Máster Universitario en
Lógica, Computación e Inteligencia Artificial
Aprendizaje Automático

Apellidos: .....

Nombre : .....

En esta primera parte vamos a hacer uso de las redes neuronales para la clasificación de datos de cáncer de mama. Para ello, vamos a utilizar Neuroph, que es un entorno escrito en Java para el desarrollo de redes neuronales. Neuroph puede encontrarse en `http://neuroph.sourceforge.net/` y está disponible para Windows y Linux. Para instalar Neuroph necesitas Java 1.8 (o superior). La forma más sencilla de instalarlo es descargar el instalador `neurophstudio-windows-xxx.exe` (para Windows) o `neurophstudio-linux-xxx.sh` y eguir las instrucciones, donde `xxx` es la versión correspodiente. Para instalarlo en linux basta abrir un terminal y, en el directorio donde está el instalador, teclear `sh neurophstudio-linux-xxx.sh`. Una vez instalado, basta buscarlo entre las aplicaciones y lanzarlo.

En primer lugar, seguiremos el tutorial de la profesora Jovana Trisic, de la Facultad de Ciencias de la Universidad de Belgrado. El tutorial está disponible aquí y es uno de los tutoriales recomendados por los creadores de Neuroph para empezar a manejar la herramienta.

Objetivo

El objetivo es crear y entrenar una red neuronal para predecir si un cáncer de mama tiene carácter maligno o benigno. Para ello usaremos como conjunto de entrenamiento una base de datos obtenida por el Dr. William H. Wolberg en los hospitales de la Universidad de Wisconsin en 1991. El nombre de la base de datos es *Wisconsin Breast Cancer Database* y se puede obtener en este enlace. Si tienes problemas para la descarga, junto a este fichero también debes tener una copia del fichero de datos.

Entra en *Data Folder* y descárgate el fichero `breast-cancer-wisconsin.data`. Este fichero puede abrirse con una hoja de cálculo (por ejemplo Excel de MS Office en Windows o LibreOffice Calc en Linux) indicando que la separación se realiza mediante comas. El fichero contiene datos de 699 muestras. La primera columna es el identificador de la muestra. La última columna es la clase, que toma valores 2 y 4 (2 representa benigno y 4 representa maligno). Las otras nueve columnas tienen los valores de los siguientes atributos

```
Clump Thickness
Uniformity of Cell Size
Uniformity of Cell Shape
Marginal Adhesion
Single Epithelial Cell Size
Bare Nuclei
Bland Chromatin
Normal Nucleoli
Mitoses
```

Cada atributo toma valores entre 1 y 10.

Preprocesado

El fichero `breast-cancer-wisconsin.data` necesita ser preprocesado para ser tratado con Neuroph. Realizaremos el siguiente tratamiento del fichero:

1. Eliminamos la primera columna, que corresponde a los identificadores.
2. Normalizamos los valores de las nueve columnas siguientes, de manera que tengan valores entre 0 y 1.
3. Sustituimos la última columna por otras dos, de manera que la clasificación `benigno` que ahora se denota con un 2, se represente mediante el par de valores $(1, 0)$ y la clasificación `maligno` se sustituya por el par $(0, 1)$.
4. Eliminamos aquellas instancias en las que faltan valores.

El fichero `breast-cancer-wisconsin.data` contiene los datos originales y `datos-preprocesados-bcw.txt` contiene los datos preprocesados (separados por tabulador). El preprocesado incluye otros aspectos, como el tratamiento de los datos en los que falta algún valor del atributo (en este caso han sido eliminadas, pero no siempre es la mejor opción), el tratamiento de valores discretos para los que no exista un orden natural o la combinación de bases de datos procedentes de civersas fuentes. Obviamente, el preprocesado del fichero dependerá del formato en que nos llegue.

Un proyecto Neurph

En la ventana principal de Neuroph, en *File*, pulsa en *New project* para crear un nuevo proyecto. Elegimos la categoría *Neuroph*. En la ventana siguiente le asignamos un nombre, por ejemplo, *Estudio de cáncer de mama*.

Vemos que cada proyecto tiene asociada una red, un conjunto de entrenamiento y un conjunto de prueba. A continuación, determinamos el conjunto de entrenamiento. Pulsamos con el botón secundario en *Training Set* y luego en *New* y en *Other*. Como tipo de fichero, elegimos *Data Set*. Llamamos al conjunto de entrenamiento *Wisconsin Data Set* e indicamos que haremos aprendizaje supervisado. Consideramos 9 neuronas en la capa de entrada y 2 neuronas en la capa de salida. A continuación cargamos el fichero `datos-preprocesados-bcw.txt` usando el tabulador como delimitador. No olvides ir al final de la tabla, pulsar OK y grabar el conjunto de entrenamiento que acabas de crear.

A continuación, vamos a crear la red neuronal. Dentro del proyecto que estamos creando vamos a *Neural Network* → *New* → *Other* → *File Type: Neural Network* → *Multilayer Perceptron*. Crearemos una red con 9 neuronas en la capa de entrada, 1 capa oculta con 1 neurona y 2 neuronas en la capa de salida. Usaremos la función sigmoide con neuronas de sesgo *(bias)* y realizaremos aprendizaje mediante retropropagación con *momentum*.

