Práctica 3 de Neurocomputación

Otras aplicaciones de las Redes Neuronales Artificiales

Ares Aguilar Sotos
Daniel Giménez Llorente

Tabla de Contenidos

1. Introducción

En esta práctica nos marcamos como objetivo el desarrollo de otros tipos de redes neuronales, en concreto el autoencoder y las series temporales.

A la hora de definir nuestra solución, hemos continuado con la práctica anterior, añadiéndole las funcionalidades requeridas en esta prácticas. Estas funcionalidades son añadir los modos necesarios para las nuevas redes (-m autoencoder/timeseries).

2. Autoencoders

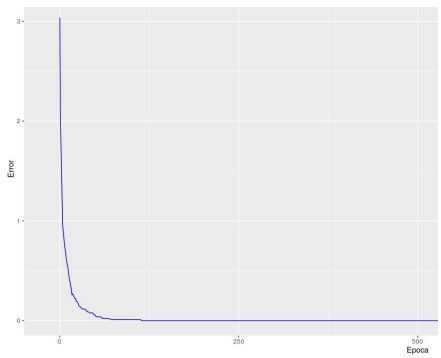
Para la creación del autoencoder no han sido necesarios demasiados cambios. Al estar desarrollando una práctica de redes neuronales incremental, tuvimos en cuenta que podíamos tener que añadir otro tipo de redes neuronales. Además añadimos el cálculo del número de píxeles errados para poder observar el buen funcionamiento de la red.

2.1 Creación del fichero alfabeto.txt

Para ello, creamos un script en awk que cuando detectaba un patrón nuevo, (que se correspondía con una letra) lo imprimía en un fichero en el formato que acepta nuestra red neuronal.

2.2 Alfabeto sin ruido

En este apartado, usamos el alfabeto entero para aprender y luego comprobamos el error resultante de todo el fichero. Hemos usado un learning rate de 0.02 con 1000 épocas y 12 neuronas. Con estos parámetros, el error pasa a ser 0 tanto en el train como en el test por lo que es capaz de aprender todo el abecedario.



Para ejecutar esta simulación basta llamar al makefile con el objetivo p3.2.2-alfabeto.

2.3 Testear con datos ruidosos

Para generar los datos ruidosos, basta con hacer “make p3-databases”. Esta orden genera tres ficheros usando un script de awk con 1, 3 y 5 errores con 10 repeticiones cada una.
Después hemos entrenado el alfabeto con los datos limpios y testeado con los datos ruidosos.

Errores	# pixeles errados (Train)	# pixeles errados (Test)
1	0.000000	0.222028
3	0.000000	0.698427
5	0.000000	1.203671

Como se puede observar, la red aprende bastante bien y la media de píxeles errados es bastante menor a los mismatches introducidos.

Para ejecutar las simulaciones mencionadas, hay que llamar a los objetivos p3.2.3-alfabeto-ruido1, p3.2.3-alfabeto-ruido3 y p3.2.3-alfabeto-ruido5 del makefile.

2.4 Entrenamiento y test con ruido

Para este apartado hemos usado todo el conjunto ruidos tanto para train como para test. Seguimos entrenando 12 neuronas 1000 épocas con los mismos parámetros.

Errores	# pixeles errados (Train)	# pixeles errados (Test)
1	0.020979	0.020979
3	0.361014	0.361014
5	0.667832	0.667832

Como se puede observar, los resultados son bastante mejores si tenemos en cuenta el caso anterior. Esto se debe a que, como en el conjunto de entrenamiento tiene versiones ruidosas de las letras, es menos específica y puede aprender mejor.

Para ejecutar estas simulaciones, hay que llamar a los objetivos p3.2.4-alfabeto-ruido1, p3.2.4-alfabeto-ruido3 y p3.2.4-alfabeto-ruido5 del makefile.

3. Series temporales

Para la implementación de este apartado, hemos creado un modo nuevo (-m timeserie), que computa el error de una forma diferente y además añade el error con respecto a un modelo de referencia .

