Iván Fernández Navarro
NIA: 100383564

Carlos García Corral
NIA: 100383380

Grupo 84

Práctica 1: Problema de regresión

Redes de Neuronas Artificiales

ÍNDICE:

1. Introducción

En esta práctica comprobaremos la eficacia de dos modelos de redes de neuronas supervisados a la hora de predecir el precio de la vivienda de diferentes distritos de California en 1990
Los modelos que pondremos a prueba son el modelo lineal Adaline y el modelo no lineal Perceptrón Multicapa.

Cada distrito tendrá 8 atributos que lo identifiquen:

- Longitud
- Latitud
- Antigüedad media (mediana) de una casa dentro de un distrito
- Cantidad total de habitaciones en las casas de un distrito
- Cantidad total de camas en las casas de un distrito
- Cantidad total de residentes en un distrito
- Cantidad total de grupos familiares en un distrito
- Ingreso medio de los grupos familiares de un distrito

Usando esos 8 valores, nuestro modelo deberá intentar predecir el precio medio de la vivienda en ese distrito, valor que luego comprobaremos con el precio real medio de la vivienda en ese distrito.

2. Preproceso de los datos

Antes de entrenar y poner a prueba los modelos de regresión, es importante tratar los datos de forma que aumente la exactitud de las predicciones.
Es recomendable normalizar los *inputs* de manera que sus valores estén en el rango de 0 a 1 para cada atributo. Es decir, en cada atributo, el valor máximo será 1 y el mínimo 0, y el resto de valores tendrán un valor proporcional a estos.
Otro paso del preproceso es la aleatorización de los datos, es decir, cambiar el orden aleatoriamente de los datos, para evitar posibles patrones que se encuentren según la colocación de los datos.
El último paso consiste en la separación de los datos en tres datasets: entrenamiento, validación y testing.
El set de entrenamiento, el más grande, tendrá el 60% de los datos y será el que usemos a la hora de entrenar la red de neuronas.
El set de validación, con el 20% de los datos, nos servirá para decidir los hiperparámetros óptimos para el problema.
Por último, el set de testing nos mostrará la capacidad de la red entrenada en un set no utilizado en el entrenamiento.

3. Modelo Adaline

3.1. Experimentación realizada

Para este experimento, desarrollamos un programa en Python que realiza el aprendizaje Adaline. El programa contiene una función a la cual se le pasan los hiperparámetros y, ésta, realiza todo el aprendizaje.
De esta manera, hemos podido probar muchos ratios de aprendizaje sin tener que manualmente ir cambiando el código. Los valores de ratio de aprendizaje que hemos probado son: 0.001, 0.005, 0.01, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5 y 1.

Para comprobar que el programa realizaba el aprendizaje de forma correcta, guardamos la evolución de los errores (de entrenamiento y validación) en un archivo .csv.
A continuación, se muestran una serie de gráficas de errores de los experimentos más relevantes:

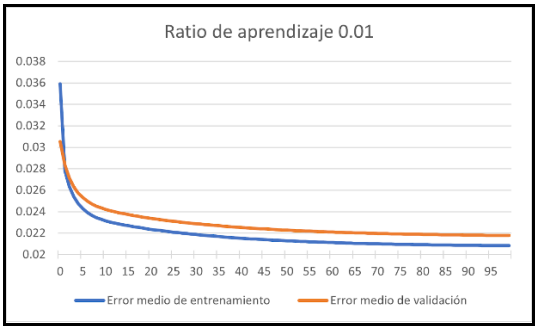


Ilustración 1. Ratio de aprendizaje 0.01

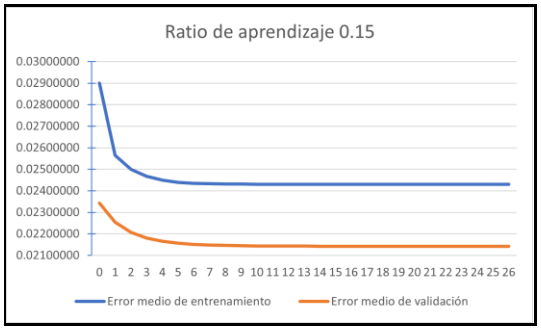


Ilustración 2. Ratio de aprendizaje 0.25

3.2. Resultados obtenidos

En esta tabla mostramos para cada modelo sus hiperparámetros y sus resultados:

