RAZAVET Mael 27/02/2014

Data Mining – Exercise 6

Question 1:

```
perceptron=function(D, eta=0.1, class){
 b <- 0
 converged <- FALSE
 w <- rep(1,(ncol(D) - 1))
 while(!converged){
        converged <- TRUE
        for(i in 1:nrow(D)){
                a <- sum(w*D[i,]) + b

                if(class[i]*a <= 0){
                        w <- w + class[i]*eta*D[i,]
                        converged <- FALSE
                }
        }
 }
 return (w)
}
```

Question 2:

To classify my data, I chose to select the attribute Sepal.Width and Sepal.Length.

```
d <- iris[iris$Species == c('setosa', 'versicolor'), -(3:4)]
row.names(d) <- NULL
d$Species <- as.numeric(d$Species == "setosa")#setosa equals to 1
d$Species[d$Species == 0] <- -1 #versicolor equals to -1

w <- perceptron(d[,-3], 0.1, d$Species) #we split the data into input (1st parameter) and
output values (3rd parameter)
```

Random order of the iris data:

```
df <- sample(1:nrow(d), nrow(d), replace=FALSE)
df <- subset(d[training,])
row.names(df) <- NULL
```

I observe that in any cases, my algorithm will always converge. By classifying the randomized data or the original one, by varying the η or the b, the algorithm always converge.

I experimented with values of η and b in range of 0.1 to 5. This is always converging.

Learning rate	W1	W2	b
0.1	-0.64	1.01	0.1
0.5	-3.75	6.15	0.1
1	-6.4	10.8	0.1
2	-12	20.04	0.1
5	-45.5	73.5	0.1
0.1	-0.78	1.12	0.5
0.5	-0.95	1.45	0.5
1	-6.90	11	0.5
2	-12	20.40	0.5
5	-45.50	73.50	0.5
0.1	-0.51	0.58	1
0.5	-4.05	6.20	1
1	-4.80	7.40	1
2	-13	20.80	1
5	-45.50	73.50	1
0.1	-0.93	1.04	2
0.5	-4.25	6.30	2
1	-7.30	11.20	2
2	-13	20.80	2
5	-45.50	73.50	2

I observe that only changing the intercept b does not change the weights considerably. For example, when the learning rate equals 5, the weights stay at -45.50 and 73.50. These weights change if we set b as very high like 100.
Changing the learning rate does not affect the convergence of the algorithm.

However, the higher is the learning rate, the more the weight 1 (the weight of the attribute Sepal.Length) is decreasing and the more the weight 2 (the weight of the attribute Sepal.Width) is increasing to adapt the hyperplane.

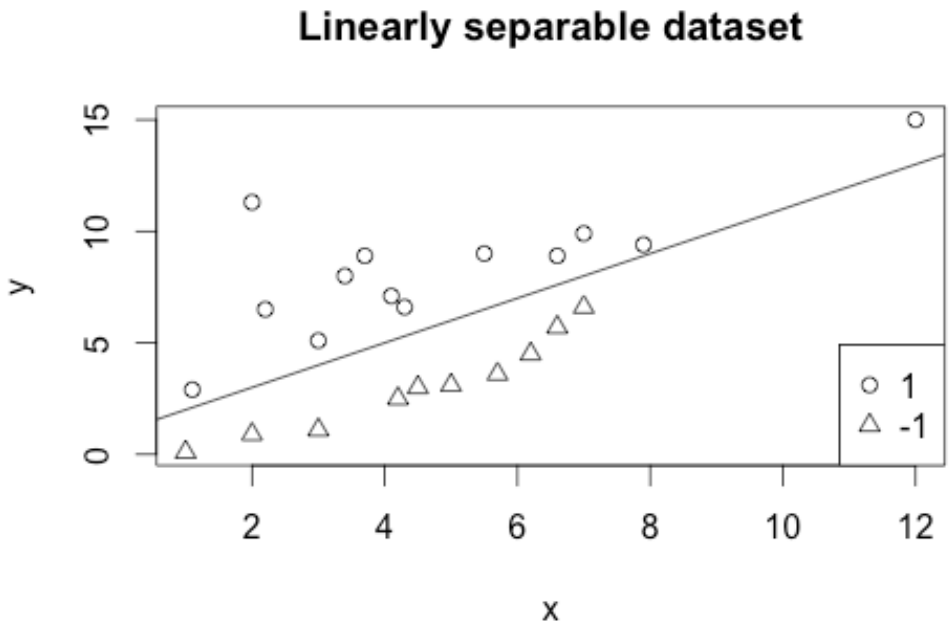
Question 3:

First of all, we need to create a linearly separable, 2-dimensional data of 25 instances.
I used a trivial way to create this data. I chose to plot a line that passes through 1 on the y-axis and draw some points over and under this line. This means that the intercept is 1. I chose the slope to be 1 as well.

```
x <- c(2.2, 3, 1.1, 7.9, 4.1, 3.7, 3.4, 2.0, 5.5, 7.0, 4.3, 6.6, 12)
```

```
y <- c(6.5, 5.1, 2.9, 9.4, 7.1, 8.9, 8, 11.3, 9.0, 9.9, 6.6, 8.9, 15)
class<-c(1,1,1,1,1,1,1,1,1,1,1,1,1)
m<-data.frame()
m<-cbind(x,y,class)
x2<-c(1,2,3,4.2,4.5,5.0, 5.7, 6.2,6.6,7.0)
y2<-c(0.1,0.9,1.1,2.5,3.0,3.1,3.6,4.5,5.7,6.6)
c2<-c(-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1)
for(i in 1:length(x2)){
  m <- rbind(m, x2[i])
  m[i + 13,2] <- y2[i]
  m[i + 13,3] <- c2[i]
}
plot(m[,-3], main='Linearly separable dataset', pch=ifelse(m[,3] > 0, 1, 2))
abline(1,1)
```

Here is the plot of this dataset with the line that separates the two classes.

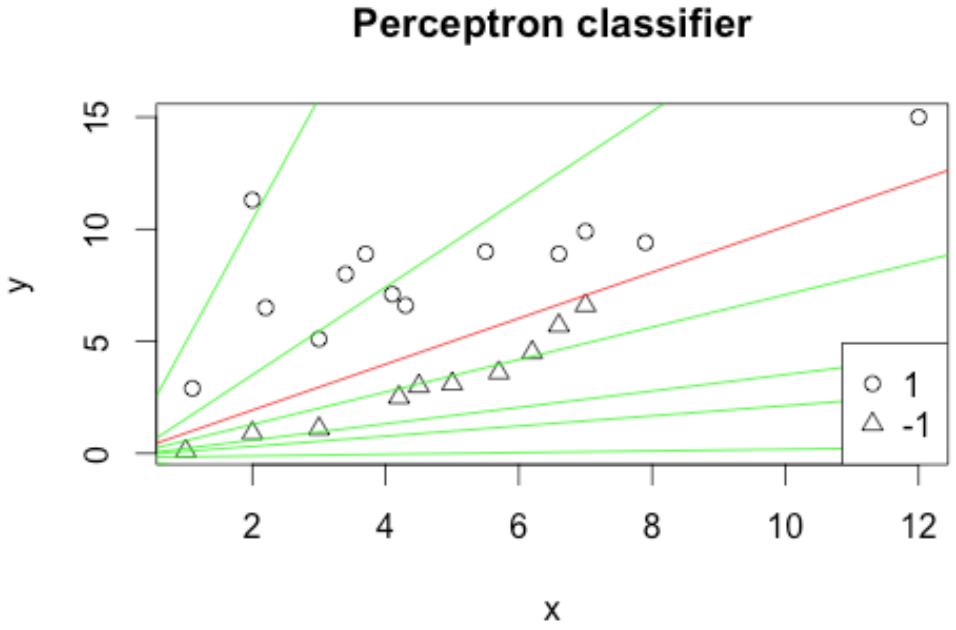


In order to be able to visualize each intermediate classifier, I had to update the perceptron algorithm by storing each weight in a dataframe.
The new code is displayed below:

```
perceptron=function(D, eta, class){
  r <- data.frame()
  b <- 1
  converged <- FALSE
  w <- rep(1,(ncol(D) - 1))
  while(!converged){
        converged <- TRUE
```

```
        for(i in 1:nrow(D)){
                a <- sum(w*D[i,]) + b
                if(class[i]*a <= 0){
                        w <- w + class[i]*eta*D[i,]
                        r <- rbind(r, w)
                        converged <- FALSE
                }
        }
 }
 return (r)
}
```

By using my perceptron algorithm to classify the dataset, we can observe that it converges and it needs 10 iterations to find the best hyperplane that classifies well the data. In the next plot, we can visualize the intermediate hyperplanes in green and the final classifier in red.



Here is the code in R that provides the plot above:

```
w <- perceptron(m[,-3], 0.1, m[,3])
plot(m[,-3], main='Perceptron classifier', pch=ifelse(m[,3] > 0, 1, 2))
for(i in 1:nrow(w)){
 w1 <- w[i,1]
 w2 <- w[i,2]
 b <- 0.1
 if(i != nrow(w))
  abline(-b/w2, -w1/w2,col='green')
 else
  abline(-b/w2, -w1/w2, col='red')
```

```
}
```

I observed that by changing the learning rate does not affect also the convergence of the algorithm, it only affects the number of iterations needed to find the final classifier.