Sistemas de Inteligencia Artificial

TRABAJO PRÁCTICO ESPECIAL 2 - INFORME

REDES NEURONALES

GRUPO 1 - 1C 2017

DOCENTES

- Parpaglione, María Cristina
- Pierri, Alan

INTEGRANTES

- Comercio Vázquez, Matías Nicolás - 55309
- Ibars Ingman, Gonzalo - 54126
- Mercado, Matías - 55019
- Moreno, Juan - 54182

Introducción

En el siguiente trabajo práctico se describen y comparan los resultados obtenidos de implementar redes neuronales con aprendizaje supervisado, utilizando como entradas y salidas esperadas los patrones asignados al Grupo 1.

Implementación

Parametrización

- De los patrones, el porcentaje elegido para entrenar y para testear es 70 y 30 respectivamente. Estos porcentajes fueron elegidos por recomendación de la cátedra, considerando además que la cantidad de patrones dados permite tomar estos porcentajes y que la red aún sea entrenada con un considerable número de patrones. Estos conjuntos se obtienen pseudo-aleatoriamente y no se discriminan los patrones en cada conjunto.
- Los pesos iniciales obtenidos pseudo-aleatoriamente son los que corresponden al rango $[-\sigma_m, \sigma_m]$ siendo m la cantidad de conexiones entrantes a esa unidad y $\sigma_m = m^{-1/2}$. Esta decisión se tomó para reducir la posibilidad de que las conexiones se saturen tempranamente, es decir, antes de la que red esté siendo verdaderamente entrenada.
- Se permite modificar los siguientes parámetros previo al comienzo del entrenamiento:
    - η (eta) ó learning rate
    - ε (epsilon) ó threshold de entrenamiento
    - la función de activación y su derivada en las capas ocultas y en la capa de salida (pueden ser distintas entre sí). En caso de utilizar la función tangencial o exponencial se puede elegir el β que multiplica a la entrada de la función.
    - la cantidad de neuronas en cada capa oculta y la cantidad de capas ocultas
    - α (alpha) ó peso sobre el $\Delta w_{pq}(t)$ aplicado en las Variaciones/Optimizaciones (si α = 0 se deshabilita *momentum*)
    - habilitación/deshabilitación de η adaptativo
    - habilitación/deshabilitación de variación de los deltas de cada capa, según lo explicado en Variaciones/Optimizaciones

Algoritmo de aprendizaje

A continuación se indican los puntos más importantes del algoritmo de aprendizaje.

- El bias de cada neurona se obtiene según lo visto en la teórica, es decir, simulando una conexión con una neurona de la capa anterior con salida constante e igual a -1.
- Cada un cierto intervalo de épocas definido por el usuario se guarda toda la configuración de la red, de manera tal que el usuario pueda terminar la ejecución del entrenamiento en cualquier momento. Además, durante el entrenamiento, se guarda

la última red en la cual se detectó una “intersección” entre la normalización o promedio del error cuadrático medio del entrenamiento y de testeo. La detección se realiza utilizando alguno de los siguientes criterios:

- Se calcula la diferencia entre el error cuadrático medio de entrenamiento y de testeo de la época actual y de la anterior y se les aplicaba la función signo[1]. La condición se cumple si el signo de ambas difieren.
- La diferencia entre el error cuadrático medio de entrenamiento y testeo de la época actual es igual a cero.

Sin embargo, cabe aclarar que la información de la época en la que ocurre esta intersección (la cual incluye los pesos para cada capa de la red en ese momento) sólo se devuelve al finalizar el entrenamiento exitosamente.

Condiciones de corte

- *Threshold* ε de entrenamiento: debe ocurrir que para todo patrón dentro del conjunto de patrones de entrenamiento la diferencia entre la salida devuelta por la red neuronal y la salida esperada sea menor a ε.
- Forzar la finalización del proceso de entrenamiento (el usuario debe hacerlo manualmente). Las redes devueltas son las mencionadas en el último ítem de la sección de Algoritmo de aprendizaje.

