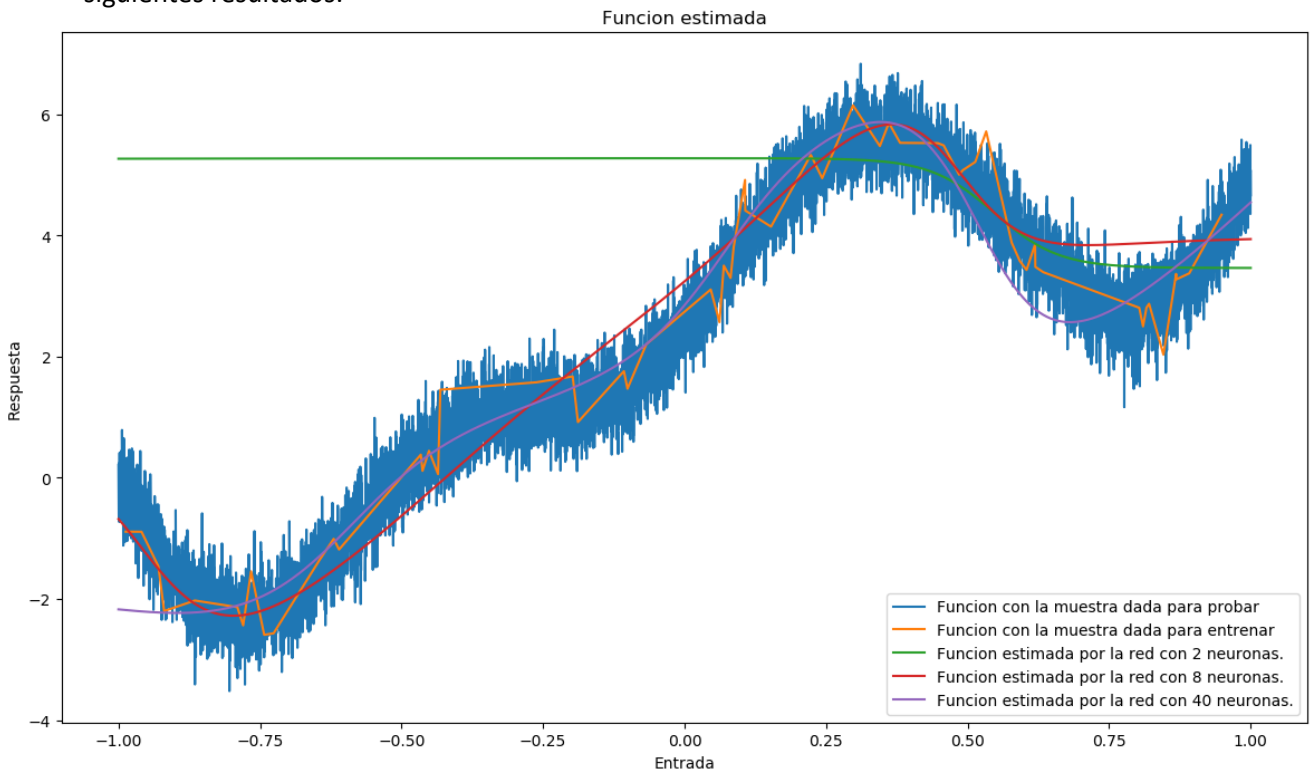
Universidad Simón Bolívar
Introducción a las Redes Neuronales Artificiales
Rafael Cisneros
Carnet: 13-11156

Tarea 4

1)

