Práctica: Caja de herramientas de red neuronal

Neural Network Toolbox proporciona algoritmos, modelos preentrenados y aplicaciones para crear, entrenar, visualizar y simular redes neuronales con una capa oculta (llamadas redes neuronales superficiales) y redes neuronales con varias capas ocultas (llamadas redes neuronales profundas). Mediante el uso de las herramientas ofrecidas, podemos realizar clasificación, regresión, agrupamiento, reducción de dimensionalidad, pronóstico de series de tiempo y modelado y control de sistemas dinámicos.

Hay varias formas de utilizar el software Neural Network Toolbox; los esenciales son las cuatro formas que se enumeran aquí:

- El más cómodo utiliza interfaces gráficas de usuario de MATLAB. Podemos iniciar la ventana principal a través del comando nnstart. De esta forma, podemos realizar automáticamente las siguientes tareas: ajuste de funciones (nftool), reconocimiento de patrones (nprtool), agrupación de datos (nctool), análisis de series temporales (ntstool).
- Podemos usar operaciones básicas de línea de comandos. Las operaciones de línea de comandos ofrecen una mayor flexibilidad pero requieren más conocimiento. Es por eso que deberá recordar todas las funciones necesarias sin la ayuda de los menús e íconos que normalmente aparecen en la GUI.
- Podemos personalizar la caja de herramientas. De hecho, podemos crear nuestras propias redes neuronales.
- Puede crear redes con conexiones arbitrarias y continuar capacitándolas utilizando las funciones de capacitación existentes de la caja de herramientas en la GUI.
- Finalmente, podemos modificar las funciones en la caja de herramientas. Cada componente computacional está escrito en código MATLAB y es completamente accesible.

Como puede ver, hay algo para todos, desde el novato hasta el experto. Hay herramientas simples disponibles para guiar al nuevo usuario a través de aplicaciones específicas y herramientas más complejas que permiten personalizar la red para probar nuevas arquitecturas.

Cualquiera que sea la forma en que abordemos estos problemas, un análisis adecuado con el uso de redes neuronales debe incluir los siguientes pasos:

1. Recopilar datos.
2. Cree la red.
3. Configure la red.
4. Inicializar los pesos y sesgos.
5. Capacitar a la red.
6. Valide la red.
7. Pruebe la red.

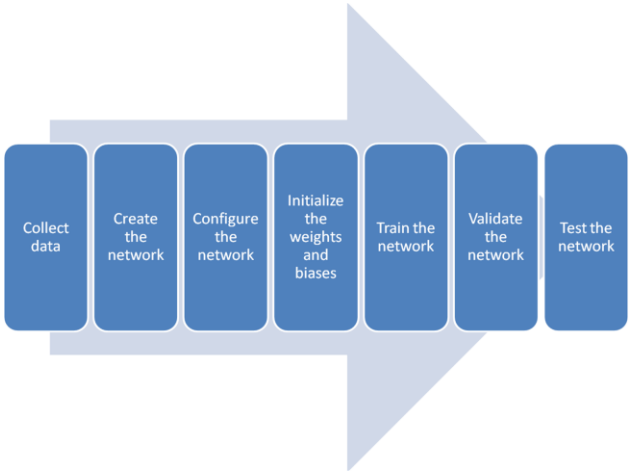

Figura 14: Flujo de trabajo para el diseño de redes neuronales

El primer paso consiste en recopilar los datos que queremos analizar; esto generalmente se realiza fuera del entorno MATLAB. Obviamente, esta fase es de suma importancia para el resultado final, ya que la calidad de los datos recopilados nos permitirá extraer el conocimiento adecuadamente.

Luego, vamos a crear la red; en este caso, podemos usar una serie de funciones disponibles en la caja de herramientas que nos permiten crear una red neuronal a través de uno de los algoritmos esperados. Este procedimiento conduce a la creación de un objeto de red neuronal. Este objeto se utiliza para almacenar toda la información que define una red neuronal. Tiene una serie de propiedades que definen sus características. Algunos de ellos se enumeran aquí con breves descripciones:

1. General: Propiedades generales de las redes neuronales
2. Arquitectura: la cantidad de subobjetos de red (entradas, capas, salidas, objetivos, sesgos y pesos) y cómo están conectados
3. Estructuras de subobjetos: arreglos de celdas de estructuras que definen cada una de las entradas, capas, salidas, objetivos, sesgos y pesos de la red.
4. Funciones: definen los algoritmos que se utilizarán cuando una red se adapte, se inicialice, se mida su rendimiento o se entrene.
5. Valores de peso y sesgo: estos definen los parámetros ajustables de la red (sus matrices de peso y vectores de sesgo)

El tercer paso implica la configuración de la red, que consiste en examinar los datos de entrada y salida, establecer las dimensiones de entrada y salida de la red para que se ajusten a los datos y elegir la configuración de procesamiento de entrada y salida que permita un mejor rendimiento de la red. La fase de configuración normalmente se realiza automáticamente cuando se llama a la función de entrenamiento. Sin embargo, se puede ejecutar manualmente, utilizando la función de configuración.

El cuarto paso implica la inicialización de pesos y sesgos, que consiste en establecer los valores iniciales desde los cuales comenzar y luego entrenar la red. Este paso ocurre automáticamente una vez que se ha decidido el algoritmo de entrenamiento, pero puede ser configurado por el usuario.

El quinto paso implica la formación de redes; en esta etapa, los pesos y sesgos deben ajustarse para optimizar el rendimiento de la red. Representa la fase más importante de todo el proceso ya que cuanto mejor sea la red, mejor podrá operar la generalización con nuevos datos desconocidos para ella. En esta etapa, parte de los datos recopilados se toma al azar (generalmente el 70 por ciento de los casos disponibles).

El sexto paso implica la validación de la red, en la que una fracción de los datos recopilados aleatoriamente (generalmente el 15 por ciento de los casos disponibles) se pasa a la red para estimar qué tan bien se ha entrenado nuestro modelo. A partir de los resultados obtenidos en esta fase, se podrá decidir si el modelo elegido refleja adecuadamente las expectativas iniciales o si es necesario volver atrás y elegir un modelo diferente.

