SISTEMAS DE INTELIGENCIA ARTIFICIAL

TRABAJO PRÁCTICO ESPECIAL 2

REDES NEURONALES
APROXIMACIÓN DE TERRENOS
GRUPO 10

ALUMNOS

- **52056 – Juan Marcos Bellini**
- **53559 – Natalia Navas**
- **54308 – Francisco Bartolomé**

Tabla de Contenidos

Descripción del Problema

El problema consiste en aproximar la función de altura de un terreno mediante el uso de una red neuronal multicapa con aprendizaje supervisado. Para ésto se tiene una serie de datos de dicho terreno, con los cuales la red debe aprender a aproximar la función.

Implementación

Para aproximar la función del terreno pedido, se utilizó el algoritmo de backpropagation junto con ciertas variaciones.

Inicialización de los pesos

Los pesos fueron inicializados con valores aleatorios dentro del rango:

$$(-k^{1/2}, k^{1/2})$$

para la capa *i*, donde *k* es el número de entradas que alimentan a la capa *i*.

Funciones de Activación

Como funciones de activación se utilizaron las siguientes, con sus correspondientes derivadas:

- Función Hiperbólica (para funciones entre -1 y 1):

$$g(h) = tanh(\beta h)$$
$$g'(h) = \beta(1 - g^2)$$

- Función Exponencial (para funciones entre 0 y 1):

$$g(h) = \frac{1}{1+e^{-2\beta h}}$$
$$g'(h) = 2\beta g(1 - g)$$

Normalización de datos

Para que la red pueda calcular valores que estén por fuera del intervalo de la imágen de la función de activación usada, se decidió normalizar la salida de datos. En primer lugar, a la hora de

entrenar la red, se normalizan los datos de la salida deseada para que se encuentren en un intervalo poco menor que la imágen de la función de activación. Por ejemplo, si la imágen de la función de activación es [-1,1], entonces se normaliza la salida deseada a valores que estén en el intervalo [-0.9,0.9]. De esta manera se logra que la red pueda calcular valores más extremos. A la hora del testo, para que el valor de la salida se encuentre en el intervalo correcto, se realiza el proceso inverso desnormalizando la salida de la red.

Por otro lado, se decidió normalizar las entradas (tanto para el testeo como para el entrenamiento), para que sea más difícil que las neuronas se saturen. Esta normalización lleva los datos de entrada al intervalo de la imágen de la función de activación.

Todas las normalizaciones se realizaron con la siguiente fórmula

$$norm(x) = a + \frac{(x-min)\,(b-a)}{(max-min)},$$

donde *a* y *b* son los valores mínimo y máximo respectivamente del intervalo en el que se quiere normalizar los datos, y *min* y *max* son los valores mínimos y máximos del conjunto de datos a normalizar.

El conjunto de valores de entrada varía entre -4 y 4. Por lo tanto, en la función de normalización, *min* y *max* valen -4 y 4, respectivamente. Por otro lado, el conjunto de valores de salida está entre -10 y 10, tomando dichos valores *min* y *max* respectivamente. En ambos casos, los valores de *a* y *b* son los valores entre los que varía la imagen de la función de activación escogida.

A continuación se pueden observar las funciones utilizadas.

Función de activación	Normalización de los datos de entrada	Normalización de los datos de salida
$g(h) = tanh(\beta h)$	$norm^{tanh}_{[-4,\,4]}(x) = \frac{(x+4)}{4} - 1$	$norm^{tanh}_{[-10,\,10]}(x) = \frac{(x+10)}{10} - 1$
$g(h) = \frac{1}{1+e^{-2\beta h}}$	$norm^{exp}_{[-4,\,4]}(x) = \frac{(x+4)}{8}$	$norm^{exp}_{[-10,\,10]}(x) = \frac{(x+10)}{20}$

Mejoras

Momentum

La mejora consiste en tomar en cuenta los pesos anteriores para calcular los nuevos. Se multiplica el peso anterior por una constante α, que determina el impacto de dichos pesos en los nuevos.

