EJERCICIO 1

CÓDIGO:

```
db_Inputs = xlsread('SI.xlsx','Entradas RNA')';

db_Targets = xlsread('SI.xlsx','Salidas RNA')';


%Arquitecturas

s1 = [4];

s2 = [8];

s3 = [16];

d1 = [4 4];

d2 = [4 8];

d3 = [8 4];


% Dos redes de una capa activa

rna1 = patternnet(s1);

rna1.trainParam.showWindow = false;

[rna1 tr1] = train(rna1, db_Inputs, db_Targets);

%resultados rna1

tr1.best_epoch

InputsTest1 = db_Inputs(:,tr1.testInd);

TargetsTest1 = db_Targets(:,tr1.testInd);

OutputsTest1 = sim(rna1, InputsTest1);

precisionTest1 = 1-confusion(TargetsTest1, OutputsTest1)


rna2 = patternnet(s2);

rna2.trainParam.showWindow = false;

[rna2 tr2] = train(rna2, db_Inputs, db_Targets);

%resultados rna2

tr2.best_epoch

InputsTest2 = db_Inputs(:,tr2.testInd);

TargetsTest2 = db_Targets(:,tr2.testInd);
```

```
OutputsTest2 = sim(rna2, InputsTest2);
precisionTest2 = 1-confusion(TargetsTest2, OutputsTest2)

% Dos redes de dos capas activas
rna3 = patternnet(d1);
rna3.trainParam.showWindow = false;
[rna3 tr3] = train(rna3, db_Inputs, db_Targets);
%resultados rna3
tr3.best_epoch
InputsTest3 = db_Inputs(:,tr3.testInd);
TargetsTest3 = db_Targets(:,tr3.testInd);
OutputsTest3 = sim(rna3, InputsTest3);
precisionTest3 = 1-confusion(TargetsTest3, OutputsTest3)

rna4 = patternnet(d2);
rna4.trainParam.showWindow = false;
[rna4 tr4] = train(rna4, db_Inputs, db_Targets);
%resultados rna4
tr4.best_epoch
InputsTest4 = db_Inputs(:,tr4.testInd);
TargetsTest4 = db_Targets(:,tr4.testInd);
OutputsTest4 = sim(rna4, InputsTest4);
precisionTest4 = 1-confusion(TargetsTest4, OutputsTest4)
```

¿Cuándo y por qué se para el entrenamiento?

El entrenamiento finaliza cuando transcurren 6 ciclos en los que no se no se mejora el error de validación. Por este motivo, cada vez que se hace un entrenamiento se queda el campo "validation checks" = 6.

Una vez que el entrenamiento se para, ¿cuáles son los resultados que se obtienen? ¿A qué ciclo corresponden? ¿Por qué son los de ese ciclo y no los de otro?

Para este código con esta configuración concreta los resultados obtenidos se miden en precisión, siendo de:

- *0.8936 para la arquitectura s1 ([4])*
- *0.9574 para la arquitectura s2 ([8])*
- *0.9255 para la arquitectura d1 ([4 4])*
- *0.9149 para la arquitectura d2 ([4 8])*

Estos resultados corresponden a los ciclos:

- *13 para la arquitectura s1 ([4])*
- *40 para la arquitectura s2 ([8])*
- *32 para la arquitectura d1 ([4 4])*
- *27 para la arquitectura d2 ([4 8])*

Los resultados que se obtienen son los de la iteración correspondiente a las iteraciones totales menos 6 (parámetro de número de ciclos sin mejora de validación), que se corresponde al valor "best_epoch" del "training record". El error de validación es quien determina qué red hay que devolver al finalizar el entrenamiento. Ej: Se realizaron 18 iteraciones en total, entonces en nuestro caso se devolvería la red de la iteración 12.

