Exercício 01 - Reconhecimento de Padrões

Aluno: Giovanni Martins de Sá Júnior - 2017001850

6 de setembro de 2023

Exercício 1: Amostragem de Dados

Neste primeiro exercício, estudaremos a implementação de um Perceptron simples. Nesse sentido, começamos implementando a plotagem inicial dos dados de entrada utilizados para o modelo. Inicialmente, foram plotados dois conjuntos de dados centrados nos pontos (2,2) e (4,4), com um desvio padrão de 0.4, conforme veremos na Figura 1.

Após a plotagem dos dados amostrais, foi então definida a equação da superfície de separação. Nesse caso, a superfície será uma reta uma vez que os conjuntos de dados de entrada se encontram em duas dimensões. Sobre a reta, ela se trata de um separador $X_2 = -X_1 + 6$, com os respectivos pesos $W_1 = 1, W_2 = 1$ e $\theta = -6$. A seguir, é apresentado o código desenvolvido na linguagem R para a apresentação dos conjuntos de dados C_1 e C_2, denotados pelas cores vermelha e azul, respectivamente. Além delas, é mostrada também a superfície (reta) de separação, denotada na cor laranja.

```
# Declaração dos conjuntos de entrada e plotagem dos dados
sd1 <- 0.4
sd2 <- 0.4
tam <- 100

xc1 <- matrix(rnorm(tam * 2), ncol = 2) * sd1 + t(matrix(c(2,2), ncol = tam, nrow = 2))
xc2 <- matrix(rnorm(tam * 2), ncol = 2) * sd2 + t(matrix(c(4,4), ncol = tam, nrow = 2))

plot(xc1[,1], xc1[,2], col = "red", xlim = c(0,6), ylim = c(0,6), xlab = "x_1", ylab = "x_2")
par(new = T)
plot(xc2[,1], xc2[,2], col = "blue", xlim = c(0,6), ylim = c(0,6), xlab = "x_1", ylab = "x_2")

x1_reta <- seq(6/100, 6, 6/100)
x2_reta <- (-x1_reta) + 6

par(new = T)
plot(x1_reta, x2_reta, type = "l", col = "orange", xlim = c(0,6), ylim = c(0,6), xlab = "x_1",
ylab = "x_2")
```

Abaixo, é possível ver o resultado obtido a partir da implementação mencionada acima:

