REDES DE NEURONAS – PRÁCTICA 1

Jorge Hevia Moreno – 100317565

Luis Víctor Hevia Moreno – 100317546

Contents

Introducción

En esta práctica se ponía como objetivo abordar un problema de regresión lineal utilizando el modelo lineal Adaline y el modelo no lineal de Perceptron Multicapa.

Este problema de regresión se aplicaría sobre un conjunto de datos acerca del hormigón para determinar su resistencia a partir de distintos datos como la edad de éste, sus componentes y sus cantidades.

La primera parte de la práctica consiste en programar Adaline y realizar una serie de pruebas con los datos del hormigón proporcionados para un posterior análisis.

La segunda parte consiste en utilizar la librería RSNNS en el lenguaje R para realizar una serie de pruebas con los datos y analizar dichas pruebas.

Parte 1 – Adaline

Introducción al programa

En esta parte se nos pedía programar Adaline en un lenguaje a nuestra elección. Nosotros escogimos Python por su sencillez, potencia y sus librerías de matemáticas y plotting.

Para poder ejecutar el programa, es necesario tener los siguientes paquetes instalados:

- Python 2.7
- Tkinter
    - En debian y derivados se instala mediante: *apt install python-tk*
- Las librerías de Python:
    - Numpy
    - Pandas
    - Matplotlib
    - Que pueden ser instaladas mediante pip
        - *pip install pandas numpy matplotlib*

Estructura del programa

El programa está escrito en un único archivo de Python (.py) con las siguientes secciones:

1. La primera sección importa las librerías necesarias y prepara los archivos y variables iniciales para poder ejecutar adaline. Además, esta sección contiene la prevenciones de errores (líneas 1 a 73).
2. La segunda sección contiene todas las *funciones* que se ejecutarán, con comentarios sobre qué hace cada función, su entrada y su salida (líneas 74 a 182).
3. La tercera sección contiene la sección principal del programa, donde se harán las llamadas a las funciones con los datos procesador en la primera sección (líneas 182 a 238).

Ejecución del programa

Para ejecutar el programa se hará desde una terminal con el siguiente comando:

```
python adaline.py <training-file> <validation-file> <test-file> <number of inputs/weights> <number of cycles> <learnfactor>
```

En el caso de que no se introduzcan los suficientes argumentos o los archivos sean incorrectos, se notificará y se pondrá un ejemplo de caso de uso.

Análisis de los resultados

El programa se ha ejecutado nueve veces con los siguientes argumentos:

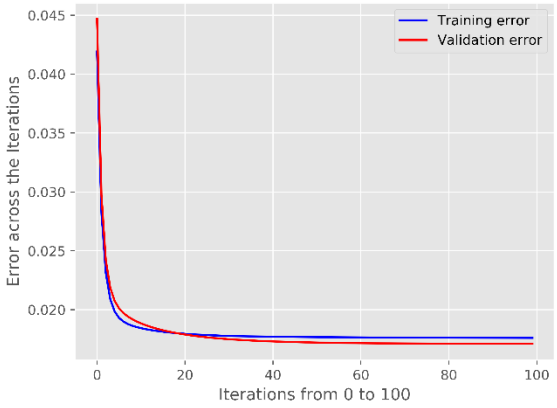
Número de ciclos	Factor de Aprendizaje
100	0.01
100	0.001
100	0.0001
1000	0.01
1000	0.001
1000	0.0001
2000	0.01
2000	0.001
2000	0.0001

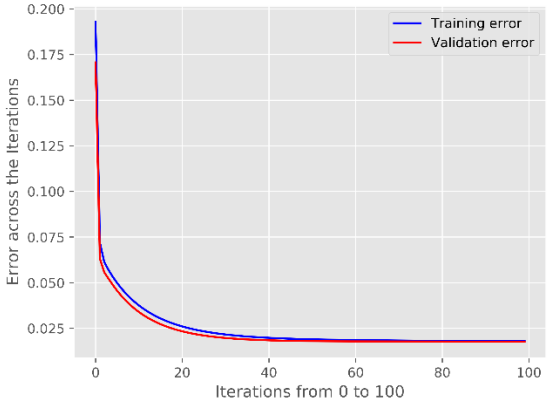
A continuación, se desgranarán los resultados obtenidos por cada una de las ejecuciones y después se realizará un análisis global de los resultados obtenidos.

