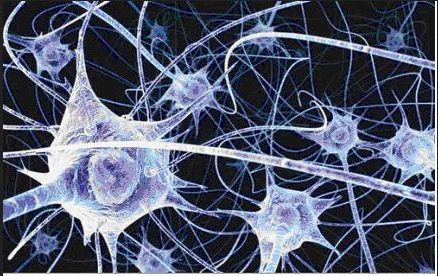
**PRACTICA 2: REDES NEURONALES**



Autor: Joaquin Vasalo Vicedo

Correo: [jvv7@alu.ua.es](mailto:jvv7@alu.ua.es), ximo.vasalo@gmail.com

DNI: 48675238Q

Grupo: lunes de 11:00-13:00

**INDICE**

1º Sesión 1

2º Sesión 2

3º Sesión 3

4º Sesión 4

5º Sesión 5,6,7

**Sesión 1:**

Contesta a las siguientes preguntas en tu memoria:

¿Cómo se puede acceder a una determinada imagen del conjunto de entrenamiento?

Con esta orden se devuelve el índice de una muestra. Si no existe el índice se devuelve una muestra aleatoria.

var mnist = require('mnist');

var set = mnist.set(8000, 2000);

var trainingSet = set.training;

var testSet = set.test;

Mnist[0].get(100) //[0,0,0,0,1,1… ,0,0]

mnist[0].get(100) // [0,0,0,1,1...,0,0]

Cuál es su tamaño?

28x28

Como se almacenan los píxeles de la imagen?

fimg = open(fname\_img, 'rb')

magic\_nr, size, rows, cols = struct.unpack(">IIII", fimg.read(16))

img = pyarray("B", fimg.read())

fimg.close()

Como se podría visualizar una imagen del conjunto de entrenamiento?

from pylab import \*

from numpy import \*

images, labels = load\_mnist('training', digits=[2])

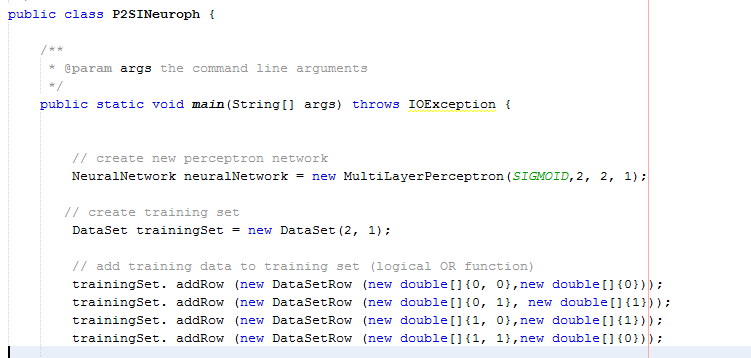
imshow(images.mean(axis=0), cmap=cm.gray)

show()

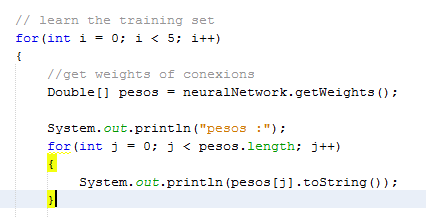
**Sesión 2**

Haz una red neuronal multicapa para calcular una XOR.

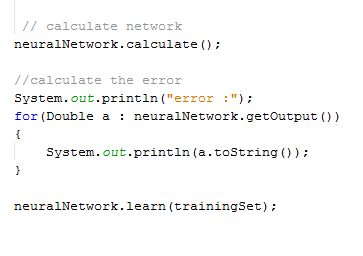
Planteamiento del training set, donde le damos las soluciones al programa



Imprimimos los pesos que tienen las conexiones entre neuronas



Calculamos el error cometido en cada iteración.



Los resultados obtenidos en 5 iteraciones son:

Iteración 1:

Pesos:

0.14487097997432316

-0.7605441736745082

0.6891215481667163

0.1819427880137239

0.0467389484731024

0.6003405229440776

0.926874923401772

0.01340978289016348

-0.7814877184523105

Error:

0.4611430914939685

Iteración 2:

Pesos:

-4.166508480751998

-4.158457155274505

6.064450026390104

4.606038482938229

4.610644540081937

-2.1420871296072215

5.174279740892435

5.305890706343384

-7.474523230051794

Error:

0.15651003091537297

Iteración 3:

Pesos:

-5.469712304311119

-5.4589695712907895

8.075103799730313

6.045968857219137

6.052097242000153

-2.8565513642425975

6.890099789354388

7.067545472905379

-9.959996136500369

Error:

0.07463019529525

Iteración 4:

Pesos:

-7.179410201567089

-7.165282676723566

10.754924348165884

7.935236895747098

7.943294991460816

-3.8065771430049278

9.176849230614074

9.414047333521935

-13.268413910396488

Error:

0.026393199353307573

Iteración 5:

Pesos:

-9.42308182980841

-9.404533785707677

14.3254574615665

10.414929790845457

10.425510939826534

-5.070956087261719

12.223578535739039

12.53975901005758

-17.67429842029587

Error:

0.00668934310030179

Resultados en forma de diagrama:

Esquema de errores de XOR

**Sesión 3**

En esta sesión tomaremos contacto con el programa modificando algunas variables para ver como se altera la capacidad de aprendizaje de la red neuronal y contestaremos a las siguientes preguntas:

¿Qué tamaño tiene que tener la capa de entrada de la red?

28\*28

¿Cuántas neuronas de salida debe de tener la red?

10

¿Cuántas neuronas en capa oculta?

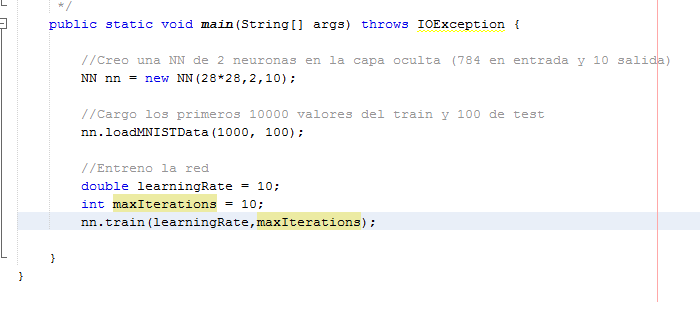
25 con 25 alcanzamos cuotas buenas de rendimiento

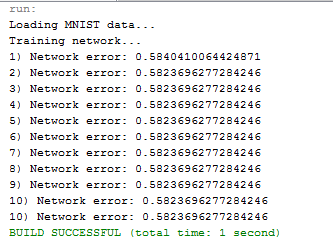
Con estos valores no nos da un valor razonable

2neuronas

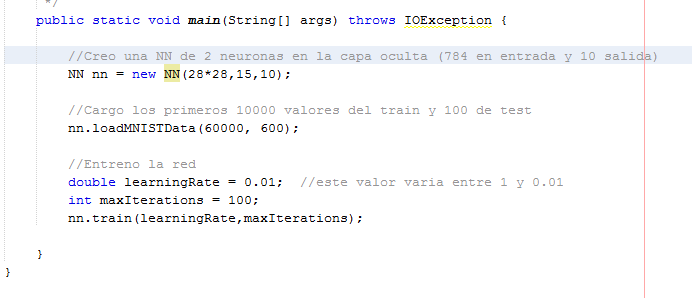
Learning rate = 10

maxIteration = 10

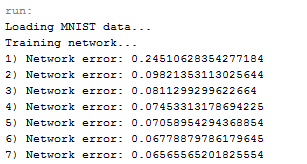




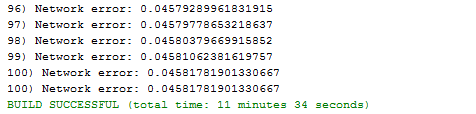
Con 15 capas intermedias learning rate de 0.01 y iteraciones 100



Nos da de 0.24 iniciales



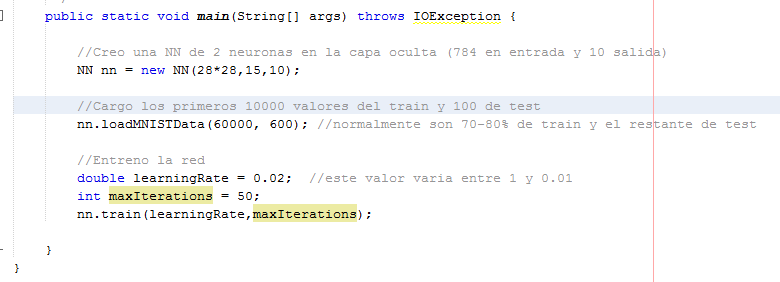
A 0.045 finales

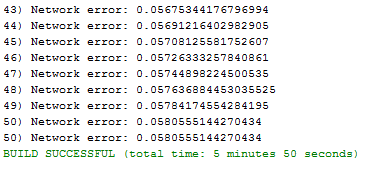


Con 15 capas intermedias learning rate de 0.02 y iteraciones 50

He reducido iteraciones porque el valor llega un punto en que empieza a oscilar

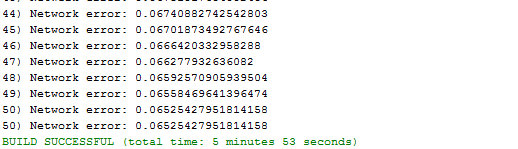
Los resultados son peores al anterior con 0.01 de learning rate.





