Redes Neurais Artificiais

Exercício 9 - MLP

Vítor Gabriel Reis Caitité - 2016111849

03/09/2021

Enunciado Exercício 9

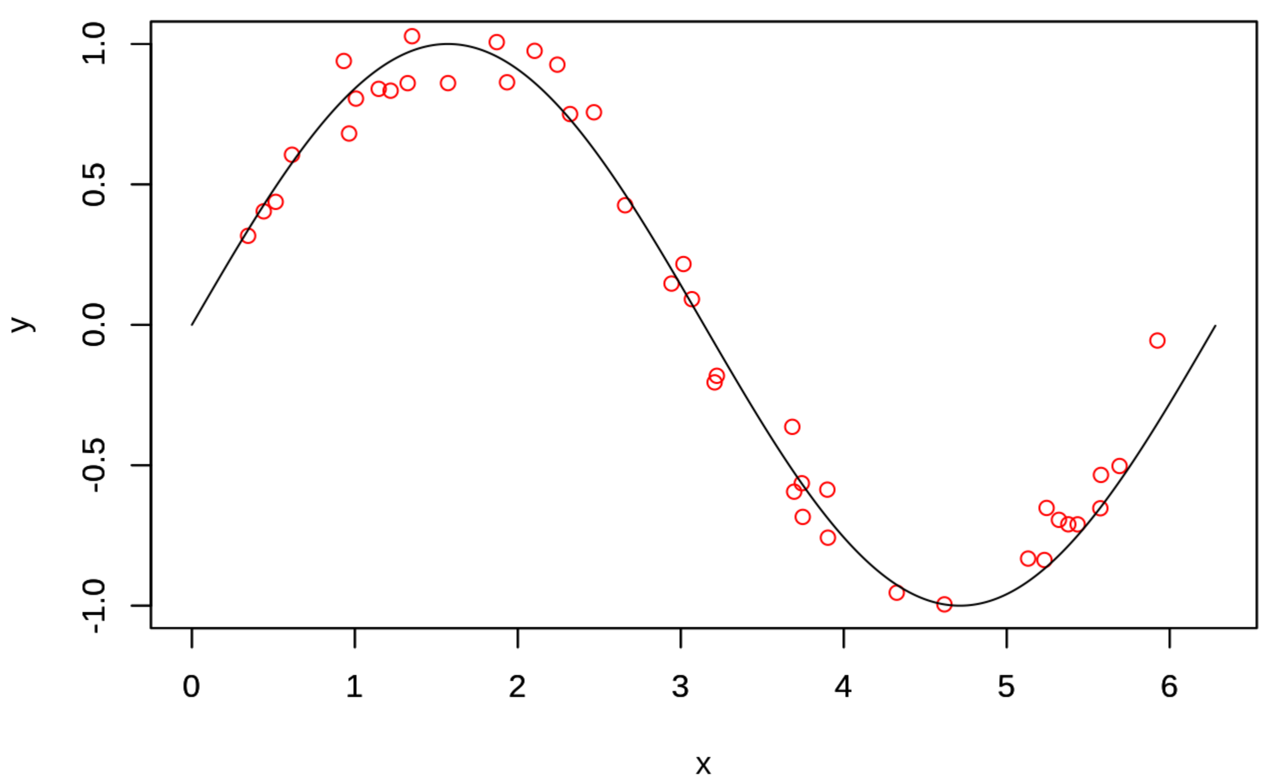
Para observar que o MLP é capaz de aproximar qualquer função contínua, deve ser realizada a regressão de um ciclo de uma senoide com backpropagation. A função de ativação da camada de saída deve ser linear, e a camada escondida deve ser composta de 3 neurônios. Deve ser adaptado o código desenvolvido na Vídeoaula 26.

O conjunto de treinamento deve ser constituído de 45 amostras com valores de x amostrados entre 0 e 2π e valores de y = seno(x) + ruído. O ruído deve ser uniformemente amostrado no intervalo [-0.1,0.1]. O conjunto de teste deve ser composto de valores de x entre entre 0 e 2π, obtidos com passo δ = 0.01, e y = seno(x).