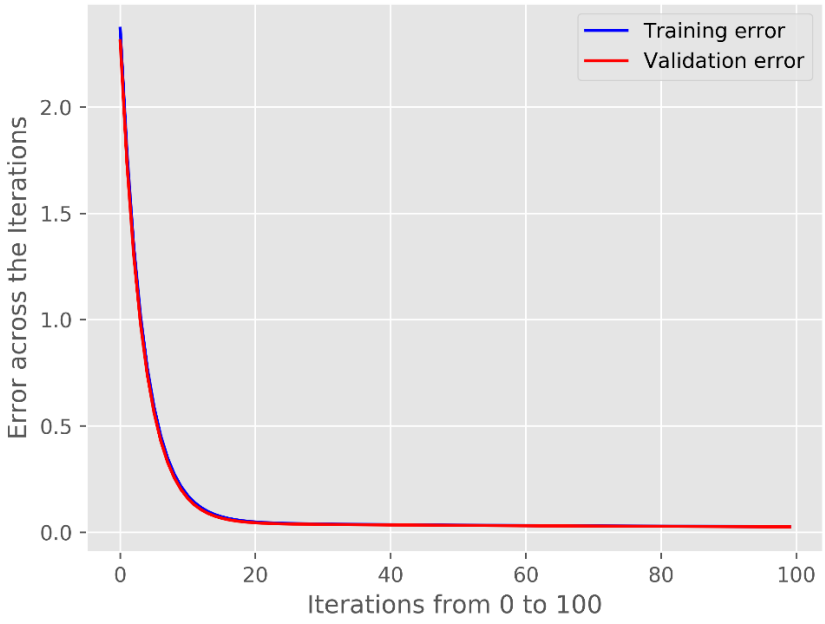
En función de los ciclos

100 ciclos

Para 100 ciclos se puede comprobar que no son suficientes para que se estabilice adaline hasta que disminuimos el factor de aprendizaje a 0.0001, y aun así todavía se podría estabilizar más.





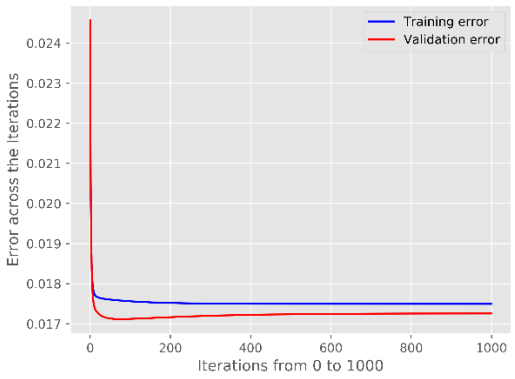


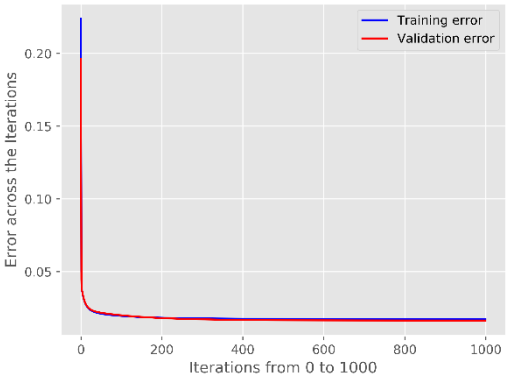
En orden:

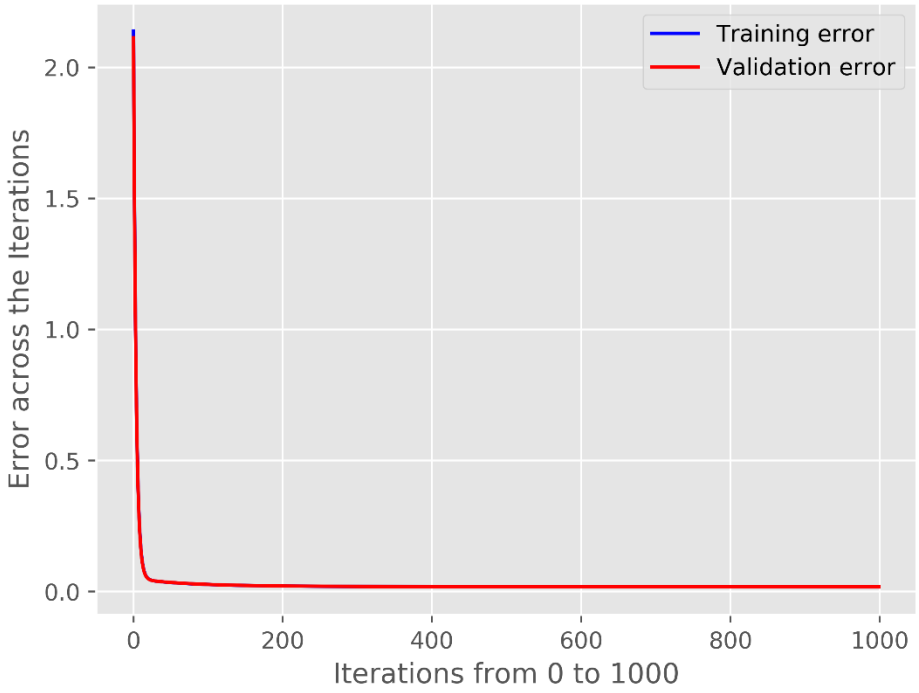
- Factor de aprendizaje 0.01
- Factor de aprendizaje 0.001
- Factor de aprendizaje 0.0001

1000 ciclos

Con 1000 ciclos se aprecia mejor que la línea se estabiliza, y que los valores del factor de aprendizaje se ajustan más cuanto más pequeño es éste.





