Sistemas de Inteligencia Artificial
Trabajo Práctico Nº2
Redes Neuronales

Grupo 3

Cavo, María Victoria 53202
Di Nucci, Nicolás Federico 54091
Kljenak, Iván 54019

Lunes 23 de mayo del 2016

Índice

1. Introducción

El objetivo de este trabajo práctico es la realización de una red neuronal con aprendizaje supervisado que se usará en un videojuego para simular un terreno real. La función $f : \Re^2 \rightarrow \Re$ que define el terreno es desconocida pero se cuenta con un muestreo de 441 puntos. En la *figura 1* se puede observar el terreno.

2. Implementación

Para la implementación, se decidió optar por el lenguaje de programación Octave, por su buen rendimiento en la realización de cálculos y operaciones con matrices.

2.1 Perceptrón multicapa

La red neuronal está constituida por un conjunto de neuronas distribuidas en capas, cada una de las neuronas tiene un valor de activación asociado y está conectada a todas las neuronas en la capa siguiente. Cada una de las conexiones entre las neuronas tiene un peso asociado. Una red multicapa, recibe ese nombre porque tiene al menos una de sus capas oculta entre las neuronas de entrada y las de salida.
Representamos al perceptrón multicapa, mediante la estructura *Cell Array* provista por el lenguaje, que define un arreglo de objetos no homogéneos. Esta estructura se usa para almacenar los pesos asociados a las conexiones entre capas.
Para almacenar los pesos correspondientes a una capa (capa *m*) se define una matriz de I^m filas y S^m columnas. La capa *m* contiene una neurona umbral y otras I^m *- 1* neuronas, que se ven representadas en cada una de las filas de la matriz . El parámetro S^m corresponde a la cantidad de neuronas en la siguiente capa, descontando el umbral. Considerando que cada una de las neuronas en la capa *m* está conectada a todas las neuronas de la capa *m + 1*, hay S^m aristas que conectan una neurona de la capa *m* con una de la capa *m + 1*, para cada una de las cuales se usa una columna de la matriz. La idea de la matriz es entonces reflejar en el elemento de la fila *i* y la columna *j* el peso de la conexión entre la neurona *i* de la capa *m* en la neurona *j* de la capa *m + 1*.
La decisión de usar la estructura de Octave obedece a la diferencia de tamaños que puede darse entre las matrices, debido a la diferente cantidad de neuronas en cada capa.
En la *figura 2*, se muestra un ejemplo de la representación de la red neuronal.

2.2 Funcionamiento de la red

El objetivo de la red neuronal es que esta reciba valores de entrada ξ_i representando patrones para los cuales se espera un resultado determinado, y luego, tras una serie de cálculos, los valores obtenidos a la salida de la red sean tan aproximados como sea posible a los valores esperados.
Para los cálculos involucrados es necesario definir los siguientes conceptos:

- a_i^m *:* valor de activación de la neurona *i* de la capa *m*.
- w_{ij}^m *:* peso de la arista entre la neurona *i* de la capa *m* y la neurona *j* de la capa *m + 1*
- *g*: *función de activación*
- h_i^m : potencial de activación de la neurona *i* de la capa *m*

Para las neuronas en la primera capa, se asignan los parámetros iniciales a los valores de activación

$$a_i^0 = \xi_i \; y \; a_0^0 = -1 \; .$$

Para el resto de las neuronas se calcula en principio su potencial de activación, como la suma de los productos de los valores de activación de las neuronas en la capa previa y el peso de la conexión entre estas y la neurona en cuestión. Luego, se aplica la función de activación sobre este resultado.

$$a_i^m = g(h_i^m) = g(\sum_{j=0}^{I^m} w_{ji}^m . a_j^{m-1}) \; con \; i = \{1, ..., S^m\} \; y \; a_0^m = -1$$

En esta implementación, se utilizaron dos funciones de activación diferentes: una función sigmoide, la tangente hiperbólica y una función logística, la exponencial.

- $g(x) = tanh(\beta x)$
- $g(x) = (1 + e^{-2\beta x})^{-1}$

siendo β una constante que es recibida como parámetro.

