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*.

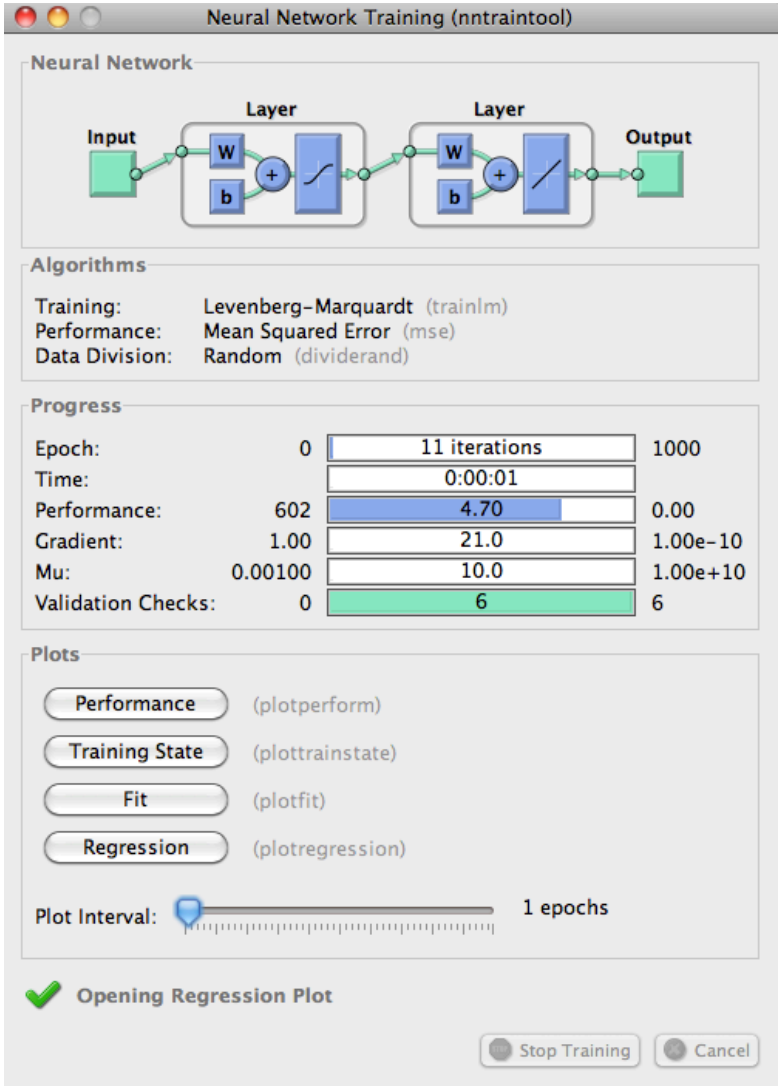
- 1. Cargar los datos, que consisten en una matriz de datos de entrada (bodyfatInputs), con 13 filas correspondientes a los factores considerados y 252 datos de ejemplo, y un vector de salida (bodyfatTargets) con 252 componentes que representa el nivel de grasa corporal.
  »load bodyfat_dataset;
- 2. 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);

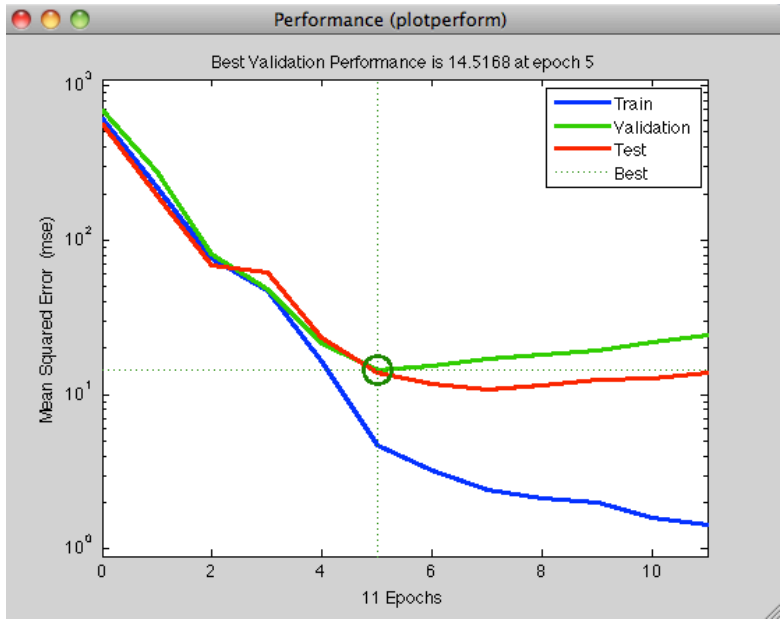
- 3. Entrenar la red. Por defecto se usa el algoritmo de Levenberg-Marquardt. Este algoritmo divide los datos disponibles en tres grupos:
  - 60 %: Usados para entrenar la red.
  - 20 %: Usados para validar la red y detener el entrenamiento cuando haya *overfitting*.
  - 20 %: Usados para testear la red.

  »net=train(net,bodyfatInputs,bodyfatTargets);