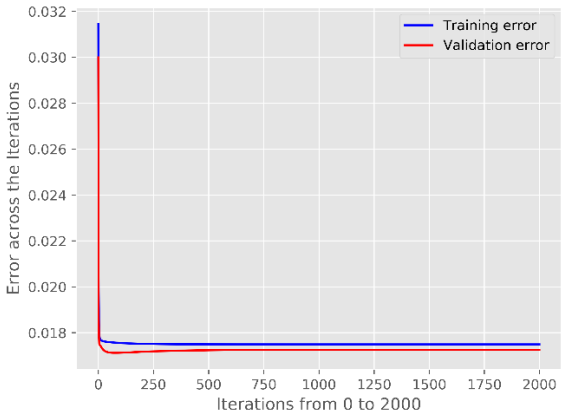


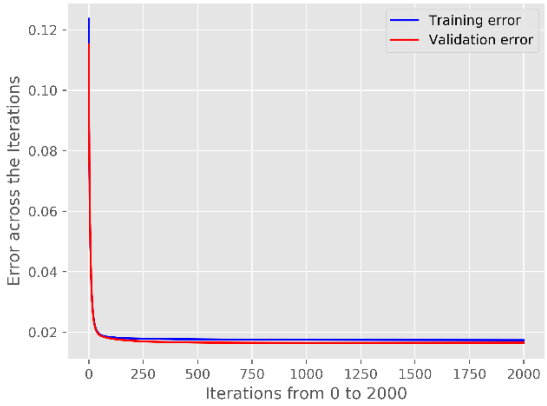
En orden:

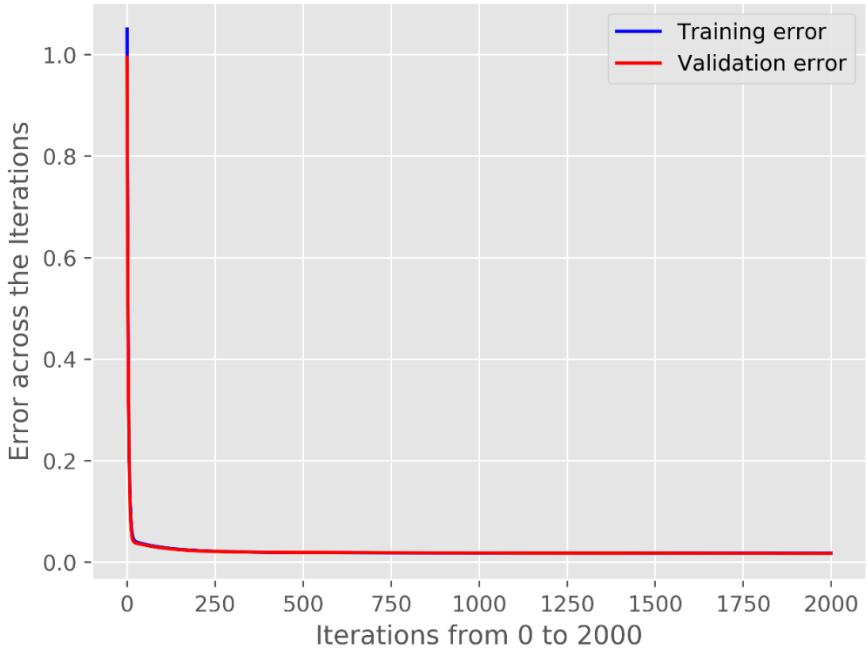
- Factor de aprendizaje 0.01
- Factor de aprendizaje 0.001
- Factor de aprendizaje 0.0001

2000 ciclos

Al hacer 2000 ciclos ya podemos asegurar que la función se ha estabilizado por completo (o casi), lo que nos indica que independientemente del factor de aprendizaje que utilicemos, deberíamos correr adaline durante al menos 2000 ciclos para obtener unos resultados más estables.





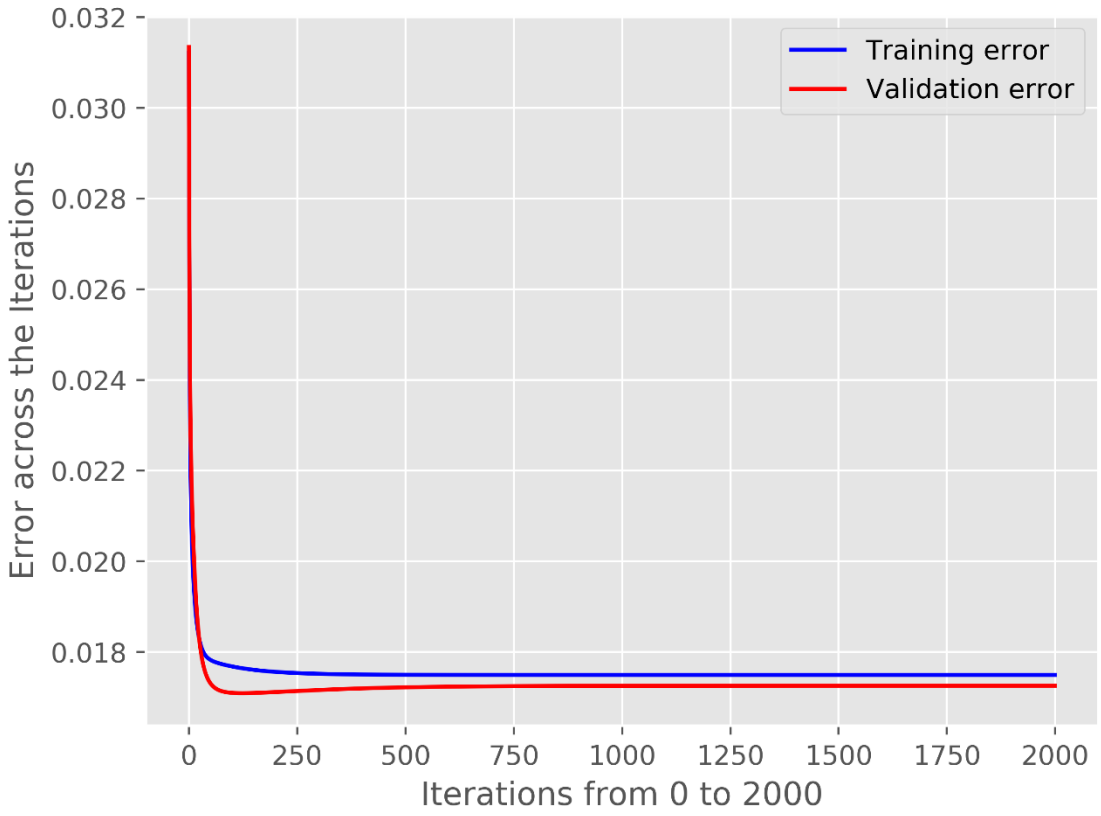


En orden:

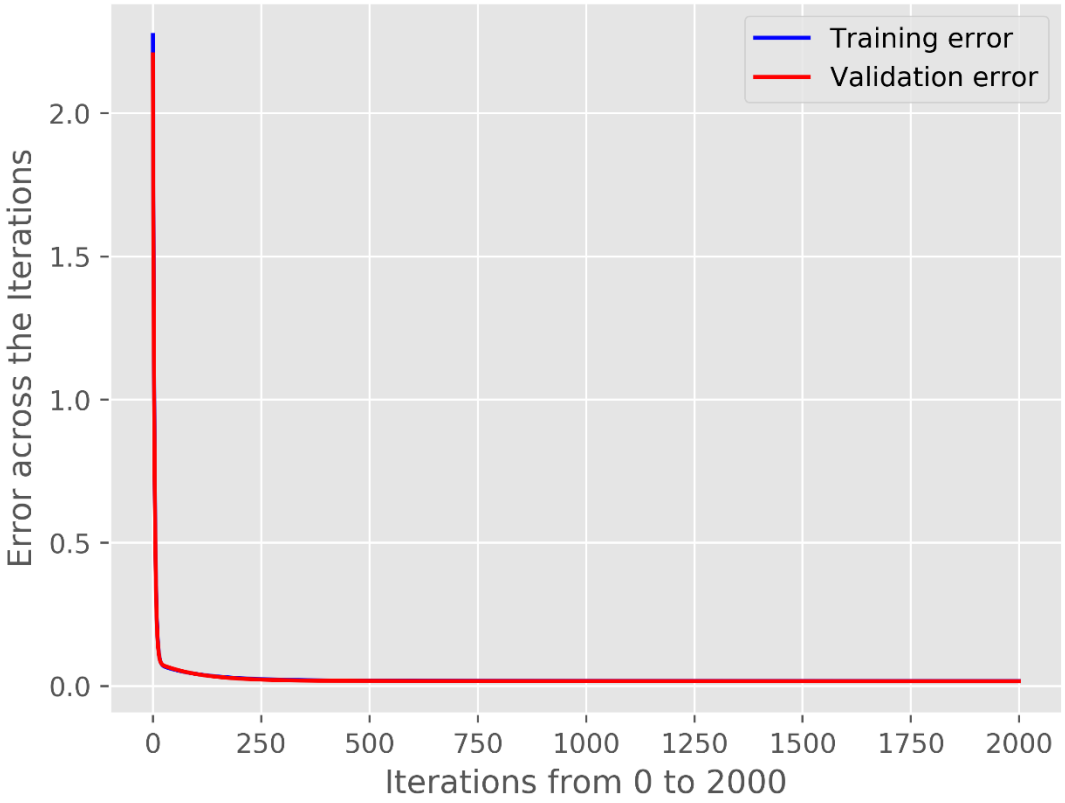
- Factor de aprendizaje 0.01
- Factor de aprendizaje 0.001
- Factor de aprendizaje 0.0001

En función del factor de aprendizaje

Como se ha podido observar anteriormente, al disminuir el factor de aprendizaje hacemos que la red sea más precisa entre los datos de entrenamiento y los de validación. A continuación, se muestra (para 2000 ciclos) la diferencia entre un factor de aprendizaje de 0.01 y 0.0001:



1Factor de aprendizaje 0.01 (2000 ciclos)



2Factor de aprendizaje 0.0001 (2000 ciclos)

Como se puede ver, al utilizar una menor tasa de aprendizaje la función tarda un poco más en estabilizarse (en torno al ciclo 250) mientras que con una mayor tasa de aprendizaje se estabiliza un poco antes de los 250 ciclos, aunque de forma un poco más *brusca*.

Análisis de los pesos obtenidos

A continuación, se muestra una tabla unificando todos los pesos y umbrales obtenidos.

W1	W2	W3	W4	W5	W6	W7	W8	THR	TEST
0.718123	0.516074	0.25216	-0.1592	0.156007	0.13505	0.165057	0.510513	-0.22691	**100-0.01**
0.665657	0.469704	0.197693	0.020148	0.359626	0.190021	0.17609	0.522435	-0.32576	**100-0.001**
0.438981	0.249344	0.212225	-0.13209	0.509024	-0.05923	0.125492	0.792802	-0.03483	**100-0.0001**
0.625368	0.42494	0.194188	-0.28665	0.130976	0.05177	0.051241	0.504997	0.006009	**1000-0.01**
0.752225	0.550112	0.269647	-0.10015	0.170528	0.172658	0.212566	0.515382	-0.30966	**1000-0.001**
0.649595	0.455538	0.180803	-0.01104	0.363671	0.17913	0.151108	0.504029	-0.27837	**1000-0.0001**
0.624856	0.424436	0.193865	-0.28734	0.130852	0.051316	0.050616	0.504965	0.007282	**2000-0.01**
0.656972	0.456811	0.209948	-0.23242	0.14307	0.086171	0.095208	0.509745	-0.06971	**2000-0.001**
0.552975	0.356234	0.135106	-0.28672	0.197654	0.029492	-0.00776	0.503074	0.104371	**2000-0.0001**

Nota: en los ficheros cvs, el noveno valor es el threshold

Como se puede observar, tanto los pesos como el umbral son aleatorios.