2.3 Aprendizaje

El aprendizaje consiste en dos etapas: *feed forward* y *backpropagation.* La primera etapa ha sido descrita en el apartado Funcionamiento de la red. En esta instancia, se calculan los valores de salida de cada capa para cada patrón de entrada.

La siguiente etapa tiene el objetivo de corregir los posibles errores y refinar los resultados. Es necesario definir, en principio, como calcular el error. En el presente trabajo se optó por usar como función de costo el cálculo del error cuadrático medio:

$$E(W) = \sum_{i\mu} (S_i^\mu - o_i^\mu)^2$$

siendo S^μ la capa de salida deseada y o_i^μ la capa de salida obtenida para el patrón de entrada μ.

En base al error calculado, es necesario definir un método que indique cómo realizar una corrección de los pesos de las aristas que apunte a reducir tal error. En este caso, se optó por el método del gradiente descendiente que es un algoritmo iterativo que define las variación del peso que es necesario aplicar a una conexión, como la derivada del error respecto de ese peso.

$$\Delta w_{ij} = -\eta \, \partial E / \partial w_{ij} = \eta \delta_i^\mu \xi_j^\mu$$

siendo η una constante recibida como parámetro.

La etapa de *backpropagation* consiste en recorrer las capas en el sentido inverso (de la salida hacia la entrada), para cada uno de los patrones de entrada, calculando para cada arista su correspondiente variación, que eventualmente será utilizado para recalcular el peso de esa arista.

Fueron utilizados dos métodos diferentes de aprendizaje supervisado, que difieren en el momento en que son modificados los pesos de las aristas:

- Método incremental: Se actualiza el peso de todas las aristas luego del análisis de cada patrón.
- Método batch: durante una época (análisis de todos los patrones de entrada) se acumulan las variaciones de los pesos para cada patrón y al final de la misma, se realiza la actualización de los pesos.

2.4 Variaciones de backpropagation

Debido a la lentitud del proceso de backpropagation se implementaron una serie de modificaciones al algoritmo con el objetivo de optimizar el funcionamiento y lograr un aprendizaje más rápido. Por otra parte, estas variaciones ayudan a prevenir el estancamiento del aprendizaje en mínimos locales

2.4.1 Momentum

Si el valor del parámetro η mencionado anteriormente es lo suficientemente chico, el método del gradiente descendiente puede ser demasiado lento; por otro lado, si en cambio su valor es lo bastante grande, esto induce oscilaciones, es decir, no hay una convergencia al resultado esperado.

La idea del momentum es darle a cada conexión una cierta inercia para que cambie en la dirección del descenso promedio. Esto se logra adicionando al cálculo de la variación del peso de una arista una proporción de la variación calculada para la misma arista en el análisis del patrón anterior en el caso de estar usando el método incremental, o en la época anterior si se optó por utilizar el método incremental.

$\Delta w_{pq}(t+1) = -\eta\, \partial E/\partial w_{pq} + \alpha\, \Delta w_{pq}(t)$ con $0 < \alpha < 1$ y siendo generalmente 0.9.

2.4.2 Parámetros adaptativos

Como ya definimos previamente, existe un parámetro η que modifica el desempeño de la red durante el aprendizaje. Resulta complejo identificar un valor para este parámetro que aplique a todos los problemas posibles, y por otro lado, lo que resulta conveniente en una etapa del aprendizaje, puede no serlo tanto en otra. El objetivo de esta variación es adaptar el valor de este parámetro a lo largo de la ejecución de acuerdo a la instancia en la que se encuentre.

En el caso del método incremental, luego de finalizar la etapa de backpropagation de un patrón se compara el resultado de la función de costo con el obtenido en el paso anterior. Si luego de haber analizado *k* patrones, el error disminuyó en cada una de las comparaciones, el valor de η se aumenta en el valor de la constante *a*, que es recibida como parámetro. A partir de ese momento, el valor de η es aumentado tras cada patrón en ese mismo valor, mientras el error siga disminuyendo, debido a que se asume que el valor óptimo de η es mayor al que está siendo usado. Si en algún momento la diferencia entre el error del paso actual y el anterior indica un aumento, se disminuye el valor de η. La disminución se da en una proporción del valor de η, siendo otro de los valores recibidos como parámetro (*b*), la constante de proporcionalidad.