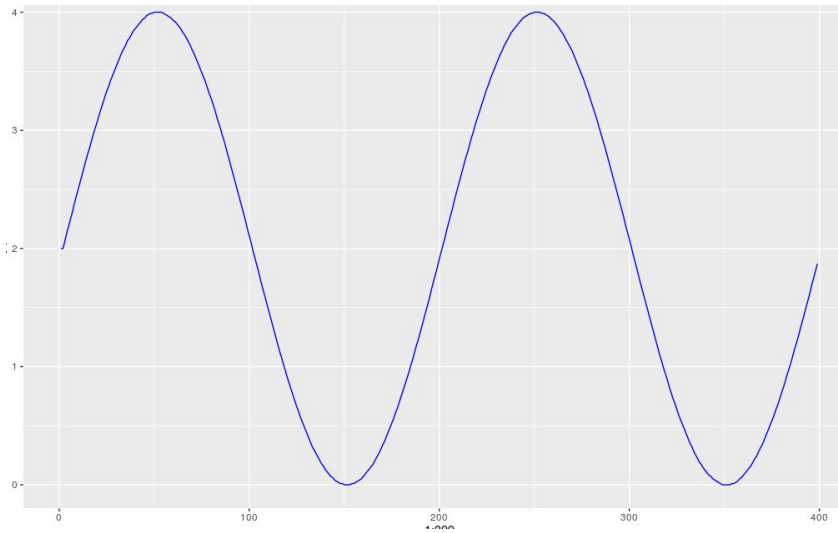
Además hemos creado una función que recibe :
- Nombre del fichero con la serie original
- Nombre del fichero que se quiere crear
- Número de puntos anteriores (Na) que se usan para predecir.
- Número de puntos siguientes (Ns) que se quieren predecir.

Y crea un fichero de tipo serie temporal que es adecuado a la entrada de nuestra red neuronal. Esto se realiza haciendo una llamada a un script que formatea el fichero.

3.1 p3_serie1.txt

3.1.1 Gráfica

Lo primero ha sido representar el serie en forma de gráfica:

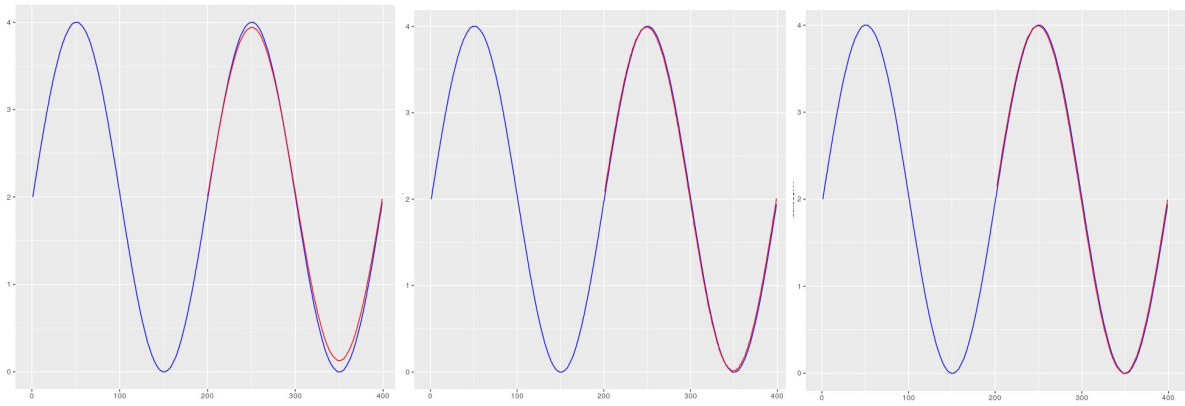


Tiene una forma sigmoidal que parece relativamente fácil de aprender.

3.1.1 Predicciones

Hemos entrenado la red con 200 patrones, mientras que hemos dejado de test otros 200. Después hemos ido variando Na teniendo fijo el número de Ns=1. Se han mantenido las 12 neuronas con 1000 épocas del apartado anterior.