Una vez creada la red, arrastramos el fichero de aprendizaje hasta la capa de entrada. Vemos que se habilitan los botones superiores. Pulsamos en *Train* para inicial el entrenamiento. Fijamos el error máximo en 0.03, el factor de aprendizaje en 0.4 y el *momentum* en 0.4.

Una vez realizado el aprendizaje, en la pestaña correspondiente a la red, pulsamos en *Test*. Ahí vemos los errores cometidos en cada ejemplo. Vemos los valores obtenidos *(Output)* y los que aparecen en le conjunto de entrenamiento *(Desired output)* para cada ejemplo. Al final aparece el error medio cuadrático.

Vemos que aunque se ha obtenido un error cuadrático por debajo del umbral 0.03 (se ha obtenido 0.023801718131087373), hay ejemplos puntuales para los que hay errores muy grandes (Ver, por ejemplo, los errores en las instancias 245 o 252 en el fichero `Error_1.ods`).

Una posible mejora sería volver a entrenar la red con un error máximo de 0.01, pero vemos que en este caso no se produce convergencia. Puedes poner un número máximo de iteraciones para evitar que el sistema siga indefinidamente, p.e., 2000 iteraciones. De todos modos tienes a tu disposición el botón `Stop` en la parte superior de la ventana.

Un segundo intento

Vamos a construir una segunda red. En este caso pondremos una capa oculta con 8 neuronas. Para el aprendizaje, tomaremos los valores por defecto, i.e., *Error máximo=0.01*, *Tasa de aprendizaje=0.2* y *Momentum=0.7*. La tabla de errores la puedes ver en el fichero `Error_2.ods`. ¿Se ha producido una mejora?

Puedes probar con varias capas intermedias, cambiando el número de neuronas en esas capas. También puedes cambiar el factor de aprendizaje y el *momentum*. ¿Cuáles son las mejores decisiones?

Ejercicio 1. En http://repository.seasr.org/Datasets/UCI/arff/ podemos encontrar un conjunto de bases de datos pública en formato `arff`. Es un repositorio de bases de datos de la Universidad de California en Irvine. Elige uno de ellos y úsalo para estudiar cómo se pueden usar las redes neuronales como herramienta de clasificación.

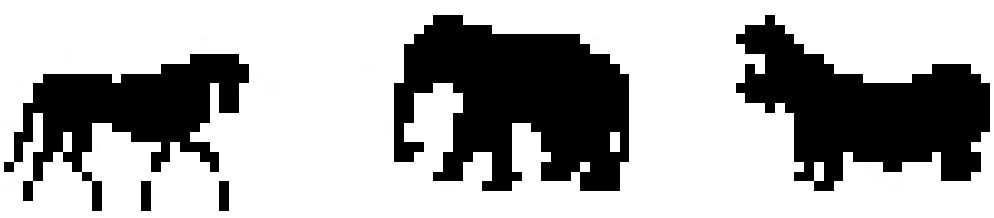
Como sugerencia, puedes tomar el fichero `arrythmia.arff` (también lo tienes junto al enunciado de esta práctica). Contiene un conjunto de 452 ejemplos y 279 atributos. El objetivo es aprender el tipo de arritmia a partir de los datos de electrocardiogramas. Esta base de datos se usó en el artículo de investigación

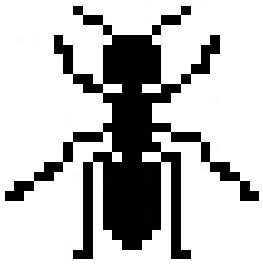
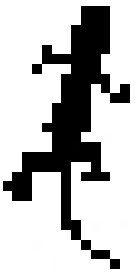
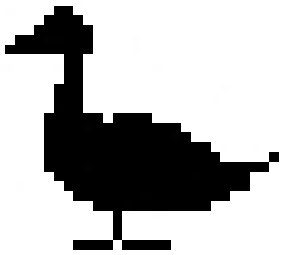
http://www.cs.bilkent.edu.tr/tech-reports/1998/BU-CEIS-9802.pdf

---

Animales

El objetivo de esta segunda parte es entrenar una red neuronal con seis imágenes para las que se conoce su clasificación y, una vez entrenada, pedir que clasifique imágenes nuevas. El conjunto de entrenamento consiste en seis imágenes que contienen siluetas de animales. Son las siguientes



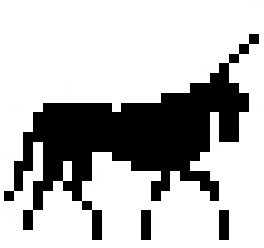
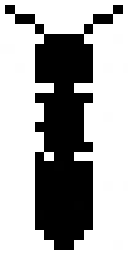
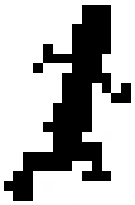




Las siluetas están guardadas en seis ficheros jpg de 30×30 píxeles proporcionados con este enunciado. Son los ficheros `1_caballo_30x30.jpg`, `2_elefante_30x30.jpg`, `3_hipopotamo_30x30.jpg`, `4_hormiga_30x30.jpg`, `5_lagartija_30x30.jpg`, `6_pato_30x30.jpg`.