El método batch es similar, solo que se realiza la comparación del error luego de haberse completado una época, es decir, de haber analizado todos los patrones.

No obstante eso, atendiendo a la necesidad de continuar sin realizar cambios aún cuando el error aumente, se contempló la posibilidad de agregar un parámetro que decida, de acuerdo a una probabilidad dada, si se realiza el cambio en η o se ignora la diferencia en el error.

Cuando el error aumenta, para evitar impulsar el aprendizaje en una dirección errónea, es necesario suspender la aplicación del momentum hasta tanto la ejecución vuelva a encauzarse en una camino de disminución del error.

$$\Delta\eta = \begin{cases} +a & si \quad \Delta E < 0 \quad consistentemente \\ -b\eta & si \quad \Delta E \geq 0 \\ 0 & si\ no \end{cases}$$

2.5.3 Inicialización de los pesos

Otra variación implementada, está dada en los pesos iniciales. Inicialmente, las conexiones reciben un peso aleatorio dentro de un rango recibido como parámetros. La particularidad de este rango debe ser tal que evite la saturación de las neuronas, si los valores son demasiado grandes, puede ocurrir que el potencial de activación sea un valor tal que al serle aplicada la función de activación, siempre de valores extremos. Por ejemplo, en el caso de usar la tangente hiperbólica, estos valores serían 1 y -1. De darse este caso, el sistema podría quedar atrapado en un mínimo local.

Se intentó apelar a una estrategia que teóricamente podría funcionar mejor para elegir los pesos de manera aleatoria, que consiste en tomar como w_{ij} un valor del orden de $1/\sqrt{k_i}$ donde k_i es la cantidad de neuronas que tienen conexión con la neurona *i*. No obstante eso, tras probarlo prácticamente, no se obtuvieron los resultados esperados, y se decidió que los pesos sean seleccionados aleatoriamente en un rango entre -0.5 y 0.5.

3. Pruebas realizadas

En simultáneo a la ejecución del entrenamiento, se realiza un gráfico que muestra la variación del error, paso tras pasos, y en el caso, de usar parámetros adaptativos, también se grafican las variaciones en el parámetro **η**.

Cuando culmina la ejecución, se muestra un gráfico que compara el terreno esperado con el simulado por la red neuronal. Además, un mismo gráfico permite comparar los errores de entrenamiento y testeo. A modo de ejemplo, las figuras 7, 8, 9 y 10 muestran los gráficos mencionados.

Se procedió a la realización de diferentes pruebas orientadas a analizar varios aspectos relevantes de la red neuronal implementada.

En principio, habiendo desarrollado dos métodos de aprendizaje diferentes, incremental y batch, se buscó comparar el rendimiento de cada uno. El propósito fue entender las diferencias en tiempos de ejecución como una medida de la eficiencia de cada método. Los resultados indicaron una clara ventaja del método batch por sobre el incremental. Los razón de este comportamiento radica en que el método incremental realiza una serie de operaciones matriciales al analizar cada uno de los patrones de entrenamiento, mientras que el método batch sólo realiza esta operaciones al final de cada época. Siendo menor la cantidad de cálculos realizados, es menor el esfuerzo requerido y por lo tanto, es menor el tiempo insumido.

A continuación, se realizaron una serie de pruebas empíricas variando en los parámetros recibidos por el sistema que han sido mencionados a lo largo del presente informe:

- β : la constante propia de las funciones de activación
- α : la constante utilizada en la implementación del momentum
- η : constante utilizada en el cálculo de la variación del peso de las aristas
- a, b: las constantes del método que permite adaptar el valor de η

Al haber concluido esta instancia, se obtuvieron valores de las constantes considerados aceptables para ser usados en las pruebas subsiguientes. Dichos valores fueron para el caso del método batch $a = 0.002$, $b = 0.1$, $\alpha = 0.9$, $\beta = 0.5$, $\eta = 0.002$. En el caso del método incremental, los valores son $a = 0.02$, $b = 0.1$, $\alpha = 0.9$, $\beta = 0.5$, $\eta = 0.02$. Por otra parte, se definió en esta instancia un criterio para identificar cuándo concluye una etapa de entrenamiento. El entrenamiento concluye una vez que el error es menor a una cota definida o si independientemente del error transcurren una cierta cantidad de épocas de entrenamiento. En este último caso, el estado final de la red corresponde al estado en el momento en que el mínimo error fue medido.