Resumen de Errores de los experimentos

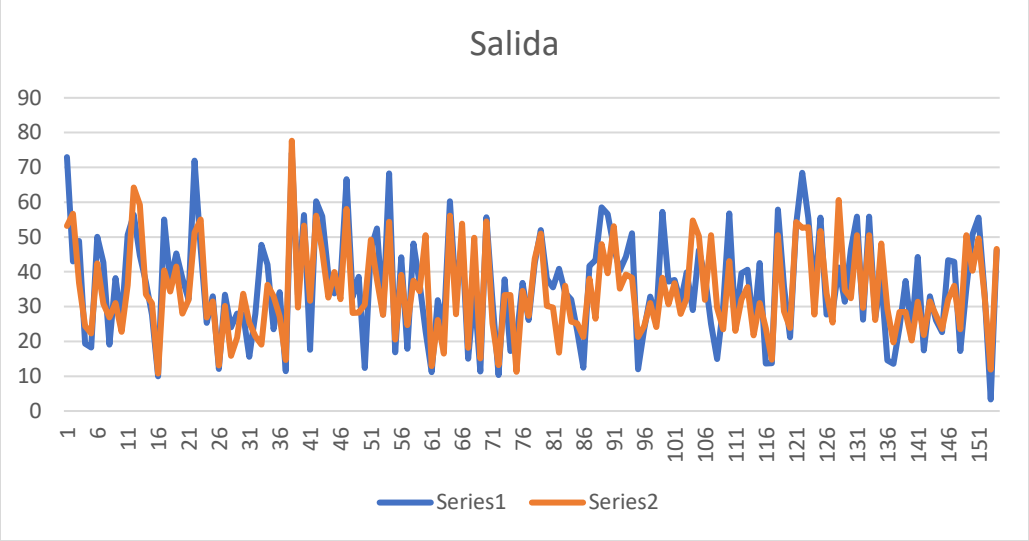
En la tabla se muestran los errores obtenidos en cada modelo final probado:

	100-0.01	100-0.001	100-0.0001	1000-0.01	1000-0.001	1000-0.0001	2000-0.01	2000-0.001	2000-0.0001
Training	0.017099556	0.01884932	0.033524794	0.017252802	0.016459252	0.017819268	0.017257399	0.01656426	0.016650021
Validation	0.017759556	0.018670657	0.033756237	0.017494235	0.017492393	0.017909854	0.017493858	0.017323524	0.017431324

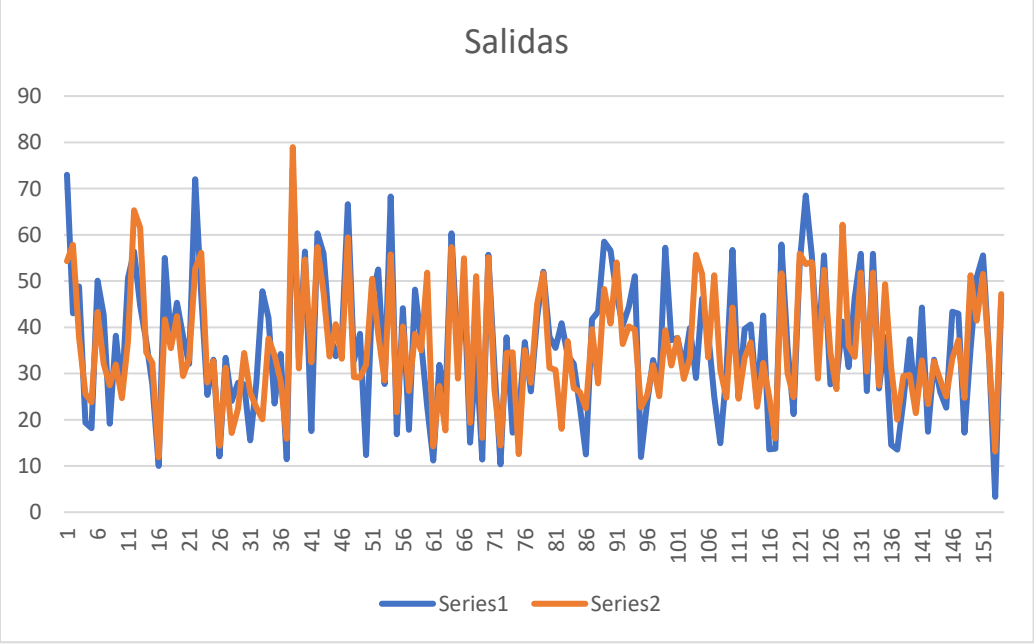
De estos datos sacamos la conclusión de que el modelo con menor error de validación es el de 1000 ciclos con una razón de aprendizaje del 0.001, siendo el error de 0.017492393.

