Universidad Autónoma de Madrid

Neurocomputación

Práctica 3 - Autoencoders y series temporales

Enrique Cabrerizo Fernández
Guillermo Ruiz Álvarez

2 de mayo de 2016

Índice

1. Autoencoders.

En esta sección se muestran varios resultados obtenidos por el autoencoder para el reconocimiento de caracteres.

En primer lugar, se ha modificado el fichero `alfabeto.dat` obteniendo un fichero `alfabeto.txt` en el que en cada línea se encuentran los 35 atributos representando el caracter correspondiente (7×5 píxeles por caracter), y 35 datos de salida que son iguales a los de entrada. Este fichero ha sido formateado de tal forma que sea legible por la red neuronal que utiliza *backpropagation* implementado en la práctica anterior, de tal modo que las predicciones son píxeles negros (1) o blancos (0) en función de si cada salida de la red neuronal es mayor, o menor o igual que cero, respectivamente.

La red de retropropagación se ha modificado de tal forma que acepta dos parámetros nuevos:

- `f_training:` Indica el número de distorsiones que se añadirán al conjunto de training.
- `f_test:` Indica el número de distorsiones que se añadirán al conjunto de test.

En ambos casos, en caso de que el valor del nuevo parámetro sea $n > 0$, se generarán 10 versiones ruidosas para cada letra, esto es, se tendrán 10 copias de cada muestra y para cada una de ellas se alerarán n valores aleatorios. En caso de que $n = 0$, entonces se omitirá este parámetro, es decir, se realizarán los cálculos sin añadir versiones ruidosas.

El cambio se realizará de la siguiente forma: si p es el valor del píxel a cambiar, el nuevo valor será el siguiente (% representa la operación módulo, de tal modo que se cambiará el valor 0 por 1 y viceversa):

$$p = (p + 1)\,\%2$$

El programa informará, tras la ejecución cual ha sido el valor de `PE_Test:` el error promedio cometido en la fase de test medido en **PE** (píxeles errados por letra).

Tras la modificación del fichero para que la red que utiliza retropropagación pueda tomarlo como entrada, se han realizado tres casos de prueba:

- **Caso 1:** Tanto el conjunto de entrenamiento como el conjunto de test no contienen versiones ruidosas.
- **Caso 2:** El conjunto de entrenamiento no contiene versiones ruidosas y el conjunto de test contiene 10 versiones ruidosas por cada letra. Se utilizan valores para `f_test` de $1, 3$ y 5.
- **Caso 3:** Tanto el conjunto de entrenamiento como el de test tendrán 10 versiones ruidosas por cada letra. Se utilizan valores para `f_training` y `f_test` de $1, 3$ y 5.

1.1. Caso 1: Abecedario sin ruido.

Para este caso se han realizado varias pruebas con distinto número de nodos de la capa interna de la red. El resto de parámetros utilizados son:

- Número máximo de épocas: 200
- Porcentaje de datos utilizado para la fase de training: 100 %
- Porcentaje de datos utilizado para la fase de test: 100 %
- Tasa de aprendizaje: $\alpha = 0{,}01$
- Ruido en fase de training: `f_training` $= 0$
- Ruido en fase de test: `f_test` $= 0$

Los resultados obtenidos son los siguientes:

N.Nodos capa oculta	**PE_Test**
10	0,2692
15	0,1154
18	0,0385
20	0,00
25	0,00
30	0,00
35	0,00

Es decir, se tiene que se puede aprender el abecedario completo poniendo un número de neuronas de la capa oculta ≥ 20. Por lo que no es necesario escoger un subconjunto menor del conjunto total, ya que la red es capaz de aprender el abecedario completo.