El siguiente paso fue la evaluación de ambos métodos (incremental y batch). Con parámetros fijos, la intención fue comparar las distintas variantes de backpropagation explicadas. Así, se ejecutaron rondas de entrenamiento sin momentum ni parámetros adaptativos, con cada uno de estos activados por separado y, finalmente, con ambos en simultáneo. En todos los casos, las pruebas fueron realizadas usando tanto la función de evaluación sigmoidea como la logística, en una red neuronal con cuatro capas, con dos neuronas en la capa de entrada, siete en cada una de las intermedias y una neurona en la capa de salida. Los resultados de esta prueba pueden verse en la sección anexa en la figura 11.

Luego, se procuró analizar las diferencias en el proceso de aprendizaje al alterar la arquitectura de la red neuronal. Una vez con los valores de las constantes obtenidos en las pruebas anteriores, se ejecutaron entrenamientos variando la cantidad de capas ocultas en la red y la cantidad de neuronas en cada capa. También en estos casos se utilizaron los dos métodos de aprendizaje. La figura 12, en el anexo del presente documento, muestra los datos obtenidos

Finalmente, se quiso investigar el porcentaje de generalización. Es decir, luego de la etapa de entrenamiento, se alimentó a la red neuronal con conjuntos de pruebas, para comparar la salida de la red con los valores esperados. Para calcular el éxito del aprendizaje, se calcula la resta de los valores esperados y los obtenidos, normalizados. Si esta resta, elevada al cuadrado y dividida por dos, es menor a 0.01, el aprendizaje se considera exitoso. Estas pruebas fueron realizadas aplicando las condiciones que en pruebas previas resultaron más eficientes. En este caso, el método elegido fue batch y la función de activación, la tangente hiperbólica.
Se realizaron pruebas con distintos conjuntos de entrenamiento y testeo. En principio se dividió el terreno a la mitad y se tomaron los puntos correspondientes a una de las mitades como conjunto de entrenamiento, y los correspondientes a la otra mitad como conjuntos de pruebas. En otras pruebas, se consideraron a los puntos con

un orden en filas y columnas y alternativamente cada una de las filas o columnas eran agregadas a los conjuntos de entrenamiento y prueba. Las figuras 3, 4, 5 y 6 muestran los puntos incluidos en los conjuntos de entrenamiento y los incluidos en los conjuntos de pruebas. Las figuras 13 y 14 ilustran los resultados conseguidos para dos arquitecturas diferentes.

4. Conclusiones

Luego de las pruebas realizadas se concluye que por motivos de implementación el aprendizaje es significativamente más rápido usando el método batch que usando el incremental. Como se mencionó previamente, esto obedece a la diferencia en la cantidad de cálculos y operaciones matriciales realizadas en cada caso.

También se encontraron evidencias de que utilizando la tangente hiperbólica como función de activación se obtienen mejores resultados que al usar la función exponencial. Asimismo las pruebas permiten concluir que utilizando el método batch con la adición de la estrategia del momentum, el aprendizaje de la red es el más cercano a lo óptimo.

En cuanto a las arquitecturas, la comparación permite ver significativas mejoras entre redes con una sola capa oculta frente a las que tienen dos capas ocultas. La comparación entre redes con dos y tres capas ocultas, en cambio, no arroja mayores diferencias.

Con respecto a la generalización, tal como era esperable, la división del terreno en mitades no produjo resultados útiles. Esto se debe a que la red neuronal no puede tener conocimientos sobre la segunda mitad del terreno, no habiendo sido entrenada con datos de este sector, con lo cual la generalización se torna imposible. Sin embargo si se aprecia la generalización para la segunda definición de los conjuntos de entrenamiento y prueba.