Variaciones/Optimizaciones

La variación implementada es la solución de compromiso entre el error cuadrático medio y la medida entrópica que mantiene los δ diferentes a cero, incrementando en 0.1 la evaluación de la derivada de la función de activación. Otras optimizaciones implementadas son: momentum y η adaptativo.
Cabe aclarar que si bien el η adaptativo está implementado, al comenzar a entrenar la red, se pudo notar que la cantidad de épocas que requería el entrenamiento de la misma era considerablemente mayor que la requerida sin esta optimización, con lo cual se decidió no utilizarla para el estudio del presente trabajo.

Criterios de análisis

Error de entrenamiento y testeo

Durante el entrenamiento de la red, se obtienen los errores cuadráticos medios (normalizados o promediados) de entrenamiento y testeo en cada época. Los mismos se grafican en tiempo real para poder visualizar el entrenamiento de la red neuronal, detectando, por ejemplo, anomalías, falta de entrenamiento y/o sobreentrenamiento.

[1] https://www.mathworks.com/help/matlab/ref/sign.html

Distancia lineal por patrón de la salida esperada

Para cada patrón del conjunto de patrones de entrenamiento se realiza un gráfico de barras (actualizado cada cierto período configurable por el usuario, medido en épocas,) cuya altura representa la diferencia absoluta entre la salida actual de la red y la salida esperada. El objetivo de graficar esta distancia es el de poder visualizar fácilmente la existencia de patrones de entrenamiento que pueden llegar a evitar que la red complete su entrenamiento (es decir, que alcancen el ε configurado). En el eje x se indica el índice del patrón al que corresponde la distancia lineal que se está graficando.

Terrenos

Durante el entrenamiento de la red, cada una cierta cantidad de épocas (configurable en el archivo correspondiente) se grafican todas las entradas junto a las salidas de la red neuronal para poder visualizar la evolución del terreno (en estilo *mesh* y por puntos comparando las salidas de la red con las esperadas). Si bien no se pueden obtener datos concretos de estos gráficos, los mismos permiten visualizar cómo evoluciona la red como función aproximadora a través de las sucesivas épocas.
Luego de la condición de corte, se generan una cantidad de puntos aleatorios en un rango que abarca el dominio de los patrones dados, se le pasan como entrada a la red neuronal para obtener la salida, y finalmente se grafican dichos puntos y sus resultados para mostrar cómo funciona la red como aproximadora de la función real en un gráfico de tipo *mesh*.
Asimismo, en otro gráfico se representan los mismos puntos aleatorios en conjunto con los correspondientes a los patrones asignados junto con sus salidas esperadas y con las salidas obtenidas por la red.

Comparaciones al cambiar el número de patrones

Para comparar los errores cuadráticos medios de testeo para una misma arquitectura de red pero diferente cantidad de patrones elegidos, se normalizan los valores de error dividiendo por la cantidad de patrones. Cabe aclarar que la normalización o promedio de dichos errores se realizó para poder comparar estos errores independientemente de la cantidad de patrones que se tome en cada caso.

Capacidad de generalización de la red

Dado que una condición de corte posible es utilizar el *threshold*, se decidió medir la capacidad de generalización de la red utilizando los patrones de testeo. Sea **x** la cantidad de errores (diferencia entre salidas esperadas de los patrones de testeo y salidas de la red) por debajo de ε, **y** la cantidad total de patrones en el conjunto de patrones de testeo; luego, CGR = **x** / **y** * 100.