1.2. Caso 2: Conjunto de test con ruido.

Para este caso se han realizado diversas pruebas variando el valor del parámetro `f_test`, tomando éste los valores 1, 3 y 5. Es decir, se generarán 10 copias ruidosas de cada letra y en cada copia se invertirán 1, 3 y 5 píxeles, respectivamente, en posiciones aleatorias. Los parámetros utilizados para las pruebas han sido los siguientes:

- Número máximo de épocas: 200
- Número de nodos en la capa oculta: 25.
- Porcentaje de datos utilizado para la fase de training: 100 %
- Porcentaje de datos utilizado para la fase de test: 100 %
- Tasa de aprendizaje: $\alpha = 0{,}01$
- Ruido en fase de training: `f_training` $= 0$

Los resultados obtenidos son los siguientes:

f_test	PE_Training	PE_Test
1	0,00	0,1192
1	0,00	0,1577
1	0,00	0,1423
3	0,00	0,8769
3	0,00	0,8154
3	0,00	0,8038
5	0,00	2,1231
5	0,00	2,0923
5	0,00	2,1269

Como se puede observar, en todo momento la red aprende sin fallos todo el abecedario (el valor de `PE_training` es siempre 0), ya que no se alteran los datos en la fase de training y se ha elegido utilizar 25 neuronas en la capa oculta. Sin embargo, el número medio de errores cometidos en la fase de test es distinto de cero en todos los casos debido al ruido introducido.

Aún así, incluso con el ruido introducido, el número de errores promedio que se ha cometido en la fase de test (`PE_Test`) es siempre menor que el número de alteraciones introducidas (`f_test`). Por lo que la red neuronal es capaz de disminuir el ruido introducido.

1.3. Caso 3: Conjuntos de training y test con ruido

Para este caso se han realizado diversas pruebas variando tanto el valor del parámetro `f_training` como el valor del parámetro `f_test`, tomando estos los valores 1, 3 y 5. Es decir, para ambos conjuntos de training y test se generarán 10 copias ruidosas de cada letra y en cada copia se invertirán 1, 3 y 5 píxeles, respectivamente, en posiciones aleatorias. Los parámetros utilizados para las pruebas han sido los siguientes:

- Número máximo de épocas: 200
- Número de nodos en la capa oculta: 25.
- Porcentaje de datos utilizado para la fase de training: 100 %
- Porcentaje de datos utilizado para la fase de test: 100 %
- Tasa de aprendizaje: $\alpha = 0{,}01$

Los resultados obtenidos son los siguientes:

f_training	**f_test**	**PE_Training**	**PE_Test**
1	1	0,0000	0,0538
1	1	0,0000	0,0615
1	1	0,0115	0,0692
3	3	0,0038	0,5346
3	3	0,0000	0,4077
3	3	0,0038	0,4269
5	5	0,0038	1,4654
5	5	0,0154	1,4923
5	5	0,0692	1,3962

Se puede observar que esta vez sí que varía el valor de `PE_Training` debido al ruido introducido en el entrenamiento. Sin embargo, el error promedio cometido durante la fase de entrenamiento no llega a 0,1 píxeles por letra. Esto implica que a veces la red no es capaz de aprender al 100 % el abecedario completo, sin embargo, al ser entrenada con varias muestras con ruido introducido y siempre la salida correcta, es capaz de disminuir de manera importante el ruido introducido en la fase de test.

En la siguiente tabla se muestra una comparación de los valores de `PE_Test` entre el **Caso 2** y el **Caso 3**.

Caso 2: PE_Test	**Caso 3: PE_Test**	**% de mejora**
0,1192	0,0538	121,56 %
0,1577	0,0615	156,42 %
0,1423	0,0692	105,64 %
0,8769	0,5346	64,03 %
0,8154	0,4077	100,00 %
0,8038	0,4269	88,29 %
2,1231	1,4654	44,88 %
2,0923	1,4923	40,21 %
2,1269	1,3962	52,34 %

En la tabla se muestra el porcentaje de mejora del error obtenido en el **caso 2** y el **caso 3**. Se puede observar que en todo momento se supera el 40 % de mejora. Por tanto, aunque haya casos en los que la red no pueda aprender el abecedario completo con ruido introducido, se obtienen mejores resultados a la hora de predecir datos con ruido.

2. Series temporales.

2.1. Código Implementado.

La implementación para los primeros cuatro ejercicios de series temporales se encuentran en la carpeta **bp**/ (scripts) y en la subcarpeta **bp**/**src** (funciones).