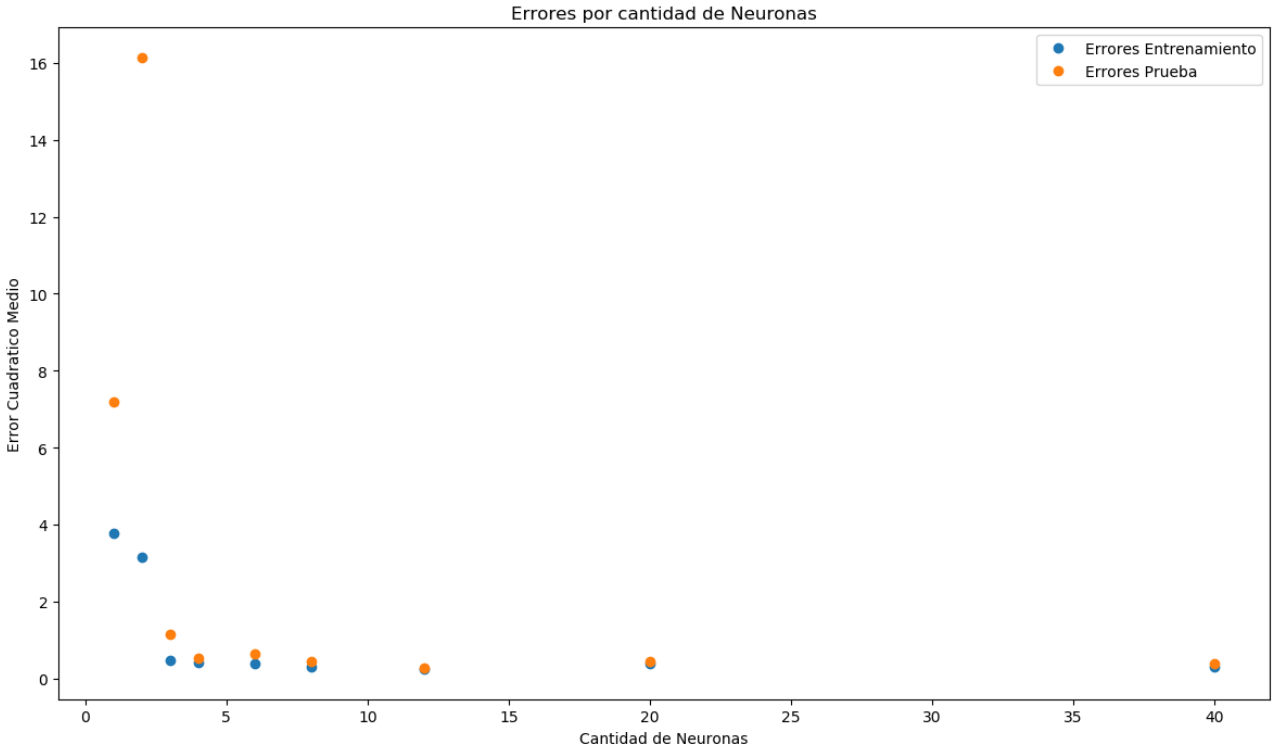
MLP

Se implementó una red neuronal multicapas utilizando el lenguaje de programación Python. La arquitectura de la red consta de 1 capa oculta, con distintas cantidades de neuronas en dicha capa según se experimentaba, todas ellas con funciones de activación logística; 1 capa de salida con una única neurona con función de activación lineal. Se procedió a realizar el entrenamiento de dicha red usando el conjunto encontrado en el archivo “reglin_train.csv”, en un total de 700 épocas usando el algoritmo de descenso de gradiente, y realizando la verificación de la red con el conjunto de prueba encontrado en el archivo “reglin_test.csv”. Se obtuvieron los siguientes resultados:



Comparación función original con los datos dados y la función obtenida por la red con 2,8 y 40 neuronas en la capa oculta.

Se puede deducir de la gráfica que es necesario un mínimo de neuronas en la capa oculta para lograr una buena aproximación, lo que hace pensar que cada neurona podría representar un grado extra en el polinomio que aproxima la función representado en la red. Se observa que en los valores centrales la función que parece aproximar mejor es la generada por la red de 40 neuronas, sin embargo tener gran cantidad de neuronas puede afectar la generalización de la red.



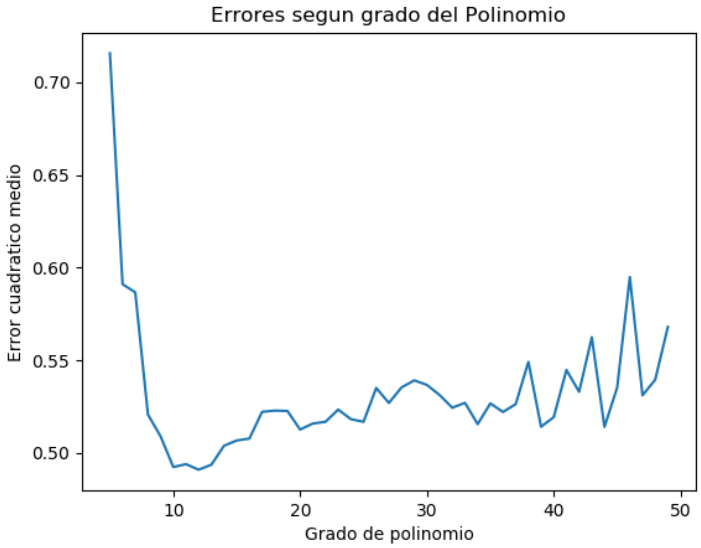
Errores cuadráticos medios según la cantidad de neuronas en la capa oculta.