η adaptativo

Esta mejora consiste en que la constante de proporcionalidad de aprendizaje se vaya adaptando relativamente a la evolución del error. Si el error disminuye consistentemente (una cantidad *k* de pasos), entonces se le suma a la constante de proporcionalidad de aprendizaje una constante *a*. Si por otro lado el error aumenta entonces se le resta otra constante *b*, y se "deshace" el paso anterior. Es decir, se reemplazan los pesos por los últimos pesos guardados.

El problema que surge a partir de esto es que, cuando se cae en un mínimo local, para salir de dicho mínimo, el error debe aumentar. En este caso el algoritmo deshace este paso, ya que el error estaría aumentando. Para solucionar esto, se agregó un factor de probabilidad. Si un valor tomado al azar uniformemente entre 0 y 1 es mayor a 0.5, entonces se deshace el paso. En caso contrario, no se lo hace. De esta manera, el algoritmo sale eventualmente del mínimo local.

Comparación de Resultados

Comparación de las funciones de activación

La comparación se basa en el error cuadrático medio. Para realizar las pruebas se decidió limitar la condición de corte del algoritmo de backpropagation a 2000 épocas. La cantidad de patrones de entrenamiento se mantiene fija en 300 (siendo siempre los mismos patrones). El valor de η es de 0.05. Por otro lado, las arquitecturas fueron elegidas de manera aleatoria.

Arquitectura de la red (capas ocultas)	Error Cuadrático Medio de entrenamiento (Tangente hiperbólica)	Error Cuadrático Medio de entrenamiento (Exponencial)	Error Cuadrático Medio de testeo (Tangente hiperbólica)	Error Cuadrático Medio de testeo (Exponencial)
[6 4 4 4]	0.0018	0.0063	0.3011	4.6498
[7 7]	0.0011	0.0015	0.1499	0.9579
[9 9 9]	7.4455e-04	4.1799e-04	0.1358	0.2558
[10 8 5 4 2]	0.0013	0.0031	0.2630	2.2431
[20 10]	7.6796e-04	7.8030e-04	0.0805	0.5036
[5 20 5]	0.0020	0.0047	0.2398	3.5731
[3 3 3 3 3 3]	0.0044	0.0071	0.6951	5.0215

Se observa que en el entrenamiento, hay mayor error cuando se usa la función exponencial como función de activación. De todos modos, la mayor diferencia se observa en el testeo de los datos, donde también el uso de la función exponencial resulta en un mayor error cuadrático medio.

A partir de estos resultados, se decidió realizar el resto de las pruebas usando como función de activación la tangente hiperbólica.

El la figura 1, podemos observar la variación en los errores de testeo y entrenamiento para una de las redes con arquitectura [6 4 4 4]. Se puede observar como el error es mayor si se utiliza como función de activación la exponencial. También se puede ver que el error de testeo es más intermitente que el de entrenamiento, variando más sus valores.

Comparación del tamaño del conjunto de entrenamiento

Para el siguiente experimento se consideraron distintas cantidades de patrones para entrenar una serie de arquitecturas. A partir de esto, se calculó el error cuadrático medio del testeo. Se realizaron 5 pruebas para cada caso particular, y se calculó el promedio. Para cada prueba, las redes fueron entrenadas una cantidad de 2000 épocas, utilizando un η de 0.05.