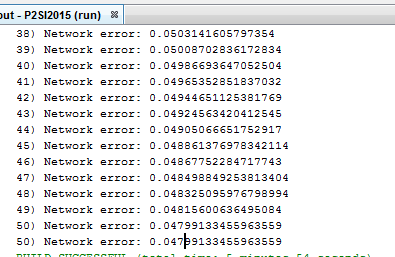
Ahora probaremos ajustando el learning a 0.001

Con ese learning sale mas error



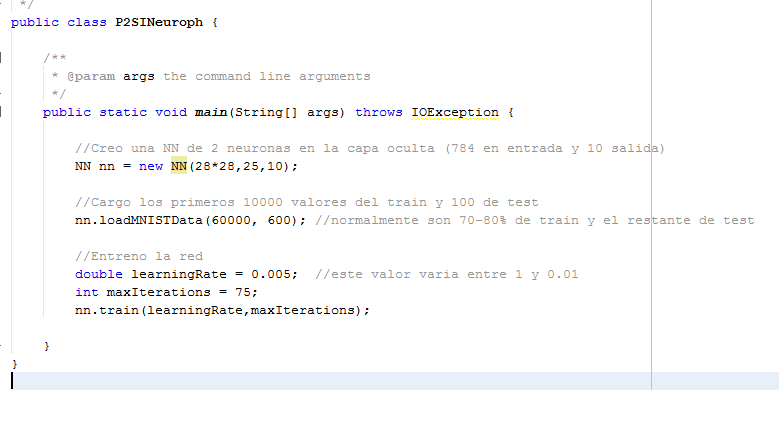
Vamos a probar con 0.005 de learning

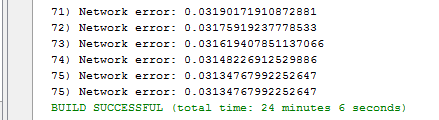
Con 0,005 va bien pero necesita mas iteraciones, no llega al valores de estanque en los que oscile.



Pondremos 25 neuronas , 0.005 de learning y unas 75 iteraciones.

Salen mejores resultados

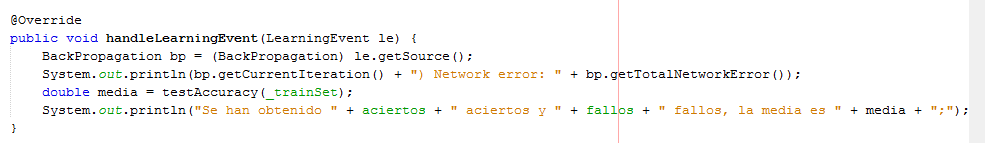




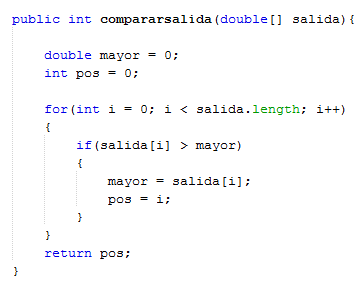
**Sesión 4**

Creamos las siguientes funciones en la zona del NN.java para poder mostrar el porcentaje de fallos y aciertos que comete nuestra red.

modificamos el handleLearingEvent anadiendo la media y un nuevo println para que muestre los fallos y aciertos obtenidos en otra funcion que mas adelante explicaremos.



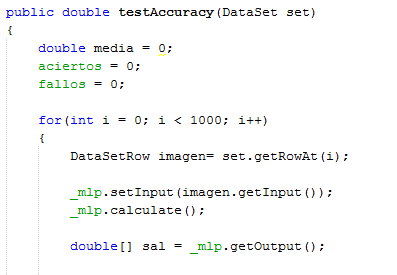
Con la sigiente funcion analizamos la lista de salida para comprobar si el resultado obtenido es valido o no es valido

