

Neurocomputación
Práctica 3:
Aplicaciones del Perceptrón Multicapa.

Daniel Gutiérrez Navío

Matías Bender

Grupo 2461

30 de abril de 2014

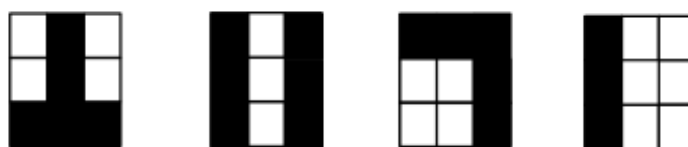
Índice

1. Autoencoder	1
1.1. Imágenes de 3x3	1
1.2. Imágenes más grandes	4
2. Predicción tendencia en series temporales	6
2.1. Predicción sobre serie sinusoidal	6
2.2. Serie Caótica	11
2.3. Conclusión	12

1. Autoencoder

Un autoencoder es una red neuronal usada para aprender y buscar codificaciones eficientes. Sus aplicaciones más importantes son la eliminación de ruido y distorsiones en la entrada, y la compresión de datos. El objetivo de este es aprender a comprimir, desde un conjunto de datos, una codificación. Para ello se utilizan redes cuya salida es igual a la entrada. Se busca que la red capture la estructura interna del problema. En la siguiente sección analizaremos el problema de compresión de codificaciones utilizando esta idea.

Buscaremos codificar una conjunto de imagenes de N pixeles \times N pixeles con dos tonos, compuestas solamente por un máximo de dos lineas verticales y/o horizontales.



Para ello utilizaremos un Perceptron Multicapa al que entrenaremos con todas las posibles imagenes, poniendo como objetivo que las aprenda en su totalidad. El parámetro a estimar es la cantidad de neuronas en la capa oculta necesarias para que el problema sea resuelto. Si obtenemos un número menor de neuronas en esta capa que en la de entrada, entonces podremos codificar o decodificar estas imagenes utilizando para eso las activaciones de la capa oculta, que serán menos que los datos de entrada.

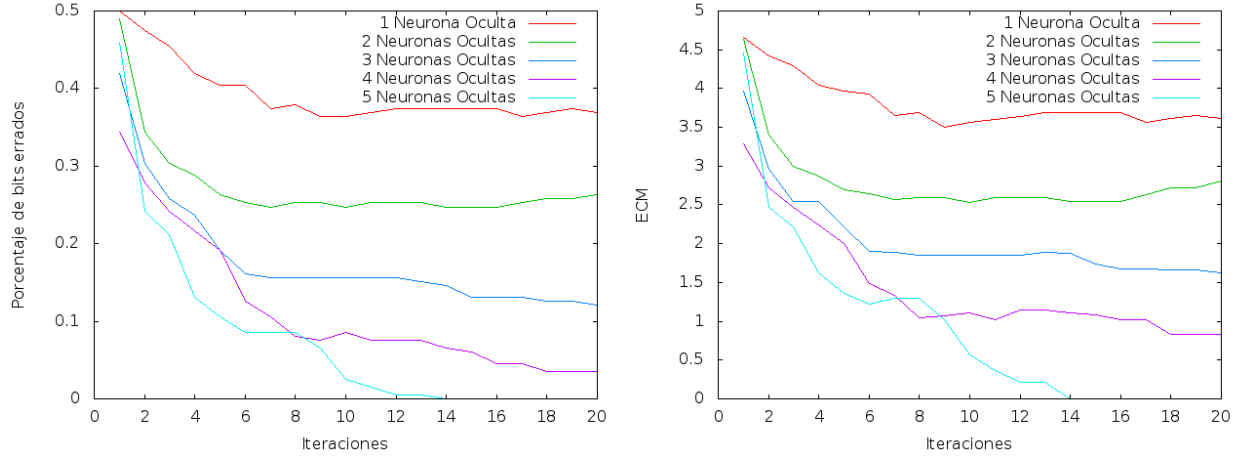
1.1. Imágenes de 3x3

Considerando todas las imagenes de 3 pixeles \times 3 pixeles que cumplen las condiciones de este problema, la cantidad de neuronas necesarias, en la capa oculta, para aprender el problema es 5. Dado que no importa demasiado cuántas iteraciones tarda el algoritmo en aprender, si no que aprenda correctamente, el factor de aprendizaje puede ser tomado como un número chico, en este caso 0,1.

