REDES NEURONALES
Practico 3
Red Feedforward
Perceptron Multicapas

Prof: F. A. Tamarit
Autor: Eric N. Jurio

20 de diciembre de 2012

Introducción

Este trabajo esta basado en la aplicación de redes neuronales feedforward multicapas a distintas situaciones planteadas.

Las redes neuronales Feedforward

La red neuronal feedforward fue el primer tipo y posiblemente el más simple de red neuronal artificial ideado. En esta red, la información se mueve en una sola dirección (hacia adelante), a partir de los nodos de entrada, a través de los nodos ocultos (si los hay) terminando en los nodos de salida. Como característica principal destacamos que es un red sin bucles de flujo de información.

Están inspiradas en redes neuronales biológicas (en particular, en el sistema nervioso de los animales).

Este tipo de redes son aplicada para aprendizaje de patrones, mapeo de funciones (control) y clasificación, principalmente. Una de las aplicaciones que tiene es para el aprendizaje de funciones con su dominio e imagen en espacios n-dimencionales de tipo binario, por eso estas redes son también conocidas como redes perceptron. Pero el modelo no limita solo a casos binarios, ya que se puede variar la función de activación de las neuronas (neuronas binarias, lineales, continuas derivables no lineales, etc).

El concepto de feedforward es amplio y no limita a la estructura interna del conexionado mas que en el requerimiento antes mencionado. Uno podría tener una red de neuronas en donde la salida de la neurona A se conecte con B y con C, pero a su vez, que la salida de B se conecte con C formando un ciclo no dirigido. Notar que esto no contradice lo anterior, ya que la información sigue viajando hacia adelante.

Lo anterior es solo un ejemplo, por lo general las redes se estructuran en capas de neuronas y las neuronas de una capa solo se conectan con las neuronas de las capa siguiente (ie, las conexiones entre neuronas no saltan capas) y pueden o no estar totalmente conectadas las capas de neuronas (todos los pares de neuronas entre 2 capas dadas). Lo más común es hablar de redes perceptron simple (de una capa de neuronas) o perceptron multicapas (capas sucesivas de neuronas). Una red perceptron simple solo tiene la capacidad de aprender funciones que sean linealmente separable, ie una perceptron simple no tiene la capacidad de aprender las función XOR, por ejemplo. Rumelhart y otros autores, en 1986, presentan la "Regla Delta Generalizada" para adaptar los pesos propagando los errores hacia atrás, es decir, propagar los errores hacia las capas ocultas inferiores. De esta forma se consigue trabajar con múltiples capas y con funciones de activación no lineales. Se demuestra que el perceptron multicapa es un aproximador universal. Un perceptron multicapa puede aproximar relaciones no lineales entre los datos de entrada y salida.

Uno de los algoritmos que se pueden utilizar para entrenar este tipo de redes, es el algoritmo backpropagation, el cual esta basado en el concepto de descenso

por el gradiente (minimizar una función objetivo), esto lo hace modificando los pesos sinápticos entre las neuronas.

Objetivos

En este informe analizaremos varios aspectos de las redes Feedforward Perceptron. Para ello se plantean los siguientes ejercicios:

1. Compararemos el error de generalización promedio de redes perceptron (de distintas cantidades de neuronas de entrada, como de neuronas en la capa oculta) en función del tamaño de los conjuntos de entrenamiento, usando como función a aprender, la función paridad.

2. Analizaremos cualitativamente y de manera gráfica, resultados de aproximar dos funciones continuas ($\frac{1}{x}$ y $\log(x)$) por medio de redes neuronales (de estructura prefijadas para cada función respectivamente) entrenadas con distintos tamaños de conjuntos de entrenamientos.

3. Analizaremos el problema de codificación sobre las capas ocultas de un par de redes neuronales de iguales cantidades de entradas y salidas pero distinto numero de neuronas en sus capas ocultas. En ambas redes mapearemos en si mismas entradas ortogonales y compararemos los resultados de la codificación en sus respectivas capas ocultas.

Detalles de la implementación

He desarrollado mi propia librería open source para el modelo perceptron (entre otros modelos que continuo desarrollando).

No voy a entrar en detalles, algunas características implementadas son:

- Implementación de redes feedforward simple y multicapas. (definición de numero de capas, tipo de neurona en la capa y numero de neuronas por capa al momento de construcción de la red)

- Aprendizaje de conjuntos de entrenamientos en modo batch u online.

