Sistemas de Inteligencia Artificial

Redes Neuronales

Informe TP2

Grupo 7:
Nicolás Buchhalter
Francisco Depascuali
Agustina Fainguersch

1. Introducción

A lo largo de este TP se implementó un perceptrón multicapa con la intención de enseñarle la función propuesta por la cátedra. Se buscó además mejorar la performance del mismo a través de la implementación del algoritmo de backpropagation complementado con momentum y la utilización de un η adaptativo.

Luego de analizar el comportamiento básico del algoritmo se intentaron optimizar los resultados a partir de cambios en la arquitectura. Para esto se modificaron los valores iniciales de η, beta y momentum, así como también la cantidad de capas ocultas y de neuronas por capa oculta.

2. Notación

A lo largo de nuestro trabajo utilizaremos la notación [1 a b 1] para representar la arquitectura que tiene dos capas ocultas de "a" y "b" neuronas respectivamente. Por ejemplo [1 15 10 5 1] tiene tres capas ocultas de 15, 10 y 5 neuronas respectivamente.

3. El problema

La función propuesta por la cátedra para nuestro grupo fue la siguiente:

$$f(x) = sin(x) + 6\,cos^2(x) \text{ con } x \in [-15,\ 15]$$

4. Implementación del algoritmo

Construcción del perceptrón

El algoritmo propuesto busca construir el perceptrón recibiendo por un archivo de configuración los valores deseados según la arquitectura.

Los parámetros de entrada de la función son:
Los patrones de aprendizaje, un array con la lista de la cantidad de neuronas por capa, el η inicial, la función de activación a utilizar, la constante de dicha función, el valor del error deseado, dos variables que permiten activar o desactivar el momentum y η adaptativo y fijar sus respectivos valores, y las constantes "a" y "b" del η adaptativo.

Dentro de la implementación del perceptrón multicapa, se utiliza una función que calcula, con valores aleatorios entre -0,5 y 0,5, todas las matrices de pesos iniciales según la cantidad de neuronas y capas que se tienen en dicha arquitectura.

Para realizar el aprendizaje se itera sobre todos los conjuntos de entrenamiento, y se los pasa por la red neuronal con los pesos actuales. Luego se actualizan los pesos de manera incremental, haciendo backpropagation. Se arranca con un valor derivado de la diferencia entre la salida obtenida y la deseada. Luego, para calcular los valores de salida actual, se fijan los pesos y se ejecutan todos los patrones de entrada. Finalmente, estos valores son utilizados para calcular el error y graficar.

Error utilizado

Se utilizó el error cuadrático medio para evaluar los resultados y actualizar los pesos luego de cada época.

Funciones de activación

Se utilizaron dos posibles funciones de activación. Estas pueden ser intercambiadas a través del parámetro de entrada gValue. A partir de ahora nos referiremos a ellas con los nombres g1 y g2:

g1: La tangente hiperbólica: $g(h) = \tanh(\beta h)$ para unidades entre - 1 y 1.

g2: La función exponencial: $g(h) = 1/(1 + \exp(-2\beta h))$ para unidades entre 0 y 1.

5.Selección del conjunto de entrenamiento

Se utilizaron dos conjuntos de entrenamiento distintos. El primero intentó entrenar al perceptrón con todos los valores de la función entre -15 y 15 con 0.1 como paso de separación. Al ver que el perceptrón no aprendía del todo bien en el rango de -10 a -15 y de 10 a 15 se decidió cambiar a un conjunto de entrenamiento que tuviera mayor densidad de puntos en los extremos que en el centro. Es decir, tomamos valores separado por 0.01 entre -15 y -12.5, y entre 12.5 y 15, también tomamos valores entre -12.5 y -5 y entre 5 y 12.5 separados por 0.05 y finalmente entre -5 y 5 tomamos valores separados por 0.1. De esta forma se logró un mejor aprendizaje en el rango requerido. Se debe notar además que los patrones de entrenamiento se ejecutaron de forma aleatoria en cada época para mejorar el aprendizaje.

