Universidade Federal de Minas Gerais

Exercício 3

Guilherme Vinícius Amorim

Como o *Adaline* funciona?

O *Adeline* é um algoritmo criado em 1960 e tem como objetivo resolver problemas lineares. O *Adeline* faz uso da expressão $Y = W * X$, onde Y é a saída, X é a entrada e W é o vetor de parâmetros. O algoritmo utilizado por esse método chega ao valor final dos parâmetros de W pelo método de minimização da função de custo quadrático, que, diferentemente do método dos mínimos quadrados, é feita de forma iterativa. Segue as fórmulas da função de custo quadrático, assim como a fórmula utilizada para se atualizar os parâmetros:

$$J = \sum_{i=1}^{N} (y_i - \hat{y}_i)$$

$$w(t+1) = w(t) + \eta * ei * xi, onde\ ei = y_i - \hat{y}_i$$

Segue abaixo o algoritmo implementado do *Adeline* em R:

```
trainadaline <- function(xin,yd,eta,tol,maxepocas,par){

  #yd: tem que ser garado para as xin (concatenado xall), metade 0 e metade 1
  #xin:Entrada Nxn de dados de matriz
  #eta: Peso de atualizacao do passo
  #tol: tolerancia do erro
  #maxepocas: numero maximo de epocas permitido
  #par: par=1 indica que -1 precisa ser acrescido a xinA

  N<-dim(xin)[1] #recebe as linhas
  n<-dim(xin)[2] # recebe as colunas

  if (par==1){
    wt<-as.matrix(runif(n+1)-10^(-n-2)) #inicializa um vetor de n+1 elementos
    xin<-cbind(1,xin)
  }
  if (par==0){
    wt<-as.matrix(runif(n)-0.5) #inicializa um vetor de n+1 elementos
  }

  nepocas<-0
  eepoca<-tol+1
  #inicializa vetor erro evec,
  evec<-matrix(nrow=1,ncol=maxepocas)
  while ((nepocas < maxepocas) && (eepoca>tol))#eepocas erro da epoca e tol tolerancia
  {
    ei2<-0
    #sequencia aleatoria para treinamento
    xseq<-sample(N)
    for (i in 1:N)
    {
      #padrao para sequencia aleatoria
      irand<-xseq[i]
      yhati<-xin[irand,] %*% wt
      ei<-yd[irand]-yhati
      dw<-eta * ei * xin[irand,]
      #atualizacao do peso w
      wt<-wt + dw
      #erro acumulado
      ei2<-ei2+ei*ei
    }
    #numero de epocas
    nepocas<-nepocas+1
    evec[nepocas]<-ei2/N
    #erro por epoca
    eepoca<-evec[nepocas]
  }
  retlist<-list(wt,evec[1:nepocas])
  return(retlist)

}
```

Figura 1: Algoritmo *Adeline* em R.

Exercício *Adaline*

Exercício 1

Um estudante de engenharia estava fazendo o estudo de um sistema e durante um intervalo de tempo ele observou na entrada (x) uma senoide diferente daquela encontrada na saída (y), o aluno concluiu que aquela senoide da entrada havia sido multiplicada por um termo e somada a outro de forma que $y = a + b * x$. O estudante então pediu a você para encontrar estes parâmetros utilizando os conceitos da *Adaline* que você aprendeu. Para isso ele te forneceu o tempo de amostragem $Ex1t$, os pontos de entrada $Ex1x$ e a saída $Ex1y$. Para achar os parâmetros você deverá usar 70% dos dados para treinamento e 30% para teste. Calcule o erro médio quadrático para as amostras de teste. Plote o gráfico da saída, considerando os parâmetros encontrados, para todos os pontos da entrada. Quais são os parâmetros do modelo?

O primeiro passo para o problema dado foi ler os dados fornecidos pelo aluno e separá-los em dados de treinamento e dados de teste. O código abaixo mostra como isso foi feito em R:

```
# Reading the sampling time, Input data, and output data
t <- as.matrix(read.table("Ex1_t", header = TRUE))
X <- as.matrix(read.table("Ex1_x", header = TRUE))
Y <- as.matrix(read.table("Ex1_y", header = TRUE))

# Separating 70% of the data for training and 30% for test
index <- sample(1:20, size = 14, replace = FALSE)
Xtraining <- as.matrix(X[index,])
Ytraining <- as.matrix(Y[index,])
Xtest <- as.matrix(X[-index,])
Ytest <- as.matrix(Y[-index,])
```

Figura 2: Código em R lendo-se os dados e separando-os em dados de treinamento e de teste.