- Aprendizaje con momento. (momento por defecto de 0,5 o seleccionable en $[0{,}0; 1{,}0]$)

- Parámetro de adaptación auto-regulado en un rango preseleccionado (rango por defecto $[0{,}02; 0{,}05]$; rango ajustable en $[0{,}0; 1{,}0]$)

Para más información de las características y del proceso de desarrollo ver en la pagina del proyecto:

El proyecto (en general) se encuentra en http://bettercodes.org/projects/neural-network

La implementación en particular para este trabajo esta en http://code.google.com/p/redes-neuronales/ (utiliza librerías del proyecto recién mencionado)

Red Feedforward Perceptron Multicapas

Error de generalización promedio sobre la función Paridad

Implementé un programa de aprendizaje usando el algoritmo de backpropagation (batch) en a una red neuronal con una capa oculta de M neuronas que computa una función booleana de N entradas.

Compararemos el error de generalización promedio (dividido el numero total de ejemplos 2^N) en función de la fracción de ejemplos utilizados en el entrenamiento $\frac{p}{2^N}$, donde el promedio se calcula sobre una muestra de L conjuntos diferentes de entrenamiento de p ejemplos, tomados aleatoriamente con igual probabilidad; teniendo en cuenta en el calculo del promedio solamente aquellos casos en que el error de aprendizaje es inferior a una tolerancia que garantice la salida correcta para todos los patrones del conjunto de entrenamiento. Tomamos $M = N$ par $N = 4; 6; 8$ y $N = 8$ aplicando aprendizaje con momento, tomando $L = 5000$.

A continuación presentamos la gráfica resultante.

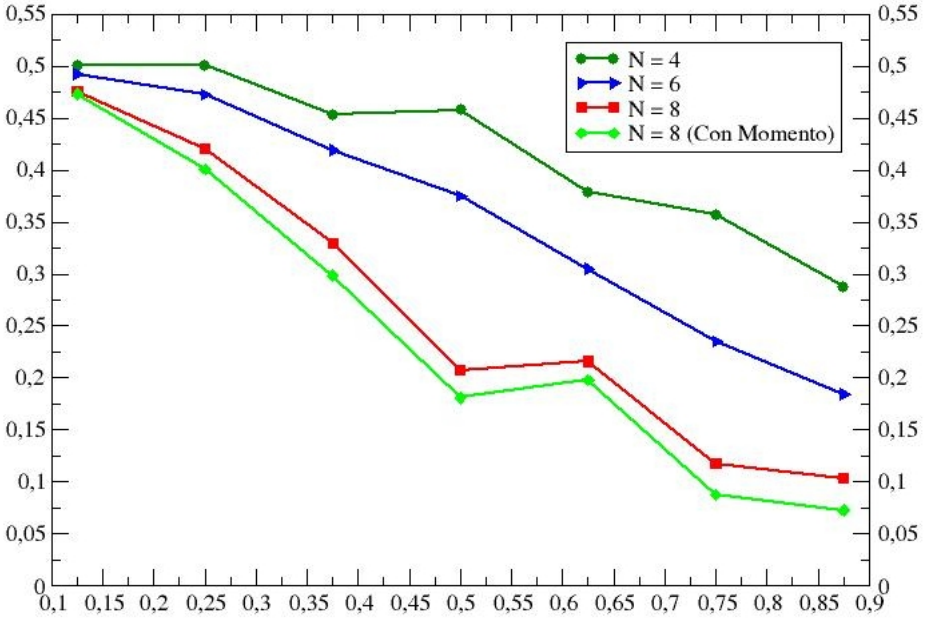

Figura 0.1: Error de generalización promedio respecto de la fracción de ejemplos utilizados en el entrenamiento $\frac{p}{2^N}$.

En todas las capas, todas las neuronas usan como función de activación la $\tanh(x)$. Usamos como función base a aprender, la función paridad, la cual tiene como característica que para entradas de datos que varían en un solo bit da resultados opuestos, lo cual representa una dificultad para las redes neuronales ya que son muy tolerantes a pequeños cambios, haciendo de esta función una de las más difíciles de aprender por redes neuronales.