El paso final involucra el uso de la red; parte de los datos recopilados tomados al azar (generalmente el 15 por ciento de los casos disponibles) se pasan a la red para probarlos. Luego, el objeto de la red neuronal se puede guardar y usar tantas veces como desee con cualquier dato nuevo.

La siguiente figura muestra cómo se ha dividido el conjunto de datos original:

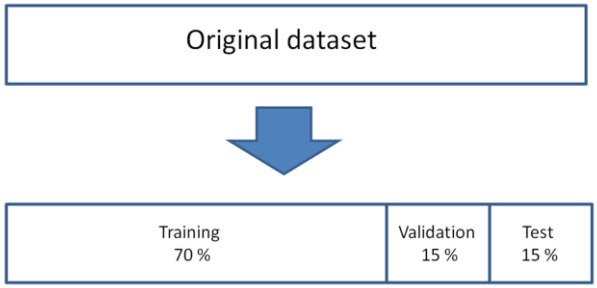

Figura 15: Subdivisión de datos originales en etapas de entrenamiento, validación y prueba

Al describir los diferentes pasos en los que se divide el flujo de trabajo para el diseño de redes neuronales, nos hemos referido a un desglose de los datos recopilados en tres conjuntos: Entrenamiento, validación y prueba. Ahora los describimos en detalle:

Conjunto de entrenamiento (generalmente el 70 por ciento de los casos disponibles): un conjunto de ejemplos utilizados para el aprendizaje, para ajustarse a los parámetros de la red neuronal. Usamos el conjunto de entrenamiento para encontrar los valores óptimos de pesos y sesgos.

Conjunto de validación (generalmente el 15 por ciento de los casos disponibles): un conjunto de ejemplos utilizados para ajustar los parámetros de una red. Usamos el conjunto de validación para encontrar el número óptimo de unidades ocultas o para determinar un punto de parada para el algoritmo utilizado.

Conjunto de prueba (generalmente el 15 por ciento de los casos disponibles): un conjunto de ejemplos que se utilizan solo para evaluar el rendimiento de una red completamente entrenada. Usamos el conjunto de prueba para estimar la tasa de error después de haber elegido el modelo final. Después de evaluar el modelo final en el conjunto de prueba, no debemos ajustar más el modelo.

En este punto, el flujo de trabajo está suficientemente claro, por lo que es hora de ponerse manos a la obra. Pero antes de comenzar, tenga en cuenta una última cosa sobre los datos que se van a analizar. Anteriormente hemos señalado que el proceso de recopilación de datos a menudo se realiza fuera del entorno MATLAB. Esto significa que debe tener disponible un archivo de datos correctamente recopilado para iniciar un análisis en MATLAB. Pero estamos aquí para aprender, por lo que no tenemos los datos disponibles, al menos no ahora. No temas, como siempre, MATLAB viene al rescate.

De hecho, varios conjuntos de datos de muestra están disponibles en el software Neural Network Toolbox. Podemos usar estos datos para experimentar con la funcionalidad de la caja de herramientas. Para ver los conjuntos de datos que están disponibles, use el siguiente comando:

```
>> help nndatasets
Neural Network Datasets
-----------------------
simplefit_dataset   - Simple fitting dataset.
abalone_dataset     - Abalone shell rings dataset.
bodyfat_dataset     - Body fat percentage dataset.
building_dataset    - Building energy dataset.
chemical_dataset    - Chemical sensor dataset.
cho_dataset         - Cholesterol dataset.
engine_dataset      - Engine behavior dataset.
vinyl_dataset       - Vinyl bromide dataset.
```

El código que acabamos de ver muestra un breve resumen. Tenga en cuenta que todos los conjuntos de datos tienen nombres de archivo con el formato nombre_conjunto de datos. Dentro de estos archivos estarán las matrices nameInputs y nameTargets. Por ejemplo, podemos cargar un conjunto de datos en el espacio de trabajo con el siguiente comando:

```
>> load abalone_dataset
```

Esto cargará abaloneInputs y abaloneTargets en el espacio de trabajo. Si desea cargar las matrices de entrada y destino con nombres diferentes, puede usar el siguiente comando:

```
>> [Input,Target] = abalone_dataset;
```

Esto cargará las entradas y los objetivos en los arreglos Input y Target. Puede obtener una descripción de un conjunto de datos con este comando:

```
>> help abalone_dataset
```

En la siguiente figura se muestra el resultado de este comando en el entorno MATLAB:

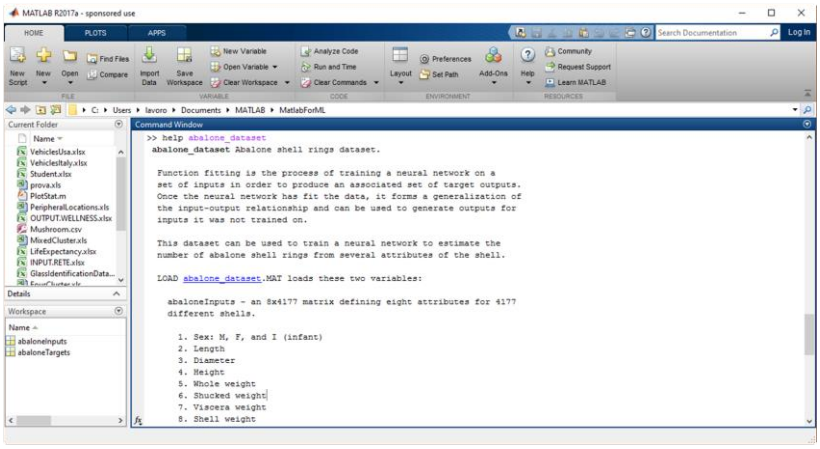

Figura 16: Breve resumen del contenido de abalone_dataset

Como se puede observar en la Figura 16, se propone una descripción del contenido del conjunto de datos:

número de atributos, número de elementos, lista de variables, etc. Además, se sugieren posibles usos.

