REDES DE NEURONAS ARTIFICIALES

PRÁCTICA 1

(Problema de Regresión)

Grado en Ingeniería Informática

Campus de Colmenarejo

Curso 2020/2021

Autor

Eduardo Ureña Toledano - 100329937@alumnos.uc3m.es

Daniel Martinez Lianes - 100346118@alumnos.uc3m.es

Índice

1. Introducción

En esta práctica se va a tratar un problema de regresión con Redes de Neuronas Artificiales, utilizando dos modelos:

Adaline: Es un modelo lineal, que a diferencia del perceptrón, que utiliza una salida binaria (sólo tiene en cuenta si se equivoca o no), utiliza directamente la salida de red teniendo en cuenta cuánto se ha equivocado.

Perceptrón Multicapa: Es un modelo no lineal, que se diferencia del modelo Adaline en que sus funciones de activación no lineales permiten la obtención de regiones discriminantes curvas.

Se hará una predicción del precio medio de la vivienda en diferentes distritos de California, a partir de información derivada del censo nacional de 1990.

Antes del aprendizaje se realizará un preprocesamiento de los datos, normalizando, aleatorizando, y separando los datos en 3 conjuntos: entrenamiento (60%), validación (20%) y test (20%).

En cada modelo se realizarán varios experimentos, cuyos resultados se mostrarán en una tabla, y también varias gráficas representativas de la evolución de los errores de entrenamiento y validación de los mejores experimentos, además de una gráfica comparativa con los valores obtenidos frente a los deseados del mejor experimento, y finalmente se compararán los dos modelos sacando las conclusiones pertinentes.

2. Preprocesamiento de los datos

Para utilizar reglas de aprendizaje en redes de neuronas, los datos tienen que ser numéricos, y deben estar normalizados.

La fórmula para normalizar es: *VarNorma = (VarOriginal - ValorMin) / (ValorMax - ValorMin)*

También es importante aleatorizar o desordenar los datos para evitar sesgos en el aprendizaje. En nuestro caso los datos se separan en 3 subconjuntos: conjunto de entrenamiento (aprendizaje), conjunto de validación (parada del aprendizaje y determinación de los hiperparámetros de la red para evitar el sobreaprendizaje) y conjunto de test (generalización).

Para realizar la normalización, aleatorización y separación de los datos en los 3 subconjuntos, se ha utilizado el programa “Excel”. Los datos suministrados se han pegado en una hoja de “Excel”, poniendo cada atributo en una columna; se ha buscado el valor máximo y mínimo de cada atributo, ordenando cada columna de menor a mayor, y se han utilizado para generar los datos normalizados con su fórmula en columnas adyacentes a cada atributo. Estas columnas se han pegado en otra hoja como valores. En esta nueva hoja se ha generado una nueva columna con la función “ALEATORIO()”, que genera números aleatorios entre 0 y 1, y se han ordenado las filas en base a dicha columna, de manera que los datos han quedado así aleatorizados. Esta hoja se ha separado en 3, habiendo en cada una el 60%, el 20% y el 20% respectivamente. Finalmente, cada hoja se ha guardado como un fichero .txt, y en cada uno de estos ficheros se han cambiado las “,” por “.” para que sea más fácil su lectura desde los programas que ejecuten cada uno de los modelos.

3. Predicción con Adaline

El modelo ha sido programado en java, y a continuación se describen las fases por las que pasa el programa:

- Se crean dos funciones para leer y escribir ficheros.

- Se declara una variable que contendrá el número de atributos; se leen los tres ficheros de entrenamiento, evaluación y test cargandolos en tres listas de String; se crean tres matrices de Double con tantas filas y columnas como líneas de entrenamiento y atributos haya; y se rellenan las matrices con los datos contenidos en las listas convirtiéndolos en Double.

- Se inicializan las variables. En la inicialización de estas cabe destacar que en todos los experimentos, inicialmente, el número de ciclos es igual a 10.000, pero posteriormente se queda como definitivo el número en el que se obtiene el menor error de validación de ese experimento; se realizarán 6 experimentos con las siguientes tasas de aprendizaje: 0.85, 0.5, 0.1, 0.01, 0.001 y 0.0001; los pesos y el umbral se inicializan aleatoriamente entre -1 y 1, y se usan los mismos para todos los experimentos.

