Regresión

Descripción del set de datos

Se trata de un conjunto de datos de la NASA que estudia superficies aerodinámicas "NACA 0012" de diferentes tamaños y las pone a prueba en un túnel de viento a varias velocidades y diferentes ángulos de ataque. Contiene un total de 6 atributos:

Entrada: Frecuencia, en hercios, ángulo de ataque (en grados), longitud de la cuerda, (en metros), velocidad de flujo libre, (m/s), espesor de desplazamiento del lado de succión (en metros).

Salida (nuestro objetivo en la regresión): Nivel de presión sonora (en decibelios).

Caracterización del set de datos

	Frecuency_Hz	Angle	Chord_length	Free_stream_velocity	Suction_side_displacement_thickness	Scaled_sound_pressure_level
count	1502.000000	1502.000000	1502.000000	1502.000000	1502.000000	1502.000000
mean	2887.769640	6.786818	0.136436	50.847137	0.011146	124.835034
std	3153.162983	5.917509	0.093471	15.569029	0.013153	6.900864
min	200.000000	0.000000	0.025400	31.700000	0.000401	103.380000
25%	800.000000	2.000000	0.050800	39.600000	0.002535	120.190000
50%	1600.000000	5.400000	0.101600	39.600000	0.004957	125.719000
75%	4000.000000	9.900000	0.228600	71.300000	0.015840	129.997750
max	20000.000000	22.200000	0.304800	71.300000	0.058411	140.987000

Dividimos los datos en un 80% de entrenamiento y 20% de prueba y ajustamos los valores medidos en diferentes escalas a una escala común

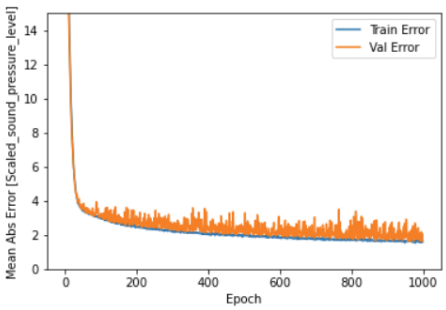
Regresión mediante algoritmo lineal

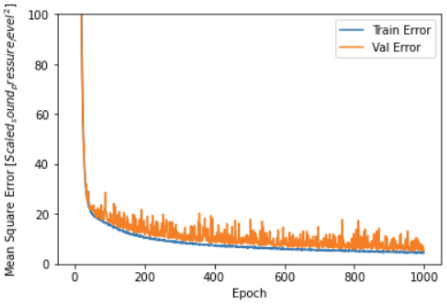
Para la construcción de este modelo vamos a utilizar el algoritmo de regresión lineal que nos aporta la librería scikit-learn. Después de entrenarlo frente a los datos de entrenamiento normalizados, obtenemos las siguientes estadísticas de predicción:

- **Puntuación**: 0.58 – Un cero sería no haber aprendido nada y con un 1 el modelo lo habría aprendido a la perfección. En este caso ha aprendido algo, pero no parece suficiente.
- **R2 Score**: 0.50 – Indica el nivel de varianza de lo predicho. el valor más correcto sería un 1, por tanto, nuestro resultado de 0.5 sería bastante pobre.
- **Error cuadrático medio - RMSE**: 4.89 - Representa a la raíz cuadrada de la distancia cuadrada promedio entre el valor real y el valor pronosticado. Los valores más bajos de RMSE indican un mejor ajuste. En nuestro caso es demasiado alto.

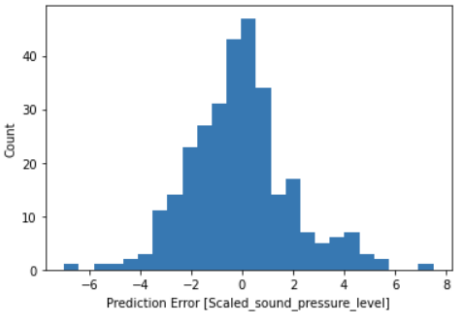
Regresión basada en redes neuronales - TensorFlow

Añadimos 2 capas intermedias densamente conectadas, activación 'relu' que aplica la función de activación de la unidad lineal rectificada, con 64 unidades y una capa de salida de 1 unidad que devuelve un único valor continuo. Para el optimizador escogemos un RMSprop de 0.001. Y para el entrenamiento escogemos 1000 épocas (es donde encontramos mejores resultados). Se trata de un modelo pequeño





Como resultado nos encontramos una media de error absoluto de +/-1.49. Que para nuestro set de datos es un valor muy bueno, es muy preciso.



