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 *Perceptron Multicapa*.

El *Adaline* es un modelo **lineal**, mientras que el *Perceptron 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

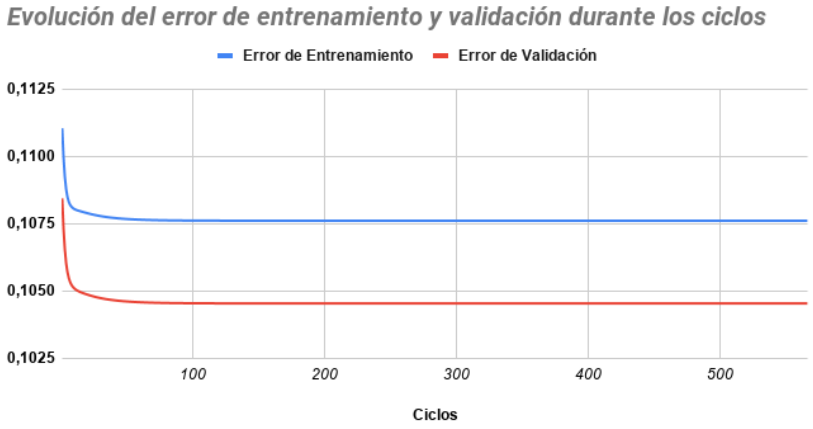
Tabla de resultados obtenidos por experimento:

	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**.

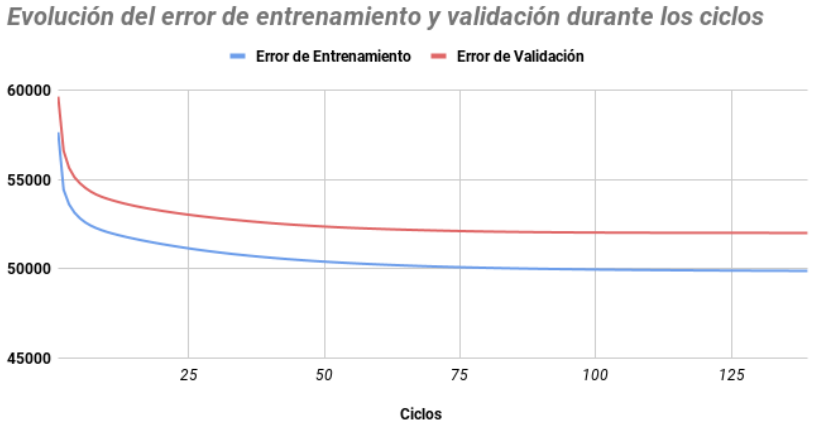


Salida No Normalizada

Tabla de resultados obtenidos por experimento:

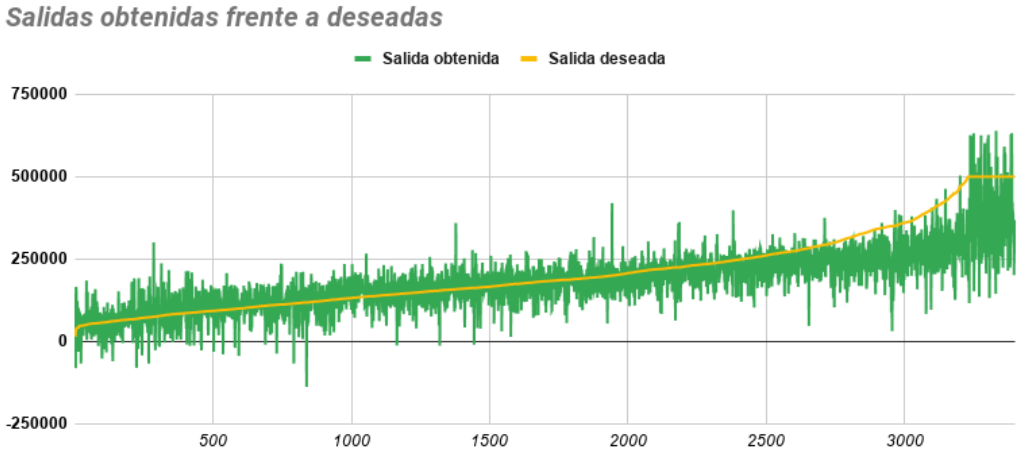
	Experimento 1	Experimento 2	Experimento 3	Experimento 4
Tasa aprend.	0.5	0.22	0.05	0.01
Ciclos	5	8	30	139
Err. entren.	69245.64846716617	55508.988233795564	50017.42808457685	49891.46770073132
Err. valid.	69642.07395718427	56733.54239803611	51961.06532331261	52020.971503368535
Err. test	69986.82287304835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

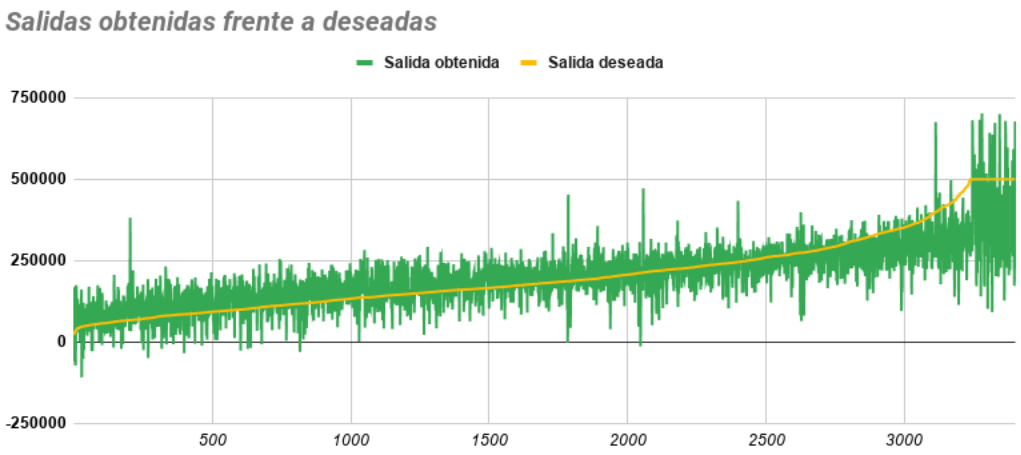


Como se aprecia en la gráfica anterior, la salida obtenida se acerca a la salida deseada, excepto en los valores más altos(a partir de $250000), donde se produce una mayor disparidad. Esto puede deberse a que, los valores que van desde los $0 hasta los $250000, siguen una distribución relativamente lineal, 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 **2000**.

4.2. Resultados Obtenidos

	Experimento 1-1	**Experimento 2-8**	**Experimento 3-8**	**Experimento 4-8**	**Experimento 5-8**
Topología	c( )	c(20)	c(20,20)	c(10)	c(10,10)
Tasa aprend.	0.01	1	1	1	1
Ciclos ópt.	2000	1571	1993	2000	2000
Err. entren.	0.10026694	0.07942846	0.07477689	0.08227377	0.0789468
Err. valid.	0.09714262	0.0796467	0.07589745	0.08090108	0.07901947
Err. test	0.09627861	0.07973496	0.0756311	0.08191064	0.07879357

4.3. Análisis

5. Comparación de Modelos

6. Conclusiones