EJERCICIO 2

CÓDIGO:

```
% Se inicializan las bases de datos
db_Inputs = xlsread('SI.xlsx','Entradas RNA')';
db_Targets = xlsread('SI.xlsx','Salidas RNA')';

% Arquitecturas
s1 = [4];
s2 = [8];
s3 = [16];
s4 = [32];
s5 = [64];
d1 = [4 4];
d2 = [4 8];
d3 = [8 4];
d4 = [8 8];
d5 = [16 16];

arqsS = [s1; s2; s3; s4; s5];
arqsD = [d1; d2; d3; d4; d5];
mediasprecision = [];
desviacionestipicas = [];

% Entrenamiento con arquitecturas simples
for i=1:5,
precisionEntrenamiento = [];
precisionValidacion = [];
precisionTest = [];

for j=1:50,
rna = patternnet(arqsS(i,:));
```

```
rna.trainParam.showWindow = false;
[rna tr] = train(rna, db_Inputs, db_Targets);
Ouputs = sim(rna, db_Inputs);
precisionEntrenamiento(end+1) = 1-confusion(db_Targets(:,tr.trainInd),Ouputs(:,tr.trainInd));
precisionValidacion(end+1) = 1-confusion(db_Targets(:,tr.valInd),Ouputs(:,tr.valInd));
precisionTest(end+1) = 1-confusion(db_Targets(:,tr.testInd), Ouputs(:,tr.testInd));
end;

mediasprecision(end+1) = mean(precisionTest);
desviacionestipicas(end+1) = std(precisionTest);
end;

% Entrenamiento con arquitecturas de dos capas
for i=1:5,
precisionEntrenamiento = [];
precisionValidacion = [];
precisionTest = [];

for j=1:50,
rna = patternnet(arqsD(i,:));
rna.trainParam.showWindow = false;
[rna tr] = train(rna, db_Inputs, db_Targets);
Ouputs = sim(rna, db_Inputs);
precisionEntrenamiento(end+1) = 1-confusion(db_Targets(:,tr.trainInd),Ouputs(:,tr.trainInd));
precisionValidacion(end+1) = 1-confusion(db_Targets(:,tr.valInd),Ouputs(:,tr.valInd));
precisionTest(end+1) = 1-confusion(db_Targets(:,tr.testInd), Ouputs(:,tr.testInd));
end;

mediasprecision(end+1) = mean(precisionTest);
desviacionestipicas(end+1) = std(precisionTest);
end;
```

```
% Muestra de resultados
precision_desviaciontipica = [];

for i=1:10,
precision_desviaciontipica = [precision_desviaciontipica;mediasprecision(i), desviacionestipicas(i)];
end;
precision_desviaciontipica

% Gráfica
subplot(2,1,1)
plot(mediasprecision, '--g')
%axis([1 10 0 1])
title("Precision")
xlabel("Arquitectura (1-10)")
subplot(2,1,2)
plot(desviacionestipicas, 'r')
%axis([1 10 0 1])
title("Desviacion tipica")
xlabel("Arquitectura (1-10)")
```

Resultados:

Arquitectura	***Media***	***Desviación Típica***
s1 = [4]	*0.9228*	*0.0440*
s2 = [8]	*0.9502*	*0.0353*
s3 = [16]	*0.9500*	*0.0250*
s4 = [32]	*0.9468*	*0.0262*
s5 = [64]	*0.9472*	*0.0280*
d1 = [4 4]	*0.9251*	*0.0375*
d2 = [4 8]	*0.9087*	*0.0402*
d3 = [8 4]	*0.9460*	*0.0342*
d4 = [8 8]	*0.9489*	*0.0342*
d5 = [16 16]	*0.9511*	*0.0240*

¿Qué arquitectura es mejor para resolver este problema concreto?

La mejor arquitectura es d5 ([16 16]) puesto que tiene la mayor precisión de todas las arquitecturas empleadas y a su vez también tiene la menor desviación típica, lo que significa que la diferencia entre sus resultados es la menor de entre todas las arquitecturas. Analizando los datos, es posible que una arquitectura doble con un número aún mayor de neuronas funcionase mejor.