INSTITUTO FEDERAL DE EDUCAÇÃO
CIÊNCIA E TECNOLOGIA DE MINAS GERAIS -
CAMPUS BAMBUÍ

BACHARELADO EM ENGENHARIA DE
COMPUTAÇÃO

Gabriela Dâmaso Rezende

# Tópicos Especiais em Engenharia de Computação

## Relatório 3

Bambuí - MG

2023

# 1 Qustão 1

Figura 1: Qustão 1

```
#ATIVIDADE 3

#1 - UTILIZE A REDE PERCEPTRON COM APRENDIZADO POR REGRA DE HEBB PARA CLASSIFICAR DOIS TIPOS DE FLOR

#Sao 50 amostras de Setosa e 50 de versicolor

#O dataset iris deve ser carregado da seguinte forma:

dados <- read.table("iris.csv",header=T,sep=",")
dados$variety <- ifelse(dados$variety %in% "Setosa", -1, dados$variety)
dados$variety <- ifelse(dados$variety %in% "Versicolor", 1, dados$variety)

setosa <- dados[dados$variety==-1,]
versicolor <- dados [dados$variety==1,]

setosaMistura<- setosa[sample(nrow(setosa)),]

versicolorMistura<- versicolor[sample(nrow(versicolor)),]
#nº 2 ----------------------------------------
```

Fonte: O autor.

Note que na figura 1 altera-se o valor da coluna variety de 1 e 2 para -1 e 1, respectivamente. O valor da coluna variety é utilizado para classificar o tipo de flor, sendo que 1 representa a flor Setosa e 2 representa a flor Versicolor. A partir da alteração do valor da coluna variety, o algoritmo de classificação passa a classificar a flor Setosa como -1 e a flor Versicolor como 1.

# 2 Qustão 2

Figura 2: Qustão 2

```
#nº 2 ----------------------------------------
setosaT<- setosaMistura[1:30,]
versicolorT<- versicolorMistura [1:30,]
setosaR<- setosaMistura[31:50,]
versicolorR<- versicolorMistura[31:50,]
treino <- rbind(setosaT, versicolorT)
teste<- rbind(setosaR, versicolorR)
#deletando
rm(setosaMistura)
rm(setosaR)
rm(setosaT)
rm(versicolorMistura)
rm(versicolorR)
rm(versicolorT)
```

Fonte: O autor.

Na figura 2 realizamos a separação do conjunto de dados em treino e teste, sendo que 60% dos dados foram utilizados para treino e 40% para teste. O conjunto de dados foi separado de forma aleatória. A mesma lógica foi aplicada para o conjunto de dados da flor Versicolor e Setosa

# 3 Qustão 3

Já na questão 3 realizamos o treino do RNA com o conjunto de dados de treino.

Figura 3: Qustão 3

```
#Nº 3----------------------------------------

#COLETANDO INFORMACOES DO DATASET
#quantidade de elementos na amostra
N <- dim(treino)[1]
#quantidade de entradas (a subtracao de 1 diz respeito ao fato de que a ultima
#coluna corresponde a saida esperada e nao a um atributo/entrada)
n <- dim(treino)[2] - 1
#separando a amostra de entrada
amostra <- treino[,1:n]
#inserindo o bias
bias <- rep(-1,N)
entradas <- cbind(amostra,bias)

#capturando apenas a saida y esperada
y <- treino[,n+1]

#removendo variaveis nao mais utilizadas da memoria
rm(bias)
rm(dados)
rm(amostra)

#PASSO 3: Iniciar o vetor w com valores aleatorios pequenos
#gerando o vetor de pesos iniciais aleatoriamente
#o +1 e referente ao peso w0 do bias
w <- runif(n+1, min=-0.3, max=0.3)

#PASSO 4: Especificar a taxa de aprendizagem eta
eta <- 0.07

#PASSO 5: Iniciar o contador de numero de epocas
nepocas <- 0

#COLOCANDO ALGUNS CRITERIOS DE PARADA
#Definir o maximo de epocas permitido
maxepocas <- 300
#Tolerancia minima aceitavel
tol <- 0
```

Fonte: O autor.

