
test <- fakedata(z, 100)
training.classify<- classify(training, z)
test.classify <- classify(test, z)
robort <- perceptrain(training, training.classify)
robort.test <- perceptrain(test, test.classify)
```

4. Plotting 3D datapoints back into 2D plots For training data with a set of lines from the Z history:

```
library(ggplot2)
training.classified <- cbind(training[[1]], training[[2]])
new.training.classified <- as.data.frame(training.classified)
datapoints.plot <- ggplot(data = new.training.classified,
                          aes(x = V1, y =V2, color = V4)) +
  geom_point() +
  ggtitle("2D Plot of Training Data and Z History")

datapoints.plot
```

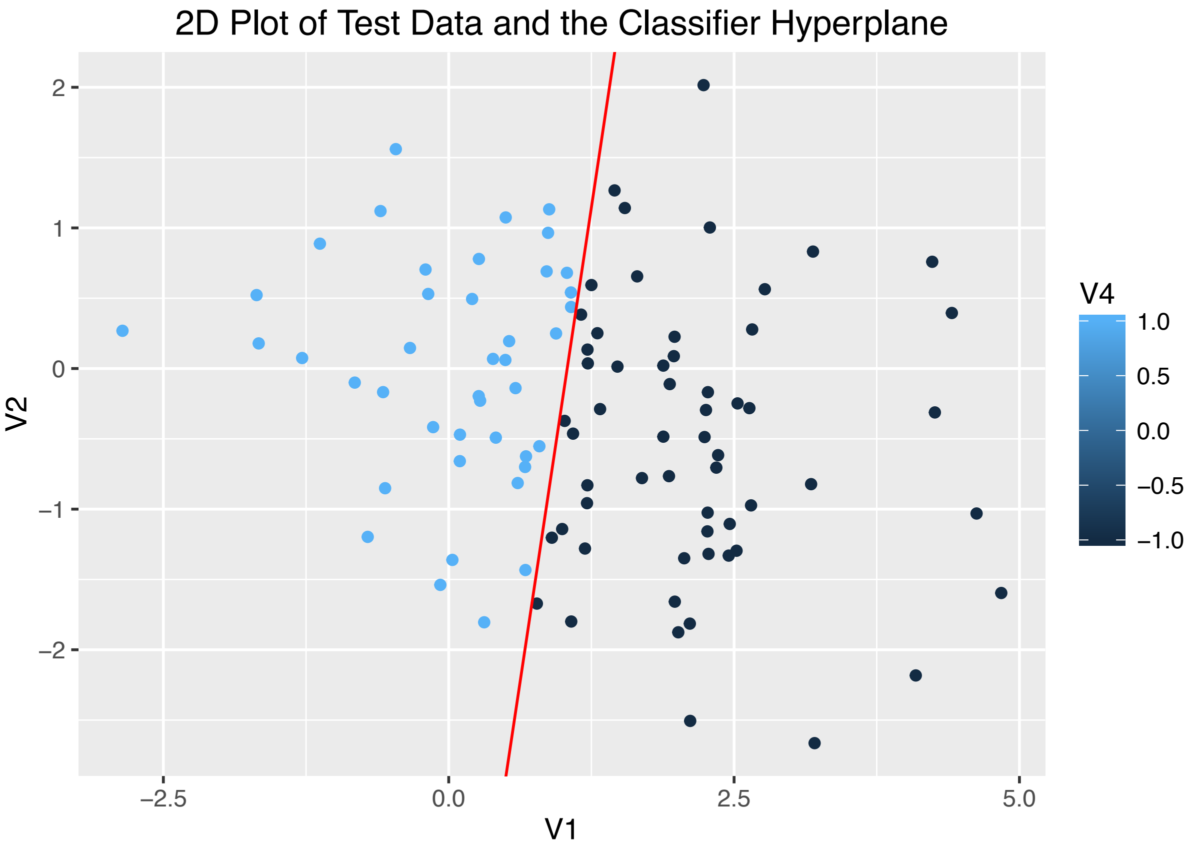


```
n <- nrow(robort[[2]])

for (i in 1:nrow(robort[[2]])){
  norm.matrix <- 1/(sqrt(robort[[2]][i,1]^2 +
                           robort[[2]][i,2]^2)) * robort[[2]][i,]
  z.homo.coord <- Null(norm.matrix[1:2]) +
    norm.matrix[3] * c(norm.matrix[1], norm.matrix[2])

  if (i < nrow(robort[[2]])){
    b <- (Null(norm.matrix)[2])/(Null(norm.matrix)[1])
    datapoints.plot <- datapoints.plot +
      geom_abline(intercept = -sign(norm.matrix[2]) *
                  norm.matrix[3] * sqrt(b^2+1),
                  slope = b, alpha = 0.3, color = "blue")
  } else{
    datapoints.plot <- datapoints.plot +
      geom_abline(intercept = -sign(norm.matrix[2]) *
                  norm.matrix[3] * sqrt(b^2+1),
                  slope = b, color = "black")
  }
}
datapoints.plot
```



For test data with the classifier hyperplane:

```
test.classified <- cbind(test[[1]], test[[2]])
new.test.classified <- as.data.frame(test.classified)

n <- nrow(robort.test[[2]])
norm.matrix2 <- 1/(sqrt(robort.test[[2]][n,1]^2 +
                          robort.test[[2]][n,2]^2)) *
  robort.test[[2]][n,]

z.homo.coord2 <- Null(norm.matrix2[1:2]) +
  norm.matrix2[3] * c(norm.matrix2[1], norm.matrix2[2])

b <- (Null(norm.matrix2)[2])/(Null(norm.matrix2)[1])

ggplot(data = new.test.classified, aes(x = V1, y =V2, color = V4)) +
  geom_point()+
  ggtitle("2D Plot of Test Data and the Classifier Hyperplane") +
  geom_abline(intercept = -sign(norm.matrix2[2]) *
              norm.matrix2[3] * sqrt(b^2+1),
              slope = b, color = "red")
```


2D Plot of Test Data and the Classifier Hyperplane
V2
2
1
0
−1
−2
V1
−2.5
0.0
2.5
5.0
V4
1.0
0.5
0.0
−0.5
−1.0