

Control Inteligente

Diseño de Redes Neuronales con Matlab L8

Luis Moreno, Santiago Garrido, Dorin Copaci

Dpto. Ing. de Sistemas y Automática
Universidad Carlos III
Madrid

Oct. 2019



Introducción a las redes neuronales con Matlab

- En Matlab (en este curso 2018) se dispone de la Neural Network Toolbox que incluye un conjunto de funciones que permiten implementar redes neuronales. Su uso permite, principalmente, realizar las siguientes tareas:
 - Identificación de funciones.
 - Reconocimiento de patrones.
 - Agrupamiento de datos.

Neural Network Toolbox

Identificación de funciones

- Las redes neuronales son una buena herramienta para identificar funciones. Supongamos para ello un ejemplo que consiste en estimar la grasa corporal de alguien a partir de diversas mediciones **[bodyfat]**. Se busca una red neuronal que sea capaz de predecir el nivel de grasa corporal a partir de 13 datos de entrada como edad, peso, altura etc.. Se dispone de un total de 252 ejemplos con el valor de los 13 parámetros y su valor.
- El Toolbox te permite resolver el problema por tres vías distintas:
 - ① Llamadas a funciones (desde línea de comandos).
 - ② Herramienta gráfica *nftool*.
 - ③ Herramienta gráfica *nntool*.

Identificación de funciones: línea de comandos

- 1. Cargar los datos, que consisten en una matriz de datos de entrada (`bodyfatInputs`), con 13 filas correspondientes a los factores considerados y 252 datos de ejemplo, y un vector de salida (`bodyfatTargets`) con 252 componentes que representa el nivel de grasa corporal.

» `load bodyfat_dataset;`

- 2. Posteriormente se crea la red neuronal. Para este ejemplo se usa la función `newfit`. Se usa una red *feedforward* on funciones de activación *tan-sigmoid* en la capa oculta y lineal en la capa de salida (configuración por defecto). Esta estructura es útil para este tipo de problemas.

- Se usarán 40 neuronas (por ejemplo) en una única capa oculta.
- La red tendrá una neurona de salida porque sólo hay un precio asociado al vector de entrada.

» `net = newfit(bodyfatInputs, bodyfatTargets, 40);`

Identificación de funciones: línea de comandos

- 3. Entrenar la red. Por defecto se usa el algoritmo de Levenberg-Marquardt. Este algoritmo divide los datos disponibles en tres grupos:
 - 60 %: Usados para entrenar la red.
 - 20 %: Usados para validar la red y detener el entrenamiento cuando haya *overfitting*.
 - 20 %: Usados para testear la red.
- » *net=train(net,bodyfatInputs,bodyfatTargets);*

Identificación de funciones: línea de comandos

- Se abre la siguiente ventana con el estado de proceso. Permite interrumpir el entrenamiento tecleando *Stop Training*.

The screenshot shows the 'Neural Network Training (ntraintool)' window. At the top, there is a diagram of a neural network with an 'Input' block, two 'Layer' blocks, and an 'Output' block. Each layer contains a weight matrix 'W', a bias vector 'b', an addition node '+', and an activation function block.

Below the diagram, the 'Algorithms' section is configured as follows:

- Training: Levenberg-Marquardt (trainlm)
- Performance: Mean Squared Error (mse)
- Data Division: Random (dividerand)

The 'Progress' section displays the following metrics:

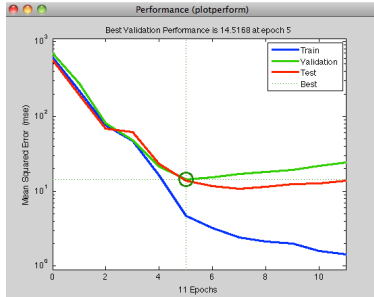
Epoch:	0	11 iterations	1000
Time:		0:00:01	
Performance:	602	4.70	0.00
Gradient:	1.00	21.0	1.00e-10
Mu:	0.00100	10.0	1.00e+10
Validation Checks:	0	6	6

The 'Plots' section includes buttons for 'Performance' (plotperform), 'Training State' (plottrainstate), 'Fit' (plotfit), and 'Regression' (plotregression). A 'Plot Interval' slider is set to 1 epochs.

At the bottom, there is a green checkmark icon and the text 'Opening Regression Plot'. Two buttons, 'Stop Training' and 'Cancel', are located at the bottom right.