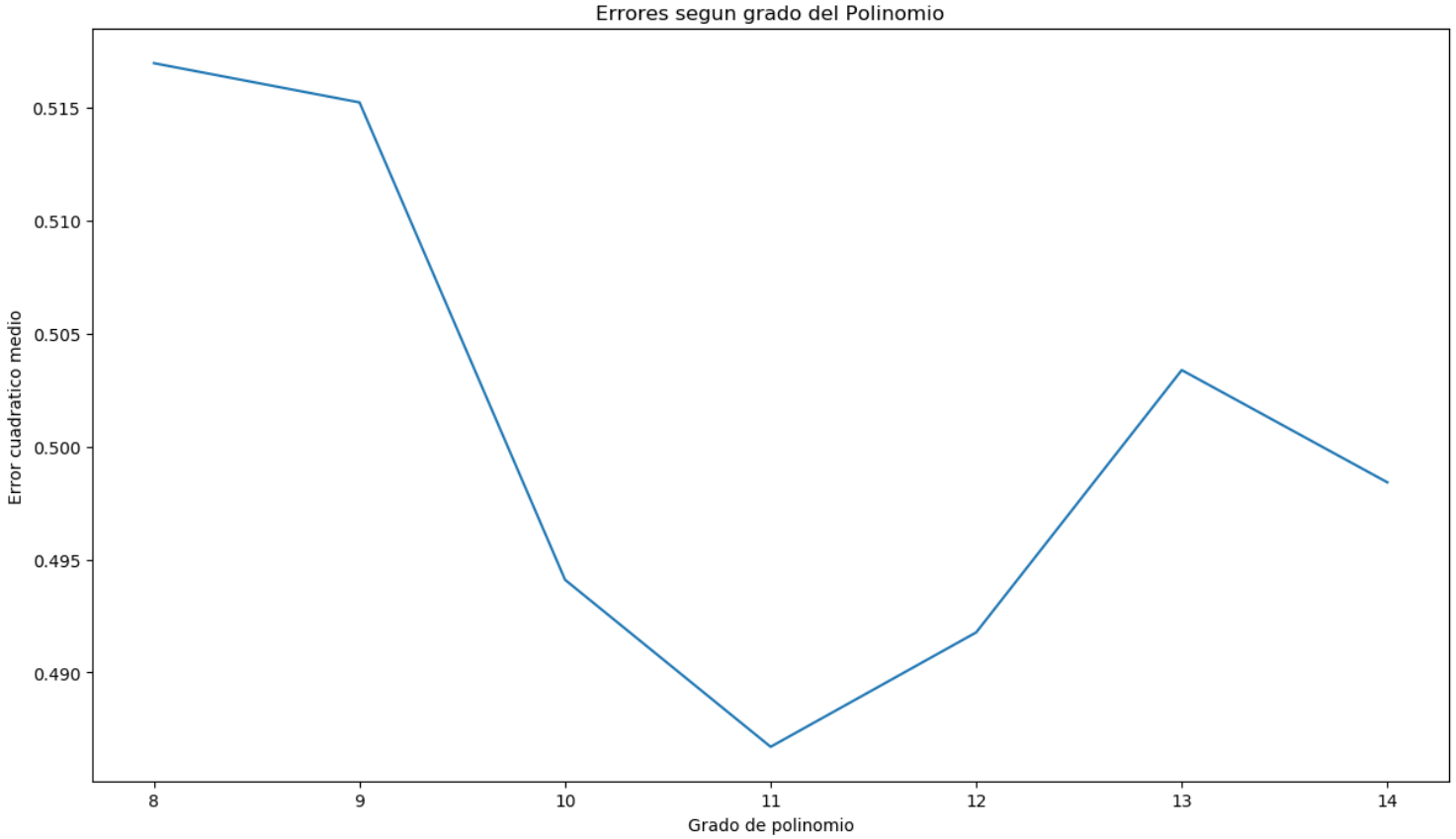
En esta grafica se observa que se obtienen mejores resultados de generalización cuando se tienen alrededor de 12 neuronas que cuando se tienen 40 neuronas, confirmando que es mejor tener una red más sencilla para la generalización. En todos los casos se obtienen errores menores cuando se realiza el entrenamiento que cuando se realiza la prueba.