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

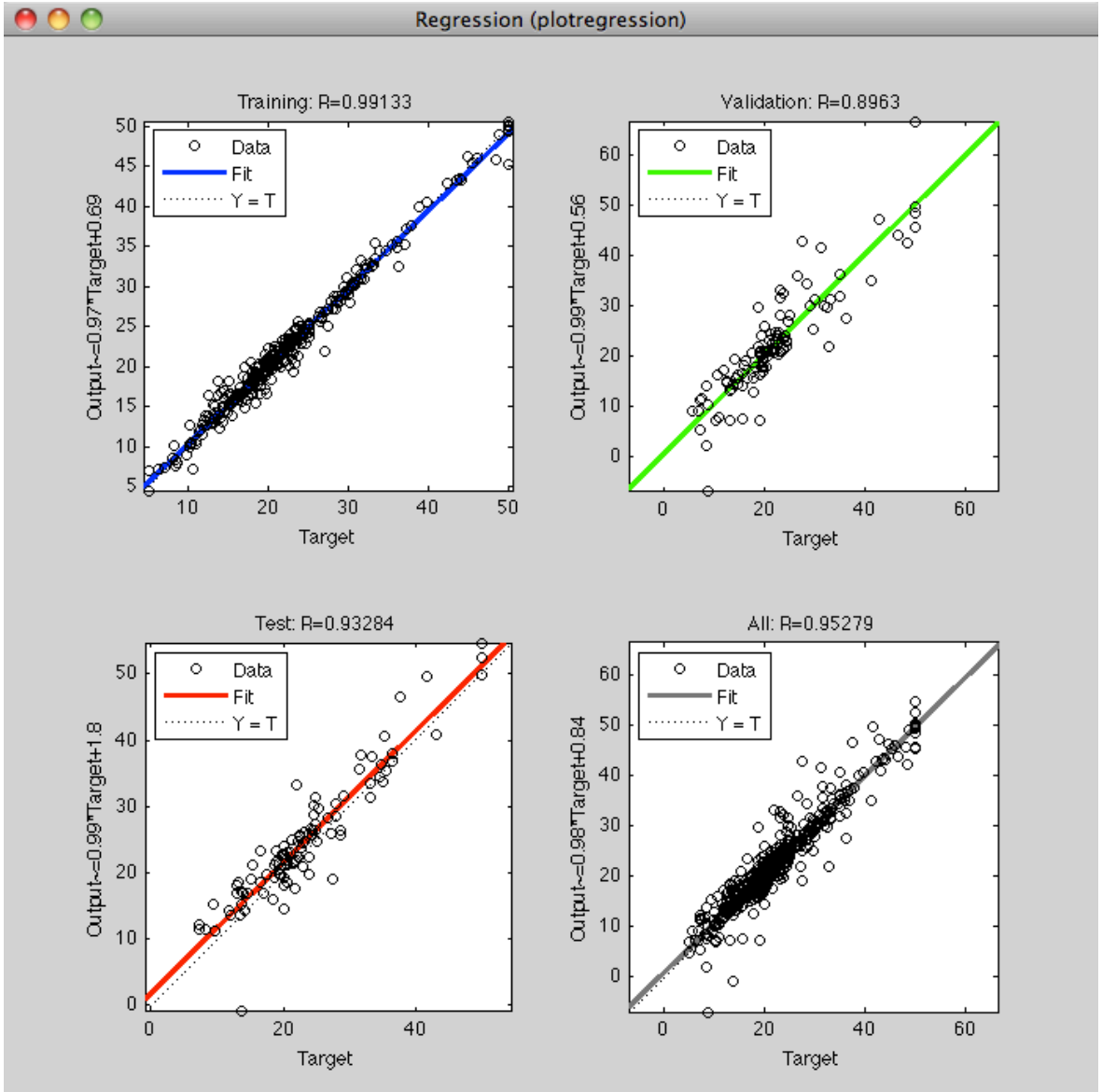


- En este ejemplo usa el método *train* para el entrenamiento. También se puede usar el método *adapt*. Consultar la ayuda sobre *Training Styles* para mas información.
- El entrenamiento termina cuando el error de validación no mejora durante 6 iteraciones, lo que ocurre en la iteración 11. Si se abre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



- 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 correspondentes valores objetivo. Se abre la figura de la siguiente transparencia.

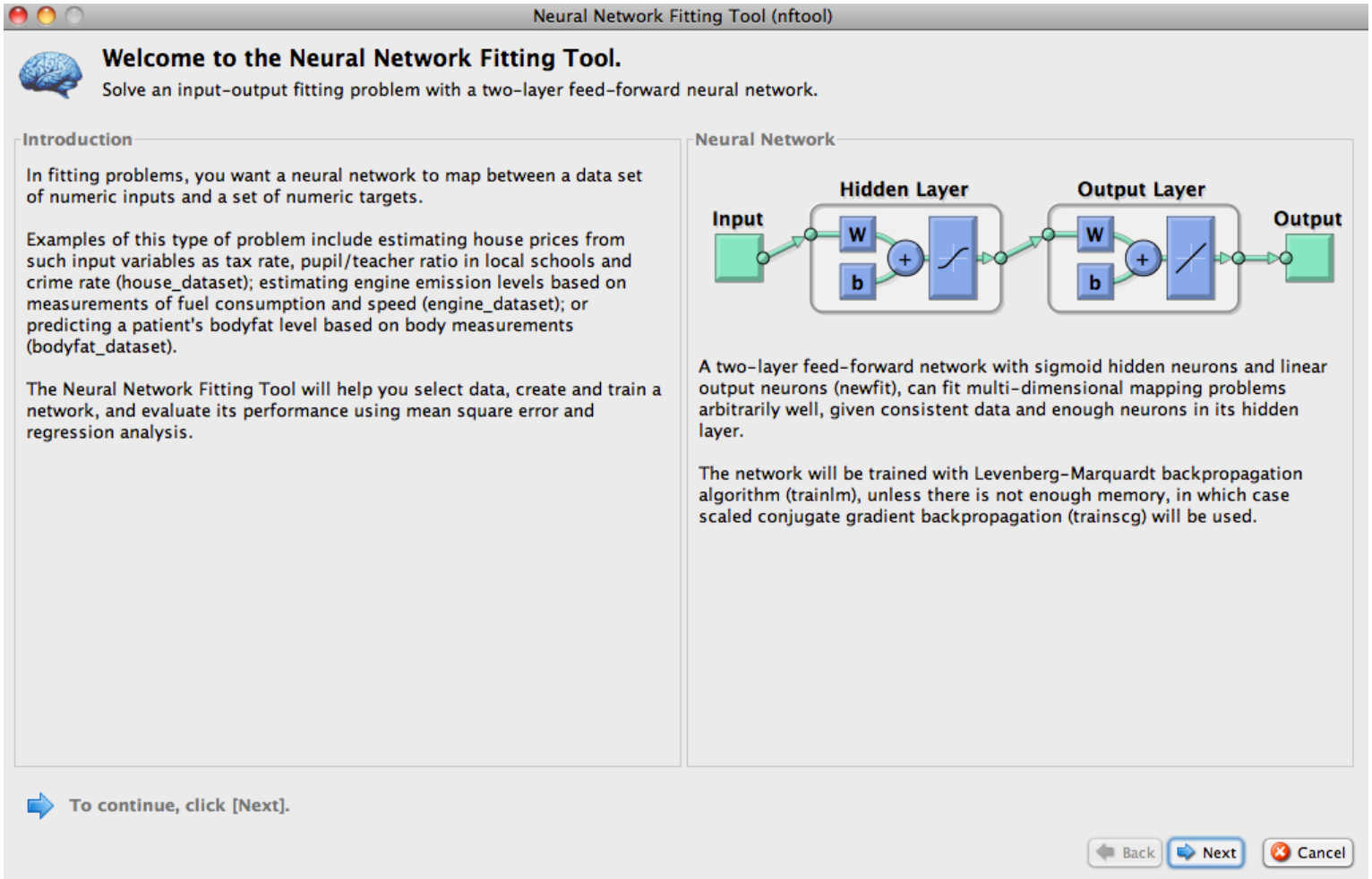
Identificación de funciones: línea de comandos



