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

Exercício 09: Redes Neurais Artificiais

Regressão de uma Senóide com Backpropagation

Neste exercício, teremos como objetivo implementar uma rede MLP no intuito de aproximar uma função contínua usando a regressão de um ciclo de uma senóide. Os valores de y foram definidos como uma função seno com a adição de ruído com seu valor variando entre - 0.1 e 1. O conjunto de treinamento inicial contém 45 amostras definidas de forma aleatória entre 0 e 2 π.

```
rm(list = ls())
library('plot3D')
library('plyr')
library('caret')
library('corpcor')
library('ggplot2')
library('insight')
library('knitr')
library('mlbench')
library('ctmle')

sech2 <- function(u){
  return(((2 /(exp(u) + exp(-u))) * (2/(exp(u) + exp(-u)))))
}

x <- runif(45, min = 0, max = 2*pi)
y <- sin(x) + runif(45, min = -0.1, max = 0.1)

# Bias
i1 <- 1
i3 <- 3

# Camada Intermediaria
w4_1 <- runif(1) - 0.5
w4_2 <- runif(1) - 0.5

w5_1 <- runif(1) - 0.5
w5_2 <- runif(1) - 0.5

w6_1 <- runif(1) - 0.5
w6_2 <- runif(1) - 0.5

# Camada de Saida
w7_3 <- runif(1) - 0.5
w7_4 <- runif(1) - 0.5
w7_5 <- runif(1) - 0.5
w7_6 <- runif(1) - 0.5
```

```r
# Treinamento
tol <- 0.01
nepocas <- 0
eepoca <- tol + 1
eta <- 0.01
max_epocas <- 2500
evec <- matrix(nrow = 1, ncol = max_epocas)
N <- length(x)

while((nepocas < max_epocas) && (eepoca > tol)) {
  ei2 <- 0
  iseq <- sample(N)

  # FEED FORWARD
  for(i in (1:N)) {
    #Entrada
    i2 <- x[iseq[i]]
    # Saida
    y7 <- y[iseq[i]]
    # Camada Intermediaria
    u4 <- i1*w4_1 + i2*w4_2
    u5 <- i1*w5_1 + i2*w5_2
    u6 <- i1*w6_1 + i2*w6_2

    i4 <- tanh(u4)
    i5 <- tanh(u5)
    i6 <- tanh(u6)

    # Camada de Saida
    u7 <- i3*w7_3 + i4*w7_4 + i5*w7_5 + i6*w7_6
    i7 <- u7

    # BACKPROPAGATION

    # Erros
    e7 <- y7 - i7
    # Deltas da saida
    d7 <- e7 * 1
    # Pesos para a camada de saida
    dw7_3 <- eta * d7 * i3
    dw7_4 <- eta * d7 * i4
    dw7_5 <- eta * d7 * i5
    dw7_6 <- eta * d7 * i6
    # Deltas dos neurônios da camada intermediaria
    d4 <- (d7*w7_4) * sech2(u4)
    d5 <- (d7*w7_5) * sech2(u5)
    d6 <- (d7*w7_6) * sech2(u6)
```

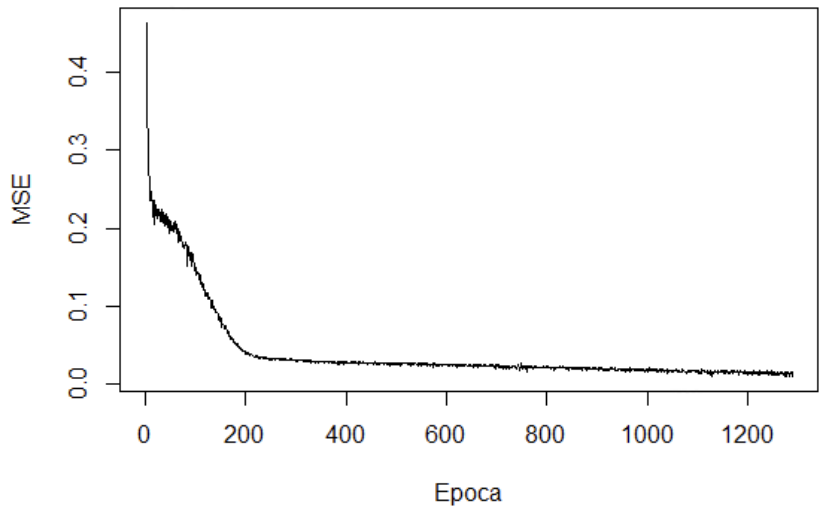
```
    # Pesos para a camada intermediaria
    dw4_1 <- eta * d4 * i1
    dw4_2 <- eta * d4 * i2
    dw5_1 <- eta * d5 * i1
    dw5_2 <- eta * d5 * i2
    dw6_1 <- eta * d6 * i1
    dw6_2 <- eta * d6 * i2

    # Ajuste de pesos
    # Camada de Saida
    w7_3 <- w7_3 + dw7_3
    w7_4 <- w7_4 + dw7_4
    w7_5 <- w7_5 + dw7_5
    w7_6 <- w7_6 + dw7_6

    # Camada Intermediaria
    w4_1 <- w4_1 + dw4_1
    w4_2 <- w4_2 + dw4_2
    w5_1 <- w5_1 + dw5_1
    w5_2 <- w5_2 + dw5_2
    w6_1 <- w6_1 + dw6_1
    w6_2 <- w6_2 + dw6_2

    # Calculo do Erro
    ei <- e7 * e7
    ei2 <- ei2 + ei
  }
  # Mudança de Epoca
  nepocas <- nepocas + 1
  evec[nepocas] <- ei2 / N
  eepoca <- evec[nepocas]
}

plot(evec[1, (1:nepocas)], type = 'l', xlab = 'Epoca', ylab = 'MSE')
```



Ao observarmos acima, ao realizar o treinamento com um limite de 2500 épocas e adicionam uma tolerância de 0.01 no erro quadrático, observa-se como o erro converge com o passar das iterações.

```r
xrange <- seq(0, 2 * pi, 0.01)
yrange <- sin(xrange)
N <- length(xrange)
yrangehat <- list()
ei2 <- 0

for(i in (1:N)) {
  # Entrada
  i2 <- xrange[i]

  # Saida
  y7 <- y[i]

  # Camada Intermediaria
  u4 <- i1*w4_1 + i2*w4_2
  u5 <- i1*w5_1 + i2*w5_2
  u6 <- i1*w6_1 + i2*w6_2

  i4 <- tanh(u4)
  i5 <- tanh(u5)
  i6 <- tanh(u6)

  # Camada de Saida
  u7 <- i3*w7_3 + i4*w7_4 + i5*w7_5 + i6*w7_6
  i7 <- u7
  yrangehat[i] <- i7

  # Erros
  e7 <- y7 - i7

  # Calculo do Erro
  ei <- (e7 * e7) / 2
  ei2 <- ei2 + ei
}

plot(x, y, xlim = c(0, 2*pi), ylim = c(-1, 1), type = 'p', col = 'blue')
par(new = T)
plot(xrange, yrangehat, xlim = c(0, 2*pi), ylim = c(-1, 1), type = 'l', col = 'red', xlab = 'x', ylab = 'y')
legend(5, 0.7, legend = c("Treinamento", "Teste"), col = c("blue", "red"), lty = c(NA, 1), pch = c(1, NA), cex = 0.8, box.lty = 1)
```