SISTEMAS INTELIGENTES
Departamento de Ciencia de la Computación e Inteligencia Artificial
Universidad de Alicante **Curso 2015/2016**

Práctica 2. Redes Neuronales

Objetivos:

- Aprender las bases del algoritmo backpropagation
- Conocer una base de datos común para pruebas en redes neuronales
- Desarrollar una aplicación para clasificar dígitos manuscritos
- Analizar el efecto de los parámetros de la red en el proceso de entrenamiento

Enunciado.

En esta práctica se desarrollará una red neuronal que sea capaz de clasificar dígitos manuscritos. Para ello utilizaremos la conocida base de datos MNIST (http://yann.lecun.com/exdb/mnist/) que incluye 70000 dígitos manuscritos del *United States' National Institute of Standards and Technology*.

Cada elemento de esta base de datos contiene un dígito escaneado, que es una imagen de 28 x 28 píxeles y además una etiqueta con el valor real de dicho dígito. El conjunto está dividido en 60000 dígitos que se utilizarán para entrenar una red y 10000 para probar dicho entrenamiento (estos diez mil son de personas distintas al conjunto de entrenamiento). Se trata de una base de datos muy común para comprobar el funcionamiento de una red neuronal.

Librerías.

Para facilitar el desarrollo de la práctica se ha incluido en la plantilla proporcionada la librería Neuroph (http://neuroph.sourceforge.net/index.html) que será la que utilizaremos para el desarrollo de la misma. Utilizaremos solamente el *framework* de programación y por tanto no será necesario el uso de *neurophstudio*. De todas maneras el código y librerías necesarios ya están incluidas en la plantilla y no hace falta descargarse nada más. El API de Neuroph está disponible aquí http://neuroph.sourceforge.net/javadoc/index.html.

Sesión 1. Cargando MNIST. Empezando con el entorno

Aunque el formato de MNIST esta explicado convenientemente en http://yann.lecun.com/exdb/mnist/, la librería neuroph proporciona directamente métodos para cargar dichos conjuntos y facilitar su uso. Concretamente en la plantilla de la práctica, en la clase NN existe un método *loadMNISTData* que se encarga de cargar los primeros *n* y *m* valores del conjunto de entrenamiento y test.

Contesta a las siguientes preguntas en tu memoria:

¿Cómo se puede acceder a una determinada imagen del conjunto de entrenamiento? ¿Cuál es su tamaño? ¿Cómo se almacenan los píxeles de la imagen? ¿Cómo se podría visualizar una imagen del conjunto de entrenamiento?

Sesión 2. Introducción a backpropagation

En la memoria debes describir en pseudocódigo backpropagation para una función de activación sigmoidea y una función de coste cuadrática (serán las que utilizaremos en el resto de la práctica). Realízalo de manera muy detallada y añade un ejemplo donde se indique, paso por paso como mínimo para una iteración completa, como se ejecuta en una red sencilla. Por ejemplo utiliza una con la siguiente estructura *(input,hidden,output) = (2,2,1)* para el problema de aprender una puerta XOR. Muestra los *pesos* y *bias* iniciales y como varían conforme se ejecuta el algoritmo.

Puedes apoyarte en una implementación de dicha red si lo consideras necesario. Utiliza como base la propuesta que se hace en este documento http://neuroph.sourceforge.net/Getting%20Started%20with%20Neuroph%202.7.pdf

