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

Exercício 3: Redes Neurais Artificiais

1. Modelo Univariado

Para a resolução da primeira atividade desta primeira lista, foi implementada a função de treinamento de Adaline, como pode ser vista logo abaixo:

```r
trainAdaline <- function(xin, yd, eta, tol, maxEpocas, par) {
  dimXin <- dim(xin)
  N <- dimXin[1]
  n <- dimXin[2]

  if(par == 1) {
    wt <- as.matrix(runif(n + 1) - 0.5)
    xin <- cbind(1, xin)
  } else {
    wt <- as.matrix(runif(n) - 0.5)
  }

  nEpocas <- 0
  eEpoca <- tol + 1
  evec <- matrix(nrow = 1, ncol = maxEpocas)

  while ((nEpocas < maxEpocas) && (eEpoca > tol)) {
    ei2 <- 0
    xSeq <- sample(N)

    for(i in 1:N) {
      iRand <- xSeq[i]
      yHat <- 1.0 * ((xin[iRand,] %*% wt))
      ei <- yd[iRand] - yHat
      dw <- eta * ei * xin[iRand,]
      wt <- wt + dw
      ei2 <- ei2 + ei * ei
    }

    nEpocas <- nEpocas + 1
    evec[nEpocas] <- ei2 / N
    eEpoca <- evec[nEpocas]
  }

  retList <- list(wt, evec[1:nEpocas])
  return(retList)
}
```

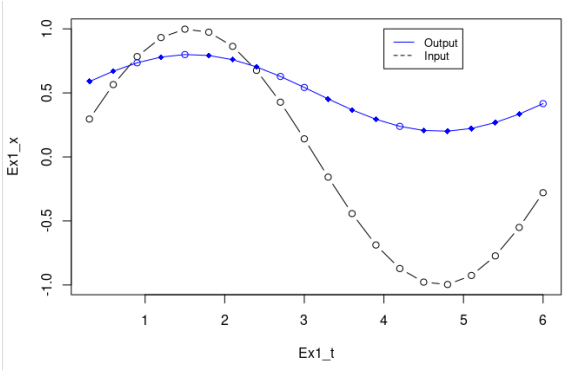
Figura 1. Implementação da função de treinamento de Adaline

Além da função, foram realizadas também a leitura dos dados disponibilizados conforme é mostrado no código a seguir:

```r
# Leitura de Arquivos:
Ex1_t <- as.matrix(read.table("~/Projetos/Trabalhos-Faculdade/Redes\ Neurais\ Artificiais/Exercicios/03/dados/Ex1_t"))
Ex1_x <- as.matrix(read.table("~/Projetos/Trabalhos-Faculdade/Redes\ Neurais\ Artificiais/Exercicios/03/dados/Ex1_x"))
Ex1_y <- as.matrix(read.table("~/Projetos/Trabalhos-Faculdade/Redes\ Neurais\ Artificiais/Exercicios/03/dados/Ex1_y"))

train = sample(1:20)

x_t <- as.matrix(Ex1_x[train[1:14]])
y_t <- as.matrix(Ex1_y[train[1:14]])
t_t <- as.matrix(Ex1_t[train[1:14]])
```

Figura 2. Leitura dos dados disponibilizados

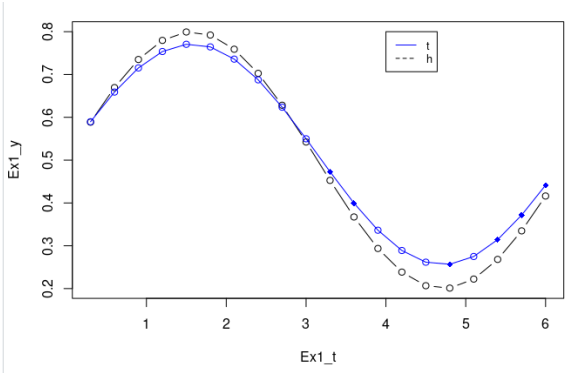
Com isso, foi implementado o primeiro plot com entrada e saída do modelo, nas Figuras 3 e 4, destacando os pontos amostrais utilizados no treinamento:




```r
plot(Ex1_t, Ex1_x, col = "black", type = "b")

for(k in 1:20) {
  if (k %in% train[1:14]){
    points(Ex1_t[k], Ex1_y[k], col = "blue", pch = 18)
  } else {
    points(Ex1_t[k], Ex1_y[k], col = "blue", pch = 21)
  }
}

lines(Ex1_t, Ex1_y, col = "blue")
legend(4, 1, legend = c("Output", "Input"), col = c("blue", "black"), lty = 1:2, cex = 0.8)
```

Figuras 3 e 4. Entrada e Saída de Dados

E a partir disso, foi realizada uma comparação entre o modelo gerado pela função de Adaline, destacado na cor Azul, e o modelo original na cor preta.