Análisis de la salida

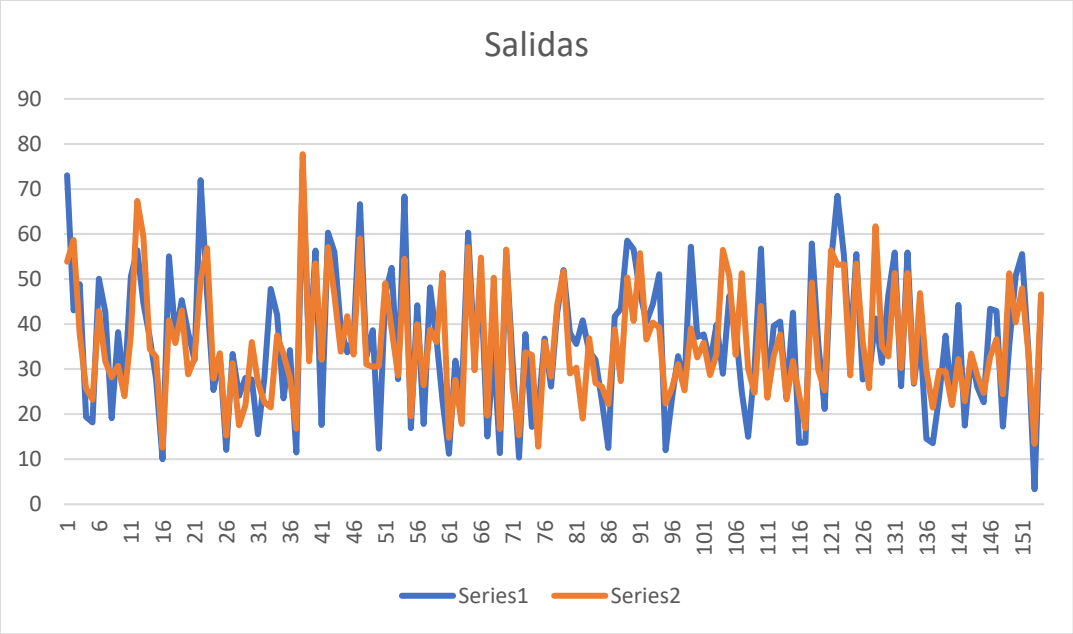
A continuación se mostrarán unas gráficas comparando la salida obtenida con la salida deseada en función de los ciclos (2000) y el factor de entrenamiento, que variará entre 0.01 y 0.0001, donde la línea azul representa la salida esperada y la naranja la obtenida:



1 Ciclos: 2000, Aprendizaje: 0.01



2 Ciclos: 2000, Aprendizaje: 0.001



3 Ciclos: 2000, Aprendizaje: 0.0001

Como se puede ver, aunque la función de error a 2000 ciclos ya esté normalizada, la salida obtenida no llega a ser igual que la deseada en la mayoría de casos, aunque cuanto mayor es el factor de aprendizaje, los resultados varían menos de un resultado a otro.

Parte 2 – Perceptrón Multicapa

Experimentación con Perceptrón Multicapa

Hemos realizado pruebas cambiando tanto el número de Neuronas ocultas, como la razón de aprendizaje, alternando entre 6, 18, 64 y 128 neuronas ocultas y con razones de aprendizaje de 0.1, 0.05, 0.01, y una última prueba extra con 128 neuronas y 0.001 de razón de aprendizaje. Además, hemos escogido realizar las pruebas con 10000 ciclos de ejecución, de forma que de tiempo a que se estabilice el error en todos los casos.

El motivo por el cual solo hemos hecho una prueba adicional con una razón de aprendizaje de 0.001 es porque al ser un valor tan bajo, el aprendizaje es muy lento y lleva mucho tiempo realizar cada prueba y dado que no buscamos realmente un valor óptimo en la red, tampoco creímos necesario ajustar tanto para el análisis.

Para las tablas de las gráficas, la leyenda es la que sigue:
N/Neuronas: Neuronas ocultas
RdA: Razón de Aprendizaje
ETraining: Error de entrenamiento
EValidation: Error de Validación
ETest: Error de Test

Para las gráficas:
Rojo: Error de Validación
Negro: Error de Entrenamiento

Valores en los distintos modelos:

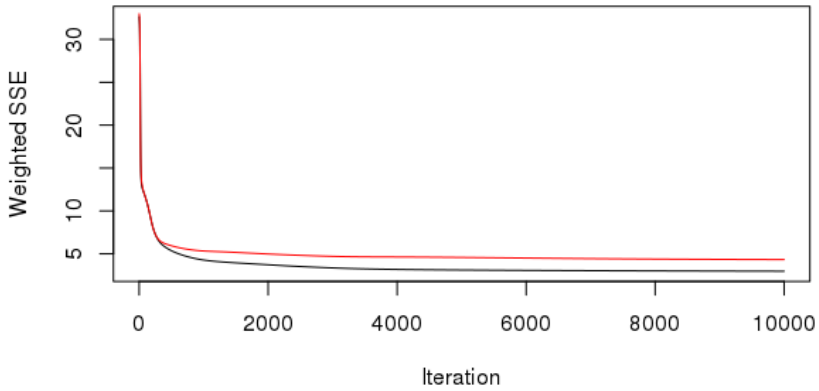
	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13
Neu	6	18	64	128	6	18	64	128	6	18	64	128	128
RdA	0.1	0.1	0.1	0.1	0.01	0.01	0.01	0.01	0.05	0.05	0.05	0.05	0.001

Gráficas de las pruebas

Hemos escogido los modelos M1, M2, M6 y M13 debido a que son los resultados más representativos de los experimentos, dado que se repiten sus patrones en el resto de pruebas.

