Vision Artificial. GIEC.

Sistemas de Vision Artificial. GIC.

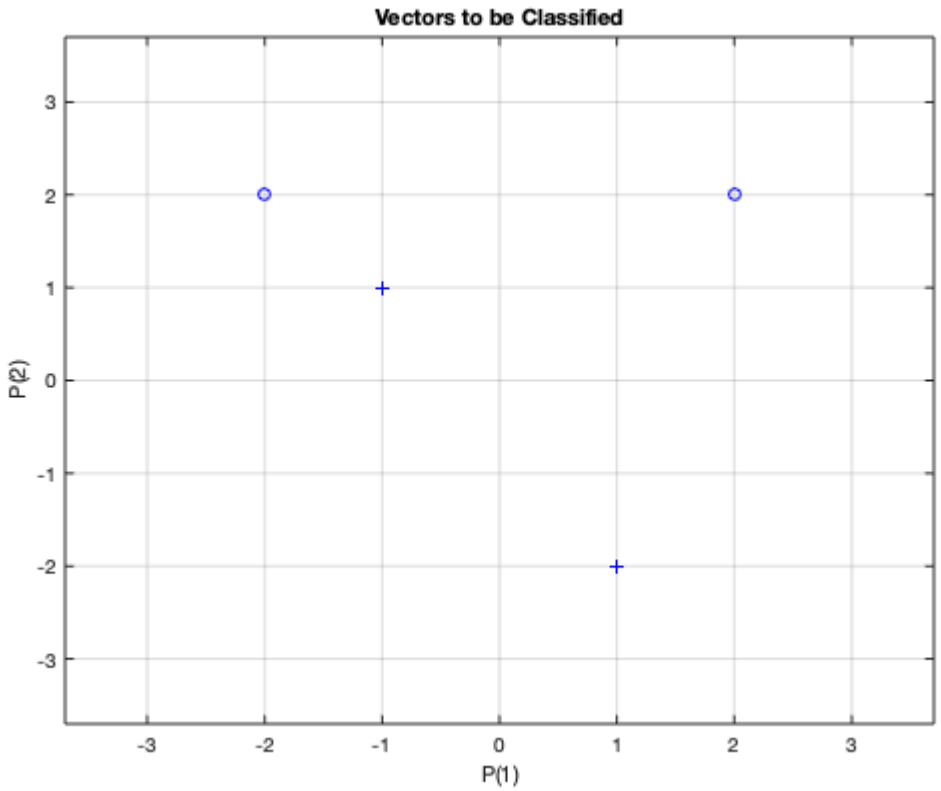
Miguel Angel Garcia, Juan Manuel Miguel, Sira Palazuelos.

Departamento de Electrónica. Universidad de Alcalá.

Tema 5: ejercicio 01 - Perceptrón

Step 1) Input and target vectors

```
clear all
close all
% Each of the four column vectors in X defines a 2-element input vectors
P = [2 1 -2 -1; 2 -2 2 1];
% The row vector T defines the vector's target categories.
T = [0 1 0 1];
% Plot input/target vectors
plotpv(P,T);
grid on;
```