# modelo	# máximo de ciclos	# de ciclos óptimo	Ratio de aprendizaje	Error de entrenamiento	Error de validación
1	100	100	0.001	0.02214297	0.023081
2	100	100	0.005	0.02110499	0.021894
3	100	100	0.01	**0.02082618**	0.021772
4	100	61	0.05	0.02171973	0.021421
5	100	48	0.1	0.02294096	0.021464
6	100	27	0.15	0.024307490	**0.021420**
7	100	16	0.2	0.02583818	0.021506
8	100	13	0.25	0.02756161	0.021835
9	100	100	0.3	0.029516248	0.022509
10	100	7	0.35	0.03174731	0.023601
11	100	6	0.4	0.03431917	0.024913
12	100	5	0.45	0.03731486	0.026237
13	100	99	0.5	0.04086115	0.027555
14	100	64	1	0.57370919	0.087459

Con estos resultados podemos concluir basándonos en el error de validación que el mejor ratio de aprendizaje es **0.15**. Si nos basamos en cambio en el error de entrenamiento, el mejor ratio sería **0.01.** La razón por la cual elegimos 0.15 en lugar de 0.01 es porque los resultados del ratio 0.01 nos indican que sufre de un caso de *overfitting* o sobreaprendizaje al dataset de entrenamiento, es decir, que los pesos se han ajustado demasiado al set de entrenamiento sacrificando a cambio su capacidad de predicción general.

Los resultados finales del ratio de aprendizaje **0.15** son:

Error MSE (normalizado): 0.01964389719727566
Error medio (raíz del MSE) (sin normalizar): 72135.9405
Umbral final: 0.7073238010171267

Vector de pesos final

Peso final w1	Peso final w2	Peso final w3	Peso final w4	Peso final w5	Peso final w6	Peso final w7	Peso final w8
-0.883957125	-0.815139706	0.114454709	-0.679010652	1.488625733	-2.277549373	0.433568429	1.203854049

3.3. Análisis de los resultados

Debido a que tenemos muchos datapoints, es difícil visualizar con claridad cómo de preciso es nuestro modelo, para ello es más preciso mirar los resultados numéricos. No obstante, a continuación, mostraremos diferentes visualizaciones para representar gráficamente nuestros resultados:

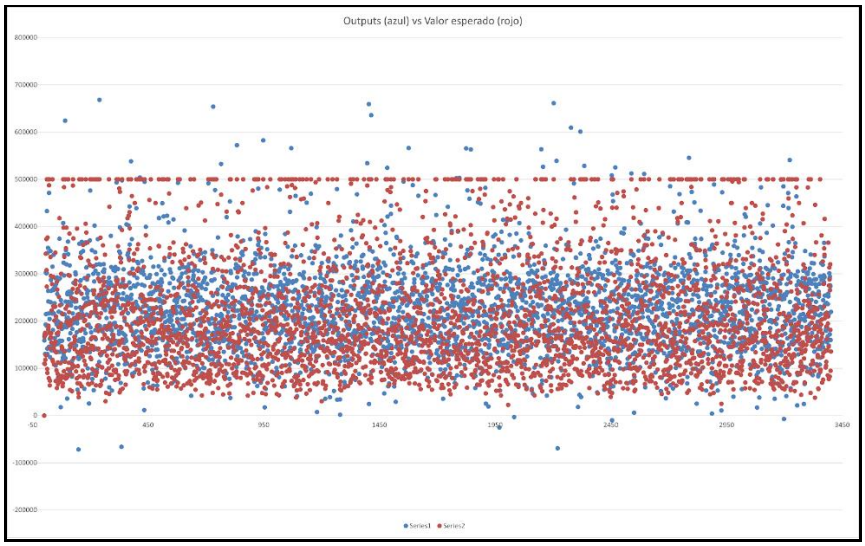
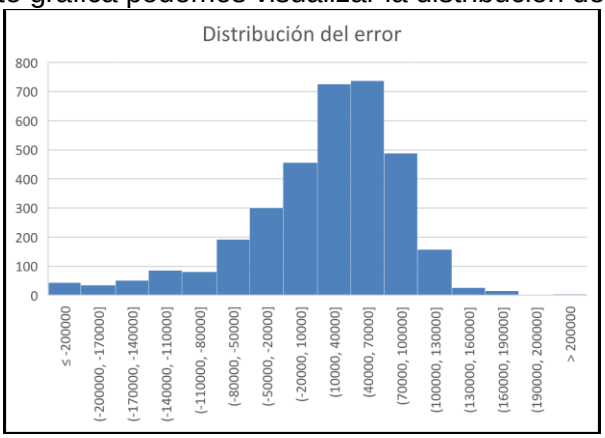