Devem ser executadas 5 inicializações diferentes da rede MLP e, para cada uma, deve ser calculado o erro quadrático médio (MSE). Ao final das 5 execuções, devem ser apresentados a média e o desvio-padrão dos valores de MSE.

Para uma das execuções, deve ser gerado um gráfico comparando a saída da função aproximada e os valores esperados de y.

```
#Exemplo
x_train <- matrix(runif (45, 0, 2*pi), ncol=1)
y_train <- matrix(sin(x_train) + rnorm(45, 0, 0.1), ncol=1)

x_test <- seq(0, 2*pi, 0.01)
y_test <- sin(x_test)

plot(x_train, y_train, xlim = c(0,2*pi), ylim = c(-1,1), xlab = "x", ylab = "y", col='re
d')
par(new=T)
plot(x_test, y_test, type = 'l', xlim = c(0,2*pi), ylim = c(-1,1), xlab = "", ylab = "")
```



Resolução

```
rm(list=ls())

sech2<-function(u){
  return(((2/(exp(u)+exp(-u)))*(2/(exp(u)+exp(-u)))))
}
mse_total=rep(0,5);

for (execution in 1:5) {
  x_train <- matrix(runif (45, 0, 2*pi), ncol=1)
  y_train <- matrix(sin(x_train) + rnorm(45, 0, 0.1), ncol=1)

  x_test <- seq(0, 2*pi, 0.01)
  y_test <- sin(x_test)

  #plot(x_train, y_train, xlim = c(0,2*pi), ylim = c(-1,1), xlab = "x", ylab = "y", col
='red')
  #par(new=T)
  #plot(x_test, y_test, type = 'l', xlim = c(0,2*pi), ylim = c(-1,1), xlab = "", ylab =
 "")
  n<-1
  p<-3
  #Inicialização dos pesos.
  #Matriz Z nxp, neste caso n+1xp ==> 3x2
  Z<-matrix(runif((n+1)*p)-0.5,ncol=p,nrow=n+1)

  #Matriz W pxm, nestecaso p+1xm ==> 3x2
  W<-matrix(runif((p+1)*1)-0.5,ncol=1,nrow=p+1)

  xatual<-matrix(nrow=2,ncol=1)

  tol<-0.01
  eta<-0.01
  maxepocas<-2000
  nepocas<-0
  eepoca<-tol+1
  N<-45
  evec<-matrix(nrow=maxepocas,ncol=1)

  while((nepocas<maxepocas)&&(eepoca>tol))
  {
    ei2<-0
    #Sequênciaaleatóriadetreinamento.
    xseq<-sample(N)
    for(i in 1:N)
    {
      #Amostradadodasequênciaaleatória.
      irand<-xseq[i]
      xatual[1,1]<-x_train[irand,1]
      xatual[2,1]<-1

      yatual<-y_train[irand, 1]

      U<-t(xatual)%*%Z
      #xatualé3x1eZé3x2
      H<-tanh(U)
      Haug<-cbind(H,1)#Haugé1x3

      O<-Haug%*%W
      yhat<-O

      e<-yatual-yhat
      flinhaO<-1
      dO<-e*flinhaO#Produtoelementoaelemento

      Wminus<-W[-(p+1),]
      #Saı´dadoviésnãosepropaga
      ehidden<-dO%*%t(Wminus)#dO1x2eW3x2,ehiddené1x2
      flinhaU<-sech2(U)
      dU<-ehidden*flinhaU#Produtoelementoaelemento
```

