Universidad
Carlos III de Madrid

Grado en Ingeniería Informática

Curso 2020/2021

Redes de Neuronas Artificiales

Problema de Regresión

Predicción del precio medio de la vivienda en California

Autores:

Alba Reinders Sánchez	*100383444*
Alejandro Valverde Mahou	*100383383*

Índice

1. Introducción

El problema a resolver es la **predicción del precio medio de la vivenda en California**, usando dos modelos diferentes: *ADALINE* y *Perceptrón Multicapa*.

El *ADALINE* es un modelo **lineal**, mientras que el *Perceptrón Multicapa* es un modelo **no lineal**. El objetivo de esta práctica es realizar una comparativa entre estos dos modelos mediante la experimentación y análisis de los resultados, para averiguar cuál de los dos es capaz de encontrar la solución más cercana a la solución óptima.

2. Preparación de Datos

Los datos proporcionados son: *longitude*, *latitude*, *housingMedianAge*, *totalRooms*, *totalBedrooms*, *population*, *households*, *medianIncome*, ***medianHouseValue***.

La salida de los modelos deberá ser ***medianHouseValue*** en función del resto de atributos.

El conjunto de ejemplos proporcionados es de **17000**.

2.1. Normalización

El primer paso en el preprocesado de los datos es la **normalización**. Esta técnica consiste en acotar todos los datos en un rango de 0 a 1. Se ha decidido para este problema, normalizar exclusivamente los atributos de entrada, y no la salida, porque, experimentalmente, resulta en menos error.

La normalización de los datos se realiza cuando los distintos atributos de entrada no están en la misma escala, ya que tener atributos con escalas muy diferentes puede dar lugar al cálculo erróneo de los pesos, lo que deriva en modelos ineficaces.

La transformación lineal que se aplica a cada atributo es:

$$atr'_i = \frac{atr_i - \text{mín}\,(atr)}{\text{máx}\,(atr) - \text{mín}\,(atr)}$$

2.2. Aleatorización

Para evitar un entrenamiento inadecuado, es necesario otorgar a los modelos una lista de datos desordenados, o con orden aleatorio. De esta forma el modelo no se ajusta a un rango concreto de valores, que podrían darse seguidos si no se organizan aleatoriamente.

2.3. Separación en conjuntos de datos

Dado que este problema tiene una cantidad suficientemente grande de datos, se puede realizar la división del conjunto de datos en 3 subconjuntos:

- **Conjunto de Entrenamiento**:

  Con él se realiza el aprendizaje (ajuste de pesos) del modelo. Es el conjunto más grande, pues tiene el 60 % de los datos (10200 instancias).

- **Conjunto de Test**:

  Se usa para evaluar la precisión y capacidad de generalización del modelo. Este conjunto tiene el 20 % de los datos (3400 instancias).

- **Conjunto de Validación**:

  Se usa para determinar los mejores hiperparámetros del modelo. Este conjunto tiene el 20 % de los datos (3400 instancias).

3. Adaline

El lenguaje de programación elegido para el desarrollo del algoritmo **ADALINE** ha sido *Python*.

3.1. Experimentación

Se ha decidido realizar distintos experimentos, tanto con salida normalizada como con salida no normalizada, con el objetivo de comprobar cuál ofrece mejores resultados. Además, para cada experimento, se han realizado varias pruebas para encontrar la razón o tasa de aprendizaje más adecuada en cada caso.

En lugar de elegir arbitrariamente un número de ciclos para cada experimento, se ha usado un criterio de parada más específico: el aprendizaje termina cuando el error en el conjunto de validación es mayor o igual que en los 4 ciclos anteriores.

Este criterio de parada es eficaz dado que ayuda a determinar automáticamente el momento en el que el algoritmo converge en un valor concreto, o comienza a tener sobreaprendizaje.

Los experimentos consistirán en ejecutar el algoritmo con una tasa de aprendizaje inicial de *0.5*, que se irá ajustando a lo largo de los experimentos hasta alcanzar la tasa que obtenga los resultados más adecuados.

