Sistemas de control inteligente

Práctica de laboratorio 2

Grado en Ingeniería Informática



Raúl López Llana

Oscar Pozo de Cádiz

# Identificación y control neuronal (I)

* 1. Ejercicio 1

Comenzaremos este ejercicio definiendo los datos de entrada y las clases a las que pertenecen:

Texto

Descripción generada automáticamente

El diseño y entrenamiento del perceptrón se realiza de la siguiente manera:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Después podemos visualizar el resultado ofrecido por el perceptrón, así como su estructura:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Tabla

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

Podemos observar que la red separa correctamente los datos. Tiene dos neuronas de salida ya que estamos clasificando los datos en 4 clases usando números binarios (necesitamos hacerlo así porque el perceptrón usa una función hard-limit que sólo da como resultado 0 ó 1).

Veamos que ocurre si añadimos un nuevo punto [0.0, -1.5] perteneciente a la clase 3:

Texto

Descripción generada automáticamente

Tabla

Descripción generada automáticamente

Podemos observar que el entrenamiento termina porque hemos llegad al límite preestablecido de 1000 épocas, no porque se haya conseguido un entrenamiento óptimo (como sí que ocurrió en el ejemplo anterior).

Gráfico, Gráfico de líneas

Descripción generada automáticamente

También observamos que ahora la red no es capaz de clasificar correctamente los datos. Esto se debe a que, con ese nuevo dato que hemos introducido, el conjunto de datos ya no es linealmente separable.

# Ejercicio 2

Primero defino la función que vamos a utilizar y la muestro gráficamente

Texto

Descripción generada automáticamente

Creo la red neuronal. Indico el tipo de entrenamiento que va a tener y divido los datos en 3 tipos indicando que porcentaje va a tener cada uno de ellos.

Imagen de la pantalla de un celular de un mensaje en letras blancas

Descripción generada automáticamente con confianza baja

Empiezo a entrenar la red y muestro el resultado.

Texto

Descripción generada automáticamente

Primero miramos el resultado de entrenar la red con un número distinto de neuronas y estos son los resultados:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Como se puede observar vemos que con una cantidad de neuronas menor que 7 no llega a aprender los suficiente como para replicar la función.

Con una cantidad mayor a 10 neuronas observamos que comienza a sobre aprender y no consigue generalizar.

Por todo esto vemos que lo más eficiente es usar 7 neuronas.

En este punto comenzamos a cambiar los métodos de entrenamiento para ver cuál es más eficaz.

Interfaz de usuario gráfica, Gráfico, Aplicación

Descripción generada automáticamente

Como se puede observar en los resultados obtenidos, el mejor con diferencia es el método “trainlm” para entrenar una red ya que replica a la perfección la función original.

# Ejercicio 3

Haciendo uso del código proporcionado en el enunciado de práctica, podemos observar los siguientes resultados comparando el uso de dos conjuntos de datos distintos.

Un conjunto de letras negras en un fondo blanco

Descripción generada automáticamente con confianza media

Claramente se puede observar que existe una gran diferencia entra la fila de arriba y la de abajo ya que se debe a que son dataset distintos y sus datos varian completamente.

Lo que si podemos destacar que tienen en común a ambos es el propio diseño de la red.

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Vista esta gran diferencia, pasamos con los resultado de aplicar distintos métodos de entrenamiento sobre el dataset “simplefit\_dataset”.

Interfaz de usuario gráfica, Gráfico, Aplicación

Descripción generada automáticamente

Se puede observar que el método de entrenamiento que comete mejor error es el “Trainlm” y el que más es “Trainrp”.

Si pasamos a evaluar el numero de épocas que requiere obtener el modelo vemos lo siguiente.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Pese a que el entrenameiento Trainlm obtiene el menor error de todos los entrenamiento, requiere 237 época mientras que el resto tienen solo una media de 34 épocas.

Tomando como referencia el modelo de entrenamiento “trainlm” y el dataset de “simplefit\_dataset”, vamos a ver como afecta cambiar el porcentaje de datos utilizados para entrenar, validar y test.