Na=1 Na=2 Na=5



Na	Error Train (%)	Error Train Modelo(%)	Error Test(%)	Error Test Modelo(%)
1	0.007570	0.001993	0.007537	0.002002
3	0.000147	0.001993	0.000147	0.001993
5	0.000037	0.001964	0.000038	0.001982

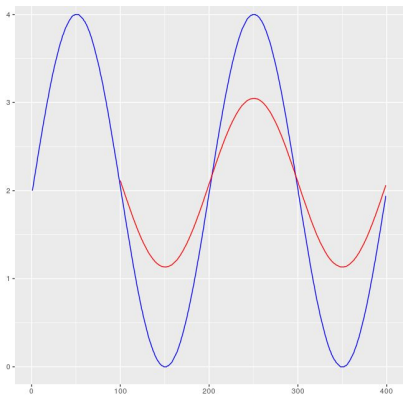
Como se puede observar en las gráficas, el error con Na= 2 y 5 es minúsculo, ya que éstas coinciden. Si lo comparamos con el modelo de referencia, vemos como nuestra red neuronal tiene más precisión excepto en el primer caso;lo cual tiene sentido ya que solo entrena con un valor de entrada.

Para ejecutar las simulaciones de este apartado hay que llamar a los objetivos p3.3.1.1-timeserie1, p3.3.1.1-timeserie2 y p3.3.1.1-timeserie5 del makefile.

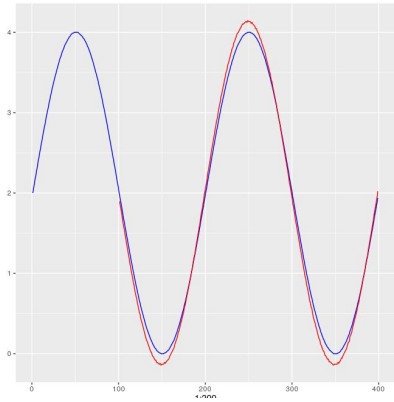
3.1.2 Predicciones (100-300)

Para este apartado hemos hecho lo mismo que el apartado anterior, pero con un porcentaje diferente de train y test.

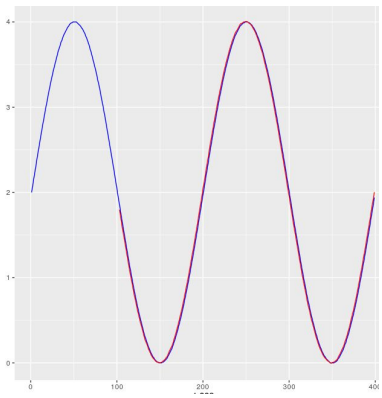
Na=1



Na=2



Na=5



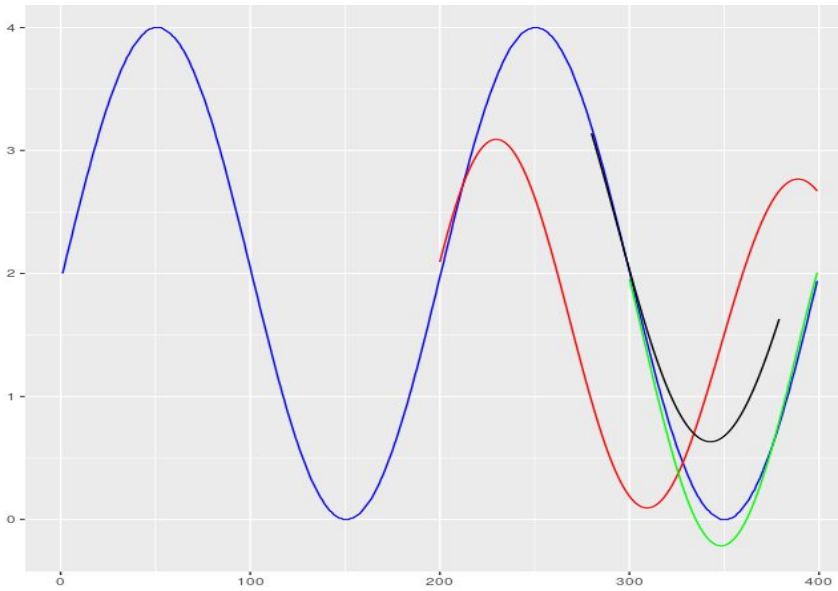
Na	Error Train (%)	Error Train Modelo(%)	Error Test(%)	Error Test Modelo(%)
1	0.436406	0.001983	0.591388	0.002002
3	0.009408	0.001967	0.009028	0.001993
5	0.000048	0.001964	0.000030	0.001982

En este caso vemos que con Na=1 la red neuronal no es capaz de captar la altura de la curva, sin embargo, según vamos aumentando su valor, acaba acertando con un error mínimo. No puede acertar bien porque solo tiene como conjunto de train la parte superior de la curva y además solo un dato de entrada.