Figura 4: Qustão 3

```
#VARIAVEL QUE IRA DENOTAR A EXISTENCIA DE ERRO NA EPOCA CORRENTE
#Por exemplo, abaixo esta sendo setado do valor dela pois sabemos
#que na primeira iteracao existe erro
eepoca <- tol+1

#VARIAVEL QUE VAI ARMAZENAR O ERRO TOTAL COMPUTADO EM CADA EPOCA
erromedioEpocaAEpoca<-matrix(nrow=1,ncol=maxepocas)

#PASSO 6: Repetir instrucoes
while ((nepocas<maxepocas) && (eepoca>tol))
{
  #variavel que vai armazenar o erro total a
  #cada apresentacao do conjunto de dados
  erroConvergencia <- 0

  #distribui os indices dos elementos da amostra de forma aleatoria
  #para retirar o determinismo do processo, ou seja, embaralha os
  #elementos e nao apresenta eles ordenadamente a RNA
  indElementosEmbaralhados <- sample(N)

  for (i in 1:N)
  {
    #captura o indice do elemento a ser apresentado a rede
    indiceCorrente <- indElementosEmbaralhados[i]

    #captura o elemento corrente
    elemento <- as.numeric(entradas[indiceCorrente,])

    #calcula o limiar de ativacao
    u <- t(w) %*% elemento

    #gera a saida prevista com base na funcao de ativacao
    yhat <- degrauBipolar(u)

    #calcula o erro
    erro <-as.numeric(y[indiceCorrente])-yhat
```

Fonte: O autor.

Figura 5: Qustão 3

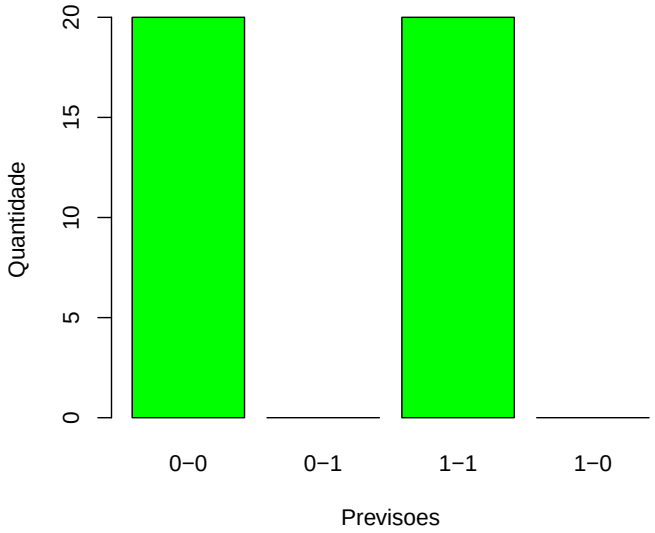
```
    #calculo do delta que incrementara ou decrementara os pesos
    delta<-(eta*erro)*elemento

    #substitui os pesos antigos pelos novos pesos apos esta epoca
    w <- w + delta

    #somatorio dos erros de convergencia epoca a epoca
    #o erro esta elevado ao quadrado pois pode ser que o mesmo seja
    #negativo. Estamos interessados na distancia para o ponto desejado
    erroConvergencia <- erroConvergencia + (erro^2)
  }

  #incrementa o contador de epocas para saber em que epoca a RNA esta
  nepocas <- nepocas+1

  #calcula a media dos erros na epoca corrente
  #e armazena o resultado num vetor para que, depois,
  #seja possivel avaliar o erro caindo epoca apos epoca
  erromedioEpocaAEpoca[nepocas] <- erroConvergencia/N

  #a variavel eepoca e utilizada no laco do while para saber
  #se o erro medio existente ate o momento e maior ou menor do que
  #a tolerancia estabelecida. E um criterio de parada.
  eepoca <- erromedioEpocaAEpoca[nepocas]
}


#plotando a convergencia do algoritmo com base no erro epoca a epoca
plot(as.numeric(erromedioEpocaAEpoca), type="l", ylab="erro medio", xlab="epocas", main="Convergencia do erro")

#exibindo os pesos calibrados pela regra de Hebb
print("Conjunto ideal de pesos:")
print(w)

#exibindo epocas necessarias a convergencia
print("Epocas necessarias a convergencia:")
print(nepocas)
```

Fonte: O autor.