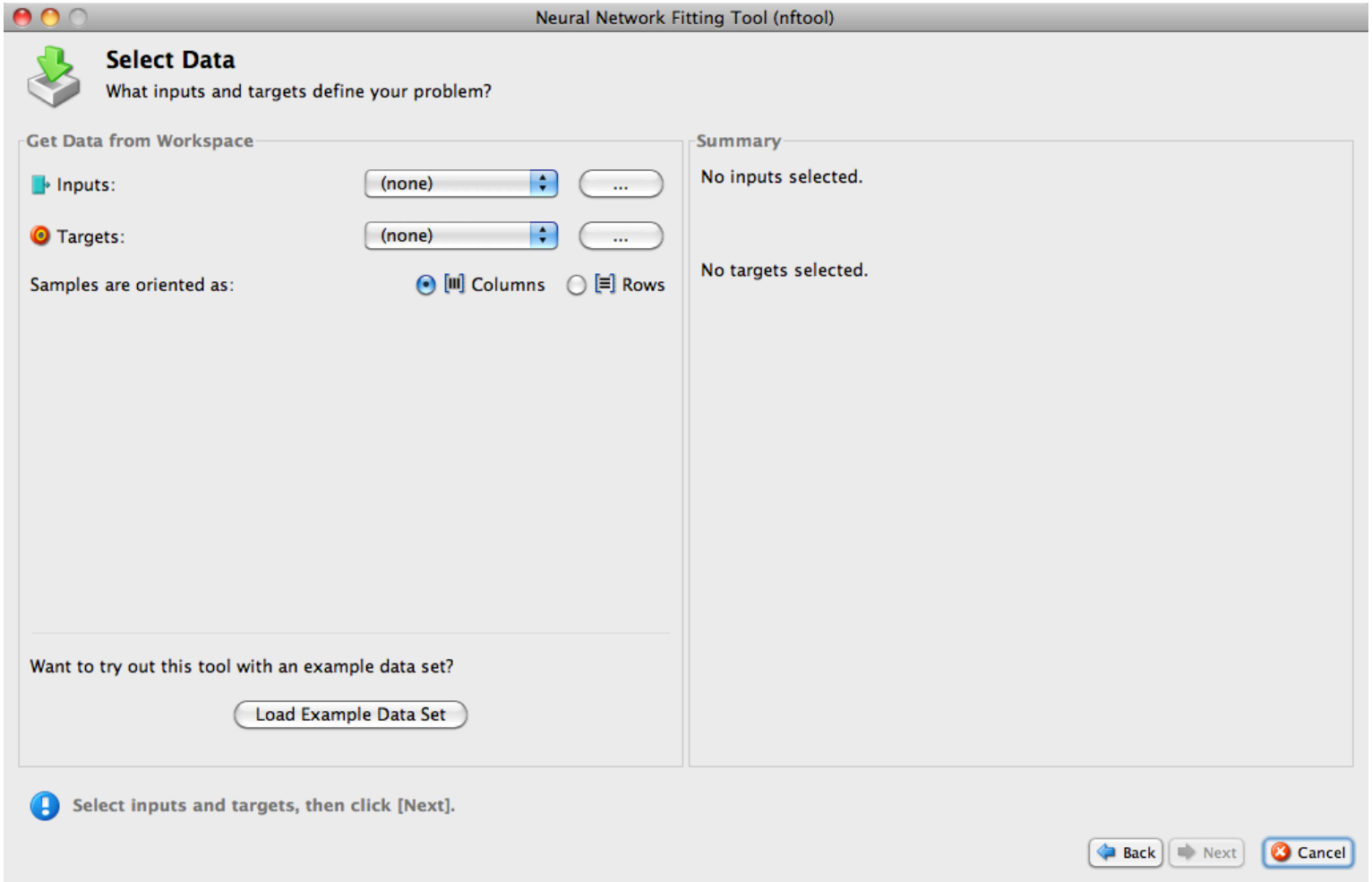
- La salida sigue muy bien los objetivos para entrenamiento, validación y test, y el valor R está por encima de 0,95 para la respuesta total.
- Si se quisieran obtener mejores resultados se puede probar con lo siguiente:
  - Reiniciar la red y entrenarla de nuevo con los valores ya obtenidos como pesos 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*.

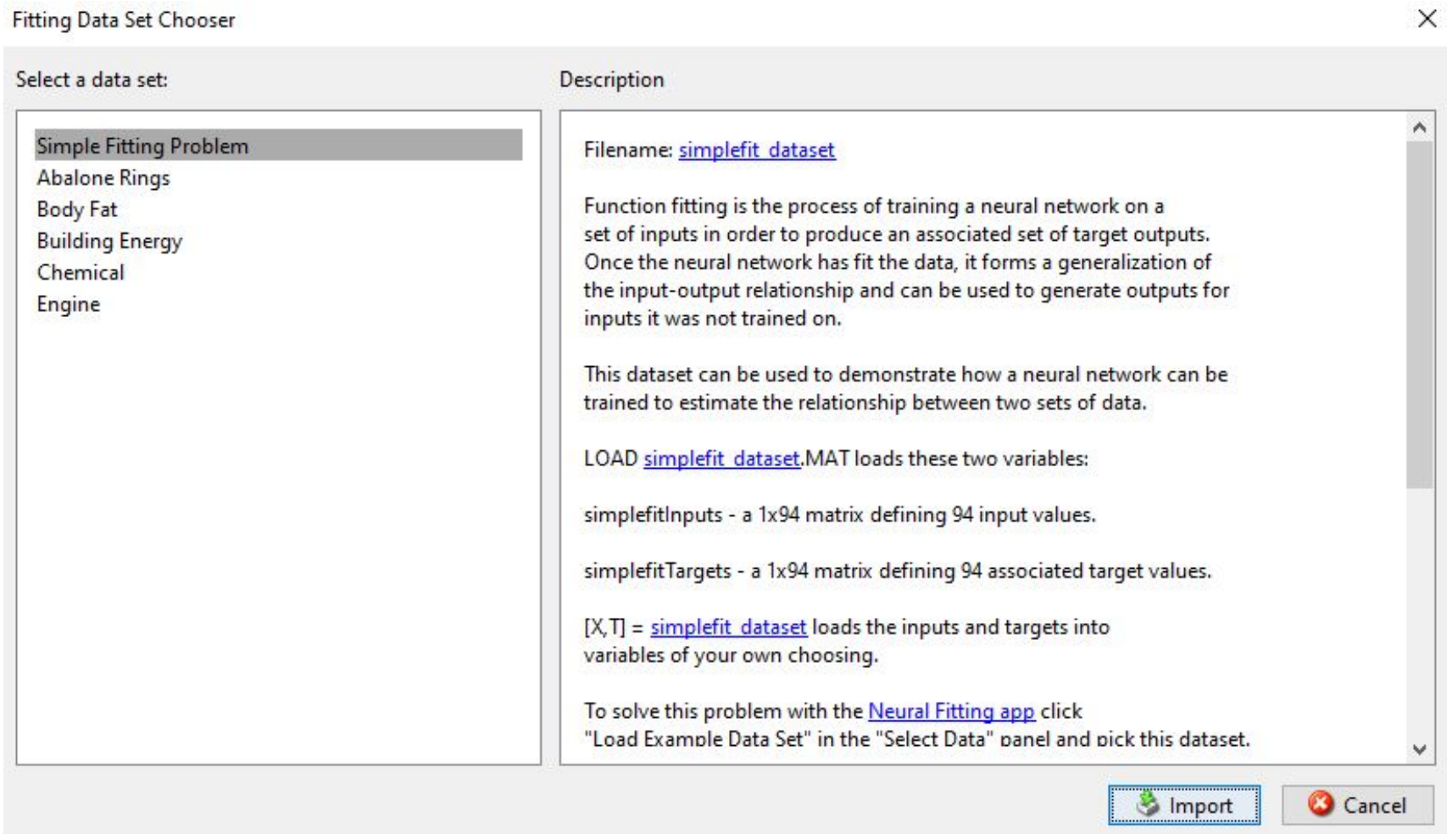
- Una forma alternativa para implementar redes neuronales es mediante la Neural Network Fitting Tool GUI.
- 1. Para abrir esta herramienta se usa el comando *nftool*.