GUI de inicio de red neuronal

Una GUI es una interfaz que permite a los usuarios interactuar con las computadoras a través de iconos gráficos e indicadores visuales en lugar de interfaces basadas en texto. Las GUI se introdujeron para hacer que las operaciones interactivas fueran más convenientes en comparación con los procedimientos más desafiantes requeridos por las interfaces de línea de comandos, que requieren que los comandos se escriban en el teclado de una computadora. Las acciones en una GUI generalmente se realizan mediante la manipulación directa de los elementos gráficos.

Para que la aplicación de redes neuronales sea lo más simple posible, la caja de herramientas nos brinda una serie de GUI. La principal es la función nnstart(), que abre una ventana con botones de lanzamiento para herramientas de ajuste de redes neuronales, reconocimiento de patrones, agrupación y series temporales:

>> nnstart

Este comando abre la GUI de inicio de red neuronal, como se muestra en la siguiente figura:

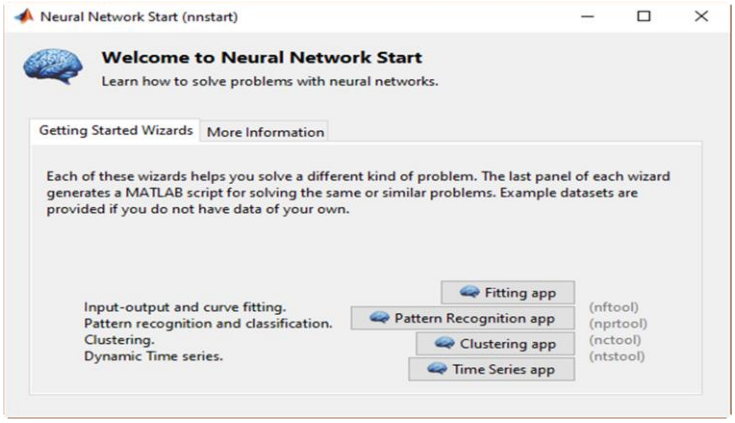

Figura 17: La GUI de inicio de la red neuronal

Esta GUI representa el tablero de toda la caja de herramientas, en el sentido de que desde aquí es posible realizar cualquier tipo de análisis que involucre el uso de redes neuronales mediante interfaces gráficas. Como puede verse en la Figura 17, hay cuatro tipos de problemas disponibles:

- Ajuste de funciones (nftool)
- Reconocimiento de patrones (nprtool)
- Agrupación de datos (nctool)
- Análisis de series temporales (ntstool)

Para cada tarea, hay disponible una GUI que nos guía paso a paso a lo largo del proceso de análisis. La primera herramienta (nftool) resuelve un problema de ajuste. En un problema de ajuste, se crea una red neuronal para mapear entre un conjunto de entradas numéricas y un conjunto de objetivos numéricos. La aplicación Neural Fitting nos ayuda a seleccionar datos, crear y entrenar la red, y evaluar su rendimiento utilizando el error cuadrático medio y el análisis de regresión. Abordaremos un problema de ajuste en detalle.

La segunda herramienta (nprtool) resuelve problemas de reconocimiento de patrones. En los problemas de reconocimiento de patrones, usamos una red neuronal para clasificar las entradas en un conjunto de categorías objetivo. Esta aplicación nos ayudará a seleccionar datos, crear y entrenar una red y evaluar su rendimiento utilizando matrices de confusión y entropía cruzada.

La tercera herramienta (nctool) resuelve los problemas de agrupamiento. En problemas de agrupamiento, usamos una red neuronal para agrupar datos por similitud. Esta aplicación nos ayudará a seleccionar datos, crear y entrenar una red y evaluar su rendimiento utilizando una variedad de herramientas de visualización.

La última herramienta (ntstool) resuelve problemas de series temporales no lineales con redes neuronales dinámicas. Las redes neuronales dinámicas se utilizan para el filtrado y la predicción no lineales. La predicción es un tipo de filtrado dinámico en el que se utilizan valores pasados de una o más series temporales para predecir valores futuros. Esta herramienta nos permite resolver tres tipos de problemas de series temporales no lineales:

- Autorregresivo no lineal con entrada externa
- Autorregresivo no lineal

- Entrada-salida no lineal

Ajuste de datos con redes neuronales

El ajuste de datos es el proceso de construir una curva o una función matemática que tenga la mejor coincidencia con un conjunto de puntos recopilados previamente. El ajuste de curvas puede relacionarse tanto con interpolaciones, donde se requieren puntos de datos exactos, como con suavizado, donde se construye una función plana que aproxima los datos. Estamos hablando del ajuste de curvas en un análisis de regresión, que se ocupa más de los problemas de inferencia estadística, así como de la incertidumbre de que una curva coincida con los datos observados que tienen errores aleatorios. Las curvas aproximadas obtenidas del ajuste de datos se pueden usar para ayudar a mostrar datos, para predecir los valores de una función donde no hay datos disponibles y para resumir la relación entre dos o más variables.

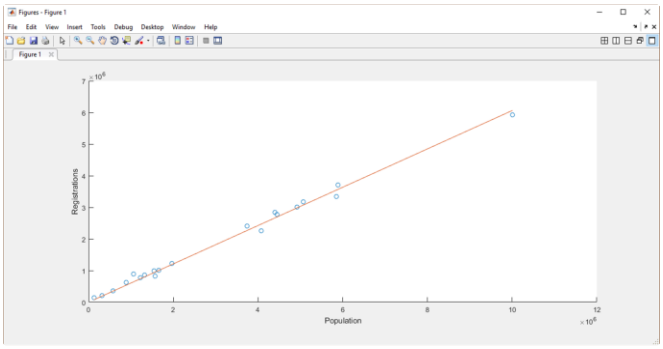

Figura 18: Interpolación lineal de los datos recopilados

Las fórmulas matemáticas que nos permiten predecir la tendencia de una determinada distribución que puede ser complicada, y muchas veces no son capaces de representar todos los datos. Al menos no podemos predecir la tendencia de los datos en todo el rango de su existencia. En estos casos, es útil utilizar algoritmos de aprendizaje automático que, como sabemos, son capaces de construir un modelo sin recurrir a complicadas fórmulas matemáticas. En particular, las ANN son adecuadas para operaciones de ajuste de datos para predecir la tendencia de los datos. De hecho, una red neuronal simple puede adaptarse a cualquier función práctica.

