Seminario de Estadística:
Aprendizaje Estadístico Automatizado
Semestre 2021-1
Proyecto Final

Aguirre Armada Guillermo
Figueroa Torres Ivan Emiliano
Luna Gutiérrez Yanely
Ortiz Silva Ana Beatriz

8 de febrero 2021

1 Introducción

En aprendizaje supervizado buscamos establecer una relación entre las variables predictoras y la variable respuesta, lo cual nos puede permitir predecir los valores de la variable respuesta conociendo los valores de las predictoras. En un problema de clasificación, como el del presente trabajo, tenemos como propósito ajustar un modelo que reduzca el error de clasificación de los datos.
La base de datos con la que se trabajó durante este proyecto contiene el registro de 60,000 imagenes de dígitos escritos a mano. Las observaciones de esta base fueron divididas en tres subconjuntos: entrenamiento (con 30,000 observaciones), prueba (16,000 observaciones) y validación (14,000 observaciones).
Usando el conjunto de entrenamiento ajustamos tres modelos para clasificar las observaciones de acuerdo al dígito que representan y evaluamos su poder predictivo con el conjunto de prueba para finalmente realizar predicciones en el conjunto de validación.

2 Resultados

Modelo	Train	Test	Cross-Validation
DNN (97/74/94/144/62 nodos, 5.87 Epocas)	4.19	5.90	5.58
Regresión logística	4.95	9.40	9.32
Regresión logística regularizada ($\alpha = 0, \lambda = 1.83 \times 10^{-4}$)	4.77	8.86	6.93

Table 1: Tasas de error de clasificación global de los tres modelos seleccionados expresadas en porcentaje.

2.0.1 Modelo de red neuronal profunda

Para el modelo de red neuronal profunda creamos 3 mallas una con 9 modelos y las otras dos con cuatro modelos cada usando las semillas 16, 1999, 9 respectivamente, se usó un mínimo de 10 nodos y un máximo de 160. La información detalla de los modelos generados se puede encontrar en el anexo, el seleccionado es el modelo número 4, debido a que presentó el mínimo error de prueba, sin embargo fue el que presentó un mayor tiempo de procesamiento.

	Train	Test	Cross-Validation
Global	4.19	5.90	5.58
0	1.58	2.28	5.58
1	1.42	2.51	2.47
2	5.41	5.47	1.51
3	1.51	3.73	6.82
4	3.87	5.05	8.28
5	8.75	12.92	5.58
6	2.30	2.85	3.46
7	9.00	8.84	5.71
8	3.10	12.23	8.96
9	4.19	4.59	6.52

Table 2: Tasas de error de clasificación por categoría y global para el modelo de red neuronal profunda. Para *cross-validation* se usó $k = 5$. Las tasas se encuentran expresadas en porcentaje.

Notamos que este modelo presenta tasas de error bajas, existiendo la mayor confusión entre los dígitos 5 y 8.

2.1 Modelo de regresión logística no regularizado

En este modelo ajustamos una regresión logística no regularizada con la libreria $h2o$ y la función $h2o.glm()$, recordando que λ es aquella que se encarga de regularizar el modelo, dejamos el valor $\lambda = 0$ y utilizamos un $k - folds = 10$.Adjustando los errores tenemos los resultados de la siguiente tabla:

	Train	Test	Cross-Validation
Global	4.95	9.40	9.32
0	1.46	3.84	4.32
1	1.57	2.54	3.02
2	6.21	12.90	12.18
3	8.07	12.19	13.38
4	4.46	9.72	8.07
5	7.43	14.63	12.92
6	2.38	5.43	5.65
7	4.11	6.60	8.10
8	8.18	15.16	15.27
9	6.30	12.25	11.48

Table 3: Tasas error de clasificación por categoría y global para el modelo de regresión logística no regularizado con $\lambda = 0$. Las tasas están expresadas en porcentaje y se utilizó k=10 para cross-validation.

2.2 Modelo de regresión logística regularizado

Para el modelo de regresión lineal regularizado probamos con varios valores de α y decidimos usar *ridge* ($\alpha = 0$) por ser el que minimizaba la tasa de error de clasificación en el conjunto de prueba. Una vez seleccionado el valor de α, usamos la función `h2o.glm` para ajustar el modelo seleccionando el valor de $\lambda = 1.83 \times 10^{-4}$. Las tasas de error de clasificación para cada categoría con este modelo se muestran en la Tabla 2.2.

	Train	Test	Cross-Validation
Global	4.77	8.86	6.93
0	1.39	3.19	2.62
1	1.45	2.65	2.16
2	6.00	12.66	9.04
3	7.95	11.76	10.00
4	4.19	9.16	5.74
5	6.89	13.20	10.13
6	2.11	5.06	3.54
7	4.40	6.49	6.48
8	7.64	13.66	11.03
9	6.27	11.80	9.44

Table 4: Tasas de error de clasificación por categoría y global para el modelo de regresión logística regularizado con *ridge* y $\lambda = 1.83 \times 10^{-4}$. Para *cross-validation* se usó $k = 10$. Las tasas se encuentran expresadas en porcentaje.