Para adaptar el fichero se ha creado el script *adapta_fichero.m* que hace uso de la función *adapta_fichero_serie* implementada acorde a las especificaciones del ejercicio 1.

Para cumplir lo pedido en los ejercicios 2 y 3, se han creado copias de las funciones de entrenamiento de perceptron lineal *bp_train.m* y *network_test.m* y se han adaptado al nuevo código cambiando las funciones de transferencia y computando el error cuadrático medio en test. Dichas funciones se encuentran en *bp_train_time.m* y *network_test_time.m*. Los scripts que ejecutan los ejercicios de series temporales son *series.m* y *series_recursive.m*, este último, como pide el ejercicio 8, se apoya en la función *predice_recursivamente.m*.

2.2. Ejercicio 5.

En la siguiente figura podemos ver la gráfica correspondiente a la serie temporal *p3_serie1.txt*.

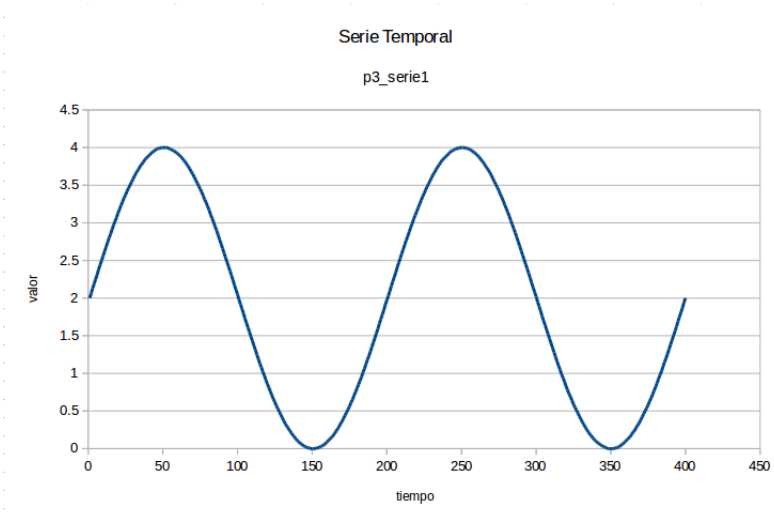

Figura 1: Serie temporal del fichero *p3_serie1.txt*.

Como se puede comprobar es una serie temporal bastante sencilla (es periódica) que se corresponde con una ecuación sinusoidal. La expresión explícita de la serie que representa es:

$$valor(t) = 2 + 2sin\left(\frac{\pi}{100}t\right)$$

2.3. Ejercicios 6 y 7.

A continuación se muestra una tabla con los valores del error cuadrático medio obtenido para las predicciones y la predicción básica con las diferentes pruebas realizadas.

	NA=1		**NA=2**		**NA=5**	
Training %	25 %	50 %	25 %	50 %	25 %	50 %
MSE	0.613495	0.183084	0.403417	0.035622	0.087812	0.038827
MSE básico	0.002003	0.002002	0.001993	0.002002	0.1982	0.001985

Como se puede comprobar, los errores son mayores con la predicción realizada con el perceptrón que con la predicción básica, lo que nos indica que en general no está resultando un buen método para resolver el problema. Podemos ver sin embargo que los mejores valores de error se encuentran para $Na = 2$.

Indagando en este aspecto, podemos ver que el seno es una función que cumple $sin''(t) = -sin(t)$. La segunda derivada la podemos aproximar como combinación lineal de los valores en $sin(t)$,$sin(t-1)$ y $sin(t-2)$, de forma que la función seno en t se puede aproximar linealmente con los valores en $t-1$ y $t-2$. No obstante, nuestra función de transferencia (sigmoide bipolar), no es lineal, con lo que la red se ve obligada a trabajar con pesos en una zona en la que la sigmoide sea lo más lineal posible (pesos muy cercanos a 0). Aún así el resultado no es demasiado satisfactorio. Para contrastar este hecho intuitivo, se ha probado a cambiar la función de transferencia a la identidad, logrando un error cuadrático medio en test de 0,000026 para $Na = 2$, mejorando considerablemente el error de la predicción básica.