3.2. Resultados Obtenidos

Salida Normalizada

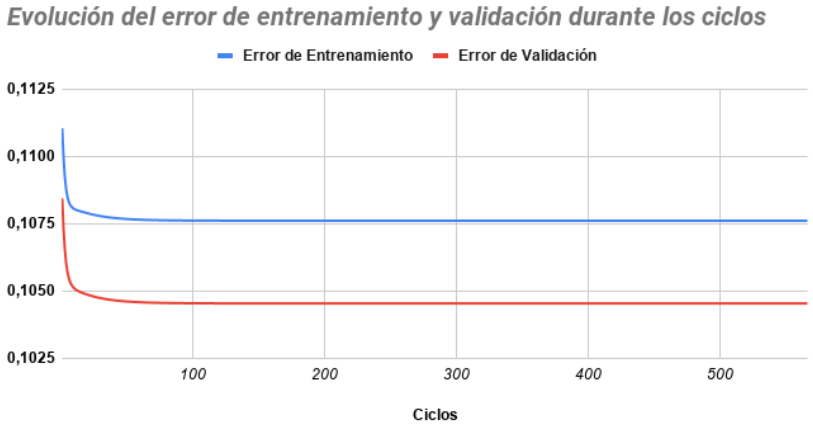
Tabla de resultados obtenidos por experimento:

	Experimento 1	**Experimento 2**	**Experimento 3**	**Experimento 4**
Tasa aprend.	0.5	0.3	0.2	0.1
Ciclos	5	5	308	566
Err. entren.	0.1537454242037602	0.12999495781099646	0.11828832839640416	0.10761552626685748
Err. valid.	0.1510858340453369	0.1273892127623424	0.11546352829888222	0.10454524622576246
Err. test	0.1493666202359257	0.12539101364386532	0.11387419742858149	0.10391975176932682

Los experimentos que solo llegan a 5 ciclos es debido a que la tasa de aprendizaje es demasiado grande, y hace que el error crezca en lugar de decrecer.

Experimentalemente, la inicialización aleatoria de los pesos produce que el número de ciclos varíe de enorme manera de una ejecución a otra, usando los mismos hiperparámetros. Los valores que figuran en la tabla son aquellos que se han obtenido con mayor frecuencia a la hora de realizar los experimentos.

Según se ha ido disminuyendo el valor de la tasa de aprendizaje, se han encontrado mejores resultados, hasta llegar a un valor de **0.1**(Experimento 4). Cualquier valor más pequeño que ese, hace que el algoritmo no acabe en un número de ciclos razonable. La siguiente gráfica muestra la evolución de errores del **Experimento 4**.

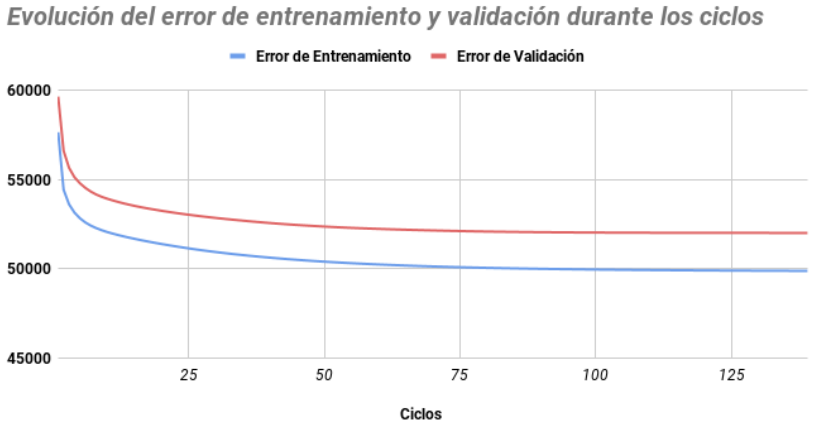


Salida No Normalizada

Tabla de resultados obtenidos por experimento:

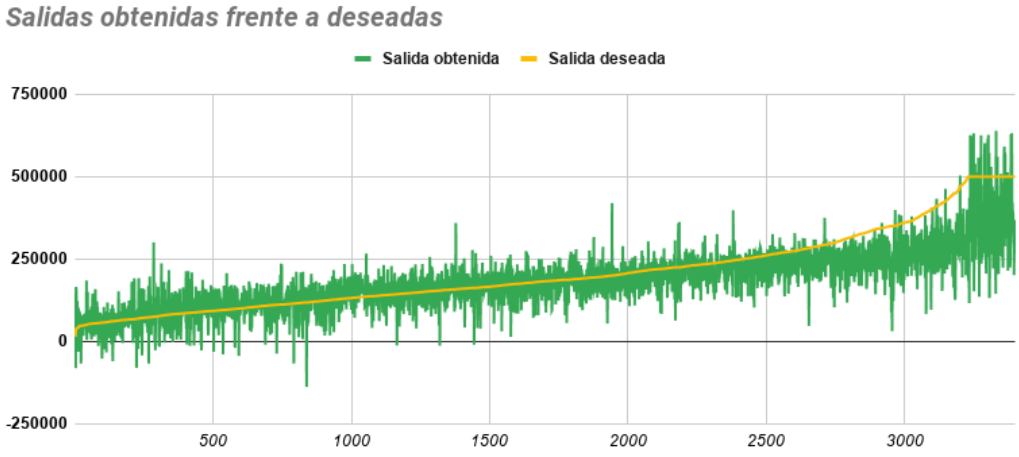
	Experimento 1	Experimento 2	Experimento 3	Experimento 4
Tasa aprend.	0.5	0.22	0.05	0.01
Ciclos	5	8	30	139
Err. entren.	69245.64846716617	55508.988233795564	50017.42808457685	49891.46770073132
Err. valid.	69642.07395718427	56733.54239803611	51961.06532331261	52020.971503368535
Err. test	69986.82287304835	55858.881817193804	50276.791119059315	50082.97953387804

Igual que en el caso anterior, se ha ido disminuyendo la tasa de aprendizaje, y en este caso se ha alcanzado el valor de **0.01**. A partir de ese valor, el algoritmo no converge en un número razonable de ciclos.



3.3. Análisis

Salida Normalizada

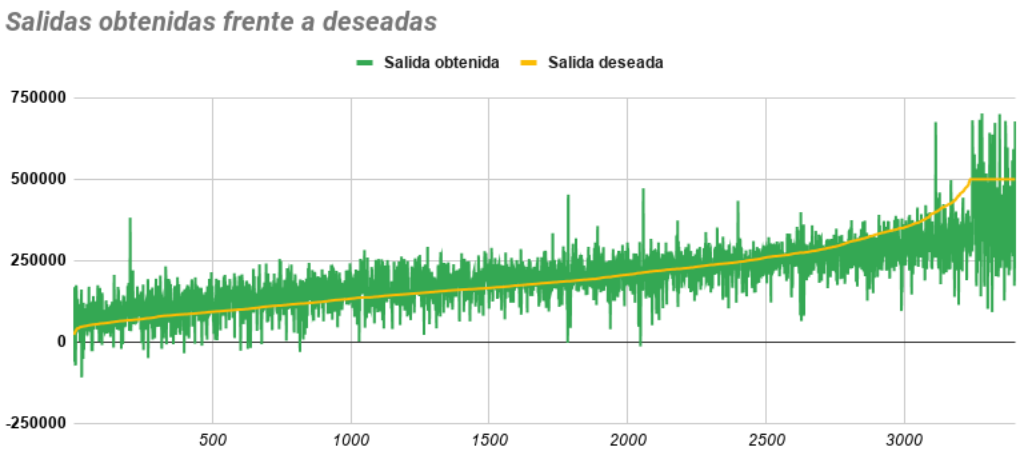


Como se aprecia en la gráfica anterior, la salida obtenida se acerca a la salida deseada, excepto en los valores más altos(a partir de $250000), donde se produce una mayor disparidad. Esto puede deberse a que, los valores que van desde los $0 hasta los $250000, siguen una distribución relativamente lineal, pero a partir de ese valor parece que aumenta de forma exponencial.