Ilustración 2. Outputs (en azul) vs. los resultados esperados (en rojo)

En la siguiente gráfica podemos visualizar la distribución de errores:



Como podemos ver, los errores siguen una distribución muy similar a la normal, centrándose en torno al 0. Si bien, hay más fallos en los que el modelo se queda muy por debajo en vez de muy por encima del precio real.

4. Modelo Perceptrón Multicapa

4.1. Experimentación realizada

Para ver qué parámetros resultan mejor para el modelo de perceptrón multicapa, se va a ir experimentando con el número de capas ocultas y sus respectivos nodos, y el ratio de aprendizaje.
El número máximo de ciclos quedará constante, siendo así este parámetro igual a 10,000 ciclos.
En cuanto al ratio de aprendizaje analizaremos los valores 0.001, 0.005, 0.01, 0.02, 0.03, 0.1 y 0.5. De esta manera, tendremos 7 ratios de aprendizaje con los que experimentar.
Si hablamos sobre las capas ocultas y sus nodos, se ha querido hacer experimentaciones con 1, 2 y 3 capas ocultas cambiando la distribución de nodos. Se han escogido como nodos los valores 10, 15, 20, 30 y sus diferentes combinaciones. De esta manera, tendremos 16 topologías diferentes.
Como se ha experimentado con 114 modelos diferentes, y cada uno se ejecuta con 10,000 ciclos, se ha modificado el código del *script* de tal forma que se automatice la experimentación mediante bucles.
Los atributos *topología* y *razón* se han convertido en listas, en las que se inicializaran las 16 topologías y los 7 ratios de aprendizaje. Mediante el bucle se crearán los 114 modelos combinando las topologías y los ratios.

4.2. Resultados obtenidos

# modelo	# máximo de ciclos	# de ciclos óptimo	Capas ocultas	Ratio de aprendizaje	Error de entrenamiento	Error de validación	Error de test
1	10000	10000	10	0.001	0,0170576422356683	0,0175594035983698	0,0156282975361634
2	10000	10000	10	0.005	0,0158614790203677	0,0160415727284565	0,014635152903398
3	10000	10000	10	0.01	0,0156627025703318	0,0157439695496615	0,0144002521449591
4	10000	10000	10	0.02	0,0144273189389724	0,0145550928426109	0,0135768197796112
5	10000	10000	10	0.03	0,0135838777396183	0,0140589187641201	0,0131091122884042
6	10000	10000	10	0.04	0,0132932874374683	0,0138029919239397	0,0131093189346682
7	10000	10000	10	0.05	0,013194850122439	0,0137488095249645	0,0130784983351719
8	10000	10000	10	0.1	0,012840849759102	0,0133803481701779	0,0127272917469635
9	10000	10000	10	0.5	0,0124425023383251	0,0125849955413208	0,0119515364283218
10	10000	10000	20	0.001	0,0174921071180028	0,0179635649825387	0,0158837104137358
11	10000	10000	20	0.005	0,0159232264247701	0,0160844911389531	0,0146848701116031
12	15000	15000	20	0.01	0,0152306518412324	0,0153003333495377	0,0142465965103645
13	15000	15000	20	0.02	0,0137555614637376	0,0141398241335515	0,0132222332623574
14	15000	15000	20	0.03	0,0132932658668329	0,0137813794337037	0,0130524986335277
15	10000	10000	20	0.1	0,012802049301178	0,0130481532244734	0,012309663958015
16	10000	10000	20	0.5	0,0118875658899403	0,0127134455746532	0,0122666749529429
17	10000	10000	30	0.001	0,0176833924563731	0,0183035089097439	0,0160989508417248
18	10000	10000	30	0.005	0,0158507058872686	0,0160155791588783	0,0146382483701615
19	10000	10000	30	0.01	0,01551866396244	0,0156888028148061	0,0144257594601747
20	10000	10000	30	0.02	0,0137791132534504	0,0141456492689174	0,0131809305369665
21	10000	10000	30	0.03	0,0133362122978792	0,0138776729709279	0,013079929354883
22	10000	10000	30	0.1	0,0126880174257789	0,0136787865979116	0,0129589612023026
23	10000	10000	30	0.5	0,0123654553141516	0,0134869339396087	0,0124618456369766