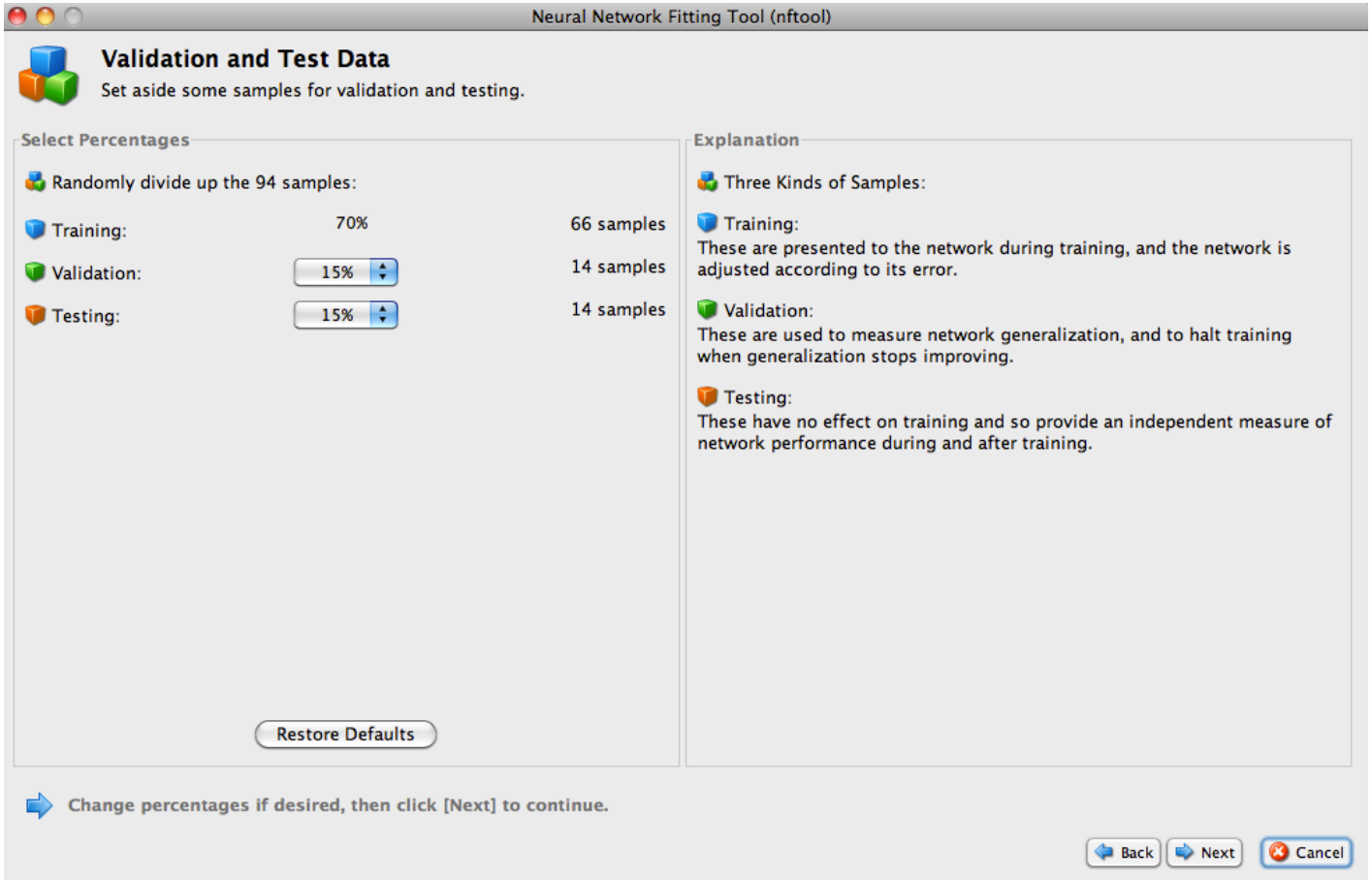
- 2. Teclear *Next*.



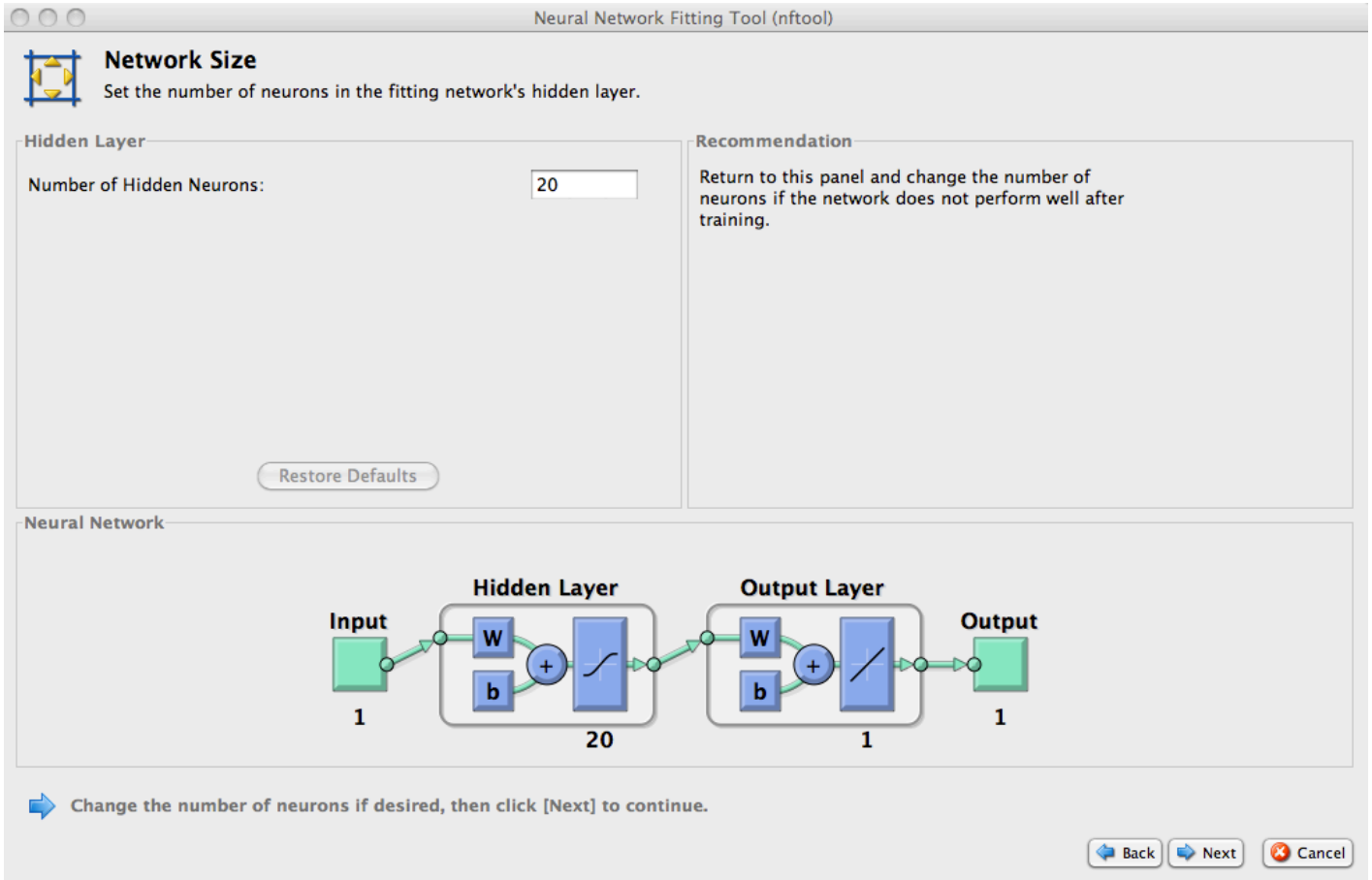
- 3. Teclear *Load Example Data Set*.