De la Figura 0.1 podemos observar lo siguiente:

- A medida que aumenta la cantidad de entradas (aumenta N) también aumenta cuantitativamente el numero de elementos en los conjuntos de entrenamiento para una proporción dada y vemos que las curvas de error están desplazadas cada ves más cerca de 0.
- También podemos observar en el gráfico, que en todos los casos a mayor porcentaje cubierto por el conjunto de datos de entrenamiento, menor es el error general de la red.
- Respecto a la red de 8 entradas, entrenada con momento, vemos que el error de generalización esta por debajo de todos los otros, inclusive de la red de 8 entradas entrenada sin aplicara momento, por lo que es una técnica a considerar siempre.

Aproximación de funciones continuas

Implementé un programa de aprendizaje usando el algoritmo de backpropagation (on-line) en una red neuronal con una capa oculta de M neuronas para aproximar funciones continuas de una variable $f(x)$. Como función de activación para las neuronas de la capa oculta usamos la función $\tanh(x)$ y para la neurona de la capa de salida usamos la función lineal: $OutNet = \sum_{i=1}^{M}(outHide_i * w_i)$ donde OutNet es la salida de la neurona de salida (lineal), $outHide_i$ son las salidas de las neuronas de la capa oculta y w_i son los pesos sinápticos de entrada a la neurona lineal.

Entrenamos sobre p puntos de la función tomados aleatoriamente en un intervalo finito de la variable independiente x. Imponiendo un limite de tolerancia de error de aprendizaje por patrón . Aproximamos las siguientes funciones en el intervalo $x \in [1; 5]$:

- $f(x) = \frac{1}{X}$ con $M = 4$, $\epsilon = 0{,}01$ para $p = 5; 10; 20$.
- $f(x) = \ln(x)$ con $M = 6$, $\epsilon = 0{,}02$ para $p = 5; 10; 20$.

A continuación podemos ver las gráficas simultáneas de la función a aproximar, la función de ajuste aprendida por la red y los puntos utilizados para obtener dicha aproximación.

Aproximación de $f(x) = \frac{1}{X}$

A continuación las Figuras 0.2, cuatro curvas superpuestas. ($f(x)$ y sus 3 aproximaciones)

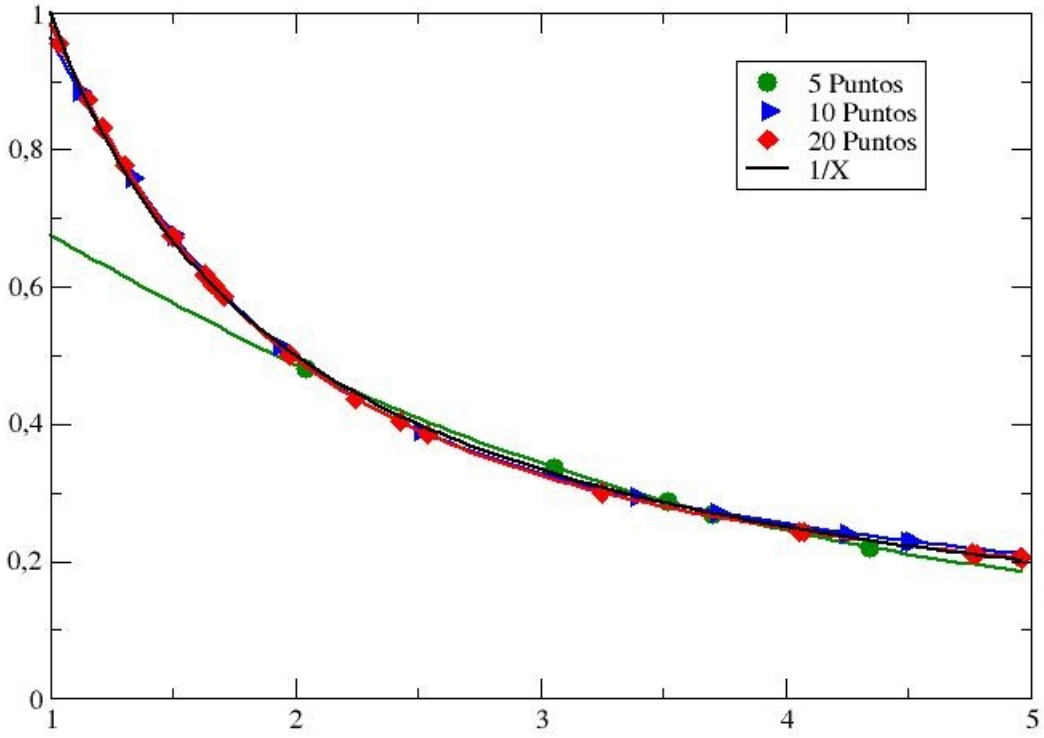

Figura 0.2: Aproximación de $f(x) = \frac{1}{X}$ por redes feedforward de una capa oculta, entrenadas con $5; 10; 20$ puntos