24	10000	10000	10, 10	0.001	0,0168570939576749	0,0173131117635647	0,0154086043047257
25	10000	10000	10, 10	0.005	0,0156597175346825	0,015719385003953	0,0144138700671535
26	10000	10000	10, 10	0.01	0,0153067551452584	0,0154731347383938	0,0143804717316202
27	10000	10000	10, 10	0.02	0,0142465867088882	0,0143304580925904	0,013395289672557
28	10000	10000	10, 10	0.03	0,0129168600146129	0,0130849175446665	0,0125778617933425
29	10000	10000	10, 10	0.1	0,0115993491530094	0,0119492239114197	0,0114663057174138
30	10000	10000	10, 10	0.5	0,0111454798885436	0,0120722231986246	0,0113361460132647
31	10000	10000	10, 20	0.001	0,017109610702181	0,0175475273583398	0,0155936663088284
32	10000	10000	10, 20	0.005	0,0156870313610871	0,0157907562687984	0,0144434394069605
33	10000	10000	10, 20	0.01	0,0154926087725218	0,0156138744074595	0,0143470588742202
34	10000	10000	10, 20	0.02	0,0136005187897567	0,0138755282962115	0,0130363535856479
35	10000	10000	10, 20	0.03	0,0123724611252778	0,0130773084456253	0,0122501268048879
36	10000	10000	10, 20	0.1	0,0108055777376753	0,0118146354668687	0,0113486062316452
37	10000	10000	10, 20	0.5	0,0104623571635824	0,0116007546723604	0,0113601062918615
38	10000	10000	10, 30	0.001	0,0171536743875418	0,0175772706294214	0,0156133590866238
39	10000	10000	10, 30	0.005	0,0156251928669685	0,0158604614711052	0,0145108802594941
40	10000	10000	10, 30	0.01	0,0152696738338595	0,0156781866249311	0,0142583592204149
41	10000	10000	10, 30	0.02	0,014340258134296	0,0144808381091524	0,0135461826625408
42	10000	10000	10, 30	0.03	0,0121601372408983	0,012714910169475	0,0122469898941492
43	10000	10000	10, 30	0.1	0,0111244839848763	0,0116334290551115	0,0111382343744924
44	10000	10000	10, 30	0.5	0,0099797348799804	0,0114878895747315	0,0110154300227089
45	10000	10000	15, 30	0.001	0,0171435324868255	0,0176009258870368	0,0156022060906066
46	10000	10000	15, 30	0.005	0,015659306834186	0,0158967399584959	0,0145265597994642
47	10000	10000	15, 30	0.01	0,0151788903450091	0,0154589598012199	0,0142028013562033
48	10000	10000	15, 30	0.02	0,0128635925438301	0,0134198188688308	0,0129172447920469
49	10000	10000	15, 30	0.03	0,012250220430973	0,0129010485771679	0,0121002775011409
50	10000	10000	15, 30	0.1	0,0098681953048370	0,0111462000329206	0,0107519025763142