6. Selección de parámetros

Se utilizaron los siguientes valores:

- Por recomendación de la cátedra se utilizaron los siguientes β: 0.5, 0.75, 1.
- Para asegurarnos de que el perceptrón llegue a la solución, se utilizaron valores pequeños de eta dependiendo del intervalo. η: 0.01, 0.001, 0.0001

7. Optimizaciones en la implementación del perceptrón

Momentum: Se eligieron 0.5 y 0.9 para los valores de α, utilizando la siguiente fórmula:

$$\Delta w_{pq}(t+1) = -\eta \frac{\partial E}{\partial w_{pq}} + \alpha \Delta w_{pq}(t)$$

η Adaptativo: Se utiliza un valor k, pasado por parámetro, para decidir si se debe aumentar o reducir el η según corresponda.

$$\Delta\eta = \begin{cases} +a & si \;\; \Delta E < 0 \;\; consistentemente \\ -b\eta & si \;\; \Delta E > 0 \\ 0 & en \; otro \; caso \end{cases}$$

8. Análisis de resultados

Como se puede observar en la Figura 1 del Anexo, primero se analizó el intervalo -5:5 con el objetivo de diferenciar un mínimo de neuronas y capas necesarias ya que podía entrenarse al perceptrón en una cantidad reducida de épocas con el error deseado.

- **Una capa oculta:**

Los resultados muestran que con menos de 10 neuronas en la capa, ésta se satura ó el perceptrón queda atrapado en un mínimo local. Al aumentar la cantidad de neuronas, el perceptrón puede diferenciar mejor los patrones, y requiere menos épocas para llegar el error pedido. Alterando estos valores se logró encontrar una cantidad ideal de neuronas para resolver el problema (con una sola capa). Superando esta cantidad de neuronas, puede observarse en la figura 1 que la cantidad de épocas para llegar al error deseado se mantiene igual o aumenta. En la Figura 1 se puede observar a simple vista que los resultados obtenidos no son del todo deseables ya que tarda entre 900 y 2496 épocas en resolver el problema.

- **Dos capas ocultas:**

Para las arquitecturas con dos capas ocultas se notó que el resultado óptimo ocurre cuando η es 0.01 y β 0.5. Una vez obtenido un valor deseado, se fijó el momentum, el η y el β y se analizaron múltiples arquitecturas con 10 y 15 neuronas respectivamente en la primer capa oculta, para poder ir variando la cantidad de neuronas en la segunda capa oculta y así encontrar el valor óptimo.

Los resultados obtenidos con dos capas ocultas superaron ampliamente a los obtenidos con una sola: en general se llegó a entrenar el perceptrón en 400 épocas, cuando antes se necesitaron alrededor de 2000 épocas.

En la primer capa oculta, con:

- 10 neuronas: La mejor arquitectura resultó ser [1 10 6 1]
- 15 neuronas: La mejor arquitectura resultó ser [1 15 5 1]

Además, a partir de los resultados (dado que en el 90% de los casos la cantidad de épocas necesarias para aprender es mayor) se decidió proseguir los análisis utilizando solamente la función g1.

Como se puede observar en la Figura 4 del Anexo a partir de aquí se utilizó el intervalo [-15:0.1:15]. Para realizar estas tablas se tomaron diferentes arquitecturas arbitrarias, modificando la cantidad de capas y neuronas en cada una.

De acuerdo a los resultados de la Figura 4 se obtuvieron las tres arquitecturas con mejores resultados (menor cantidad de épocas para aprender con el error determinado) para realizar un análisis aún más profundo sobre ellas. Se modificaron los valores de β, η y momentum, y se registraron en la figura 5. Luego de analizar esta figura, la arquitectura que contiene [1 512 256 128 64 32 16 1] con β=0.75 y momentum 0.5 entrena al perceptrón en la menor cantidad de épocas.