Gráfico

Descripción generada automáticamente

Pese a que vemos que con una distribucción de 90% 5% 5% tiene un menor error, esto no es lo más optimo ya que está sobreaprendiento y no será capaz de generalizar.

Despues de ese vemos que la mejor distribucción es de 70 % 15% 15%.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

# Ejercicio 4

En este ejercicio vamos a comparar varios métodos de entrenamiento de una red para detectar patrones en un data set sobre el cáncer.

Debido a que cada vez que entrenemos la red nos dará un rendimiento ligeramente diferente, a la hora de compararlas haremos 10 entrenamientos y tenderemos en cuenta la media de sus rendimientos, para minimizar esas pequeñas diferencias.

Comenzamos con el método de entrenamiento de ejemplo:

Texto

Descripción generada automáticamente

Con los siguientes resultados:

Texto

Descripción generada automáticamente

Y su último entrenamiento obtuvo la siguiente matriz de confusión:

Aplicación, Tabla

Descripción generada automáticamente

Podríamos comprobar si aumentar el número de neuronas en la capa oculta mejora el resultado del entrenamiento. Probemos con 20 neuronas:

Texto

Descripción generada automáticamente

Aplicación, Tabla

Descripción generada automáticamente

Comprobamos que aumentar el número de neuronas en la capa oculta no mejora lo más mínimo el rendimiento de la red, por lo que volvemos a las 10 neuronas.

Vamos ahora a comprobar el resultado de modificar la proporción de datos para entrenamiento, validación y testeo:

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Hemos comprobado que aumentar la cantidad de datos de entrenamiento mejora el resultado de la red. Ahora vamos a comprobar qué efectos tiene el cambiar la función de entrenamiento:

* Levenberg-Marquart:
  + Imagen que contiene Texto

    Descripción generada automáticamente
  + Texto

    Descripción generada automáticamente
  + Texto

    Descripción generada automáticamente
* Regularización Bayesiana:
  + Imagen que contiene Texto

    Descripción generada automáticamente
  + Texto

    Descripción generada automáticamente
  + Texto

    Descripción generada automáticamente
* BFGS cuasi-Newton:
  + Imagen que contiene Texto

    Descripción generada automáticamente
  + Captura de pantalla de un celular

    Descripción generada automáticamente
  + Texto

    Descripción generada automáticamente con confianza baja

Todos los algoritmos de aprendizaje que hemos comprobado han resultado ser mejores que el que usa Matlab por defecto (Gradiente conjugado escalado), al menos para este data set.

