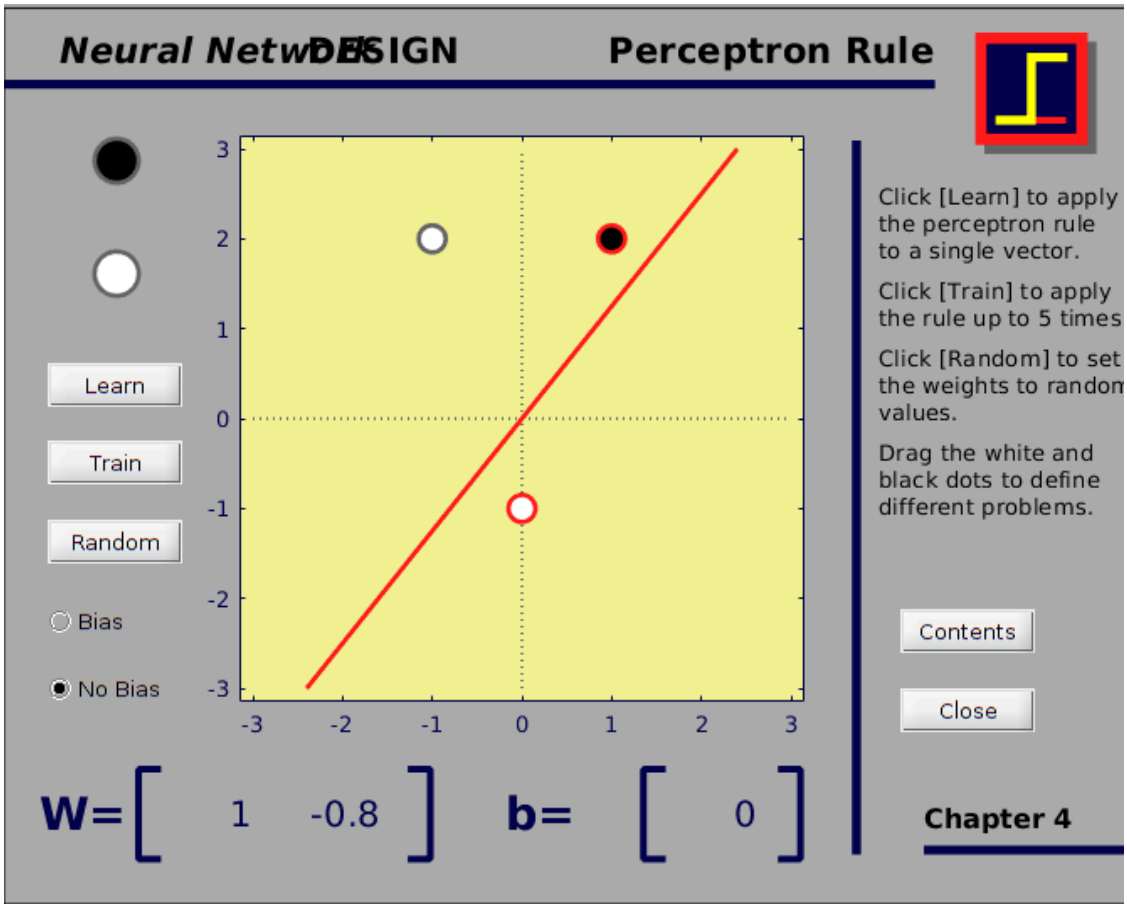
Table of Contents


```
% Universidad Nacional Autónoma de México
% Facultad de Ingeniería
% Aguilar Enriquez Paul Sebastian
% 415028130
% Temas Selectos de Sistemas Inteligentes - 2020-1
```

Ejercicio 0, Simuladores

```
nnd4db;
nnd4pr;
```



Ejercicio 1, Perceptrón de entradas de dos dimensiones

```
clear variables;
clear global;
close all;

% Creamos un nuevo perceptrón que tendra entradas en dos dimensiones,
% la primera dimensión tiene valores de 0 a 1 y la segunda de -2 a 2
net = newp([0 1; -2 2],1);

%  Los puntos de entrada, 4 de 2 dimensiones
P = [0 0 1 1; 0 1 0 1];

% Las etiquetas o 'targets' para cada punto
T = [0 1 1 1];

% Vemos los valores actuales asignados a la red dada la entrada P
Y = net(P)

% Asiganmos la cantidad de epocas para el entrenamiento, en este caso
 20
net.trainParam.epochs = 20;
```

```
% Entrenamos la red con los puntos P y las etiquetas T
net = train(net,P,T);

% Vemos los valores actuales de las salidas despues del entrenamiento
Y = net(P)
```

```
Y =

     1     1     1     1


Y =

     0     1     1     1
```