Arquitectura de la red (capas ocultas)	50 patrones	100 patrones	150 patrones	200 patrones	300 patrones	400 patrones
[7 7]	1.6535	0.3969	0.1799	0.2389	0.1178	0.1179
[9 9 9]	2.6667	0.8255	0.6085	0.3671	0.1093	0.0865
[10 8 5 4 2]	2.2025	1.7230	0.8926	0.2408	0.1327	0.2227
[20 10]	3.8901	1.8837	0.5185	0.3363	0.1902	0.1344
[5 20 5]	3.0995	1.4130	0.5783	0.2803	0.1967	0.1765
[3 3 3 3 3 3]	3.7518	2.6656	1.3272	0.5757	0.2392	0.1065

Se puede observar en la tabla, que a medida que la cantidad de patrones para el entrenamiento aumenta, el error cuadrático medio para el testeo decrece. Estas caídas en el error son considerables (se reducen a más de la mitad cuando se cambia de categoría), pero luego de los 300 patrones no se logra observar una mejora notable en el error cuadrático medio. De hecho, hay casos en donde el error es más grande si se entrena con 400 patrones que si se entrenase con 300. Si bien esto último puede ser causado debido al cierto grado de aleatoriedad en el entrenamiento (por eso se realizaron diversas corridas), también puede ocurrir que la red se sobreentrene, perdiendo capacidad de generalización.

De todos modos, al no encontrar una mejora sustancial se cree que con 300 patrones es suficiente para el propósito de este trabajo. Los siguientes experimentos serán realizados utilizando dicha cantidad de patrones de entrada para el entrenamiento de las redes.

En la figura 2 se pueden observar los errores cuadráticos medios (en promedio) para cada categoría.

Comparación de α en backpropagation con momentum

Utilizando un valor de η de 0.05, 2000 épocas y variando en arquitecturas se midieron los resultados para distintos valores de α, para ver cuál es el mejor. La siguiente tabla muestra el error cuadrático medio del testeo para cada caso.

	$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$	$\alpha = 0.7$	$\alpha = 0.9$
[10 8 5 4 2]	0.3245	0.1744	0.0897	0.1163	0.1511
[20 10]	0.1032	0.1091	0.1390	0.0915	0.3412
[3 3 3 3 3 3]	0.2264	0.2435	0.2143	0.3693	4.3248
[5 20 5]	0.0904	0.1238	0.2121	0.1083	0.3140
[7 7]	0.0783	0.1384	0.1146	0.1085	0.1335
PROMEDIO	*0,1646*	*0,1578*	*0,15384*	*0,1588*	*1,05292*

Si bien no se puede distinguir un valor de α que mejore considerablemente el error cuadrático medio respecto al resto de los α, tomando el promedio del error el mejor α resulta ser el que vale 0.5. Es por esto que se decidió tomar dicho valor como el α óptimo.

En la figura 3 se pueden observar los errores cuadráticos medios para cada valor α.

Comparación de a y b con backpropagation adaptativo

Para el siguiente experimento, se utilizó el algoritmo de backpropagation adaptativo. Se mantuvieron constantes las variables testeadas anteriormente y se variaron los valores de *a* y *b*. Cada prueba fue realizado 5 veces, y se calculó el promedio de los errores cuadráticos medios del testeo para cada una.

	$a = 0.001$ $b = 0.3$	$a = 0.01$ $b = 0.3$	$a = 0.04$ $b = 0.3$	$a = 0.08$ $b = 0.3$	$a = 0.1$ $b = 0.3$
[10 8 5 4 2]	0.9437	0.2515	0.36753	0.24361	0.4470
[20 10]	0.6730	0.3672	1.2280	2.3203	9.0764
[3 3 3 3 3 3]	4.0828	0.2387	0.2364	1.6561	2.5039
[5 20 5]	1.0217	0.3327	0.59478	1.4872	0.4713
[7 7]	0.4717	0.1607	0.57822	0.96164	0.6055

[9 9 9]	0.3697	0.1071	0.20331	0.9029	0.2981

	$a = 0.01$ $b = 0.08$	$a = 0.01$ $b = 0.1$	$a = 0.01$ $b = 0.2$	$a = 0.01$ $b = 0.3$	$a = 0.01$ $b = 0.4$
[10 8 5 4 2]	0.2231	0.2569	0.1004	0.2683	2.7090
[20 10]	0.3597	0.7814	0.1799	0.1297	1.7373
[3 3 3 3 3 3]	5.2786	0.5181	0.1786	0.6484	1.4899
[5 20 5]	0.2390	0.4099	0.1820	0.2464	3.1027
[7 7]	0.1101	0.1347	0.1821	0.1692	0.1687
[9 9 9]	0.1160	0.1667	0.1453	0.1658	0.1458

De estos resultados, se concluyó que, para el backpropagation adaptativo, es conveniente usar valores de *a* y de *b* de 0.01 y 0.2, respectivamente.