Após separação entre dados de treinamento e dados de teste, foi-se utilizado o algoritmo do *Adaline* com os dados de treinamento para se encontrar os termos a e b do problema dado ($y = a + b * x$). Os parâmetros de entrada utilizados foram:

$$\eta = 0.01$$
$$tolerância = 0.0001$$
$$Número\ \max de\ épocas = 3000$$

Dessa forma, os parâmetros de retorno do algoritmo do *Adaline* são o vetor de parâmetros W e o vetor com o erro médio quadrático por época. Segue abaixo o valor retornado do vetor de parâmetros W, assim como um gráfico relacionando época e erro por época.

```
> W
           [,1]
[1,] 0.4988773
[2,] 0.3117777
```

Figura 3: Vetor de parâmetros W.

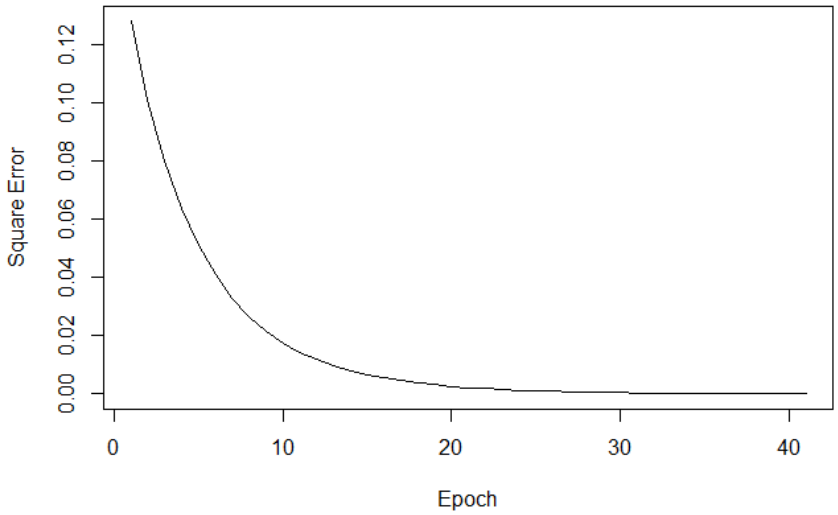

Figura 4: Gráfico com a evolução do erro quadrático a cada época no *Adaline*.

Após ter-se encontrado o vetor de parâmetros W, o erro médio quadrático foi calculado para a base de dados separada para ser testada. Segue abaixo o algoritmo para se encontrar o erro médio quadrático da base de testes, assim como o seu respectivo valor:

```
# Calculating the mean square error of data separated to be tested
YhatTest <- cbind(1, Xtest) %*% W
meanSquareError <- sum((YhatTest-Ytest)^2)/6
```

Figura 5: Algoritmo utilizado para o cálculo do erro médio quadrático da base de testes.

```
> meanSquareError
[1] 9.414624e-05
```

Figura 6: Erro médio quadrático encontrado a partir da base de testes

Segue abaixo o gráfico com a saída, considerando os parâmetros encontrados, para todos os pontos da entrada.

