Memoria Práctica 3

Álvaro Fernández García

Mayo 2018

---

1. Problema de Regresión

Se plantea la resolución del siguiente problema de regresión:

Se dispone de una base de datos de la NASA, la cual comprende un conjunto de pruebas aerodinámicas y acústicas de dos y tres secciones de álabes aerodinámicos realizas en un túnel de viento. Se sabe que el conjunto de datos comprende perfiles de diferentes tamaños NACA 0012 (el perfil aerodinámico en cuestión) a diferentes velocidades del túnel de viento y ángulos de ataque.

Cada ejemplo del conjunto del datos está formado por un total de 6 características: la Frecuencia en Hercios; ángulo de ataque en grados; longitud de cuerda en metros; velocidad de flujo libre en metros por segundo y espesor de desplazamiento del lado de succión también en metros.

Por último, cada ejemplo tiene una única salida: nivel de presión de sonido escalado, en decibelios.

Queda bastante claro que es un problema de regresión en el que se pretende predecir, dadas las 6 características anteriores, qué nivel de presión de sonido tendría en decibelios.

Con respecto a la base de datos, simplemente destacar que, tanto las características anteriores así como la salida, son números reales (no necesitan ningún tipo de categorización). Por su parte, con respecto al tamaño de la muestra, no es que sea especialmente pequeño, pero ni mucho menos es grande. Diremos que es un tamaño pequeño-moderado, con solo 1503 ejemplos.

Una vez comprendido el problema que queremos resolver, así como las características que tiene la muestra, debemos plantearnos qué clase de funciones vamos a probar para intentar estimar nuestro modelo. Como se trata de un problema de regresión, dentro de los modelos líneales tenemos los siguientes:

- Regresión Lineal.
- Gradiente Descendente Estocástico (para regresión).
- Lasso.
- Ridge.

Debido a que se trata de un problema en el que la mayoría de los datos están obtenidos a través de sensores, es prácticamente seguro que vamos a encontrar ruido en las mediciones, por tanto puede ser bastante interesante aplicar regularización para lidiar con ello (contribuyendo así a intentar evitar que se produzca sobreajuste).

La Regresión Lineal por defecto no aplica regularización, por lo queda descartada. Al Gradiente Descendente Estocástico hay que especificarle una función de pérdida y normalmente se utiliza como optimización de otros modelos, por lo que tampoco es de utilidad.

Esto nos deja con dos candidatos: Lasso y Ridge. Lasso es una Regresión Lineal con penalización (regularización) L1 ($\lambda(\sum_{i=1}^{N} |w_i|)$), que nos permite eliminar aquellas características de los ejemplos de la muestra que no sean relevantes para la predicción. Realmente los ejemplos tienen únicamente 6 características, un número bastante pequeño. No parece necesario aplicar este tipo de regularización, por lo que podemos descartar este modelo.

Con esto nos queda un solo modelo: Ridge. Ridge es una regresión Lineal con penalización L2 ($\lambda(\sum_{i=1}^{N} w_i^2)$), que reduce la importacia de aquellas características que no influyen demasiado en las predicciones (pero a diferencia de Lasso, no las hace nulas). Realiza regularización y es razonable para nuestro problema, por lo que este será el modelo que probaremos.

Puesto que vamos a aplicar un modelo con regularización L2, será necesario estandarizar las características primero. Los modelos lineales regularizados asumen que todas las características están centradas alrededor de 0 y tienen varianza en el mismo orden (normalmente una varianza de 1). Si una característica tiene una varianza que es de un órden de magnitud mayor que otras, podría dominar la función objetivo y hacer que el estimador no pueda aprender de otras características como se esperaba. Por tanto, el primer paso será preprocesar los datos de la muestra, para dejarlos según estos requisitos.

El siguiente paso será decidir cuantos datos de la muestra elegimos para Train y cuantos para Test. Como la muestra es de tamaño moderado, elegiremos un 20 % de los datos para Test (301 ejemplos) y un 80 % para Train (1202). La partición se realizará de forma aleatoria.

