

Práctica 3. Redes de Neuronas Artificiales (reconocimiento de dígitos) Inteligencia Artificial 2009/10

Enero-2010

Índice

| | |
|---|----------|
| 1. Descripción de la práctica | 1 |
| 1.1. Objetivos | 1 |
| 1.2. Tareas a realizar | 2 |
| 1.3. Normas de entrega | 2 |
| 2. Desarrollo de la práctica | 3 |
| 3. Recursos a utilizar | 4 |
| 3.1. Librería de simulación de perceptrones multicapa | 4 |
| 3.2. Datos de entrenamiento y evaluación | 5 |
| 3.3. Código de partida | 5 |

1. Descripción de la práctica

El objetivo de la práctica es construir y evaluar un sistema de reconocimiento de dígitos sencillo utilizando perceptrones multicapa (MLP). Para simular y entrenar los perceptrones multicapa se usará la librería Neuroph implementada en Java.

1.1. Objetivos

- Experimentar con herramientas para la simulación de RNAs.
- Evaluar el uso de perceptrones multicapa en una aplicación de procesamiento de imágenes sencilla

1.2. Tareas a realizar

El trabajo a realizar es eminentemente de experimentación para evaluar distintas configuraciones de los perceptrones multicapa y determinar su rendimiento en la tarea de reconocer dígitos. Las tareas concretas a realizar en la práctica serán las siguientes:

1. Diseño de las configuraciones a estudiar
2. Ejecución de las pruebas con las configuraciones seleccionadas
3. Análisis de los resultados

1.3. Normas de entrega

- La práctica se realizará de forma individual o en parejas.
- La fecha límite de entrega para la convocatoria de febrero será el **19/2/2010** en el despacho 303.
- Antes de esa fecha es posible entregar la práctica en horario de prácticas o de tutorías.
- Se deberá entregar:
 - Memoria en papel con la estructura que se indica
 - Si se considera necesario, se podrá acompañar del código fuente empleado para desarrollarla.
- En el momento de la entrega se comprobará el funcionamiento de la práctica y el modo en que fue desarrollada, por lo que, salvo causas de fuerza mayor, será necesaria la presencia de todos los miembros del grupo.
- Estructura de la memoria (aprox. 5-7 págs):
 - Descripción del problema (herramientas utilizadas, datos empleados y modo en que se ha hecho, etc)
 - Descripción de la implementación o modificaciones realizadas (si procede)
 - Descripción de los experimentos realizados y resultados obtenidos
 - Conclusiones y problemas encontrados (resultados, posibles mejoras, etc)
 - Bibliografía consultada

Nota: Incluir Nombre, DNI y e-mail en la portada

2. Desarrollo de la práctica

La practica consiste en aplicar perceptrones multicapa [MLP] a un problema de reconocimiento de dígitos (de 0 a 9).

En nuestro caso, para el entrenamieto se partirá de un conjunto de muestras de dígitos de 0 a 9 en forma de imágenes en formato PGM.

Los datos originales fueron tomados de la colección Optical Recognition of Hand-written Digits Data Set del repositorio UCI.

Se cuenta con dos versiones de este conjunto de datos:

- **Imágenes originales:** imágenes de 32x32 pixels en blanco y negro
- **Imágenes reducidas:** imágenes de 16x16 pixels en escala de grises con profundidad 4 (blanco + 4 niveles de gris) generadas a partir de las anteriores.
Para generarlas se tomaron 256 regiones de 2x2 pixels de cada imagen original sin solapamiento y se contó el número de pixels de color negro en cada región para establecer su nivel de gris.

El nombre de cada fichero identifica el dígito que representa la imagen que contiene.

Para este ejercicio se empleará el conjunto de imágenes reducidas. El objetivo de la red a entrenar será identificar el dígito contenido en cada imagen de 16x16 pixels.

- Las imágenes están en formato PGM con una profundidad de 4 niveles de gris (1 byte/pixel: 0=negro, 4=blanco).
 - Se incluye una clase Java (`FicheroPGM`) que carga estas imágenes en una matriz de enteros de dimensión 16x16.
- Para alimentar los perceptrones estas imágenes se convierten en un array de 256 elementos de tipo *double* con valores entre 0 y 1 (0=blanco, 1=negro).
 - Los componentes de cada vector se normalizan dividiéndolos por el valor más alto para garantizar un rango de valores uniforme.
 - Se incluye una clase Java (`Util`) que implementa estas transformaciones.
- El perceptrón multicapa a utilizar tendrá 256 neuronas de entradas, una o más capas ocultas y 10 neuronas de salida (una para cada posible dígito).
 - Se intepretará que un valor igual a 1 (o cercano) en la neurona K de la capa de salida predice que la imagen contiene el dígito K .

