Universidade Federal de Minas Gerais
Aluno: Giovanni Martins de Sá Júnior
Matrícula: 2017001850

Exercício 4: Redes Neurais Artificiais

1. Treinamento e Visualização

O código usado para realizar a função do Perceptron pode ser visto logo abaixo:

```r
trainPerceptron <- function(xin, yd, eta, tol, maxEpocas, par) {
  dimXin <- dim(xin)
  N <- dimXin[1]
  n <- dimXin[2]

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

  nEpocas <- 0
  eEpoca <- tol + 1
  evec <- matrix(nrow = 1, ncol = maxEpocas)

  while ((nEpocas < maxEpocas) && (eEpoca > tol)) {
    ei2 <- 0
    xSeq <- sample(N)
    for(i in 1:N) {
      iRand <- xSeq[i]
      yHat <- 1.0 * ((xin[iRand,] %*% wt) >= 0)
      ei <- yd[iRand] - yHat
      dw <- eta * ei * xin[iRand,]
      wt <- wt + dw
      ei2 <- ei2 + ei * ei
    }
    nEpocas <- nEpocas + 1
    evec[nEpocas] <- ei2 / N
    eEpoca <- evec[nEpocas]
  }
  retList <- list(wt, evec[1:nEpocas])
  return(retList)
}
```

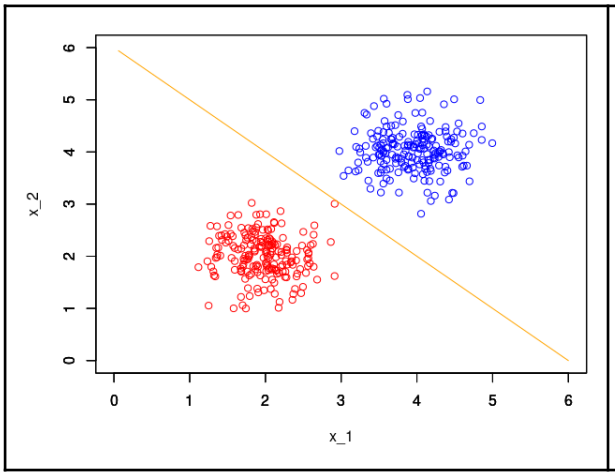
Figura 1. Implementação da função de treinamento do Perceptron

Além desta função, foi também implementado um outra denominada *yPerceptron*, como pode ser vista logo abaixo:

```r
yPerceptron <- function (xvec, w, par) {
  if (par == 1){
    xvec <- cbind(1, xvec)
  }
  u <- xvec %*% w
  y <- 1.0 * (u >= 0)
  return(as.matrix(y))
}
```

Figura 2. Implementação da função yPerceptron

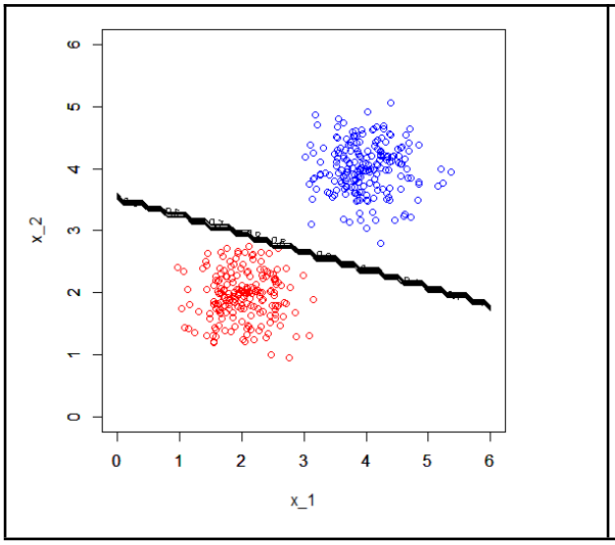
Com isso, os dados do problema, bem como o código para gerá-los com a reta de separação dos espaços amostrais também podem ser vistas nas Figuras 3 e 4:




```r
s1 <- 0.4
s2 <- 0.4
nc <- 200
xc1 <- matrix(rnorm(nc * 2), ncol = 2) * s1 + t(matrix((c(2,2)), ncol = nc, nrow = 2))
xc2 <- matrix(rnorm(nc * 2), ncol = 2) * s2 + t(matrix((c(4,4)), ncol = nc, 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 = '', ylab = '')

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 = '', ylab = '')
```