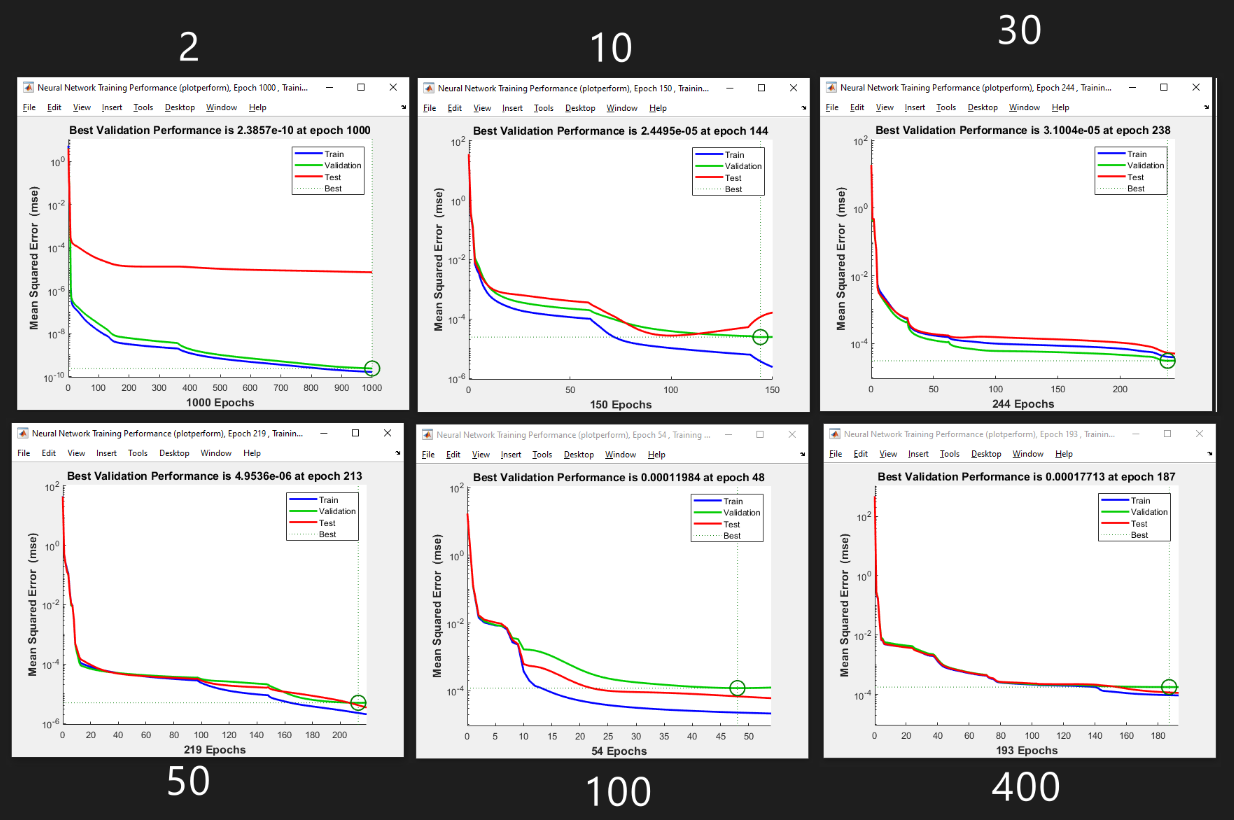
De todos los probados, el de regulación bayesiana ha dado los mejores resultados.

También hemos notado que el uso de algunos de estos algoritmos también forzaba el uso de una función de rendimiento en concreto: tanto el aprendizaje bayesiano como el Levenberg-Marquartusan error cuadrático medio.