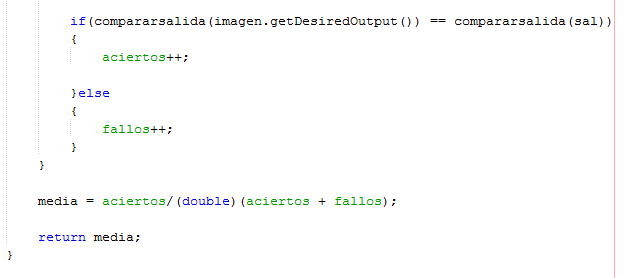


Este es la función testAccurary que hay que crear obligatoriamente si queremos mostrar lo aciertos y fallos de la red. aciertos y fallos son dos variables globales que irán modificando su valor en cada iteración mostrando los errores y aciertos cometidos y como avanzan son globales para que no se tengan que ir pasando de un lado a otro del programa y así evitar hacer trazas innecesarias en caso de error.

básicamente su funcionamiento se basa en coger la imagen y compararla con el resultado obtenido y el esperado y si difieren aumentar los errores y si son iguales entonces aumentar los aciertos, haciendo un compararsalida de las dos imágenes podremos averiguarlo.

Al final haremos una media para saber que porcentaje medio hay de errores respecto de fallos y con que grado de probabilidad puede fallar el programa.





**Sesión 5,6 y 7**

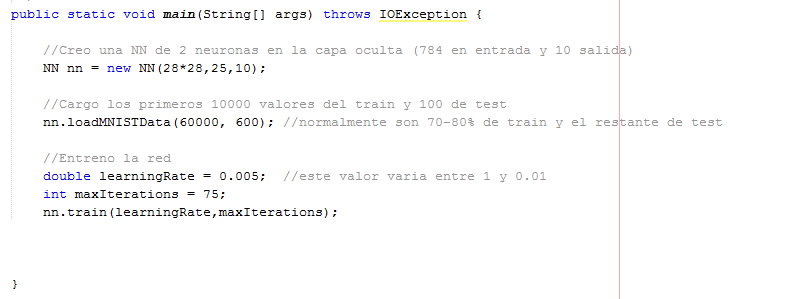
Ahora haremos pruebas para tener el valor mas bajo en nuestra red tal y como hicimos en la sesión 3 para familiarizarnos con las variables a modificar.

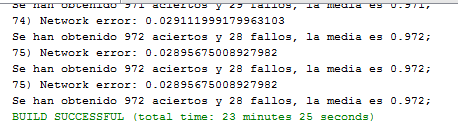
Con estos valores no nos da un valor razonablemente pequeño.

25 neuronas

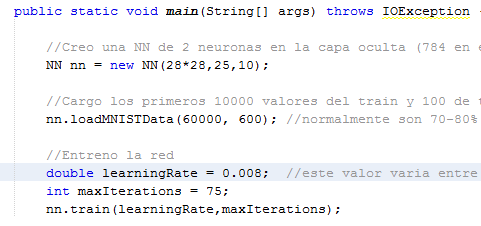
Learning rate = 10

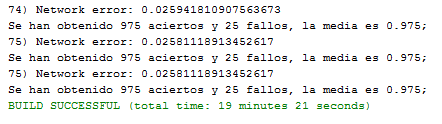
maxIteration = 75





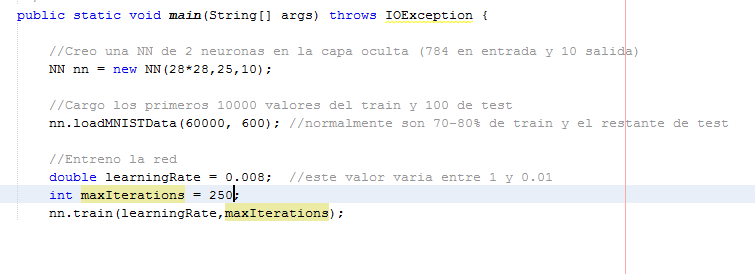
con 0.008 se obtiene 975 aciertos y 25 fallos



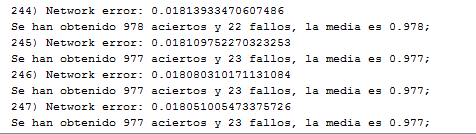


Con el 0,008 y 25 neuronas hemos obtenido los mejores resultados de todas las demás pruebas que al no ser mejores que la actual lo las he documentado ya que no aportan mas información.

Ahora aumentare el numero de iteraciones para que la red este por lo menos iterando 1h y pico para poder determinar que tasa de aprendizaje puedo alcanzar con 250 iteraciones.



como se puede observar a continuación a partir de la iteración 246 se vuelven a subir los errores



Estimo por tanto que estos datos son los mejores en la red con 978 aciertos y 22 fallos