5.Anexo

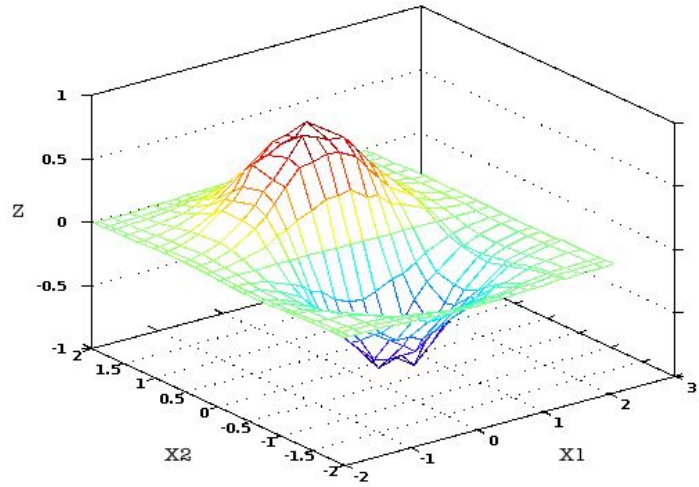

Figura 1: Aproximación del terreno del videojuego a partir del muestreo otorgado.

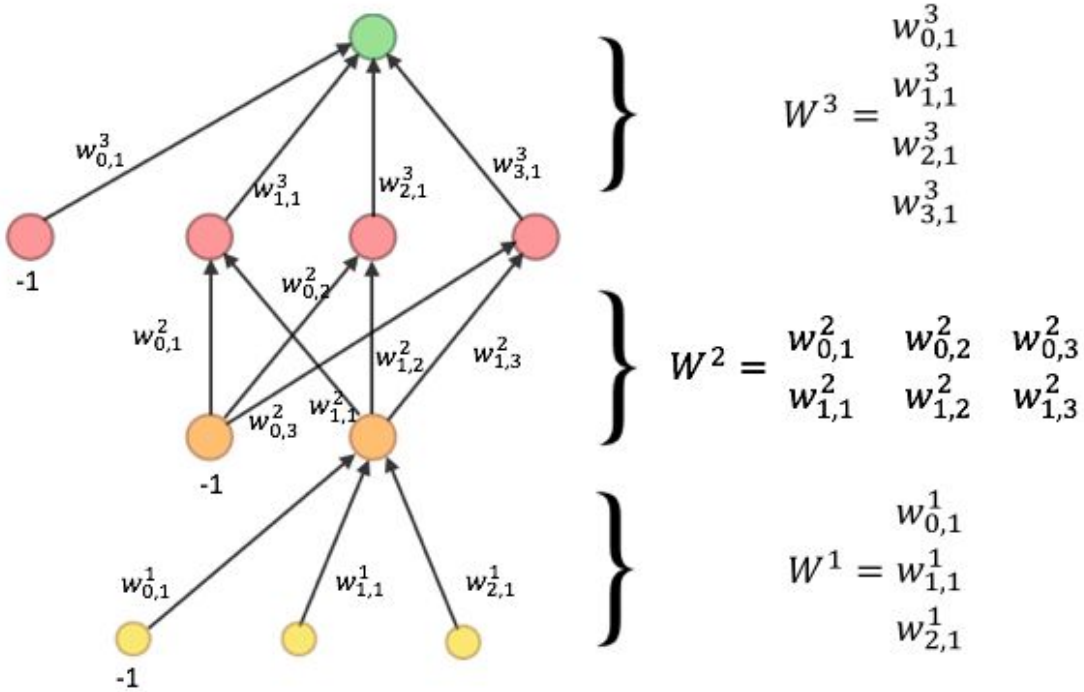

Figura 2:Representación de una red neuronal de arquitectura [2 1 3 1].