En las figuras 4 y 5 se pueden ver los errores para distintos valores de a y b respectivamente.

Comparación de Backpropagation con y sin implementaciones de mejoras

Se compararon resultados realizando los entrenamientos sin aplicar ninguna mejora, aplicando cada una de ellas, y aplicando ambas mejoras a la vez. Los valores de α, *a* y *b* son los óptimas calculados en las secciones anteriores 0.5, 0.01 y 0.2, respectivamente)

A diferencia del resto de las pruebas, se decidió entrenar las redes una mayor cantidad de épocas, 3000. Esto se debe a que se cree que más épocas darán un mayor efecto en cada una de las mejoras.

Para cada caso, se realizó cada entrenamiento 5 veces, y se hizo un promedio de los errores cuadráticos medios de testeo.

	Sin mejora	**Con momentum**	**Con η adaptativo**	**Con momentum y η adaptativo**
[10 8 5 4 2]	0.1113	0.0935	0.0876	0.8128
[20 10]	0.1011	0.1659	0.0922	0.1400
[3 3 3 3 3 3]	1.1096	0.1646	0.8251	0.2618
[5 20 5]	0.1923	0.2371	0.1545	0.1689
[7 7]	0.1216	0.1891	0.1488	0.1858
[9 9 9]	0.0790	0.0778	0.0945	0.1433

PROM	*0.2858*	*0.1547*	*0.2383*	*0.2854*

Calculando un promedio de los errores obtenidos en cada una de las arquitecturas llegamos a los promedios de 0.2858, 0.1547, 0.2383 y 0.2854 para los algoritmos sin mejora, con momentum, con n adaptativo y con ambos respectivamente. Se puede ver que el algoritmo que utiliza momentum es el que mejor resultados obtiene. Curiosamente, éste obtiene mejores resultados, que el algoritmo que implementa las dos mejoras. Por éste motivo, se cree que la mejora de η adaptativo no resulta de gran efectividad, por lo que se decidió prescindir de ella en las pruebas futuras.

Para el resto de las pruebas se utiliza el algoritmo que implementa la mejora del momentum.

En las figura 6 se pueden ver los errores cuadráticos medios para cada una de las categorías.

Comparación de distintos η

Manteniendo constantes la cantidad de iteraciones, el alfa, el tipo de algoritmo, siendo estos 1000, 0.5 y momentum backpropagation respectivamente, se probaron distintas arquitecturas con distintos valores de η, para obtener el óptimo, basado en el error cuadrático medio. Se probó únicamente con 1000 épocas ya que, para distinguir la diferencia entre los distintos valores de η, dicha cantidad es suficiente.

	$\eta = 0.01$	$\eta = 0.03$	$\eta = 0.05$	$\eta = 0.07$	$\eta = 0.1$	$\eta = 0.2$
[10 8 5 4 2]	0.1828	1.1090	0.2582	0.4957	0.1754	11.5765
[20 10]	0.1290	01664	0.1696	0.2136	0.6405	20.8082
[3 3 3 3 3 3]	0.9758	0.1710	0.2398	0.3321	0.2330	0.2334
[5 20 5]	1.8028	0.2526	0.2400	0.2425	11.8052	21.3515
[7 7]	0.1931	0.1361	0.1288	0.3710	0.0955	1.2865
[9 9 9]	0.2533	0.1756	0.1098	0.2269	0.6161	9.6574
PROM	0,5895	0,3351	0,191	0,3136	2,26095	10,8189

Se puede observar en la tabla de resultados que los valores obtenidos se alejan en mayor medida de la solución real utilizando valores de η grandes (mayores o iguales a 0.1), en comparación con valores de η menores.