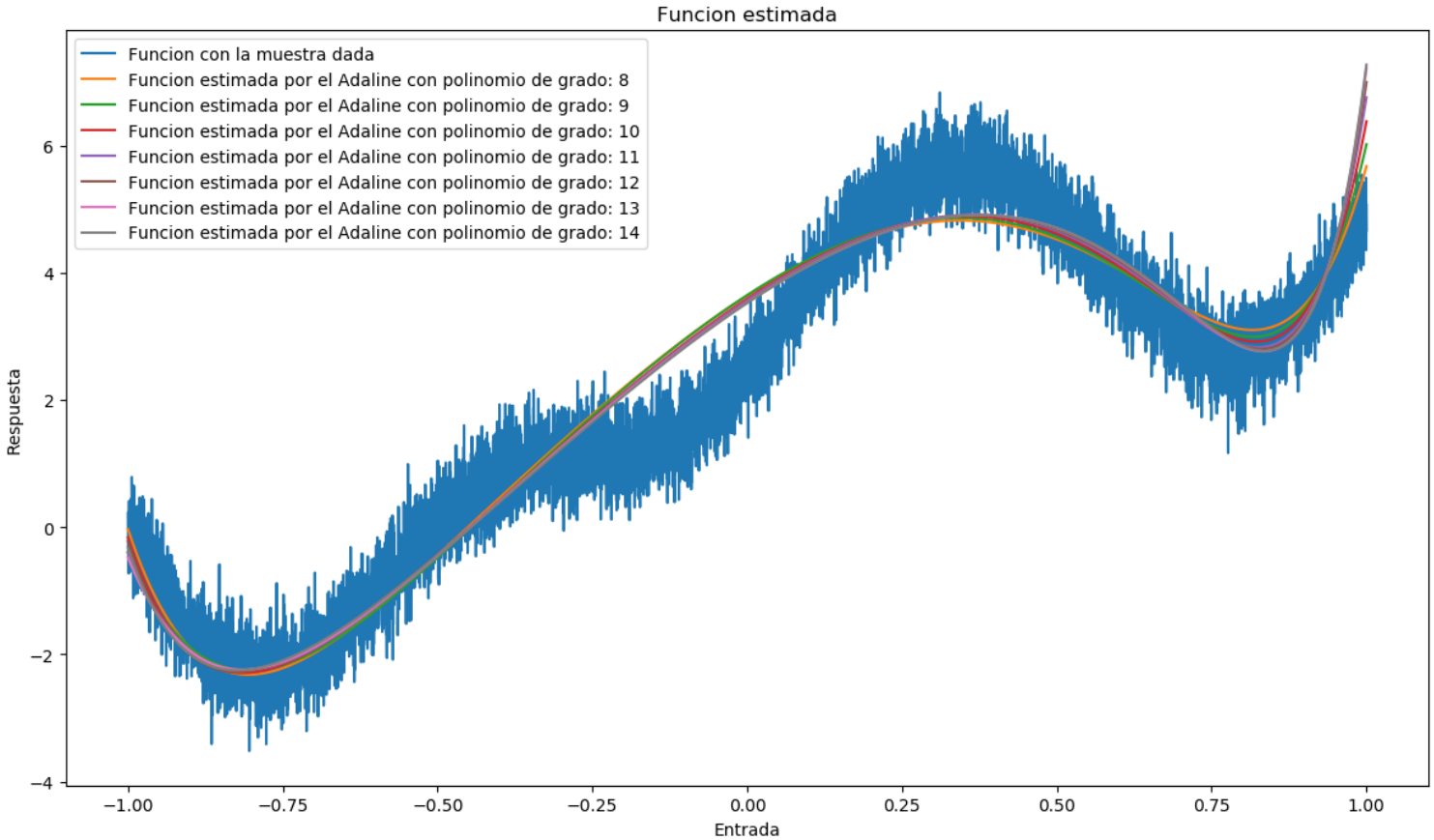
Adaline

Se resolvió el mismo problema haciendo uso de un dispositivo Adaline, una única neurona que recibe un solo parámetro, dicho parámetro es pasado a través de cada peso con una potencia distinta, teniendo tantos pesos como el grado del polinomio deseado.