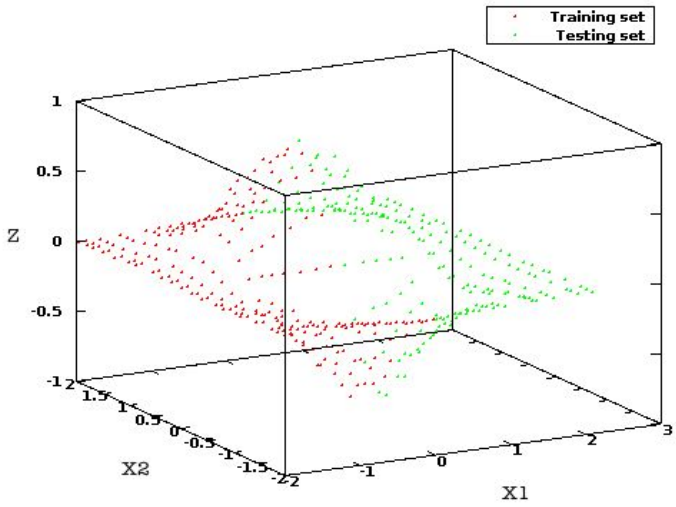

Figura 3: Half1.

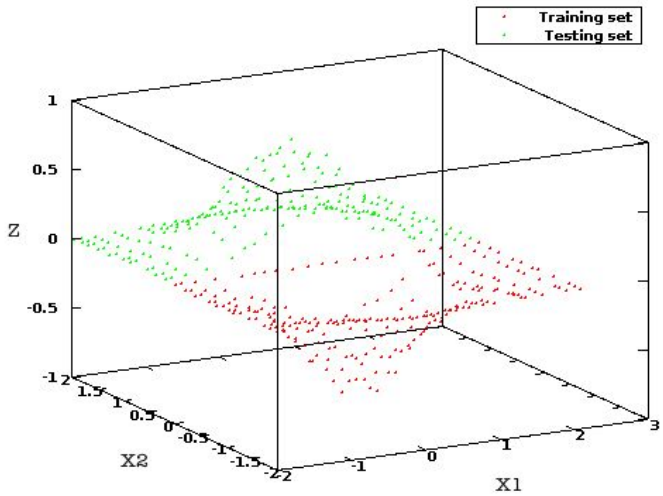

Figura 4: Half2.