# Identificación y control neuronal (II)

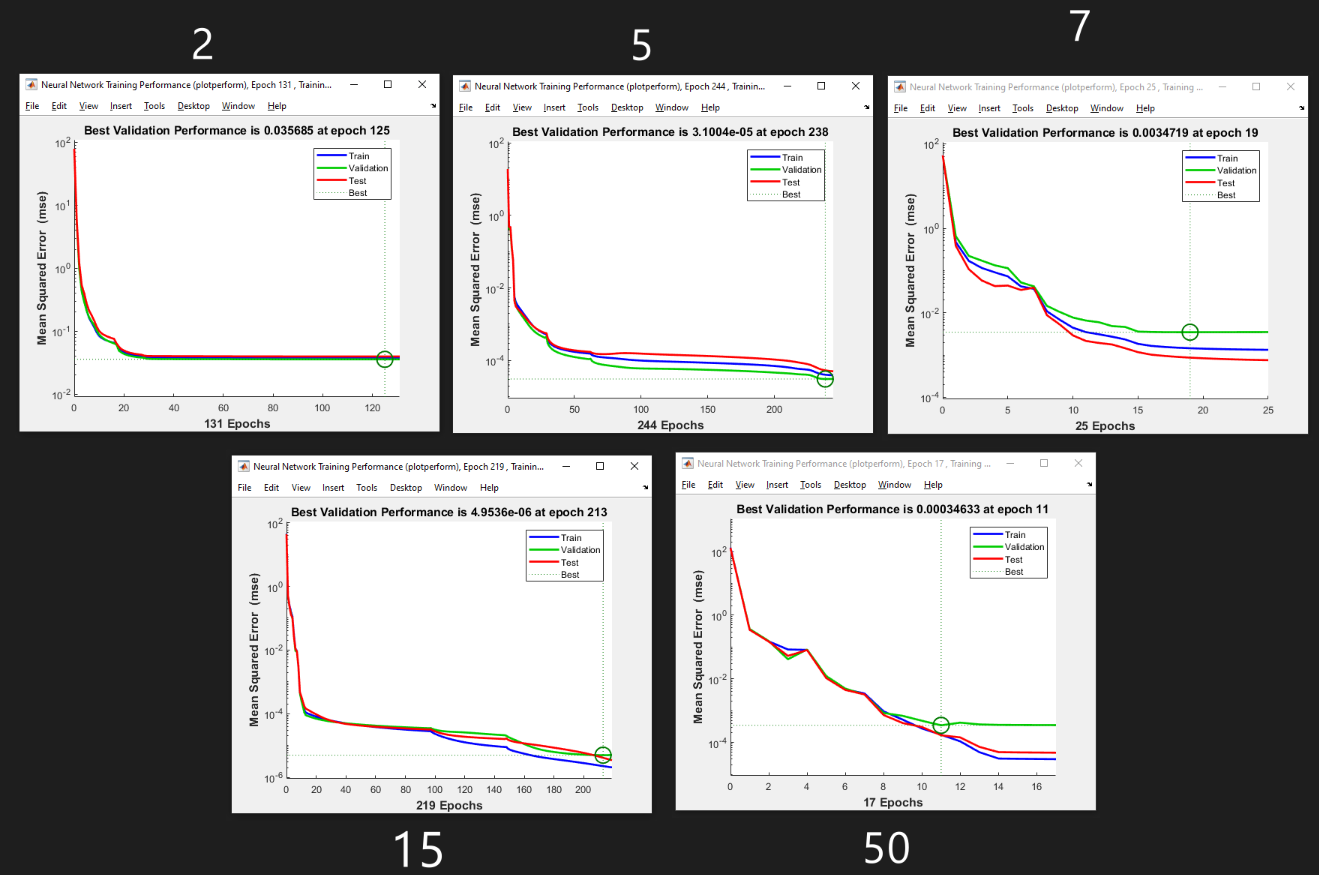
Tras seguir todos los pasos indicados en el enunciado y construida una red, comenzaremos a modificarla para ver cuáles son los valores que hacen que nuestra red neuronal replique con una buena precisión el controlador que nos han proporcionado.