Al ver los promedios de los errores cuadráticos medios para cada uno de los valores de η se concluye que el valor óptimo η es de 0.05. El resto de las redes serán entrenadas utilizando 0.05 como valor de velocidad de convergencia.

Comparación de arquitecturas

Una vez calculados los óptimos para cada una de las variables relevantes al problema, se prosiguió a encontrar la mejor arquitectura. Se decidió utilizar solo arquitecturas que constan de 2 capas ocultas ya que se considera que dicha cantidad de capas es suficiente.

Se utilizó el algoritmo de backpropagation con momentum, con un valor alfa de 0.5, un valor n de 0.05 y 2000 iteraciones.

Arquitectura	Error medio	Error cuadrático medio
[3 3]	0.4502	0.4268
[4 4]	0.3032	0.1454
[5 5]	0.2625	0.1303
[3 5]	0.3727	0.2650
[6 6]	0.2967	0.1616
[6 3]	0.2664	0.1204
[3 6]	0.2983	0.1679
[8 5]	0.2682	0.1374
[7 7]	0.2554	0.1069
[8 8]	0.2185	0.0813
[9 9]	0.2445	0.0978
[9 5]	0.2331	0.0964
[8 5]	0.2682	0.1374
[5 9]	0.3064	0.1708
[11 9]	0.2279	0.0903
[11 11]	0.3602	0.2326

Luego de diversas pruebas, se concluye que la red óptima (de dos capas) para resolver el problema es la que consta de dos capas ocultas de 8 neuronas cada una, utilizando el algoritmo backpropagation con la mejora de momentum usando 0.5 como valor de α, 0.05 como valor de η, con 300 patrones de entrenamiento y utilizando la tangente hiperbólica como función de activación.

Se puede observar en la figura 8 una representación de los datos de la tabla, y en la figura 9 una representación del terreno calculado con la red óptima donde los círculos rojos que se pueden observar son los valores reales de la función a aproximar. En la figura 8 las arquitecturas están en el mismo orden que en la tabla con los datos.

Se entiende que los resultados podrían ser más precisos si la restricción del entrenamiento de la red no fuese la cantidad de época, sino el error. De todos modos se decidió basar el análisis en dicha restricción por cuestiones prácticas. Sería interesante a futuro realizar las pruebas con una mayor cantidad de épocas para el entrenamiento, o con una condición de corte basada únicamente en el error de entrenamiento.

De todos modos, se cree haber derivado en una red capaz de resolver el problema planteado de manera precisa y eficiente.

Anexo

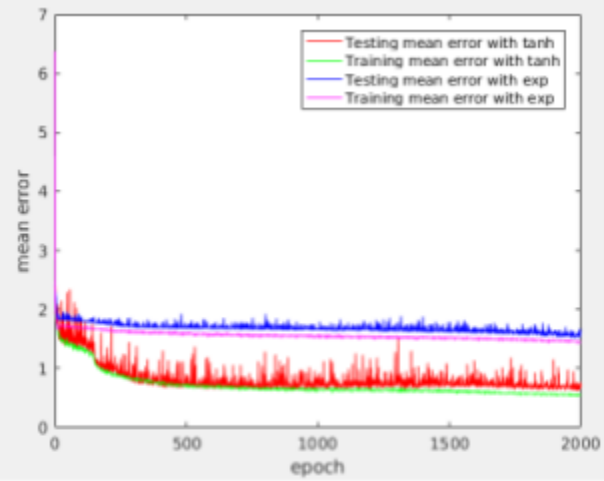


Fig 1: Error medio de testeo y entrenamiento para arquitectura de [6 4 4 4] usando distintas funciones de activación.