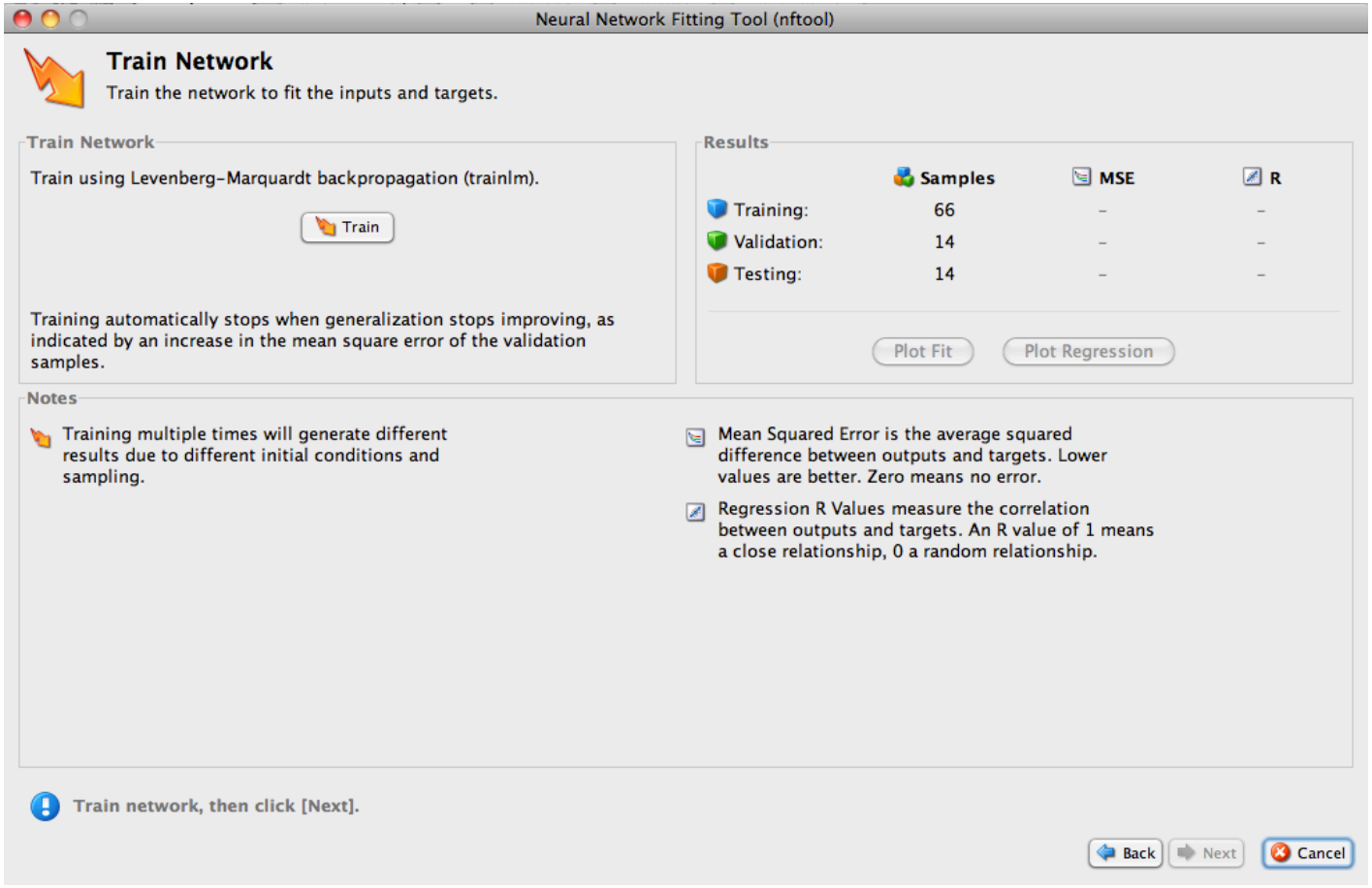
- 4. Seleccionar *Simple Fitting Problem*, y teclear *Import*. Esto nos devuelve a la ventana de selección de datos.
- 5. Teclear *Next* para pasar a la ventana *Validate and Test Data*. Se toma un 15 % de los datos originales para validación y test.