Ridge requiere un hiperparámetro (λ), por lo que emplearemos validación cruzada para seleccionarlo. La validación cruzada se ha realizado sobre el conjunto de entrenamiento con un total de 5 particiones (nuevamente, como el tamaño de la muesta es moderado nos quedaremos con el 20 % para validación, lo que nos da un total de 5 particiones aproximadamente), y probaremos como hiperparámetros distintos órdenes de magnitud que van desde 10^{-4} hasta 10^4, multiplicando por 10 en cada paso y obviando el 0. (Tamaños mayores y menores no aportan nada al ajuste, además de que incluyen más coste computacional en los cálculos).

De todos los hiperparámetros probados con la validación cruzada nos quedaremos con aquel que nos de un mayor Score. Pero claro, para ello también tenemos que definir que métrica utilizaremos para medir la bondad de los modelos. En este caso, como se trata de un problema de regresión he optado por elegir el coeficiente de determinación R^2 (ya que muestra porcentajes y me prace más sencillo de interpretar que el error cuadrático por ejemplo), el cual nos da un valor entre $-\infty$ y 1 (donde 1 es el mejor score posible), que determina el porcentaje de información (concretamente la varianza), que el modelo está explicando sobre los datos (valores negativos indican un ajuste pésimo). En consecuencia, tendremos que buscar aquel λ que en la validación cruzada nos de el score más cercano a 1 posible.

Una vez realizada la validación cruzada, y elegido el λ con el mejor score posible para Ridge, es hora de entrenar el modelo con ese hiperparámetro y todos los datos de train (hay que tener en cuenta que en validación no estábamos entrenando con todos los datos de train, estabamos entrenando con $\frac{4}{5}$ de los datos, dejando $\frac{1}{5}$ para validar. El mejor ajuste posible para train solo lo conseguiremos si entrenamos con todos los datos disponibles, por eso, una vez determinado el mejor hiperparámetro, debemos entrenar nuevamente el modelo completo).

Tras entrenar el modelo con el λ determinado, es hora de calcular el E_{in}, es decir, el coeficiente de

determinación R^2 para los datos de train. Sin embargo, nos llevamos una sorpresa: el coeficiente R^2 sobre los datos de la muestra estandarizados, y el con el mejor λ encontrado, nos da un score de 0.55 aproximadamente en train. Eso quiere decir que el modelo ajustado explica únicamente un 55 % de la varianza de los datos, y los puntos intermedios nunca son buenos (si tenemos un modelo que no se equivoca nunca, es un buen modelo, si se equivoca siempre, también es un buen modelo, solo hay que predecir lo contrario, pero un punto intermedio nunca es bueno).

Una de las posibles causas de este mal ajuste puede ser que los datos no sean explicables por una línea recta (el equivalente a que no sean linealmente separables en un problema de clasificación). La solución es probar a añadir a los datos características polinomiales.

Ahora tenemos una nueva cuestión que resolver: ¿hasta que grado le añado a las características? ¿Grado 2? ¿4?. Para responder a esta pregunta recurriremos nuevamente a la validación cruzada. En este caso debemos de fijar dos hiperparámetros: el λ de la regularización y también el grado del polinomio. Además debe de extraerse la mejor combinación de ambos.

Para el grado del polinomio probaremos valores de 2 a 8, 2 porque es el mínimo grado posible y hasta 8 porque con 8 tendremos que cada ejemplo tiene un total de 40 características (y tampoco interesa complicar más el modelo, ya que para obtener un buen ajuste en un modelo complejo, según la teoría, necesitamos un tamaño de muestra grande, cosa que no tenemos).

Realizamos la validación cruzada (con el 20 % para validación y el mismo rango que antes para λ) y para cada grado de polinomio, probamos cual es el mejor λ posible (como siempre, los mejores hiperámetros serán los de mejor score). Tras los cálculos, se obtiene que el mejor grado de polinomio es añadir hasta grado 4, con un $\lambda = 10$.

Una vez fijados amnbos hiperparámetros volvemos a calcular el E_{in} (R^2 para train) y en este caso obtenemos un coeficiente de 0.85.

Según los resultado, podemos ver que, efectivamente, los datos no eran explicables por una línea recta, y que al añadir las características polinomiales hemos conseguido mejorar considerablemente la calidad del ajuste, de hecho, tenemos que el nuevo ajuste con las características polinomiales consigue explicar un 85 % de la varianza de los datos. ¿Es un ajuste razonable? Teniendo en cuenta que como ya hemos dicho antes los datos proporcionados han sido medidos a través de sensores y estos siempre tienen ruido, es bastante improbable que seamos capaces de conseguir un ajuste mejor, debido a la gran cantidad de ruido estocástico. Por tanto la respuesta es que sí, es un ajuste razonable.

