# REDES DE NEURONAS ARTIFICIALES


## Práctica 1

**Repositorio de GitHub:**
**https://github.com/Cocytus72/Redes-de-Neuronas-P1**

**05/11/2023**

---

Lucas Gallego Bravo - 100429005
E-mail: 100429005@alumnos.uc3m.es

Prof. Ángel Carrasco Fernández

# Índice:

# 1. Introducción

Para esta primera práctica de la asignatura, se nos ha pedido abordar un problema de regresión haciendo uso de dos modelos de neuronas supervisados, el Adaline y el Perceptrón multicapa.

Para esta tarea se nos ha proporcionado un conjunto de datos de regresión denominado “Gas Turbine CO and NOx Emission Data Set”, el cual contiene los datos de las medidas realizadas a una turbina de gas a lo largo de 4 años.

Dado este dataset, se nos pide utilizar los modelos de redes de neuronas para predecir una de las variables de este dataset, el “TEY”, o mejor dicho, “Turbine energy yield”. Una vez se tiene el dataset, lo primero que se hace es realizar un preproceso de los datos para mejorar el rendimiento y aprendizaje de nuestros modelos.

Con el preproceso de los datos finalizado, es momento de utilizarlos para entrenar a nuestros modelos y realizar la predicción. El primer modelo de redes de neuronas que se utiliza es el Adaline, el cual se ha tenido que construir desde cero, para posteriormente utilizar los mismos datos en el Perceptrón multicapa, pudiendo así comparar ambos modelos.

Cabe destacar que ambos modelos se encuentran dentro del mismo notebook.

# 2. Preproceso de los datos

El preproceso de los datos realizado se puede dividir en cuatro partes, la primera de estas partes se corresponde a la apertura y normalización de todos los datos. Para esta normalización, se utiliza un bucle el cual va ejecutarse para todas las variables de nuestro dataset. Primero, fuera del bucle, se obtienen todos los máximos y mínimos de nuestras variables, este paso es crucial para la normalización de nuestros datos, ya que sin ellos, no se podría normalizar ninguno de los datos, una vez estamos en el bucle, se calcula el valor normalizado para cada valor de las variables (para cada fila de cada columna), realizando los cálculos necesarios, una vez hemos obtenido el valor normalizado, este se guardará en su posición correspondiente dentro del dataset.

El siguiente paso de nuestro preproceso es más sencillo, pues es únicamente realizar la aleatorización de nuestro dataset, lo cual se puede realizar en una única línea de código. En este paso es crucial añadir un random state, o una random seed, ya que aunque estemos aleatorizando los datos, se quiere que los resultados sean reproducibles, es decir, que de siempre los mismos resultados independientemente de cuándo o dónde se ejecute.

El tercer paso consiste en dividir el dataset en tres subconjuntos (train, validación y test), el primer subconjunto, el de train, será un 70% del dataset original, mientras que el 30% restante, será dividido equitativamente entre el subconjunto de validación y el de test.

Finalmente, los tres subconjuntos serán guardados en ficheros separados.

# 3. Adaline

Como ya se ha mencionado en la introducción anterior, este es el primer modelo de red de neuronas que se ha utilizado para la predicción de “TEY” en nuestro dataset. Este modelo Adaline se ha programado desde cero, obviando algunas funciones de la clase como la activación, ya que esta devuelve el mismo valor que se le manda, por lo que al haberlas considerado redundantes, se ha preferido no implementarla. El Adaline propuesto se puede dividir en 8 pasos.

Primero se inicializan los pesos y umbral de manera aleatoria (usando siempre random state y random seed). El segundo paso es presentar un patrón de entrada, para posteriormente calcular la salida de la red de neuronas. Gracias a ese cálculo, se pueden actualizar los pesos de la red así como el umbral haciendo uso de la regla delta. Una vez se han ejecutado los pasos anteriores, para todas las filas de X, se calcula el MSE de nuestro subconjunto de entrenamiento y validación, haciendo uso de la función ‘calculate_error’ implementada en el Adaline para facilitar estos cálculos. Estos errores se guardan en una lista dentro del Adaline para posteriormente poder hacer gráficas.