Por otra parte, en la figura 6 se ve la diferencia de variar β, manteniendo la arquitectura y los parámetros. A diferencia de $\beta = 0.5$, con $\beta = 1$ el error varía considerablemente porque al aplicarle la función a valores de x cercanos la variación de la salida de la neurona resulta mayor que con 0.5.
En la Figura 7, se puede ver claramente que el al ser $\eta = 0.00005$ (muy pequeño), las variaciones en el error también resultan pequeñas. Esto se debe a que en cada iteración los pesos varían muy poco.
Al notar que al perceptrón le costaba más definir los patrones a los costados, en la Figura 8 se modificó la densidad de puntos, analizando el intervalo [-15 : 0.05 : -12.5] [-12.5 : 0.25 : -5] [-5 : 0.5 : 5] [5 : 0.25 : 12.5] [12.5 : 0.05 : 15]. Sin embargo, el resultado no se dió como esperábamos (al perceptrón le llevó una cantidad de épocas similar que con el intervalo con densidad uniforme).
Por último, se muestra en la Figura 9 el aprendizaje del perceptrón con y sin eta adaptativo. Con eta adaptativo se decidió terminar el algoritmo al pasar las 20000 épocas, dado que el error casi no variaba (Δe en el orden de 10^{-3}).

9. Conclusiones

En conclusión durante el proceso de construcción, entrenamiento y testeo del perceptrón, pudimos observar los siguientes aspectos:

Cuanto más grande es la arquitectura y al entrenar la red con gran cantidad de patrones (en nuestro caso en el intervalo [-15:0.1:15]), menor debe ser el η. Esto sucede porque si se tiene un η muy grande (por ejemplo 0.1 resulta grande para el intervalo [-15:0.1:15]), se obtienen muchos picos en el gráfico del error y no solo nunca va a aproximar bien sino que además no va a llegar a una solución. De lo contrario, si el η es chico va a bajar más lento el error pero de forma más directa.

Se deben utilizar varias capas para resolver funciones más complejas. Cada capa ayuda a poder resolver los puntos en las partes más radicales de la función.

Se deben modelar las capas ocultas del perceptrón como un triángulo, donde debe haber más neuronas en las capas de abajo y menos en las de arriba porque si las primeras son pocas se pueden saturar y comenzar a enviar el mismo resultado para las capas siguientes.

Existe un punto en el que agregar más neuronas a una capa no tiene efecto porque en esa capa el perceptrón ya es capaz de procesar todos los valores de entrada, por lo que más neuronas en esa capa solo resultaría en un mayor tiempo de procesamiento.

El η adaptativo hace al algoritmo más lento ya que descarta todas las iteraciones que aumentan el error y en nuestra función la mejor forma de llegar a un error bajo implica utilizar un η un poco más grande e ir teniendo pequeños picos de errores. El η adaptativo genera una curva de error con forma descendente pero con una pendiente muy cercana a cero.

El momentum ayuda en la mayoría de los casos si se usa un valor de alpha de 0,5 ya que de esta forma suma solo una porción del delta anterior y por lo tanto varía solo lo suficiente como para escapar a mínimos locales y converger a un resultado correcto.
Con valores de alpha elevados, el error puede llegar a variar considerablemente, de manera que no se arribe al error mínimo propuesto. Por otro lado, si alpha es muy cercano a cero, es más probable que el error terminé cayendo en los mínimos locales de todas formas.

Al calcular el error con la fórmula de error cuadrático medio entregada por la cátedra, llegamos a la conclusión de que el error depende de la cantidad de patrones que se estén evaluando. Ya que se calcula como la suma de los cuadrados de las diferencias, dividido 2. Como este 2 es una constante, cuanto más patrones haya para sumar diferencias, más aumentará el error. Por eso no es un buen parámetro al menos para comparar entre dos distintos conjuntos de patrones de entrenamiento iniciales.