- En cada ciclo, por cada línea de entrenamiento, se calcula la salida de la red con la fórmula $y(x) = f(x_1w_1 + x_2w_2 + ... + x_nw_n + \theta)$. La diferencia entre el valor obtenido y el esperado se utiliza para calcular la variación que hay que aplicar en los pesos y en el umbral de la siguiente iteración ($\Delta_p w_i = \gamma(d^p - y^p)x_i$; ; $\Delta_p\theta = \gamma(d^p - y^p)$) con las siguientes fórmulas:

$w_j(t + 1) = w_j(t) + \Delta_p w_j$; ; $\theta(t + 1) = \theta(t) + \Delta_p\theta$

- Con los pesos y umbral obtenidos al final de un ciclo, se vuelve a entrenar, sin cambiar ya los pesos y el umbral, para calcular el error cuadrático medio del ciclo, utilizando en cada patrón para luego hacer la media, la siguiente fórmula:

$$E = \frac{1}{m}\sum_{p=1}^{m}(d^p - y^p)^2$$

Se realiza lo mismo con los datos del conjunto de validación, para obtener el error cuadrático medio de validación. El error obtenido en cada ciclo se compara con el del anterior, y si es menor se guarda en una variable, quedando así guardados en dicha variable los pesos y el umbral obtenidos cuando el error cuadrático medio de validación es mínimo.

- Una vez se ha llegado al número de ciclos máximo, se vuelven a realizar las acciones anteriores con los datos del conjunto de test, usando los pesos y umbral obtenidos en el ciclo en que el error cuadrático medio de validación ha sido mínimo.

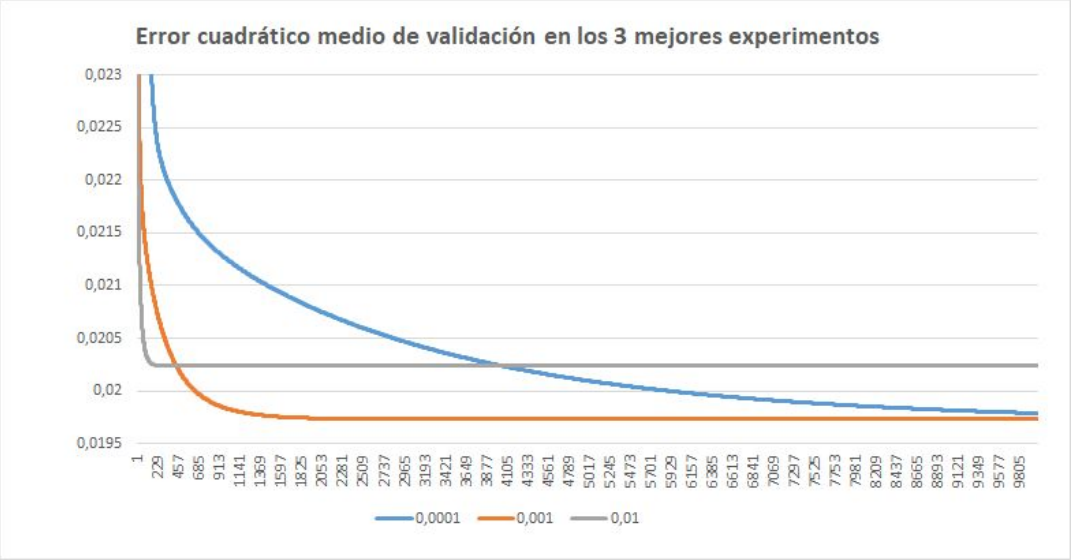
- Se guardan pesos y umbral obtenidos en el último ciclo, y el último error cuadrático medio de entrenamiento, validación y test. Al ser los últimos pesos, umbral y errores los que se guardan, será necesario volver a ejecutar el programa, pero con los ciclos máximos igual al ciclo en que se ha obtenido el menor error de validación.