Ejercicio 2, Perceptrón de entradas de tres dimensiones

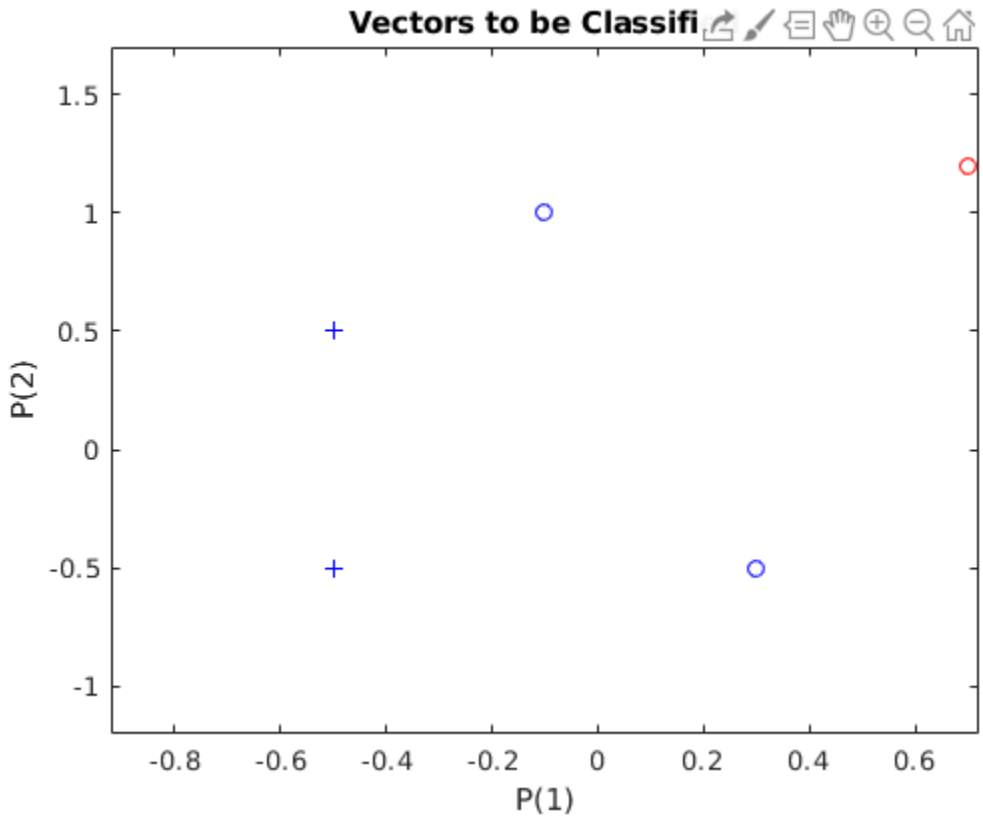
```
clear variables;
clear global;
close all;

% Creamos un nuevo perceptrón que tendra entradas en tres dimensiones,
% las tres dimensiones tienen valores de -1 a 1
net = newp([-1 1; -1 1; -1 1],[0 1]);

%  Los puntos de entrada, 2 de 3 dimensiones
P = [1 -1 -1; 1 1 -1];

% Las etiquetas o 'targets' para cada punto
T = [0 1];

% Pesos iniciales
W = initzero(1, [1 -1 -0.5]);

% Bias inicial
B = initzero(1, [1]);

% Vemos los valores actuales asignados a la red dada la entrada P
Y = net(P')

% Asiganmos la cantidad de epocas para el entrenamiento, en este caso
 10
net.trainParam.epochs = 10;

% Entrenamos la red con los puntos P y las etiquetas T
net = train(net,P',T);

% Vemos los valores actuales de salidas despues del entrenamiento
```

```
Y = net(P')

% Valores actuales del bias despues del entrenamiento
bias = net.b{1}

% Valores actuales de los pesos despues del entrenamiento
pesos = net. IW{1,1,1}

% Definimos dos entradas distintas para una simulación o 'test'
P2 = [1 0 -1; 1 0 -1];
P3 = [1 -1 0; 0 1 -1];

% Hacemos la simulación o 'test' con las entradas anteriores y la red
% entrenada
a = sim(net, P')
b = sim(net, P2')
c = sim(net, P3')


Y =

     1     1


Y =

     0     1


bias =

     0


pesos =

     0     2     0


a =

     0     1


b =

     1     1


c =

     0     1
```