Dado que el algoritmo **ADALINE** ofrece una salida lineal, es capaz de adecuarse con mayor facilidad que a la segunda parte de los datos.

El valor absoluto medio del error que el modelo produce para el conjunto de test es de **$65400.287447627044**.

Salida No Normalizada



Al igual que en el caso del experimento con salida normalizada, se ajusta mejor con los datos de menor valor, pero en este caso sacrifica un poco de precision en los valores de $0 hasta $200000 aproximadamente. Sin embargo, hace que el error que produce en los valores altos sea menor.

Gracias a este ajuste, consigue un error medio menor en el conjunto de entrenamiento: **$50082.97953387804**.

4. Perceptrón Multicapa

El **Perceptrón Multicapa** se ha ejecutado usando el *script* proporcionado, que está en el lenguaje de programación *R*.

4.1. Experimentación

En este caso, es necesario trabajar con la salida normalizada porque, sino, se produce el efecto de saturación en las neuronas de la red, lo que hace que no sean capaces de aprender.

Por otro lado, la condición de parada en este caso es diferente. Consiste en evaluar el modelo con un número fijo de ciclos y decidir en qué ciclo se ha obtenido mejor resultado, tras ello se vuelve a entrenar a la red usando el número de ciclos óptimo.

Los parámetro configurables de este algoritmo son la **topología**, la **razón de aprendizaje** y el número de **ciclos máximos**. Se ha decidido fijar el número de ciclos máximos para todos los experimentos a **10000**.

Los experimentos se han llevado a cabo teniendo en cuenta el resultado de experimentos anteriores, para intentar encontrar mejorers resultados con cada uno. A su vez, cada experimento está compuesto de distintos *subexperimentos*, donde se intenta encontar la mejor razón de aprendizaje para cada uno.

Los experimentos que se han llevado a cabo son:

- **Experimento 1**: la topología de la red consiste en no tener **ninguna capa oculta**, por lo que se podría considerar que no llega a ser un *Perceptrón Multicapa* al no poseer capas ocultas, pero sirve de caso base para comparar el resto de experimentos.

- **Experimento 2**: tras el experimento anterior se decide configurar un modelo que posee **una capa oculta** con **20 neuronas** para buscar la mejora de resultados al añadir una capa oculta.
- **Experimento 3**: como al añadir una capa oculta se comprueba que mejora bastante, se prueba a aumentar el número de neuronas, por lo que este experimento tiene también **una capa oculta** pero esta vez tiene **30 neuronas**.
- **Experimento 4**: el último modelo consiste en crear una topología de red con **dos capas ocultas** cada una con **30 y 20 neuronas** respectivamente con la que se pretende mejorar aún más la precisión del modelo.

4.2. Resultados Obtenidos

Tabla de resultados obtenidos por experimento:

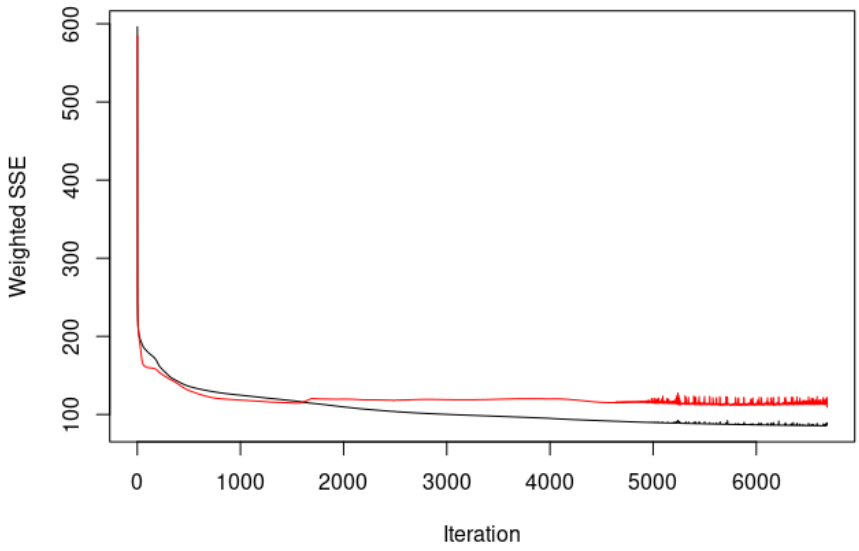
	Topología	***Tasa aprend.***	***Ciclos ópt.***	***Err. entren.***	***Err. valid.***	***Err. test***
Experimento 1-1	c( )	0.01	9998	0.09985901	0.09648482	0.09615899
Experimento 1-2	c( )	0.05	2835	0.09971082	0.09638341	0.09597574
Experimento 1-3	c( )	0.03	6553	0.09981516	0.09646387	0.09609764
Experimento 1-4	c( )	0.04	3368	0.09975728	0.09641738	0.09603159
Experimento 2-1	c(20)	0.01	10000	0.08838755	0.08637155	0.084649
Experimento 2-2	c(20)	0.07	10000	0.07790314	0.07747784	0.07685197
Experimento 2-3	c(20)	0.15	10000	0.07818915	0.07721448	0.07657498
Experimento 2-4	c(20)	0.4	6096	0.076175	0.0759147	0.07687894
Experimento 3-1	c(30)	0.1	10000	0.07710104	0.07670325	0.07649249
Experimento 3-2	c(30)	0.4	10000	0.07562609	0.07617276	0.0765667
Experimento 3-3	c(30)	0.7	10000	0.07562471	0.07674079	0.07711822
Experimento 3-4	c(30)	0.9	6828	0.07747128	0.07813840	0.07780503
Experimento 4-1	c(30,20)	0.1	9880	0.06917029	0.07142923	0.07368347
Experimento 4-2	c(30,20)	0.3	10000	0.0644995	0.07030901	0.07212733
Experimento 4-3	c(30,20)	0.7	6687	0.06347896	0.06924343	0.07099247
Experimento 4-4	c(30,20)	0.6	2525	0.06820952	0.07108485	0.07261085

Los experimentos que tienen un total de 10000 ciclos como número de ciclos óptimos se deben a que la red no consigue converger y al llegar a este número se considera que debe detenerse la ejecución para evitar que cada experimento dure demasiado tiempo.

En el caso de que uno de estos experimetos, que no han tenido tiempo de converger, sea uno de los mejores o el mejor, se volverá a realizar la ejecución pero con un mayor número de ciclos máximos.

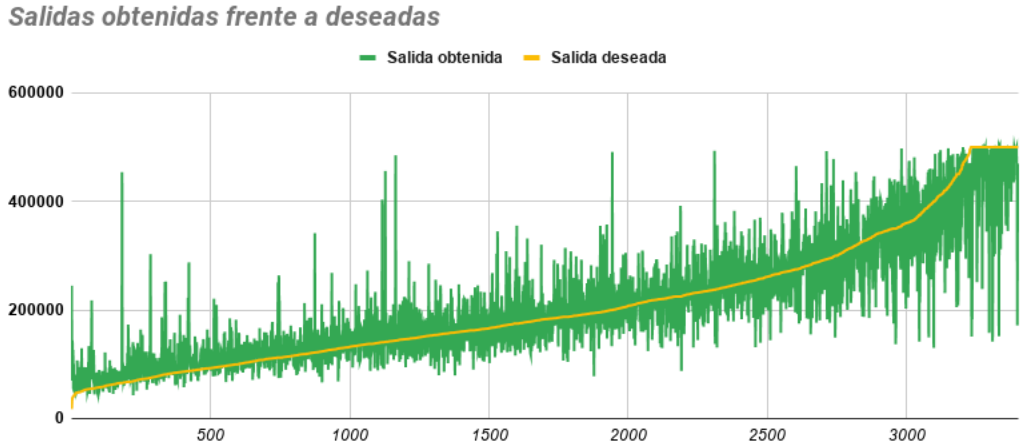
Tal y como se puede ver en la tabla anterior, el error de test disminuye según aumenta la complejidad de la topología del modelo. Siendo el **Experimento 4-3** el que genera menos error con un valor de **0.07099247**.