Estas simulaciones se corresponden a los objetivos del makefile p3.3.1.2-timeserie1, p3.3.1.2-timeserie2, p3.3.1.2-timeserie5.

3.1.3 Recursiva

Hemos implementado una función que predice de forma recursiva todos los puntos futuros a partir de un punto dado. Hemos creado las gráficas para observar si funciona adecuadamente o no este modelo:



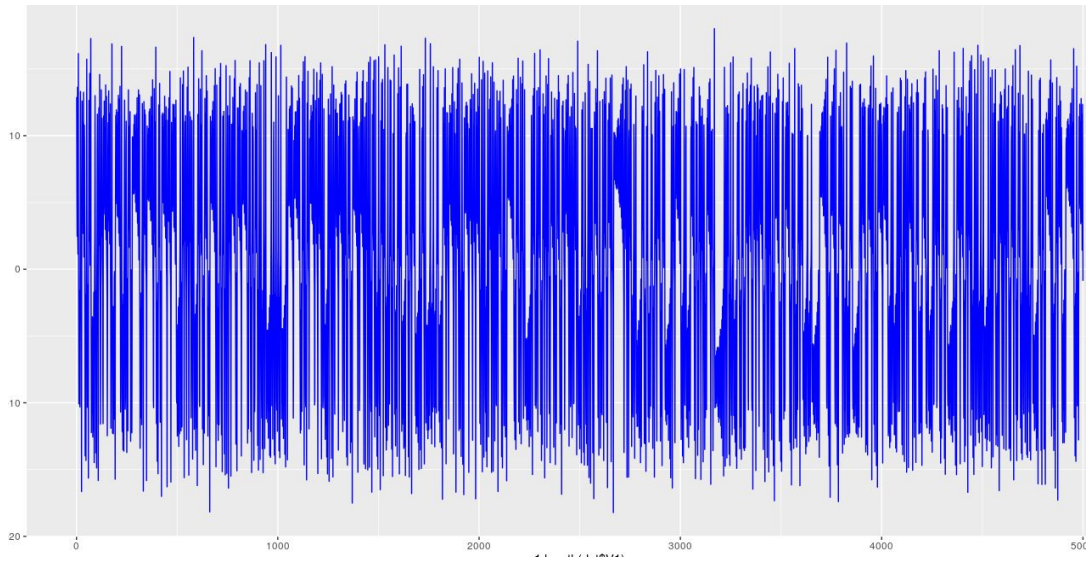
Hemos lanzado la función desde diferentes partes de la gráfica. Como se puede observar, al principio se parecen mucho pero al cabo de un rato empiezan a separarse. Cuanto más se acumula el error mayor es la diferencia.

Esta simulación puede ejecutarse con el objetivo p3.3.1.3-recursiva del makefile.