Problemas encontrados

Las salidas esperadas en los patrones dados son todos valores en el rango [0,1], exceptuando un caso en que la salida esperada es 1.055. Ésto trajo el siguiente problema: al utilizar la tanh[2] cómo función de activación en la capa de salida (el caso es análogo para exp), por lo visto en la teórica, existe baja probabilidad de que las salidas alcanzadas estén cerca de los bordes (-1 y 1) haciendo más difícil que se cumpla la condición de corte del *threshold*. Tomando un ε igual a 0.1 ocurre que la salida de la red debiera ser aproximadamente 0.955 para que se alcance dicha condición de corte, un valor que se observó difícil de alcanzar en los resultados con respecto al resto de las salidas esperadas.
Una posible solución a este problema es cambiar la función de activación de la capa de salida por una función lineal identidad[3] de tal manera que el rango de valores de salida de la red no sea acotado. Sin embargo, utilizar esta solución acarreó que los deltas en el algoritmo de aprendizaje se fuesen a infinito (valor representado por Matlab) utilizando un η mayor o igual a 0.5, haciendo que los pesos de todas las conexiones tiendan a infinito también, imposibilitando el aprendizaje de la red.
Otra posible solución es controlar que ese patrón siempre se encuentre en el conjunto de patrones de testeo y no en el de entrenamiento.
Se decidió utilizar la primera solución propuesta porque se consideró que quitar un patrón del posible conjunto de patrones de testeo haría perder aleatoriedad en la forma en que se estaba corriendo el algoritmo. Esta solución fue posible de implementar al asegurar la utilización de un η menor que 0.5 y al no hacer uso del η adaptativo, por las razones expuestas en la sección de Variaciones/Optimizaciones.,

Casos de análisis

Los primeros casos de análisis tienen como objetivo analizar el comportamiento y aprendizaje de la red neuronal a través de diferentes estructuras o configuraciones de red, que se obtuvieron cambiando la cantidad de capas ocultas y el número de neuronas en cada una de ellas. Cabe aclarar que se utilizó una configuración de parámetros que se encontró empíricamente pertinente para poder realizar múltiples corridas sin que el entrenamiento de la red fuera demasiado costoso.
Se comenzó con una red cuyas capas ocultas son [10, 5, 5, 3], es decir, se utilizaron 4 capas ocultas, cuya cantidad de neuronas es 10 para la primera capa, 5 para la segunda, y así sucesivamente.
Luego se redujo en 1 la cantidad de capas para observar de qué manera se comportaba la red (cómo y qué aprendía al ser entrenada), tomando la cantidad de neuronas en las capas intermedias de la prueba anterior (5 y 5) en una única capa.
A continuación, se buscó reducir la cantidad de neuronas en las capas ocultas. Se prestó especial atención al comportamiento de la red al modificar la cantidad de neuronas de la última capa oculta, logrando disminuirla a un número de dos. Ésto se hizo para corroborar si incluso disminuyendo la cantidad de neuronas de esta última capa aún se podían distinguir

[2] https://www.mathworks.com/help/matlab/ref/tanh.html
[3] https://en.wikipedia.org/wiki/Identity_function

las salidas. Finalmente, se redujo la cantidad de capas ocultas a 2 y se probaron dos configuraciones diferentes: Con la primera se buscó averiguar si la red podía aprender únicamente con dos capas ocultas, mientras que con la segunda se buscó averiguar si aún con una menor cantidad de neuronas en la primera capa oculta, la red podía seguir aprendiendo con el ε elegido.
En el segundo casos de análisis, se buscó comparar la *performance* de entrenamiento de la red (es decir, la cantidad de épocas de entrenamiento que se requiere para que la red alcance el *threshold* especificado por el usuario), habilitando y deshabilitando las diferentes optimizaciones implementadas, con el objetivo de encontrar la configuración de red que minimice la cantidad de épocas necesarias para alcanzar el *threshold* configurado por el usuario.
El tercer caso de análisis buscó comprobar que la utilización de una función exponencial en las capas ocultas no es una buena decisión en cuanto al alcance de la condición de corte y, que la utilización de la tangente hiperbólica en todas las capas (incluyendo la de salida) resulta en el problema mencionado en la sección Problemas encontrados. Para estos análisis se utilizó una arquitectura de red elegida arbitrariamente de entre las utilizadas en el primer caso de análisis.
En el cuarto y último caso de análisis, se buscó comparar los errores cuadráticos medios normalizados obtenidos al entrenar una red con una arquitectura en particular, utilizando una cantidad diferente de patrones en cada caso. De nuevo, para este análisis se utilizó una arquitectura de red elegida arbitrariamente de entre las utilizadas en el primer caso de análisis.