51	10000	10000	15, 30	0.5	0,0115440999021597	0,0119988301822522	0,0114328530640084
52	10000	10000	20, 10	0.001	0,0171956302962337	0,0176833966289902	0,0156163116038739
53	10000	10000	20, 10	0.005	0,0156925956741283	0,0157512518377106	0,014378886457119
54	10000	10000	20, 10	0.01	0,0152137523622239	0,0153049977859621	0,0140768937120398
55	10000	10000	20, 10	0.02	0,0129233321822874	0,0134791244430039	0,0126462336092167
56	10000	10000	20, 10	0.03	0,0118338198647529	0,0123040572317863	0,0118422567240255
57	10000	10000	20, 10	0.1	0,0103178613032126	0,0111850299901117	0,0106376170769877
58	10000	10000	20, 10	0.5	0,0098157919795279	0,0111257349511927	0,0107413538303206
59	10000	10000	20, 20	0.001	0,0172869251571347	0,0177470305036115	0,0156709937843332
60	10000	10000	20, 20	0.005	0,0156101759430379	0,0158272773869406	0,0144257841341379
61	10000	10000	20, 20	0.01	0,0151713934943961	0,015340888489511	0,0141245963142821
62	10000	10000	20, 20	0.02	0,0128102794595651	0,0135595624085357	0,0127107406311323
63	10000	10000	20, 20	0.03	0,0118390121153418	0,0124002835174504	0,0119915865322827
64	10000	10000	20, 20	0.1	0,0106217649903627	0,0115057548463978	0,0111639508590467
65	10000	10000	20, 20	0.5	0,0105455529576375	0,0114008637762232	0,011019153143031
66	10000	10000	20, 30	0.001	0,0174329658109453	0,0179662548207976	0,0158183381193779
67	10000	10000	20, 30	0.005	0,0158078521612555	0,0159530358206859	0,0145478562548639
68	10000	10000	20, 30	0.01	0,0155385712622272	0,0156997631747277	0,0143596443244833
69	10000	10000	20, 30	0.02	0,0133346714180118	0,0138225644461816	0,0130406978291033
70	10000	10000	20, 30	0.03	0,0123885632957618	0,0131018618943917	0,0126288758611288
71	10000	10000	20, 30	0.1	0,0094443924250773	0,0108536857487913	0,0105829840683345
72	10000	10000	20, 30	0.5	0,0108526331345017	0,0116901451170394	0,011394173243666
73	10000	10000	30, 15	0.001	0,0174148633245095	0,0178658072759362	0,0157644336417652
74	10000	10000	30, 15	0.005	0,0157028197228211	0,0158014259701602	0,0144499677006771
75	10000	10000	30, 15	0.01	0,0150477090986005	0,0151897721748175	0,0141367357841531
76	10000	10000	30, 15	0.02	0,0126887335022515	0,0133184920120345	0,0126322358470246

