Redes Neurais Artificiais

André Costa Werneck, Matrícula: 2017088140

LISTA 3

24/04/2022

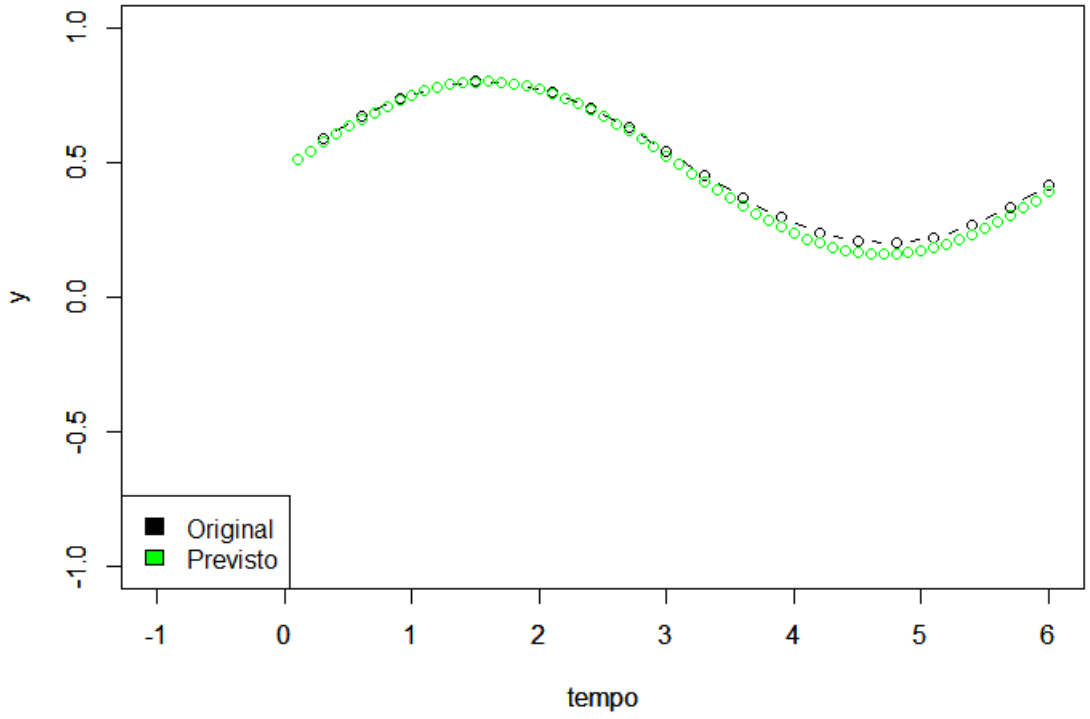
1) Ao realizar a leitura do presente exercício, conclui-se que ele apresenta o objetivo de realizar a implementação do modelo do Adaline e, com ela, providenciar a familiarização dos alunos com seu algoritmo, ou seja, com o modo de funcionamento do modelo tanto para problemas univariados, quanto para problemas multivariados.

Desta forma, usando como base os dados já fornecidos pelo professor como dados de treinamento, implementou-se e treinou-se o modelo. Os pesos obtidos foram os que seguem:

```
          [,1]
[1,] 0.3207739
[2,] 0.4787549
```

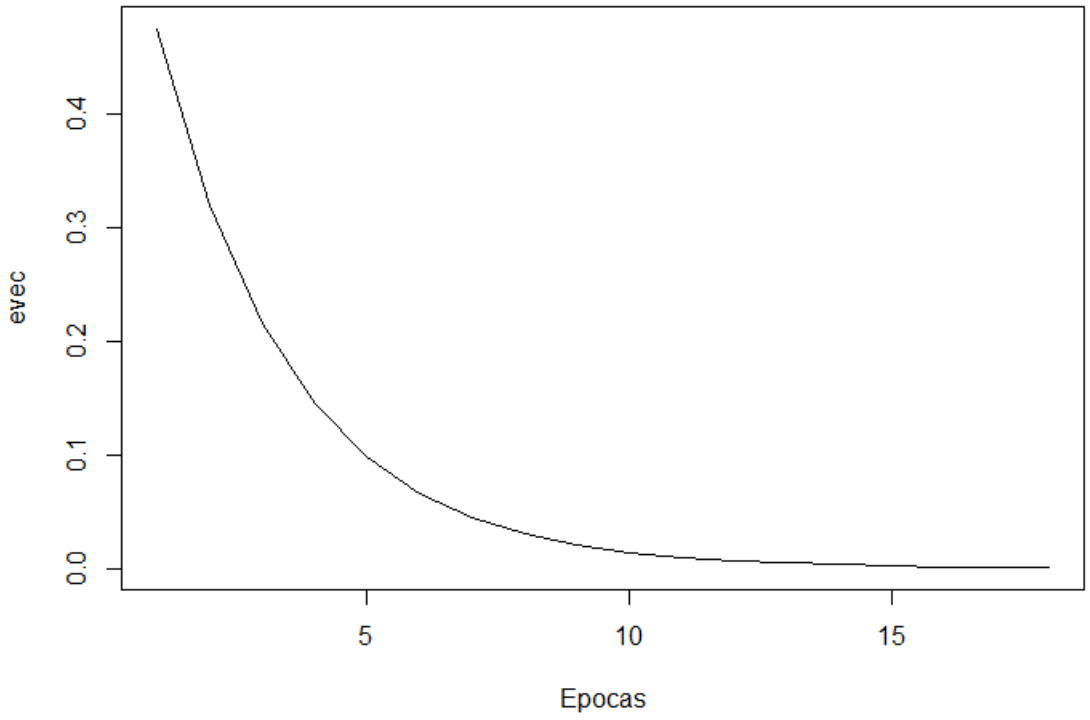
Nos quais o termo de polarização é o 2º.