Notamos que en general las aproximaciones son buenas, excepto la aproximación hecha solo por 5 puntos de entrenamientos. En este caso vemos que la interpolación es buena, pero la extrapolación no lo es. Lo apreciamos con más detalle en las Figuras 0.3 y 0.4

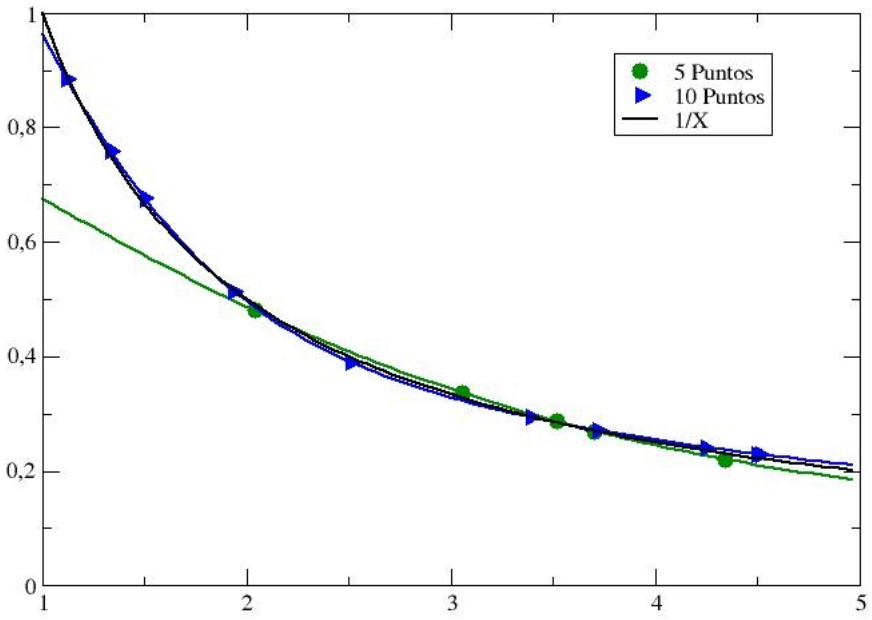

Figura 0.3: $f(x) = \frac{1}{X}$, aproximación por redes entrenadas con 5 y 10 puntos

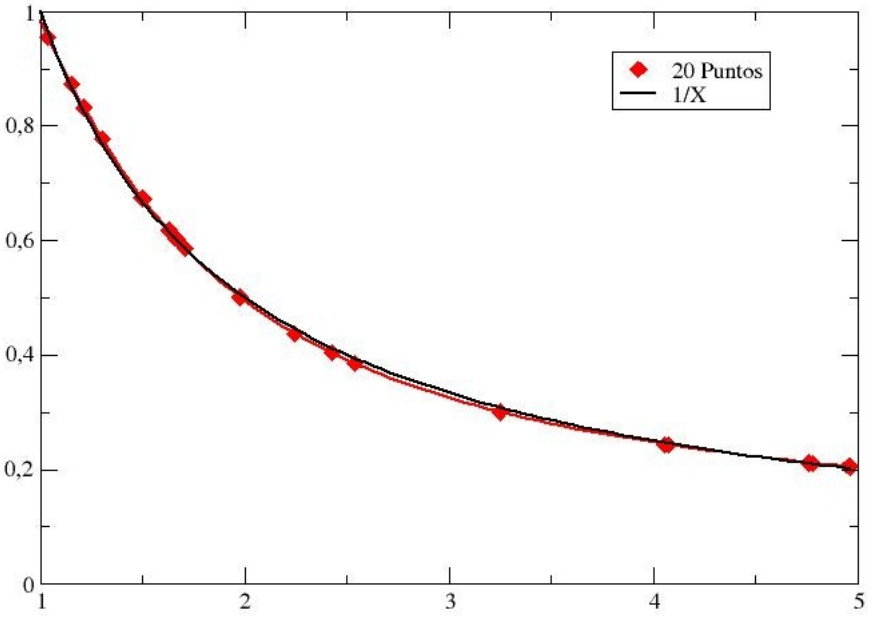

Figura 0.4: $f(x) = \frac{1}{X}$, aproximación por red entrenada con 20 puntos

Aproximación de $f(x) = \ln(x)$

A continuación las Figuras 0.5, cuatro curvas superpuestas. ($f(x)$ y sus 3 aproximaciones)