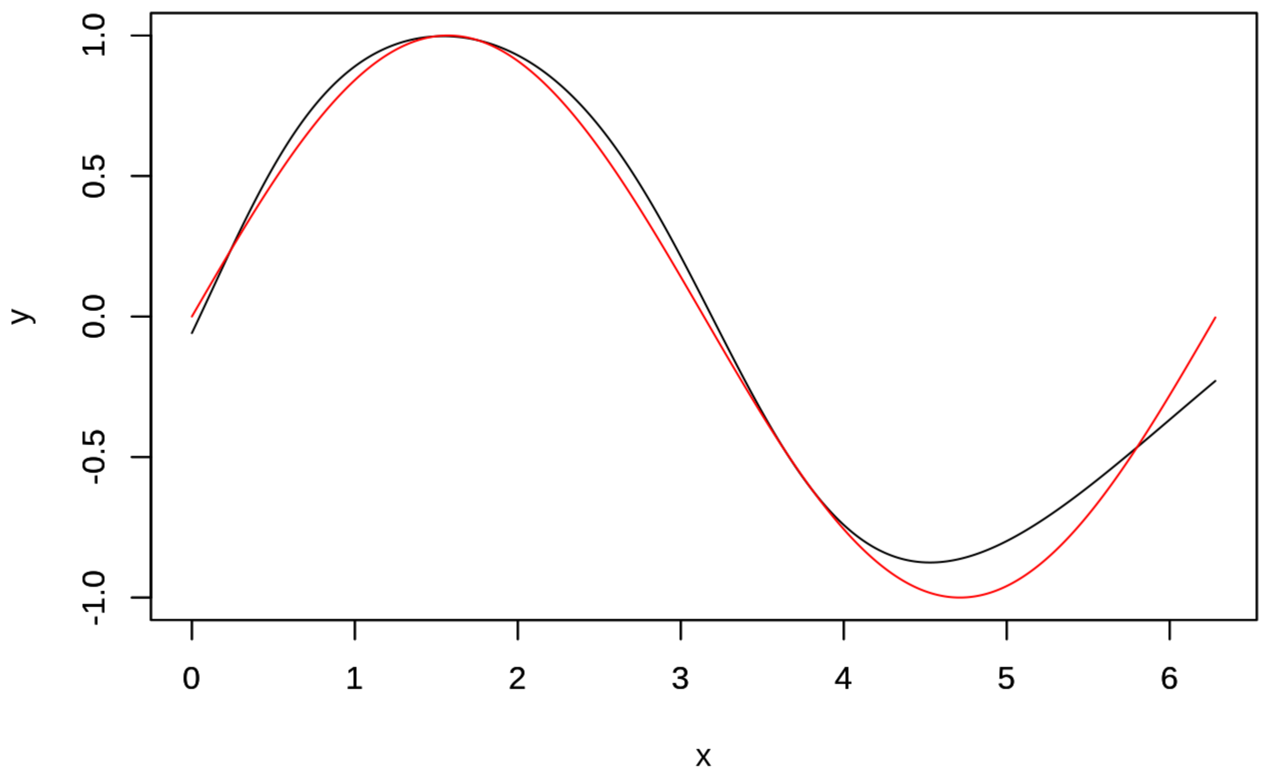
```
      W<-W+eta*(t(Haug)%*%dO)

      Z<-Z+eta*(xatual%*%dU)

      ei2<-ei2+(e%*%t(e))
    }
    #Incrementa número de épocas.
    nepocas<-nepocas+1
    evec[nepocas]<-ei2/N
    #Armazena erro por época.
    eepoca<-evec[nepocas]
  }
  print(paste("O erro quadrático médio da execução ", execution, " foi: ", ei2/N * 100,
"%"))
  mse_total[execution] = ei2/N
}
```

```
## [1] "O erro quadrático médio da execução  1  foi:  2.10767319428137 %"
## [1] "O erro quadrático médio da execução  2  foi:  1.55877318503274 %"
## [1] "O erro quadrático médio da execução  3  foi:  1.56834399111761 %"
## [1] "O erro quadrático médio da execução  4  foi:  0.97062358026724 %"
## [1] "O erro quadrático médio da execução  5  foi:  1.36723618372987 %"
```

