**Universidade Federal de Sergipe**

**Centro de Ciências Exatas e Tecnologia**

**Departamento de Computação**

**Disciplina: Redes Neurais**

**Profesor: Luís Brunelli**

**Exercício Hebb e Perceptron**

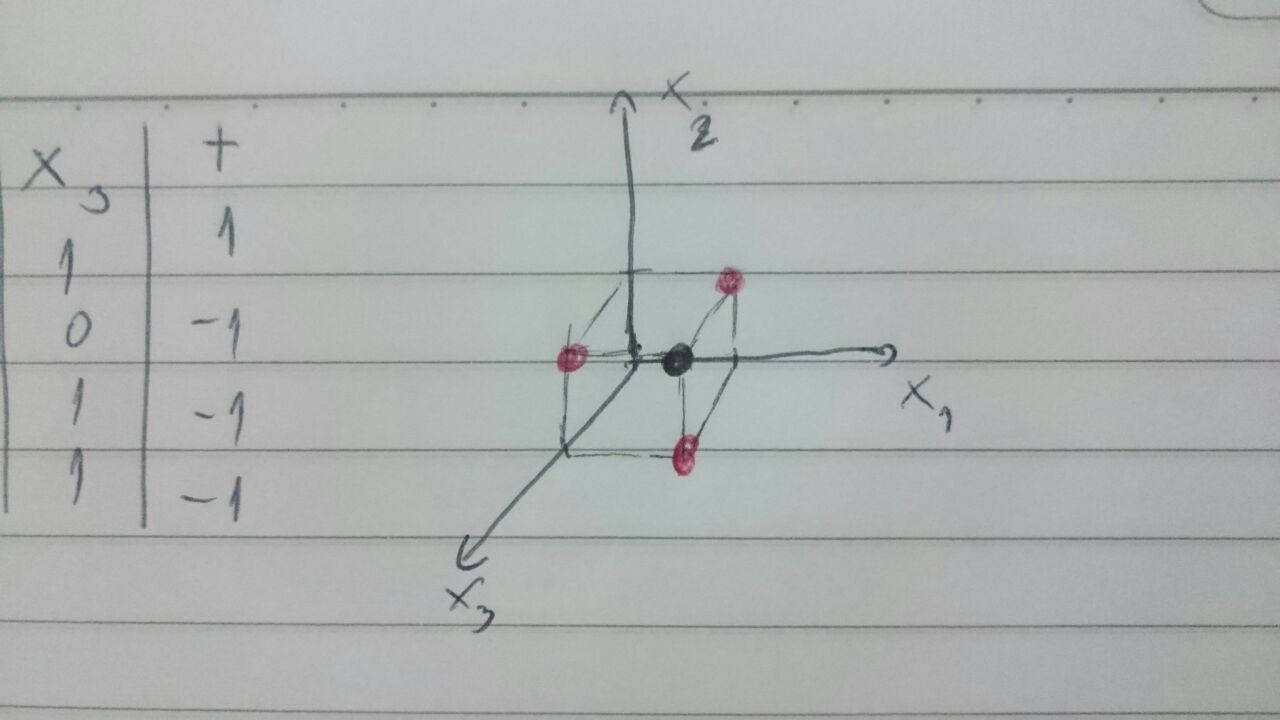
Thales Francisco Sousa Sampaio Alves dos Santos

São Cristóvão/ 2017

Considerando a tabela

|  |  |  |  |
| --- | --- | --- | --- |
| X1 | X2 | X3 | Target |
| 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | -1 |
| 1 | 0 | 1 | -1 |
| 0 | 1 | 1 | -1 |

Plotando os potnos no Gráfixo X1, X2 e X3 e colorindo como preto pra t=1 e vermelho pra t=-1 temos o seguinte gráfico



Pelo gráfico dá pra perceber que o problema não é separável por uma linha, precisa de ao menos um plano para separar o ponto preto (t=1) dos vermelhos (t=-1).

Resolvendo o problema utilizando a regra de Hebb chegamos à seguinte tabela:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| INPUT | | | | TARGET | Mudanças | | | | Pesos | | | | Equações |
| X1 | X2 | X3 | 1 | dW1 | dW2 | dW3 | db | w1 | w2 | w3 | b |
|  |  |  |  |  |  |  |  |  | 0 | 0 | 0 | 0 |  |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | X1+X2+X3+1=0 |
| 1 | 1 | 0 | 1 | -1 | -1 | -1 | 0 | -1 | 0 | 0 | 1 | 0 | X3=0 |
| 1 | 0 | 1 | 1 | -1 | -1 | 0 | -1 | -1 | -1 | 0 | 0 | -1 | X1=-1 |
| 0 | 1 | 1 | 1 | -1 | 0 | -1 | -1 | -1 | -1 | -1 | -1 | -2 | -X1-X2-X3-2=0 |