El ajuste de funciones es el proceso de entrenar una red neuronal en un conjunto de entradas para producir un conjunto asociado de salidas objetivo. Una vez que la red neuronal ha ajustado los datos, forma una generalización de la relación de entrada-salida y se puede utilizar para generar salidas para entradas en las que no se entrenó.

Como ya hemos dicho, en Neural Network Toolbox, puede resolver un problema de ajuste de datos de varias maneras. En concreto, los más utilizados son:

- Utilice una interfaz gráfica de usuario (nftool)
- Usar funciones de línea de comandos

Para principiantes, es mejor comenzar con la aplicación nftool (interfaz gráfica de usuario). La utilidad de tal enfoque, al menos en las etapas preliminares, se debe a la capacidad de adquirir las habilidades necesarias para poder hacer lo mismo más tarde sin la ayuda de la interfaz gráfica.

De hecho, la GUI nos ayuda ofreciéndonos una serie de ventanas en las que solo tendremos que hacer nuestras elecciones utilizando los menús e iconos propios de un sistema operativo de interfaz gráfica. En otras palabras, no se nos pedirá que recordemos todos los comandos para crear la red y luego entrenarla.

Posteriormente, una vez que hemos creado la red, podemos generar el script. Desde el análisis del script podemos acceder a las funciones que necesitamos para realizar todas las operaciones. De esta forma, podremos conocer todas las funciones involucradas en las diferentes fases y cómo se utilizan.

Como siempre, antes de comenzar, independientemente del método que elija utilizar, primero debe definir el problema seleccionando un conjunto de datos adecuado. Para comenzar, podremos usar uno de los conjuntos de datos de muestra que hemos mencionado anteriormente. Hay uno para cada uso de la caja de herramientas, especialmente para el ajuste de curvas. De hecho, acceder al conjunto de datos de ayuda se realiza de la siguiente manera:

```
>> help nndatasets
```

Una lista de todos los conjuntos de datos disponibles se agrupa por categoría. Específicamente para el ajuste de curvas, encontraremos el siguiente conjunto de datos:

- simplefit_dataset: conjunto de datos de ajuste simple
- abalone_dataset: conjunto de datos de anillos de concha de abulón
- bodyfat_dataset: conjunto de datos de porcentaje de grasa corporal
- building_dataset: conjunto de datos de energía del edificio
- chemical_dataset: conjunto de datos del sensor químico
- cho_dataset: conjunto de datos de colesterol
- engine_dataset: conjunto de datos de comportamiento del motor
- vinyl_dataset: conjunto de datos de bromuro de vinilo

Después de utilizar los ejemplos en la primera fase de aprendizaje, pasaremos a casos reales. Si tenemos un problema específico que queremos resolver, podemos cargar nuestros datos en el espacio de trabajo de MATLAB.

Cómo usar la aplicación Neural Fitting (nftool)

En un problema de ajuste, se crea una red neuronal para mapear entre un conjunto de entradas numéricas y un conjunto de objetivos numéricos. La aplicación Neural Fitting nos ayuda a seleccionar datos, crear y entrenar la red, y evaluar su rendimiento utilizando el error cuadrático medio y el análisis de regresión. Para comenzar, iniciamos la aplicación simplemente escribiendo el siguiente comando en el indicador de MATLAB:

```
>> nftool
```

Esta es la página de bienvenida de la aplicación Neural Fitting:

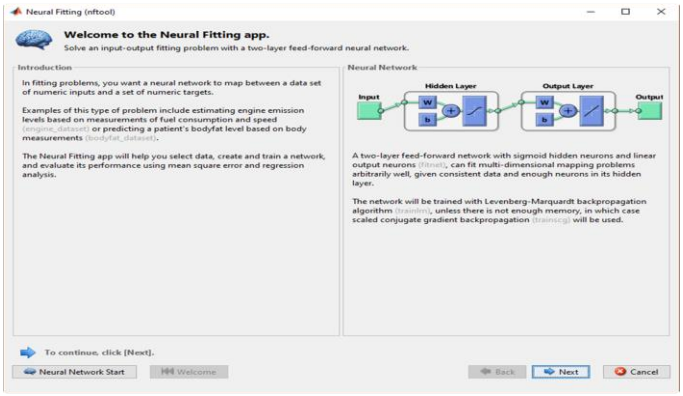

Figura 19: La página de bienvenida de la aplicación Neural Fitting

La página de bienvenida de la aplicación Neural Fitting (Figura 19) introduce el tema explicando en qué nos podrá ayudar la aplicación y nos recuerda los ejemplos listos para usar que podemos adoptar para comprender los mecanismos de análisis; pero lo más importante, describe la arquitectura de la red que se creará.

La aplicación Neural Fitting resuelve un problema de ajuste de entrada y salida con una red neuronal de avance de dos capas. ¿Pero de qué se trata? En una red neuronal feed-forward, las conexiones entre las unidades no forman un ciclo. En esta red, la información se mueve en una sola dirección, hacia adelante, desde los nodos de entrada a través de los nodos ocultos hasta los nodos de salida. No hay ciclos o bucles en la red. En la fase de entrenamiento, solo se ajustan los pesos.

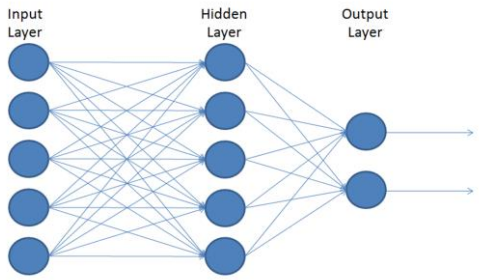

Figura 20: Una arquitectura de red neuronal feed-forward

En la parte inferior de la página de bienvenida de la aplicación Neural Fitting (Figura 19) se encuentran los botones necesarios para continuar con el análisis. En concreto, en la parte inferior derecha se encuentran los botones que nos permiten pasar a la página siguiente o retroceder sobre nuestros pasos. Tal y como sugiere la misma app, pulsamos en el siguiente botón para continuar.