Identificación de funciones: línea de comandos

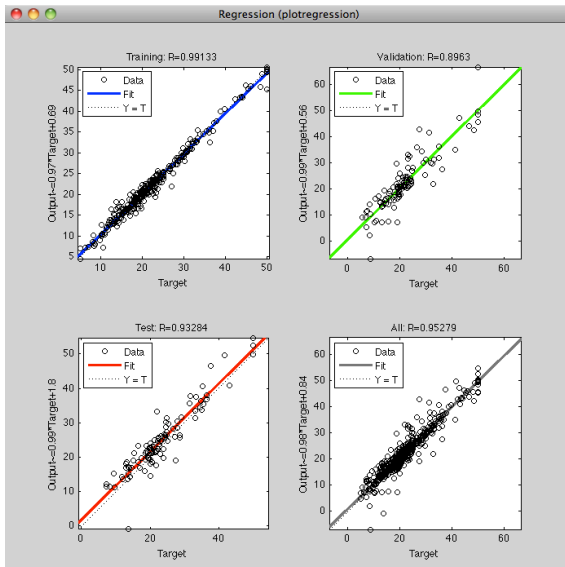
- En este ejemplo usa el método *train* para el entrenamiento. También se puede usar el método *adapt*. Consultar la ayuda sobre *Training Styles* para mas información.
- El entrenamiento termina cuando el error de validación no mejora durante 6 iteraciones, lo que ocurre en la iteración 11. Si se abre la ventana *Performance* se puede ver un gráfico con el error de entrenamiento, validación y test.



Identificación de funciones: línea de comandos

- El error final es pequeño.
- Los errores de validación y test son similares.
- No hay un *overfitting* significativo después de la iteración 5 (cuando se llega al mejor error de validación).
- 4. En este punto se analiza la respuesta de la red neuronal. Si se teclea *Regression* en la ventana de entrenamiento, se efectúa una regresión lineal entre las salidas de la red y sus correspondientes valores objetivo. Se abre la figura de la siguiente transparencia.

Identificación de funciones: línea de comandos



Identificación de funciones: línea de comandos

- La salida sigue muy bien los objetivos para entrenamiento, validación y test, y el valor R está por encima de 0,95 para la respuesta total.
- Si se quisieran obtener mejores resultados se puede probar con lo siguiente:
 - Reiniciar la red y entrenarla de nuevo con los valores ya obtenidos como pesos y offsets de entrada.
 - Incrementar las neuronas de la capa oculta.
 - Incrementar las muestras de entrenamiento.
 - Incrementar las entradas si hay más información disponible que sea significativa.
 - Probar un algoritmo de entrenamiento diferente.

Identificación de funciones: herramienta gráfica *nftool*.

- Una forma alternativa para implementar redes neuronales es mediante la Neural Network Fitting Tool GUI.
- 1. Para abrir esta herramienta se usa el comando *nftool*.

Welcome to the Neural Network Fitting Tool.
Solve an input-output fitting problem with a two-layer feed-forward neural network.

Introduction

In fitting problems, you want a neural network to map between a data set of numeric inputs and a set of numeric targets.

Examples of this type of problem include estimating house prices from such input variables as tax rate, pupil/teacher ratio in local schools and crime rate (`house_dataset`); estimating engine emission levels based on measurements of fuel consumption and speed (`engine_dataset`); or predicting a patient's bodyfat level based on body measurements (`bodyfat_dataset`).

The Neural Network Fitting Tool will help you select data, create and train a network, and evaluate its performance using mean square error and regression analysis.

Neural Network

A two-layer feed-forward network with sigmoid hidden neurons and linear output neurons (`newfit`), can fit multi-dimensional mapping problems arbitrarily well, given consistent data and enough neurons in its hidden layer.