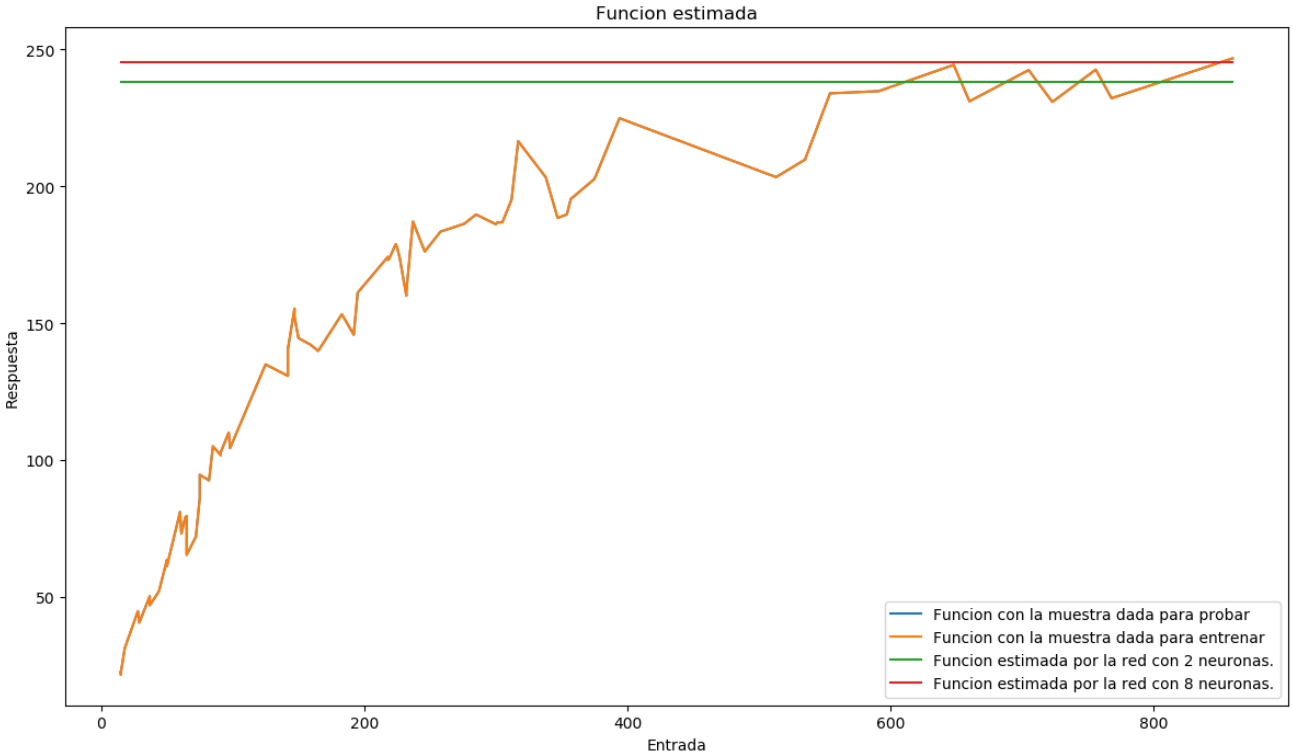
En la siguiente grafica se observa el error cuadrático medio obtenido con distintos grados de polinomio, teniendo mejores resultados cuando el grado del polinomio es de 11



