EPC2: ADALINE

Guilherme Rocha Gonçalves[1], Rodolfo Delapicola[1]

Universidade de São Paulo
São Carlos, Brasil

1 Introdução

1.1 Contextualização

Segunda atividade do estudo sobre Redes Neurais Artificiais desenvolvida durante o curso da disciplina SEL5712 Redes Neurais Artificiais, oferecida no primeiro semestre do ano de 2019 para os alunos do programa de mestrado em Engenharia Elétrica e oferecido pelo Professor Dr. Ivan Nunes da Silva.

1.2 Objetivo

Esta atividade têm como objetivo a criação de uma rede neural ADELINE para solução de um problema de engenharia. Além disso, este trabalho pode ser considerado como um complemento ao que foi lecionado em sala de aula.

2 Materiais e Métodos

2.1 Problema

Uma rede neural artificial precisa ser treinada para minimizar (ou eliminar) a influência do efeito do ruído em um sinal que controla duas válvulas A e B.

2.2 Bases de treino e de teste

O treinamento da rede neural será feito usando a *Regra Delta*, também conhecida como *Regra de aprendizado de Widrow-Hoff*. A Regra Delta visa minimizar o *Erro Quadrático* entre o resultado da Função Somadora e a saída esperada, essa minimização do erro é obtida através do gradiente. Esse princípio torna única a solução do ADELINE, uma vez que o Erro Quadrático Mínimo é também único. Os dados de treino e de testes são fornecidos pelo EPC2.

2.3 Linguagem

Os códigos deste trabalho foram feitos usando *python*. Nenhuma biblioteca de Redes Neurais foram usadas, apenas bibliotecas para manipulação de matrizes e visualização de gráficos. O código está no Apêndice e também pode ser acessado através do link https://github.com/grgoncal/SEL5712—Redes-Neurais-Artificiais.

3 Resultados e discussão

3.1 Treinamento da Rede ADELINE

Foram feitos 5 treinamentos para a rede. O vetor de pesos foi iniciado com valores aleatórios de 0 a 1, a taxa de aprendizagem foi definida como 0.0025 e a tolerância foi definida para $10^{-}6$. O resultado dos treinamentos estão documentado na Tabela 1.

Treino	w0	w1	w2	w3	w4	w0	w1	w2	w3	w4	Épocas
1	0.82547749	0.29347523	0.05171485	0.08482625	0.42272082	-1.81136035	1.3125454	1.64137329	-0.42653662	-1.17712197	910
2	0.37872179	0.61415888	0.37956492	0.68539994	0.35938553	-1.8112957	1.31260652	1.64143071	-0.42634814	-1.17715072	897
3	0.71739028	0.56160827	0.24477393	0.16623727	0.43512531	-1.8113435	1.31255957	1.64138604	-0.426491	-1.17712818	901
4	0.40580157	0.78323681	0.94703793	0.40403682	0.94440323	-1.81130898	1.31259517	1.64142041	-0.42638449	-1.17714568	883
5	0.19850148	0.14063758	0.17215918	0.16957175	0.00205753	-1.81137254	1.31251465	1.64133835	-0.42661027	-1.17710261	871

Tabela 1: Resultado dos treinamentos.

3.2 Erro Quadrático Médio

Foram gerados os gráficos de Erro Quadrático Médio por épocas. É possível observar que os gráficos têm o mesmo comportamento entre sí. Os gráficos começam com um erro relativamente alto e caem até um valor fixo (e comum entre os casos de teste). Esse comportamento mostra o funcionamento da Regra Delta, minimizando cada vez mais o erro. As Figuras 1 e 2 mostram esse comportamento para os dois primeiros treinos.

Figura 1: Erro Quadrático Médio para o teste 1.

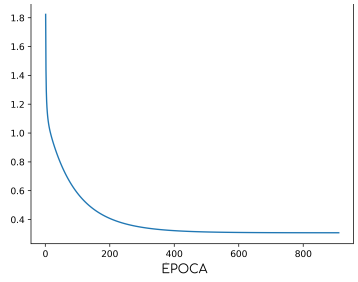
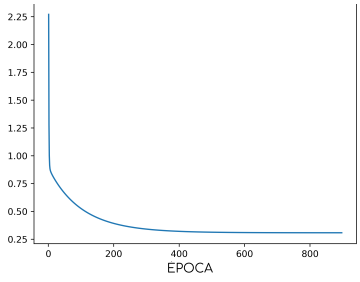


Figura 2: Erro Quadrático Médio para o teste 2.



3.3 Rede ADALINE em casos de testes

Foram fornecidos alguns objetos de teste para que a rede possa classificar após cada iteração de treinamento. Os resultados foram iguais para todos os casos, como era esperado. A Tabela 2 mostra os resultados.