Resultados

A continuación se muestran las tablas que corresponden con los pasos explicados en la sección anterior. En todos los casos, a menos especificado la contrario, se utilizaron los siguientes parámetros de configuración:

- ε = 0.1
- α = 0.9
- η = 0.1
- β = 1
- Se utilizó todo el conjunto de patrones proveídos por la cátedra

En las consecutivas tablas, entiéndase ***+ .1*** como la optimización de sumar 0.1 a las derivadas de las funciones de activación para el cálculo de los delta; ***α*** como la optimización de *momentum*, y ***η*** como la optimización del η adaptativo.

Neuronas en capas ocultas	**g salida**	**g ocultas**	**+ .1**	**α**	**η**	**# épocas entrenamiento**	**Capacidad de generalización de la red (%)**
10, 5, 5, 3	lineal	tanh	Sí	Sí	No	263	96.21
10, 10, 5	lineal	tanh	Sí	Sí	No	251	98
5, 5, 2	lineal	tanh	Sí	Sí	No	871	96

12, 3	lineal	tanh	Sí	Sí	No	1382	99.24
15, 2	lineal	tanh	Sí	Sí	No	740	96.21

Tabla 1: Resultados con diferentes arquitecturas de redes

Si se observa la *Tabla 1*, la arquitectura de red con mayor capacidad de generalización es la [12, 3]; sin embargo, no pueden descartarse el resto de los resultados ya que de volver a ejecutar los entrenamientos podría ocurrir que los mismos varíen (estos resultados dependen fuertemente del orden y de los patrones de entrenamiento y testeo elegidos).
En cuanto a la cantidad de épocas de entrenamiento, el que obtuvo la mayor performance fue la arquitectura del segundo caso.

Neuronas en capas ocultas	**g salida**	**g ocultas**	**+ .1**	**α**	**η**	**# épocas entrenamiento**	**Capacidad de generalización de la red (%)**
10, 10, 15	lineal	tanh	Sí	Sí	Sí	-	-
			Sí	Sí	No	2730	94.70
			Sí	No	No	> 4515	-
			No	No	No	4261	93.18

Tabla 2: Resultados con activación/desactivación de optimizaciones

Para el segundo caso de análisis, se encontró que al usar un ε = 0.1, los entrenamientos con y sin optimizaciones finalizaban utilizando aproximadamente la misma cantidad de épocas. En base a esto, se supuso que ε estaba siendo tomado demasiado *laxo* (pues no se llegaba a apreciar el efecto esperado de las optimizaciones), con lo que se decidió tomar ε = 0.05 (un *threshold* el doble de bajo que para las primeras cinco pruebas) y volver a entrenar a la red.
En la *Tabla 2*, se puede observar que, efectivamente, las optimizaciones comienzan a tener sentido una vez que a la red se le exige aprender con un error que escapa de los escenarios en que podría aprender sin utilizar dichas optimizaciones.
Notar que se puede observar empíricamente que las optimizaciones ayudan a decrementar la cantidad de épocas necesarias para entrenar la red cuando el ε es lo suficientemente bajo para que su efecto sea apreciado.
Como se dijo en la sección Variaciones/Optimizaciones, el eta adaptativo no fue considerado a la hora de realizar los análisis por los motivos allí expuestos. Es por eso que la primer fila de la *Tabla 2* se encuentra sin completar.
La configuración de optimizaciones indicada en la tercera fila de dicha tabla reza que la cantidad de épocas de entrenamiento es mayor a 4515. Esto se debe a que se consideró prueba suficiente para el caso en análisis que el entrenamiento con dichas optimizaciones llevase esa cantidad de épocas y aún no hubiese terminado. Por este motivo, se decidió terminar forzosamente el entrenamiento de la red en esta época y se registró ese hecho de esta forma.

Cabe aclarar que si bien el valor que representa la capacidad de generalización de la red disminuyó con respecto a la *Tabla 1*, no se debe perder de vista que dicho valor es ahora obtenido con un ε más chico (ε = 0.05 vs. ε = 0.1 de la primer tabla). Con esto, se debe entender que, comparativamente con los anteriores resultados, la capacidad de generalización de la red podría ser incluso mayor en estos segundos casos que en los primeros.