En los siguientes gráficos se puede observar como evoluciona tanto el Error Cuadrático Medio, medido en la cantidad de pixeles equivocados, como el porcentaje de estos pixeles sobre el total hasta converger totalmente en 5 neuronas.

Es interesante observar que con 4 neuronas la red aprende casi todos los patrones. El único patrón en el que siempre falla con esa arquitectura es en la imagen sin lineas. Más adelante reflexionaremos sobre ello.

De esta manera, con solo 5 números podemos caracterizar el patrón del cual estamos hablando. Sin embargo, hay que tener mucho cuidado si se está pensando en una codificación real. Los números de los cuales se hablan son números de punto flotante, o sea que ocupan 32b cada uno (el doble si son de doble precisión). Si la codificación original toma un Integer (32b)



(a) Porcentaje de píxeles equivocados según época

(b) ECM de los píxeles equivocados

Figura 1: Entrenamiento con distinta cantidad de neuronas en la capa oculta

para almacenar cada píxel, el beneficio de la nueva codificación es de 180 % la original. Sin embargo, podrían llegar a codificarse 32 píxeles en un solo Integer, por lo que en este caso, la codificación del autoencoder sería 20 % de la original, o sea gastaría 5 veces su cantidad de memoria.

Vale la pena tomarse un tiempo (quizás no tan corto) para analizar qué está ocurriendo en la red neuronal entrenada una vez que aprende perfectamente todos los patrones. Para lograr dicho objetivo decidimos observar cuál es la activación de cada neurona en la capa oculta según cada ejemplo de entrada y con eso agruparlas según clases que permitan observar lo que está ocurriendo. El primer acercamiento consistió en tomar dos clases distintas, la de las activaciones mayores a cero y las de las menores. Sin embargo, no era poder identificar unívocamente a cada elemento conociendo simplemente la clase a la que pertenece según cada neurona oculta. O sea, había dos entradas que pertenecían a las mismas clases en cada neurona, por lo que no se podía ver qué era lo que las diferenciaba. El segundo acercamiento consistió en cuatro clases según la salida de cada neurona, las menores a $-0,5$, las negativas mayores a $-0,5$, las positivas menores a $0,5$ y las positivas mayores a $0,5$. A continuación se muestra el resultado de esta clasificación.

	$-1 < n < -0,5$	$-0,5 < n < 0$	$0 < n < 0,5$	$0,5 < n < 1$
Neurona 1				
Neurona 2				
Neurona 3				
Neurona 4				
Neurona 5				

En estos gráficos podemos notar diferentes cosas. La primera es que la mayoría de las entradas causan que las neuronas de la capa oculta tomen valores de salida distantes entre ella, o sea, las neuronas clasifican con más seguridad dado que el patrón que detectan los detectan con bastante seguridad, dado que toman valores de salida, en modulo, cercanos a 1. Esta es la causa fundamental por la que la codificación es eficiente, dado que se encuentran patrones determinantes para cada imagen.

Por otro lado, podemos notar como los complementos de las imagenes se encuentran siempre en categorías enfrentadas. Esto ocurre por la codificación que se tiene de los parámetros donde los blancos se representan con $-0,9$ y los negros con $0,9$, por lo que invertir los colores es multiplicar por -1 las entradas.

Yendo más al detalle, podemos observar como la Neurona 1 asigna valores más bajos a todas las entradas que tienen una linea vertical en el centro. A su vez, asigna valores más grandes a todas las que tienen una linea vertical a la derecha en mayor medida y a izquierda en menor, salvo que caigan en el caso anterior. La Neurona 2 asigna valores más bajos a quien tenga una linea horizontal arriba, y más altos a quien la tenga en el centro. La Neurona 3 asigna

valores más bajos a las imágenes con mayor concentración de píxeles arriba a la izquierda, mientras que más altos a quienes se concentran más a la derecha, aquí se puede ver como la clasificación no es tan exacta teniendo mayor grado de imprecisión (varios valores menores en módulo a $-0,5$). La Neurona 4, asigna valor más bajo a las imágenes con más de una línea y mayor a las que tienen menos líneas. Por último, la Neurona 5 se comporta como la neurona 3, pero pondera distinto según las líneas horizontales.