Step 2) Perceptron

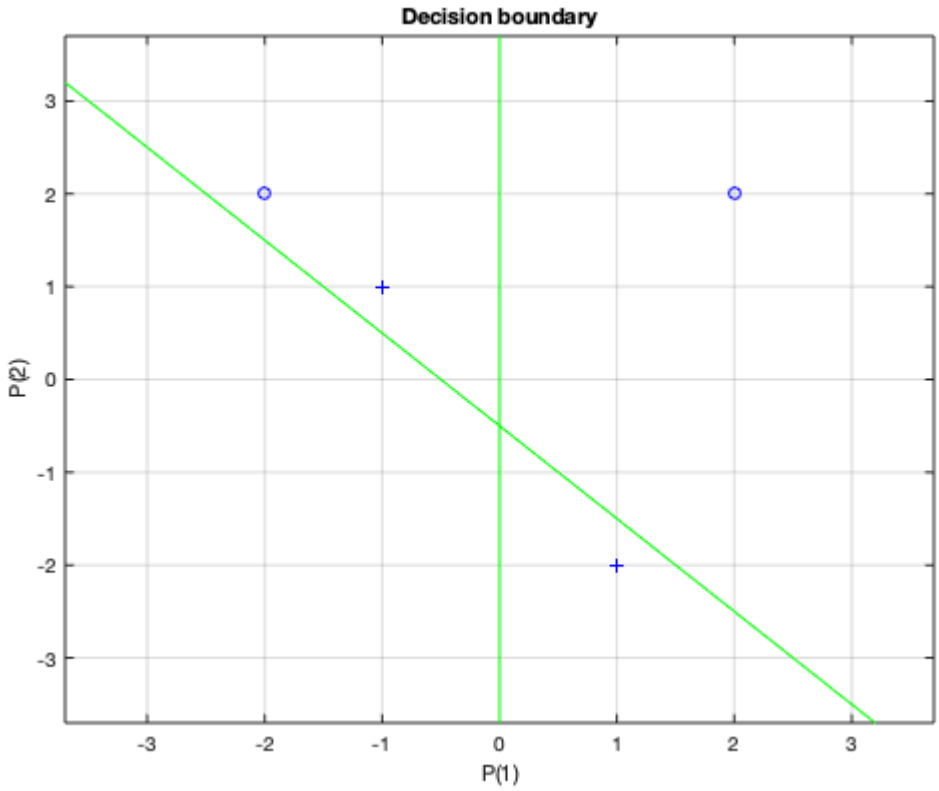
```
% PERCEPTRON crea una nueva red neuronal con una sola neurona
net = perceptron;
% CONFIGURE configura la red con los datos P y T (Normalmente el paso de configuración
net = configure(net,P,T);
```

Step 3) Training

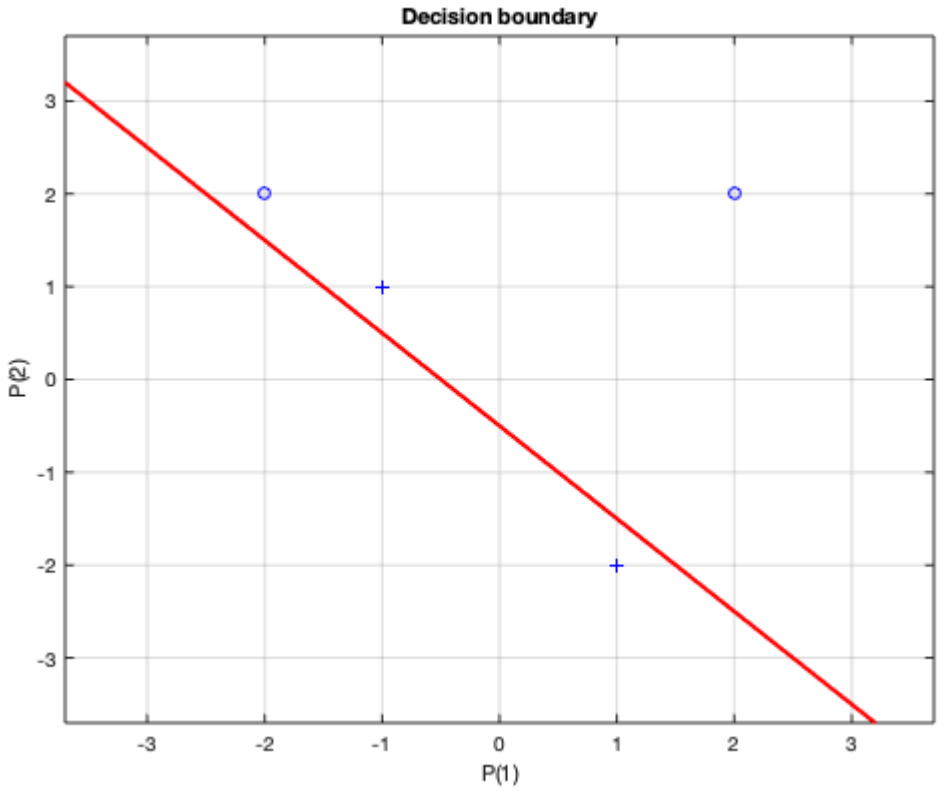
```
Weight(1,:) = net.IW{1};
Bias(1,:) = net.b{1};
Y = []; % valores de salida de la red
E = []; % errores de la red
plotpv(P,T);
grid on;
for n=1:length(P)

    [net,Y(n),E(n)] = adapt(net,P(:,n),T(:,n));
    Weight(n+1,:) = net.IW{1};
    Bias(n+1,:) = net.b{1};

    linehandle = plotpc(net.IW{1},net.b{1});
    set(linehandle, 'Color', 'g', 'LineWidth', 1);
    title('Decision boundary');
end
```




```
% Final plotting decision boundary
plotpv(P,T);
grid on;
linehandle = plotpc(net.IW{1},net.b{1});
set(linehandle, 'Color', 'r', 'LineWidth', 2);
title('Decision boundary');
```



