Exercício 3 - Redes Neurais

Evandro Augusto Guimarães Alves

A função Trainadaline(), recebe como parâmetros um conjunto de entradas X , as saídas esperadas para as ‘n’ entradas X, o eta, a tolerância esperada, o número máximo de treinamento e por fim o parâmetro “par”, caso haja necessidade de colocar o bias.

Os dados de entrada, saída e tempo de amostragem tiveram que ser lidos de um arquivo e logo após passar por um tratamento como mostrado no trecho do código abaixo:

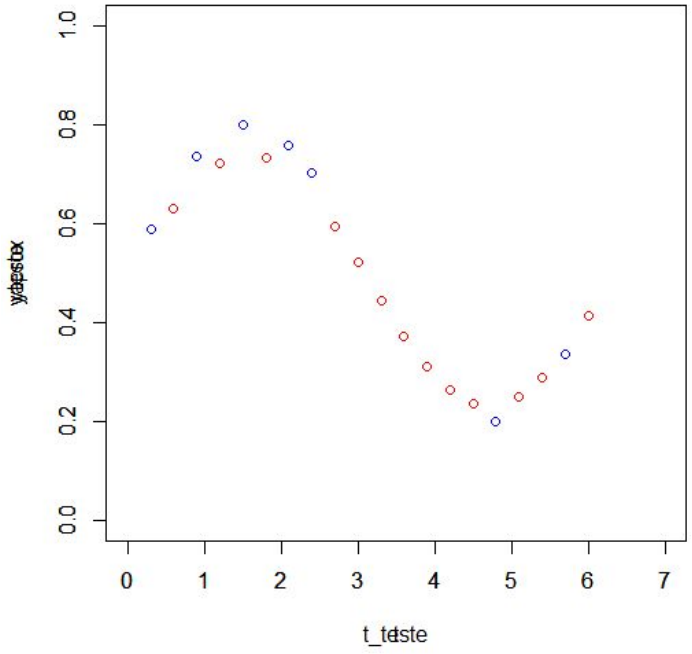
```
40  # Lendo os dados dos arquivos
41  t<-read.table("Ex1_t")
42  x<-read.table("Ex1_x")
43  y<-read.table("Ex1_y")
44  # transformando os dados em tipo matrix
45  tin <- as.matrix(t)
46  xin <- as.matrix(x)
47  yin <- as.matrix(y)
48
49  # pegando indices aleatorios do vetor
50  #index <- sample(NROW(xin)*0.7)
51  index <- sample(NROW(xin)) # indices aleatorios do vetor
52  index70 <- index[1:round(0.7*length(index)-1)] # primeiros 70% numeros do index
53  index30 <- index[round(0.7*length(index)):length(index)] # ultimos 30% numeros do index
54
55  # ultilizando os indices aleatorios para embaralhar as posicoes dos dados
56  x <- xin[index70,]
57  x <- as.matrix(x)
58  xteste <- xin[index30,]
59  xteste <- as.matrix(xteste)
60  #x[index[round(0.7*length(index))]]
61
62  # Y INDEX
63  y <- yin[index70,]
64  y <- as.matrix(y)
65  yteste <- yin[index30,]
66  yteste <- as.matrix(yteste)
67
68  # t INDEX
69  t <- tin[index70,]
70  t <- as.matrix(t)
71  t_teste <- tin[index30,]
72  t_teste <- as.matrix(t_teste)
73  Resul <- Trainadaline(x,y,0.001,0.001,1000,1)
74
75  H <- cbind(1,x) # a + bx
23:19  Trainadaline(X, y, eta, tol, maxepocas, par)  R Script
```

Os dados do arquivo foram lidos pela função read.table() e atribuídos a variáveis, pela função read.table(), pela função as.matrix() colocamos os dados de uma forma mais fácil de se trabalhar e que a função Trainadeline() conseguiria ler. Nas variáveis index70 e index30 é armazenado respectivamente 70% e 30% dos índices dos vetores de forma aleatória, usando essas variáveis consegue-se 70% dos dados para treinamento e os outros 30% dos dados para teste, assim foi feito entre as linhas “51” e “73” do programa.

Usamos 70% dos dados para o treinamento na função Trainadeline(), cujo uma das suas variáveis de retorno é uma lista com os pesos, esses pesos são atribuídos a variável “w” e usados para recuperar a função original, de acordo com as linhas 75 a 80 no código abaixo.

```
71  t_teste <- tin[index30,]
72  t_teste <- as.matrix(t_teste)
73  Resul <- Trainadaline(x,y,0.001,0.001,1000,1)
74
75  H <- cbind(1,x) # a + bx
76
77  w <- Resul[[1]]
78  w <- as.matrix(w) # pesos aproximados
79
80  yaprox = H %*% w # funcao aproximada
81
82  plot(t,yaprox,col='red',xlim=c(0,7),ylim=c(0,1))
83
84  par(new=T)
85  plot(t_teste,yteste,col='blue',xlim=c(0,7),ylim=c(0,1))
86
87  #Erro quadratico médio
88  N <- length(yteste)
89  Erro = y-yaprox
90  Erro = Erro^2
91  Erro2 = 0
92  for (i in 1:N)
93  {
94    Erro2 <- Erro2+ Erro[i]
95  }
96  Erro2 = Erro2/N
97  print(paste('O erro quadratico é:',Erro2))
```

Nos trechos entre a linha 82 e 85 do código são plotados respectivamente os pontos relacionadas aos dados aproximados e aos dados reais, que são vistos nas imagens abaixo também respectivamente pelos pontos vermelhos e azuis. E nas linhas 87 a 97 calcula e imprime o erro quadrático médio na tela.



Exercício 2

O código é basicamente o mesmo a única coisa que mudou é que agora, a função Trainadeline() recebe como parâmetro múltiplas entradas e após o treinamento a saída da função aproximada é plotada em vermelho e da saída dos dados originais em azul.