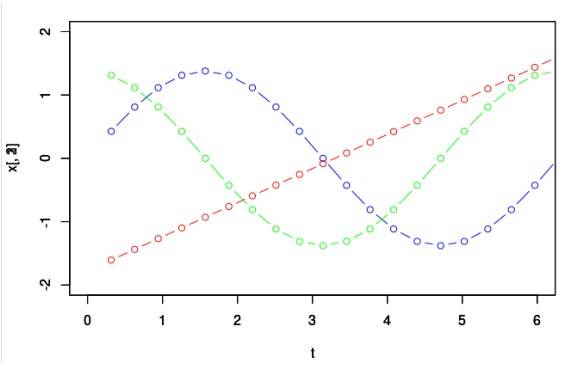



```r
retlist <- trainAdaline(x_t, y_t, 0.001, 0.001, 1000, 1)
w <- matrix(retlist[[1]], ncol = 1)
y_hat <- cbind(1, Ex1_x) %*% w

plot(Ex1_t, Ex1_y, col="black", type = 'b')

for (k in 1:20) {
  if (k %in% train[1:14]) {
    points(Ex1_t[k], y_hat[k], col = 'blue', pch = 21)
  } else {
    points(Ex1_t[k], y_hat[k], col = 'blue', pch = 18)
  }
}

lines(Ex1_t, y_hat, col = 'blue')
legend(4, 0.8, legend = c('t', 'h'), col = c('blue', 'black'), lty = 1:2, cex = 0.8)
```

Figuras 5 e 6. Comparação e implementação entre a saída real e a da função de Adaline

2. Modelo Multivariado

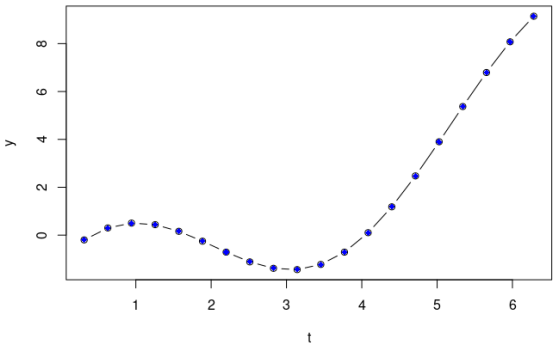
De maneira análoga ao que foi implementado no primeiro exercício desta lista, foi realizada uma implementação similar, agora com três diferentes entradas ao contrário de uma:




```
<<>>=
t <- as.matrix(read.table("~/Projetos/Trabalhos-Faculdade/Redes\ Neurais\ Artificiais/Exercicios/03/dados/t"))
x <- as.matrix(read.table("~/Projetos/Trabalhos-Faculdade/Redes\ Neurais\ Artificiais/Exercicios/03/dados/x"))
y <- as.matrix(read.table("~/Projetos/Trabalhos-Faculdade/Redes\ Neurais\ Artificiais/Exercicios/03/dados/y"))

treinamento = sample(1:20)

plot(t, x[,1], col = 'blue', type = 'b', xlim = c(0, 6), ylim = c(-2, 2))
par(new = T)
plot(t, x[,2], col = 'green', type = 'b', xlim = c(0, 6), ylim = c(-2, 2))
par(new = T)
plot(t, x[,3], col = 'red', type = 'b', xlim = c(0, 6), ylim = c(-2, 2))
@
```

Figuras 7 e 8. Parâmetros de Entrada do Modelo Multivariado

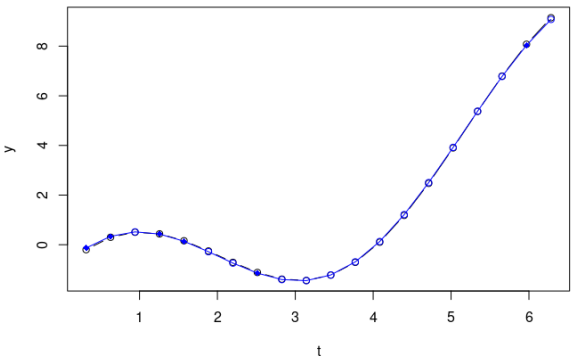
A partir destas entradas, foi possível gerar uma saída do modelo, destacando os pontos utilizados para o treinamento:




```
plot (t, y, col = 'black', type = 'b')

for (k in 1:20) {
  if (k %in% 1:20) {
    points(t[k], y[k], col = 'blue', pch = 18)
  } else {
    points(t[k], y[k], col = 'black', pch = 21)
  }
}
@
```

Figuras 9 e 10. Saída do Modelo

Por fim, foi comparado a saída após a chamada da função de treinamento de Adaline com a saída da figura anterior, denotadas logo abaixo:




```
retlist <- trainAdaline(x, y, 0.001, 0.001, 1000, 1)
w <- matrix(retlist[[1]], ncol = 1)
y_hat <- cbind(1, x) %*% w

plot(t, y, col = 'black', type = 'b')

for(k in 1:20) {
  if(k %in% treinamento[1:14]){
    points(t[k], y_hat[k], col = 'blue', pch = 21)
  } else {
    points(t[k], y_hat[k], col = 'blue', pch = 18)
  }
}

lines(t, y_hat, col = 'blue')
legend(4, 0.8, col = c('blue', 'black'), lty = 1:2, cex = 0.8)
```

Figuras 11 e 12. Comparação de Saídas

Com isso, foi observado que o modelo de adaline se aproximou muito bem do modelo, sendo bem eficaz para a resolução do problema, já que não foi observada alguma distorção relevante, em comparação ao modelo original. Assim, foi observado que o modelo satisfaz às especificações do exercício proposto.

