Inteligencia Artificial

Redes Neuronales

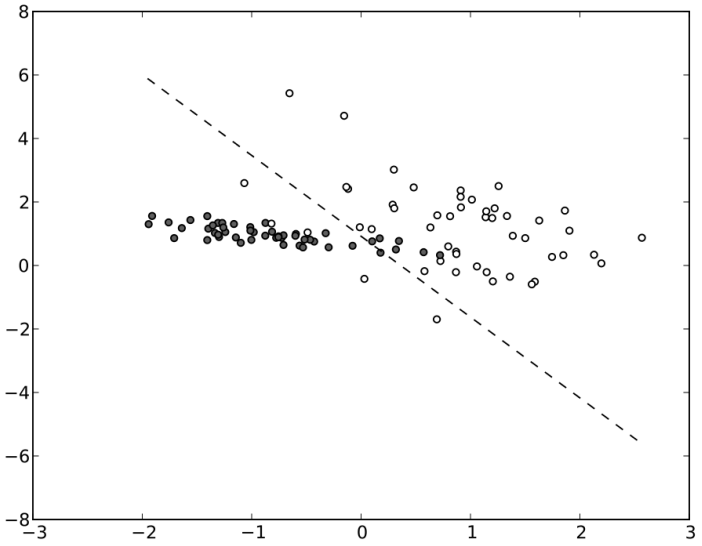
Perceptrones

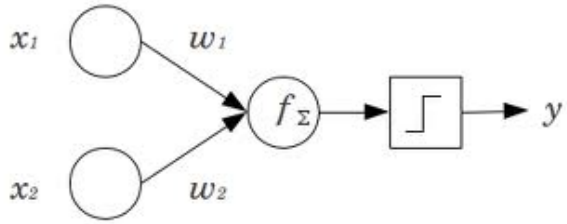
Dr. Alonso Ramirez Manzanares
Depto. de Matemáticas
Univ. de Guanajuato

e-mail: alram@cimat.mx
web: http://www.cimat.mx/~alram/info_apli2/

Entrenamiento supervisado, Idea :

Separar un conjunto de datos mediante entrenamiento





Idea :

Modelo matemático

$$w = [w_0, w_1, w_2]$$
$$x = [1, x_1, x_2]$$

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x > 0 \\ 0 & \text{otherwise} \end{cases}$$

Idea :

Notación

- el $x(j)$ denota el elemento en la posición j en el vector de la entrada
- el $w(j)$ el elemento en la posición j en el vector de peso
- el y denota la salida de la neurona
- el δ denota la salida esperada
- el α es una constante tal que $0 < \alpha < 1$

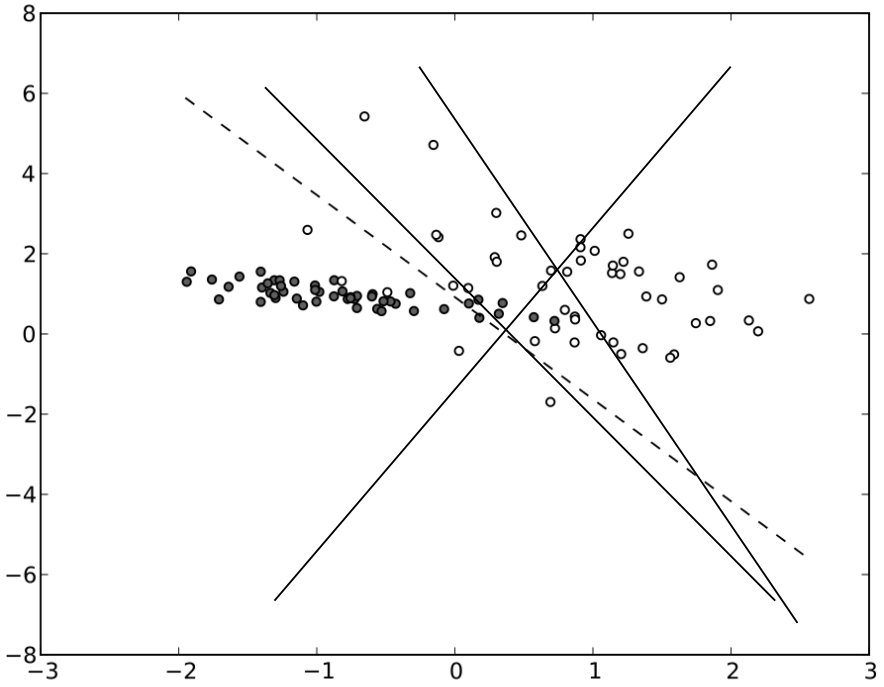
Entrenamiento

Inicializar los pesos w a cero.
En cada iteración el vector de peso es actualizado como sigue:

- Para cada pareja ordenada (x, y) en $D_m = \{(x_1, y_1), \ldots, (x_m, y_m)\}$
- Pasar (x_i, y_i, w_i) a la regla de actualización $w(j)' = w(j) + \alpha(\delta - y)x(j)$

Idea :

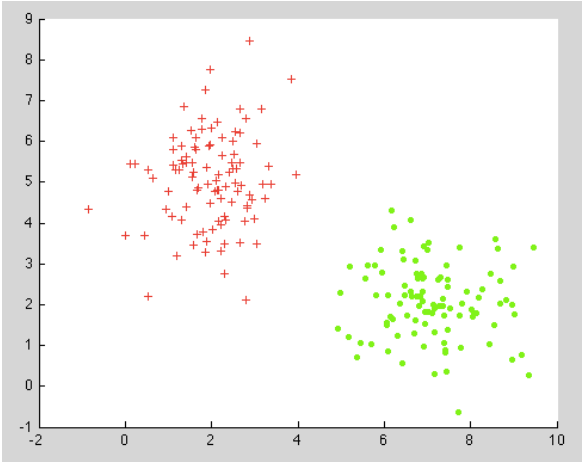
Lo que uno realmente hace es:



El algoritmo de entrenamiento

El algoritmo para entrenar un perceptrón de tal forma que separe los puntos es el siguiente:

Tenemos un conjunto entrenamiento de 200 puntos, cada punto en el plano, con coordenadas $p_i = (x_i, y_i)$, $i = 1,2,\ldots,200$, pertenece a la clase 0 (los rojos) o a la clase 1 (los verdes). La manera de indicar a que clase pertenece es con δ_i que puede ser 1 ó 0 para cada punto



El algoritmo de entrenamiento

Entonces tenemos que formar el conjunto de datos

$x_1, \quad y_1, \quad \delta_1$
$x_2, \quad y_2, \quad \delta_2$
…
$x_{200}, y_{200}, \delta_{200}$

```
function main_perceptron

    clc
    close all


    n_entrena = 100; % numero de datos de entrenamiento de cada clase
    n_predice = 100;  % numero de datos de prueba de cada clase

    sigma1 = 1; % dispersion
    media1 = [2 ;5]; % centro

    sigma2 = 1;  %dispersi?n 2
    media2 = [7 ;2]; % centro 2


    data = generaData(n_entrena,media1,sigma1,media2,sigma2);

    figure; hold on;

    plotClases(data,'r+','g.')
```

El algoritmo de entrenamiento

Y usamos las funciones

```
function data = generaData(n,media1,sigma1,media2,sigma2)

    xC1 = randn(n,1)*sigma1 + media1(1);
    yC1 = randn(n,1)*sigma1 + media1(2);
    output1 = zeros(n,1);  % esto es la clase 0

    xC2 = randn(n,1)*sigma2 + media2(1);
    yC2 = randn(n,1)*sigma2 + media2(2);
    output2 = ones(n,1);  % esto es la clase 1

    % pongo todo junto en una matriz
    data = [xC1 yC1 output1 ; xC2 yC2 output2];

    % lo desordeno
    idxs = randperm(n*2);
    data = data(idxs,:);


end
```

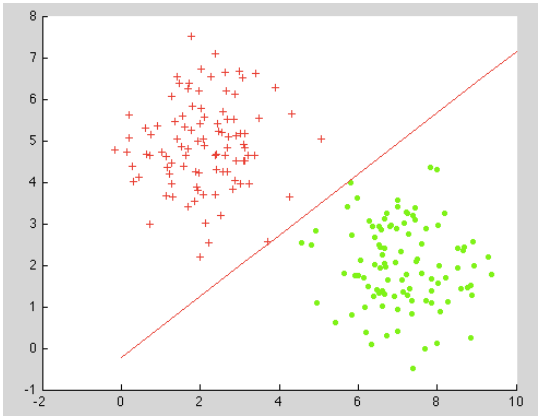
```
function plotClases(data,cad1,cad2)

    n = size(data,1);

    for i=1:n
        if data(i,3)==0
            plot(data(i,1),data(i,2),cad1);
        else
            plot(data(i,1),data(i,2),cad2);

        end
    end

end
```