Ejercicio 3, Graficación de un clasificador basado en un perceptrón

```
clear variables;
clear global;
close all;

%  Los puntos de entrada, 4 de 2 dimensiones
P = [ -0.5 -0.5 +0.3 -0.1; ...
      -0.5 +0.5 -0.5 +1.0];

% Las etiquetas o 'targets' para cada punto
T = [1 1 0 0];

% Graficamos los puntos P dadas sus etiquetas T
plotpv(P, T);

% Creamos un nuevo perceptrón que tendra entradas en dos dimensiones,
% ambas dimensiones van de -1 a 1
net = newp([-1 1; -1 1], 1);

% Graficamos los puntos P dadas sus etiquetas T
plotpv(P, T);

% Graficamos la recta del clasificar
plotpc(net.IW{1}, net.b{1});

% Asiganmos la cantidad de epocas para el entrenamiento, en este caso
 10
net.adaptParam.passes = 3;

% Adaptamos (entrenamos) la red neuronal con los valores de entrada
net = adapt(net, P, T);

% Gráficamos la recta del clasificador despues de entrenar
plotpc(net.IW{1}, net.b{1});

% Definimos un nuevo punto de entrada
p = [0.7; 1.2];

% Hacemos una prueba con la red entrenada y el punto definido
a = sim(net, p);

% Graficamos los puntos etiquetados con respectos a las salidas
plotpv(p, a);
point = findobj(gca, 'type', 'line');
set(point, 'Color', 'red');

% Graficamos la recta del clasificar
hold on;
plotpv(P, T);
plotpc(net.IW{1}, net.b{1});
```

```
hold off;
```



Ejercicio 4, feed-forward backpropagation network

```
clear variables;
clear global;
close all;

% Definimos los puntos de entrada
P = [1 2; 2 0; 4 2; 4 4; 6 4; 2 -3; 6 -2; -2 1; -4 0; -4 2; -4 -3; -6
 4;...
    1 3; 2 4; 4 6; 6 8; -2 5; -2 7; -4 4; -5 5;]
P2 = [1 2.5; 2 0.5; 4 2.5; 4 4.5; 6 4.5; 2 -3.5; 6 -2.5; -2 1.5; -4
 0.5; -4 2.5; -4 -3.5; -6 4.5;...
    1.5 3; 2.5 4; 4.5 6; 6.5 8; -2.5 5; -2.5 7; -4.5 4; -5.5 5;]

% Definimos las etiquetas para los puntos
T = [1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1;...
 0; 0; 0; 0; 0; 0; 0; 0;]

% Creamos una nueva red que tendra entradas en dos dimensiones,
% ambas dimensiones van de -8 a 8, las etiquetas de 0 a 1 y tendra dos
% capas
net = newff([-8 8; -8 8], [0 1], 2);
```

```
% Definimos las epocas para el entrenamiento
net.trainParam.epochs = 10000;

% Definimos el error para ele ntrenamiento
net.trainParam.goal = 0.0001;

% Definimos el learning rate para el aprendizaje
net.trainParam.lr = .5;

% Entrenamos la red con los puntos y las etiquetas previas
net = train(net, P', T');

% Vemos la salida de la red
out = net(P')

% Vemos el error actual en la red
errors = out - T'

% Vemos el performance de nuestra red dadas sus salidas y las
 etiquetas
% previas
perf = perform(net, out, T')

% Hacemos dos pruebas para dos conjutnos de puntos distintos
a = sim(net, P')
b = sim(net, P2')

% Graficamos los puntos y las rectas del clasificador
figure;
subplot(1, 2, 1);
plotpv(P', round(a));

point = findobj(gca, 'type', 'line');
set(point, 'Color', 'red');
hold on;
plotpc(net.IW{1}, net.b{1});
hold off;

subplot(1,2, 2);
plotpv(P2', round(b));

point = findobj(gca, 'type', 'line');
set(point, 'Color', 'red');
hold on;
plotpc(net.IW{1}, net.b{1});
hold off;


P =

     1     2
     2     0
     4     2
```