1.2. Imágenes más grandes

Se analizaron casos donde las imágenes constaban de más píxeles, pero dados los tiempos de procesamiento y la búsqueda de los valores óptimos se analizará el caso concreto de 5 píxeles X 5 píxeles. Aquí podemos encontrar que se necesitan 9 neuronas en la capa oculta para aprender perfectamente el problema, y se vuelve a apreciar el hecho de que con 8 neuronas no aprende el caso de la imagen vacía. Creemos que la razón de esto se haya en el hecho de que hay más píxeles en blanco que en negro, por lo que el peso de un pixel en negro es ponderado más que un pixel en blanco dado que transmite más información. Luego, la imagen vacía, o toda en blanco, tiene poca información que hace que la red esté indecisa y falle. Esto es esperable dado que siempre es mejor entrenar a la red neuronal con casos parejos entre los valores de la entrada, misma cantidad de negros y de blancos entre el total de todas las imágenes.

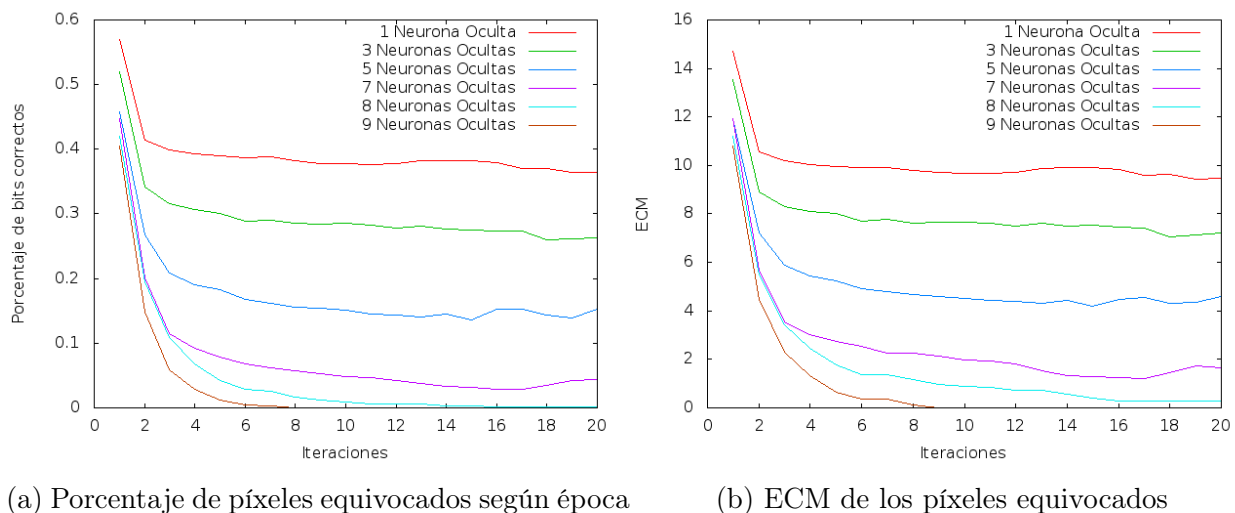
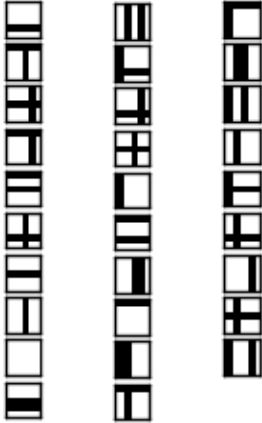