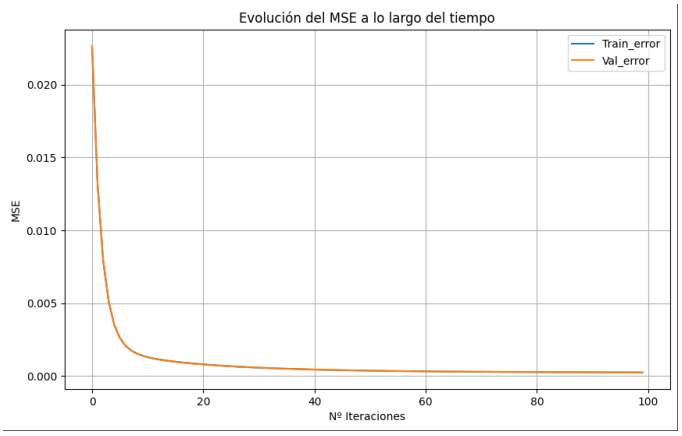
Si el error de validación obtenido es mejor que el que se tenía anteriormente, se guarda la información del modelo en ese punto.

Finalmente se imprimen en pantalla los valores de los errores de entrenamiento y validación, al igual que el número de la iteración en la que se encuentren. Estos pasos se repetirán tantas veces como iteraciones se indiquen en los hiper parámetros.

## 3.1 Experimentación realizada y evolución de los errores:

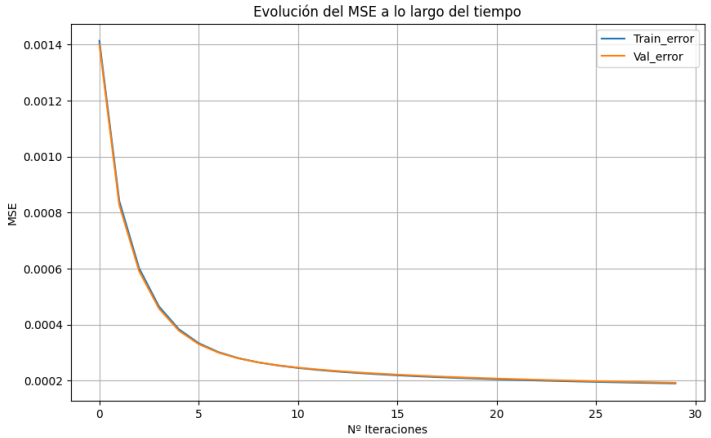
Para seleccionar el mejor modelo para nuestro Adaline con el el dataset, debemos primero de experimentar con los hiper parámetros cambiandolos y viendo como evolucionan los errores en los diferentes subconjuntos. Lo primero que se va a mostrar en este apartado es la evolución de los errores de entrenamiento y validación a lo largo de la etapa de aprendizaje de nuestro modelo Adaline. Únicamente se mostrará en forma de grafo, la evolución de aquellos modelos que se consideren más significativos,

La siguiente gráfica muestra la evolución del error en el primer experimento que se realizó:

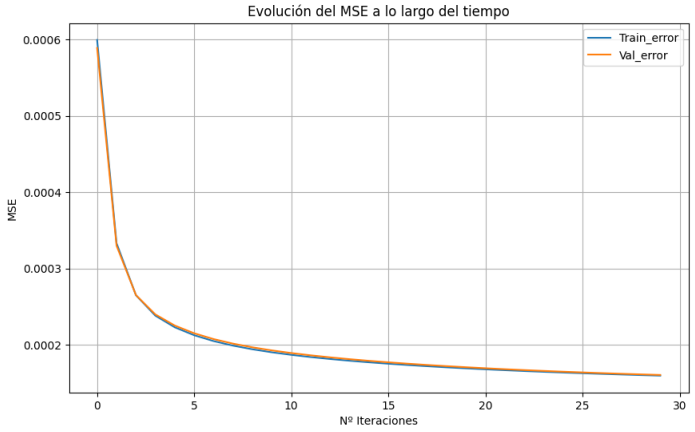


Como se puede observar, al ser tantas iteraciones, y la diferencia tan mínima, la diferencia entre train y validación es casi indistinguible.