- Finalmente, se generan diversos ficheros necesarios para comparar los diferentes experimentos.

En la siguiente tabla se muestran los resultados obtenidos para los experimentos realizados:

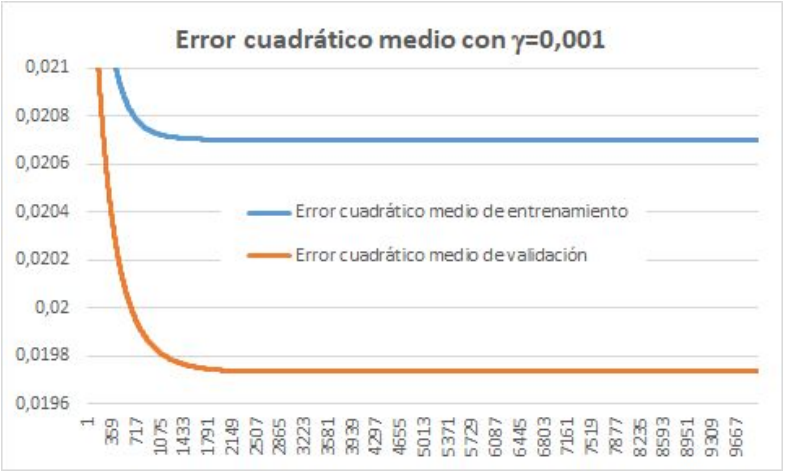
	Exp.1 ***γ = 0,85***	**Exp.2** ***γ = 0,5***	**Exp.3** ***γ = 0,1***	**Exp.4** ***γ = 0,01***	**Exp.5** ***γ = 0,001***	**Exp.6** ***γ = 0,0001***
Peso 0	-0,4800550 031419943	-0,6669403 576616372	-0,8371393 861091752	-0,8687316 457604357	-0,8797310 878051676	-0,8819729 49426195
Peso 1	-0,7652427 127189897	-0,7881038 675075418	-0,8154353 055822681	-0,8336153 682178526	-0,8226041 708170521	-0,8227595 305376872
Peso 2	0,60505168 69913013	0,28686149 70709041	0,16400535 16286915	0,12986796 43964551	0,12594796 33078975	0,12508562 62600304
Peso 3	-0,5376704 269070984	-0,6981315 505763994	-0,6869029 674789062	-0,6417676 957099341	-0,6560793 369241731	-0,5064449 702327827
Peso 4	1,21820143 34994826	1,14974952 77735714	1,40276343 65503645	1,37247511 56580217	1,39383357 14680703	1,04927126 09576864
Peso 5	-2,4695764 468441865	-2,5558954 812035637	-2,5386464 749182056	-2,5981402 970880487	-2,6046304 539713474	-2,4401992 311214755
Peso 6	0,93175404 7148159	0,79276878 80194877	0,55789224 17215345	0,65151497 30434712	0,65456330 55651451	0,80389269 13335221
Peso 7	1,62721859 78035587	1,37688373 29381276	1,26335602 6164042	1,22558819 44513514	1,21303213 153729	1,19499025 99456954
Umbral	0,67088663 06064021	0,75842586 00270553	0,74154066 86356373	0,72817067 11974879	0,71868030 8778907	0,72063783 22473146
Ciclos necesarios	1	3	455	294	2958	10000
Error de entrenamiento	0.33509469 76653821	0.09824347 447474929	0.02668688 549067251	0.02112182 308567369	0.02069784 37687523	0.02070724 58439116
Error de validación	0.33370089 873696557	0.09838697 896637534	0.02616062 504128316	0.02024114 489406749	0.01973510 654237664	0.01978852 6122763
Error de test	0.32915293 48156684	0.09575884 025048957	0.02621178 588975177	0.02120255 463265233	0.02091592 205378153	0.02103219 67444813

Cómo puede apreciarse, el menor error, tanto de entrenamiento cómo de validación y test, se encuentra con una tasa de aprendizaje de 0,001. Cabe destacar que hemos puesto como criterio de parada que la red se pare al haber realizado un máximo de 10000 ciclos, y que con una tasa de aprendizaje de 0,0001 no da tiempo a que encuentre el menor error que puede encontrar, pero dicho error lo encontraría a los 29354 ciclos frente a los 2958 en que se encuentra el menor error con una tasa de 0,001, y además el error encontrado sería menor en tan sólo un 0,01%. Si tenemos en cuenta que la media del precio medio de las viviendas es aproximadamente 207301 $, la diferencia de error sería de media de 20,73 $, por lo que no merecería la pena tener un gasto computacional más alto, teniendo en cuenta la poca mejora que se obtiene.