Se abre una nueva ventana (Figura 21), donde podemos seleccionar los datos a procesar: entradas y objetivos que definen nuestro problema de ajuste. Hay dos opciones disponibles:

- Get Data from Workspace
- Load Example Data Set

La primera opción nos permite utilizar datos ya disponibles en el espacio de trabajo de MATLAB. Estos datos deben cargarse en dos matrices diferentes: una para entrada y otra para salida. Para

comprender cómo se deben formatear nuestros datos, podemos echar un vistazo a los conjuntos de datos de ejemplo ya mencionados anteriormente.

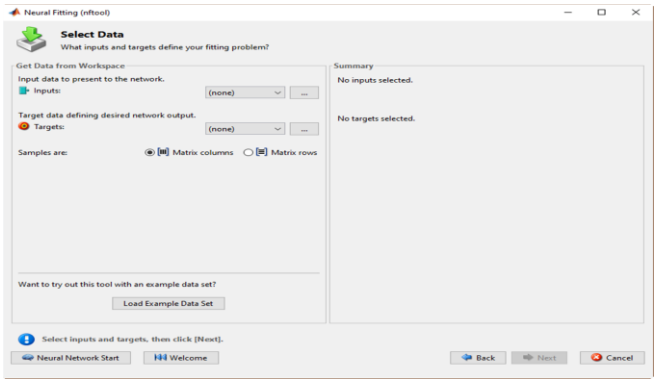

Figura 21: La ventana Seleccionar datos de la aplicación Neural Fitting

Como puede verse en la Figura 21, en el área Obtener datos del espacio de trabajo, hay disponibles dos campos claramente diferenciados:

- Datos de entrada para presentar a la red.
- Datos de destino que definen la salida de red deseada.

En ambos casos podemos usar el menú desplegable para seleccionar los datos ya disponibles en el espacio de trabajo, o hacer clic en el botón de tres puntos para importar nuevos datos a MATLAB. Permítanme hacer una aclaración final sobre el área Obtener datos del área de trabajo. Hay dos botones de radio:

- **Matrix columns**
- **Matrix rows**

Seleccione Columnas de matriz (opción predeterminada) si los datos se recopilan por columnas, es decir, si cada columna representa una observación. Seleccione Filas de matriz si los datos se recopilan por filas, es decir, si cada fila representa una observación. La segunda opción nos permite cargar un conjunto de datos de ejemplo para entender cómo proceder. Esta es la opción por utilizar en la primera etapa de aprendizaje, en la que no tenemos que preocuparnos por la calidad de los datos fuente, ni siquiera por su formato. Podemos aprovechar los recursos que nos proporciona la caja de herramientas para comprender cómo realizar correctamente un análisis de ajuste. Para elegir esta opción, simplemente haga clic en el botón Cargar conjunto de datos de ejemplo. Se abre la siguiente ventana:

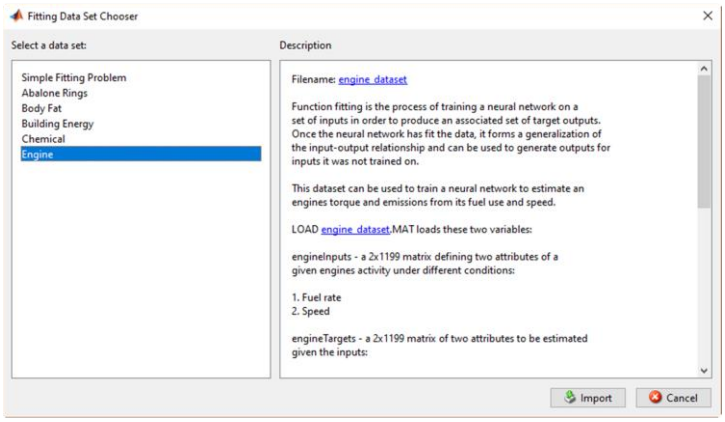

Figura 22: Ventana del selector de conjuntos de datos de ajuste

En esta ventana, se presenta una lista de conjuntos de datos disponibles. Para cada elemento seleccionado, se propone una descripción completa del conjunto de datos. Para comenzar, utilizaremos este segundo enfoque cargando engine_dataset. Este conjunto de datos se puede usar para entrenar una red neuronal para estimar el par y las emisiones de un motor a partir de su uso de combustible y velocidad. Este conjunto de datos contiene dos variables:

- Entradas del motor: una matriz de 2x1199 que define dos atributos de la actividad de un motor dado en diferentes condiciones: tasa de combustible y velocidad
- EngineTargets: una matriz de 2x1199 de dos atributos que se estimarán dadas las entradas: torque y emisiones de óxido nitroso

Recapitulando, para cargar el conjunto de datos del motor, simplemente tenemos que seleccionar ese elemento en la ventana del selector de conjunto de datos de ajuste (Figura 22) y hacer clic en el botón Importar. Esto le devuelve a la ventana Seleccionar datos. Se importan dos matrices al espacio de trabajo de MATLAB:

Entradas del motor y objetivos del motor. Al mismo tiempo, estas matrices se cargan en sus respectivos campos en el área Obtener datos del espacio de trabajo en la ventana Seleccionar datos de la aplicación Neural Fitting (Figura 21). Haga clic en Siguiente para mostrar la ventana Datos de prueba y validación, que se muestra en la siguiente figura:

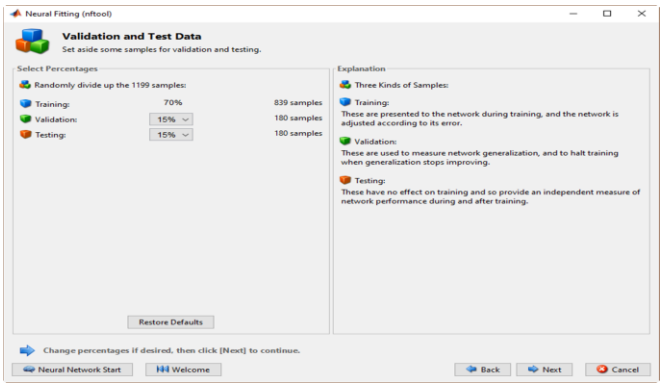

Figura 23: La ventana Datos de prueba y validación

En la ventana Datos de validación y prueba (Figura 23) se encuentra la subdivisión aleatoria de las 4177 muestras en Entrenamiento, Validación y Prueba. El porcentaje del conjunto de entrenamiento está bloqueado, mientras que los otros dos pueden ser configurados por el usuario.