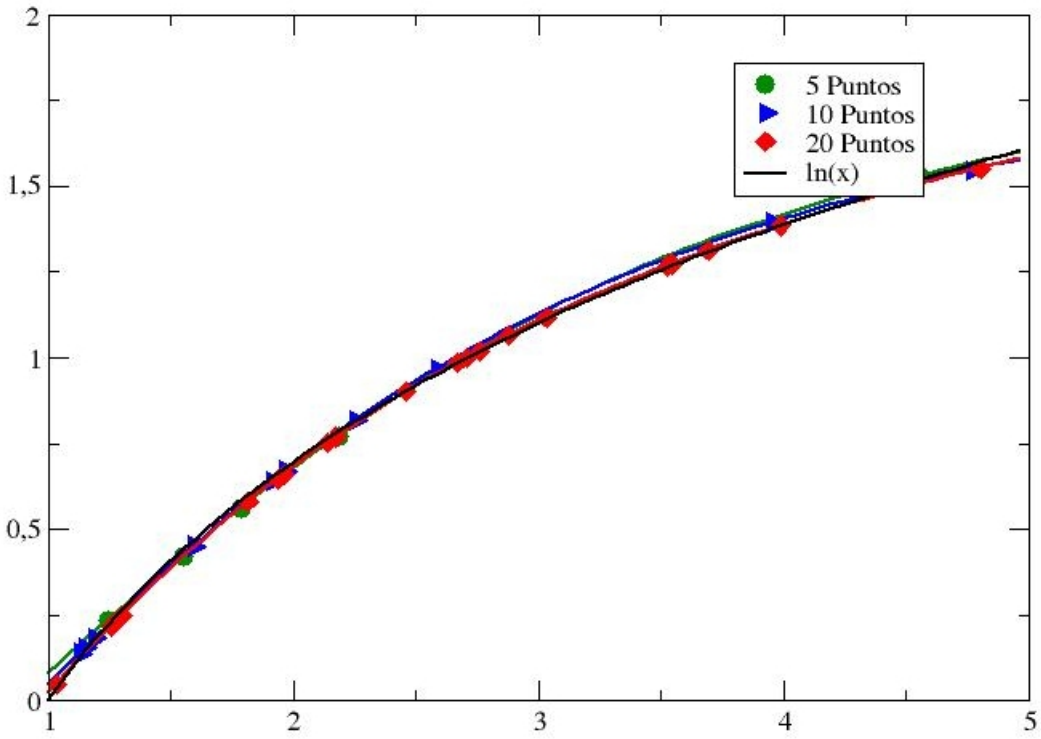

Figura 0.5: Aproximación de $f(x) = \ln(x)$ por redes feedforward de una capa oculta, entrenadas con $5; 10; 20$ puntos

Podemos ver que la aproximación es buena cerca de los puntos entrenados y el error es más grande al alejarse. Apreciamos con más detalle en las Figuras 0.6 y 0.7