Resolvendo o sistema de equações chega-se a X2=2 e X2=-1 , o que é uma contradição, logo o sistema não tem solução

Resolvendo o problema utilizando o perceptron em C, a partir de uma solução de perceptron para duas variáveis utilizando dados aleatórios e não usando theta encontrada no endereço <https://github.com/RichardKnop/ansi-c-perceptron/blob/master/perceptron.c> fiz uma adaptação para inicialização sempre com zeros, funcionar com 3 variáveis e utilizando theta para definição de valores, resultando no código abaixo:

#include <stdio.h>

#include <stdlib.h>

#define LEARNING\_RATE 0.1

#define MAX\_ITERATION 100

#define THETA 0.1

int calculateOutput(float weights[], float x, float y, float z){

int resultado;

float sum = x \* weights[0] + y \* weights[1] + z \* weights[2] + weights[3];

if(sum>THETA) resultado = 1;

else if(sum<-(THETA)) resultado = -1;

else resultado = 0;

return resultado;

}

int main() {

srand(time(NULL));

float localError, globalError;

int patternCount, i, p, iteration, output;

float x[4] = {1.0,1.0,1.0,0.0};

float y[4] = {1.0,1.0,0.0,1.0};

float z[4] = {1.0,0.0,1.0,1.0};

int outputs[4] = {1,-1,-1,-1};

float weights[4] = {0.0,0.0,0.0,0.0};

patternCount = 4;

iteration = 0;

do {

iteration++;

globalError = 0;

for (p = 0; p < patternCount; p++) {

output = calculateOutput(weights, x[p], y[p], z[p]);

localError = outputs[p] - output;

weights[0] += LEARNING\_RATE \* localError \* x[p];

weights[1] += LEARNING\_RATE \* localError \* y[p];

weights[2] += LEARNING\_RATE \* localError \* z[p];

weights[3] += LEARNING\_RATE \* localError;

globalError += (localError\*localError);

}

printf("Pesos para a epoca %d: w1=%.2f w2=%.2f w3=%.2f bias=%.2f\n", iteration, weights[0], weights[1], weights[2], weights[3]);

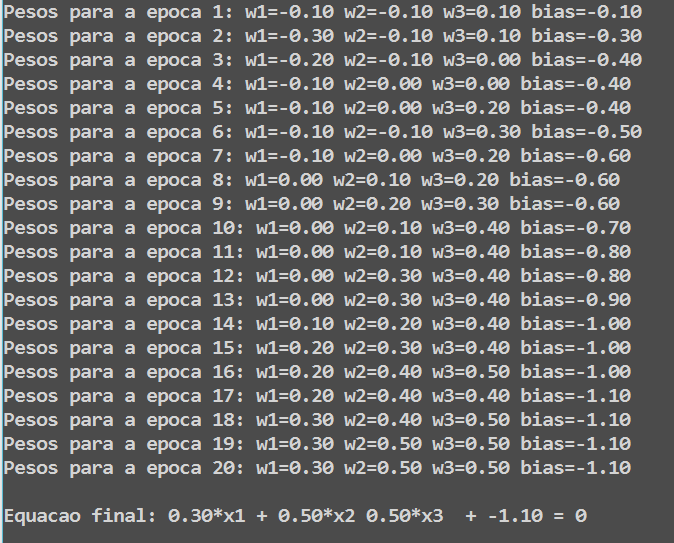
} while (globalError != 0 && iteration<=MAX\_ITERATION);

printf("\nEquação final: %.2f\*x1 + %.2f\*x2 %.2f\*x3 + %.2f = 0\n", weights[0], weights[1], weights[2], weights[3]);

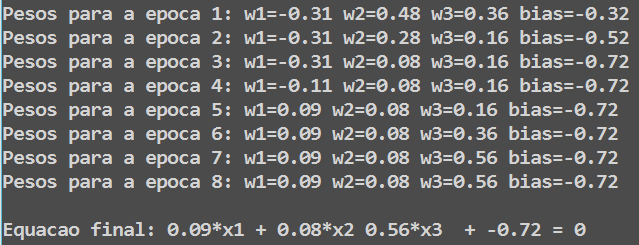
return 0;

}

Desta forma o código termina seu treinamento em 20 épocas:



Testes com inicialização randômica dos pesos e não utilizando um theta para decidir, utilizando apenas 1 ou -1 como resultado do output chegou a treinar entre 8 e 30 épocas, mas mantendo uma média de 20 épocas



Variações na taxa de aprendizado e no theta também fazem variar os resultados e número de épocas para treinamento, e percebe-se que o perceptron é um modelo de neurônio artificial muito mais efetivo que o Hebb que é apenas calculado e não pode ser treinado em várias épocas para o aprendizado das regras.