E, após gerar novos dados de entrada para testar o modelo, realizou-se a avaliação e análise da seguinte curva aproximada:



E conclui-se que, visualmente, a aproximação está muito boa. Com isso, fica susceptível pensar que a função geradora desses dados de entrada é uma reta de equação y = 0.3x + 0.5 e, nossa análise pode ser estendida numericamente além de apenas visual. Logo, observa-se a semelhança entre os parâmetros obtidos para a função geradora, que no caso da saída do modelo, tem equação y = 0.321x + 0.479.

Vale ressaltar ainda que o erro de treinamento convergiu para quase zero dentro do número de épocas de treinamento previstas, assim como desejado. Segue curva do erro:

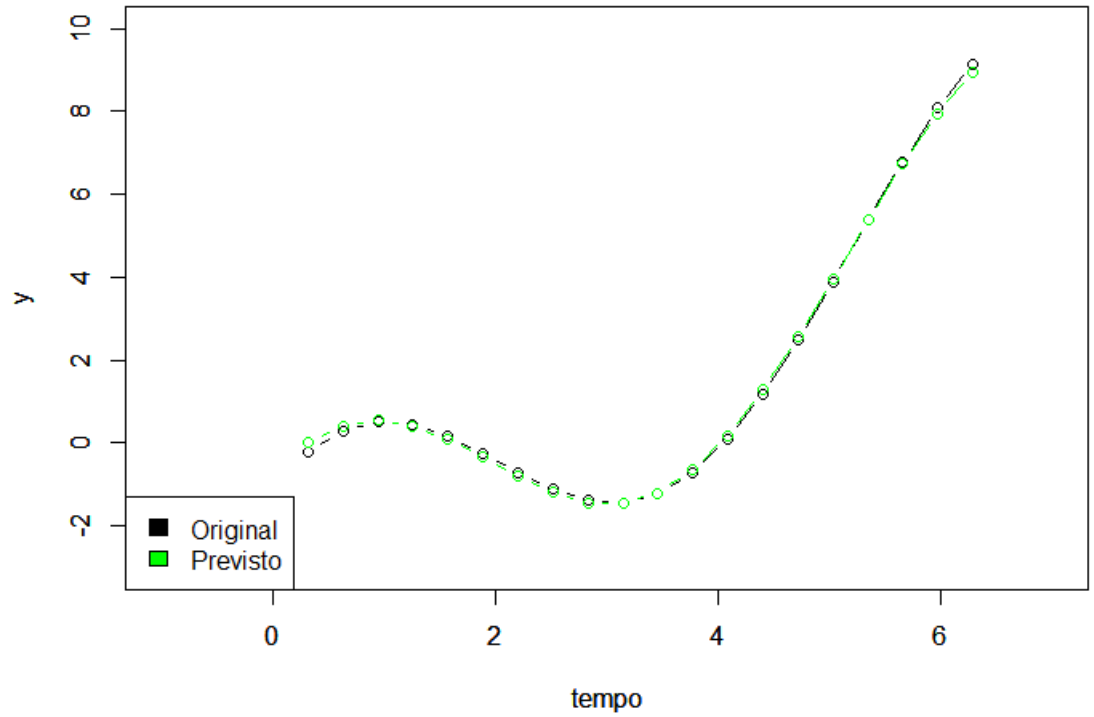


2) Para o exercício 2, utilizou-se a mesma implementação do Adaline, uma vez que ela é independente da dimensão uni ou multivariada. Partindo, também, de dados de entrada já fornecidos, treinou-se o modelo e obteve-se os seguintes pesos:

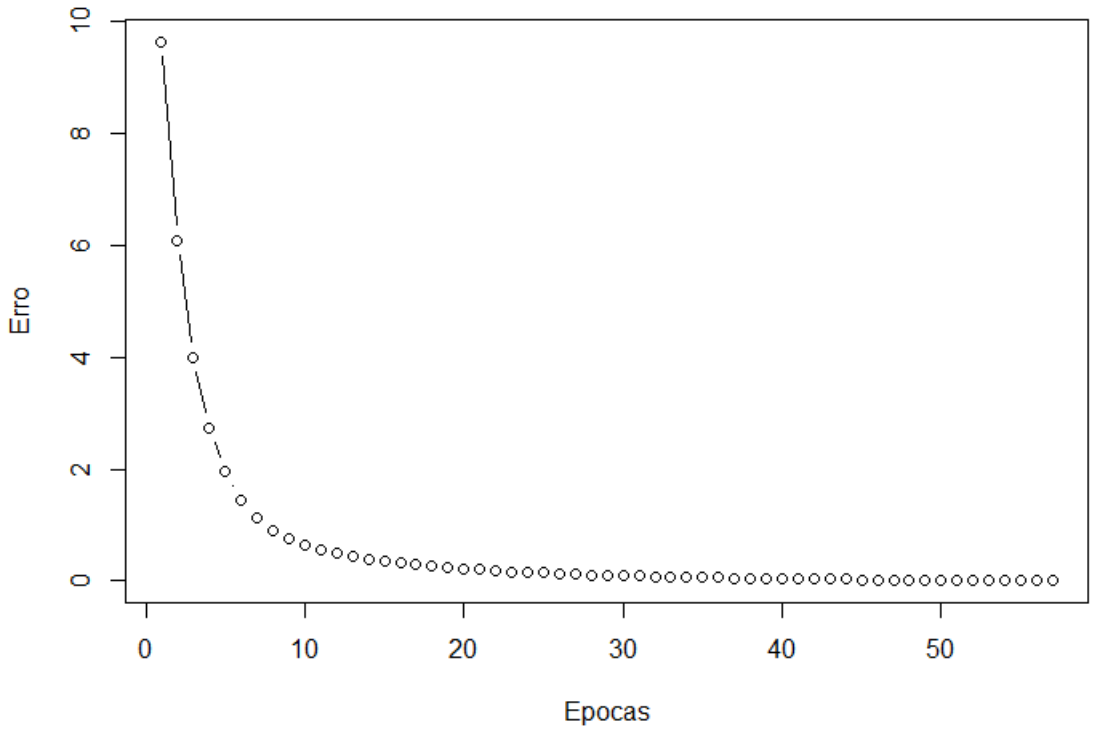
```
     [,1]
[1,] 0.8547059
[2,] 2.0229795
[3,] 2.8514893
[4,] 1.5709546
```

Nos quais o termo de polarização é o 4º.

Dessa forma, visando ter a mesma base de comparação fornecida pelo professor no documento do exercício, utilizou-se o mesmo conjunto de treino para teste. Obteve-se a seguinte curva:



Da curva acima, conclui-se mais uma vez que o treinamento foi bem sucedido para, uma vez que, visualmente, as curvas originais e previstas se assemelham muito. A mesma análise numérica feita para o exercício 1 pode aqui ser feita, bastando apenas arredondar os coeficientes do polinômio encontrado pelo Adaline devido a semelhança visual. Além disso, segue a curva do erro de treinamento:



Ela nos mostra que o treinamento para um problema multivariado demandou mais épocas para convergência do erro e aprendizado do modelo. Mas, mesmo assim, o modelo conseguiu ter sucesso no aprendizado e erro convergindo para 0 dentro do número de épocas proposto.