Amostra	x1	x2	x3	x4	y(1)	y(2)	y(3)	y(4)	y(5)
1	0.9694	0.6909	0.4334	3.4965	-1	-1	-1	-1	-1
2	0.5427	1.3832	0.639	4.0352	-1	-1	-1	-1	-1
3	0.6081	-0.9196	0.5925	0.1016	1	1	1	1	1
4	-0.1618	0.4694	0.203	3.0117	-1	-1	-1	-1	-1
5	0.187	-0.2578	0.6124	1.7749	-1	-1	-1	-1	-1
6	0.4891	-0.5276	0.4378	0.6439	1	1	1	1	1
7	0.3777	2.0149	0.7423	3.3932	1	1	1	1	1
8	1.1498	-0.4067	0.2469	1.5866	1	1	1	1	1
9	0.9325	1.095	1.0359	3.3591	1	1	1	1	1
10	0.506	1.3317	0.9222	3.7174	-1	-1	-1	-1	-1
11	0.0497	-2.0656	0.6124	-0.6585	-1	-1	-1	-1	-1
12	0.4004	3.5369	0.9766	5.3532	1	1	1	1	1
13	-0.1874	1.3343	0.5374	3.2189	-1	-1	-1	-1	-1
14	0.506	1.3317	0.9222	3.7174	-1	-1	-1	-1	-1
15	1.6375	-0.7911	0.7537	0.5515	1	1	1	1	1

Tabela 2: Resultado dos testes.

4 Conclusão

Pelo fato de a rede ADALINE procurar pelo Erro Quadrático Mínimo, a solução encontrada é ótima e também é única. Isso acontece porque só existe apenas um valor mínimo do erro. Podem ocorrer pequenas variações nos pesos encontrados que ocorrem devido a influência da tolerância, porém, essa variação é mínima.

O número de épocas pode mudar entre treinos devido aos valores iniciais dos pesos, mas se manteve relativamente próximo entre os treinamentos, como foi observado nos resultado dos treinos, onde o número de épocas foi por volta de 900.

A ADALINE é portanto uma rede com solução ótima que pode ser usada como classificadora.

5 Apêndice

5.1 Código fonte

O código fonte dos algoritmos desenvolvidos está disponível no repositório público [1] para livre acesso.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

#//////////////////////
# CONSTANTS /////////
#//////////////////////

learningRate = 0.0025
errTol = 0.000001
trainNumber = 5
maxEpochs = 5000

#//////////////////////
# TRAIN ///////////////
#//////////////////////

def train(x, d, w, testNumber):
    # INITIALIZE EPOCHS AND ERR ------
    epoch = 0
    Eqm = 1
    lEqm = 0

    print ("\n[W INITIAL]   " + str(w.T.values))

    chart = [[],[]]

    while epoch < maxEpochs and abs(Eqm - lEqm) >= errTol:
        lEqm = Eqm
        Eqm = 0

        for i in range(0,x.shape[0]):
            u = w.iloc[:,0] * x.iloc[i,:]
            u = u.sum()
            w = (w.T + learningRate * (d.iloc[i] - u) * x.iloc
    [i,:]).T
            Eqm += (d.iloc[i] - u) * (d.iloc[i] - u)
        epoch += 1
        Eqm = Eqm/x.shape[0]
        chart[0].append(epoch)
        chart[1].append(Eqm)

    plt.figure(1)
    plt.plot(chart[0], chart[1])
    plt.savefig("./" + str(testNumber) + ".png", dpi = 500)
    plt.close()


    print ("[W FINAL]   " + str(w.T.values))
    print ("[EPOCHS]   " + str(epoch) + "\n")
```

```
50      return w
51
52  #////////////////////
53  # TEST#//////////////
54  #////////////////////
55
56  def test(w):
57      x = pd.read_csv('./test.csv', header = None)
58      for name in reversed(x.columns.values):
59          x.rename(columns = {name : name + 1}, inplace = True)
60      x.insert(loc = 0, column=0, value = np.full((x.shape[0],1)
        , -1))
61
62      y = []
63
64      for i in range(0, x.shape[0]):
65          u = w.iloc[:,0] * x.iloc[i,:]
66          u = u.sum()
67          if u >= 0:
68              y.append(1)
69          else:
70              y.append(-1)
71
72      return y
73
74  #////////////////////
75  # MAIN ///////////////
76  #////////////////////
77
78  # TRAIN DATASET ------
79  data = pd.read_csv('./train.csv', header = None)
80
81  # GET INPUTS, OUTPUTS AND WEIGTHS ------
82  x = data.iloc[:,1:(data.shape[1] - 1)]
              # GET INPUTS
83  x.insert(loc = 0, column=0, value = np.full((x.shape[0],1),
        -1))     # ADD -1 INPUT
84  d = data.iloc[:,(data.shape[1] - 1)]
              # GET OUTPUTS
85
86  for i in range(1, trainNumber + 1):
87      w = pd.DataFrame(np.random.rand((data.shape[1] - 1)))
88      print("[TRAINING NUMBER " + str(i) + "] --------")
89      y = test(train(x, d, w, i))
90      print("[RESULT OF TRAINING NUMBER " + str(i) + "] " + str(
        y) + "\n")
```

Listing 1.1: Python code

Referências Bibliográficas

[1] Gonçalves, G.R.: Github: Sel5712 - redes neurais artificiais. github.com/grgoncal/SEL5712—Redes-Neurais-Artificiais, acessado: 2019-04-05