M1

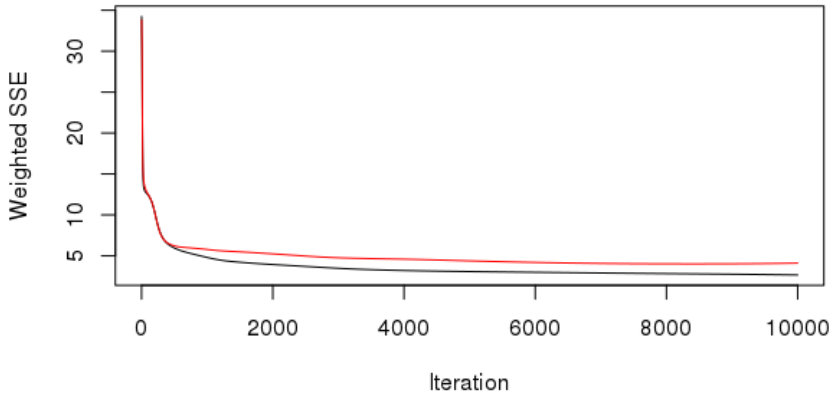
Neuronas	RdA	ETraining	EValidation	ETest
6	0.1	0.00405789438957465	0.0060037991796675	0.0048107133412337



Podemos comprobar por la gráfica, que ambos errores se estabilizan a partir de los 4000 ciclos, pero no llega a producirse ninguna animalia o sobreajuste.

M2

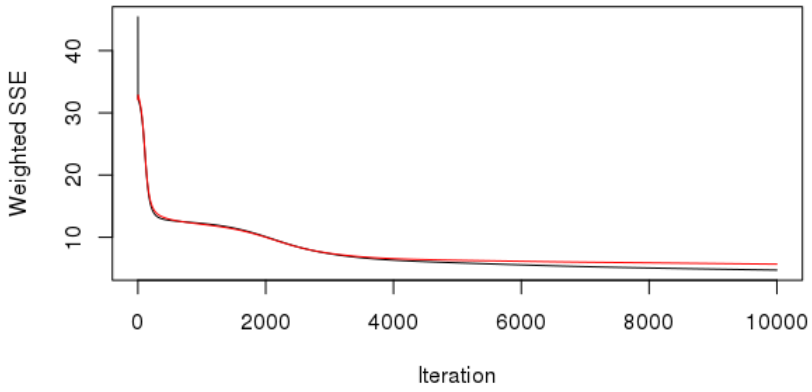
Neuronas	RdA	ETraining	EValidation	ETest
18	0.1	0.00362463448326727	0.00569446695546777	0.00458645101026672



Podemos comprobar por la gráfica, que ambos errores se estabilizan a partir de los 7000 ciclos, con un ligero sobreajuste a partir de los 9000 (el error de validación empieza a subir, mientras que el de entrenamiento vuelve a bajar.

M6

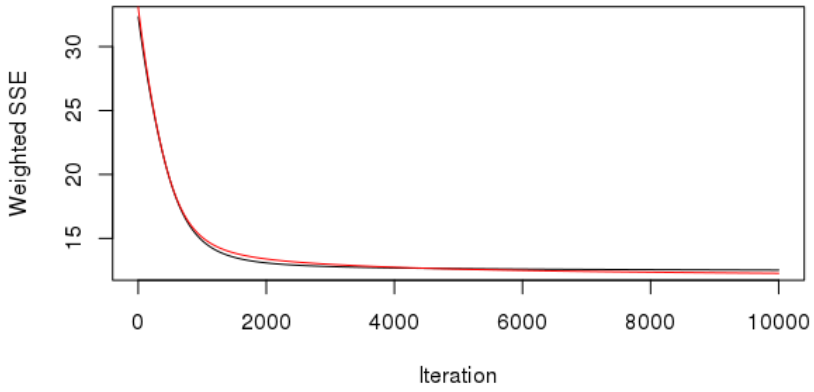
Neuronas	RdA	ETraining	EValidation	ETest
128	0.01	0.00652809568652377	0.00787082864670585	0.00609868956011866



Podemos comprobar por la gráfica, que parece que el error de validación se estabiliza, pero el de entrenamiento sigue bajando, lo cual nos indica que tal vez para un mejor resultado serían necesarios alrededor de 2000 ciclos más.

M13

Neuronas	RdA	ETraining	EValidation	ETest
128	0.001	0.00652809568652377	0.00787082864670585	0.00609868956011866



Podemos comprobar por la gráfica, que ambos errores no están estabilizados y que es probable que necesitasen más ciclos debido a su baja tasa de aprendizaje. Estimamos que a lo mejor serían necesarios alrededor de 5000 ciclos extra, basándonos en los resultados de los anteriores experimentos, puesto que tiene pinta de ser un valle antes de una bajada.

