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

Luiz Felipe Machado Votto

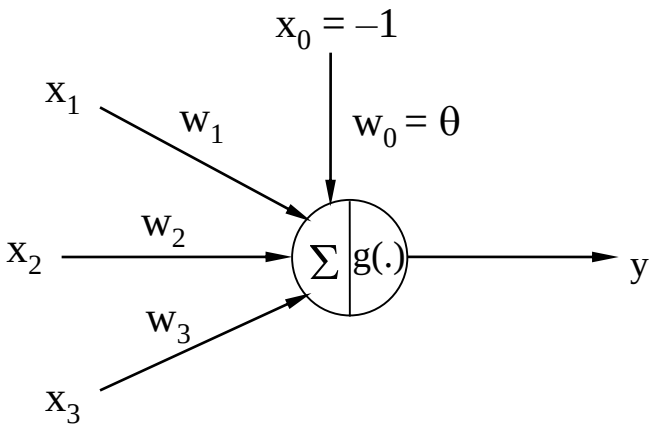
Redes Neurais Artificiais

EPC-1

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. -

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

Treinamento	Vetor de Pesos Inicial				Vetor de Pesos Final				Número de Épocas
	w_0	w_1	w_2	w_3	w_0	w_1	w_2	w_3	
1º (T1)	0,9912	0,1337	0,3620	0,5988	-1,5688	0,8044	1,2697	-0,3743	493
2º (T2)	0,3171	0,4403	0,7191	0,6830	-1,5729	0,7952	1,2626	-0,3736	464
3º (T3)	0,1814	0,8061	0,8002	0,0099	-1,5685	0,7910	1,2617	-0,3731	436
4º (T4)	0,7440	0,1794	0,6744	0,4409	-1,5360	0,7667	1,2300	-0,3649	454
5º (T5)	0,5202	0,8329	0,4150	0,9799	-1,5698	0,8025	1,2636	-0,3739	458

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	x_1	x_2	x_3	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.

   Pois cada treinamento é inicializado com um vetor aleatório de pesos. Assim sendo, o número de passos da regra de Hebb também é diferente até a convergência do algoritmo de treinamento.

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

   Um perceptron classifica padrões em apenas duas classes linearmente separáveis.

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.

Luiz Felipe Machado Votto

ANEXO 1 – 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

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

Luiz Felipe Machado Votto

ANEXO 2 – Implementação do Perceptron em Python

```
import numpy as np
import random

def step(x):
    return 0 if x < 0 else 1

def random_array(size=1, interval=[-1, 1]):
    return np.array([random.uniform(*interval) for
                     i in range(size)])

class Perceptron():
    training_set = None

    def __init__(self, dimension=2, learning_rate=1):
        self.dimension = dimension
        self.weights = random_array(size=dimension + 1, interval=[0,1])
        self.learning_rate = learning_rate

    def __call__(self, sample):
        if sample[0] != -1:
            sample.insert(0, -1)
        return step(np.dot(self.weights, sample))

    def _hebb_rule(self, answer, y, sample):
        self.weights = self.weights \
                       + self.learning_rate \
                       * (answer - y) * sample

    def train(self, sample_set, answer_set):
        self.training_set = (sample_set, answer_set)
        print("initialized with weights:")
        print(self.weights)
        iterations = 0
        error = True
        while error and iterations < 1E5:
            error = False
            iterations += 1
            for k in range(len(answer_set)):
                u = np.dot(self.weights, sample_set[k])
                y = step(u)
                if y != answer_set[k]:
                    self._hebb_rule(answer_set[k], y, sample_set[k])
                    error = True
        print('Done in %d iterations.' % iterations)

    @property
    def is_trained(self):
        return bool(self.training_set)
```