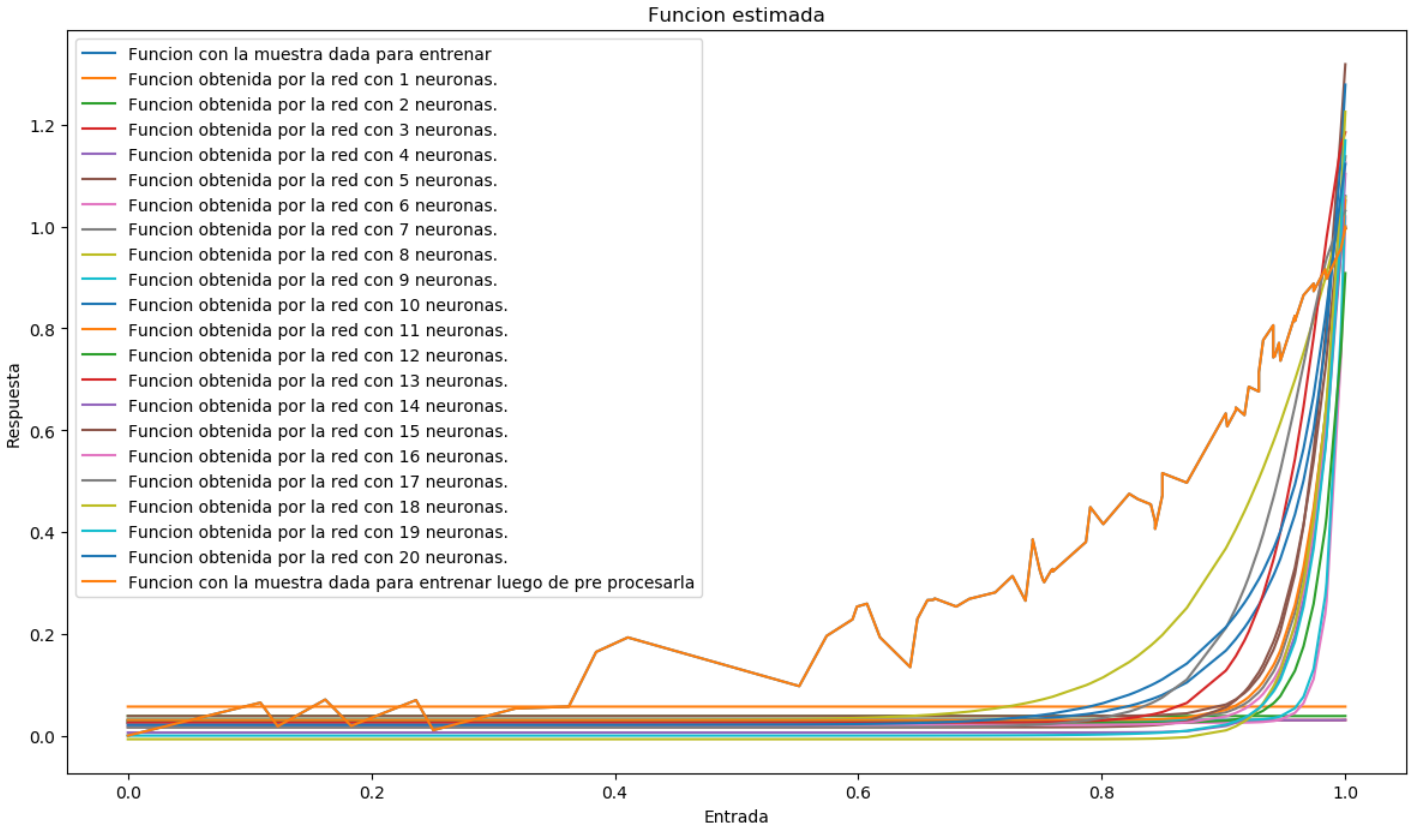

Errores segun grado del Polinomio
Error cuadratico medio
0.515
0.510
0.505
0.500
0.495
0.490
8
9
10
11
12
13
14
Grado de polinomio


Funcion estimada
Funcion con la muestra dada
Funcion estimada por el Adaline con polinomio de grado: 8
Funcion estimada por el Adaline con polinomio de grado: 9
Funcion estimada por el Adaline con polinomio de grado: 10
Funcion estimada por el Adaline con polinomio de grado: 11
Funcion estimada por el Adaline con polinomio de grado: 12
Funcion estimada por el Adaline con polinomio de grado: 13
Funcion estimada por el Adaline con polinomio de grado: 14
Respuesta
6
4
2
0
−2
−4
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
Entrada

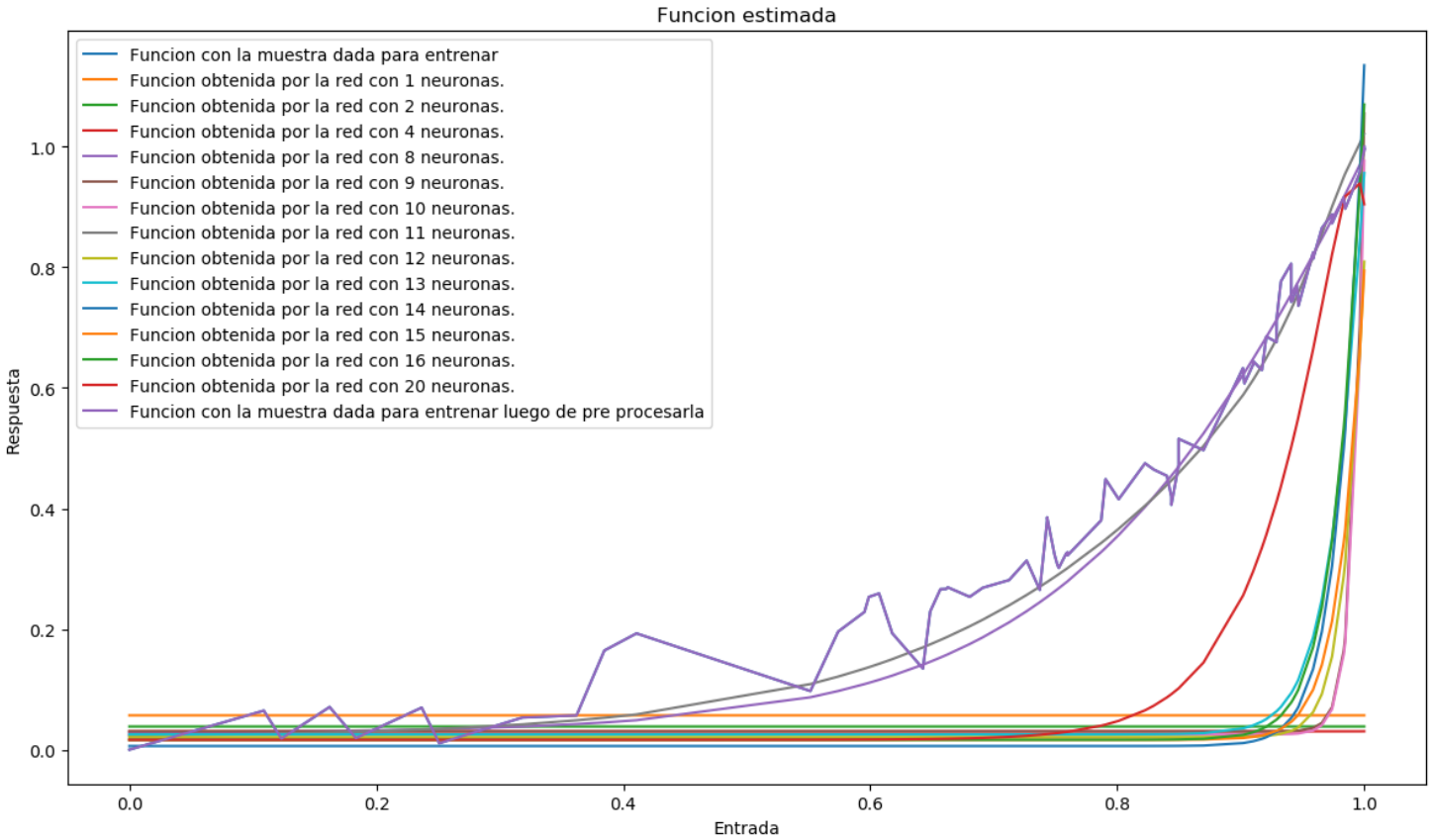
2) Se realizó una implementación de una MLP cuya arquitectura consta de una capa oculta con cantidad variable de neuronas para determinar la cantidad optima realizando pruebas, usando como función de activación la función logística, que resulto con mejores resultados sobre la tangente hiperbólica; Y una capa de salida con una única neuronal con función de activación lineal, ya que no queremos limitar la salida a un rango especifico. Se comenzó el enteramiento y debido a los malos resultados se procedió a realizar un pre procesamiento de los datos con dos estrategias: utilizando la media y varianza de los datos, y tipificando con el máximo y mínimo, resultando con mejores resultados la tipificación de máximo y mínimo. Se inició con una condición de parada de 700 épocas, pero al no conseguir resultados satisfactorios se aumentó a 2000 épocas, donde se encontraron mejores resultados con cierta cantidad de neuronas. Las mejores cantidades de neuronas fueron de 8 y 11 con errores cuadráticos medios en el conjunto de pruebas (se usó el mismo conjunto de pruebas que en el entrenamiento, pero pasando los datos sin realizar entrenamiento) de 0.003836 y 0.002769 respectivamente. El algoritmo de aprendizaje usado fue descenso de gradiente, ya que el problema es de aproximación a una función. La red obtenida puede ser usada en una red más compleja, agregando una capa inicial de una neurona que haga el trabajo de pre procesamiento del dato, y otra capa posterior a la capa de salida, que revierta el trabajo del pre procesamiento.