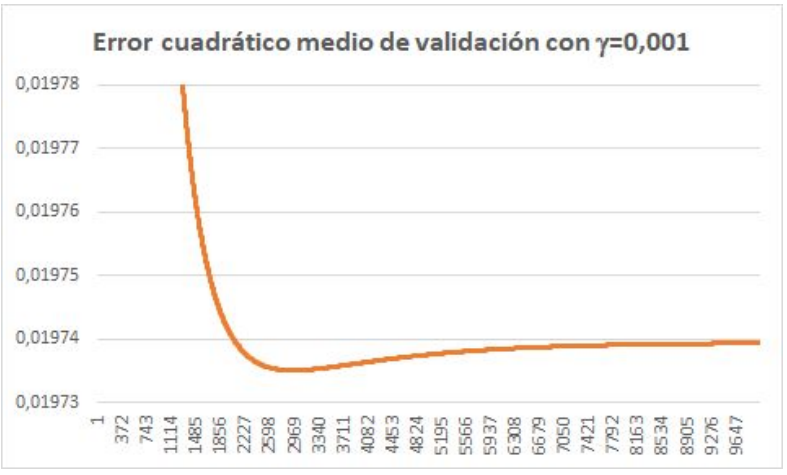


En la gráfica anterior se puede apreciar cómo se comportan los errores cuadráticos medios de validación de los 3 experimentos que han dado mejores resultados. En el experimento con tasa de aprendizaje 0,01 la red aprende demasiado rápido, por lo que se estanca a los pocos ciclos; en el experimento con tasa de aprendizaje 0,0001 la red aprende demasiado despacio, por lo que no le da tiempo a encontrar el menor error posible en 10000 ciclos; y el experimento con tasa de aprendizaje 0,001 aprende en el tiempo justo, por lo que nos proporciona el menor error de todos los experimentos en los ciclos máximos que hemos decidido poner (10000 ciclos).

En la gráfica de la derecha se puede apreciar que el error cuadrático medio de validación es algo menor que el de entrenamiento durante todos los ciclos.