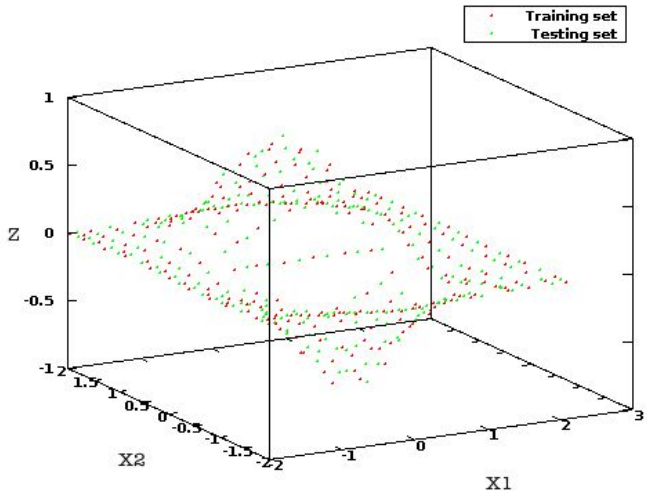

Figura 5: skipping1

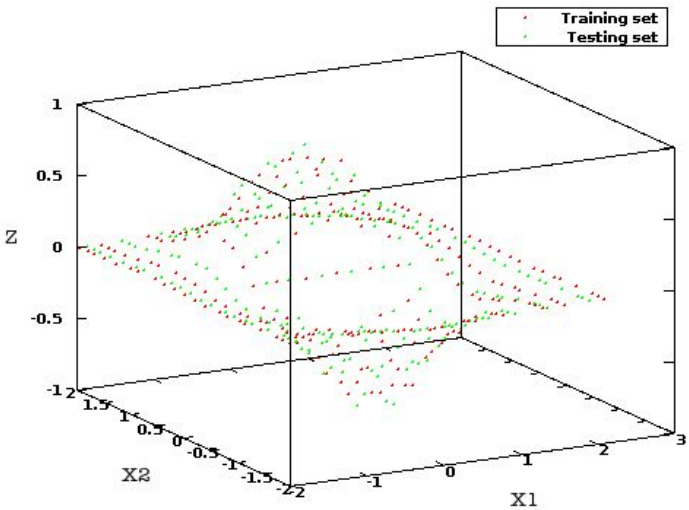

Figura 6: skippingrows

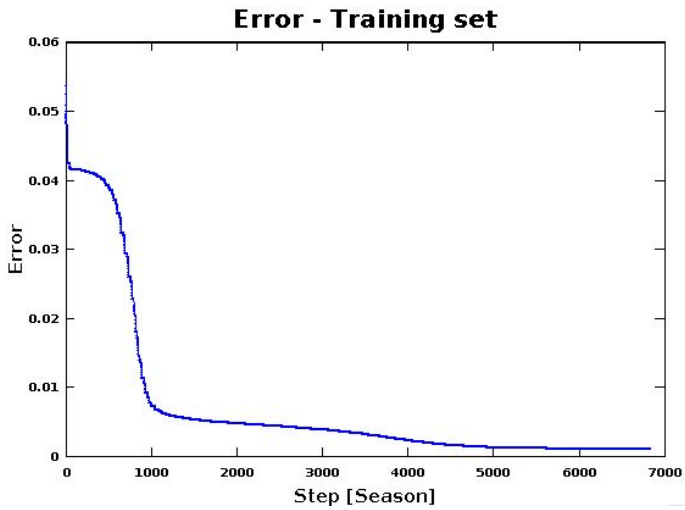

Figura 7: Error del conjunto de entrenamiento en el tiempo.

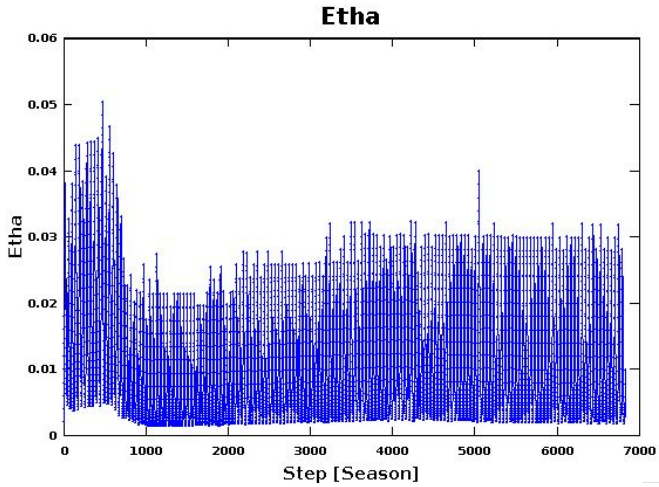

Figura 8: Valor de etha en el tiempo (etha adaptativo).

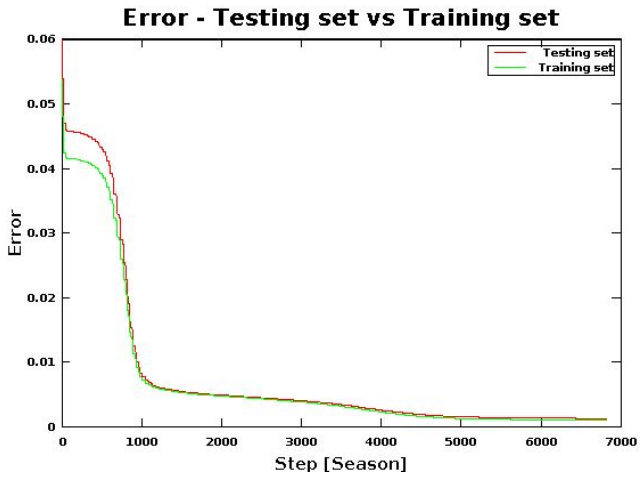

Figura 9: Comparación del error del conjunto de entrenamiento con el de testeo en el tiempo.