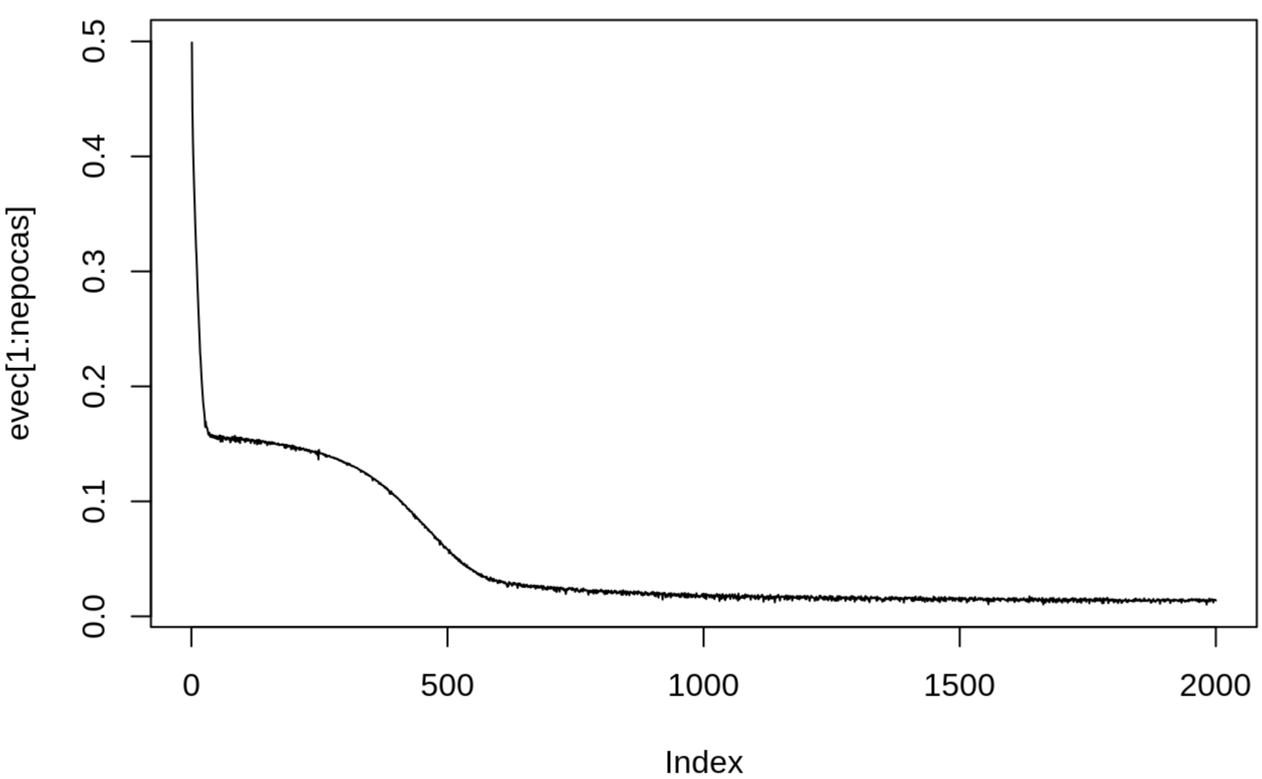
```
print(paste("O erro quadrático médio considerando as 5 execuções foi ", mean(mse_total)
 * 100, "±", sd(mse_total)*100, "%"))
```

```
## [1] "O erro quadrático médio considerando as 5 execuções foi  1.51453002688577 ± 0.41
0548996991247 %"
```

```
# TESTE:
u <- cbind(x_test,1) %*% Z
H<-tanh(u)
O<-cbind(H,1)%*%W
yhat_test<-O
plot(x_test, yhat_test,  type = 'l', xlim = c(0,2*pi), ylim = c(-1,1), xlab = "x", ylab
 = "y")
par(new=T)
plot(x_test, y_test, type = 'l', xlim = c(0,2*pi), col="red", ylim = c(-1,1), xlab = "",
 ylab = "")
```




```
plot(evec[1:nepocas],type='l')
```



Discussão

Com esse exercício pôde-se observar na prática como é construir um modelo utilizando perceptron de múltiplas camadas. Notou-se a eficiência do modelo gerado, mesmo utilizando poucos dados de treinamento (apenas 45), obteve-se menos de 3% de erro quadrático médio considerando as 5 execuções.

Como requisitado, para uma das execuções foi plotado o grafico comparando a função aproximada (em preto) e a função esperada (em vermelho). Além disso foi plotado o gráfico do erro a cada época. Nesse é possível observar a minimização desse erro durante o treinamento. No caso do grafico mostrado pode-se observar que o algoritmo atingiu o máximo de épocas estabelecido (2000). Isso ocorreu pois colocou-se uma tolerância baixa (0.01), fazendo o algoritmo percorrer todas as epocas sem atingir o critério de parada. Foi testado também utilizando uma tolerância um pouco maior (0.1) e percebeu-se que em menos de 200 épocas o treinamento já havia estabelecido o critério de parada.