Para terminar, una vez que nos hemos convencido de que nuestro ajuste es razonablemente bueno, es hora de comprobar cómo se comporta el modelo para los datos de test. Calculando el R^2, obtenemos un E_{out} de 0.77, es decir, explica un 77 % de la varianza de los datos. Obviamente es peor que E_{in} pero esto es de esperar, aún así son muy cercanos (solo los separa un 8 %), por lo que parece que no se ha producido mucho sobreajuste y continúa siendo un ajuste perfectamente razonable para este problema.

2. Problema de Clasificación

Se plantea ahora la resolución de un segundo problema, en este caso de clasificación:

Se dispone de una base de datos que contiene información sobre mapas de bits normalizados de dígitos escritos a mano. Los mapas de bits 32×32 se dividen en bloques no superpuestos de 4×4 y el número de píxeles se cuenta en cada bloque. Esto genera una matriz de entrada de 8×8 donde cada elemento es un entero en el rango $0 \ldots 16$. Los dígitos escritos a manos van desde 0 hasta 9.

Queda bastante claro que se trata de un problema de clasificación multietiqueta, en el que dadas las 64 características de entrada ($8 \times 8 = 64$), se predende predecir que etiqueta (número) le corresponde, con $y \in \{0, \ldots, 9\}$. A fin de cuentas es un problema de reconocimiento óptico de números escritos a mano.

Con respecto a la base de datos, simplemente destacar que, tanto las características anteriores así como la salida, son números enteros (por lo que al igual que en el problema anterior, no necesitan ningún tipo de categorización). Por su parte, con respecto al tamaño de la muestra, diremos que es un tamaño moderado-grande, con un total de 5620 ejemplos. Además, en este problema no vamos a tener que preocuparnos por la división de los datos en train y test, ya que se nos proporcionan los datos divididos. Concretamente, utilizaremos 3823 ejemplos para el train, y 1797 para test (un poco más del 30 % de los datos para test. Como el tamaño es moderado tirando a grande, podemos permitirnos elegir un poco más del 20 %).

Una vez comprendido el problema que queremos resolver, así como las características que tiene la muestra, debemos plantearnos qué clase de funciones vamos a probar para intentar estimar nuestro modelo. Como se trata de un problema de clasificación, dentro de los modelos líneales tenemos los siguientes:

- Gradiente Descendente Estocástico (para clasificación).
- Perceptrón.
- Regresión Logística.

Como ya hemos dicho antes, al Gradiente Descendente Estocástico hay que especificarle una función de pérdida y normalmente se utiliza como optimización de otros modelos, por lo que en principio podemos descartarlo. El Perceptrón originalmente está pensado para el caso de clasificación binaria, y extenderlo a multiclase puede ser complicado, así que también podemos descartarlo. Esto nos deja un solo candidato: la Regresión Logística, que puede aplicarse fácilmente a problemas multietiqueta y además tiene la ventaja de que nos permite aplicar tanto regularización L1 como L2 en caso de que fuese necesario.

¿Es necesario aplicar regularización en este problema? Bueno, a diferencia del anterior ya no tenemos únicamente 6 características, si no que tenemos 64. Es un número relativamente elevado, por lo que podría ser interesante aplicar regularización L1 para eliminar aquellas características irrelevantes, y ya que estamos comprobaremos también como se comporta el modelo si aplicamos regularización L2, que no elimina las características relevantes pero si disminuye su importancia. En resumen probaremos dos modelos: Regresión Logística con penalización L1 y Regresión Logística con penalización L2.

Como estamos probando modelos lineales con regularización, recordemos que será necesario estandarizar las características primero. Se aplica el mismo preprocesado que en el problema anterior,

con un pequeño paso antes, y es que como todas las variables son enteras, deben de castearse a reales.

Ambas penalizaciones tienen un hiperparámetro λ que debemos de estimar. Para estimarlo se ha utilizado un método llamado "Grid Search", el cual va a realizar validación cruzada para encontrar el mejor hiperparámetro tanto para L1 como para L2, a continuación comparará ambos Scores y devolverá aquella penalización junto con su λ asociado con mejor métrica. Con esta sencilla herramienta podremos saber cual de los modelos es más apropiado para nuestro problema, además de estimar el mejor hiperparámetro al mismo tiempo.

