Universidade de São Paulo
Escola de Engenharia de São Carlos
Departamento de Engenharia Elétrica e de Computação

Redes Neurais Artificiais

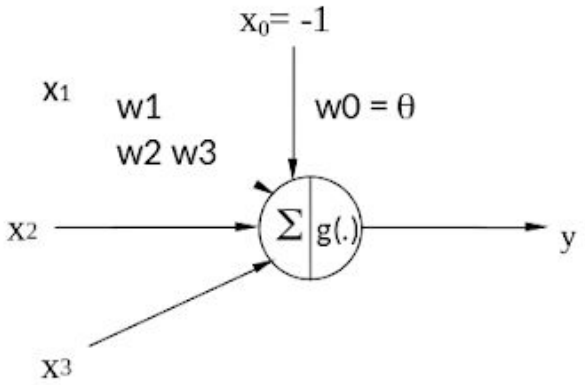
EPC1

Guilherme Rocha Gonçalves **grgoncal@gmail.com** **8006946**

A partir da análise de um processo de destilação fracionada de petróleo observou-se que determinado óleo poderia ser classificado em duas classes de pureza {C1 e C2}, mediante a medição de três grandezas {x_1, x_2 e x_3} que representam algumas das propriedades físico- químicas do óleo. A equipe de engenheiros e cientistas pretende utilizar um perceptron para executar a classificação automática destas duas classes.

Assim, baseadas nas informações coletadas do processo, formou-se o conjunto de treinamento em anexo, tomando por convenção o valor –1 para óleo pertencente à classe C1 e o valor +1 para óleo pertencente à classe C2.

Portanto, o neurônio constituinte do perceptron terá três entradas e uma saída, conforme ilustrado na figura abaixo:



Utilizando o algoritmo supervisionado de Hebb (regra de Hebb) para classificação de padrões, e assumindo-se a taxa de aprendizagem igual a 0.01, faça as seguintes atividades:

1. Execute 5 treinamentos para a rede perceptron, inicializando-se o vetor de pesos em cada treinamento com valores aleatórios entre zero e um. Se for o caso, reinicie o gerador de números aleatórios em cada treinamento de tal forma que os elementos do vetor de pesos iniciais não sejam os mesmos.

2. Registre os resultados dos 5 treinamentos na tabela seguinte:

Treinamento	Vetor de Pesos Inicial				Vetor de Pesos Final				Épocas
	w0	w1	w2	w3	w0	w1	w2	w3	
1º (T1)	*0.90518106*	*0.35501475*	*0.20326804*	*0.90178048*	*1.04518106*	*0.29796875*	*0.12178004*	*-0.22154552*	*2*
2º (T2)	*0.55919909*	*0.76994198*	*0.8311337*	*0.96034159*	*0.71919909*	*0.70461998*	*0.7413117*	*-0.28979041*	*2*
3º (T3)	*0.72717551*	*0.79284309*	*0.7107744*	*0.44688658*	*0.80717551*	*0.78800509*	*0.6635764*	*-0.12909142*	*2*
4º (T4)	*0.26593594*	*0.10589005*	*0.46855095*	*0.24276428*	*0.32593594*	*0.10619005*	*0.43481295*	*-0.16668372*	*2*
5º (T5)	*0.64827633*	*0.80172618*	*0.3532264*	*0.3293457*	*0.70827633*	*0.74426018*	*0.3340524*	*-0.1637783*	*2*

3. Após o treinamento do perceptron, aplique então o mesmo na classificação automática de novas amostras de óleo, indicando-se na tabela seguinte os resultados das saídas (Classes) referentes aos cinco processos de treinamento realizados no item 1.

Amostra	x1	x2	x3	y (T1)	y (T2)	y (T3)	y (T4)	y (T5)
1	*-0.3565*	*0.0620*	*5.9891*	*-1*	*-1*	*-1*	*-1*	*-1*
2	*-0.7842*	*1.1267*	*5.5912*	*-1*	*-1*	*-1*	*-1*	*-1*
3	*0.3012*	*0.5611*	*5.8234*	*-1*	*-1*	*-1*	*-1*	*-1*
4	*0.7757*	*1.0648*	*8.0677*	*-1*	*-1*	*-1*	*-1*	*-1*
5	*0.1570*	*0.8028*	*6.3040*	*-1*	*-1*	*-1*	*-1*	*-1*
6	*-0.7014*	*1.0316*	*3.6005*	*-1*	*-1*	*-1*	*-1*	*-1*
7	*0.3748*	*0.1536*	*6.1537*	*-1*	*-1*	*-1*	*-1*	*-1*
8	*-0.6920*	*0.9404*	*4.4058*	*-1*	*-1*	*-1*	*-1*	*-1*
9	*-1.3970*	*0.7141*	*4.9263*	*-1*	*-1*	*-1*	*-1*	*-1*
10	*-1.8842*	*-0.2805*	*1.2548*	*-1*	*-1*	*-1*	*-1*	*-1*

4. Explique por que o número de épocas de treinamento varia a cada vez que se executa o treinamento do perceptron.

Diferente do ADALINE, o *perceptron* permite que existam infinitas soluções. Portanto, dependendo dos parâmetros estáticos, tais como a taxa de aprendizagem e os valores iniciais do vetor de pesos, o número de épocas pode variar.

Se a taxa de aprendizagem for muito baixa, pode ser que mais épocas sejam necessárias para uma solução ser encontrada. Caso a taxa de aprendizagem for muito alta, o algoritmo pode ficar instável, aumentando em alguns casos o número de épocas ou na pior das hipóteses, impossibilitando a convergência.