En la distribución de errores podemos comprobar que siguen una función gaussiana alrededor del 0, y los mayores errores son 7 y -7, que está cerca del error medio que encontrábamos con el algoritmo lineal. Es un buen resultado.

Conclusión

Para las 2 formas de regresión que hemos utilizado, tenemos que el algoritmo lineal nos indica un RMSE de 4.89 mientras que TensorFlow nos da un error medio absoluto de 1. 49, el cual es bastante menor en comparación. Para estos datos funciona mucho mejor el método basado en redes neuronales que el del algoritmo de regresión lineal. A contin uación, podemos verlo gráficamente, donde se aprecia que los datos predichos se acerca n mucho más a la línea de aciertos con la red neuronal:

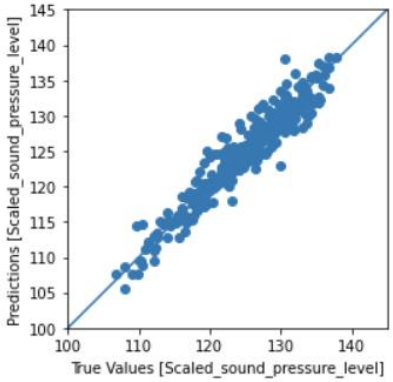


Ilustración 2 - Red Neuronal

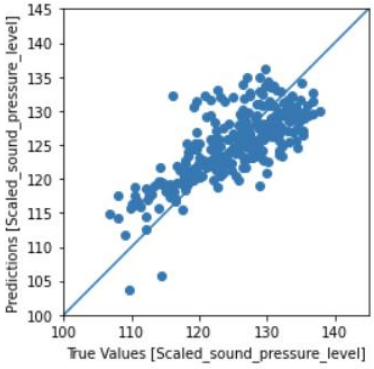


Ilustración 1 - Algoritmo Lineal

Clasificación

Descripción del set de datos

Base de datos de reconocimiento de actividad humana (HAR). Cada persona realizó seis actividades (Caminar, Caminar cuesta arriba, Caminar cuesta abajo, Sentado, De pie, Acostado) con un teléfono inteligente en la cintura.

Entrada: 561 atributos que indican en forma de vector las variables que captan los sensores (aceleración y velocidad angular)

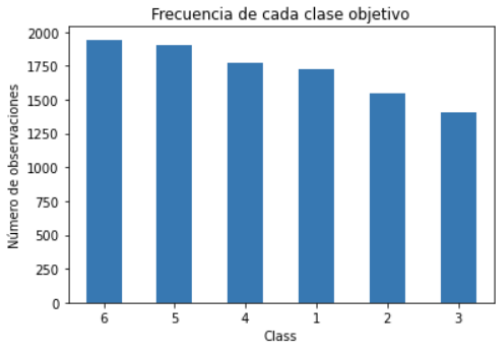
Salida (nuestro objetivo en la clasificación): Class - Variable categórica mapeada a numérica (1: Caminando, 2: Caminando cuesta arriba, 3: Caminando cuesta abajo, 4: Sentado, 5: De pie, 6: Acostado)

Caracterización del set de datos

V10	...	V553	V554	V555	V556	V557	V558	V559	V560	V561	Class
10299.000000	...	10299.000000	10299.000000	10299.000000	10299.000000	10299.000000	10299.000000	10299.000000	10299.000000	10299.000000	10299.000000
-0.466732	...	-0.298592	-0.617700	0.007705	0.002648	0.017683	-0.009219	-0.496522	0.063255	-0.054284	3.624624
0.538707	...	0.320199	0.308796	0.336591	0.447364	0.616189	0.484770	0.511158	0.305468	0.268898	1.743695
-1.000000	...	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	1.000000
-0.935788	...	-0.536174	-0.841848	-0.124694	-0.287031	-0.493108	-0.389041	-0.817287	0.002151	-0.131880	2.000000
-0.874825	...	-0.335160	-0.703402	0.008146	0.007668	0.017192	-0.007186	-0.715631	0.182028	-0.003882	4.000000
-0.014641	...	-0.113167	-0.487981	0.149006	0.291490	0.536137	0.365996	-0.521503	0.250791	0.102970	5.000000
1.000000	...	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	6.000000