Para la validación cruzada, al igual que en el problema anterior, se han realizado un total de 5 divisiones (he decidido mantener la proporción 80-20), y como hiperparámetros también se probarán los mismos que antes por los mismos motivos (de 10^{-4} hasta 10^4 multiplicando por 10 y obviando el 0).

Con respecto al Score que se aplicará tanto para la validación cruzada como para estimar el E_{in} y E_{out} se utilizará el "accuracy", que no es más que el número de ejemplos bien clasificados partido el número de ejemplos totales. Esto nos dará el porcentaje del número de ejemplos que el modelo logra clasificar correctamente, lo cual me parece bastante explicativo e intuitivo para el caso de clasificación.

Una ver realizada la "Grid Search", se tiene que la mejor penalización para este modelo es, sorprendentemente, la L2, con un $\lambda = 10$. Pese a lo que pensábamos no resulta de mucha utilidad eliminar las características irrelevantes y merece más la pena aplicar el segundo tipo de regularización. No obstante, también he probado a ver cuantas carácterísticas elimina L1, y solo considera irrelevantes 5 características, por lo que no merece la pena.

Una vez que tenemos nuestro modelo de Regresión Logística con penalización L2 y $\lambda = 10$, debemos de entrenarlo nuevamente con todos los datos train, (por lo dicho en el problema anterior, sólo podremos conseguir el mejor ajuste con los datos proporcionados si entrenamos con todos ellos, cosa que no hacíamos en la validación cruzada).

Calculando el porcentaje de aciertos en el train (E_{in}) obtenemos un 98.45 % de aciertos. Es un error muy bueno, quiere decir que estamos clasificando bien el 98 % de los datos, es decir, prácticamente todos los datos. Para ver un poco de forma más detallada que errores y aciertos está cometiendo también se muestra la matriz de confusión:

$$\begin{pmatrix} 375 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 377 & 1 & 0 & 0 & 0 & 1 & 1 & 7 & 2 \\ 0 & 0 & 379 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 383 & 0 & 2 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 386 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 372 & 0 & 0 & 0 & 3 \\ 0 & 1 & 0 & 0 & 1 & 0 & 375 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 387 & 0 & 0 \\ 1 & 7 & 1 & 0 & 4 & 0 & 1 & 0 & 365 & 1 \\ 1 & 4 & 0 & 4 & 1 & 1 & 0 & 0 & 6 & 365 \end{pmatrix}$$

Como vemos, la diagonal principal tiene bastante datos bien clasificados, y solo unos pocos errores esparcidos por el resto, de hecho, lo que más le cuesta distiguir son los 8 de los 1 con 7 errores.

En definitiva el ajuste es bastante bueno, por tanto, como nos convence, es el momento de ver cómo se comporta para los datos que no ha visto en el ajuste, por lo que calcularemos el error y la matriz

de confusión para el conjunto de test:

$$\text{Score } (E_{out}) = 94.64\,\%$$

$$\begin{pmatrix} 176 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 170 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 7 \\ 0 & 1 & 172 & 2 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 2 & 174 & 0 & 3 & 0 & 1 & 0 & 3 \\ 0 & 2 & 0 & 0 & 174 & 0 & 0 & 1 & 1 & 3 \\ 0 & 0 & 1 & 2 & 0 & 176 & 1 & 0 & 0 & 2 \\ 0 & 2 & 0 & 0 & 2 & 0 & 176 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 9 & 0 & 165 & 0 & 4 \\ 0 & 11 & 0 & 3 & 0 & 2 & 1 & 0 & 150 & 7 \\ 1 & 2 & 0 & 3 & 2 & 2 & 0 & 0 & 2 & 168 \end{pmatrix}$$

Como era de esperar, el porcentaje de aciertos en test es menor que train (un 4 % menor), pero no es mucho menos, lo cual nos indica que no se ha producido sobreajuste y que el modelo continúa siendo bastante bueno para los datos no vistos en el entrenamiento. Con respecto a la matriz de confusión, la diagonal principal muestra que hay bastantes aciertos y nuevamente le cuesta distinguir el 8 del 1 con 11 errores.