ANEXOS

O código para realização dos exercícios 1 e 2 se encontra devidamente documentado, através dos comentários, em anexo:

```
#EX1 -----

## Parsing the data
rm(list=ls())
t0 <- read.delim('Ex1_t')
x0 <- read.delim('Ex1_x')
y0 <- read.delim('Ex1_y')

t0<-strsplit(as.matrix(t0),split = ' ')
x0<-strsplit(as.matrix(x0),split = ' ')
y0<-strsplit(as.matrix(y0),split = ' ')

t<-list()
x<-list()
y<-list()

for (i in 1:20) {
  t<-append(t,as.numeric(t0[[i]][2]))
  x<-append(x,as.numeric(x0[[i]][2]))
  y<-append(y,as.numeric(y0[[i]][2]))
}

t<-matrix(unlist(t))
x<-matrix(unlist(x))
y<-matrix(unlist(y))

plot(t,x,type='l',col='red',xlim=c(-1,10),ylim=c(-1,1),xlab = "t",ylab = "x")
par(new=T)
plot(t,y,type='l',col='blue',xlim=c(-1,10),ylim=c(-1,1),xlab = "t",ylab = "x")

trainAdaline<- function(X,Y,eta,tolParada,maxEpocas){
  # pega as dimensoes de entrada
  N<-dim(X)[1]
  n<-dim(X)[2]
```

```
  # add coluna de 1s
  X<-cbind(X,1)
  #inicializa estocasticamente w
  w<-as.matrix(runif(n+1)-0.5)
  erroEpoca<-tolParada+1
  epocaAtual<-0
  vetorEpocas <- matrix(nrow = 1,ncol = maxEpocas)
  while ((erroEpoca>tolParada) && (epocaAtual<maxEpocas)){
    indexVec <- sample(N) # pega aleatoriamente os dados de treinamento
    erroQuad <- 0
    for (i in 1:N) {
      yhat <- 1.0 * (X[indexVec[i],] %*% w) #1x2 * 2x1 = 1x1
      erro <- Y[indexVec[i],] - yhat
      w <- w +eta*erro*X[indexVec[i],]
      erroQuad <- erroQuad + erro*erro
    }

    epocaAtual<-epocaAtual+1
    vetorEpocas[epocaAtual]<-erroQuad/N # erro medio
    erroEpoca<-vetorEpocas[epocaAtual] # atualiza o erro de cada epoca
  }
  result<-list(w,vetorEpocas[1:epocaAtual])
  return(result)
}


r<- trainAdaline(x,y,0.01,0.001,60)
pesos<- as.matrix(r[1][[1]])
evec<-as.matrix(r[2][[1]])

plot(1:length(evec),evec,type = 'l')

# gerando dados novos

tnew <- seq(0.1,6,0.1)
xnew <- as.matrix(sin(tnew))
# y = w1*x + w0
xnew <- cbind(xnew,1)
ynew <- xnew %*% pesos # 60x2 * 2x1 = 60x1
```

```
plot(t,y,type='b',col='black',xlim=c(-1,6),ylim=c(-1,1),xlab = "tempo",ylab = "y")
par(new=T)
plot(tnew,ynew,type='b',col='green',xlim=c(-1,6),ylim=c(-1,1),xlab = "tempo",ylab = "y")
legend('bottomleft',c('Original','Previsto'),fill = c('black','green'))

### Ex2 -------
rm(list=ls())
t2 <- as.matrix(read.table('t'))
x2 <- as.matrix(read.table('x'))
y2 <- as.matrix(read.table('y'))

#x2teste <- x2[11:20,]
#y2teste <- as.matrix(y2[11:20,])
#x2 <- x2[1:10,]
#y2 <- as.matrix(y2[1:10,])

trainAdaline<- function(X,Y,eta,tolParada,maxEpocas){
  # pega as dimensoes de entrada
  N<-dim(X)[1]
  n<-dim(X)[2]

  # add coluna de 1s
  X<-cbind(X,1)
  #inicializa estocasticamente w
  w<-as.matrix(runif(n+1)-0.5)
  erroEpoca<-tolParada+1
  epocaAtual<-0
  vetorEpocas <- matrix(nrow = 1,ncol = maxEpocas)
  while ((erroEpoca>tolParada) && (epocaAtual<maxEpocas)){
    indexVec <- sample(N) # pega aleatoriamente os dados de treinamento
    erroQuad <- 0
    for (i in 1:N) {
      yhat <- 1.0 * (X[indexVec[i],] %*% w) #1x2 * 2x1 = 1x1
      erro <- Y[indexVec[i],] - yhat
      w <- w +eta*erro*X[indexVec[i],]
      erroQuad <- erroQuad + erro*erro
    }

    epocaAtual<-epocaAtual+1
    vetorEpocas[epocaAtual]<-erroQuad/N # erro medio
    erroEpoca<-vetorEpocas[epocaAtual] # atualiza o erro de cada epoca
```

```
  }
  result<-list(w,vetorEpocas[1:epocaAtual])
  return(result)
}

r2 <- trainAdaline(x2,y2,0.01,0.01,200)
pesos2 <-r2[[1]]
evec2 <- r2[[2]]

plot(1:length(evec2),evec2,type = 'b',xlab = 'Epocas',ylab = 'Erro')

# y = w1*x1 + w2*x2 + w3*x3 + w0*1(x0)

#t22 <- seq(0.1,6,0.1)

#x21 <- as.matrix(sin(t22))
#x22 <- as.matrix(2*cos(t22))
#x23 <- as.matrix(t22 - 2.5)



x2new <- cbind(x2,1)
ynew2 <- x2new %*% pesos2

plot(t2,y2,type='b',col='black',xlim=c(-1,7),ylim=c(-3,10),xlab = "tempo",ylab = "y")
par(new=T)
plot(t2,ynew2,type='b',col='green',xlim=c(-1,7),ylim=c(-3,10),xlab = "tempo",ylab = "y")
legend('bottomleft',c('Original','Previsto'),fill = c('black','green'))
```