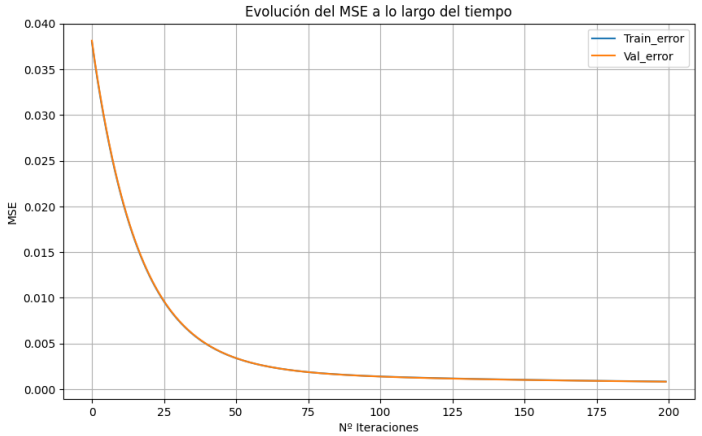
El siguiente experimento que se muestra fue aumentando la tasa de aprendizaje, pero al mismo tiempo disminuyendo el número de iteraciones a 30:



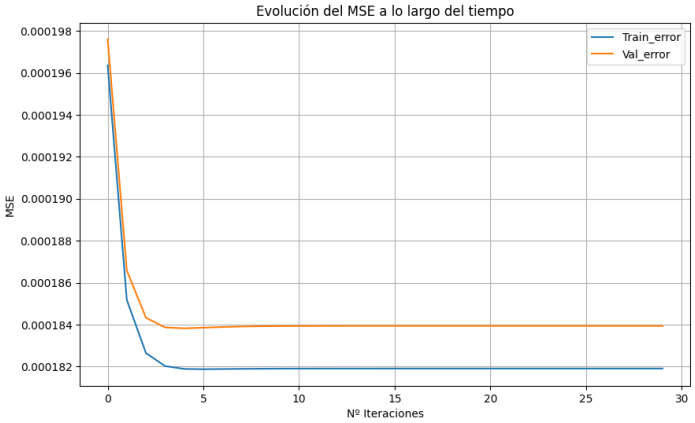

En el siguiente gráfico, se ha utilizado una razón de aprendizaje mayor (0.003):




Con el siguiente se quería ver como una menor tasa de aprendizaje afecta al modelo:




El último experimento que se quiere mostrar es el contrario al anterior, aumentar la tasa de aprendizaje:




Para todos los experimentos con la tasa de aprendizaje = 0.1, a partir de la iteración Nº 4 devuelve siempre el mismo resultado.

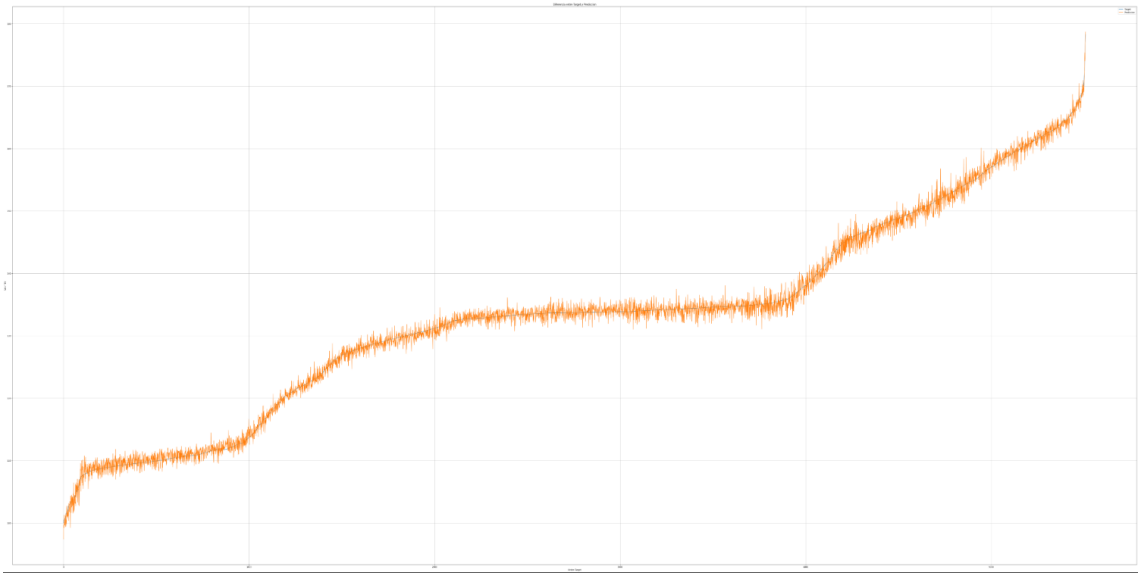
## 3.2 Todos los experimentos realizados:

A continuación se mostrará una tabla con todos los experimentos realizados en el Adaline, mostrando que hiper parámetros se han utilizado, al igual que los errores de entrenamiento, validación y test, así como el tiempo de ejecución del aprendizaje (fit) de cada modelo. La fila de color verde es el se ha considerado como mejor modelo.

| Tasa de aprendizaje | Nº Iteraciones | MSE Train | MSE Validación | MSE Test | Tiempo (Segundos) |
|---|---|---|---|---|---|
| 0.0001 | 100 | 0.000254 | 0.000255 | 0.000256 | 19.245 |
| 0.001 | 30 | 0.000190 | 0.000193 | 0.000189 | 6.475 |
| 0.001 | 100 | 0.000158 | 0.000158 | 0.000153 | 21.765 |
| 0.003 | 100 | 0.000145 | 0.000145 | 0.000141 | 19.376 |
| 0.003 | 30 | 0.000160 | 0.000161 | 0.000156 | 5.240 |
| 0.00001 | 200 | 0.000844 | 0.000829 | 0.000870 | 39.404 |
| 0.00001 | 500 | 0.000384 | 0.000379 | 0.000391 | 98.189 |
| 0.1 | 30 | 0.000182 | 0.000184 | 0.000177 | 6.942 |
| 0.1 | 300 | 0.000182 | 0.000184 | 0.000177 | 59.637 |
| 0.3 | 30 | 0.000382 | 0.000386 | 0.000343 | 5.876 |

Este modelo se ha seleccionado ya que a pesar de que tiene un MSE un pelin mayor comparado con el modelo (0.003,100), no es una diferencia tan grande, y al tardar menos tiempo se ha considerado como mejor modelo.

## 3.3 Grafo de Predicción y Target en el subconjunto test:



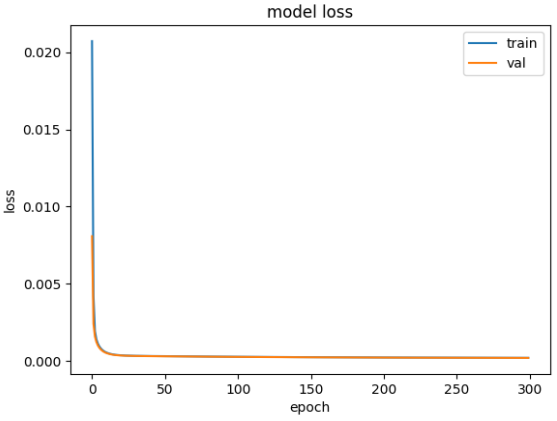
Si se hace zoom, se puede apreciar mejor la línea que sigue Target, y como la predicción trata de imitarla. Este grafo se ha realizado utilizando el mejor modelo.

# 4. Perceptrón Multicapa:

Para esta segunda parte de esta práctica, no hemos tenido que realizar ningún código desde cero como en el caso del Adaline. Si no que se nos ha proporcionado un código con los diferentes modelos disponibles para que se experimente directamente con ellos.Como ya se ha mencionado anteriormente, se ha decidido realizar esta segunda parte del laboratorio dentro del mismo documento de Google Collab para tener un trabajo más limpio, y no tener que utilizar funciones como el preproceso de los datos, ya que de ese preproceso se obtienen los valores máximos y mínimos que se utilizan para la desnormalización de los datos.

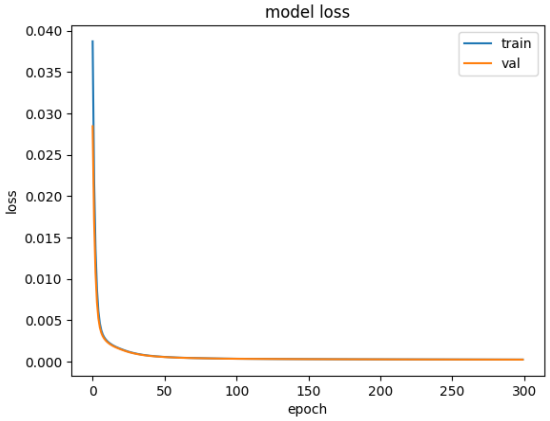
## 4.1 Experimentación realizada y evolución de los errores:

Como hemos hecho anteriormente con el Adaline, el primero modelo del que se va a mostrar gráfica es del primer modelo realizado, el cual venía de por sí dado por los profesores:

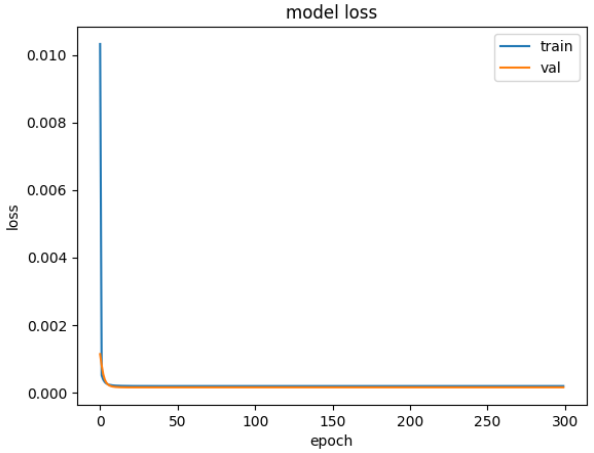


Esta gráfica corresponde al modelo sigmoid con lr=0.2, 300 iteraciones, y 20 neuronas.

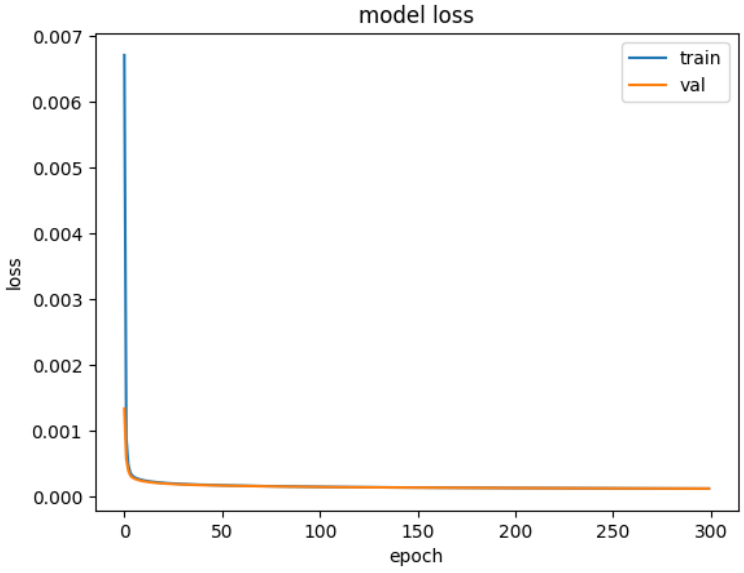
La siguiente gráfica se corresponde al modelo relu con lr = 0.01, 300 iteraciones y 60 neuronas:




El siguiente gráfico es de un modelo que ha experimentado un pequeño sobreaprendizaje y después mantuvo el mismo error (fila roja en la siguiente sección):




Finalmente, el grafo del mejor modelo (linea verde en la siguiente sección):

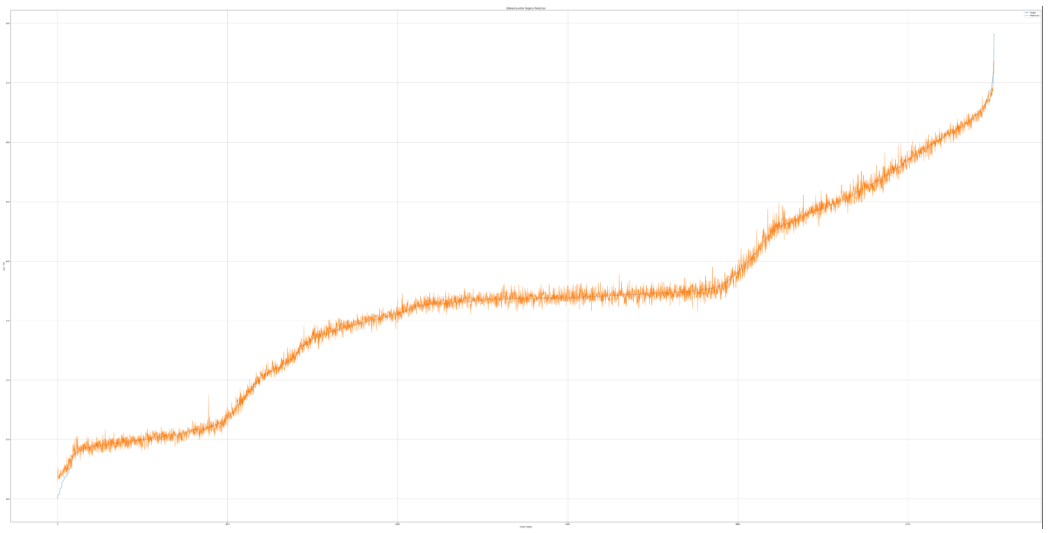


## 4.2 Todos los experimentos realizados:

Para los modelos lineales no se pone Nº de Neuronas ya que no es un hiper parámetro del modelo. La fila verde indica el mejor modelo, en rojo se marca el modelo que ha hecho sobreaprendizaje:

| Modelo Utilizado | Tasa de aprendizaje | Nº Iteraciones | Nº Neuronas | MSE Train | MSE Validación | MSE Test |
|---|---|---|---|---|---|---|
| sigmoid | 0.2 | 300 | 20 | 2.1380e-04 | 2.0704e-04 | 2.1636e-04 |
| relu | 0.2 | 300 | 20 | 1.5951e-04 | 1.5771e-04 | 1.5924e-04 |
| lineal | 0.2 | 300 | - | 1.6477e-04 | 1.4417e-04 | 1.4027e-04 |
| sigmoid | 0.01 | 300 | 60 | 4.3532e-04 | 4.2587e-04 | 4.3532e-04 |
| relu | 0.01 | 300 | 60 | 2.3058e-04 | 2.2535e-04 | 2.3663e-04 |
| lineal | 0.01 | 300 | - | 1.4497e-04 | 1.4558e-04 | 1.4135e-04 |
| sigmoid | 0.2 | 300 | 100 | 2.5880e-04 | 2.5026e-04 | 2.6050e-04 |
| relu | 0.2 | 300 | 100 | 1.2286e-04 | 1.2324e-04 | 1.2315e-04 |
| lineal | 0.22 | 200 | - | 1.8867e-04 | 1.5107e-04 | 1.4839e-04 |
| sigmoid | 0.03 | 100 | 250 | 4.2621e-04 | 4.1330e-04 | 4.2700e-04 |
| relu | 0.03 | 100 | 250 | 2.2871e-04 | 2.2310e-04 | 2.3345e-04 |
| lineal | 0.03 | 100 | - | 1.4434e-04 | 1.4433e-04 | 1.4048e-04 |

## 4.3 Grafo de Predicción y Target en el subconjunto test:



Si se hace un poco de zoom se puede ver como al principio target y pred están más separadas, pero luego se unen.

# 5. Comparación del Adaline y PM:

La mayor diferencia que se puede ver tras utilizar ambos modelos es la velocidad. El modelo Adaline se ejecuta de manera mucho más rápida que cualquiera de los modelos de PM.

Otra diferencia se encuentra en la posibilidad de cambiar el número de neuronas de los modelos relu y sigmoide del Perceptrón multicapa, lo cual hace que existan más posibilidades y combinaciones de encontrar un mejor modelo que los obtenidos con el Adaline.

Otra de las diferencias que se puede observar es que el modelo Adaline necesita menos ciclos en reducir el error, mientras que los modelos PM han necesitado de unas 300 iteraciones en llegar errores de magnitudes similares, pero por otro lado, mientras que el Adaline puede quedarse estancado en dichos errores y no aprender más, los modelos de PM han demostrado seguir aprendiendo, bajando poco a poco en cada iteración el MSE.

# 6. Conclusión de la práctica:

A modo de conclusión me gustaría decir que al haber realizado la práctica yo solo, me ha resultado bastante más complicado, ya que al tener que organizarse con otras asignaturas y con problemas personales no había nadie más que pudiera ayudarme si yo no podía.

Sin embargo, al haber realizado yo mismo la entrega entera, considero que he conseguido aprender bastante sobre estos dos modelos de redes de neuronas artificiales. El Adaline en un principio me resultó muy complejo de programar debido a que no conseguía actualizar los pesos de manera correcta, por lo que mi modelo no aprendía en lo absoluto, no fue tras haber indagado un poco más en internet y en los apuntes de Adaline proporcionados que conseguí implementar el modelo que tengo actualmente.

Por otra parte, la segunda parte de la práctica me ha parecido menos entretenida, ya que al tardar tanto los modelos, no puedes hacer otra cosa salvo esperar a que se termine la ejecución rezando por obtener un mejor resultado que con modelos anteriores. Sin embargo, que el profesorado haya dado el código de esta parte me ha sido de gran ayuda, ya que viendo los problemas que me supuso el Adaline, no creo que hubiera sido capaz de terminar esta entrega.