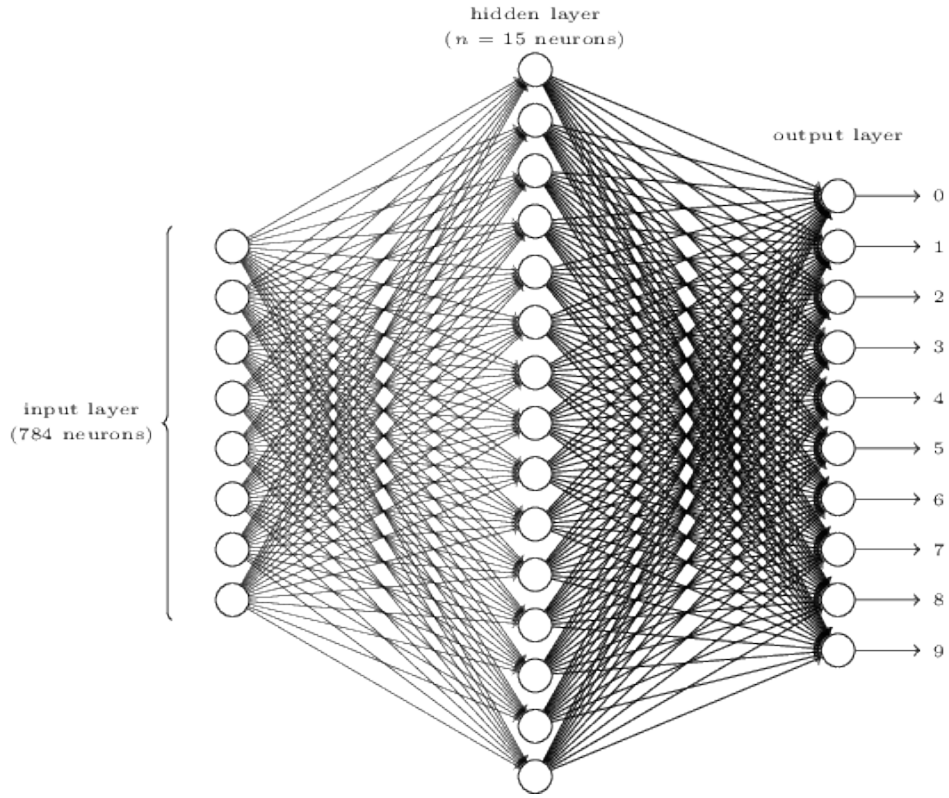
Sesión 3. Creando un perceptrón multicapa

Observa la clase NN donde se crea un perceptrón multicapa. Nos interesa clasificar imágenes de 28 x 28 píxeles, ¿qué tamaño tiene que tener la capa de entrada de la red? Solo pueden haber 10

clases posibles para cada dígito (0,1,2,3,4,5,6,7,8,9) ¿cuántas neuronas de salida debe tener la red? ¿Cuántas neuronas en la capa oculta? Para responder a todas estas preguntas deberemos realizar varias pruebas que determinen los mejores valores y opciones. Como guía podemos empezar a probar con una capa oculta de 15 neuronas y con una salida también de 10 (la neurona con mayor activación será la que determinará de que dígito se trata), pero eso no significa que estos valores sean los mejores.

Como el conjunto de entrenamiento y test de MNIST es muy grande los cálculos pueden ser lentos. Como recomendación general para el resto de la práctica empieza a realizar pruebas con pocos valores de dichos conjuntos. Cuando la red empiece a funcionar como se espera (cada iteración minimice el error de la anterior) entonces auméntalos hasta que incluyas todos los elementos. Recuerda que la función *loadMNISTData(int trainSamples, int testSamples)* te permite especificar cuantos datos de los conjuntos deseas cargar.

Si has diseñado una red con 10 neuronas en la capa de salida crea una función que determine cuál es la neurona con mayor activación para que sea sencillo verificar la salida de la red respecto a la etiqueta de una determinada imagen.



Sesión 4. Empezando las pruebas. Determinando la tasa de aprendizaje.

La tasa de aprendizaje es un parámetro fundamental para que el algoritmo de backpropagation funcione de manera correcta. Una métrica para determinar en cada iteración como aprende la red es ver el error (obtenido a partir de la función de coste cuadrática) de la red para todos los datos del conjunto de aprendizaje. Otra manera es determinar cuántos dígitos se clasifican correctamente. Crea una función llamada *testAccuracy(DataSet set)* que se encargue de calcular el porcentaje de dígitos bien clasificados para un conjunto de entrenamiento.