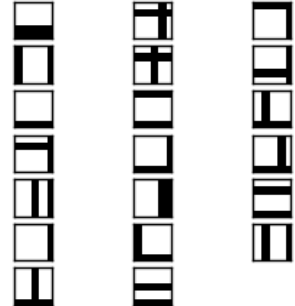


Figura 2: Entrenamiento con distinta cantidad de neuronas en la capa oculta

Aquí nuevamente se intentó, esta vez con menos éxito, analizar el tipo de clasificación que hace cada neurona. Dada la complejidad de esta tarea por la cantidad de imágenes posibles, solo haremos dos pequeñas menciones. Aquí nuevamente se observa que el clustering con dos clases falla al determinar unívocamente cada entrada, sin embargo el clustering de 4 categorías funciona perfectamente. Es importante comentar que nuevamente los valores de salida se volcaron hacia los extremos, módulo mayor a 0,5, lo que refuerza las bondades de usar estos autoencoders. Por lo dicho antes, solo mostraremos lo que ocurre con solo una

neurona de la capa oculta. Como se puede ver en la imagen, los valores más bajos corresponden a imágenes con más concentración de píxeles negros en el centro. Los más altos, los que menos píxeles en la mitad tienen.

$-1 < n < -0,5$	$-0,5 < n < 0$	$0 < n < 0,5$	$0,5 < n < 1$
			

2. Predicción tendencia en series temporales

Un problema típico para las redes neuronales es la predicción de series temporales. Mediante información sobre la tendencia en puntos anteriores, con una red neuronal, es posible predecir que valores o características tendrá una serie temporal en los siguientes puntos.

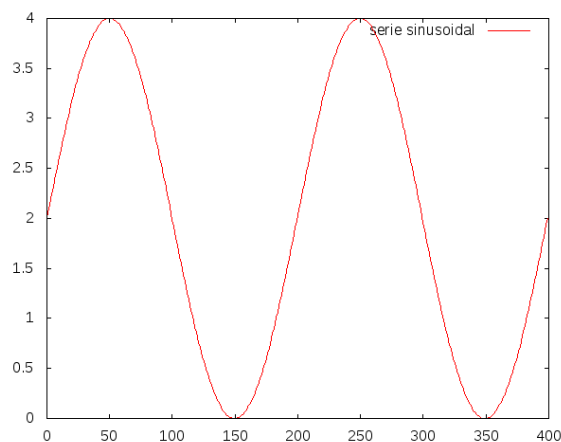
En esta parte de la práctica, deberemos realizar un sistema basado en un perceptrón multicapa que sea capaz de predecir la tendencia, ascendente o descendente, de los siguientes puntos de una serie temporal basándose en datos pasados de la serie.

Realizaremos una función, `adapta-serie-temporal`, que nos traducirá, dado un argumento llamado `np`, la serie temporal en un fichero que nuestra red pueda interpretar. Dado ese fichero, realizaremos predicciones sobre la serie temporal.

Ahora realizaremos un estudio del funcionamiento de las predicciones con distintos parámetros (valores de `np`, neuronas ocultas, y porcentaje de la serie en training), para dos series diferentes: una primera serie sinusoidal, y otra caótica.

2.1. Predicción sobre serie sinusoidal

En esta sección estudiaremos el comportamiento de las predicciones sobre esta serie. La serie en concreto, es la siguiente:



Como podemos apreciar, es una serie sinusoidal, un seno en concreto. Al realizar un comportamiento monótono creciente, hasta llegar a un punto de monotonía descendente, importará en un principio el % que introduzcamos en training. Si metemos todo el comportamiento, podrá en un principio predecir todo fácilmente, mientras que si metemos una sola parte, tendrá problemas.