No algoritmo desenvolvido para esse trabalho, o número de épocas aumentou quando a taxa de aprendizagem foi alterada para 0.001, chegando em até 8 épocas, em alguns casos.

5. Qual a principal limitação do perceptron quando aplicado em problemas de classificação de padrões.

O Perceptron só é capaz de classificar padrões entre duas classes linearmente separáveis, caso contrário, o algoritmo não irá convergir.

OBSERVAÇÕES:

1. O EPC pode ser realizado em grupo de três pessoas. Se for o caso, entregar somente um EPC com o nome de todos integrantes.

2. As folhas contendo os resultados do EPC devem ser entregue em sequência e grampeadas (não use clips).

3. Em se tratando de EPC que tenha implementação computacional, anexe (de forma impressa) o programa fonte referente ao mesmo.

ANEXO – Conjunto de Treinamento.

Amostra	x_1	x_2	x_3	d
01	*-0.6508*	*0.1097*	*4.0009*	*-1.0000*
02	*-1.4492*	*0.8896*	*4.4005*	*-1.0000*
03	*2.0850*	*0.6876*	*12.0710*	*-1.0000*
04	*0.2626*	*1.1476*	*7.7985*	*1.0000*
05	*0.6418*	*1.0234*	*7.0427*	*1.0000*
06	*0.2569*	*0.6730*	*8.3265*	*-1.0000*
07	*1.1155*	*0.6043*	*7.4446*	*1.0000*
08	*0.0914*	*0.3399*	*7.0677*	*-1.0000*
09	*0.0121*	*0.5256*	*4.6316*	*1.0000*
10	*-0.0429*	*0.4660*	*5.4323*	*1.0000*
11	*0.4340*	*0.6870*	*8.2287*	*-1.0000*
12	*0.2735*	*1.0287*	*7.1934*	*1.0000*
13	*0.4839*	*0.4851*	*7.4850*	*-1.0000*
14	*0.4089*	*-0.1267*	*5.5019*	*-1.0000*
15	*1.4391*	*0.1614*	*8.5843*	*-1.0000*
16	*-0.9115*	*-0.1973*	*2.1962*	*-1.0000*
17	*0.3654*	*1.0475*	*7.4858*	*1.0000*
18	*0.2144*	*0.7515*	*7.1699*	*1.0000*
19	*0.2013*	*1.0014*	*6.5489*	*1.0000*
20	*0.6483*	*0.2183*	*5.8991*	*1.0000*
21	*-0.1147*	*0.2242*	*7.2435*	*-1.0000*
22	*-0.7970*	*0.8795*	*3.8762*	*1.0000*
23	*-1.0625*	*0.6366*	*2.4707*	*1.0000*
24	*0.5307*	*0.1285*	*5.6883*	*1.0000*
25	*-1.2200*	*0.7777*	*1.7252*	*1.0000*
26	*0.3957*	*0.1076*	*5.6623*	*-1.0000*
27	*-0.1013*	*0.5989*	*7.1812*	*-1.0000*
28	*2.4482*	*0.9455*	*11.2095*	*1.0000*
29	*2.0149*	*0.6192*	*10.9263*	*-1.0000*
30	*0.2012*	*0.2611*	*5.4631*	*1.0000*

ANEXO – Código

O código está disponível na página:
https://github.com/grgoncal/SEL5712---Redes-Neurais-Artificiais

```
# IMPORTS -----------------------------------------------------------
import pandas as pd
import numpy as np
from numpy import genfromtxt

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

learningRate = 0.01
maxEpochs = 1000
errTol = 0
trainNumber = 5

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

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

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

  while(epoch < maxEpochs and err > errTol):
    err = 0
    for i in range(0,x.shape[0]):
      u = w.iloc[:,0] * x.iloc[i,:]
      u = u.sum()
      if u >= 0:
        y = 1
      else:
        y = -1
```

```
        if (y - d.iloc[i]) == 2:
            w = (w.T + learningRate * (d.iloc[i] - y) * x.iloc[i,:]).T
            err += 1

    print err
    epoch += 1

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

#////////////////////////////////////////////////////////////////
# TEST ////////////////////////////////////////////////////////
#////////////////////////////////////////////////////////////////

def test(w):
  x = pd.read_csv('./test.csv', header = None)
  for name in reversed(x.columns.values):
      x.rename(columns = {name : name + 1}, inplace = True)
  x.insert(loc = 0, column=0, value = np.full((x.shape[0],1), -1))

  y = []

  for i in range(0, x.shape[0]):
      u = w.iloc[:,0] * x.iloc[i,:]
      u = u.sum()
      if u >= 0:
          y.append(1)
      else:
          y.append(-1)

  return y

#////////////////////////////////////////////////////////////////
# MAIN ////////////////////////////////////////////////////////
#////////////////////////////////////////////////////////////////

# TRAIN DATASET ----------------------------------------------------
data = pd.read_csv('./train.csv', header = None)

# GET INPUTS, OUTPUTS AND WEIGTHS ----------------------------------
```

```
x = data.iloc[:,1:(data.shape[1] - 1)]                    # GET INPUTS
x.insert(loc = 0, column=0, value = np.full((x.shape[0],1), -1))    # ADD -1 INPUT
d = data.iloc[:,(data.shape[1] - 1)]                      # GET OUTPUTS

for i in range(1, trainNumber + 1):
  w = pd.DataFrame(np.random.rand((data.shape[1] - 1)))          # GENERATE WEIGHTS
  print("[TRAINING NUMBER " + str(i) + "] ------------------------------------------------")
  y = test(train(x, d, w))
  print("[RESULT OF TRAINING NUMBER " + str(i) + "] " + str(y) + "\n")
```