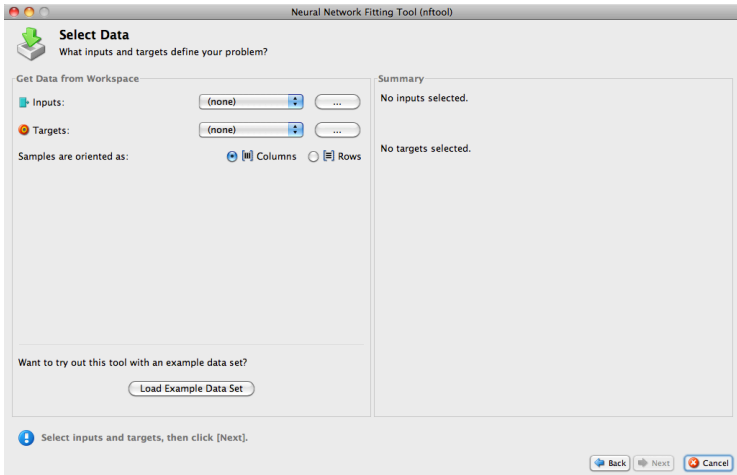
The network will be trained with Levenberg-Marquardt backpropagation algorithm (`trainlm`), unless there is not enough memory, in which case scaled conjugate gradient backpropagation (`trainscg`) will be used.

➔ To continue, click [Next].

⏪ Back ⏩ Next ⏹ Cancel

Identificación de funciones: herramienta gráfica *nftool*.

- 2. Teclear *Next*.



Identificación de funciones: herramienta gráfica *nftool*.

- 3. Teclar *Load Example Data Set*.

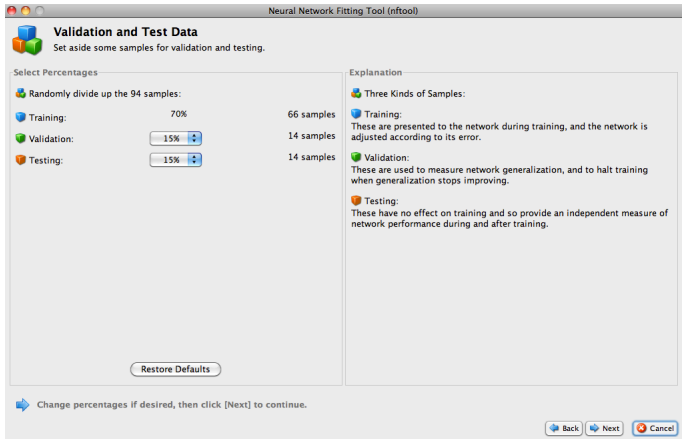
Fitting Data Set Chooser

Select a data set:	Description
<ul style="list-style-type: none">Simple Fitting ProblemAbalone RingsBody FatBuilding EnergyChemicalEngine	<p>Filename: simplefit_dataset</p> <p>Function fitting is the process of training a neural network on a set of inputs in order to produce an associated set of target outputs. Once the neural network has fit the data, it forms a generalization of the input-output relationship and can be used to generate outputs for inputs it was not trained on.</p> <p>This dataset can be used to demonstrate how a neural network can be trained to estimate the relationship between two sets of data.</p> <p>LOAD simplefit_dataset.MAT loads these two variables:</p> <p>simplefitInputs - a 1x94 matrix defining 94 input values.</p> <p>simplefitTargets - a 1x94 matrix defining 94 associated target values.</p> <p>[X,T] = simplefit_dataset loads the inputs and targets into variables of your own choosing.</p> <p>To solve this problem with the Neural Fitting app click "Load Example Data Set" in the "Select Data" panel and pick this dataset.</p>

Import Cancel

Identificación de funciones: herramienta gráfica *nftool*.

- 4. Seleccionar *Simple Fitting Problem*, y teclear *Import*. Esto nos devuelve a la ventana de selección de datos.
- 5. Teclear *Next* para pasar a la ventana *Validate and Test Data*. Se toma un 15 % de los datos originales para validación y test.



Identificación de funciones: herramienta gráfica *nftool*.

- 6. Teclar *Next*. El número de neuronas de la capa oculta se fija a 20. Se puede cambiar si la red no funciona de acuerdo a lo esperado.

Neural Network Fitting Tool (nftool)

Network Size

Set the number of neurons in the fitting network's hidden layer.

Hidden Layer

Number of Hidden Neurons:

Recommendation

Return to this panel and change the number of neurons if the network does not perform well after training.

Restore Defaults

Neural Network

Input: 1

Hidden Layer: 20

Output Layer: 1

Output: 1

Change the number of neurons if desired, then click [Next] to continue.

Back Next Cancel

Identificación de funciones: herramienta gráfica *nftool*.

- 7. Teclear *Next*.