Neuronas en capas ocultas	**g salida**	**g ocultas**	**+ .1**	**α**	**η**	**# épocas entrenamiento**	**Capacidad de generalización de la red**
12,3	lineal	exp	Sí	Sí	No	~3500	-
12,3	tanh	tanh	Sí	Sí	No	~3000	-

Tabla 3: Cambio de funciones de activación en capas ocultas y de salida

En el tercer caso, observando la *Tabla 3*, comparado con las funciones de activación utilizadas en el primer caso, se puede observar que la cantidad de épocas de entrenamiento superan en 2 veces y media a la misma, sin haber siquiera terminado de entrenar. En el primer caso se pudo deducir que utilizar una función exponencial acotada entre 0 y 1 (*standard logistic function*) dificulta el aprendizaje de la red ya que existen valores dentro de las salidas esperadas muy cercanos a 0 y a 1, difíciles de ser resultado de evaluar la función exponencial. Es debido a eso que utilizar una función tangencial al estar acotada entre -1 y 1, y cuya probabilidad de producir salidas más cercanas a cero, hace que las salidas sean más cercanas a las deseadas. El utilizar una función tangencial en la capa de salida trae aparejado el problema mencionado en Problemas encontrados. Se concluye que utilizar una función lineal en la última capa de salida y funciones tangenciales en las capas ocultas es la mejor opción para esta arquitectura.

Neuronas en capas ocultas	**g salida**	**g ocultas**	**+ .1**	**α**	**η**	**Cantidad de patrones**	**# épocas entrenamiento**	**MSE de testeo normalizado**	**Capacidad de generalización (%)**
10, 10, 5	lineal	tanh	Sí	Sí	No	441	251	7,88E-04	99.24
10, 10, 5	lineal	tanh	Sí	Sí	No	341	685	4.78e-4	100
10, 10, 5	lineal	tanh	Sí	Sí	No	241	430	0.0016	93
10, 10, 5	lineal	tanh	Sí	Sí	No	141	478	0.0070	94.44
10, 10, 5	lineal	tanh	Sí	Sí	No	41	662	0.0457	75

Tabla 4: Cambio de la cantidad de patrones utilizados

En el último caso se utilizó una cantidad de patrones entre 41 y 441 (cantidad total de patrones) con pasos de a 100.

Se puede observar que si la cantidad de patrones aumenta, el error cuadrático medio (MSE) normalizado disminuye. Es interesante notar que la capacidad de generalización es aproximadamente un 17% más bajo que la media para la última entrada de la *Tabla 4*.
Si se comparan la *Figura 1* y *la Figura 2* se puede observar que a menor cantidad de patrones tomados mayor es la diferencia entre los terrenos obtenidos, incluyendo la obtención parcial del terreno, es decir, una sóla montaña, y su forma es empinada.
En la *Figura 3* se observa la salida para un conjunto de entradas tomadas al azar y cuya salida es la devuelta por la red. La estructura de la red corresponde a la de la última entrada de la *Tabla 1*. Es interesante observar a simple vista que la salida esperada más alta (1.0553) se encuentra cerca de las salidas de la red neuronal.
En la *Figura 4* se observa a la izquierda el gráfico de barras con el valor absoluto de las diferencias entre las salidas esperadas y la salida de la red al final del entrenamiento. La arquitectura de la red corresponde con la de la anteúltima entrada de la *Tabla 1*. Se puede observar en los gráficos de error que no se produce sobre entrenamiento y que existe una tendencia de la curva del error de testeo a ser similar al de entrenamiento.
En la *Figura 5* se observa que para la cantidad de épocas pasadas, el gráfico de barras muestra que no pudo aprender un patrón; ese patrón se corresponde con el valor 1.0553. Esto se debe al problema de utilizar la función tangencial en Problemas encontrados.