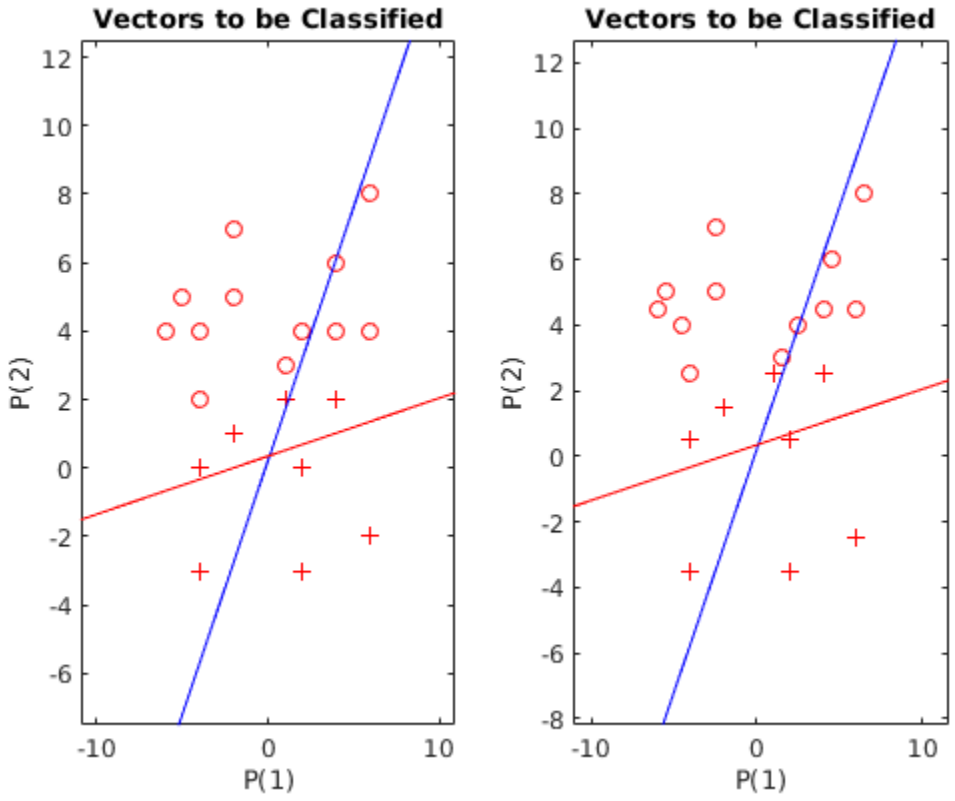
```
     4     4
     6     4
     2    -3
     6    -2
    -2     1
    -4     0
    -4     2
    -4    -3
    -6     4
     1     3
     2     4
     4     6
     6     8
    -2     5
    -2     7
    -4     4
    -5     5
```

```
P2 =

    1.0000    2.5000
    2.0000    0.5000
    4.0000    2.5000
    4.0000    4.5000
    6.0000    4.5000
    2.0000   -3.5000
    6.0000   -2.5000
   -2.0000    1.5000
   -4.0000    0.5000
   -4.0000    2.5000
   -4.0000   -3.5000
   -6.0000    4.5000
    1.5000    3.0000
    2.5000    4.0000
    4.5000    6.0000
    6.5000    8.0000
   -2.5000    5.0000
   -2.5000    7.0000
   -4.5000    4.0000
   -5.5000    5.0000
```

```
T =

     1
     1
     1
     1
     1
     1
     1
     1
     1
```

```
     1
     1
     1
     0
     0
     0
     0
     0
     0
     0
     0

out =

  Columns 1 through 7

    0.7593    1.0576    0.8790    0.3363    0.4383    1.1082    1.1074

  Columns 8 through 14

    0.8501    0.9469    0.4988    1.0815   -0.0245    0.4695    0.2352

  Columns 15 through 20

   -0.0288   -0.0970   -0.0539   -0.1296    0.0170   -0.0879

errors =

  Columns 1 through 7

   -0.2407    0.0576   -0.1210   -0.6637   -0.5617    0.1082    0.1074

  Columns 8 through 14

   -0.1499   -0.0531   -0.5012    0.0815   -1.0245    0.4695    0.2352

  Columns 15 through 20

   -0.0288   -0.0970   -0.0539   -0.1296    0.0170   -0.0879

perf =

    0.1251

a =

  Columns 1 through 7

    0.7593    1.0576    0.8790    0.3363    0.4383    1.1082    1.1074
```

```
  Columns 8 through 14

    0.8501    0.9469    0.4988    1.0815   -0.0245    0.4695    0.2352

  Columns 15 through 20

   -0.0288   -0.0970   -0.0539   -0.1296    0.0170   -0.0879

b =

  Columns 1 through 7

    0.6221    1.0256    0.7706    0.2050    0.2940    1.1094    1.1089

  Columns 8 through 14

    0.7395    0.8743    0.3480    1.0868   -0.0680    0.4991    0.2604

  Columns 15 through 20

   -0.0190   -0.0937   -0.0608   -0.1306    0.0056   -0.0922
```



Published with MATLAB® R2019a