Por otro lado, se puede comprobar que los errores son mayores también cuando entrenamos únicamente con el 25 % del conjunto ya que no enseñamos al perceptrón una oscilación completa de la función seno y por lo tanto no es capaz de aprender a predecir correctamente con valores negativos.

En las figuras 2-4 se pueden ver gráficas con las predicciones para los distintos valores de Na y porcentajes de entrenamiento.

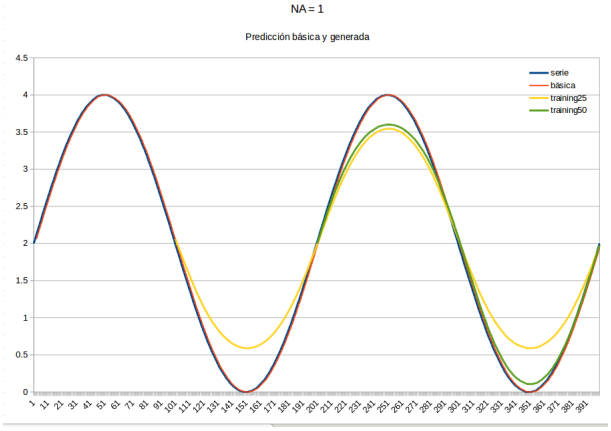

Figura 2: Predicciones para NA = 1.

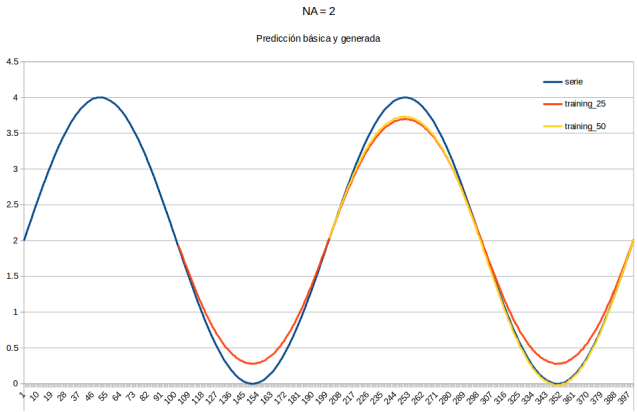

Figura 3: Predicciones para NA = 2.

Como se puede comprobar en dichas figuras, una de las mejores predicciones en cuanto a similitud de la gráfica es la realizada para NA = 2 y entrenamiento con el 50 % de las muestras (una oscilación completa), no obstante, la predicción realizada para los mismos parámetros, pero con función de transferencia lineal tanto en la capa de salida como en la capa oculta es mucho más precisa y se puede ver en la figura 5.

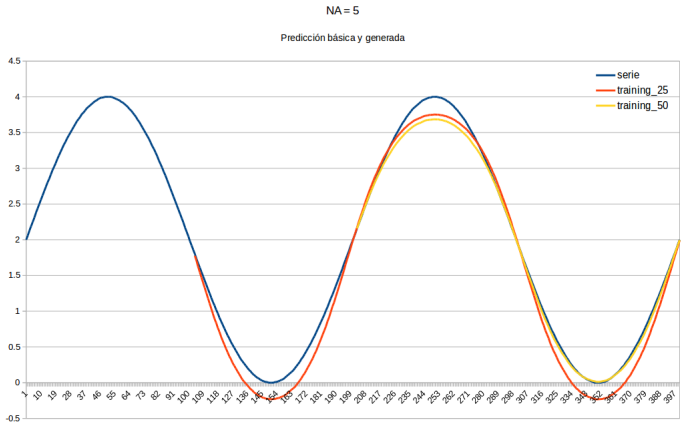

Figura 4: Predicciones para NA = 5.