En primer lugar debemos construir la base de datos y trasformar cada una de esas imágenes en un par de vectores $(\vec{x}, \vec{y})$ donde $\vec{x}$ codifique la imagen y $\vec{y}$ la clasificación correspondiente (esto es, a qué animal corresponde la imagen). El fichero `ejemplos_animales.txt` contiene esa base de datos. El fichero consta de 6 líneas donde cada una de ellas corresponde a un par $(\vec{x}, \vec{y})$ asociado a cada imagen. Cada línea contiene 906 dígitos, separados por comas. Los 900 primeros dígitos son una codificación de la imagen. La imagen consta de 30×30 píxeles blancos y negros. Hemos recorrido las imágenes por columnas, de arriba abajo, escribiendo 0 para el color blanco y 1 para el negro. Los 6 dígitos finales representan la clasificación. Cada vector $\vec{y}$ tiene seis componentes, todos iguales a cero excepto una, con la siguiente codificación

caballo $\rightarrow (1,0,0,0,0,0)$	hormiga $\rightarrow (0,0,0,1,0,0)$
elefante $\rightarrow (0,1,0,0,0,0)$	lagartija $\rightarrow (0,0,0,0,1,0)$
hipopótamo $\rightarrow (0,0,1,0,0,0)$	pato $\rightarrow (0,0,0,0,0,1)$

Una vez entrenada la red, le pediremos que clasifique las siguientes imágenes.

que corresponden a los ficheros

`1_unicornio_30x30.jpg`
`2_hormiga_sin_patas_30x30.jpg`
`3_lagartija_sin_cola_30x30.jpg`

El objetivo es comprobar que, pese a las modificaciones, la red neuronal identifique estas imágenes como *caballo*, *hormiga* y *lagartija*. El fichero `test.txt` contiene tres líneas, cada una con 906 dígitos, siguiendo la codificación anterior. Estos ejemplos no serán usados para entrenar la red, por lo que la clasificación actual no importa. Es $(0,0,0,0,0,0)$ para todos los ejemplos.

A continuación creamos un nuevo proyecto Neuroph y en él una red neuronal Perceptrón Multicapa con 900 neuronas en la capa de entrada, una única capa intermedia con 10 neuronas y una capa de salida con 6 neuronas. Tomamos la función sigmoide como función de activación y elegimos entrenar la red mediante retropropagación con *momentum*. Creamos también el conjunto de entrenamiento a partir del fichero `ejemplos_animales.txt`. Elegimos entrenarla con el método de retropropagación con *momentum* con el conjunto de ejemplos, con los siguiente parámetros: Error máximo 0.01, factor de aprendizaje 0.1 y momentum 0.7.

Vemos que el entrenamiento de la red se detiene con un error cuadrático 0.0031046127833845947 que es menor que el valor máximo 0.01 que le habíamos marcado. Con el botón *Test* podemos comparar los valores obtenido por la red con el valor marcado en el conjunto de entrenamiento.

Una vez entrenada la red, le pedimos que nos de la clasificación de los elementos del conjunto de prueba. Para ello creamos un nuevo conjunto de entrenamiento a partir de `test_animales.txt`. Una vez creado, lo arrastramos hasta la capa de entrada y en lugar de darle al botón *Train*, le damos al botón *Test*. En primer lugar, vemos que la salida para el fichero `1_unicornio_30x30.jpg` ha sido

```
Output: 0,9457; 0,0286; 0,0015; 0,0064; 0,0287; 0,0542
```

donde claramente, el valor dominante corresponde a la primera neurona, por lo que según la red neuronal, la clasificación correcta sería *caballo*. La segunda instancia corresponde la fichero `2_hormiga_sin_patas_30x30.jpg` y la salida ha sido

```
Output: 0,0735; 0,0237; 0,0158; 0,6849; 0,3957; 0,0043
```

donde el valor más alto corresponde a la cuarta neurona. Interpretamos esta salida como *hormiga*. Nótese que la quinta neurona también ofrece un valor relativamente alto, por lo que podemos interpretar que la hormiga sin patas *también se parece* a la lagartija. Por último, la tercera imagen `3_lagartija_sin_cola_30x30.jpg` ha obtenido una salida

```
Output: 0,046; 0,0567; 0,0018; 0,0819; 0,8732; 0,0039
```

donde el valor más alto corresponde a la quinta neurona y por tanto la clasificación ha sido *lagartija*.

Ejercicio 2. Se pide desarrollar un experimento de identificación de imágenes con redes neuronales a elección del alumno. Si no se os ocurre nada, podeis tomar como punto de partida el trabajo de Jovana Stojilkovic (disponible aquí) sobre reconocimiento facial. Ella propone usar Abrosoft Face Mixer que es un producto comercial, pero en Face Detection podemos encontrar software libre. Otra base de dato estándar la proporciona Tom Mitchell en este enlace.

Pero recordad que estas son sólo sugerencias. Podéis usar la base de datos de imágenes y software que queráis.