Conclusiones

Se pudo observar que la cantidad de capas y/o neuronas que tenga una red no está proporcionalmente asociada con la cantidad de épocas que se requiere para su entrenamiento (es decir, que tener más neuronas por capa no implica que la red vaya a entrenar más rápidamente).
Asimismo, se pudo observar que la capacidad de generalización no está directamente asociada con la velocidad de entrenamiento, sino con la calidad del mismo: como se explicó para los resultados obtenidos en la *Tabla 2*, si bien la cantidad de épocas que requirió el entrenamiento fue considerablemente mayor que las necesarias para entrenar los casos mostrados en la *Tabla 1* y si bien la capacidad de generalización en porcentaje es numéricamente muy similar, la capacidad de generalización de los resultados de la *Tabla 2* fue obtenida con un *threshold* mucho más chico, lo que implica un error de generalización mucho menor que para los resultados de la *Tabla 1*.
Además, no se pudo probar empíricamente que el η adaptativo fuese una optimización de la cantidad de épocas necesarias para el entrenamiento de la red. De igual manera, se debe tener en cuenta que para poder observar los verdaderos efectos del resto de los mecanismos de optimización del backpropagation, se debe forzar a la red a entrenar con el nivel de exigencia adecuado para ello.
Finalmente, vale aclarar que la configuración de red óptima depende del problema que esté siendo analizado, de la precisión con la que se desee que la red generalice, del tiempo que se está dispuesto a entrenar la red para obtener un nivel u otro de resultados, de la cantidad de patrones de los que se disponga, etc. De entre los casos analizados en el presente trabajo, se considera que la red que logró balancear correctamente todos los mencionados explicados previamente fue la exhibida en la *Tabla 2*, fila 2.