Balance de datos

Comprobamos el número de observaciones que tenemos por cada clase de salida. Muchos algoritmos son muy sensibles a estas diferencias de proporción. Vemos que el número de observaciones de cada clase en nuestro set de datos está bastante balanceado, no sobresale ninguno en exceso. Más adelante veremos si tenemos algún problema en la clasificación en este sentido.



Primero comparamos varios algoritmos para ver cual es el que mejor resultados obtiene (80% de entrenamiento y 20% de prueba). El algoritmo que mejor resultados obtiene es Random Forest Classification (RFC), por tanto, es el que vamos a utilizar contra el basado en redes neuronales

```
RFC: 0.976697 (0.004907)
KNN: 0.962739 (0.004971)
NB: 0.725453 (0.026185)
SVC: 0.952301 (0.007418)
MLP: 0.192014 (0.000479)
```

Algoritmo Random Forest Classifier

```
Reporte de clasificacion:
              precision    recall  f1-score   support

         1.0       1.00      0.99      0.99       355
         2.0       0.98      0.99      0.98       304
         3.0       0.98      0.98      0.98       284
         4.0       0.98      0.95      0.97       366
         5.0       0.96      0.98      0.97       389
         6.0       1.00      1.00      1.00       362

    accuracy                           0.98      2060
   macro avg       0.98      0.98      0.98      2060
weighted avg       0.98      0.98      0.98      2060
```

```
Clasificador: RFC
precision media:  0.975724 (0.004279)
Precision de una prediccion:  0.9815533980582525

Matriz de confusion:
 [[351   1   3   0   0   0]
 [  0 301   3   0   0   0]
 [  1   6 277   0   0   0]
 [  0   0   0 348  18   0]
 [  0   0   0   6 383   0]
 [  0   0   0   0   0 362]]
```

Tenemos una precisión media de 0.976, es un valor muy bueno. Para las clases '1', '4' y '6' (caminando, sentado y acostado) tenemos una alta precisión y un menor recall, esto nos indica que el modelo es peor detectando la clase, pero cuando lo hace es muy confiable. En las clases '2', '3' y '5' (caminando cuesta arriba, caminando cuesta abajo y estando de pie) ocurriría justo lo contrario. Si observamos el valor de f1-score, tenemos que la clase que mejor se detecta es la '6' (estar de pie), y la que peor es la '4' (estar sentado). Aunque en general los valores son cercanos a 1 por lo que se puede afirmar que el modelo maneja perfectamente todas las clases.

Clasificación basada en redes neuronales – TensorFlow

Para la construcción del modelo añadimos 2 capas intermedias densamente conectadas, activación ‘relu’ que aplica la función de activación de la unidad lineal rectificada, y una capa de salida con activación ‘softmax’. Utilizamos un algoritmo de ajuste 'hyperband' que utiliza la asignación de recursos adaptativa y la detención anticipada para converger rápidamente en un modelo de alto rendimiento. Entrena una gran cantidad de modelos durante algunas épocas y lleva solo la mitad de los modelos con mejor rendimiento a la siguiente ronda. Una vez que hemos obtenido el número de épocas adecuado creamos una nueva instancia del ‘hipermodelo’ y lo entrenamos. después predecimos y evaluamos los resultados, obteniendo un ‘Accuracy’ con datos de prueba de 0.9854.

Conclusiones

Como vemos en la gráfica, la red neuronal obtiene una precisión ligeramente mayor. Y se puede considerar más efectivo, aunque la diferencia es mínima, los 2 tienen un gran acierto. Una forma de mejorar los resultados sería teniendo en cuenta sobreajustes y optimizar más los hiperparámetros.