Recuerde: el conjunto de entrenamiento se presenta a la red durante el entrenamiento y la red se ajusta de acuerdo con su error. El conjunto de validación se utiliza para medir la generalización de la red y detener el entrenamiento cuando la generalización deja de mejorar. Finalmente, el conjunto de prueba no tiene ningún efecto y, por lo tanto, proporciona una medida independiente del rendimiento de la red durante y después del entrenamiento.

Cambie los porcentajes si lo desea y luego haga clic en el botón Siguiente para mostrar la ventana Arquitectura de red, que se muestra en la siguiente figura:

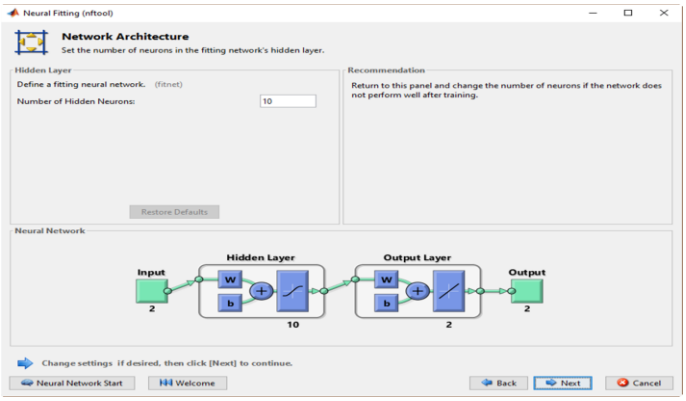

Figura 24: La ventana Arquitectura de red

La red predeterminada que se utiliza para el ajuste de funciones es una red de avance de dos capas, con una función de transferencia sigmoidea en la capa oculta y una función de transferencia lineal en la capa de salida. El número predeterminado de neuronas ocultas se establece en 10. Es posible que desee aumentar este número más adelante si el rendimiento del entrenamiento de la red es deficiente. Una vez que hemos elegido el número de neuronas en la capa oculta, podemos construir la red simplemente haciendo clic en el botón Siguiente. Se abre la ventana Train Network, como se muestra en esta figura:

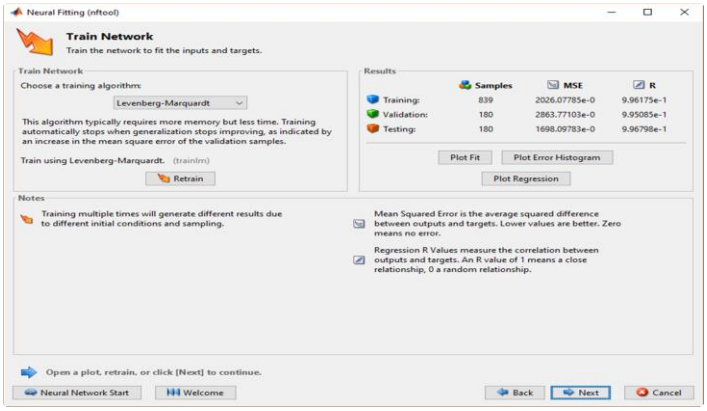

Figura 25: La ventana Red de trenes

En esta ventana, podemos elegir el algoritmo de entrenamiento. Hay tres opciones disponibles:

Se recomienda Levenberg-Marquardt (trainlm) para la mayoría de los problemas

- La regularización bayesiana (trainbr) puede llevar más tiempo, pero obtiene una mejor solución para algunos problemas pequeños y ruidosos.
- El gradiente conjugado escalado (trainscg) se recomienda para problemas grandes, ya que utiliza cálculos de gradiente, que son más eficientes en memoria que los cálculos jacobianos que utilizan los otros dos algoritmos.

Esta vez, usamos el Levenberg-Marquardt predeterminado. Una vez que hemos elegido el algoritmo, podemos iniciar el entrenamiento de la red simplemente haciendo clic en el botón Entrenar. El entrenamiento continúa hasta que el error de validación no disminuye durante seis iteraciones (parada de validación). Cuando se detiene el entrenamiento, los valores MSE y R se muestran en el área Resultados de la ventana Red de trenes.

El error cuadrático medio (MSE) es la diferencia cuadrática promedio entre las salidas y los objetivos. Los valores más bajos son indicativos de mejores resultados. Cero significa que no hay error. El valor de regresión (R) mide la correlación entre los resultados y los objetivos. Un valor de R de 1 significa una relación cercana y 0 significa una relación aleatoria.