3 Conclusión

Ahora compararemos los 3 modelos presentados anteriormente a fin de seleccionar el mejor. Podemos notar que el modelo que mejor comportamiento tuvo es el obtenido através del método DNN minimizando los errores obtenidos.

Digito	Regresion log			Regresion log reg			DNN		
	Train	Test	CV	Train	Test	CV	Train	Test	CV
Global	4.95	9.40	9.32	4.77	8.86	6.93	4.19	5.90	5.58
0	1.46	3.84	4.34	1.39	3.19	2.62	1.58	2.28	2.47
1	1.57	2.54	3.02	1.45	2.65	2.16	1.42	2.51	1.51
2	6.21	12.90	12.18	6.00	12.66	9.04	5.41	5.47	6.82
3	8.07	12.19	13.38	7.95	11.76	10.00	1.51	3.73	8.28
4	4.46	9.72	8.07	4.19	9.16	5.74	3.87	5.05	5.58
5	7.43	14.63	12.92	6.89	13.20	10.13	8.75	12.92	7.09
6	2.38	5.43	5.65	2.11	5.06	3.54	2.30	2.85	3.46
7	4.11	6.60	8.10	4.40	6.49	6.48	9.00	8.84	5.71
8	8.18	15.16	15.27	7.64	13.66	11.03	3.10	12.23	8.96
9	6.30	12.25	11.48	6.27	11.80	9.44	4.19	4.59	6.52

Table 5: Tasas de error de clasificación por categoría y global de los tres modelos seleccionados en los conjuntos entrenamiento, prueba y por validación cruzada. Las tasas se encuentran expresadas en porcentaje.

Concluimos que el modelo que mejor se predice las observaciones es el obtenido a través de una red neuronal profuna logrando obtener un error global hasta un 60% mejor al obtenido por regresión logística regularizada.

4 Anexo

Para el modelo de red neuronal profunda creamos los siguientes modelos, de los cuales elegimos el modelo número 5 (Grid 4) al ser aquel que presentaba el menor error de prueba.

	Run time	Hidden	Epochs	l1	dropout_ratio	Logloss	Logloss.1
dl1	7.380	100/100/100	20.042	0.00100	0.2	0.13473	0.1615
Grid 1	7.328	92/151/131/134	6.461	0.00046	0.0	0.09492	0.1606
Grid 2	7.353	99/101/143/152	5.974	0.00076	0.0	0.10616	0.1467
Grid 3	4.352	62/81/57/121/102	5.887	0.00059	0.0	0.09427	0.1501
Grid 4	7.973	97/74/94/144/62	8.387	0.00012	0.0	0.02679	0.1443
Grid 5	6.382	129/157/73/76	4.573	0.00068	0.0	0.14024	0.1843
Grid 6	4.529	105/57	4.927	0.00061	0.0	0.09764	0.1595
Grid 7	4.567	32/103/122/100/145	7.282	0.00066	0.0	0.12146	0.1707
Grid 8	4.675	100/49/120	4.849	0.00076	0.0	0.12529	0.1730
Grid 9	6.911	155/18/78/18/52	5.016	0.00066	0.0	0.13420	0.1895
Grid 10	6.497	115/104/117/80/70	5.056	0.00068	0.0	0.13580	0.1763

```
Grid 11    3.328           48/70/11  8.413 0.00061          0.0 0.09513    0.1784
Grid 12    7.512     148/133/127/66  4.281 0.00100          0.0 0.14103    0.1690
Grid 13    5.248         146/33/137  3.738 0.00100          0.0 0.13866    0.1695
Grid 14    2.484                 37  8.643 0.00059          0.0 0.09367    0.1717
Grid 15    5.747                118  6.348 0.00046          0.0 0.10286    0.1943
Grid 16    4.024                 88  5.887 0.00060          0.0 0.08998    0.1590
Grid 17    2.410       11/154/55/74 11.019 0.00060          0.0 0.13783    0.2120
        Logloss.2
dl1        0.1895
Grid 1     0.1673
Grid 2     0.1845
Grid 3     0.1849
Grid 4     0.1862
Grid 5     0.1877
Grid 6     0.1906
Grid 7     0.1909
Grid 8     0.1961
Grid 9     0.1961
Grid 10    0.1963
Grid 11    0.2102
Grid 12    0.2190
Grid 13    0.2258
Grid 14    0.2287
Grid 15    0.2322
Grid 16    0.2354
Grid 17    0.2467
```