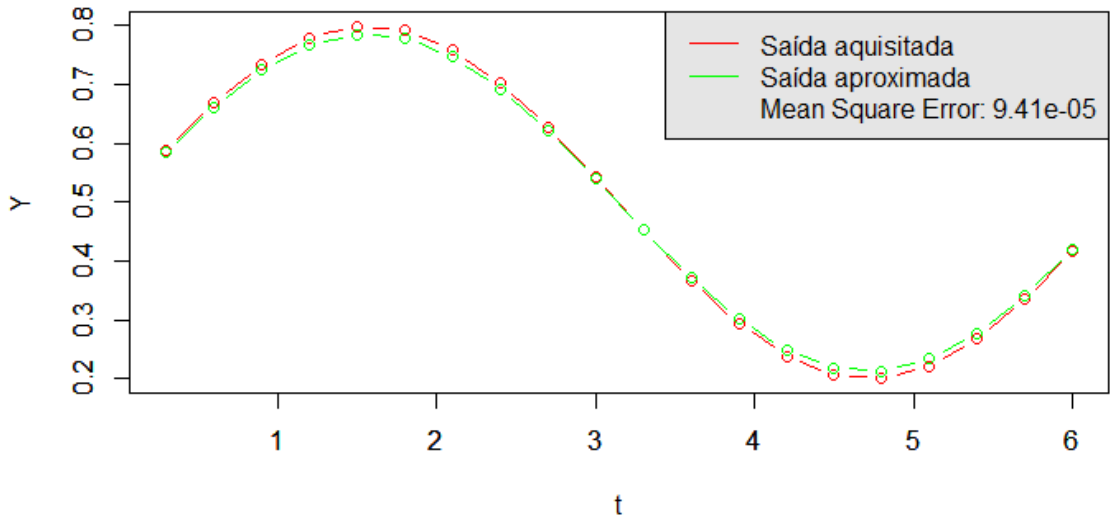

Figura 7: Gráfico com as saídas aproximada e amostrada.

Observa-se pelo gráfico que o resultado obtido foi muito próximo do esperado. Segue abaixo, portando, a solução do problema dado pelo estudante de engenharia:

$$y = a + b * x$$
$$w = [0.4988773, 0.3117777]^t$$

$$y = 0.3117777 + 0.4988773 * x$$

Exercício 2

O mesmo estudante de engenharia ficou admirado com seus conhecimentos técnicos sobre *Adaline* e resolveu pedir mais um favor. Ele observou que o novo sistema que ele estava trabalhando era constituído de três sinais na entrada e que a saída era uma mistura destes sinais da entrada mais um ganho. Mas este estudante não sabia muito bem como era esta mistura de sinais, a única coisa que ele sabia era que: $y = a + b * x1 + c * x2 + d * x3$. O aluno amostrou então os sinais na entrada e na saída para o intervalo de $[0.1\pi : 2\pi]$ e os armazenou nas variáveis t (tempos amostrais), x (entradas) e y (saída). Sendo que a primeira coluna de x é o sinal $x1$, a segunda $x2$ e a terceira $x3$. Para achar os parâmetros você deverá usar 70% dos dados para treinamento e 30% para teste. Calcule o erro médio quadrático para as amostras de teste e plote o gráfico da saída, considerando os parâmetros encontrados, para todos os pontos da entrada. Quais são os parâmetros do modelo?

Esse problema é muito similar ao problema anterior. Dessa forma, o processo de separar os dados para treinamento e para teste é idêntico em ambos os problemas. Dessa forma, o mesmo algoritmo para essa etapa foi utilizado:

```
# Reading the sampling time, Input data, and output data
t <- as.matrix(read.table("t", header = TRUE))
X <- as.matrix(read.table("x", header = TRUE))
Y <- as.matrix(read.table("y", header = TRUE))

# Separating 70% of the data for training and 30% for test
index <- sample(1:20, size = 14, replace = FALSE)
Xtraining <- as.matrix(X[index,])
Ytraining <- as.matrix(Y[index,])
Xtest <- as.matrix(X[-index,])
Ytest <- as.matrix(Y[-index,])
```

Figura 8: Código em R lendo-se os dados e separando-os em dados de treinamento e de teste.

Após essa separação dos dados, o algoritmo da *Adaline* foi utilizado nos dados de treinamento para se encontrar os parâmetros de interesse. As configurações de entrada do algoritmo foram as mesmas que foram utilizadas no problema anterior:

$$\eta = 0.01$$
$$tolerância = 0.0001$$
$$Número\ \max de\ épocas = 3000$$

Segue abaixo os valores do vetor de parâmetros W retornado pelo algoritmo, assim como um gráfico relacionando as épocas utilizadas no treinamento e seus respectivos erros médios quadráticos.

```
> W
          [,1]
[1,] 1.569651
[2,] 0.986315
[3,] 2.002463
[4,] 2.987736
```