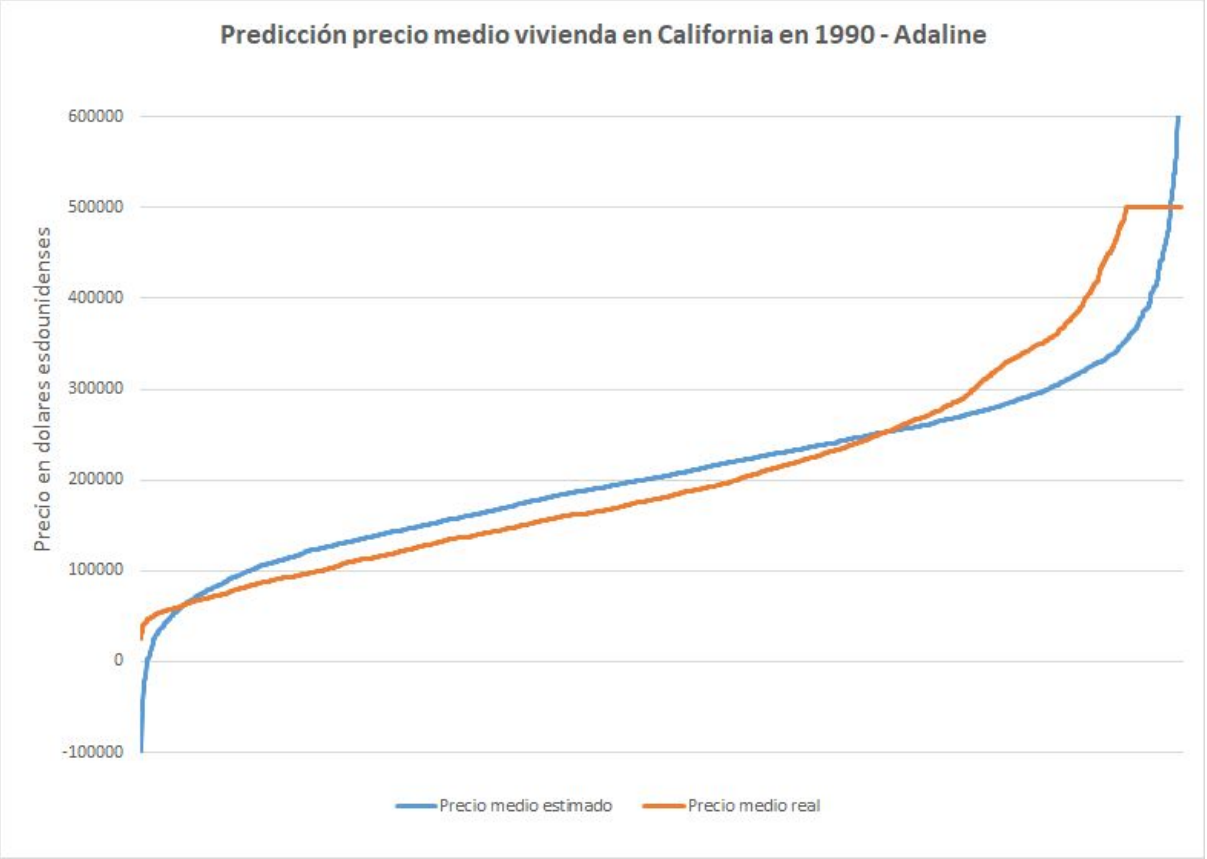


La sensación al ver la gráfica es que los errores se estancan, incluso bajan un poco a partir de los 2149 ciclos, pero según los datos el mínimo error de validación se encuentra a los 2958 ciclos, por lo que si acercamos un poco la gráfica, podemos observar mejor cómo evoluciona realmente el error cuadrático medio de validación.



En la siguiente página se puede ver la diferencia entre el precio medio real de las viviendas y el estimado en el experimento 5 usando una tasa de aprendizaje

de 0,001, apreciándose que en los extremos el precio se va demasiado abajo (izquierda) o demasiado arriba (derecha), pero en las demás partes de la gráfica el precio se estima bastante bien, produciéndose un error medio de aproximadamente ± 4091 $.



4. Predicción con Perceptrón Multicapa

En este modelo Perceptrón Multicapa, programado en R, se nos permite escoger el número de capas, el número de neuronas de cada capa, la razón de aprendizaje, y los ciclos máximos para encontrar una solución. El programa reajusta los ciclos máximos si el error cuadrático medio de validación empieza a subir y el mínimo error se encuentra antes de que pasen dichos ciclos máximos.