Para esto comenzamos viendo cómo afecta el número de datos de entrenamiento que se le dan a la red.



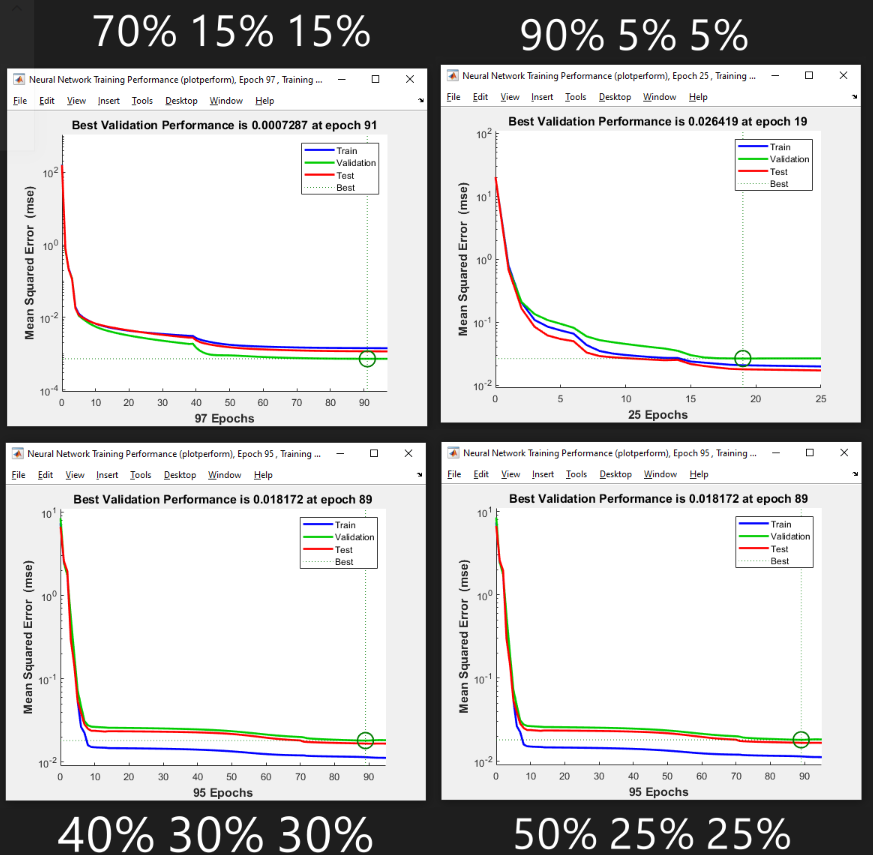
Como se puede observar en la foto superior vemos que no por dar muchos datos a la red va a tener un mejor comportamiento ya que comienza a sobreaprender. Por eso vemos que uno de los más óptimos es el de 30 datos de muestra.