Figura 9: Vetor de parâmetros W

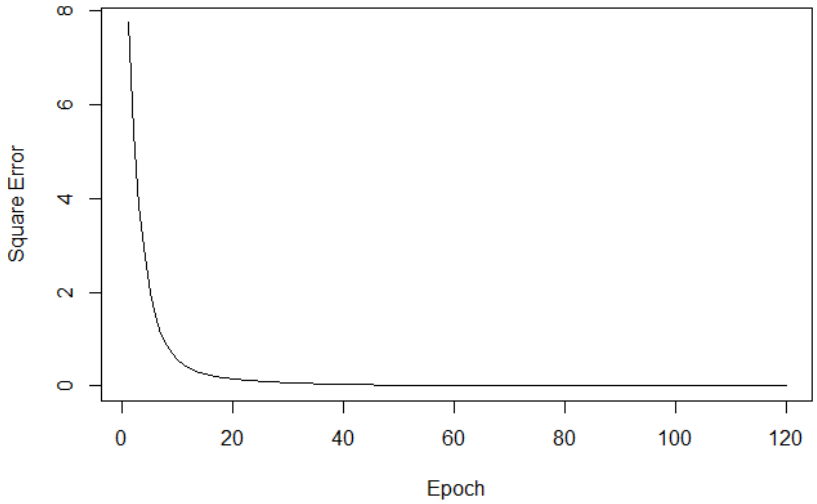

Figura 10: Gráfico com a evolução do erro quadrático a cada época no *Adaline*.

O algoritmo utilizado para calcular o erro médio quadrático da base de testes é o mesmo em ambos os problemas fornecidos pelo estudante. Dessa forma, segue abaixo a implementação do cálculo desse erro médio, assim como o seu valor encontrado:

```
# Calculating the mean square error of data separated to be tested
YhatTest <- cbind(1, Xtest) %*% W
meanSquareError <- sum((YhatTest-Ytest)^2)/6
```

Figura 11: Algoritmo utilizado para o cálculo do erro médio quadrático da base de testes.

```
> meanSquareError
[1] 8.007613e-05
```

Figura 12: Erro médio quadrático encontrado a partir da base de testes

Segue abaixo o gráfico com a saída, considerando os parâmetros encontrados, para todos os pontos da entrada.

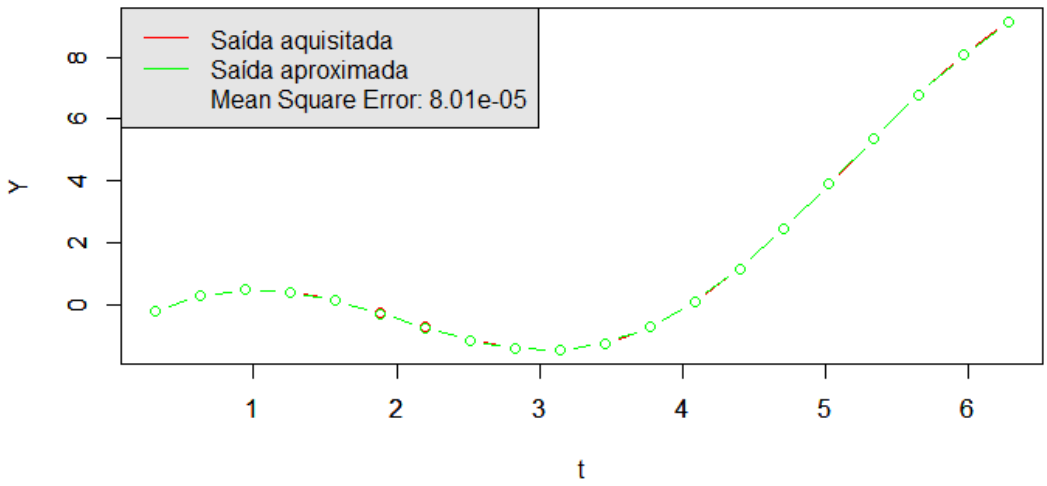

Figura 13: Gráfico com as saídas aproximada e amostrada.

Nesse problema em questão. Observa-se que a saída aproximada ficou muito próxima da saída amostrada pelo estudante, havendo sobreposição de vários pontos.

Segue abaixo, portando, a solução do problema dado pelo estudante de engenharia:

$$y = a + b * x1 + c * x2 + d * x3$$
$$w = [1.569651, 0.986315, 2.002463, 2.987736]^t$$
$$y = 2.987736 + 2.002463 * x1 + 0.986315 * x2 + 1.569651 * x3$$