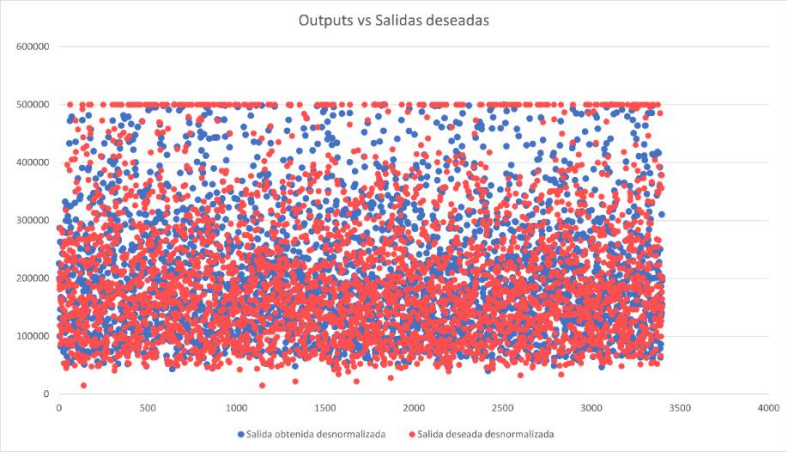
77	10000	10000	30, 15	0.03	0,0120346817530429	0,0128756146083604	0,0124508122651293
78	10000	10000	30, 15	0.1	0,0095694504461726	0,0106119397914075	0,0103382861593142
79	10000	10000	30, 15	0.5	0,0092359775346955	0,0107070383347543	0,0103462002979977
80	10000	10000	10, 10, 10	0.001	0,0174458301371796	0,0178555980849881	0,0157610733643735
81	10000	10000	10, 10, 10	0.005	0,0157262995589562	0,0158839837417012	0,0145076324037192
82	10000	10000	10, 10, 10	0.01	0,0154059015206211	0,0156459386908937	0,0143511090124749
83	10000	10000	10, 10, 10	0.02	0,0146805444007362	0,0150533782102411	0,0140047610057219
84	10000	10000	10, 10, 10	0.03	0,0120322636650547	0,0126949073400973	0,0119735829008799
85	10000	10000	10, 10, 10	0.1	0,0101630918013232	0,0110966686437163	0,0108995454756247
86	10000	10000	10, 10, 10	0.5	0,0114729335899859	0,0120274013134706	0,0114489421444773
87	10000	10000	10, 20, 10	0.001	0,0165506050239344	0,0169550385952258	0,015183291175001
88	10000	10000	10, 20, 10	0.005	0,0156821675842734	0,0158578018083957	0,0145025325766471
89	10000	10000	10, 20, 10	0.01	0,0154017047858242	0,0155658809581374	0,0143627899565918
90	10000	10000	10, 20, 10	0.02	0,0129931022317349	0,0132929700229867	0,0125027007712185
91	10000	10000	10, 20, 10	0.03	0,0120735191227549	0,0126215621243377	0,0118288276427877
92	10000	10000	10, 20, 10	0.1	0,0088612114304881	0,0106256817701045	0,0101805914021268
93	10000	10000	10, 20, 10	0.5	0,0112301142089534	0,0116149930927693	0,0113153448046081
94	10000	10000	10, 30, 10	0.001	0,0171553184008525	0,0176684098759241	0,0155895559705728
95	10000	10000	10, 30, 10	0.005	0,0157484402453943	0,0159101711434811	0,014514271326614
96	10000	10000	10, 30, 10	0.01	0,0154545488922304	0,0156406549921825	0,0143684832496435
97	10000	10000	10, 30, 10	0.02	0,0122533173328807	0,012760025214329	0,0122661999091486
98	10000	10000	10, 30, 10	0.03	0,011155663620791	0,0115880345524149	0,0112982442050481
99	10000	10000	10, 30, 10	0.1	0,0091446969556408	0,0105764861480879	0,0103573066000462
100	10000	10000	10, 30, 10	0.5	0,0105454541182133	0,0115225605311512	0,0108954814518552
101	10000	10000	15, 30, 15	0.001	0,0174130283823952	0,0178369579055297	0,0157320655784593
102	10000	10000	15, 30, 15	0.005	0,0157956968363025	0,0159944029794738	0,0145650420257537
103	10000	10000	15, 30, 15	0.01	0,0155512006545074	0,0157235691931157	0,0144261686184861
104	10000	10000	15, 30, 15	0.02	0,0126447576846267	0,0129234085787637	0,0124940899906876