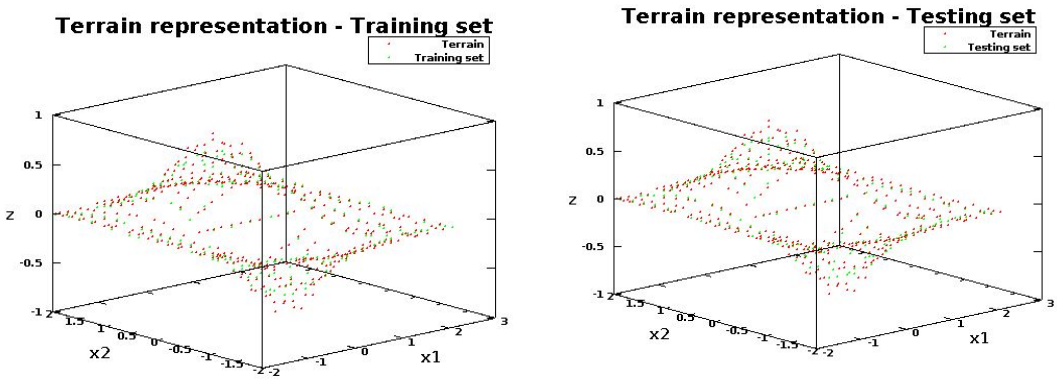

Figura 10: Visualización de la salida esperada comparada con la obtenida.

	Error cuadrático medio	Épocas	Patrones	Tiempo[sec]	Función de activación	Momentum	Etha adaptativo	% aprendido
Batch	**0,001000**	27752	12238632	99,07	tanh	no	no	95,05
	0,001000	**9964**	**4394124**	**35,19**	tanh	si	no	**95,28**
	0,001000	23560	10389960	85,73	tanh	no	si	94,78
	0,001000	12378	5458698	**45,17**	tanh	si	si	95,23
	0,025972	500000	220500000	2269,76	exp	no	no	77,82
	0,025912	500000	220500000	2415,87	exp	si	no	78,91
	0,025781	500000	220500000	2465,16	exp	no	si	65,98
	0,025765	500000	220500000	2423,27	exp	si	si	79,36
Incremental	0,042989	1000	441000	2970,71	tanh	no	no	33,33
	0,040050	1000	441000	2985,45	tanh	si	no	29,93
	0,044187	1000	441000	2978,00	tanh	no	si	40,59
	0,043600	1000	441000	2984,91	tanh	si	si	32,43
	0,053550	1000	441000	3164,47	exp	no	no	54,2
	0,053168	1000	441000	3323,19	exp	si	no	55,1
	0,052949	1000	441000	3302,66	exp	no	si	55,1
	0,043171	1000	441000	3217,89	exp	si	si	42,4

Figura 11: Comparación entre los métodos batch e incremental

	Error cuadrático medio	Épocas	Tiempo[sec]
[2 7 1]	0,002203	500000	1317,59
[2 10 1]	0,001000	319533	1227,03
[2 13 1]	0,001000	332673	1402,09
[2 16 1]	0,001000	133860	631,53
[2 7 7 1]	0,001000	7031	35,62
[2 10 10 1]	0,001000	7052	43,69
[2 13 13 1]	0,001000	12712	92,07
[2 16 16 1]	0,001000	6360	52,4
[2 7 7 7 1]	0,001000	7254	43,57

Figura 12: Comparación entre arquitecturas

	Error cuadrático medio del conj de entrenamiento	Error cuadrático medio del conj de testeo	Épocas	Patrones	Tiempo [sec]	% Aprendido del conj. de entrenamiento	% Aprendido del conj. de testeo
skipping1	0,001	0,001166	29828	6591988	75,24	95,02	94,55
skippingrows	0,001	0,001508	41168	9509808	109,37	94,81	92,86
half1	0,001	0,08	19051	4400781	52,21	94,37	32,38
half2	0,001	0,175209	6542	1511202	18,02	96,1	15,71

Figura 13: Evaluación de generalización. Arquitectura de la red [2 7 7 1]

	Error cuadrático medio del conj de entrenamiento	Error cuadrático medio del conj de testeo	Épocas	Patrones	Tiempo [sec]	% Aprendido del conj. de entrenamiento	% Aprendido del conj. de testeo
skipping1	0,001	0,001134	15351	3392571	66,67	94,57	93,63
skippingrows	0,001	0,001489	17583	4061673	75,69	93,5	92,85
half1	0,001	0,079665	9289	2145759	37,02	93,93	32,38
half2	0,001	0,198496	9106	2103486	36,43	95,67	8,57

Figura 14: Evaluación de la generalización. Arquitectura de la red [2 16 16 1]