Continuamos ahora cambiando el número de neuronas en la capa oculta.

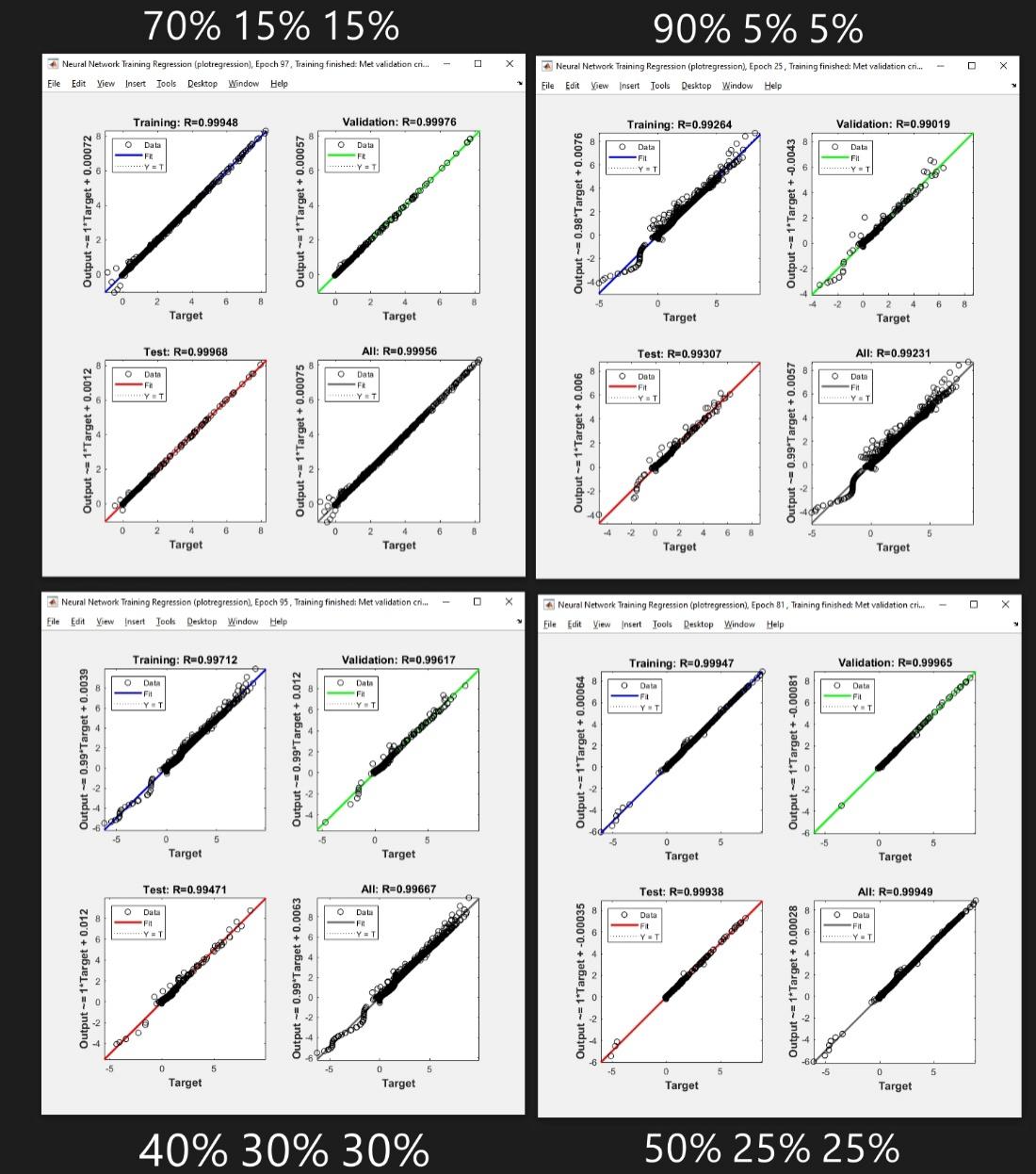


Antes incluso de realizar las pruebas que se observan en la imagen superior, sabemos que el número de neuronas no pude ser muy elevado ya que el número de datos de entrenamiento no es muy alto. Observamos que con 5 neuronas y con 15 dan unos valores muy buenos, pero se puede llegar a observar que con 15 está justo empezando a sobreaprender y deja de ser tan preciso. Por esto mismo nos quedamos con el de 5 neuronas.

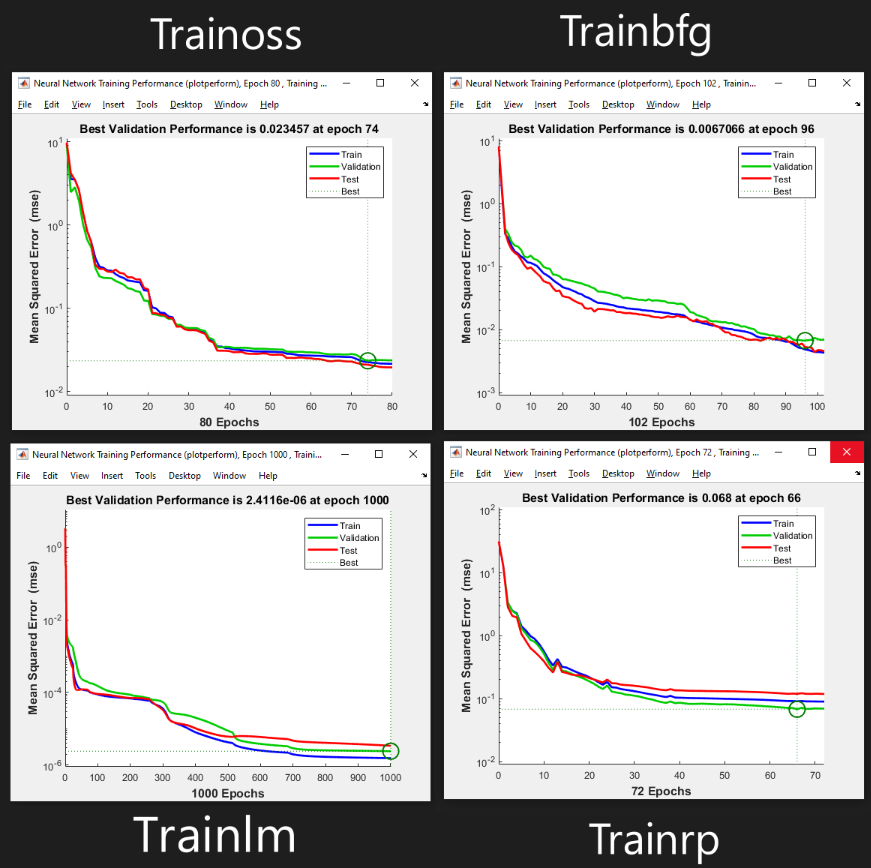
Pasamos ahora con la distribución de los datos. Esto mismo ya se vio en ejercicios anteriores que una distribución de 70 % de entrenamiento, 15% de validación y 15% de prueba era una de las más eficientes y como se puede observar a continuación, sigue siendo el caso.