Y con eso ya vemos y tenemos los datos

El algoritmo de entrenamiento

Bucamos la recta que separe los puntos por clase, esa recta es

$$w_0 + w_1 x + w_2 y = 0$$



Entonces tenemos que encontrar los pesos

$$w = [w_0, w_1, w_2]$$

El algoritmo de entrenamiento

Para encontrar los pesos $\boldsymbol{w}$ seguimos el siguiente algoritmo (mas detalles siguiente clase)

Tenemos los 200 puntos de entrenamiento

1) Iniciamos los pesos w = [0 , 0 , 0], α = 0.5

2) Para un número de iteraciones de entrenamiento M
3) Para cada uno de los 200 puntos p_1, p_2, …, p_i,… , p_{200}

4) Le preguntamos al perceptron como lo clasifica: $f_i = \begin{cases} 1 & \text{si } w^T[1\ x_i\ y_i] > 0 \\ 0 & \text{de lo contrario} \end{cases}$

5) Medimos el error $error_i = \delta_i - f_i$

6) Y calculamos nuevos pesos como:

$$w_0' = w_0 + \alpha * error_i$$
$$w_1' = w_1 + \alpha * error_i * x_i$$
$$w_2' = w_2 + \alpha * error_i * y_i$$

7) Actualizamos los pesos

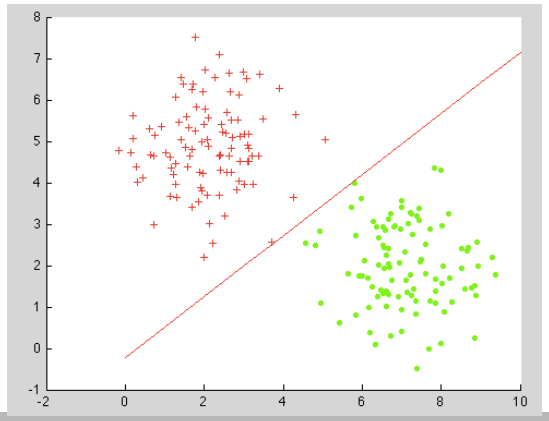
$$w = w'$$

El algoritmo de entrenamiento

Entonces , si al terminar una iteracion de entrenamiento (es decir, depués de haber probado para los 200 puntos) ya no hubo ningún error (el perceptrón clasifica bien todos), hemos terminado, de lo contrario hay que seguir dando iteraciones (hasta que se acaben las M iteraciones).

Al terminar, tenemos unos pesos $\boldsymbol{w}$ y para graficar la recta simplemente despejamos $\boldsymbol{y}$ de la ecuación $w_0 + w_1 x + w_2 y = 0$

y la mostramos en la grafica con plot

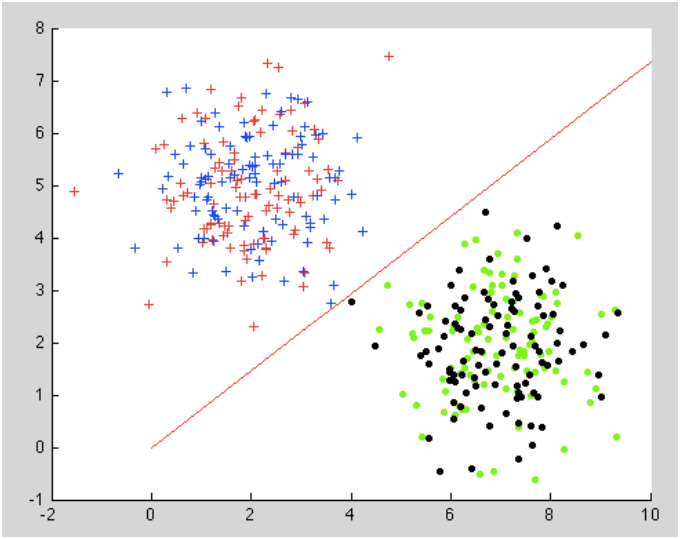


El algoritmo de entrenamiento

Mas aún, si ya que tenemos el $\boldsymbol{w}$ final, podemos generar otros 200 puntos de prueba (diferentes) usando de nuevo la función **`generaData`** y ver como los clasifica el perceptrón usando para c/u de ellos:

$$f_i = \begin{cases} 1 & \text{si } w^T[1\ x_i\ y_i] > 0 \\ 0 & \text{de lo contrario} \end{cases}$$

Lo cual genera una gráfica como:



Usando

```
dataprueba = generaData( ... );
    …
    Calcular la respuesta aquí
     del perceptron para cada punto
       y guardarla en
       la 3ª columna de
       dataprueba
    …
plotClases(dataprueba,'b+','k.')
```