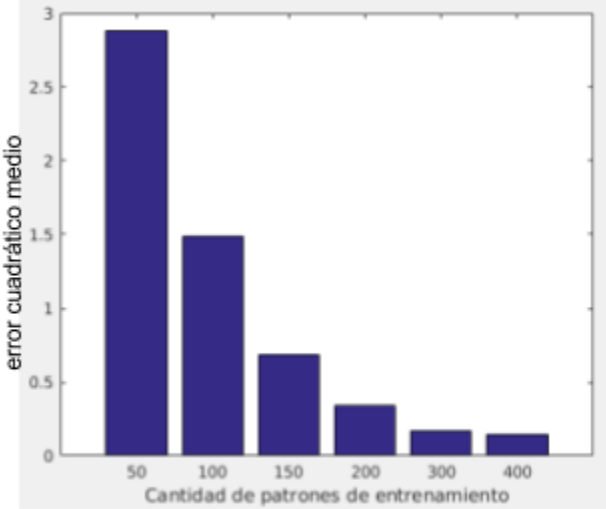


Fig 2: Promedio de los errores usando distintas cantidades de patrones de entrada

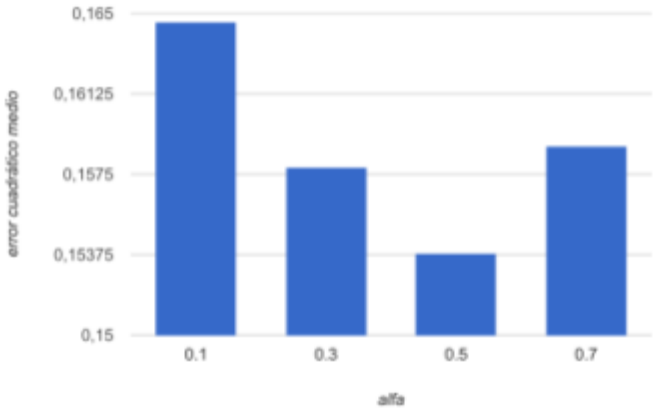


Fig 3: error promedio dependiente de valores alfa

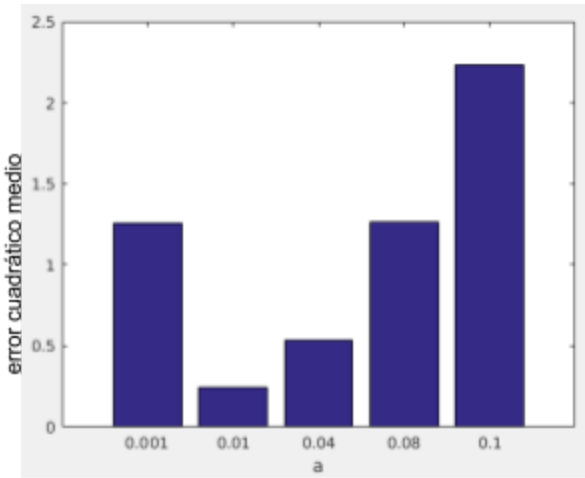


Fig 4: Promedio de errores para distintos valores a.

[1]

[1] Se eligió no graficar los valores con alfa = 0.9 ya que al hacerlo no se lograba apreciar la diferencia entre los errores producidos con los otros.

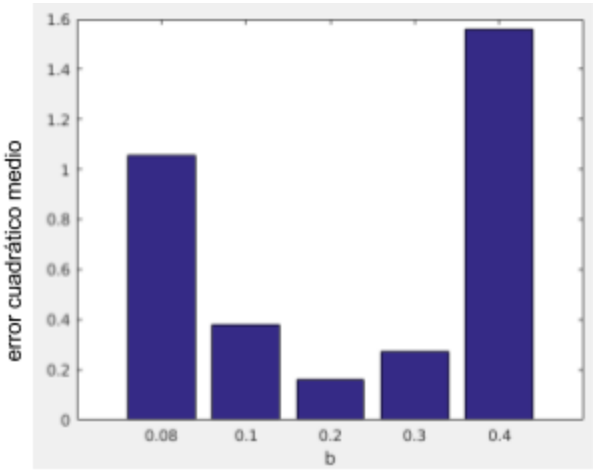


Fig 5: Promedio de errores para distintos valores b.