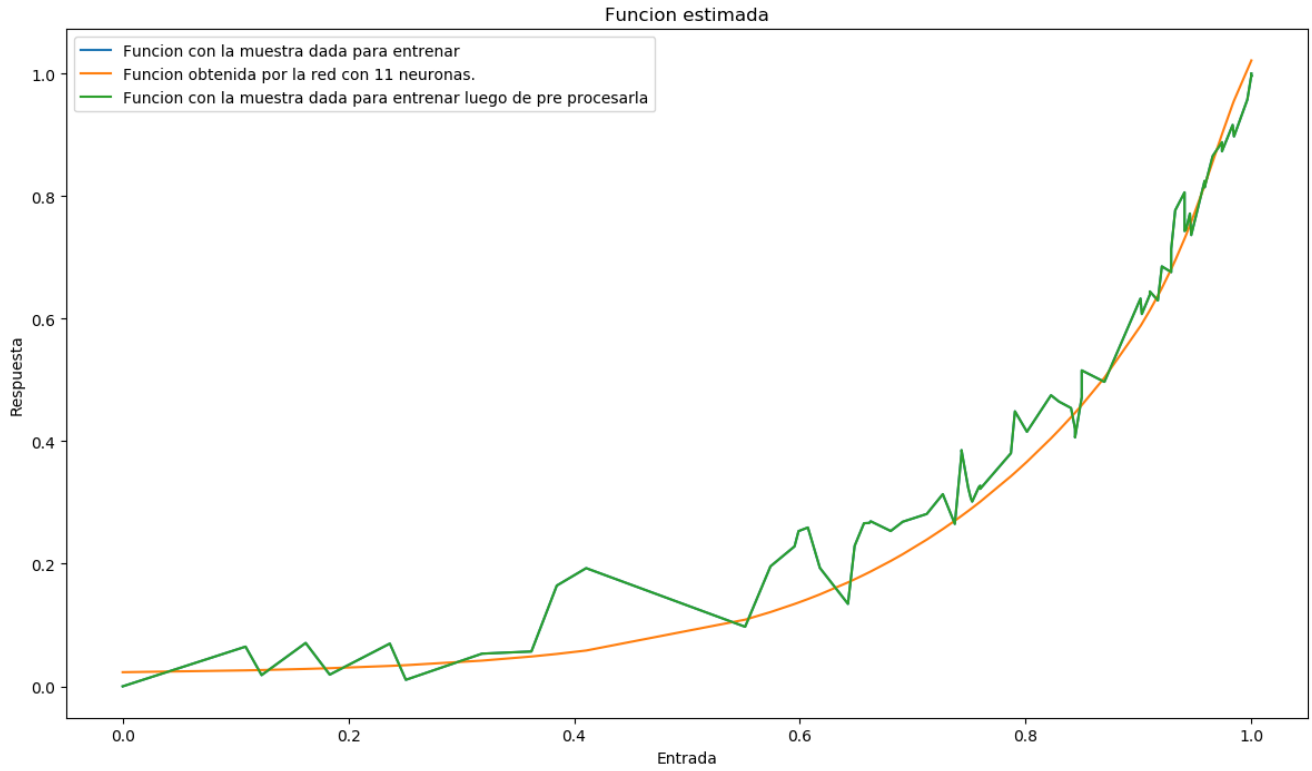
Funciones obtenidas sin procesar los datos