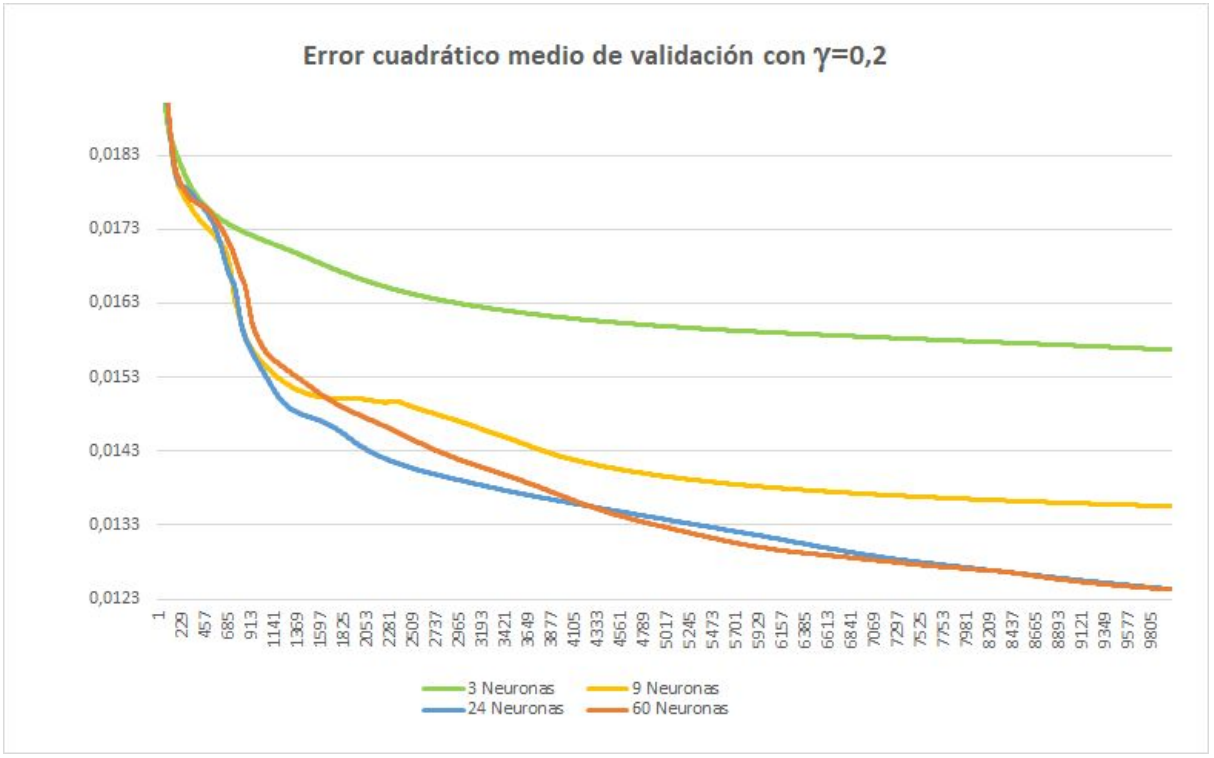
Después de algunas pruebas iniciales hemos decidido realizar todos los experimentos con 10000 ciclos, ya que con más no se obtenía una mejora significativa del error y el coste de computación era mucho más alto. Además sólo hemos usado una capa oculta dada la sencillez del problema.

En cuanto a la razón de aprendizaje, hemos usado 0.001, 0.01, 0.05, 0.1 y 0.2; y en cuanto al número de neuronas, 3, 9, 24 y 60. Los mínimos errores de validación se han obtenido con "*ɣ=0.2 - 24 neuronas*" y con "*ɣ=0.2 - 60 neuronas*". Con 60 neuronas se obtiene el mejor error de validación, pero con 24 el mejor error de test, por lo que consideramos que el mejor experimento es con "*ɣ=0.2 - 24 neuronas*".