Si no estamos satisfechos con los resultados, podemos intentar realizar más entrenamientos con la misma configuración de red, o primero modificar la configuración de la red haciendo clic en el botón Atrás y luego volver a la ventana Entrenar red para volver a ejecutar el entrenamiento. Esto cambiará los pesos y sesgos iniciales de la red y puede producir una red mejorada. Además, en el área Resultados de la ventana Tren de red, hay tres botones: Ajuste de gráfico, Histograma de error de gráfico y Regresión de gráfico. Estos son diagramas que nos brindan análisis importantes de los resultados obtenidos. Por ejemplo, se obtiene una verificación adicional del rendimiento de la red a partir del gráfico del histograma de errores, como se muestra en la siguiente figura:

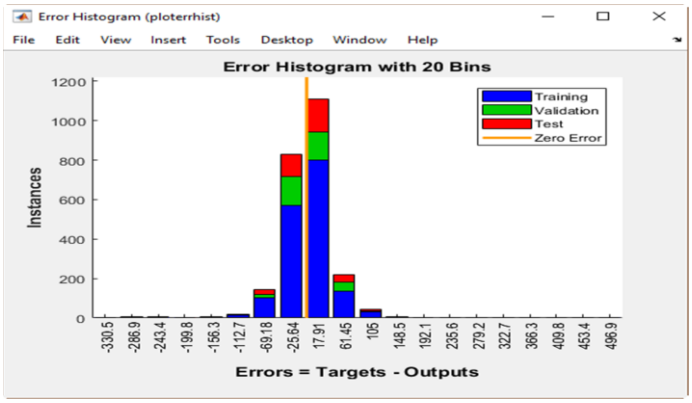

Figura 26: La gráfica del histograma de error

En el gráfico de histograma de error, las barras azules representan datos de entrenamiento, las barras verdes representan datos de validación y las barras rojas representan datos de prueba. Este gráfico puede darnos una indicación de la distribución del error; una distribución normal es indicativa de buenos resultados. Además, el histograma puede brindarnos una indicación de valores

atípicos, que son puntos de datos en los que el ajuste es significativamente peor que la mayoría de los datos. Otro gráfico utilizado para validar el rendimiento de la red es el gráfico de regresión:

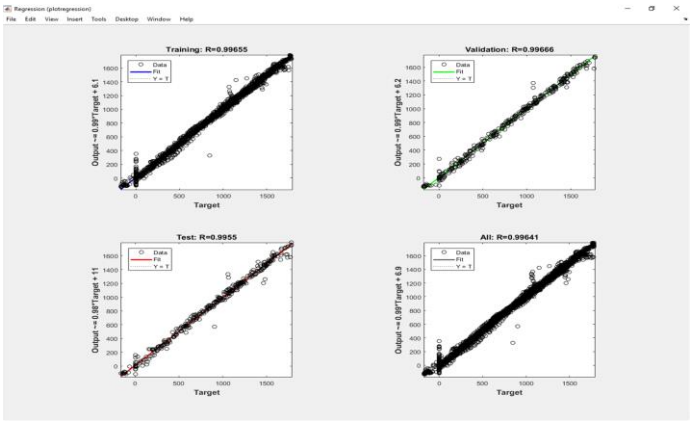

Figura 27: Un gráfico de regresión

Los gráficos de regresión muestran las salidas de la red con respecto a los objetivos para los conjuntos de entrenamiento, validación y prueba. Para un ajuste perfecto, los datos deben caer a lo largo de una línea de 45 grados, donde las salidas de la red son iguales a los objetivos. Para este problema, el ajuste es razonablemente bueno para todos los conjuntos de datos, con valores R de 0,99 o superiores en cada caso. A partir del análisis del gráfico de regresión, podemos ver si se necesitan resultados más precisos. Si es necesario, podemos volver a hacer el entrenamiento, como se indicó anteriormente.

Haga clic en Siguiente en la ventana Red de tren para evaluar la red. Se abre la ventana Evaluar red, como se muestra en la siguiente figura:

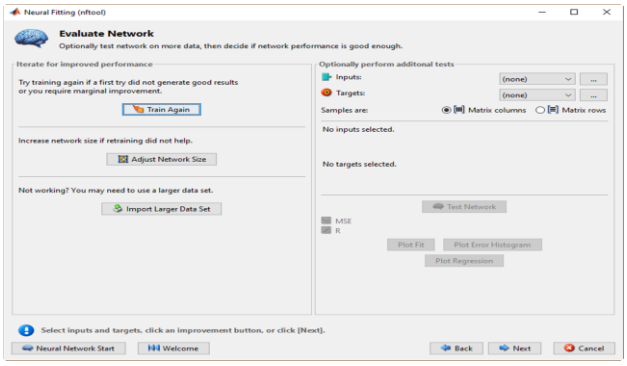

Figura 28: La ventana Evaluar red

En esta ventana, puede probar el rendimiento de la red para evaluar si es lo suficientemente bueno:

- Hay varias herramientas disponibles para mejorar el rendimiento de la red.
- Probar la red con más datos
- entrenarlo de nuevo
- Aumentar el número de neuronas.
- Obtenga un conjunto de datos de entrenamiento más grande

Recuerde: un buen rendimiento en el conjunto de entrenamiento y un peor rendimiento en el conjunto de prueba es indicativo de sobreajuste. Reducir el número de neuronas puede mejorar sus resultados. Si el rendimiento del entrenamiento es bajo, podemos aumentar el número de neuronas.

Haga clic en Siguiente en la ventana Evaluar red para generar las soluciones implementables de su red entrenada. Se abre la ventana Implementar solución, como se muestra aquí:

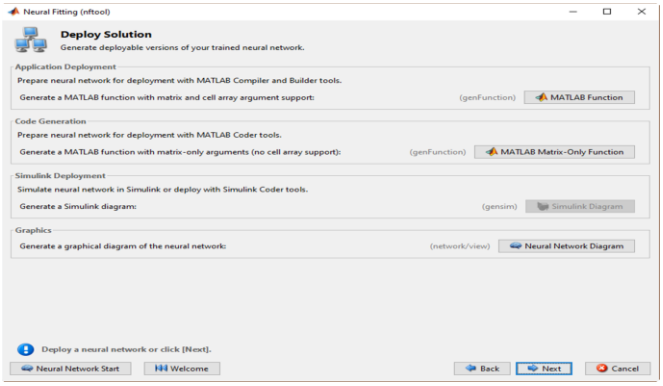

Figura 7.29: La ventana Implementar solución

Hay cuatro opciones disponibles:

- Genere una función de MATLAB compatible con argumentos de matrices y matrices de celdas
- Genere una función de MATLAB con un argumento solo de matriz (sin compatibilidad con matrices de celdas)
- Generar un diagrama de Simulink
- Generar un diagrama gráfico de la red neuronal

Usamos estas opciones para generar una función de MATLAB o un diagrama de Simulink para simular nuestra red neuronal. Además, podemos usar el código o diagrama generado para comprender mejor cómo nuestra red neuronal calcula las salidas a partir de las entradas, o implementar la red con las herramientas de compilación de MATLAB y otras herramientas de generación de código de MATLAB. Para elegir una opción, simplemente haga clic en el botón relativo a la derecha de la ventana (función de MATLAB, función solo de matriz de MATLAB, diagrama de Simulink y diagrama de red neuronal).