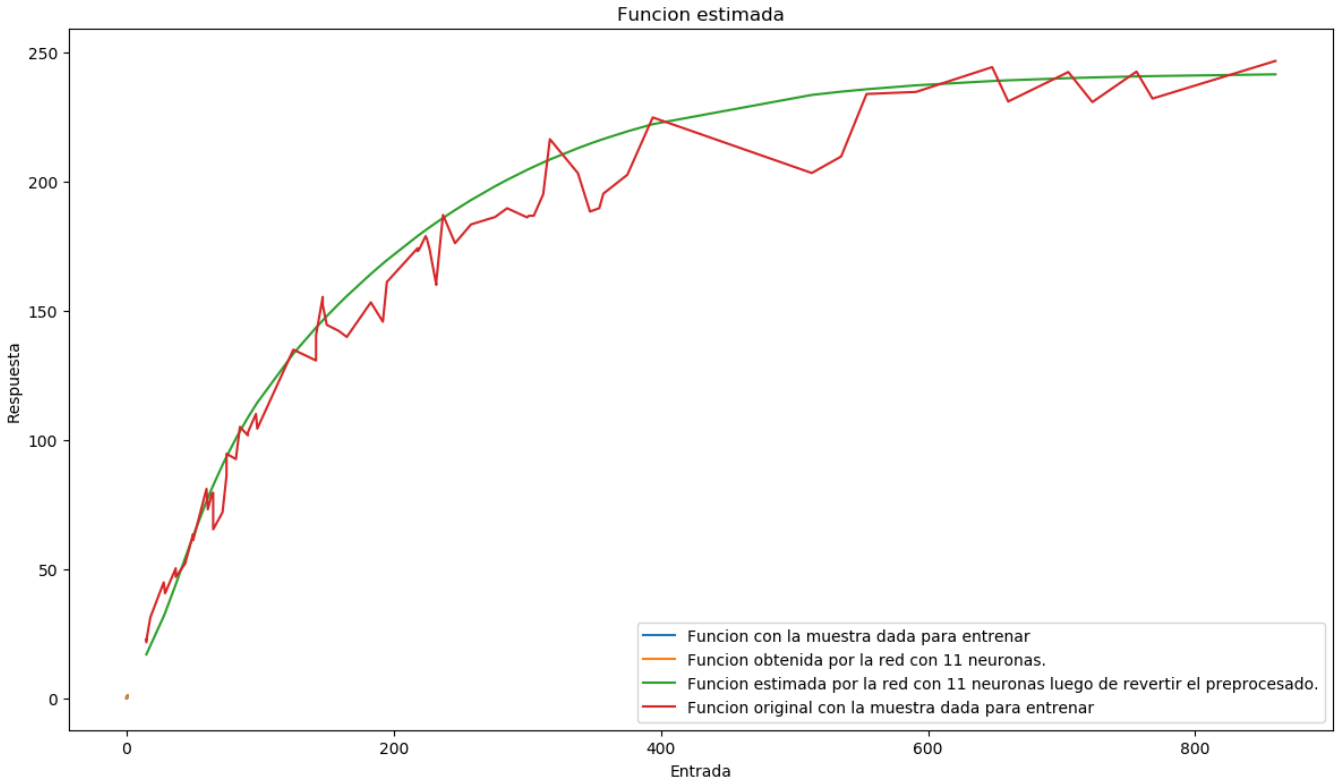
Grafica de las funciones obtenidas entrenando con 700 épocas.



Grafica de las funciones obtenidas entrenando con 2000 épocas.



Grafica de las funciones con los datos originales pre procesados y la salida original de la red.



Grafica de las funciones con los datos originales y la salida procesada de la red.