Anexo

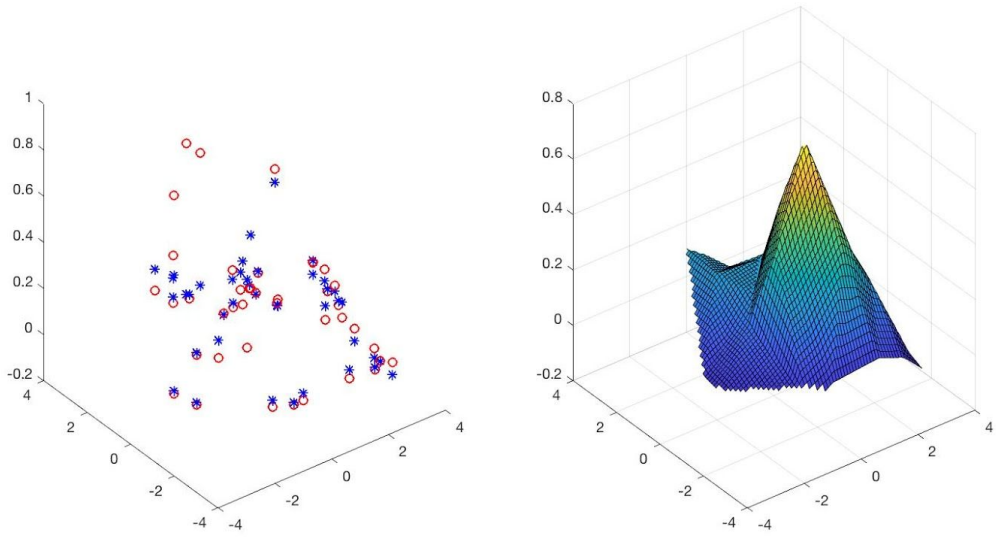

Figura 1: Ejecución del último caso de la Tabla 4

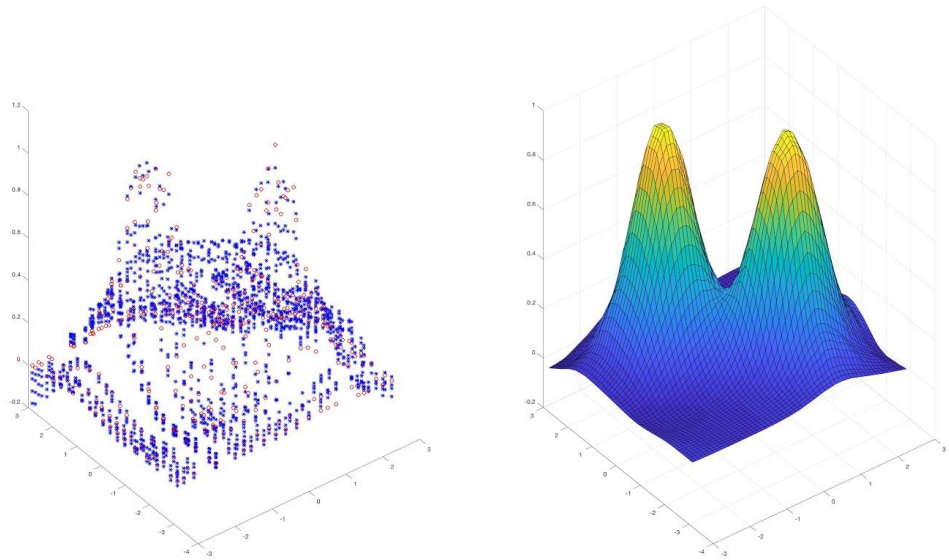

Figura 2: Ejecución del primer caso de la Tabla 4

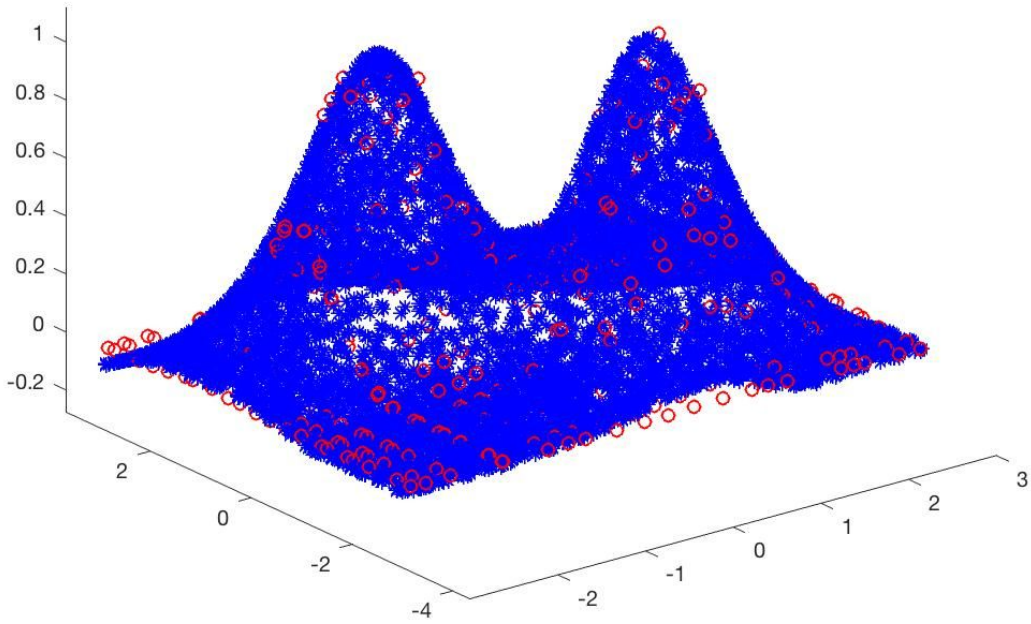

Figura 3: Terreno graficado utilizando la última entrada de la Tabla 1

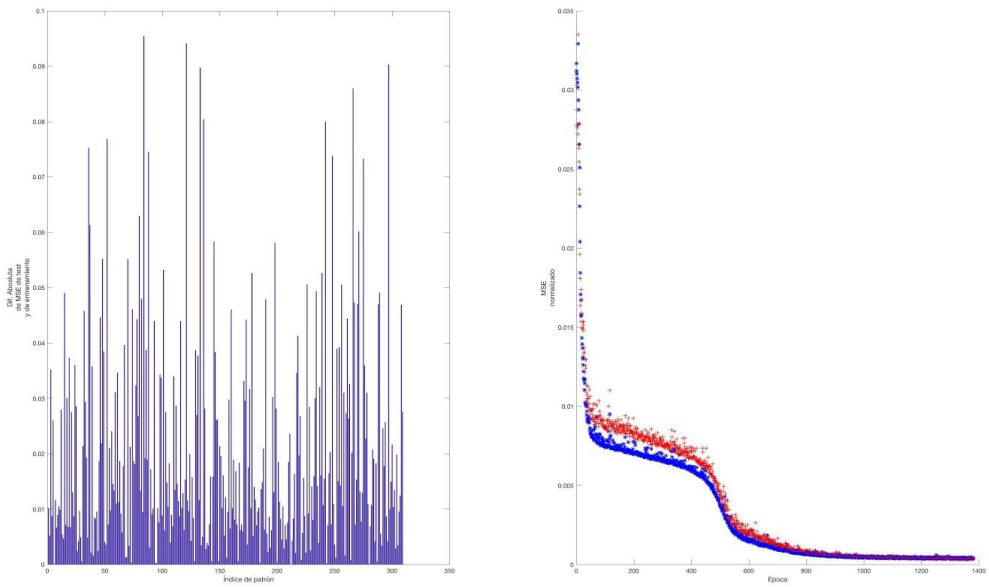

Figura 4: A la izquierda el gráfico que muestra el valor absoluto de