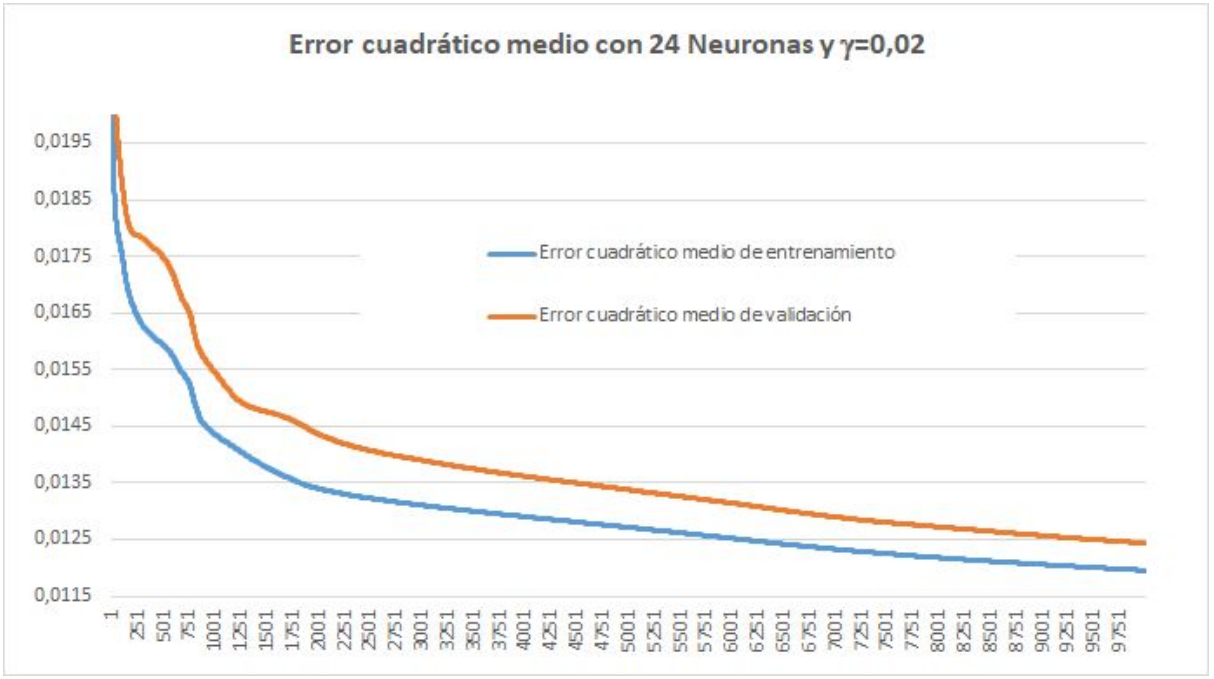
En la siguiente tabla se muestran los resultados obtenidos para los experimentos realizados:

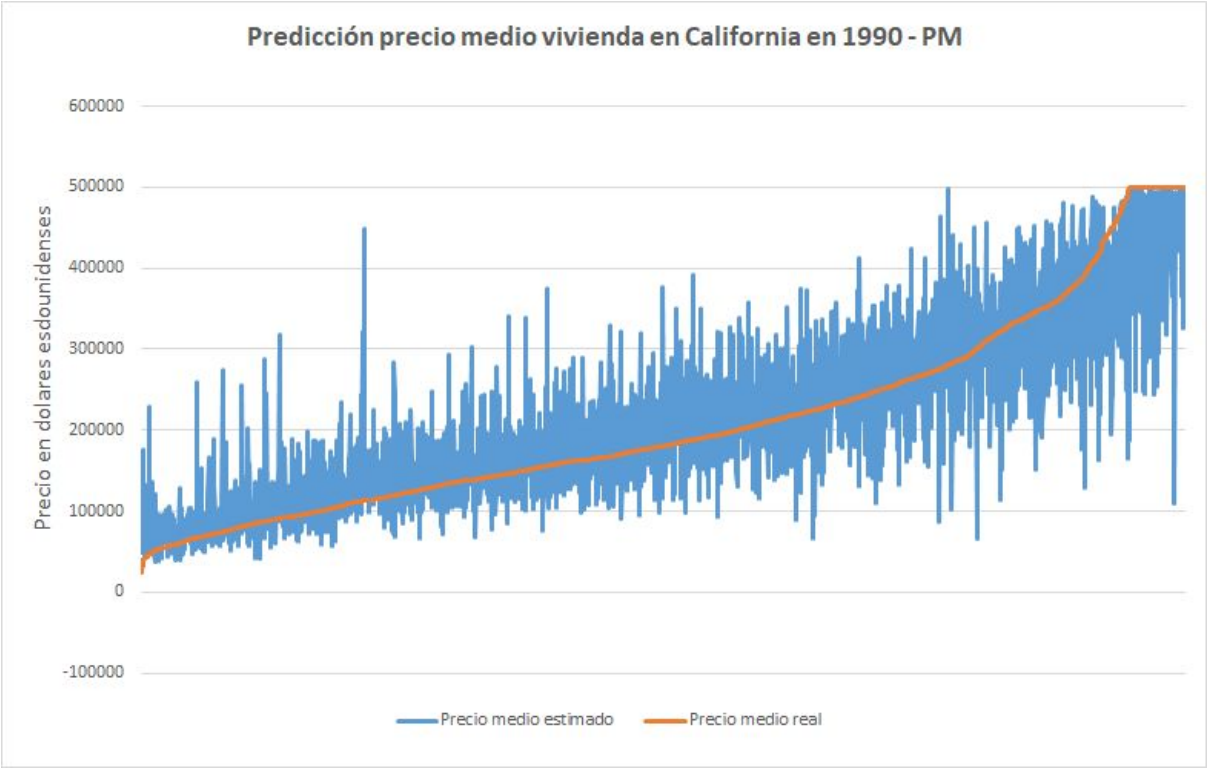
	Error de entrenamiento	Error de validación	Error de test
n=3; ɣ=0,001	0,0171212714047545	0,0167496299530167	0,017600076680424
n=3; ɣ=0,01	0,0160417141354315	0,0154936541848274	0,0166606874815541
n=3; ɣ=0,05	0,0158517549668372	0,0151979354511779	0,0165289116935054
n=3; ɣ=0,1	0,0159534387710322	0,0153241720606094	0,0166213283883653
n=3; ɣ=0,2	0,0162093590639728	0,0156690080948701	0,0167708649133134
n=9; ɣ=0,001	0,0170490154147164	0,0166728727379025	0,017517250827133
n=9; ɣ=0,01	0,0154019264265487	0,0147993985362055	0,0160015274982222
n=9; ɣ=0,05	0,0136925748021667	0,0135264100826039	0,0143253693155536
n=9; ɣ=0,1	0,0133805394244573	0,0132097573647863	0,0141003689022577
n=9; ɣ=0,2	0,0137341316748456	0,0135545248694678	0,0144410696825607
n=24; ɣ=0,001	0,0172999934395324	0,0168594342236613	0,0177001920452978
n=24; ɣ=0,01	0,0154395162897868	0,0148587117384555	0,0159592720422726
n=24; ɣ=0,05	0,0133685637865526	0,0132029480709352	0,014038620833957
n=24; ɣ=0,1	0,0131689574201245	0,0131056233501986	0,0139803263756627
n=24; ɣ=0,2	0,0122675897029732	0,0124343839229605	0,0132843049777036
n=60; ɣ=0,001	0,0178030174069381	0,017299063267227	0,0182123981250403
n=60; ɣ=0,01	0,0155261772745806	0,0149290975701233	0,0160532149982054
n=60; ɣ=0,05	0,013372377732853	0,0133102606331461	0,0139750262886751
n=60; ɣ=0,1	0,0129287574253681	0,0128540244102706	0,0136237215893
n=60; ɣ=0,2	0,0123315856430273	0,0124281849243109	0,0134483963128474

Al haber obtenido los mínimos errores de validación con *ɣ=0.2*, hemos decidido mostrar una gráfica con los errores de validación de los experimentos que hemos realizado con dicha razón de aprendizaje y los cuatro número de neuronas:



En la siguientes dos gráficas se muestran los errores de entrenamiento y validación para el que consideramos el mejor experimento (ɣ=0,2 y 24 neuronas), y la relación entre los resultados esperados y los obtenidos para el precio medio de la vivienda en California:





Cómo se puede apreciar, los valores obtenidos en la predicción no son lineales, debido a que el perceptrón multicapa produce una salida no lineal.

5. Comparación de ambos modelos y conclusiones

Después de todos los experimentos realizados, se aprecia que se obtiene un menor error con el perceptrón multicapa, del orden de 0.0076, que trasladado a dólares sobre el precio medio de las viviendas, es de ± 1579 $ de diferencia. Por ello, aunque con el perceptrón multicapa se obtiene un error menor, en problemas sencillos como este se aprecia que el modelo Adaline es perfectamente válido, porque la diferencia es pequeña.

Cómo conclusión final, para hacer estas estimaciones concretas se escogería el modelo Adaline, ya que su coste computacional es mucho menor.