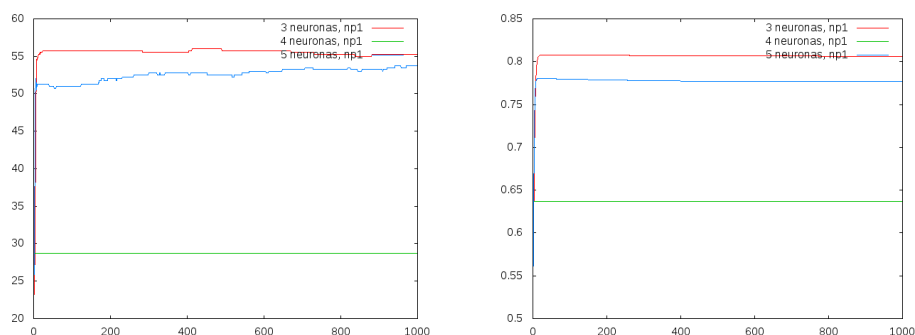
En concreto, dados sus 400 valores, con 100 no llega a capturar todo su comportamiento. Sin embargo, mostraremos que, con una cantidad suficiente de puntos previos, `np`, podrá llegar a predecir con bastante exactitud el comportamiento. Esto, es debido a que llega a predecir,

que si una parte de sus valores es descendente, y otra parte ascendente, tiene que cambiar el valor de la predicción. Esto sólo será posible con valores de np elevados, lo cuál mostraremos a continuación.

Primero, realizaremos un estudio de, según cuántas neuronas, con $np = 1, 2$ y 5 , su porcentaje de error y ECM, para ver el tamaño en neuronas adecuado para la serie. Probaremos tamaño de capa oculta de $2, 3$ y 5 .

Primero observemos el comportamiento con 100 elementos de entrenamiento.

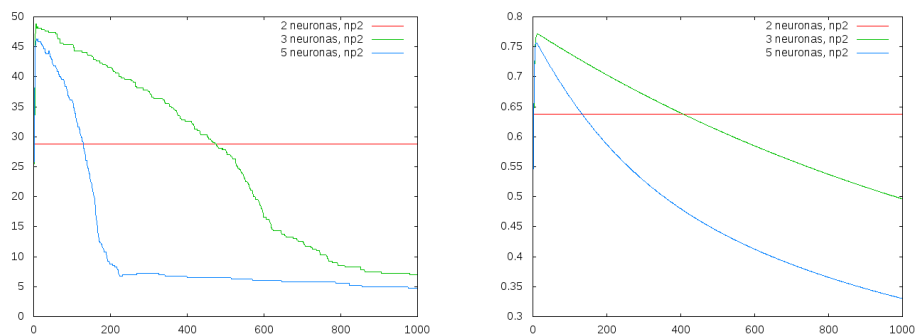
Empezaremos a mirar $np=1$. Con este tamaño, no debe ser posible con 2 neuronas, probaremos con $3, 4$ y 5 para obtener mayor información posible. Observemos los gráficos de error y ecm:



Podemos ver, que el error y el ECM son constantes, por lo que no hay entrenamiento, además de ser altos.

En el test, estos valores dan un resultado constante de 50.5 , siendo esto un resultado pésimo, dando de conclusión que $np1$ no es un tamaño adecuado. No es posible la predicción con un solo valor anterior.

Ahora, veamos con $np=2$:



Podemos ver que ahora si existe cierto entrenamiento, dado que la gráfica de ECM en 3 y 5 neuronas, tiende a descender, significa que va mejorando.

Llega a un mínimo de error en el caso de 5 neuronas sobre la época 200, mientras que llega en el caso de 3 sobre la 800-900.

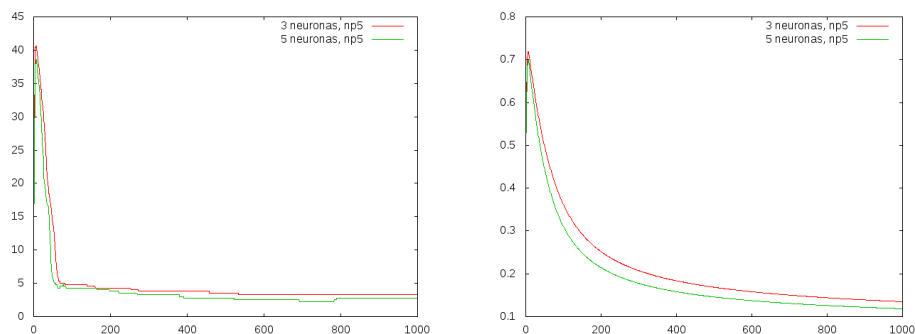
Esto sugiere un mejor comportamiento en este problema con 5 neuronas. Iremos a partir de ahora fijandonos de manera especial en el caso de 5 neuronas.