Junto a esto también podemos ver que la que tiene una mejor regresión es la de 70 % de entrenamiento, 15% de validación y 15% de prueba

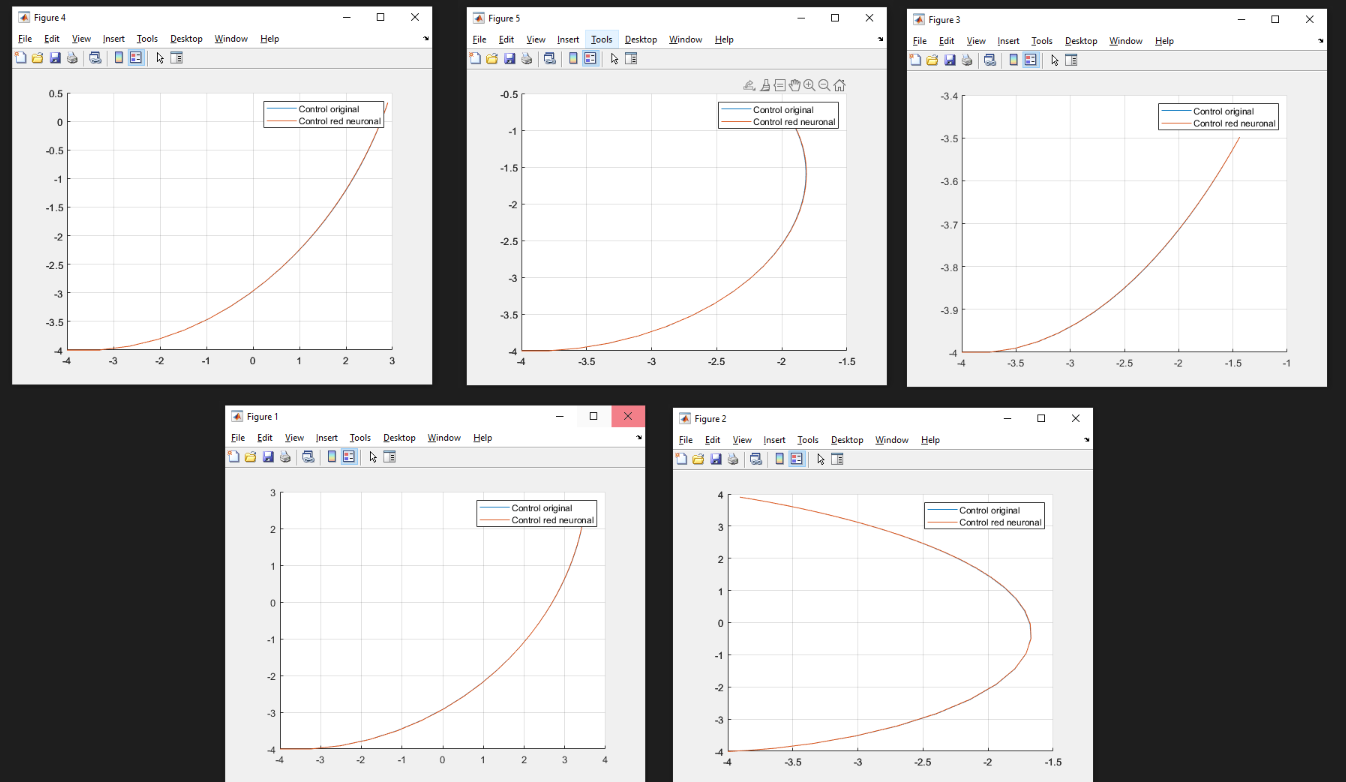


Para finalizar, probamos con distintos métodos de entrenamiento sobre una red que conteniente 30 datos de prueba y 5 neuronas en su capa oculta.



Observamos que el método que nos proporciona un mejor resultado es el “trainlm”. Este resultado se ve limitado ya que ha llegado a 1000 épocas y Matlab lo detiene.

Con toda esta información, hacemos uso de la red neuronal con 5 neuronas en su capa oculta, con 30 datos para entrenarla, una distribución de 70 % de entrenamiento, 15% de validación y 15% de prueba y utilizando “trainlm” como método de entrenamiento.



Como se puede observar el resultado es perfecto ya que consigue replicar su funcionamiento con muy buena precisión.

En cambio, si utilizásemos una red con unos datos que hemos visto que no han sido tan buenos, vemos resultados como los siguientes.