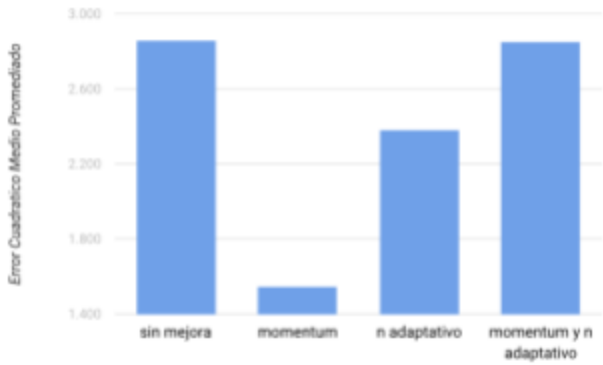


Fig 6: Errores promedio utilizando distintas versiones del algoritmo backpropagation

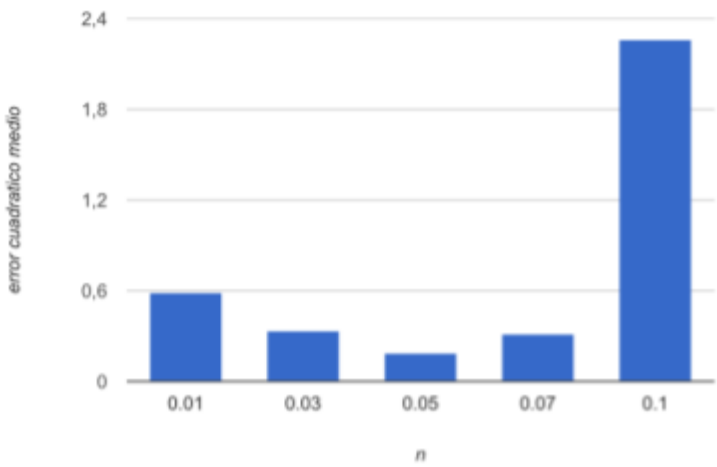


Fig 7: Errores dependiendo de distintos valores de eta.