Ahora veamos el error en el test:

Neuronas	Error
2	46.7337 %
3	59.799 %
5	79.8995 %

Se puede apreciar que el caso con 5 neuronas, es bastante más preciso que el resto. Debido a la mayor capacidad de memoria interna, muy probablemente, al tener mas pesos.

Veamos el caso de np=5, esta vez solo con 3 y 5 neuronas, dado que 2 se ha demostrado muy inferior, de hecho, sin capacidad para aprender:



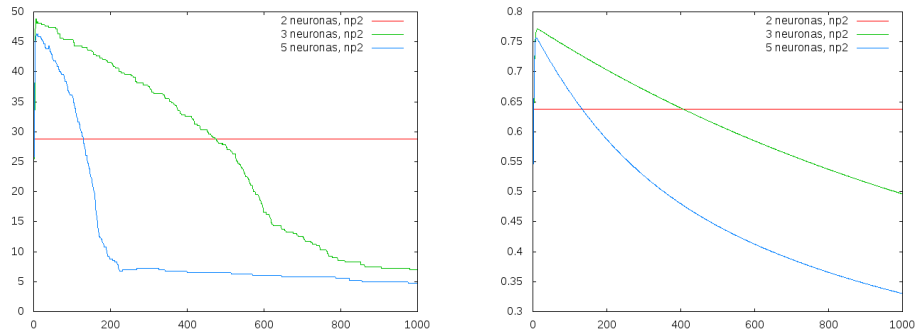
Aquí podemos ver ambos casos con muy buen resultado en entrenamiento, lo que significa que ya aprende correctamente, en ambos. Además, ambas redes aprenden al mismo tiempo prácticamente.

Observemos el resultado del test:

Neuronas	Error
3	93.9394 %
5	96.9697 %

Viendo esto, podemos afirmar que el tamaño de la capa oculta óptimo, empíricamente hablando, es de 5, y np óptimo, de 5 también.

Ahora observemos el comportamiento con 200 elementos en entrenamiento:



Podemos, en las gráficas que vemos arriba, observar el comportamiento de el error y error cuadrático medio, en la etapa de entrenamiento, para 2, 3 y 5 neuronas, durante 1000 épocas de entrenamiento, con $np = 2$.

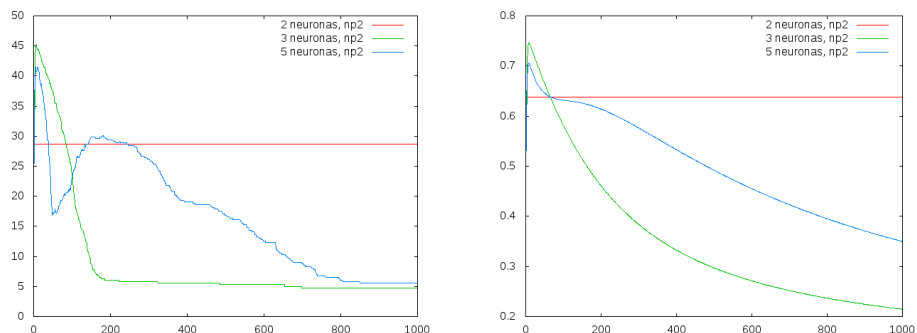
Podemos observar que en el caso de 2 neuronas, tiene un entrenamiento plano, es decir, no llega a aprender. Con 3 y 5 neuronas, se puede apreciar que aprende, llegando a convergencia en errores con 5 neuronas en la época 200, y con 3 sobre la época 900.

Observemos ahora el error que resulta de aplicar al test de las otras 200 instancias no usadas en entrenamiento:

Neuronas	Error
2	46.733 %
3	59.799 %
5	78.8995 %

Como podemos ver, en ninguno de los casos llega a un número de errores similar al test. Esto es, pese a tener la mitad, debido a que no tiene suficientes elementos en la serie.