Step 4) Simulation

Se pide

1. Ejecute el comando **`view(net)`** para ver un diagrama gráfico de la red neuronal. Identifique entradas, salidas, capas...
2. La función **`adapt`** (paso 3) actualiza la red para cada ciclo de entrenamiento y devuelve una nueva red que debería clasificar mejor. Compruebe la evolución de la matriz de pesos (**`IW{1}`**), y del vector de bias (**`net.b{1}`**) ¿por qué cambian los pesos y el bias?
3. Después de ejecutar el punto anterior. ¿Es correcta la frontera de decisión final? ¿Por qué?
4. Utilice el código 1 (CODE 1) para realizar un entrenamiento que llame a la función **`adapt`** hasta que la suma del error cuadrático (función **`sse`**) sea nula o el número de interaciones **`n`** llegue a 100. ¿se obtiene una fontera de decisión correcta?¿cuántas iteraciones han sido necesarias?
5. Utilice el código 2 para obtener nuevos valores de entrada aleatorios (**`P_sim`**). Realice la simulación, con la función **`sim`**, de estos nuevos puntos y compruebe que el perceptrón clasifica correctamente (paso 4).
6. Cambie los vectores de entrada (P) y de target (T) por estos valores: **`P =[-0.5 -0.5 +0.3 -0.1 -40; -0.5 +0.5 -0.5 +1.0 50]; T =[1 1 0 0 1];`**, que tienen una entrada atípica (outlier). Entrene una nueva red y compruebe cuántas iteraciones (épocas) son necesarias antes de obtener una frontera de decisión correcta.

7. Para reducir el número de iteraciones cuando hay entradas de valores atípicos, la solución es normalizar. La regla de percepción normalizada se implementa con la función **`learnpn`**. Cambie **`net=perceptron`** por **`net=perceptron('hardlim','learnpn');`** y comprobe si el número de iteraciones ahora es menor.
8. Finalmente, cambie los vectores de entrada (P) y de target (T) por estos valores: **`P =[ -0.5 -0.5 +0.3 -0.1 -0.8; -0.5 +0.5 -0.5 +1.0 +0.0 ]; T =[1 1 0 0 0];`**. Entrene una nueva red y compruebe que no se obtiene una frontera de decisión correcta. ¿Por qué la red alcanza el número máximo de iteraciones con un error distinto de cero?

```
% % CODE (1)
%
% E = 1; % errores de la red
% n=0;
% while (sse(E) > 0 & n < 200)
%    n = n+1;
%    [net, Y, E] = adapt(net,P,T);
% end
```

```
% % CODE (2)
%
% número de muestras de cada clase
% N = 20;
% % definir las entradas
% offset = 5; % offset para la segunda clase
% P_sim = [randn(2,N) randn(2,N)+offset]; % entradas
```