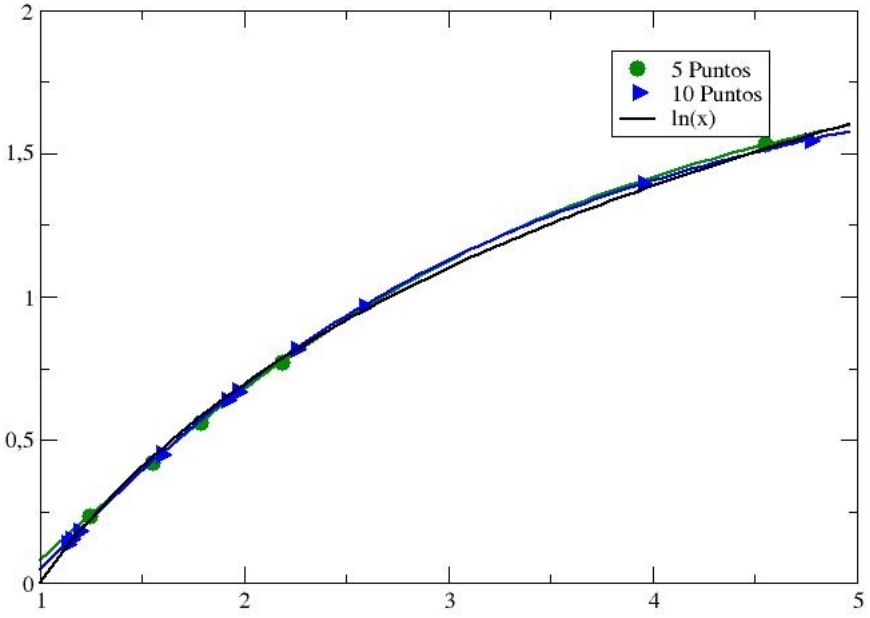

Figura 0.6: $f(x) = \ln(x)$, aproximación por redes entrenadas con 5 y 10 puntos

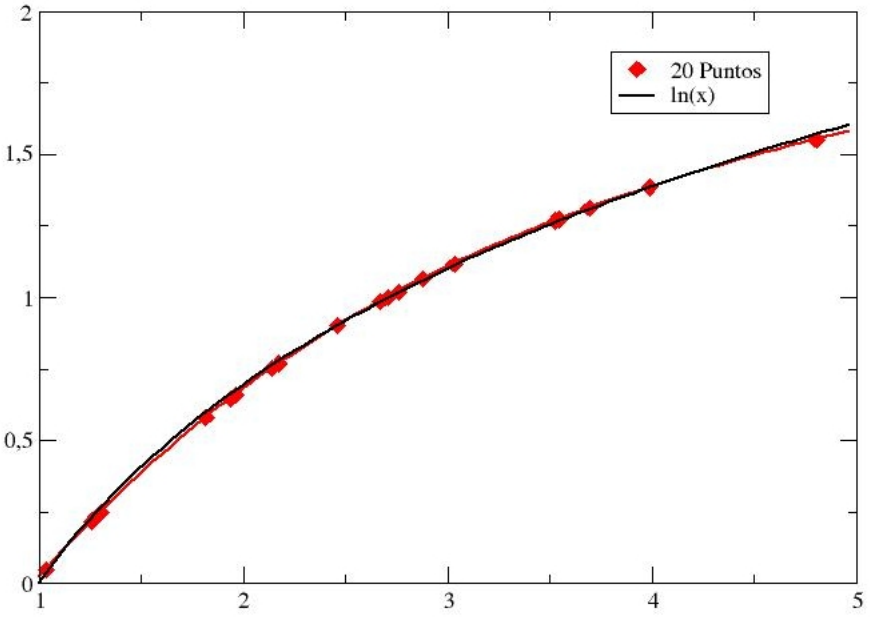

Figura 0.7: $f(x) = \ln(x)$, aproximación por red entrenada con 20 puntos

Codificación y Mapeo sobre Redes Perceptron

Analizamos el problema de codificación sobre las capas ocultas de un par de redes de iguales cantidades de entradas y salidas (8 neuronas de entra y salida), pero distinto numero de neuronas en sus capas ocultas (3 neuronas en una y 5 neuronas en la otra). En ambas redes mapeamos en si mismas entradas ortogonales (8 (2^3) arreglos ortogonales entre ellos) imponiendo como error máximo $\epsilon = 0{,}02$.

A continuación vemos los pesos sinápticos que conforman la red 8-5-8.