Anexo

Tablas comparativas

#Capas Ocultas	Capas	η	β	Momentum	η Adaptativo	a	b	Error	Épocas-g1	Épocas-g2
	[1 5 1]	0.01	0.5	0	0	0	0	1	(+6500)	(+6500)
	[1 10 1]	0.01	0.5	0	0	0	0	1	1692	944
	[1 20 1]	0.01	0.5	0	0	0	0	1	2496	3249
1	[1 100 1]	0.001	1	0.5	0	0	0	1	5241	5541
	[1 10 5 1]	0.01	0.5	0	0	0	0	1	719	3630
	[1 10 5 1]	0.01	0.5	0.5	0	0	0	1	347	2729
	[1 10 6 1]	0.01	0.5	0.5	0	0	0	1	283	565
2	[1 10 8 1]	0.01	0.5	0.5	0	0	0	1	446	534
	[1 10 9 1]	0.01	0.5	0.5	0	0	0	1	459	1077
	[1 10 10 1]	0.01	0.5	0.5	0	0	0	1	467	1426
	[1 10 6 1]	0.01	0.5	0.5	0	0	0	1	298	1587
	[1 15 5 1]	0.01	0.5	0.5	0	0	0	1	243	1124
	[1 15 6 1]	0.01	0.5	0.5	0	0	0	1	497	911
	[1 15 8 1]	0.01	0.5	0.5	0	0	0	1	423	480
2	[1 15 10 1]	0.01	0.5	0.5	0	0	0	1	376	642
	[1 15 11 1]	0.01	0.5	0.5	0	0	0	1	299	858
	[1 15 12 1]	0.01	0.5	0.5	0	0	0	1	338	406
	[1 15 13 1]	0.01	0.5	0.5	0	0	0	1	297	714
	[1 15 14 1]	0.01	0.5	0.5	0	0	0	1	382	573

Figura 1: La tabla muestra la comparación en cantidad de épocas para las arquitecturas planteadas en el intervalo [-5: 5] para una y dos capas ocultas.

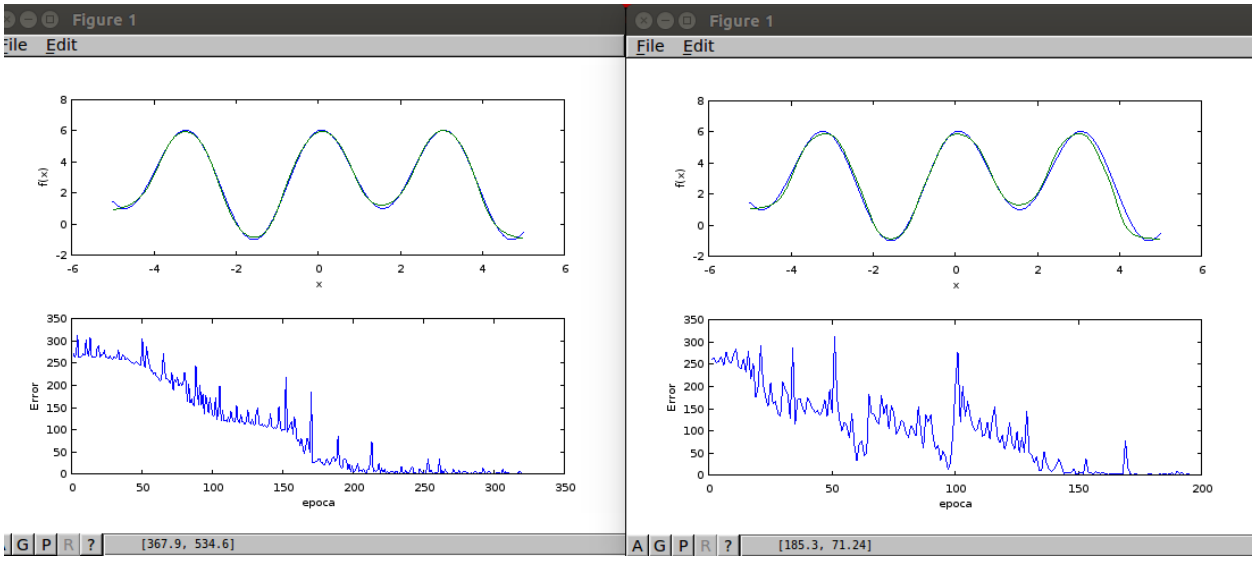

Figura 2: Las imágenes representan los resultados obtenidos para la arquitectura [1 15 10 1]. A la izquierda el aprendizaje a partir de la g2 y a la derecha el aprendizaje con g1. En la parte superior de cada imagen se encuentra: la función propuesta ($y = sin(x) + 6\,cos^2(x)$) graficada en el intervalo [-5:0.01:5] y la aproximación del perceptrón. En la parte inferior se puede observar el error cuadrático medio en función de la cantidad de épocas.