3.2 p3_serie2.txt

3.2.1 Gráfica

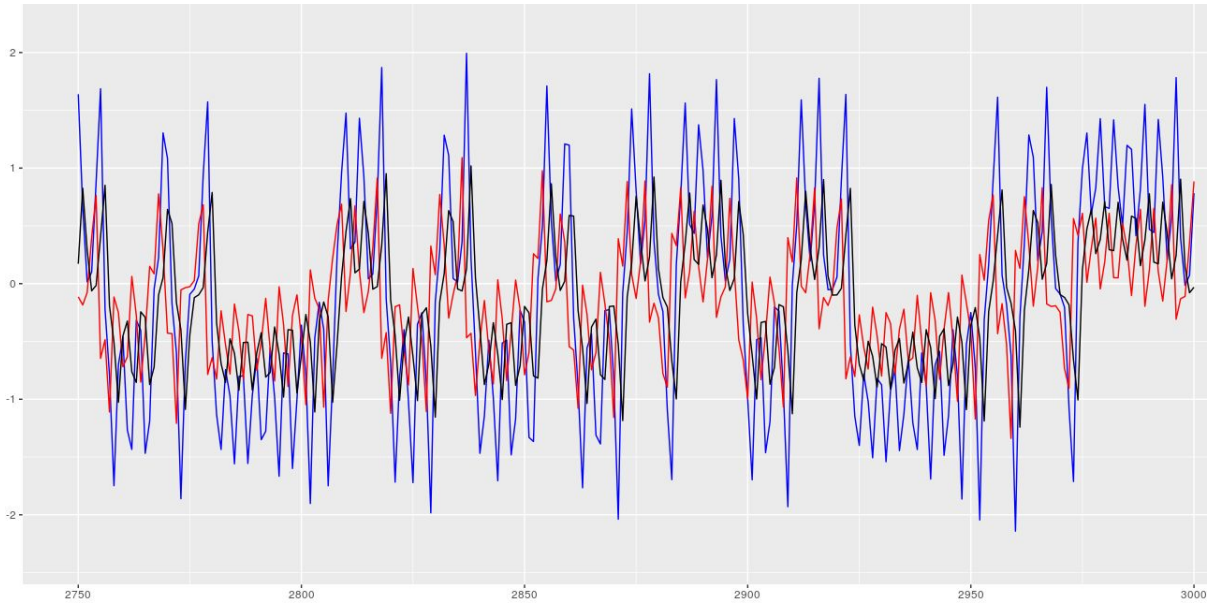
Lo primero ha sido representar el serie en forma de gráfica:



Este parece bastante más complicado que el anterior

3.2.2 Predicciones

Hemos entrenado la red con la mitad de los patrones para probar con el resto. Después hemos ido variando Na teniendo fijo el número de Ns=1. Se han mantenido las 12 neuronas con 1000 épocas del apartado anterior. Esto nos produjo un error: al usar la función identidad en todas las capas, los pesos de las conexiones empezaban a crecer y se desbordaban, por ello, antes de analizar esta serie temporal, procedimos a normalizar sus valores. El resultado fue bastante bueno:



Solamente muestra 250 muestras, pero es representativo de todo el conjunto. La línea roja se corresponde con Na=5 y la negra con Na=1.

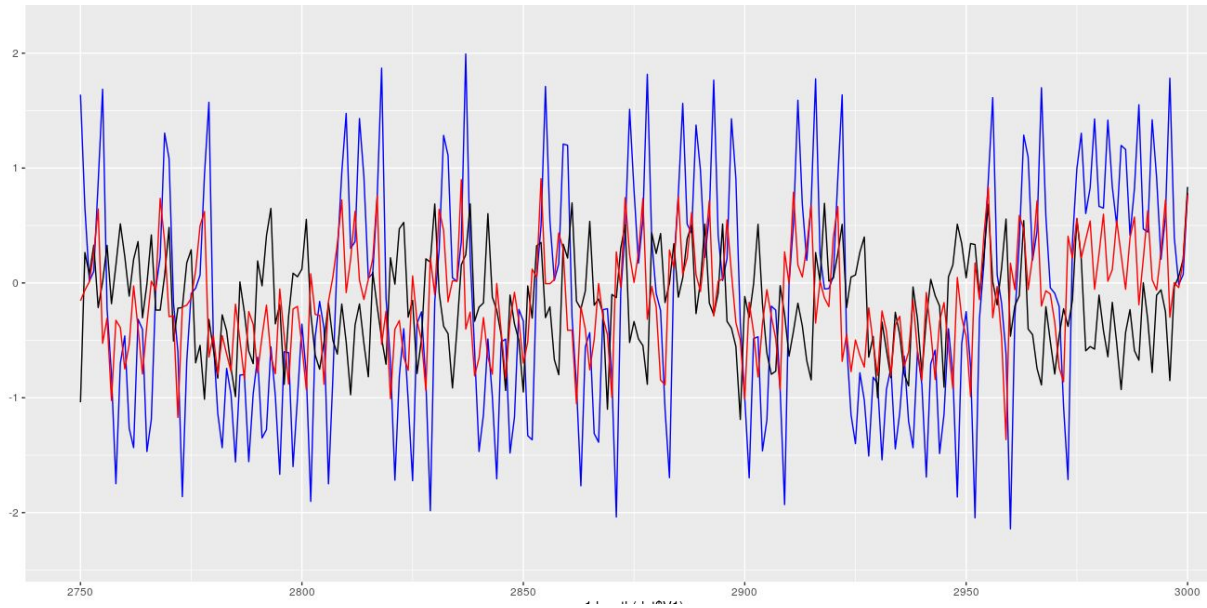
Ambas simulaciones se pueden ejecutar con los objetivos del makefile p3.3.2.2-timeserie1 y p3.3.2.2-timeserie5.