Para el entrenamiento y validación se utilizará el conjunto de datos MIT-CBCL. Se proporciona una versión completa y otra reducida con un menor número de imágenes.

El objetivo de la práctica es el de realizar una serie de ciclos entrenamiento+evaluación de forma ordenada que permitan llegar a conclusiones sobre el uso de MLPs en este tipo de tareas. Se entrenarán distintos MLPs, con diferentes topologías, para tratar de identificar configuraciones con buenos resultados. Una vez entrenado cada perceptrón con el conjunto de entrenamiento se realizará una pequeña evaluación de los resultados que se obtienen al aplicar esta herramienta sobre el conjunto de test que se proporciona.

Se proporciona código de partida con ejemplos de uso del API *Neuroph* cómo se crean las redes, como se entrenan y una implementación de un posible mecanismo de evaluación.

- En el código de ejemplo se evalúan dos aspectos relativos al comportamiento de la red entreda con los datos de validación:
 - Se calcula el error absoluto medio para el conjunto de validación considerado, calculando la media de las diferencias entre las salidas obtenidas por la red y las salidas deseadas. (*medida de error genérica*)
 - Se calcula el porcentaje de aciertos analizando los 10 valores de salida de cada ejemplo para determinar el dígito predicho. (*medida de rendimiento en la tarea de identificar dígitos*)
 - Dado que se trata de 10 neuronas de salida, con valores continuos en el rango $[0,1]$, representando cada una un dígito, se considerará como dígito predicho aquel cuya respectiva neurona de salida tome el valor más alto.

Dado que en los experimentos a realizar se tratará básicamente de variar la topología de la red, los parámetros a evaluar serían:

- Número de capas ocultas y número de neuronas en cada capa oculta (basta con probar redes de una y de dos capas ocultas)
- El número de neuronas de entrada será siempre 256 (se usan las imágenes reducidas)
- El número de neuronas de salida será siempre 10
- Número de pasadas (*epoch*) en la fase de aprendizaje

3. Recursos a utilizar

3.1. Librería de simulación de perceptrones multicapa

En la práctica se usará la librería de simulación de Redes Neuronales *Neuroph*

- Página del proyecto [descargas, Javadoc, ejemplos, ...]

3.2. Datos de entrenamiento y evaluación

Para el entrenamiento y validación se utilizará el conjunto de datos Optical Recognition of Handwritten Digits Data Set del repositorio UCI.

Se proporciona una versión completa y otra reducida con un menor número de imágenes. En ambos casos hay un directorio *train* con las imágenes a usar en el entrenamiento y un directorio *test* con las imágenes de validación

- Dataset original: original.tar.gz.
 - Imágenes de 32x32 pixels con 1 bit de profundidad
 - 3.823 imágenes de entrenamiento (subdirectorio **train**) y 1.797 de evaluación (subdirectorio **test**)
- Dataset reducido: reducido.tar.gz.
- Imágenes de 16x16 pixels con 4 niveles de gris
- 3.823 imágenes de entrenamiento (subdirectorio **train**) y 1.797 de evaluación (subdirectorio **test**)

En ambos casos el nombre de los ficheros que contienen las imágenes que se correspondan con el dígito que almacena (**cero**, **uno**, ... **nueve**).

3.3. Código de partida

Se incluye un ejemplo en Java para mostrar el uso del API *Neuroph* listo para ser usado como base para lanzar los experimentos. También se incluye código adicional para la carga y procesamiento de las imágenes en formato PGM.

Descarga (proyecto Netbeans): Ejemplodigitos.tar.gz

Compilación y uso del ejemplo de entrenamiento desde línea de comandos

```
$ tar xzvf EjemploDigitos.tar.gz
$ cd EjemploDigitos
$ cd src
$ javac -cp ../lib/neuroph.jar entrenamiento/EjemploEntrenamiento.java
$ java -cp ../lib/neuroph.jar entrenamiento.EjemploEntrenamiento "256 50 10 10" 50 prueba
```

El programa para entrenar el MLP recibe 5 parámetros

1. Especificación de la red.
 - Cadena de números enteros entre comillas.
 - Cada entero representa el número de neuronas en cada capa de la red: entrada, oculta/s y salida (en ejemplo se crea una red con 256 neuronas de entrada, 2 capas ocultas de 50 y 10 neuronas y 10 neuronas de salida)
2. Número de iteraciones (epoch) sobre el conjunto de entrenamiento

3. Identificador de la red (se usará para nombrar el archivo .net donde se almacenará la red entrenada)
4. Directorio con las imágenes de entrenamiento.
5. Directorio con las imágenes de evaluación.