

Red de Hopfield

García Prado, Sergio

24 de diciembre de 2016

I. INTRODUCCIÓN

Las redes de Hopfield son un tipo de redes neuronales recurrente inicialmente desarrolladas por John Hopfield. Estas redes son usadas como sistemas de Memoria asociativa con unidades binarias y están diseñadas para converger a un mínimo local. A pesar de ello, la convergencia a alguno de los patrones almacenados no está garantizada.

Las unidades de las redes Hopfield son binarias, es decir, solo tienen dos valores posibles para sus estados (Esto se consigue a partir del uso de una función signo). El valor se determina si las unidades superan o no un determinado umbral. Los valores posibles pueden ser 1 ó -1,

Como caso de prueba, se ha realizado lo siguiente: Utilizando como modelo mental un cubo, se pretende introducir dos vértices opuestos del mismo, que serán utilizados como patrones que se pretenden reconocer a partir de pequeñas variaciones (el resto de vértices). La red de Hopfield obtendrá como resultado en cada caso, el patrón más próximo a cada entrada,

II. IMPLEMENTACIÓN

La implementación del algoritmo, así como su prueba, se han realizado en el language Octave (debido a la facilidad y simplicidad que ofrece cuando se codifican cálculos matemáticos). A continuación se describen cada una de las 3 partes en que se ha dividido la implementación: La primera de ellas consiste en un fichero *main*, mientras que la segunda y la tercera forman el propio algoritmo, correspondiéndose la primera fase al periodo de aprendizaje de patrones y la segunda a la clasificación de entradas.

```
function main()
    clear all;

    X = [-1, -1, 1;   1, -1, -1;   1, 1, 1;   -1, 1, 1;   -1, -1, -1;   1, 1, -1]
    P = [1, -1, 1;   -1, 1, -1]

    W = hopfield_learning(P)
    S = hopfield_working(W, X)
end;
```

Figura 1: Octave: *main.m*

$$\overline{X_i} = [x_{i1}, x_{i2}, \dots, x_{ij}, \dots, x_{in}], x_{ij} \in \{1, -1\} i \in (1, 2, \dots, r), \quad (1)$$

$$\overline{P_k} = [p_{k1}, p_{k2}, \dots, p_{kj}, \dots, p_{kn}], p_{kj} \in \{1, -1\}, k \in (1, 2, \dots, s), \quad (2)$$

Tal y como se muestra en el código de la figura 1, primero definimos tanto la matriz que representan los vectores que posteriormente serán utilizados como patrones como la que representa las entradas a clasificar. Estas matrices se pueden representar matemáticamente tal y como se indica en las fórmulas 1 y 2. Posteriormente se inicializa la matriz W encargada de almacenar los patrones, tal y como se explicará a continuación. Finalmente se calcula el resultado (vector más cercano entre los patrones por cada entrada).

```
function W = hopfield_learning(P)
W = zeros(size(P,2),size(P,2));
for i = 1:size(W,1);
    for j = 1:size(W,2);
        if(i ~= j);
            W(i,j) = sum(P(:,i) .* P(:,j));
        endif;
    endfor;
endfor;
end
```

Figura 2: Octave: *hopfield_learning.m*

$$w_{ij} = \begin{cases} \sum_{k=1}^s p_{ki}p_{kj} & \forall i \neq j; \\ 0 & \forall i = j; \end{cases} \quad (3)$$

El proceso de rellenado de la matrix W , en la cual se almacenan los patrones enseñados en la fase de aprendizaje, se puede modelizar matemáticamente tal y como se indica en la ecuación 3.

```
function S = hopfield_working(W,S)
for index = 1:size(S,1);
    for i = 1:size(S,2);
        do;
            S_old = S(index,:);
            v = sum(W(i,:) .* S_old);
            if (v > 0);
                S(index,i) = 1;
            elseif(v < 0);
                S(index,i) = -1;
            endif;
        until(S_old == S(index,:));
    endfor;
endfor;
end;
```

Figura 3: Octave: *hopfield_working.m*

El proceso de reconocimiento de patrones se realiza de manera recurrente, aplicando la función S definida de la matemáticamente en la ecuación 4 (Notese que u_i es una constante de ajuste, en la implementación realizada toma el valor 0 por lo que se ha obviado).

$$s_i(t) = \begin{cases} x_i & t = 0 \\ \text{sgn}(\sum_{j=1}^n w_{ij}s_j(t-1) - u_i) & t \neq 0 \end{cases} \quad (4)$$

La función signo ($\text{sgn}(x)$) se define en la ecuación 5. Como aclaración, para $x = 0$ esta función no toma ningún valor, por lo que se mantiene el valor de la iteración anterior.

$$\text{sgn}(x) = \begin{cases} +1 & x > 0 \\ -1 & x < 0 \end{cases} \quad (5)$$

Este proceso se repite indefinidamente hasta que el resultado converja hacia un valor concreto. Dicho de otra manera, mientras se cumpla la condición de la ecuación 6.

$$s_i(t+1) \neq s_i(t) \quad (6)$$

Una vez se llega a un valor constante, el proceso termina y el patrón debería haber sido reconocido. El método de Hopfield no asegura la convergencia hacia un patrón, sin embargo, esto si que está asegurado cuando los patrones enseñados en el periodo de aprendizaje son opuestos entre si.

III. RESULTADOS

Los resultados obtenidos para una ejecución concreta con las entradas x y x_p se muestran en la figura 4. Tal y como se puede apreciar en el resultado (matriz S), los vectores de la entrada X se aproximan al patrón almacenado más cercano en cada caso.

Title: Hopfield Network: Cube Aproximation
Subject: Data Mining
Author: Sergio García Prado (garciparedes.me)

x =
-1 -1 1
1 -1 -1
1 1 1
-1 1 1
-1 -1 -1
1 1 -1

x_p =
1 -1 1
-1 1 -1

w =
0 -2 2
-2 0 -2
2 -2 0

s =
1 -1 1
1 -1 1
1 -1 1
-1 1 -1
-1 1 -1
-1 1 -1

Figura 4: Resultados obtenidos