A continuación presentamos la matriz de confusión del modelo elegido

```
 Confusion Matrix: Row labels: Actual class; Column labels: Predicted class
          0    1   2    3   4   5   6    7   8    9  Error           Rate
0       999    1   1    0   1   2   8    1   2    0 0.0158 =   16 / 1,015
1         0 1113   3    9   0   0   0    1   0    3 0.0142 =   16 / 1,129
2         5    3 909   24   4   3   4    4   5    0 0.0541 =     52 / 961
3         2    1   4 1041   0   1   1    3   2    2 0.0151 =   16 / 1,057
4         2    0   1    0 918   1   3    0   5   25 0.0387 =     37 / 955
5        12    0   1   44   1 845  10    0   7    6 0.0875 =     81 / 926
6         5    1   2    3   4   3 934    0   4    0 0.0230 =     22 / 956
7         5    3   2   12  10   0   0  988   2   31 0.0617 =   65 / 1,053
8         0    8   4   40   1   6   5    0 839   19 0.0900 =     83 / 922
9         9    0   1   11   4   1   0    5   2 1031 0.0310 =   33 / 1,064
Totals 1039 1130 928 1184 943 862 965 1002 868 1117 0.0419 = 421 / 10,038
```

Veremos ahora la comparación de errores entre nuestros modelos

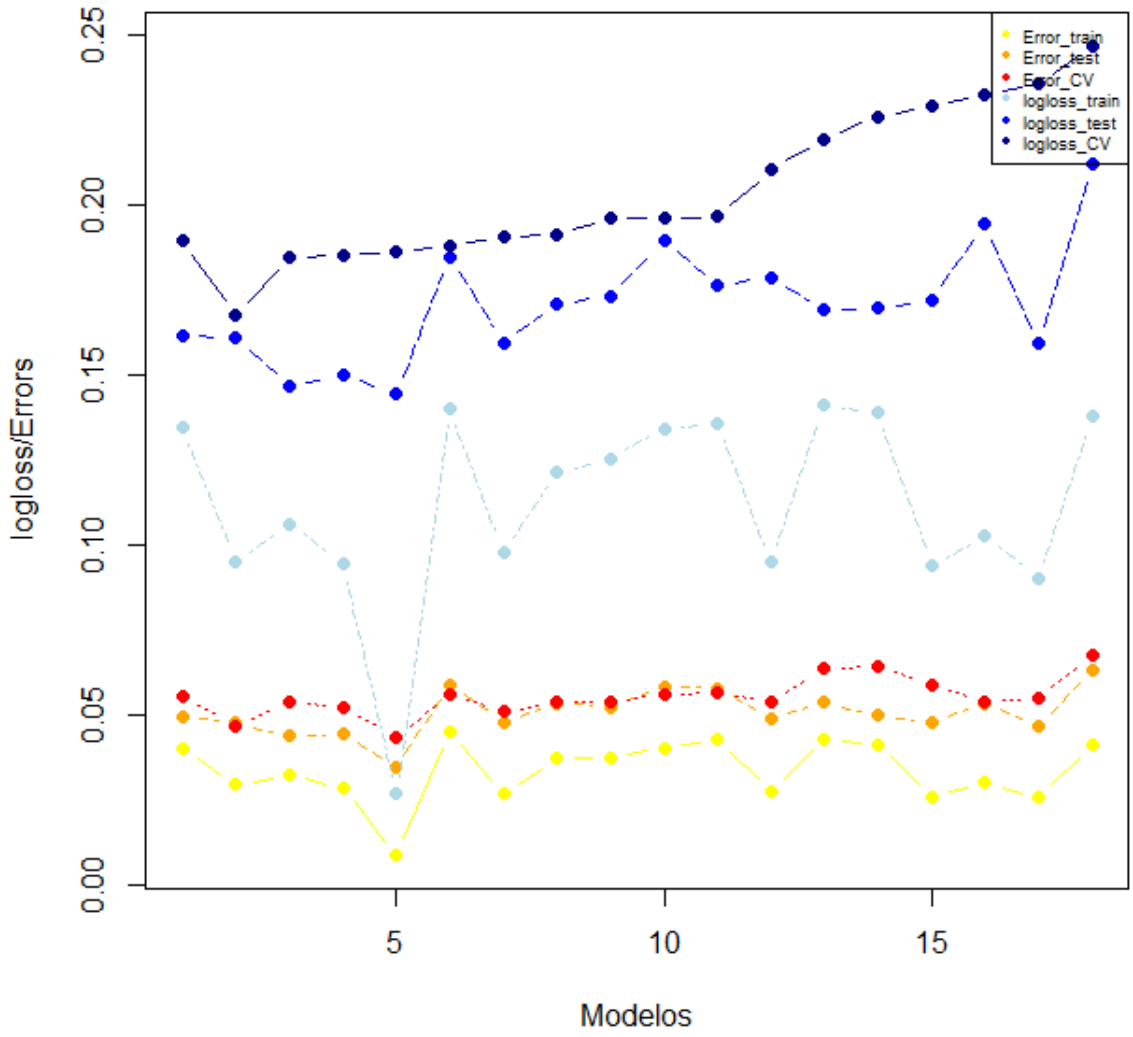

Figure 1: *Se muestran los errores obtenidos en los diferentes modelos DNN obtenids de las distintas mallas*