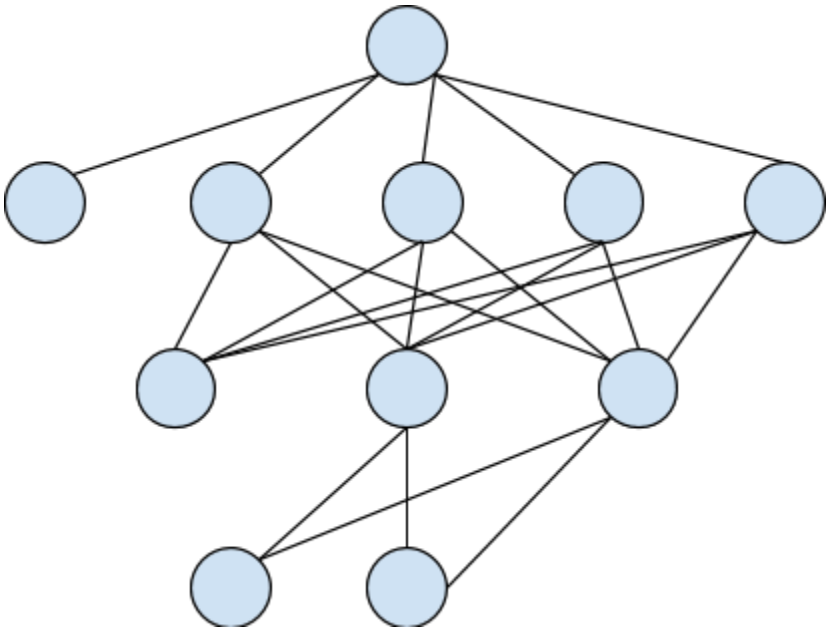

Figura 3: Ejemplo de perceptrón con una arquitectura de [1 2 4 1]

Capas	η	β	Momentum	η Adaptativo	a	b	Error	Epocas-g1
[1 100 1]	0.001	0.5	0.5	0	0	0	10	+(30000)
[1 20 15 10 1]	1	0.5	0.5	0	0	0	10	(+30000)
[1 256 128 128 64 32 16 1]	0.0001	0.75	0.5	0	0	0	1	941
[1 256 128 64 128 256 1]	0.0001	0.75	0.5	0	0	0	1	1515
[1 256 128 64 128 64 1]	1	0.75	0.5	0	0	0	1	1627
[1 512 256 128 64 32 16 8 1]	0.00005	0.75	0.5	0	0	0	1	889
[1 512 256 128 64 32 16 8 1]	0.0001	0.75	0.5	0	0	0	1	1002
[1 64 64 64 64 64 64 64 64 1]	0.0001	0.75	0.5	0	0	0	1	2687
[1 128 128 64 64 32 32 16 1]	0,00005	0,75	0,5	0	0	0	1	1477
[1 512 256 128 64 32 16 1]	0,00001	0.75	0.5	0	0	0	1	1063

Figura 4: Tabla comparativa para las distintas arquitecturas de múltiples capas ocultas.

Capas	η	β	Momentum	η Adaptativo	a	b	Error	Epocas-g1	Tiempo(s)
[1 512 256 128 64 32 16 8 1]	0,0001	0.75	0.5	0	0	0	1	843	596,84
	0,0001	0.5	0.5	0	0	0	1	1180	1.612,75
	0,0001	1	0.5	0	0	0	1	1000	2.000,00
	0,0001	0,75	0.9	0	0	0	1	2785	5.369,93
	0,0002	0,75	0,5	0	0	0	1	1348	2.381,78
	0,00005	0,75	0.5	0	0	0	1	1110	1.675,20
[1 256 128 128 64 32 16 1]	0,0001	0.75	0.5	0	0	0	1	1248	454,18
	0,0001	0.5	0.5	0	0	0	1	3041	1.798,37
	0,0001	1	0.5	0	0	0	1	1687	1.649,02
	0,0001	0,75	0.9	0	0	0	1	3655	3.218,92
	0,0002	0.75	0.5	0	0	0	1	1569	1.562,52
	0,00005	0.75	0.5	0	0	0	1	1974	1.604,96
[1 512 256 128 64 32 16 1]	0,0001	0,75	0,5	0	0	0	1	810	914,504
	0,0001	0,5	0,5	0	0	0	1	1568	2982,73
	0,0001	1	0,5	0	0	0	1	1344	2955,16
	0,0001	0,75	0,9	0	0	0	1	1753	3767,68
	0,0002	0,75	0,5	0	0	0	1	958	1914,35
	0,00005	0,75	0,5	0	0	0	1	952	1290,56