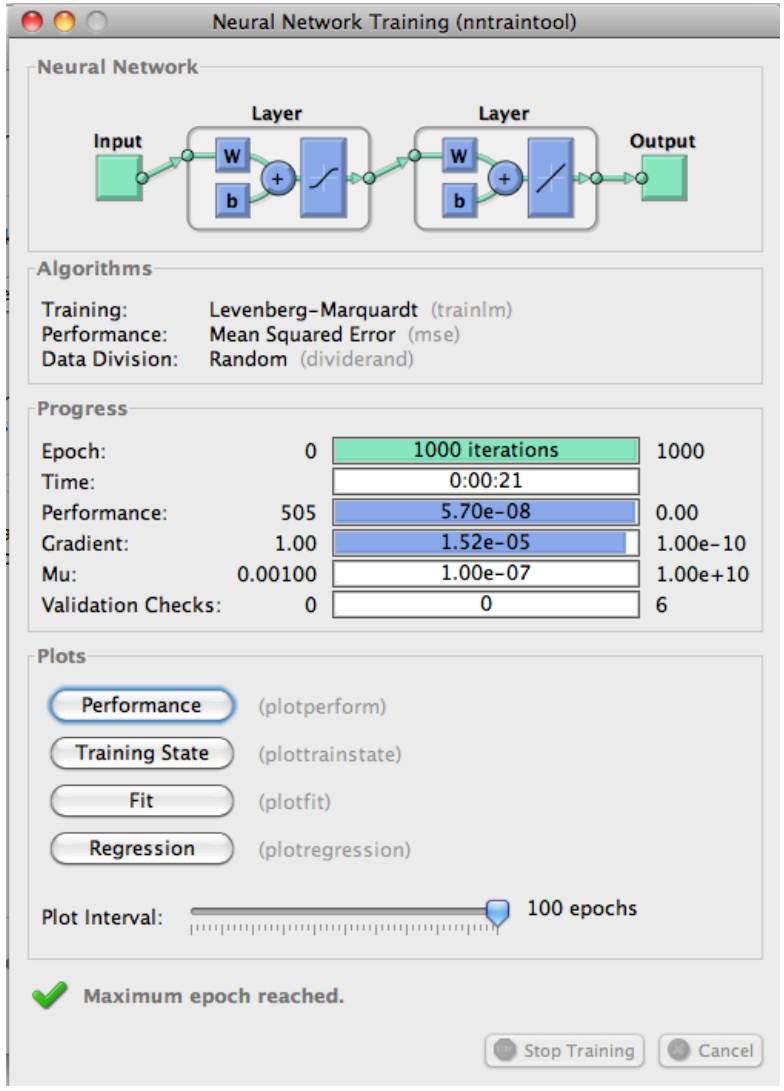
- 6. Teclear *Next*. El número de neuronas de la capa oculta se fija a 20. Se puede cambiar si la red no funciona de acuerdo a lo esperado.



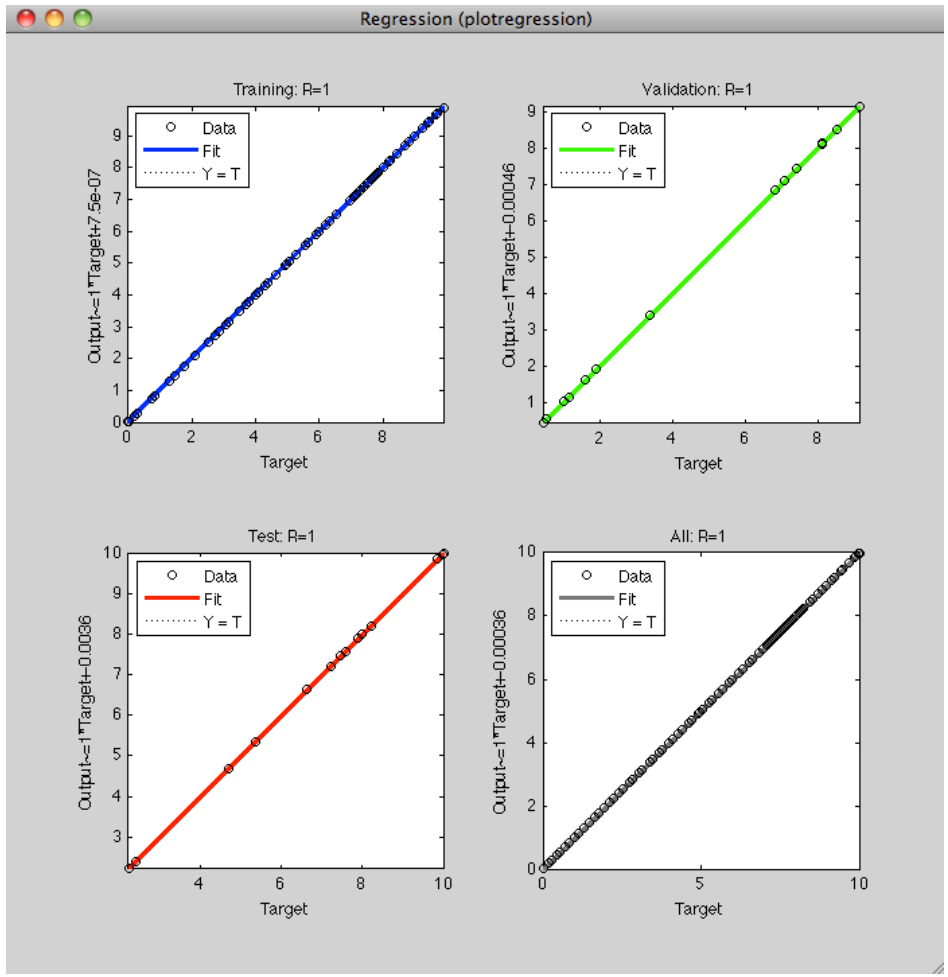
- 7. Teclear *Next*.



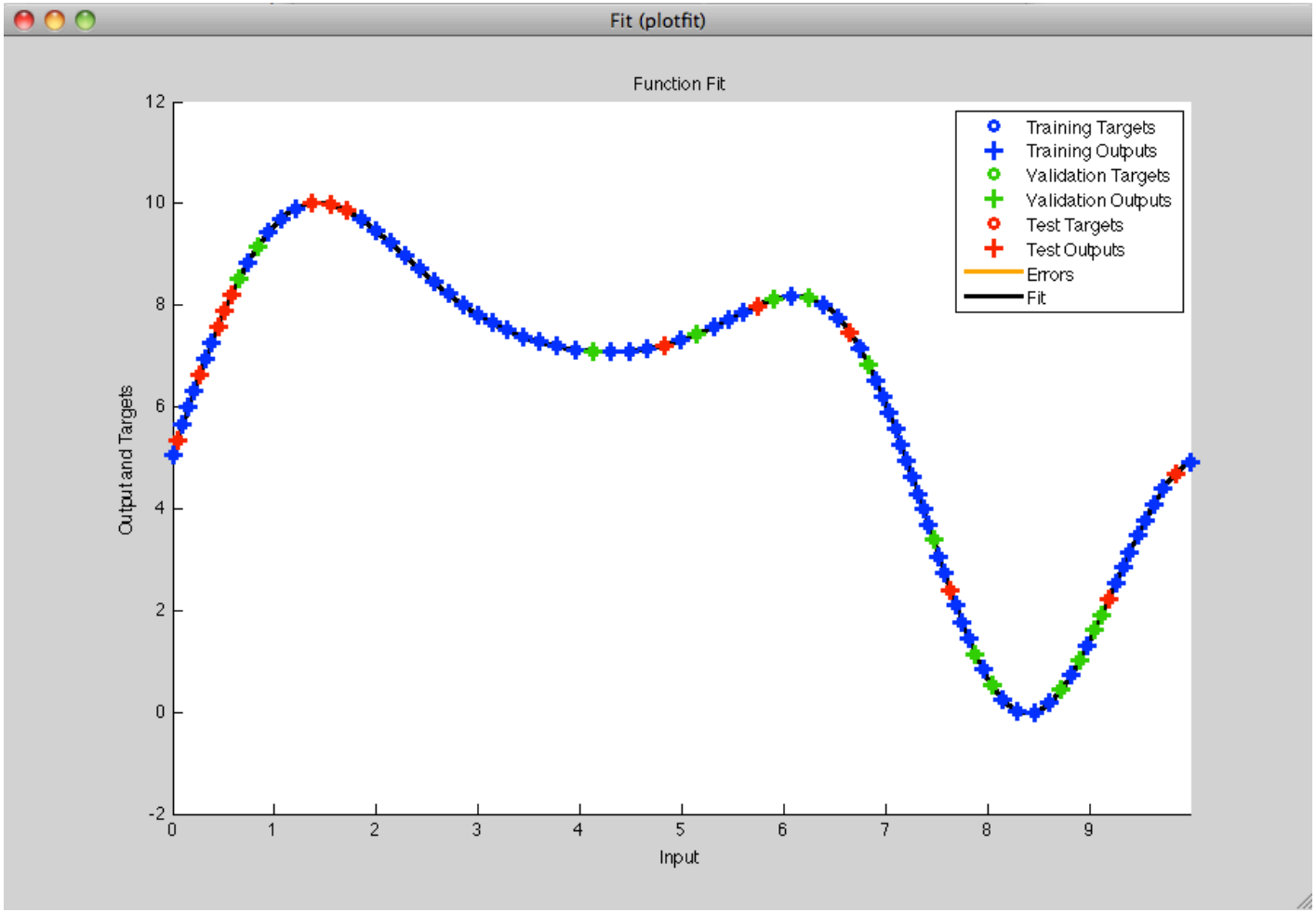
- 8. Teclear *Train*. El entrenamiento continúa hasta un máximo de 1000 iteraciones.