Figura 6: Convergência do erro


Fonte: O autor.

Figura 7: Gráfico




Fonte: O autor.

Ao alanisar os graficos e o erro, podemos concluir que o algoritmo de treino convergiu para um valor de erro muito baixo.

# 4 Qustão 4

Na questão 4 realizamos o teste do RNA com o conjunto de dados de teste, como é possivel ver na figura 8. O algoritmo de teste obteve uma acurácia de 100% para o conjunto de dados de teste como pode ser visto va figura 9.

Figura 8: Qustão 4

```
#4 ----------------------
#COLETANDO INFORMACOES DO DATASET
#quantidade de elementos na amostra
N <- dim(teste)[1]
#quantidade de entradas (a subtracao de 1 diz respeito ao fato de que a ultima
#coluna corresponde a saida esperada e nao a um atributo/entrada)
n <- dim(teste)[2] - 1
#separando a amostra de entrada
amostra2 <- teste[,1:n]
#inserindo o bias
bias <- rep(-1,N)
entradas2 <- cbind(amostra2,bias)
#capturando apenas a saida y esperada
y2 <- teste[,n+1]

#ALGORITMO DE CLASSIFICACAO
#cria um vetor para receber todas as previsoes realizadas pela rede
yhat2 <- rep(NA,N)
#armazena todas as previsoes. uma para cada elemento da amostra.
for (i in 1:N)
{
  yhat2[i] <- previsao(w,entradas2[i,])
}

#converte o -1 em 0 para uso da funcao que gera a matrix de confusao
yhat2[which(yhat2==-1)] <- 0
y2[which(y2==-1)] <- 0

#gera a matrix de confusao
confusao <- confusionMatrix(table(y2,yhat2))
print(confusao)

#acerta os dados para geracao de um grafico de barras
resultado <- c(confusao[2]$table[1],
               confusao[2]$table[2],
               confusao[2]$table[4],
               confusao[2]$table[3])
barplot(resultado, names.arg = c("0-0","0-1","1-1","1-0"),col = c("green","red","green","red"),xlab="Previsoes",y
```

Fonte: O autor.

Figura 9: Qustão 4

```
[1] "Conjunto ideal de pesos:"
[1] -0.08923511 -0.42405635  0.49005759  0.28511552 -0.08994253
[1] "Epocas necessarias a convergencia:"
[1] 2
Confusion Matrix and Statistics

   yhat2
y2   0  1
  0 20  0
  1  0 20

               Accuracy : 1
                 95% CI : (0.9119, 1)
```

Fonte: O autor.

# 5 Qustão 5

Nesta questão pede-se que se altere a tolerância e a taxa de aprendizagem e foram aterados para 1 e 0,08 respectivamente. E foi obtido o seguinte retorno:

Figura 10: Qustão 5

```
[1] "Conjunto ideal de pesos:"
[1] -0.27727201 -0.58735853  0.91352274  0.18724338  0.06722783
[1] "Epocas necessarias a convergencia:"
[1] 1
> |
```

Fonte: O autor.

Figura 11: Qustão 5


Fonte: O autor.

Ao alterar novamente os valores de tolerância e taxa de aprendizagem para 2 e 0,09 respectivamente, obtemos o mesmo gráfico da figura 11 o seguinte retorno:

Figura 12: Qustão 5

```
[1] "Conjunto ideal de pesos:"
[1] -0.0683199 -0.5737859  0.9836087  0.4639317  0.2187337
[1] "Epocas necessarias a convergencia:"
[1] 1
>
```

Fonte: O autor.

Contudo, podemos concluir que a taxa de aprendizagem e a tolerância são dois parâmetros cruciais para o treinamento de RNAs. Eles desempenham papéis fundamentais na determinação de como o modelo é atualizado durante o processo de aprendizagem.

A taxa de aprendizagem determina a velocidade com que o modelo converge para uma solução ótima. Uma taxa de aprendizagem muito alta pode fazer com que o modelo pule a solução ideal. Já a tolerância é um critério de parada usado durante o treinamento para determinar quando o modelo atingiu uma precisão ou erro aceitável. Geralmente, é definida como um valor pequeno e representa a diferença máxima permitida entre as saídas previstas pela RNA e as saídas esperadas.