la diferencia entre la salida esperada y la salida de las redes al finalizar el entrenamiento. A la derecha el error de entrenamiento corresponde a los puntos azules y el de testeo los puntos rojos

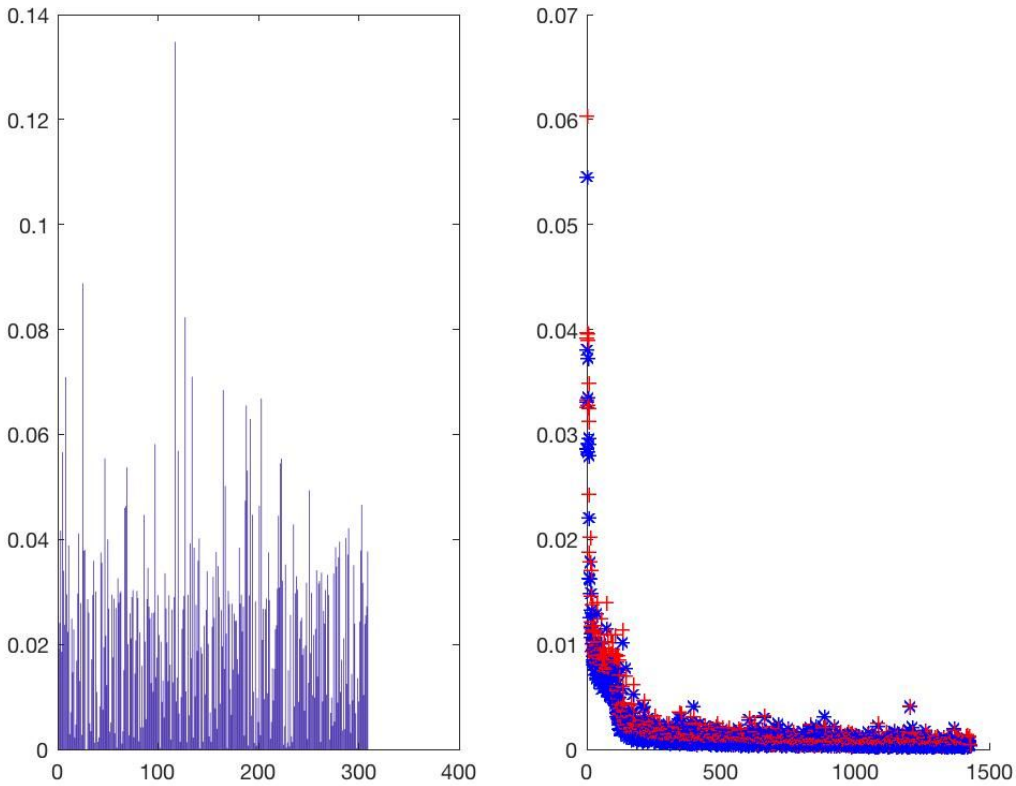

Figura 5: Arquitectura de red [5,5,5] con el error de entrenamiento y testeo a la derecha y el gráfico de barras a la izquierda que representa lo mismo que en la figura anterior.