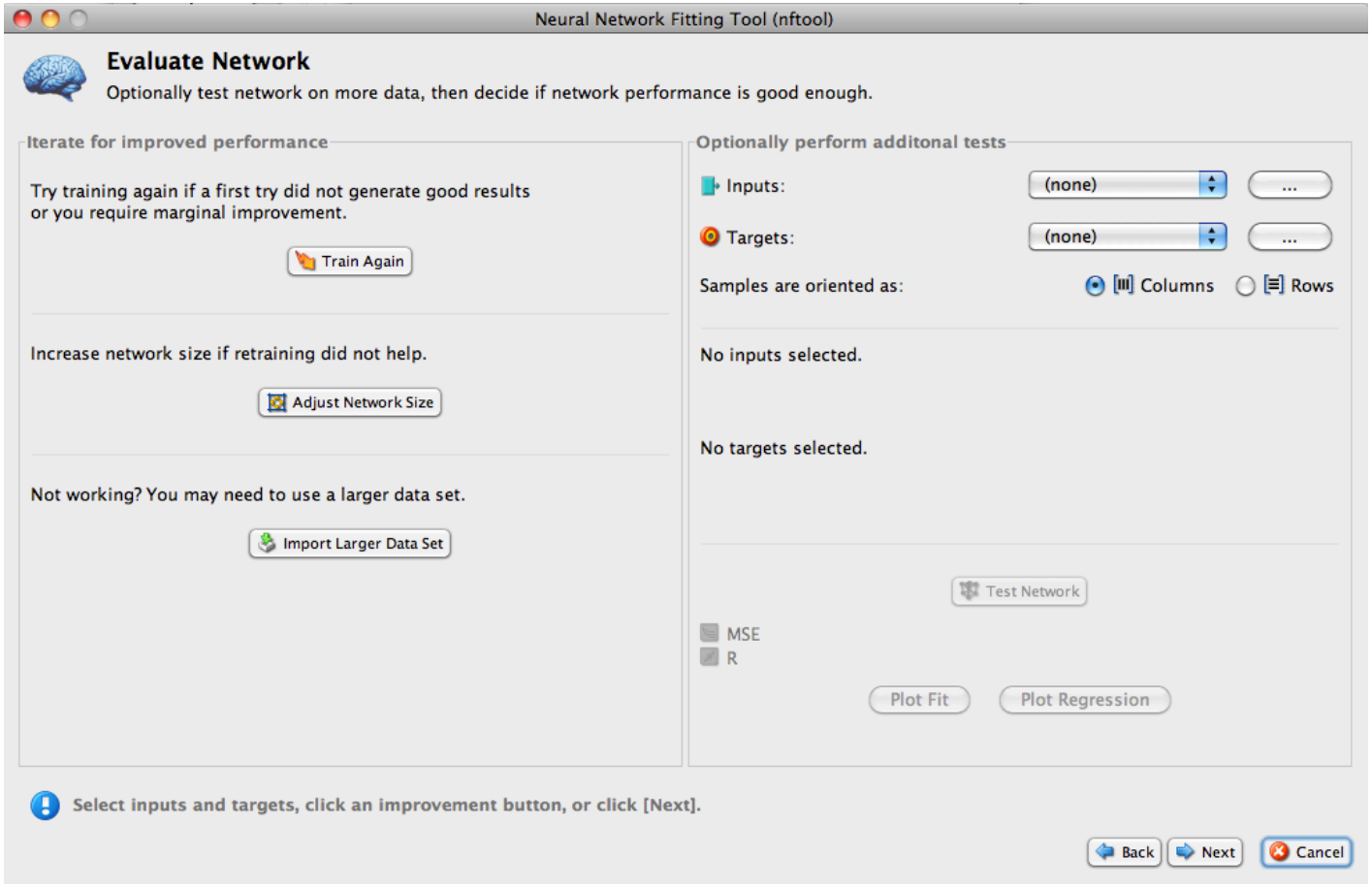
- 9. Dentro de *Plots*, seleccionar *Regression*. Para este ejemplo simple el ajuste es casi perfecto para entrenamiento, validación y test.