Figura 5: Tabla comparativa de las mejores arquitecturas resultantes de la figura 4.

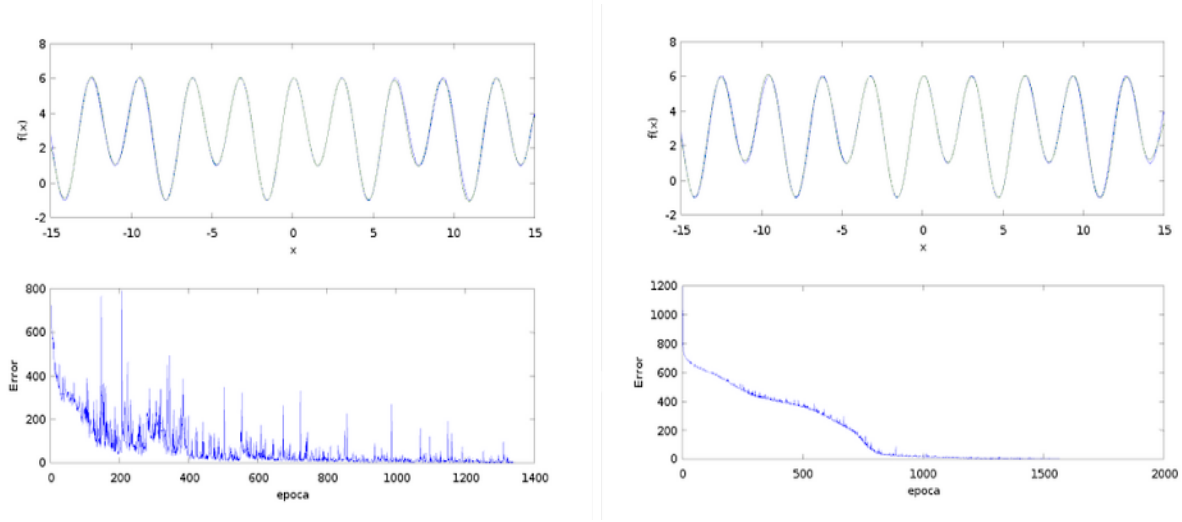

Figura 6: Una de las arquitecturas utilizadas en la Figura 5 : [1 512 256 128 64 32 16 1]. En la imagen de la izquierda β **= 1,** mientras que a la derecha β = 0.5.

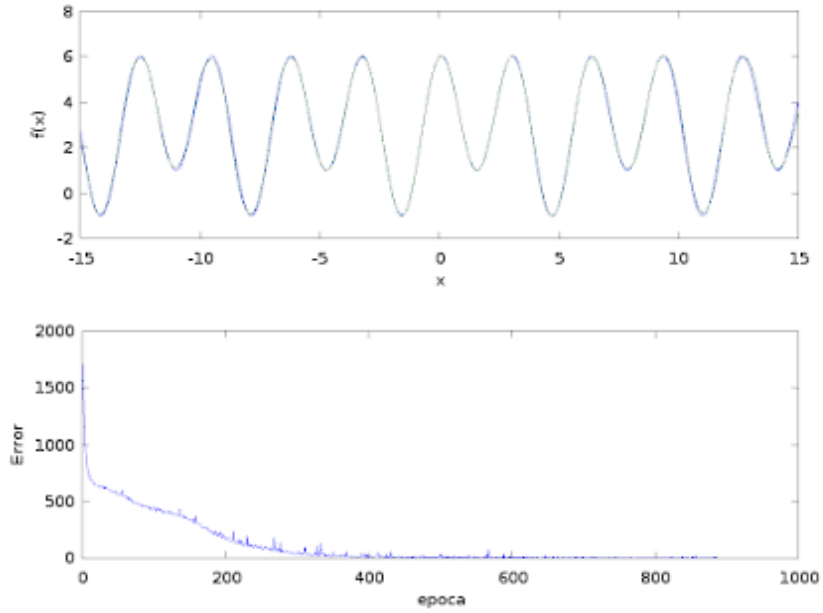

Figura 7: Gráfico de un resultado: 889 épocas, [1 512 256 128 64 32 16 8 1], η = 0.00005