Haga clic en Siguiente en la ventana Implementar solución para generar scripts o guardar sus resultados. La ventana Guardar resultados se abre como se muestra en la siguiente figura:

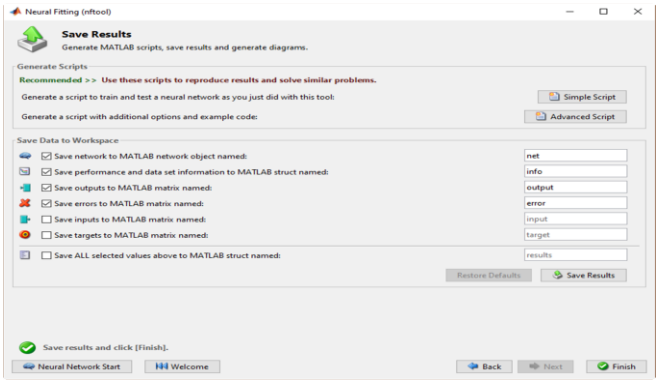

Figura 30: La ventana Guardar resultados

Hay varias opciones disponibles:

1. Genere un script para entrenar y probar una red neuronal como acaba de hacer con esta herramienta
2. Genere un script con opciones adicionales y código de ejemplo
3. Guardar datos en el espacio de trabajo

Para generar un script, haga clic en el botón relativo a la derecha de la ventana. Para guardar los datos obtenidos en MATLAB Workspace, seleccione las opciones deseadas y luego haga clic en el botón Guardar resultados. Los datos seleccionados se almacenan en el espacio de trabajo de MATLAB con el nombre presente en el cuadro de la derecha. Finalmente, haga clic en el botón Finalizar para finalizar el análisis.

¡Es asombroso! Terminamos nuestro primer análisis con redes neuronales y ni nos hemos dado cuenta. Gracias a MATLAB y la aplicación nftool.

Análisis de script

Como hemos dicho antes, analizando el script que acabamos de crear, podemos acceder a las funciones que necesitamos para realizar todas las operaciones. De esta forma, podremos conocer todas las funciones involucradas en las diferentes fases y cómo se utilizarán. Podemos modificarlos para personalizar el entrenamiento de la red. Como ejemplo, echemos un vistazo a la simple script que se creó en la sección anterior:

```
inputs = engineInputs;
targets = engineTargets;
hiddenLayerSize = 10;
net = fitnet(hiddenLayerSize);
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;
[net,tr] = train(net,inputs,targets);
outputs = net(inputs);
errors = gsubtract(outputs,targets);
performance = perform(net,targets,outputs)
view(net)
figure, plotperform(tr)
figure, plottrainstate(tr)
figure, plotfit(targets,outputs)
```

```
figure, plotregression(targets,outputs)
figure, ploterrhist(errors)
```

Después de ver lo fácil que es realizar análisis de redes neuronales con la aplicación nftool, podríamos pensar que el análisis de secuencias de comandos fue mucho más difícil. Comencemos con las dos primeras filas; definen los datos de entrada y salida ya presentes en el espacio de trabajo de MATLAB:

inputs = engineInputs;

targets = engineTargets;

Las filas tercera y cuarta crean la red. La tercera fila define el número de neuronas de la capa oculta. Hemos asignado 10 neuronas a una capa oculta en la sección anterior. La cuarta fila crea la red. Como dijimos más adelante, la red predeterminada para los problemas de ajuste de funciones, usando la función fitnet(), es una red de avance con la función de transferencia tan-sigmoidea predeterminada en la capa oculta y la función de transferencia lineal en la capa de salida. La red creada tiene dos neuronas de salida, porque hay dos valores objetivo-asociados a cada vector de entrada (par y emisiones de óxido nitroso):

```
hiddenLayerSize = 10;
net = fitnet(hiddenLayerSize);
```

Las siguientes tres filas (5,6,7) dividen aleatoriamente los datos disponibles en los siguientes tres conjuntos: entrenamiento, validación y prueba:

```
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;
```

La fila ocho entrena la red. Durante el entrenamiento, se abre la ventana Entrenamiento de redes neuronales. Esta ventana muestra el progreso del entrenamiento y le permite interrumpir el entrenamiento en cualquier momento haciendo clic en Detener entrenamiento. Para entrenar la red, hemos utilizado el algoritmo predeterminado de Levenberg-Marquardt (trainlm). Como dijimos más adelante, hay otros dos algoritmos disponibles: Regularización Bayesiana (trainbr) y Gradiente Conjugado Escalado (trainscg): Para configurar este algoritmo, simplemente escriba net.trainFcn = 'trainbr', o net.trainFcn = 'trainscg' :

[net,tr] = train(net,inputs,targets);

Las siguientes tres filas (9,10,11) prueban la red. Después de entrenar la red, puede usarla para calcular las salidas de la red. El siguiente código calcula las salidas de la red, los errores y el rendimiento general:

```
outputs = net(inputs);
errors = gsubtract(outputs,targets);
performance = perform(net,targets,outputs)
```

La duodécima fila ofrece una vista del diagrama de red:

view(net)

Las últimas filas muestran los gráficos descritos en la sección anterior:

```
figure, plotperform(tr)
figure, plottrainstate(tr)
figure, plotfit(targets,outputs)
figure, plotregression(targets,outputs)
figure, ploterrhist(errors)
```

Anteriormente hemos dicho que, para mejorar el rendimiento de la red, podemos repetir la sesión de entrenamiento varias veces. Esto se debe a los diferentes valores iniciales de peso y sesgo y a las diferentes divisiones de datos en conjuntos de entrenamiento, validación y prueba que se usan cada vez.