PERCEPTRÓN SIMPLE EN SCILAB

Clasificación de puntos mediante el algoritmo de Perceptrón

Alumno: Rubén Horacio Machuca Santos

GENERACIÓN DEL CONJUNTO DE TRABAJO

- Se generan 20 puntos aleatorios dentro del cuadrado con coordenadas entre -1 y 1.
- x tiene dos filas: la primera fila son coordenadas X, y la segunda, coordenadas Y.

```
15 nct=20; //tamaño del conjunto de trabajo
16 x=2*rand(2,nct)-1;
17 x1=x(1,:);//Arreglo x1 contiene coordenadas x
18 y1=x(2,:);//Arreglo y1 contiene coordenadas y
19 plot(x1,y1,'*');
```

2. LÍNEA DE SEPARACIÓN

- Esto representa la ecuación de una línea: y = 2x, reescrita como x - 2y = 0.
- Esta línea divide el plano en dos clases.

```
21 //Graficamos una linea arbitraria y-2x=0 f(x): y=2x
22 //Salvamos los coheficientes de x y y en el arreglo F
23 F=[1;-2];
```

3. CLASIFICACIÓN INICIAL

- Calcula si el punto está arriba o abajo de la línea.
- Usa la función sign para asignar una clase: +1 o -1.
- Se colorean los puntos de verde o azul según la clase.

```
35 //Clasificamos los puntos a la derecha e izquierda de la linea
36 //los puntos tienen la misma x, solo hay que calcular la y del punto
37 //y la y de la recta y restar,
38 //clasificamos segun el resultado
39 for i=1:nct
40     l(i)=-F(2)*x1(i)-y1(i);//l(i)=F(2)*y1(i)+F(1)*x1(i);
41     class_F(i)=sign(l(i));
42 end
```

4. VISUALIZACIÓN DE CLASES

- class_F(i) tiene el valor +1 o -1, dependiendo de si el punto está de un lado u otro de la línea.
- Según su clase, el punto se dibuja en color verde o azul con un asterisco * como marcador.

```
39 for i=1:nct
40     l(i)=-F(2)*x1(i)-y1(i);//l(i)=F(2)*y1(i)+F(1)*x1(i);
41     class_F(i)=sign(l(i));
42 end
43
44 for i=1:nct
45         if class_F(i)==1 then
46             plot(x1(i),y1(i),'gre*');
47         else
48             plot(x1(i),y1(i),'blu*');
49         end
50 end
```

5. INICIO DEL ALGORITMO PERCEPTRÓN

- Se calcula la salida del perceptrón para cada punto.
- Si la salida no coincide con la clase esperada, se guarda el índice del error.

```
55 //Calculamos la salida del perceptron con los pesos
56 //actuales para todos los puntos g(x)=w1*x1+w2*y1
57 for i=1:nct
58     g(i)=sign(w(2)*y1(i)+w(1)*x1(i));
59 end
```

6. AJUSTE DE PESOS

- Solo se usa el primer error para ajustar los pesos.
- El perceptrón aprende corrigiendo el error.

```
65 while ~isempty(ind) //si es vacio => fin
66     //actualizamos los pesos w
67     //for i=1:nct
68     //operador .* es multiplicación de matrices elemento a elemento
69     w(1) = w(1) + (class_F(ind(1))-g(ind(1))).*x1(ind(1)); //
70     w(2) = w(2) + (class_F(ind(1))-g(ind(1))).*y1(ind(1)); //
71     //end
72     //temp = sign(w'*x);
```

7. RESULTADO FINAL

- Se grafica la línea de decisión aprendida por el perceptrón.

```
for i=1:100
    y3(i)=(-w(1)*x2(i))/w(2);
    //y3(i)=-(w(2)/w(1))*x2(i);
end
plot(x2,y3,'g');
mfprintf(6,'- - Al final los pesos son:\nvector w - - - -'); //escribe a consola
disp(w);
```

RESUMEN DEL PROCESO

- Se generan puntos aleatorios en 2D
- Se clasifican inicialmente con una recta
- El perceptrón ajusta pesos para clasificar correctamente
- Se muestra la frontera de decisión final