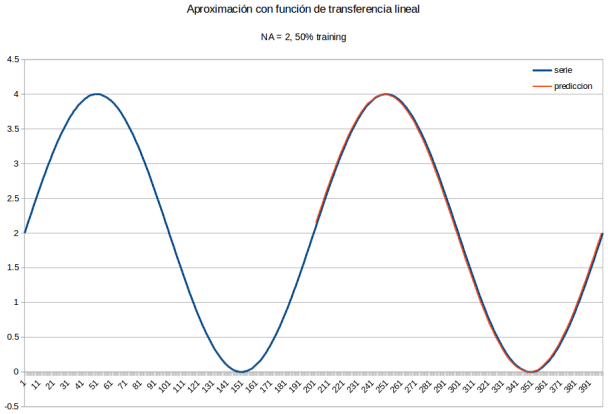

Figura 5: Predicciones para NA = 2 y función de transferencia lineal.

No obstante se ha detectado una mejora en el error cuadrático medio utilizando la función de transferencia sigmoide cuando se aumenta el número de neuronas en la capa oculta, si bien han sido necesarias 50 neuronas en la capa oculta para bajarlo hasta 0.015744, que sigue siendo mayor que el error de la predicción básica.

2.4. Ejercicio 9.

En la figura 6 podemos ver la serie original y la evolución de las predicciones recursivas desde diferentes puntos.

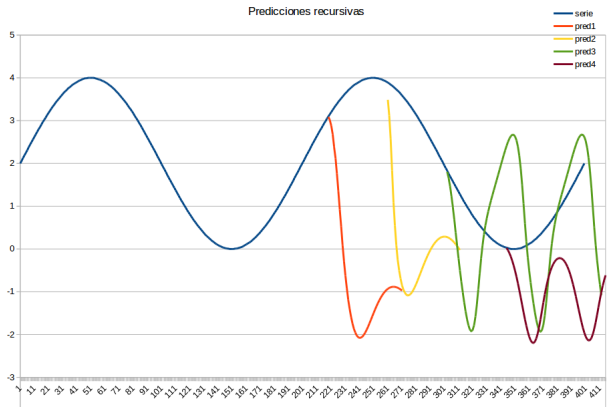

Figura 6: Predicciones recursivas. NA = 5.

Como se puede comprobar, la predicción recursiva no es, en absoluto precisa. En ocasiones se ha logrado que la serie que se representa sea bastante similar a una función sinusoidal, pero el período de dicha función suele verse reducido a $\frac{1}{4}$ del período original.

Se ha probado a realizar dichas predicciones con función de transferencia lineal también en la capa oculta obteniendo un resultado que aproxima mejor la función sinusoidal. Se puede ver en la siguiente gráfica.

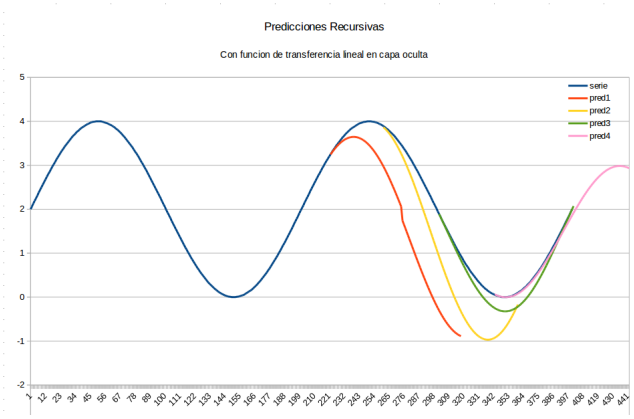

Figura 7: Predicciones recursivas con transferencia lineal.

En este caso se puede comprobar como las gráficas se aproximan muy bien inicialmente a la serie temporal original, aunque tras una serie de pasos, el error cometido se acumula y va provocando la separación entre ambas.

2.5. Ejercicio 10.

En la figura 8 vemos el aspecto de la serie temporal de la segunda serie proporcionada como material de la práctica y, como se puede ver, esta serie es considerablemente más complicada que la del ejercicio anterior, ya que aparentemente no es periódica.