Realizaremos ahora las mismas pruebas con $np=3$



En este caso, el tamaño óptimo en número de neuronas es 3, dado que alcanza un valor mejor y antes, en error y ecm. El caso de 2 neuronas, sigue siendo totalmente plano, sin llegar a aprender en ningún momento.

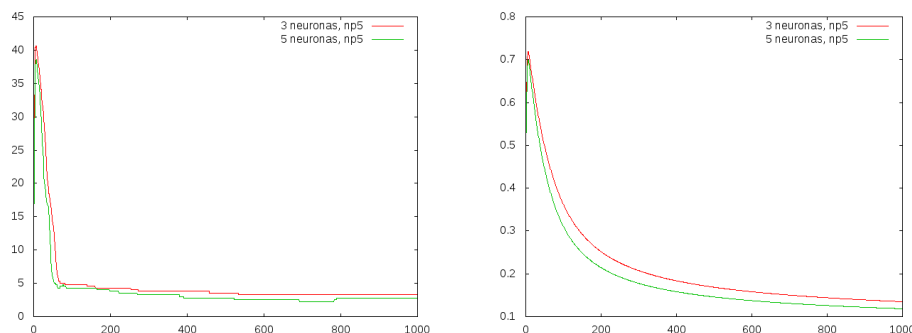
La convergencia se llega en el caso de 3 neuronas, en la época 180, y de 5 en la época 800.

Estos casos de enetrenamiento son mejores que los de $np=2$. Sin embargo, observemos los resultados del test:

Neuronas	Error
2	42.7237 %
3	85.9296 %
5	79.8995 %

El caso de 5 neuronas y 2 no ha cambiado. Sin embargo, existe una mejoría clara en el caso de 3 neuronas.

Observemos ahora el comportamiento con $np=5$. Para esta vez, ya que hemos visto el pésimo comportamiento de 2 neuronas, descartaremos este caso:



Ahora vemos, que ambas se comportan de manera similar, llegando a una convergencia prácticamente en la época 80.

Para poder observar mejor cuál es el número óptimo, deberemos ver ahora el caso de test:

Neuronas	Error
3	93.9394 %
5	96.9697 %

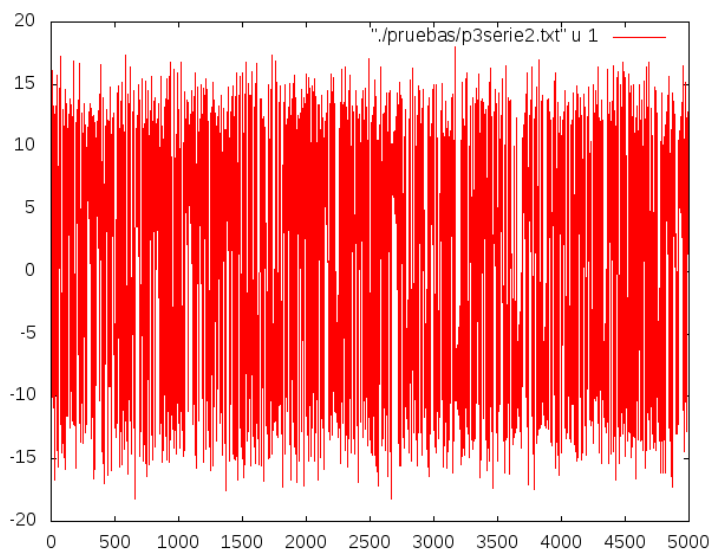
Aquí, con $np=5$, podemos observar que el perceptrón ya es capaz de conseguir predecir la monotonía futura de la función.

Comparando ambos casos, usando 100 y 200 para entrenamiento, hemos visto que con un número grande de neuronas aprende igual, y consigue un resultado identico en test.

Esto es debido a que lo que el perceptrón analiza, no es la forma que ve, sino la monotonía, y sabe que para cambiar su monotonía, debe haber un cambio en la monotonía de la serie.