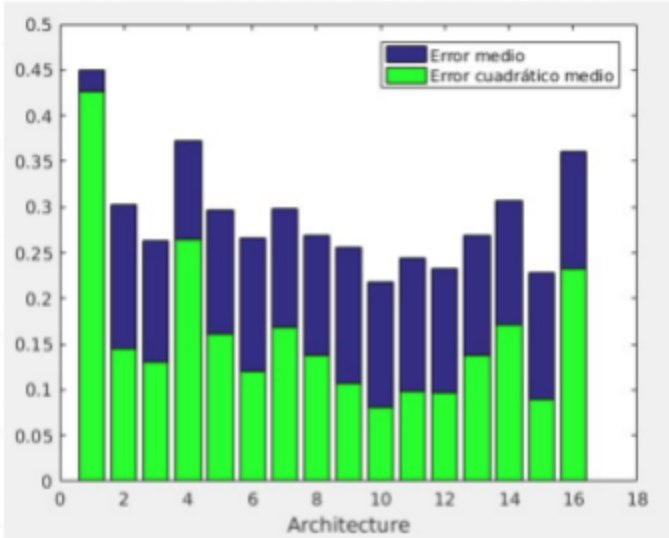


Fig 8: Errores dependiendo de la arquitectura utilizada

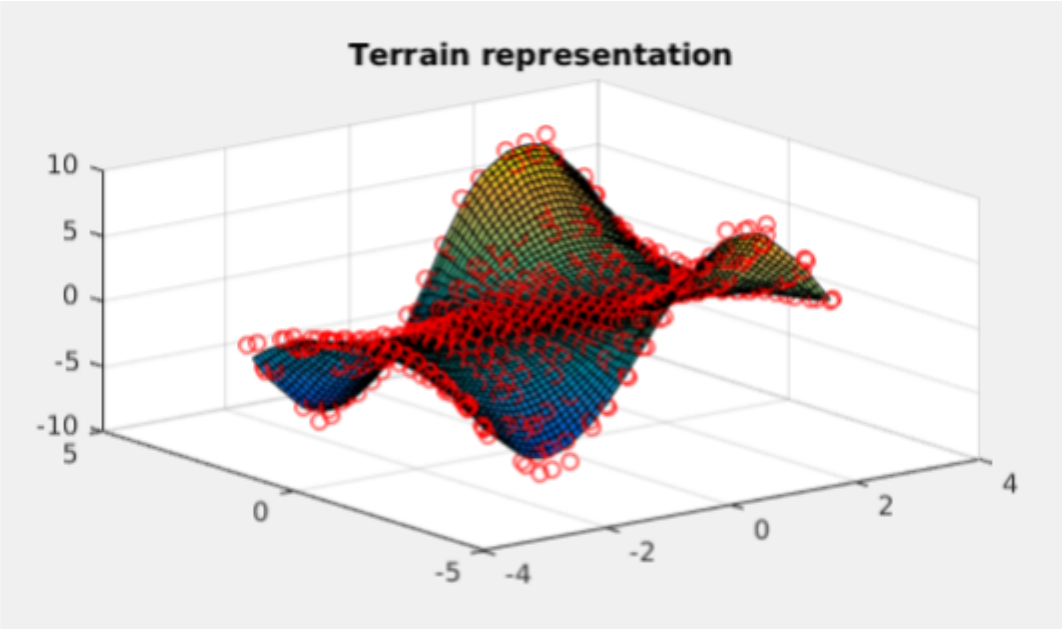
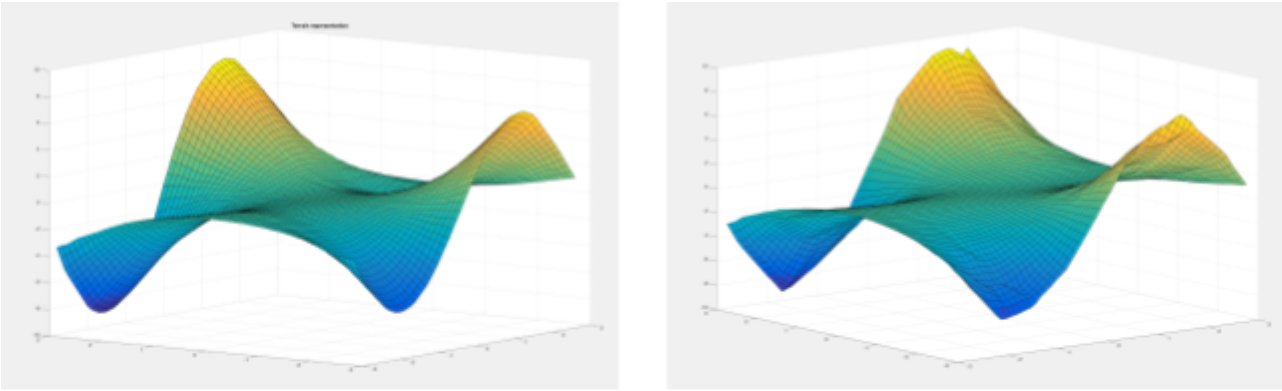


Fig 9: Representación gráfica del terreno calculado utilizando la red óptima.



aproximación real

Fig 10: Comparación de terreno real y aproximado.