Empieza probando una tasa de aprendizaje muy pequeña (0.0001) y valora como aprende la red. Si aprende lentamente (no mejora su error o el número de imágenes bien clasificadas) increméntala un orden de magnitud y continua probando. Obtén el valor de la tasa de aprendizaje con la que la red aprenda más rápidamente.

Contesta a las siguientes preguntas en tu memoria: ¿Cuál es valor óptimo de aprendizaje para este problema? ¿Qué ocurre si utilizamos una tasa de aprendizaje menor? ¿Y una mayor? ¿influye el número de datos de entrenamiento en dicha tasa?

Sesiones 5,6 y 7. Experimentación para la clasificando de dígitos manuscritos.

Realiza todas las pruebas necesarias para determinar cuál es la mejor red y la mejor tasa de aprendizaje para clasificar todos los dígitos de MNIST. Entrena la red con el conjunto _train y analiza cómo evolucionan los porcentajes de clasificación para los conjuntos _train y _test en cada iteración. La meta es conseguir la red que mayor número de elementos **de todo el conjunto de test** clasifique correctamente.

Recuerda documentar convenientemente todas las pruebas que realices. En el *main* de la clase P2SINeuroph debes dejar configurados los mejores parámetros para el aprendizaje y de la red de clasificación de dígitos para todo el conjunto MNIST. Por la salida estándar se debe emitir para cada iteración el número de elementos bien clasificados del conjunto de test, por ejemplo:

```
Loading MNIST data...

> Clasificando (892 ok; 9108 error) 0.0892%

Training network...

1) Network error: 0.09765055123683983

> Clasificando (9120 ok; 880 error) 0.912%

2) Network error: 0.06494507544614854

> Clasificando (9166 ok; 834 error) 0.9166%

. . .
```

Parte optativa (30%).

- **Opción 1.** Descarga la plantilla de la parte optativa de la práctica. En esta se implementa un cargador de la clase MNIST así como una red neuronal (donde solo está implementada la fase forward). Completa dicha implementación con el algoritmo de backpropagation y prueba su funcionamiento con la base de datos MNIST.
- **Opción 2.** Realiza una implementación libre (sin la plantilla anterior) en el lenguaje java de una red neuronal con el algoritmo backpropagation explicado en clase. Prueba dicha red con el ejemplo que desees.
- **Opción 3.** Completa el entorno de la práctica. Muestra gráficamente las imágenes de MNIST así como permite introducir dígitos manuscritos a mano. Realiza un análisis teórico de cómo podrías mejorar tu tasa de clasificación de MNIST (puedes ver el ranking mundial y las técnicas utilizadas aquí http://yann.lecun.com/exdb/mnist/)

Formato de entrega

La fecha límite de entrega es el 21/12/2015. La entrega se realizará a través de Moodle. La entrega constará de un fichero .ZIP que contendrá dos carpetas:

- `'/src'` donde se encontrará la carpeta de proyecto de Netbeans.
- `'/doc'` donde estará disponible la documentación en formato PDF.

El nombre del fichero ZIP tendrá el siguiente formato: "`NombreApellido1Apellido2.ZIP`". Un fichero de ejemplo sería `RaulMartinezSerra.zip`

- ¡¡¡IMPORTANTE!!! no cumplir cualquiera de las normas de formato/entrega anteriores puede suponer un suspenso en la práctica. Recordad que las prácticas son individuales y NO se pueden hacer en parejas o grupos. Cualquier código copiado supondrá un suspenso de la práctica para todas las personas implicadas en la copia.
- ¡¡¡IMPORTANTE!!! La documentación en esta práctica es fundamental. Si no se justifican de manera empírica los resultados obtenidos la práctica se considerará suspensa.