Capas	η	β	Momentum	η Adaptativo	a	b	Error	Epocas- g1	Tiempo(s)
[1 256 128 128 64 32 16 1]	0,0001	0,75	0,5	0	0	0	1	1180	283,16
[1 512 256 128 64 32 16 8 1]	0,0001	0,75	0,5	0	0	0	1	852	722,40
[1 512 256 128 64 32 16 1]	0,0001	0,75	0,5	0	0	0	1	903	773,41

Figura 8: Tabla comparativa entre las mejores arquitecturas de la figura 5 (en el intervalo [-15:15]), con los puntos distribuidos de la siguiente manera [[-15 : 0.05 : -12.5] [-12.5 : 0.25 : -5] [-5 : 0.5 : 5] [5 : 0.25 : 12.5] [12.5 : 0.05 : 15]].

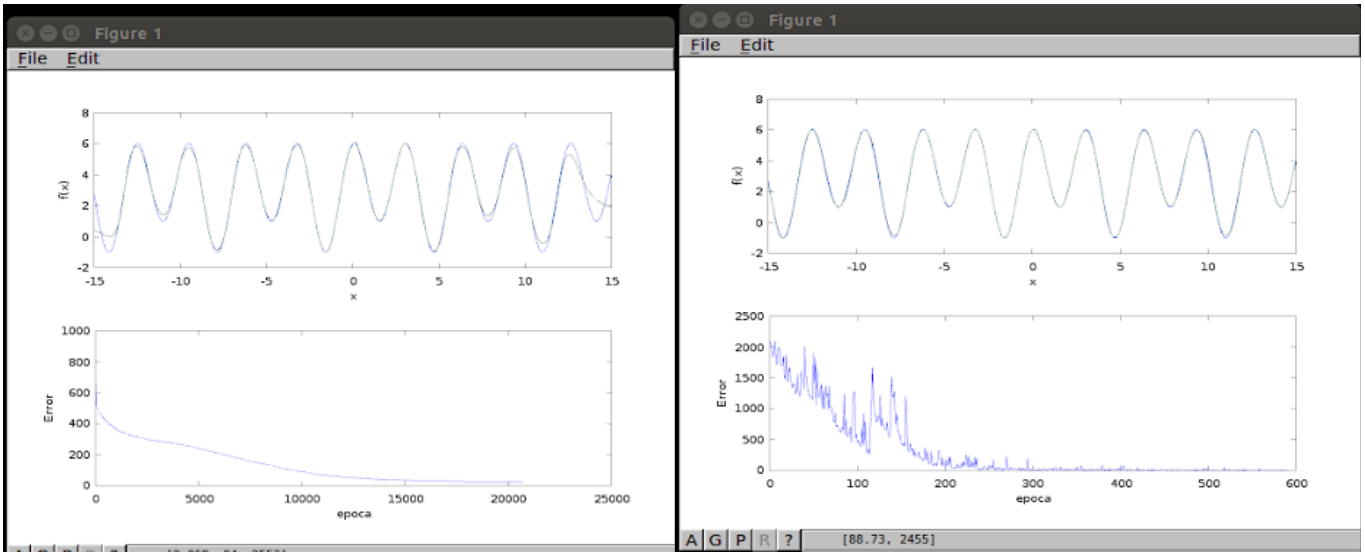

Figura 9: Análisis de la arquitectura [1 512 256 128 64 32 16 1], en ambas η = 0.0001, β=0.75 y momentum = 0.5. En la imagen de la izquierda se utiliza eta adaptativo (k=5, a=0.0005 y b=0.1).