3. Problema do Boston Housing

```r
# Implementacao do Problema Boston Housing
library("mlbench")
data("BostonHousing")

xall <- matrix(as.numeric(as.matrix(BostonHousing[, 1:13])), nc = 13)
yall <- matrix(as.numeric(as.matrix(BostonHousing[, 14])), nc = 1)

maxx <- max(xall)
xall <- xall / maxx
maxy <- max(yall)
yall <- yall / maxy

xyall <- cbind(xall, yall)
library(corrplot)
corrplot(cor(xyall), method = "number", type = "upper")
```

Figura 13. Implementação inicial do problema de Boston Housing

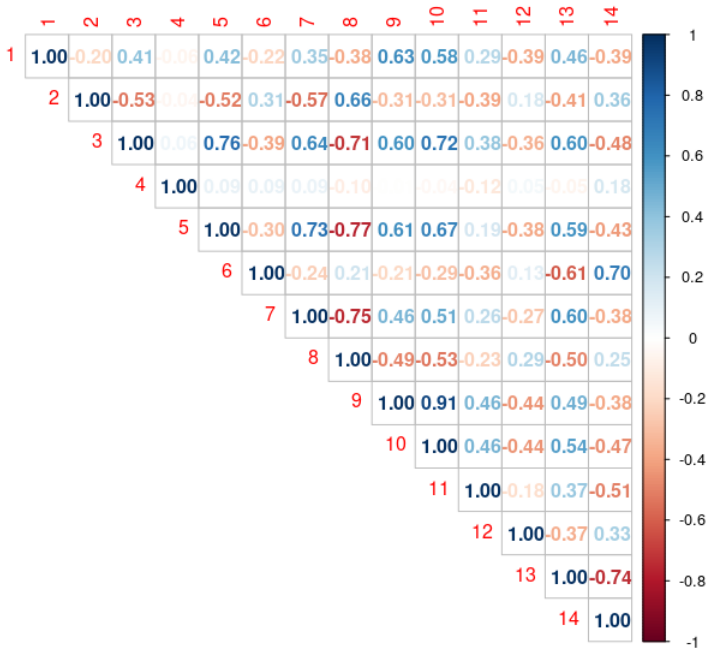


Figura 14. Matriz triangular do problema de correlações

```r
xseq <- sample(506)

xtrain <- as.matrix(xall[xseq[1:400],])
ytrain <- as.matrix(yall[xseq[1:400],])
xteste <- as.matrix(xall[xseq[401:506],])
yteste <- as.matrix(yall[xseq[401:506],])

retlist <- trainAdaline(xtrain, ytrain, 0.1, 0.01, 1000, 1)
w <- matrix(retlist[[1]], ncol = 1)
erro2 <- retlist[[2]]

plot(erro2, type = 'l', xlab = 'Epoca', ylab = 'Erro de Treinamento')
```

Figura 15. Segunda metade da implementação

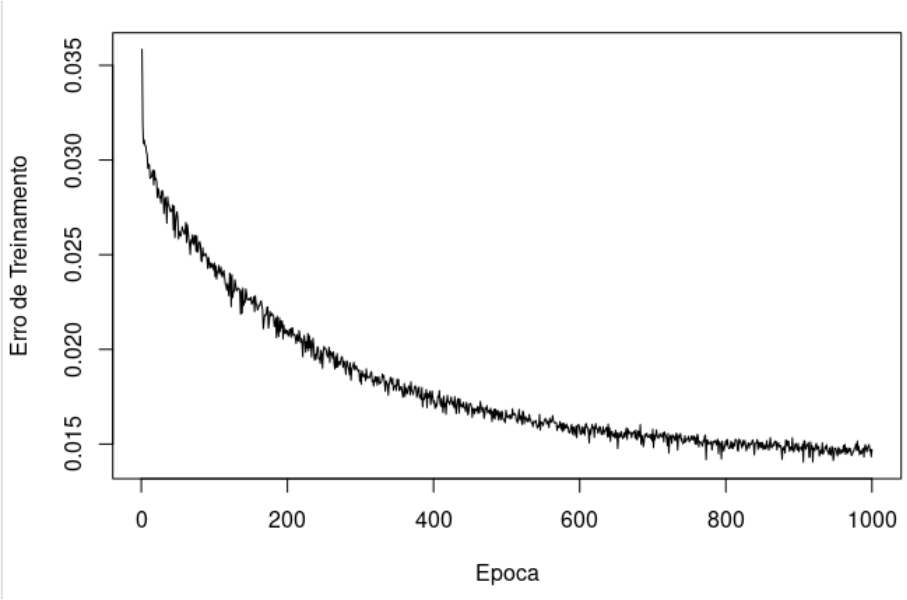


Figura 16. Plotagem do erro por época resultante