**Práctica 3**

Otras aplicaciones de las Redes Neuronales Artificiales

30 de Abril de 2014

**AUTORES**

Mauricio Guerreiro Quatrin

Olympia Muruzábal Ishigetani

Índice

Parte 1: Autoencoders 2

Creación del fichero de datos 2

Experimentos con el autoencoder 2

Parte 2: Series temporales 4

Creación del fichero de datos 4

Experimentos con las series temporales 4

Serie1 5

Serie2 6

Conclusiones 6

Esta práctica se divide en dos partes, en cada una de ellas podremos apreciar distintas aplicaciones de las redes neuronales artificiales, concretamente partimos de la red multicapa de retropropagación implementada en la práctica anterior.

# Parte 1: Autoencoders

La primera parte consiste en implementar de un autoncoder, una red cuya salida es igual a la entrada, que se aplica para comprimir datos. En nuestro caso consideraremos imágenes en rejillas de tamaño píxeles que estarán formadas por barras horizontales o verticales.

## Creación del fichero de datos

Implementamos la función ***construye\_db\_autoencoder(int n, char \* nombre\_fichero)*** en la clase Autoencoder.java. Esta función crea un fichero con el formato especificado en el enunciado, con todos los posibles patrones 1+n\*(2n+1) de una imagen de tamaño .

## Experimentos con el autoencoder

Se nos pide realizar un entrenamiento con toda la base de datos, codificando los 1’s como +0,9 y los 0’s como -0,9. Y en lugar del error de clasificación en esta ocasión calcularemos el error de reconstrucción de los pixeles.

Para ello realizamos pequeños cambios en la clase de InputData para que codifique los datos como +0.9 ó -0.9 y creamos una clase BackpropagationEncoded la cual hereda de Backpropagation que solo sobreescribe la clasificación del test. En Backpropagation añadimos lo necesario para que calcule el error de reconstrucción de los pixeles, cambiamos el número máximo de épocas para realizar pruebas con una constante de aprendizaje menor.

Las siguientes gráficas muestran el ECM y el error de reconstrucción (% de píxeles mal reconstruidos) en cada época de entrenamiento. En este caso solo mostramos los resultados con 4 y 5 neuronas.

Podemos observar que con 4 neuronas la red aprende con un error mínimo, pero comprobamos que la red aprende perfectamente el problema con 5 neuronas en la capa oculta con un 0% de error en la reconstrucción. Por lo que concluimos que para realizar una compresión de 9 píxeles son necesarias 5 neuronas, poco más que la mitad de píxeles de modo que la compresión es efectiva.

# Parte 2: Series temporales

En esta parte utilizaremos la red neuronal para predecir el comportamiento de series temporales. En el enunciado se nos dan dos opciones para abordar el problema, optamos por la primera opción: clasificación del comportamiento de la serie en el siguiente punto.

## Creación del fichero de datos

Se nos proporcionan dos series temporales en las que cada línea es un tiempo y en cada fila hay solo un valor. Tenemos que implementar una función ***adapta-fichero-serie(String entrada, String salida, int np)*** que leerá el fichero original y creará un fichero nuevo con el formato habitual. Donde np será el número de atributos y la clasificación será (1 0) si el valor de la serie aumenta o (0 1) en caso contrario. El número de patrones del nuevo fichero es igual al número de puntos del fichero original menos np.

## Experimentos con las series temporales

En este caso no se desordenan los datos del fichero, ya que se trata de una serie temporal y el orden es esencial. Los primeros datos del fichero se utilizarán para el entrenamiento del modelo de predicción y el resto de datos se usan en el test. Realizamos distintas pruebas y finalmente los resultados mostrados utilizan un coeficiente de aprendizaje de 0.01 y 5 neuronas.

Tenemos las siguientes series temporales sobre las que realizaremos las predicciones:

La serie1 como se puede observar es una sinusoidal. Mientras que la serie2 es una serie caótica.

### Serie1

A continuación se muestra el error de test de la predicción de la serie 1 con los siguientes parámetros, utilizando un coeficiente de aprendizaje de 0.01 y 5 neuronas:

Con un 25% de datos para training y un 75% para test

|  |  |  |
| --- | --- | --- |
| Np = 1 | Np = 2 | Np = 5 |
| 50.66666666666667 | 34.11371237458194 | 22.22222222222222 |

Con un 50% de datos para training y un 50% para test

|  |  |  |
| --- | --- | --- |
| Np = 1 | Np = 2 | Np = 5 |
| 50.0 | 12.060301507537687 | 5.05050505050505 |

A partir de los resultados obtenidos se puede apreciar que con Np = 1 la red no es capaz de predecir correctamente, esto puede ser debido a que con un solo punto la red no obtiene suficiente información.

Al tratarse de una serie sinusoidal que se repite periódicamente, si solo utilizamos un 25% de patrones para el entrenamiento la red no es capaz de aprender completamente. Mientras que con un 50% de datos para el entrenamiento la red es capaz de predecir adecuadamente con 5 neuronas en la capa oculta.

### Serie2

A continuación se muestra el error de test de la predicción de la serie 1 con los siguientes parámetros, utilizando un coeficiente de aprendizaje de 0.01 y 5 neuronas:

Con un 25% de datos para training y un 75% para test

|  |  |  |
| --- | --- | --- |
| Np = 1 | Np = 2 | Np = 5 |
| 23.333333333333332 | 4.107762069885303 | 5.2842273819055245 |

Con un 50% de datos para training y un 50% para test

|  |  |  |
| --- | --- | --- |
| Np = 1 | Np = 2 | Np = 5 |
| 22.919999999999998 | 4.561824729891957 | 5.524419535628502 |

Podemos observar que para tratarse de una serie caótica la red tiene un error de predicción bastante pequeña. Esto se debe a que solo realizamos una clasificación de si la serie aumenta o disminuye, por lo que la probabilidad de error en una clasificación es del 50% y la red es capaz de estimar la pendiente de la serie si aumenta o no.

# Conclusiones

En esta práctica hemos podido apreciar las distintas aplicaciones de las redes neuronales en la compresión de datos y la predicción de series temporales, usando para ello la red neuronal multicapa con el algoritmo de retropropagación.