Na	Error Train (%)	Error Train Modelo(%)	Error Test(%)	Error Test Modelo(%)
1	0.657234	0.811816	0.635726	0.780763
5	0.651432	0.812174	0.639800	0.780891

Es un problema más difícil pero aún así consigue predecir más o menos la forma.

3.2.3 Predicciones (100-300)

Para este apartado hemos hecho lo mismo que el apartado anterior, pero con un porcentaje diferente de train y test.



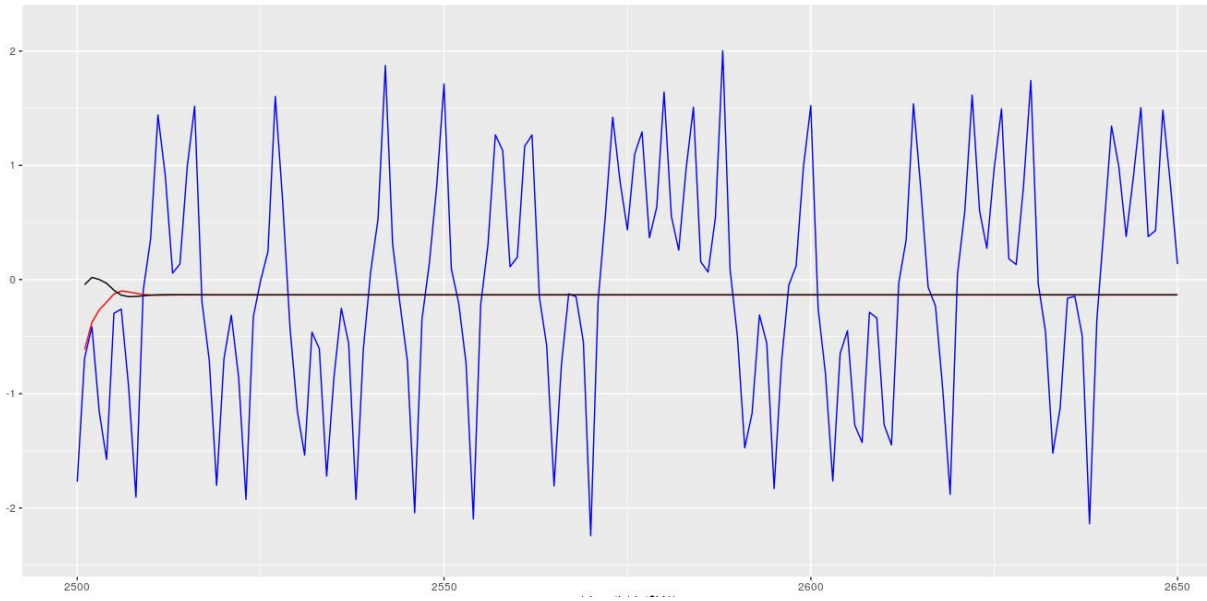
La línea roja usa Na=15 mientras que la negra Na=5. Estas dos simulaciones se corresponden con los objetivos del makefile p3.3.2.3-timeserie5 y p3.3.2.3-timeserie15.

Na	Error Train (%)	Error Train Modelo(%)	Error Test(%)	Error Test Modelo(%)
5	0.646694	0.787944	0.648939	0.799393
15	0.694689	0.815429	0.217484	0.815429

Aumentando el valor de Na, se puede conseguir que aumente la precisión hasta un máximo que nosotros conseguimos con Na=15

3.2.4 Recursiva

Hemos implementado una función que predice de forma recursiva todos los puntos futuros a partir de un punto dado. Hemos creado las gráficas para observar si funciona adecuadamente o no este modelo:



Esta simulación corresponde al objetivo p3.3.2.4-recursiva del makefile.

Parece que no funciona muy bien la función recursiva en este ejemplo, esto se puede deber a la poca regularidad que tiene el fichero.