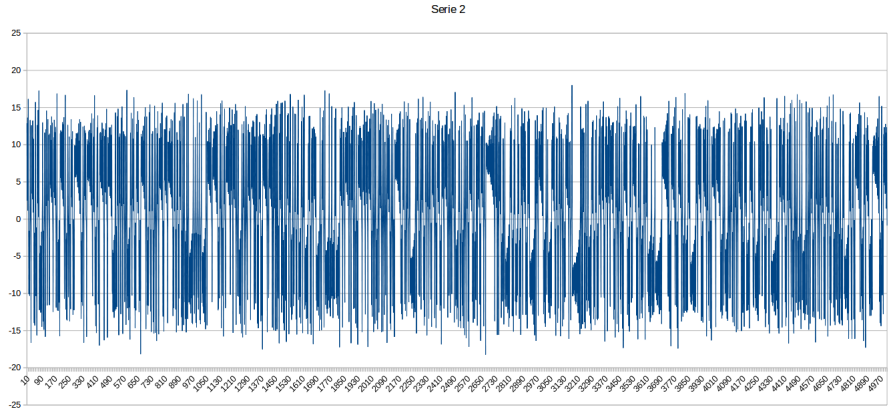

Figura 8: Serie temporal 2.

En la siguiente tabla vemos los valores del error cuadrático medio para las diferentes pruebas realizadas, en las que se ha utilizado el 50 % del set de entrenamiento:

	NA=1	**NA=2**	**NA=5**
MSE	23.855451	17.865888	11.584109
MSE básico	48.740883	48.740883	48.748843

Se observa que en esta ocasión aparentemente funciona mejor Na = 5. Para este dato la gráfica con las predicciones es la siguiente:

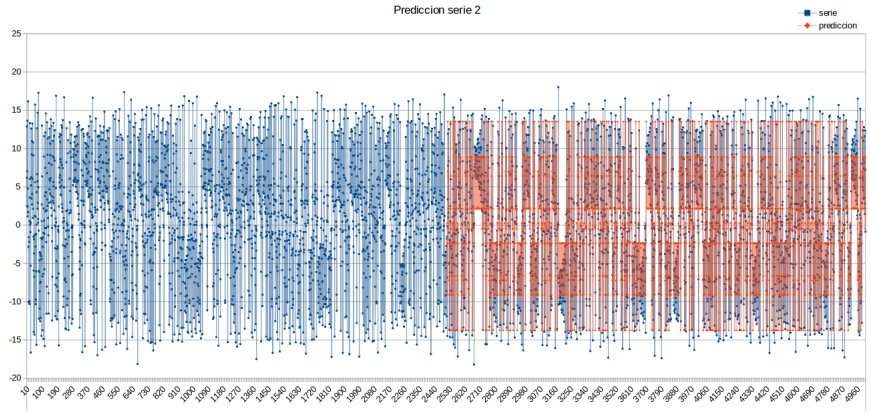

Figura 9: Serie temporal y predicciones con NA = 5.

Podemos observar a simple vista que la predicción se concentra alrededor de unos 10 valores distintos mientras que la serie original no sigue ningún patrón aparente, con lo que la predicción no es buena.

Ante este resultado no podemos esperar una predicción recursiva demasiado precisa, como vemos a continuación.

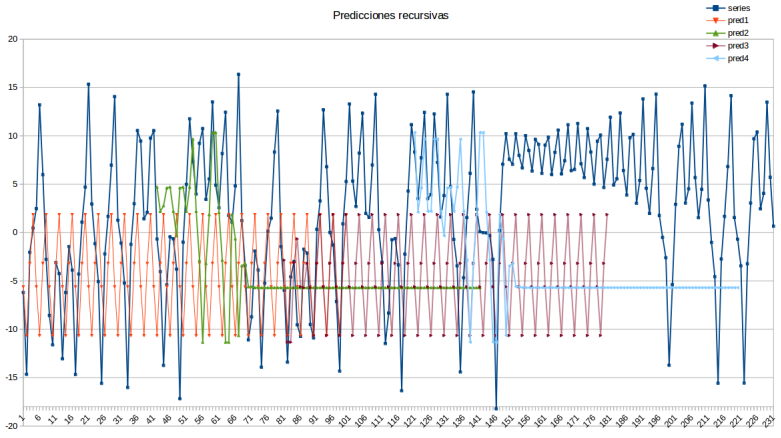

Figura 10: Serie temporal y predicciones recursivas con NA = 5.