La siguiente gráfica muestra la evolución del error de entrenamiento y validación de este experimento, en ella se puede ver como progresan de manera similar, aunque entre la iteración nº 1000 y la nº 2000 se separan y el error de validación aumenta más que el de entrenamiento. Esto puede deberse a un *overfitting*.



4.3. Análisis

En esta sección se analizan las salidas obtenidas frente a las deseadas del mejor experimento del *Perceptrón Multicapa* (Experimento 4-3):

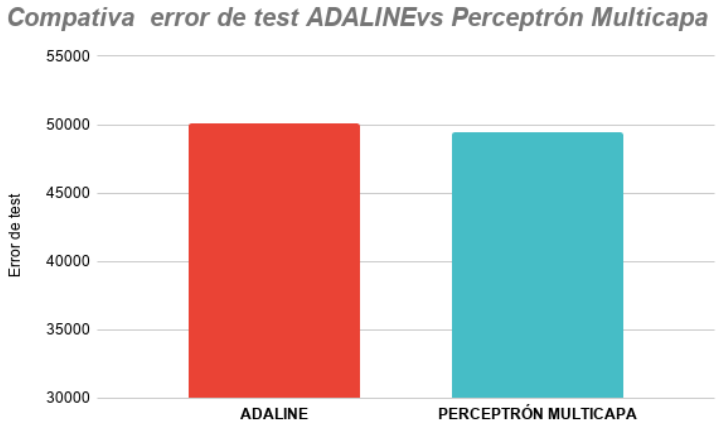


Se puede observar que tiene un comportamiento muy similar al *ADALINE*, aunque parece tener bastantes más picos. Con este ajuste consigue un error medio en test del: **$49430.49**, que sigue siendo un error muy alto.

5. Comparación de Modelos

Una vez realizados y analizados cada uno de los algoritmos por separado, se procede a su comparación con el objetivo de determinar cuál ofrece una mejor solución al problema planteado. Se van a contrastar los mejores experimetos de cada algoritmo.

Aunque el error del modelo del *Perceptrón Multicapa* sea mejor, no supone una mejora sustancial, e incluso, dado que el tiempo de entrenamiento del perceptrón multicapa es mucho mayor, se podría llegara a considerar que el error del modelo de *ADALINE* es mejor. El modelo de ADALINE usa **2836** ciclos menos que el del perceptrón multicapa.



Tal y como se puede ver en la gráfica anterior, la diferncia entre el error de un algoritmo y otro es ínfima. En concreto, el *ADALINE* se equivoca de media tan solo en $652.49 más que el *Perceptrón Multicapa*.

6. Conclusiones

A pesar de que muchas veces se puede considerar que modelos más complejos simpre suponen soluciones más precisas, en aprendizaje automático no siempre es cierto. A veces puede incluso cumplirse lo contrario. En este caso se puede ver algo por el estilo.

Aunque el modelo no lineal tiene mayor complejidad tanto en estructura como en tiempo de entrenamiento, su predicción no está muy lejos de la generada por el modelo lineal *ADALINE*.

Respecto a los conceptos de esta práctica, ha sido especialemnte útil la programación del algoritmo *ADALINE* para ganar confianza y familiarización con la estructura y método de aprendizaje de las redes neuronales, además de ver como influyen los distintos parámetros modificables en ellos.

El *Perceptrón* no aportaba demasiado en tema de conceptos, pero sí ayuda a aprender en qué consiste un proceso de experimentación con una red neuronal, y sirve como toma de contacto con el lenguaje de programación *R*.