105	10000	10000	15, 30, 15	0.03	0,0113877142946769	0,0117358709841195	0,0115126005596277
106	10000	10000	15, 30, 15	0.1	0,0094242379341183	0,0108456708405969	0,0105515734453907
107	10000	10000	15, 30, 15	0.5	0,0090473252707585	0,010815623010226	0,0104725215082804
108	10000	10000	30, 15, 30	0.001	0,0172374403571035	0,0176931129971884	0,0156228954033216
109	10000	10000	30, 15, 30	0.005	0,0156899827688009	0,0157978998822662	0,0144043984665938
110	10000	10000	30, 15, 30	0.01	0,0150432150660245	0,0153256683499161	0,0140859816262023
111	10000	10000	30, 15, 30	0.02	0,0125962407705645	0,0130935488755347	0,0123575347610196
112	10000	10000	30, 15, 30	0.03	0,0112464313792195	0,0118363002604023	0,0114983711644121
113	10000	10000	30, 15, 30	0.1	0,0088874680782442	0,0112095602337177	0,0105528628593704
114	10000	10000	30, 15, 30	0.5	0,0104631420717192	0,0114412078610615	0,0109239092696151

Como se puede observar, se han subrayado los mejores errores.
Los errores mínimos de entrenamiento y de test se han conseguido con el modelo número 92 (3 capas ocultas con 10, 20 y 10 nodos en cada una respectivamente; y un ratio de aprendizaje de 0.1), y el error mínimo de validación con el modelo número 99 (3 capas ocultas con 10, 30 y 10 nodos en cada una respectivamente; y un ratio de aprendizaje de 0.1).
Por tanto, se escogerá el modelo 92 como el mejor experimento, ya que buscamos el modelo que logre un mínimo de error en el mayor número de procedimientos.

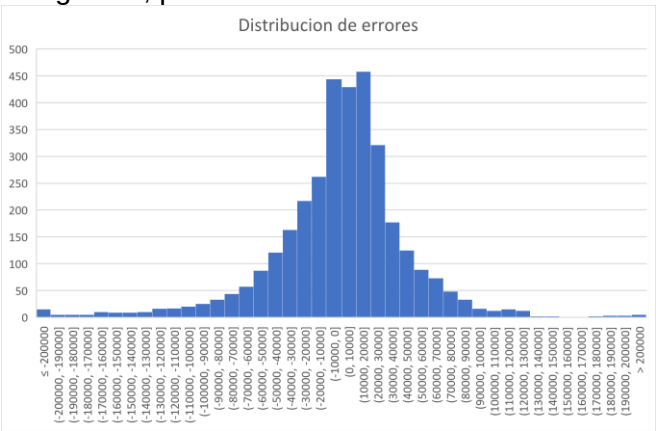
4.3. Análisis de los resultados

Analizar nuestro modelo a base de gráficas es muy difícil, puesto que hay 3400 patrones, y por cada uno de ellos 2 salidas (obtenida y deseada). Por tanto, es mejor fijarse en los datos numéricos que en gráficas. No obstante, se procederá a enseñar las gráficas.
A continuación, se muestra una gráfica del mejor experimento con la salida obtenida de la red y la salida deseada para los datos de test:



En azul, podemos ver las diferentes salidas obtenidas por nuestro modelo; y en rojo, las salidas esperadas.

En la siguiente gráfica, podemos visualizar la distribución de errores:



Como podemos ver, los errores siguen una distribución muy similar a la normal, como en el modelo Adaline, centrándose en torno al 0.

5. Comparación de modelos

En la siguiente tabla se resumirán los parámetros de cada modelo:

Modelo	# máximo de ciclos	# de ciclos óptimo	Ratio de aprendizaje	Error de test
Adaline	100	27	0.15	0.01964389719727566
Perceptrón multicapa	10000	10000	0.1	0.0101805914021268

Con esto se puede observar que el Adaline necesita un ratio de aprendizaje mayor que el Perceptrón Multicapa, es decir, mayor cambio en los pesos. Pero, necesita 9,973 ciclos menos para conseguir una diferencia de error de 0.0095 (0.95% más de error), algo prácticamente insignificante, y en un tiempo de cómputo mucho menor. No se ha experimentado, pero seguramente si dejáramos que el Adaline corriese durante 10,000 ciclos tendría un error menor que el Perceptrón Multicapa.

Por tanto, aunque el Perceptrón Multicapa en nuestras experimentaciones produzca un error de test menor, se podría concluir que un modelo tan sencillo como el Adaline se adecúa mejor, para este *dataset* inicial.

Práctica 1: Problema de regresión

Redes de Neuronas Artificiales