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

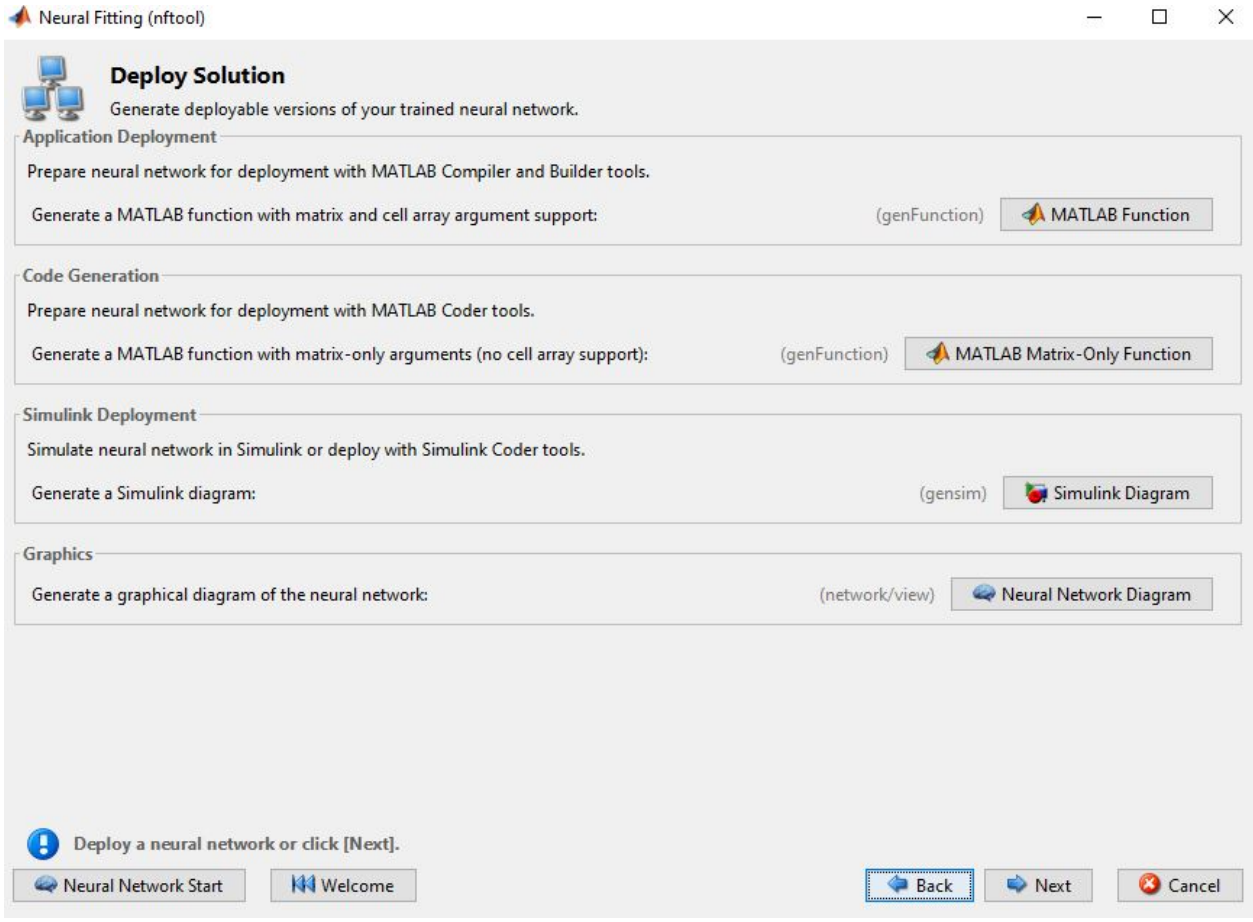


- 11. Pulsar *Next* en *Neural Network Fitting Tool* para evaluar la red.

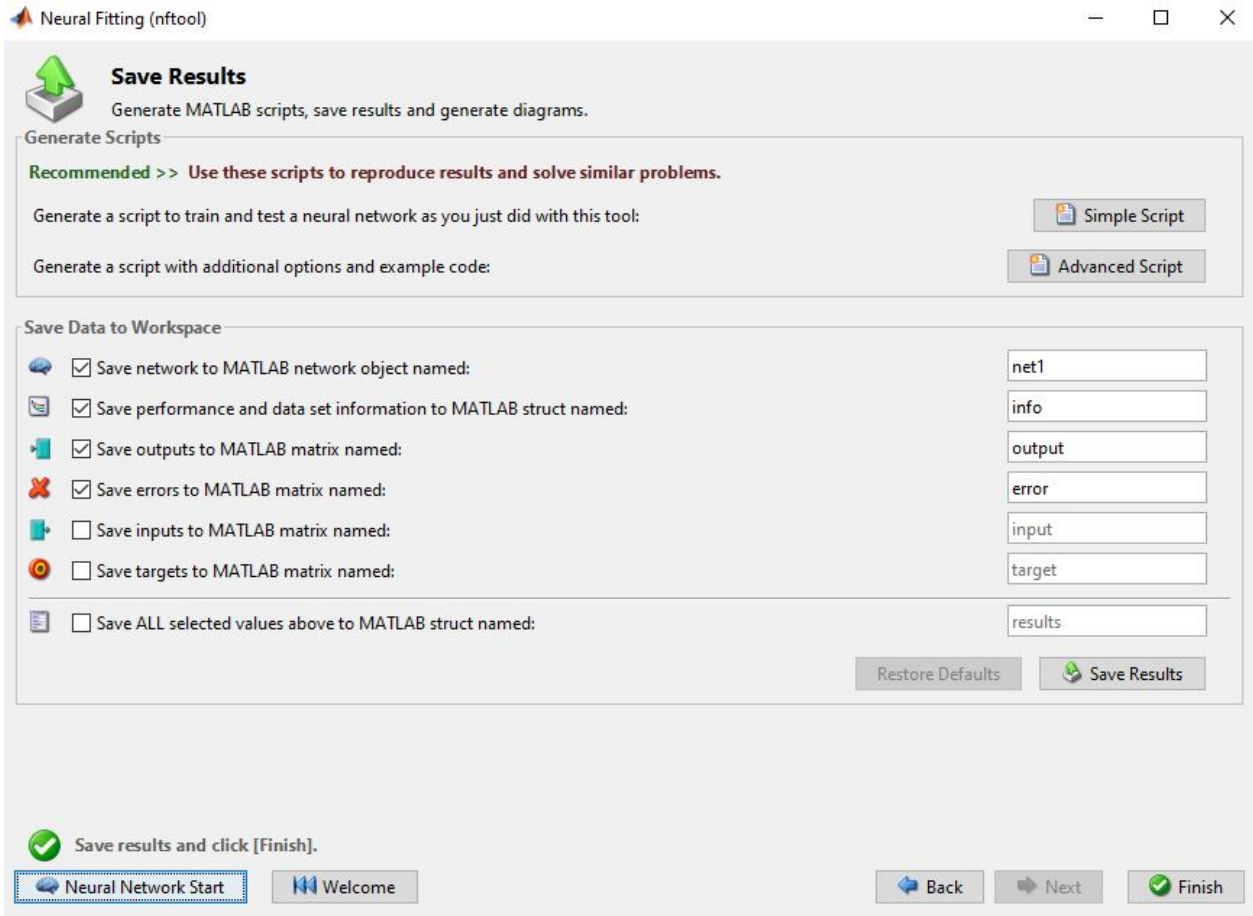


- En este punto es posible probar la red utilizando nuevos datos.
- Si no se está satisfecho con los resultados obtenidos se puede hacer lo siguiente:
  - Entrenar de nuevo la red.
  - Incrementar el número de neuroonas.
  - Utilizar un conjunto de datos de entrenamiento mayor.
- 12. Si los resultados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

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



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



- 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