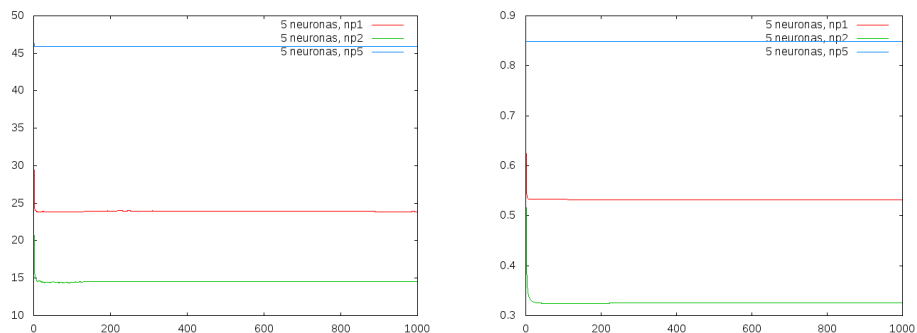
2.2. Serie Caótica

En este apartado, realizaremos un análisis similar al anterior, sobre una serie caótica dada. La serie es la siguiente:

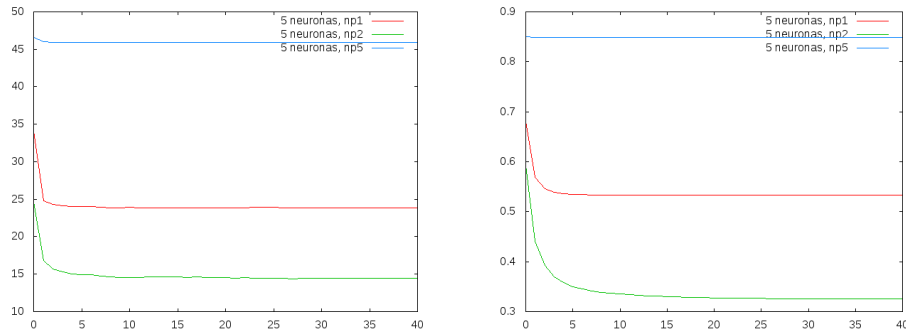


Podemos ver que no tiene un patrón fijo de subidas y bajadas. Sus valores tienen una gran variabilidad. Sin embargo, pese a que predecir esta red debe ser difícil, predecir su monotonía, no es complejo. Además, por la variabilidad de valores, el resultado usando partición del 25 % y del 50 %, debe ser similar. Por ello, haremos la predicción usando partición del 25 %. Al final demostraremos con un ejemplo esto.

Probaremos usando 5 neuronas, dado que en la anterior sección vimos que era el mejor caso, para ver cuál es el mejor np para este problema (numero de muestras anteriores). Una vez hallado el número de muestras idóneo, probaremos distinto tamaño de capa oculta.



Se puede ver que alcanza muy pronto a una convergencia, por lo que ampliaré a sólo las 40 primeras épocas.



Podemos ver que converge enseguida, debido al gran tamaño de los datos, siendo en el fichero 5000, un 25 % es 1250, lo cuál es bastante

Los valores salidos del entrenamiento son muy similares a los obtenidos con el test, dado que, al ser caótica, no tiene patrones que haya que cubrir, y siguen un mismo patrón (caótico), durante toda la serie.

Los resultados del test son los siguientes:

Neuronas	Error
1	76.1423 %
2	84.4738 %
5	57.1657 %

En este caso, podemos ver que el np idóneo para este problema, es 2, dentro de los propuestos. Esto es debido, a que el problema cuántos más datos tenga, si no aumenta el número de neuronas, generaliza más, siendo la solución a este tipo de proble muy simple, similar al gradiente.

2.3. Conclusión

Las redes neuronales sirven bien para multitud de problemas, y este en concreto, predicción de tendencias, funciona muy bien.

Eso es debido a que el perceptrón es un aproximador universal, que puede aproximar problemas, conociendo las condiciones anteriores, y, por lo tanto, predecir con bastante confianza, la tendencia de monotonía de una serie. Pudo hasta aprender cosas que no ha visto (en la serie sinusoidal).