	In1	In2	In3	In4	In5	In6	In7	In8	Umbral
Out1	1.33	-0.86	-0.63	0.49	0.59	-0.99	-0.95	1.39	0.05
Out2	0.8	-1.14	1.31	-1.29	0.85	-1.13	1.18	-0.5	0.03
Out3	-1.22	0.95	-1.33	-0.29	0.67	-1.28	1.08	1.12	-0.04
Out4	-0.7	0.17	0.48	-0.96	-1.2	0.5	0.58	0.39	-0.15
Out5	0.15	0.31	0.5	0.39	0.41	0.55	0.59	0.53	0.62

	In1	In2	In3	In4	In5	Umbral
Out1	1.58	1.5	-2.77	-1.73	1.78	2.1
Out2	-2.3	-2.58	2.54	-0.71	2.14	2.09
Out3	-2.15	2.56	-2.59	0.36	2.2	2.77
Out4	1.06	-2.93	-0.85	-2.59	1.22	1.93
Out5	0.38	1.17	2.21	-2.69	1.33	1.91
Out6	-2.23	-2.53	-2.59	-0.20	2.12	2.48
Out7	-2.23	2.5	2.51	0.51	2.43	2.74
Out8	2.73	-1.64	2.23	0.62	1.98	2.48

A continuación vemos los pesos sinápticos que conforman la red 8-3-8.

	In1	In2	In3	In4	In5	In6	In7	In8	Umbral
Out1	0.86	0.24	-1.13	1.09	-0.09	0.76	0.9	-3.67	-0.18
Out2	-2.51	3.47	-0.8	0.09	3.1	-1.91	0.42	-1.73	0
Out3	-1.18	1.15	-2.36	3.04	0.46	0.99	-2.41	1.52	0.23

	In1	In2	In3	Umbral
Out1	2.79	-9.84	-3.7	13.6
Out2	-6.76	11.9	5.16	6.68
Out3	-9.65	-1.34	-12.7	7.48
Out4	2.95	-1.08	12.3	12.8
Out5	-6.61	14.2	-3.28	8.18
Out6	3.16	-6.76	3.29	9.06
Out7	0.06	3.09	-11.7	11.2
Out8	-7.09	-1.48	6.9	4.82

Al comparar las matrices de la red 8-3-8 y 8-5-8 podemos observar que los valores que aparecen en las matrices 8-3-8 tiene máximos y mínimos notablemente superiores a los valores máximos y mínimos de la otra red. En la red 8-5-8 los valores de los pesos sinápticos son mucho más homogéneos.

También notamos diferencias en la cantidad de iteraciones necesarias para lograr el aprendizaje:

- Para entrenar la red 8-5-8, se necesitaron 33 iteraciones de aprendizaje.
- Para entrenar la red 8-3-8, se necesitaron 70 iteraciones de aprendizaje.

Conclusiones

1. Sobre el error de generalización.
    - A medida que aumenta la cantidad elementos en el conjunto de entrenamiento vemos que el error se acerca más a 0.[1]
    - (para el caso finito) A mayor porcentaje cubierto de las combinaciones de datos que definen la función a mapear por el conjunto de datos de entrenamiento, menor es el error general de la red.
    - El aprendizaje aplicando momento, resuelve algunos problemas de rebote y de aprendizaje lento, disminuyendo también el error de generalización.
2. Para el caso de aproximar funciones continuas, notamos que en general las aproximaciones son buenas siempre y cuando los datos del conjunto de entrenamiento, estén bien distribuidos en el espacio objetivo que queremos representar. En este caso vemos que la interpolación es buena pero la extrapolación no siempre lo es.

[1] Algo estudiado, pero que no vimos en el experimento es que en algunos casos, cuando la arquitectura de la red o la/s función/es de activación de las neuronas no son apropiadas, se puede dar el efecto de sobre entrenamiento, en donde la red aprende bien todos los valores de entrenamiento, pero no interpola ni extrapola bien.

3. Cuando la cantidad de neuronas son apenas las suficientes para almacenar la información requerida[2], podemos afirmar 2 cosas:

    - El entrenamiento tomara más tiempo (iteraciones de entrenamiento) que si tuviese más neuronas.[3]
    - La red puede tener pesos sinápticos muy dispares y con valores absolutos muy grandes, lo que la puede hacer poco adaptable a cambios frente futuros entrenamientos. En redes con más capacidad de información (acorde a lo que se le quiera mapear), los valores de los pesos sinápticos son mucho más homogéneos.

[2]Si no tiene la cantidad necesarias de neuronas para almacenar la información es de esperar que tome muchísimo más tiempo de entrenamiento y quizás nunca alcance el valor esperado, dentro del rango establecido por el error aceptable para los valores de entrenamiento.

[3]Si las neuronas son demasiadas, el entrenamiento también tardara mucho en cuanto a tiempo de calculo ya que las matrices de pesos sinápticos serán más grandes.