Figuras 3 e 4. Dados do Problema

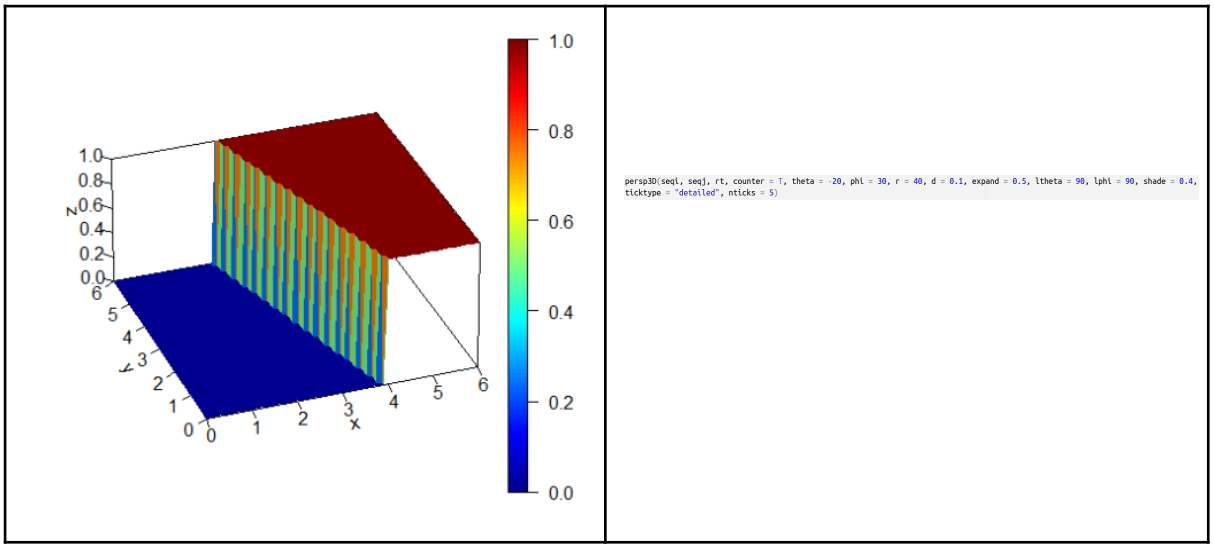
A partir disso, foi usada a função ‘trainPerceptron’ e ‘yPerceptron’ (apresentadas anteriormente) para gerar uma reta de separação assim como mostrado a seguir:




```r
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]])

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(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 = '', ylab = '')
par(new = T)
contour(seqi, seqj, rt, xlim = c(0,6), ylim = c(0,6), xlab = '', ylab = '')
```

Figuras 5 e 6. Entrada e Saída dos dados

Por último, foi implementado um plot para as três dimensões com o intuito de compreender o comportamento da função do perceptron na separação das regiões e diferenciação.



Figuras 7 e 8. Representação em 3D

Com isso, foi possível observar que, para os pontos abaixo do eixo xy, o seu valor no eixo z será de zero (Plano Azul), e para os pontos acima da reta, o valor no eixo z será de um (Plano Vermelho).

2. Treinamento e Teste

Para a resolução deste segundo exercício, é de interesse encontrar os valores de acurácia e matriz de confusão. Assim, o treinamento será realizado com 70% das amostras e o restante será utilizado para teste, seguindo o mesmo padrão do primeiro exercício.

Nesse sentido, complementando o primeiro exercício, foi resolvido um problema similar com o já apresentado, agora com três entradas no lugar de uma.

A seguir, é apresentado a implementação do código:

```
s1 <- 0.4
s2 <- 0.4
nc <- 200
xc1 <- matrix(rnorm(nc * 2), ncol = 2) * s1 + t(matrix((c(2,2)), ncol = nc, nrow = 2))
xc2 <- matrix(rnorm(nc * 2), ncol = 2) * s2 + t(matrix((c(4,4)), ncol = nc, nrow = 2))

xc1 <- cbind(xc1, 0)
xc2 <- cbind(xc2, 1)
x <- rbind(xc1, xc2)

Num = 280
amostra = sample(400)
x_train <- x[amostra[1:Num], 1:2]
y_train <- x[amostra[1:Num], 3]
Num = 281
x_test <- x[amostra[Num:400], 1:2]
y_test <- x[amostra[Num:400], 3]

retlist <- trainPerceptron(x_train, y_train, 0.1, 0.01, 100, 1)
w <- matrix(retlist[[1]])

y_hat <- yPerceptron(x_test, w, par = 1)
teste = 120

acuracia <- 1 - ((t(y_test - y_hat) %*% (y_test - y_hat)) / (teste))
Matriz_confusao <- matrix(0, 2, 2)
```

```
for(i in 1:120){
  if(y_test[i] == 1) {
    if(y_hat[i,] == 1){
      Matriz_confusao[1,1] = Matriz_confusao[1,1] + 1
    } else{
      Matriz_confusao[1,2] = Matriz_confusao[1,2] + 1
    }
  }

  if(y_test[i] == 0) {
    if(y_hat[i,] == 0){
      Matriz_confusao[2,2] = Matriz_confusao[1,1] + 1
    } else{
      Matriz_confusao[2,1] = Matriz_confusao[2,1] + 1
    }
```

Figura 9. Implementação do código

A partir disso, os valores da Matriz de Confusão e da acurácia foram de respectivamente:

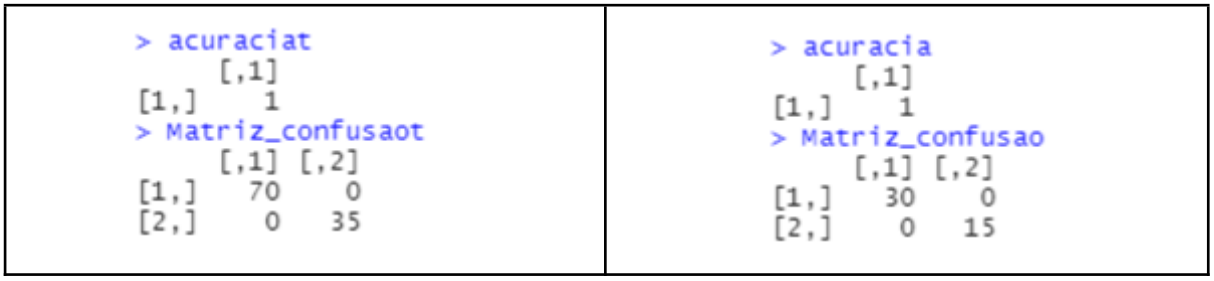
```
> Matriz_confusao
     [,1] [,2]
[1,]   65    0
[2,]    0   55
> acuracia
     [,1]
[1,]    1
```

Figura 10. Resultado obtido para a Matriz de Confusão e a Acurácia

3. Problemas de Maior Dimensão

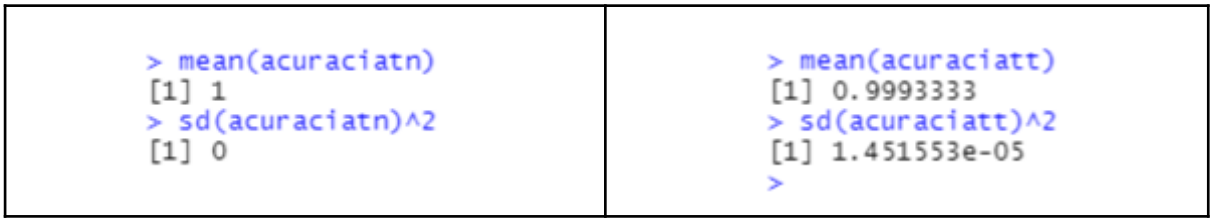
Ao se carregar os dados de 150 amostras, as 50 primeiras serão armazenadas na classe 1, enquanto o restante ficará na classe 2. Dentre essas amostras 70% serão destinadas a treinamento, e o restante para teste.

Os dados de acurácia e da matriz de confusão podem ser visualizados para o conjunto de treinamento e teste, respectivamente, conforme mostrado abaixo:



Figuras 11 e 12. Comparação de resultados

Em seguida, foi implementada um loop *for* com o intuito de repetir o treinamento 100 vezes, e encontrar o valor médio da acurácia e variância do conjunto de treinamento e teste. Abaixo, são apresentados os resultados:



Figuras 13 e 14. Comparação de resultados

Com isso, foi possível observar que o resultado final obtido foi satisfatório, já que a variância é zero, não identificando erros na solução.

4. Base de Maior Dimensão - *Breast Cancer*

Por meio da base de dados Breast Cancer, foi repetido o exercício anterior e obtido a acurácia e a matriz de confusão como é mostrado abaixo:

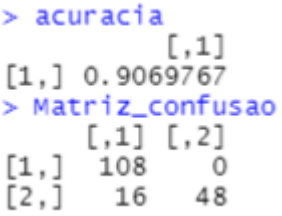


Figura 15. Acurácia e Matriz de Confusão obtidos da base de dados de *Breast Cancer*

Foi interessante observar a precisão de 90% na Figura 15, apesar da classificação ter cometido erros em alguns casos de tumores malignos, classificando em benignos.

No caso dos dados de treinamento, tem-se o seguinte resultado:

```
> acuraciat
          [,1]
[1,] 0.9244332
> Matriz_confusaot
     [,1] [,2]
[1,]  246    3
[2,]   27  121
```

Figura 16. Acurácia e Matriz de Confusão dos dados de treinamento

No caso do resultado dos dados de treinamento, observou-se uma precisão ligeiramente maior, apesar de ainda ocorrerem algumas classificações incorretas.

Com isso, foi realizado então um loop com cerca de 100 repetições similar ao executado na terceira atividade, com o intuito de obter a média e variância para o treinamento e teste. São apresentados os resultados logo abaixo:

```
> mean(acuracia)
[1] 0.8343023
> sd(acuracia)
[1] 0.144279
```

```
> mean(acuraciat)
[1] 0.9730479
> sd(acuraciat)
[1] 0.02130909
```

Figuras 17 e 18. Resultados dos dados de treinamento e teste

Neste último caso, foi observado uma diferença maior nos resultados obtidos, apesar de ainda estarem em uma faixa aceitável para o problema proposto.