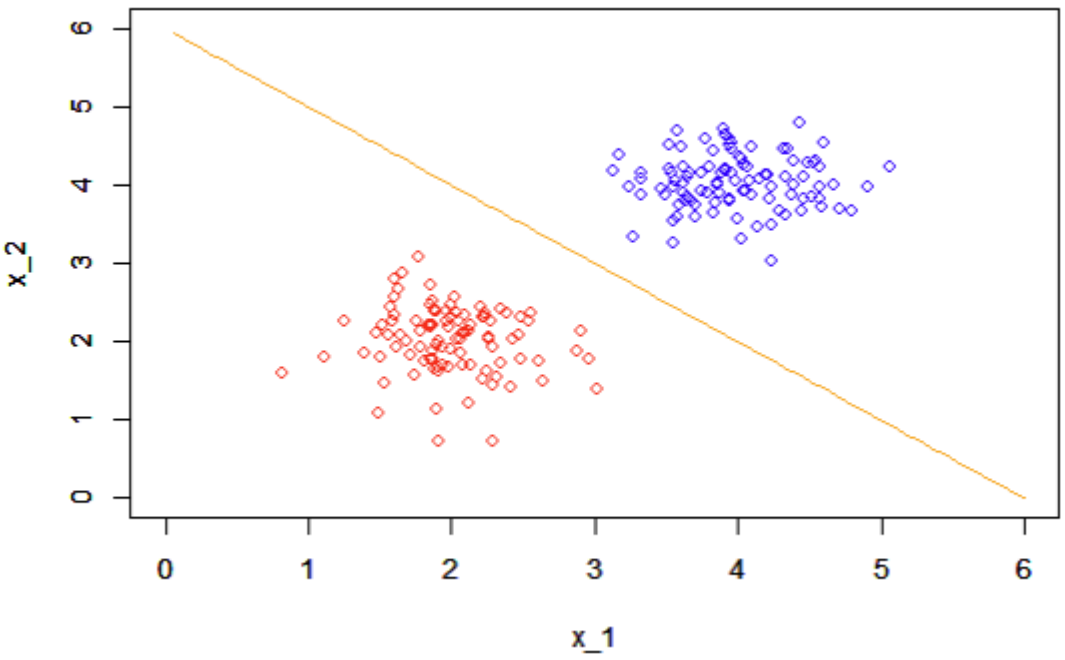

Figura 1: Plotagem dos conjuntos de entrada e da reta de separação

Exercício 2: Treinamento do Perceptron

Neste segundo exercício, trataremos de realizar o treinamento do Perceptron com o objetido de definir o vetor de pesos w para o modelo. Inicialmente, foram implementadas duas funções chamadas *trainPerceptron* e *yPerceptron*, que realizam o treinamento e a classificação do Perceptron, respectivamente. A implementação das duas funções são apresentadas a seguir:

```
# Função de treinamento do Perceptron:
trainPerceptron <- function(xin, yd, eta, tol, maxEpocas, par){
    N <- dim(xin)[1]    # Recebe as linhas
    n <- dim(xin)[2]    # Recebe as colunas

    if (par == 1) {
        wt <- as.matrix(runif(n + 1) - 0.5)     # Inicialização dos pesos
        xin <- cbind(-1, xin)
    } else {
        wt <- as.matrix(runif(n) - 0.5)         # Inicialzacao dos pesos
    }

    nEpocas <- 0    # Contador de épocas
    eEpoca <- tol + 1   # Acumulador de erro por epoca
    evec <- matrix(nrow = 1, ncol = maxEpocas)  # Inicializacao do vetor erro evec

    # Laço de treinamento
    while((nEpocas < maxEpocas) && (eEpoca > tol)){
        ei2 <- 0
        # Sequência aleatória de treinamento
        xSeq <- sample(N)
        for (i in 1:N){
            # Amostra de dado da sequência aleatória
            iRand <- xSeq[i]
```

```
            # Calculo da saída do Perceptron
            yHat <- 1.0 * ((xin[irand,] %*% wt) >= 0)
            ei <- yd[iRand] - yHat
            dw <- eta * ei * xin[iRand,]# deltaW = n * e * x
            # Ajuste dos pesos
            wt <- wt + dw
            # Erro acumulado por época
            ei2 <- ei2 + (ei * ei)
        }
        # Incremento do número de épocas
        nEpocas <- nEpocas + 1
        evec[nEpocas] <- nEpocas + 1
        # Armazena erro por época
        eEpoca <- evec[nEpocas]
    }
    # Retorno dos vetor com os pesos e erros
    retList <- list(wt, evec[1:nEpocas])
    return(retList)
}

# Função de saida do Perceptron
yPerceptron <- function(x, w, par){
    if(par == 1){
        xvec <- cbind(1, xvec)
    }
    u <- x %*% w
    y <- 1.0 * (u >= 0)
    return(as.matrix(y))
}
```

A partir da implementação das duas funções anteriormente mencionadas, foi feito finalmente o treinamento e classificação do Perceptron, com o intuito de se obter os pesos definitivos após esse processo. Com isso, foi dada a continuidade da implementação anterior, que pode ser vista logo abaixo:

```
# Treinamento e Classificação do Perceptron
xc1 <- cbind(xc1, 0)
xc2 <- cbind(xc2, 1)
x <- rbind(xc1, xc2)
retlist <- trainPerceptron(x[, 1:2], x[,3], 0.1, 0.01, 100, 1)
#w <- matrix(retlist[[1]])
w <- retlist[[1]]

seqi <- seq(0, 6, 0.1)
seqj <- seq(0, 6, 0.1)
rt <- matrix(0, nrow = length(seqi), ncol = length(seqj))
ci <- 0

for (i in seqi) {
    ci <- ci + 1
    cj <- 0
    for (j in seqj) {
        cj <- cj + 1
        #x <- as.matrix(t(c(1, i, j)))  # Adicione 1 para o termo de polarização
        x <- as.matrix(t(c(i,j)))
        rt[ci, cj] <- yPerceptron(x, w, 1)
    }
```

```
}

plot(xc1[,1], xc1[,2], col = 'red', xlim = c(0,6), ylim = c(0,6), xlab = 'x_1', ylab = 'x_2')
par(new = T)
plot(xc2[,1], xc2[,2], col = 'blue', xlim = c(0,6), ylim = c(0,6), xlab = 'x_1', ylab = 'x_2')
par(new = T)

# Plot da reta de separação
contour(seqi, seqj, rt, xlim = c(0,6), ylim = c(0,6), xlab = '', ylab = '')
```

Diante da implementação acima, foi obtido o desenho final da superfície de separação e dos pesos finais após a realização do treinamento. por meio do desenho abaixo, foi possível observar que a superfície de separação atendeu o requisitos esperados, e os pesos obtidos estão alinhados com a expectativa do modelo. Abaixo, os resultados finais alcançados:

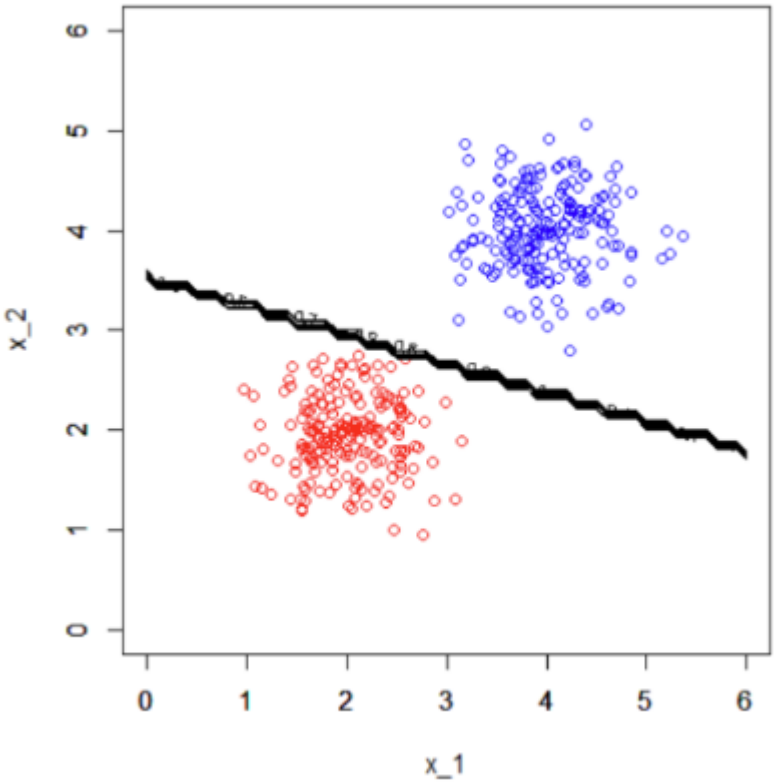

Figura 2: Definição da superfície de separação

Pesos W do modelo:

$$W = \begin{bmatrix} 1.3862 \\ 0.1533 \\ 0.277 \end{bmatrix} \tag{1}$$