The screenshot shows the 'Neural Network Fitting Tool (nftool)' window. The main title is 'Train Network' with a sub-header 'Train the network to fit the Inputs and targets.' Below this, the 'Train Network' section indicates 'Train using Levenberg-Marquardt backpropagation (trainlm)' and features a 'Train' button. A note explains that training stops when generalization stops improving. The 'Results' section displays a table with columns for Samples, MSE, and R, and rows for Training, Validation, and Testing. Below the table are 'Plot Fit' and 'Plot Regression' buttons. A 'Notes' section provides details on MSE and R values. At the bottom, a help icon and text prompt the user to 'Train network, then click [Next]'. Navigation buttons for 'Back', 'Next', and 'Cancel' are located in the bottom right corner.

Train Network
Train the network to fit the Inputs and targets.

Train Network
Train using Levenberg-Marquardt backpropagation (trainlm).

Training automatically stops when generalization stops improving, as indicated by an increase in the mean square error of the validation samples.

Results

	Samples	MSE	<input checked="" type="checkbox"/> R
Training:	66	-	-
Validation:	14	-	-
Testing:	14	-	-

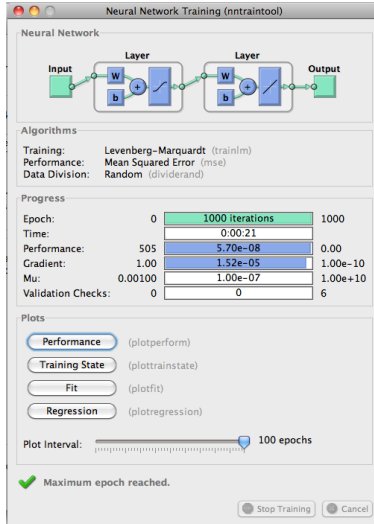
Notes

- Training multiple times will generate different results due to different initial conditions and sampling.
- Mean Squared Error is the average squared difference between outputs and targets. Lower values are better. Zero means no error.
- Regression R Values measure the correlation between outputs and targets. An R value of 1 means a close relationship, 0 a random relationship.

Train network, then click [Next].

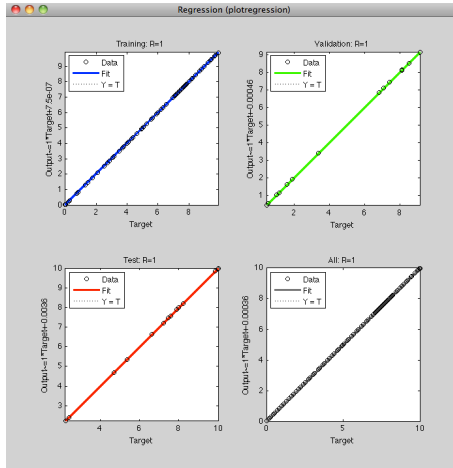
Identificación de funciones: herramienta gráfica *nftool*.

- 8. Teclear *Train*. El entrenamiento continúa hasta un máximo de 1000 iteraciones.



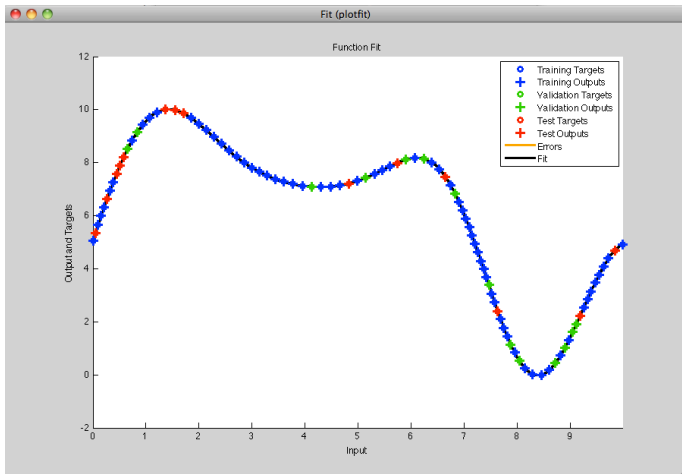
Identificación de funciones: herramienta gráfica *nftool*.

- 9. Dentro de *Plots*, seleccionar *Regression*. Para este ejemplo simple el ajuste es casi perfecto para entrenamiento, validación y test.