Resumen de errores de los experimentos

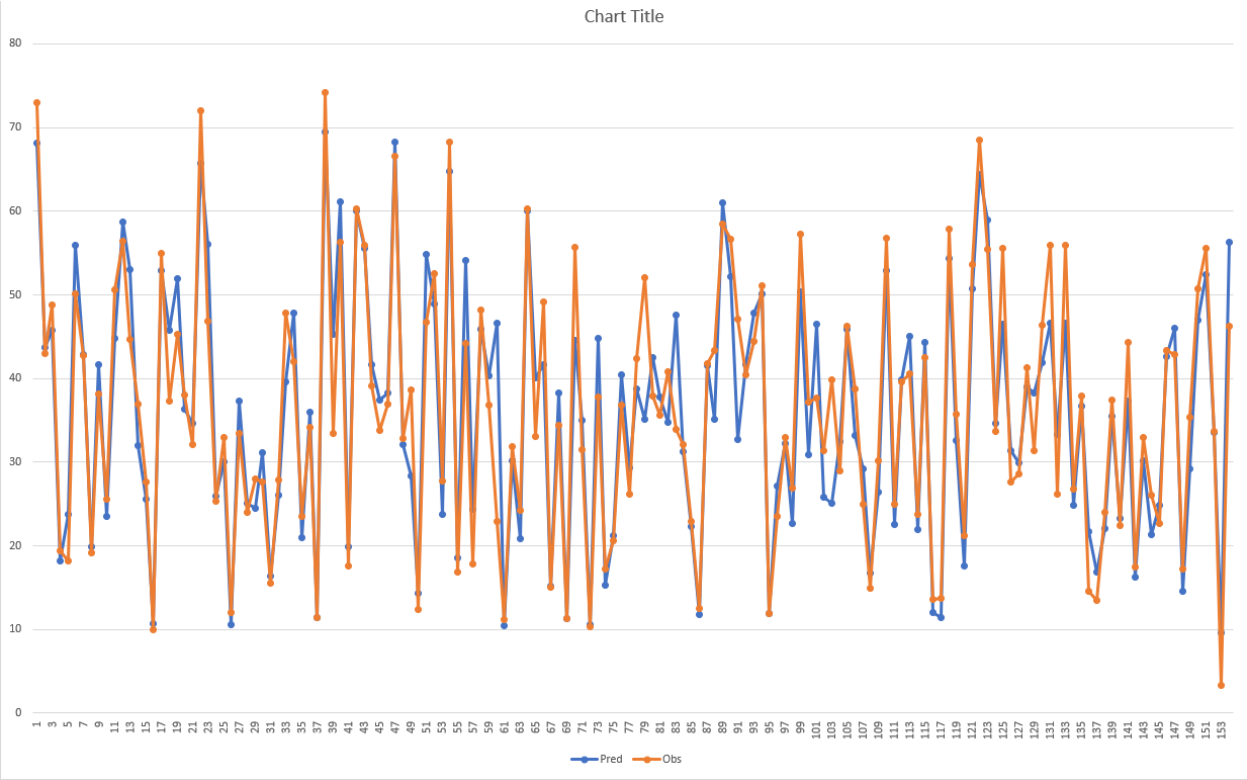
	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13
Neu	6	18	64	128	6	18	64	128	6	18	64	128	128
RdA	0.1	0.1	0.1	0.1	0.01	0.01	0.01	0.01	0.05	0.05	0.05	0.05	0.001

	ETraining	EValidation	ETest
M1	0.00405789438957465	0.0060037991796675	0.0048107133412337
M2	0.00362463448326727	0.00569446695546777	0.00458645101026672
M3	0.00339777451837281	0.00563732940989827	0.00457679849899311
M4	0.00290889839449745	0.00459561246335562	0.00450557916459781
M5	0.00583429097612119	0.00746403394530759	0.00609383535049173
M6	0.00652809568652377	0.00787082864670585	0.00609868956011866
M7	0.00654349114181189	0.00778955927957137	0.00609424194834831
M8	0.00686644202376916	0.00805662747152085	0.0062866941616176
M9	0.00422167040376538	0.00640161857654105	0.0049189463477558
M10	0.0042334348345523	0.00611390085573185	0.00474915893388222
M11	0.0041183171005792	0.0063856111984702	0.00502963303196256
M12	0.00407279617861416	0.00633267043850096	0.00498975995386294
M13	0.00652809568652377	0.00787082864670585	0.00609868956011866

De esta tabla sacamos que el modelo con el menor error de validación es el modelo M4 con 128 neuronas y una razón de aprendizaje de 0.1

Análisis de la salida

Grafica con los valores des-normalizados (Azul: Predicho por la red, Naranja: Observado)



Podemos observar que, en la mayoría de las ocasiones, el error provoca que haya una diferencia de hasta 5 MPa, aunque por supuesto hay ocasiones donde la diferencia es de tan solo 1 MPa.

Esto nos indica que los resultados deberían de ser evaluados considerando un error en la medida equivalente al caso de mayor diferencia, que aproximadamente es de 10 MPa.

Conclusiones

Tras haber realizado todos los experimentos con adaline y el perceptrón multicapa, hemos llegado a las siguientes observaciones:

- El aprendizaje de adaline es un método sencillo de implementar en casi cualquier lenguaje de programación.
- Las predicciones con adaline no son tan precisas como las del perceptrón multicapa, aunque sí que alcanza un error estable más deprisa que el perceptrón (dependiendo además de la razón de aprendizaje).
- En el perceptrón multicapa se necesitan por lo general más ciclos que en el adaline, dado que su aprendizaje es mucho más lento sobre todo cuantas más neuronas ocultas hay.
- En el perceptrón multicapa una mayor cantidad de neuronas no implica mejores resultados necesariamente, sobre todo por el alto coste que implica hacer cada ciclo, pero sí que mejora el resultado con la razón de aprendizaje adecuado.