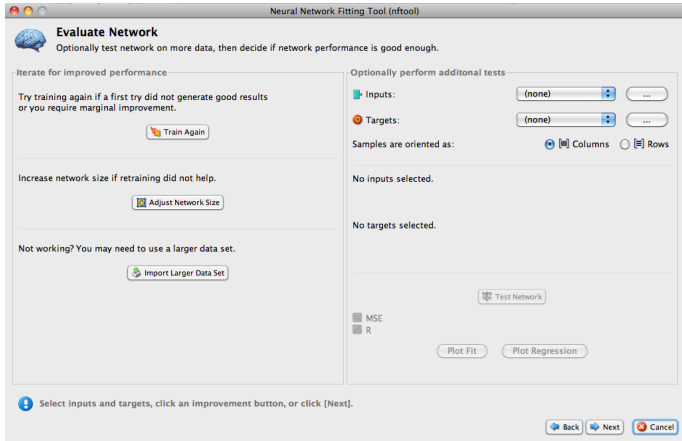
Identificación de funciones: herramienta gráfica *nftool*.

- 10. Observar la respuesta de la red. Para problemas con una entrada y una salida se puede ver seleccionando *Fit* dentro de *Plots*.



Identificación de funciones: herramienta gráfica *nftool*.

- 11. Pulsar *Next* en *Neural Network Fitting Tool* para evaluar la red.

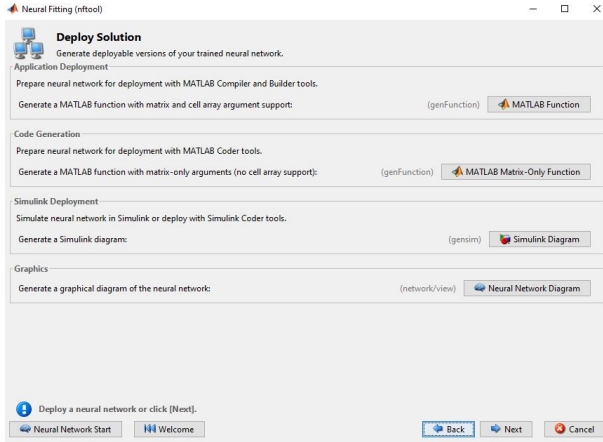


Identificación de funciones: herramienta gráfica *nftool*.

- En este punto es posible probar la red utilizando nuevos datos.
- Si no se está satisfecho con los resultados obtenidos se puede hacer lo siguiente:
 - Entrenar de nuevo la red.
 - Incrementar el número de neuronas.
 - Utilizar un conjunto de datos de entrenamiento mayor.
- 12. Si los resultados obtenidos son los adecuados, pulsar *Next*.

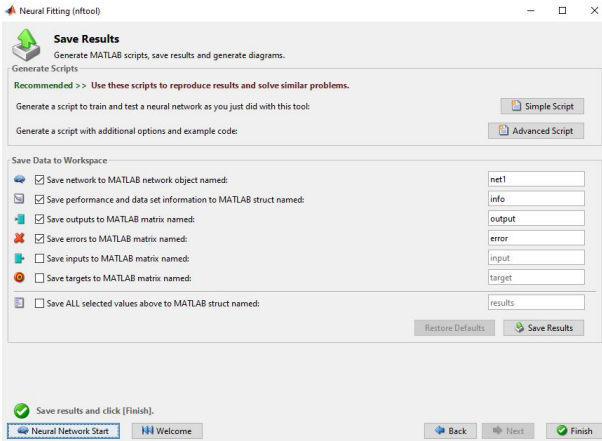
Identificación de funciones: herramienta gráfica *nftool*.

- 13. Utilizar los botones de la siguiente pantalla para exportar la red.



Identificación de funciones: herramienta gráfica *nftool*.

- 14. Utilizar los botones de la siguiente pantalla para guardar los resultados.



Identificación de funciones: herramienta gráfica *nftool*.

- Se tiene la red guardada como *net1* en el workspace. Se pueden realizar test adicionales en la red o utilizarla con nuevas entradas utilizando la función *sim*.
- También se puede crear un **M-file** que puede usarse para reproducir todos los pasos anteriores desde la línea de comandos. Crear un M-file puede ser útil si se quiere aprender como utilizar las funciones de la toolbox que se ejecutan desde línea de comandos para personalizar el proceso de entrenamiento.
- También se puede exportar la red a simulink.
- Una vez se hayan guardado los resultados, pulsar *Finish*.

Bibliografía



Matlab, Neural Net Fitting tool.

https://es.mathworks.com/help/deeplearning/ref/nftool.html?s_tid=doc_ta,

Fecha de último acceso Oct 2019

- Fin L8