Práctica 3 de Aprendizaje Automático

UNIVERSIDAD DE GRANADA

Antonio Manuel Fresneda Rodríguez

antoniomfr@correo.ugr.es

Sábado, 19 de mayo de 2018

Índice

1 Introducción

Para la realización de la práctica he usado Python con la librería Scikit-learn. Las dos bases de datos son:

- **Airfoil self noise**: Base de datos de la NASA en la que se han obtenido de una serie de pruebas aerodinámicas y test acústicos en un túnel de viento para alas de avión. Cada muestra obtenida tiene cinco características:
  - **Frequency,** en hercios.
  - **Angle of attack** en grados.
  - **Chord length** en metros.
  - **Free-stream velocity** en metros por segundo.
  - **Suction side displacement thickness** en metros.

  La variable de salida es el nivel de presión sonora medida en decibelios. Este dataset está indicado para realizar regresión, puesto que la salida es un valor real.
- **Optical Recognition of Handwritten Digits Data Set**: Base de datos en la que se han extraído mapa de bits de 32x32 bits dígitos escritos a mano de un total de 43 personas. Este mapa de bits se ha dividido en bloques de 4x4 y se cuentan el numero de pixeles. Esto genera una matriz de 8x8 donde cada elemento es un entero en el rango [0,16] La variable de salida es un entero en el rango [0,9] e indica que número es. Este dataset está indicado para realizar clasificación, ya que la salida representa el número que se ha escrito.

2 Preprocesado

Para el preprocesado de datos he usado dos tipos:

- **Normalización** En este caso, la entrada se normaliza en un rango definido por:
$$X_{normalized} = \frac{X - X_{min}}{X_{max} - X_{min}}$$
El problema que tiene es que al comprimir los datos en un rango, puede provocar un aumento del ruido dentro de las variables.

- **Escalado estándar** Si usamos este método, a cada entrada se le resta la media de la variable y se divide por la desviación típica:
$$X_{normalized} = \frac{X - X_{mean}}{X_{stddev}}$$

- **Aumento de características** A veces es útil añadir complejidad al modelo y considerar datos no lineales de entrada. Para ello, lo usamos para obtener nuevos datos usando un polinomio.

- **División en entrenamiento, validación y test** Dividimos el dataset que tenemos en tres conjuntos. Si usamos validación cruzada, tenemos que dividir el entrenamiento en entrenamiento y validación, vemos que parámetros son los que mejores resultados nos dan en **validación** y nos quedamos con los mejores. Una vez escogidos los parámetros, entrenemos el modelo con el conjunto de entrenamiento entero y probamos los resultados en test

3 Ajuste de hiperparámetros

Para decidir que parámetros voy a tener en cuenta he usado la función de Scikit-Learn gridSearch. Esta función tiene como parámetros:

- **Param grid**: Un diccionario (o una serie de diccionarios en la lista) en la que la clave es un string el parámetro que queremos probar y como valor para la clave una lista con los distintos valores para el parámetro.

- **Estimator**: Es un modelo (regresión ó clasificación) implementado en sk-learn.

- **CV**: En este parámetro se especifica el número de particiones que se van a usar para realizar una cross-validation con los datos que se han pasado (lo hace internamente).

- **Scoring**: Con este parámetro indicamos que tipo de métrica queremos que use para ver que parámetro es mejor o peor. Hay que destacar que internamente intenta maximizar el resultado de la métrica. En clasificación

no hay problema, pero a la hora de realizar la regresión no hay una métrica de error. Para resolver esto, internamente cambia el signo de la métrica y así nos devolverá el mejor hiperparámetro.

Una vez hecho esto, nos quedamos con el modelo que mejores resultados nos ha dado.

4 Modelos

Voy a comenzar comentando la necesidad de regularización. La regularización se encarga de que el modelo no se ajuste demasiado a los datos y provoque sobreaprendizaje (nos funciona muy bien en entrenamiento pero en test no acierta ninguna). En una caso ideal en el que no tuviésemos ruido este no seria un problema puesto que la medición de la muestra seria perfecta. Pero en el mundo real todas las muestras que podamos obtener van a tener ruido (en nuestro caso, ambas bases de datos van a tener ruido) a si que es **vital** el uso de regularización puesto que corremos el riesgo de que nuestro modelo se ajuste tanto a los datos que también ajuste el error estocástico y provoque que el modelo no funcione bien.

Una vez discutido esto, podemos pasar a la elección del modelo. En esta práctica solo podemos usar modelos lineales así que se han pensado las siguientes opciones:

- **Clasificación**: Ahora voy a enumerar los distintos modelos que he pensado en usar para clasificación.
  - **Mínimos Cuadrados**: Las ventajas del uso de este modelo es que es muy simple. La desventaja que tiene es que solo nos dice a que clase pertenece una muestra pero no como de seguro está el modelo de que pertenezca a la misma.
  - **Regresión Logística**: Este modelo ya es más complejo que el anterior. Este modelo nos devuelve la probabilidad (un valor entre 0 y 1 si usamos la función sigmoide) de que una muestra pertenezca a una clase. Entonces no solo nos dice a que pertenece una muestra, si no que también nos dice la seguridad de que pertenezca a la clase.
- **Regresion**:
  - **Lasso**: Esta es la forma en que se calcula la penalización:

$$\sum_{q=0}^{Q} |w_q| <= C$$

    Si usamos esta, obligamos a que algunos valores sean 0. Esto significa que Lasso es muy buena para la selección de características.

- **Ridge**: Esta es la forma en la que se calcula la penalización:

$$\sum_{q=0}^{Q} w_q^2 <= C$$

  Esta penalización no obliga a que los valores sean 0, si no que tienen valores cercanos a 0 pero nunca llegando a cero. Este tipo de penalización puede ser mejor que Lasso, ya que tiene en cuenta (aunque sea muy poco) algunas características y eso puede llevar en algunos casos a un mejor ajuste.

5 Ajuste de los modelos

El preprocesado que he usado para la base de datos de clasificación ha sido estandarizado, ya que no necesitaba que los datos estuviesen en el rango [0,1] y además por lo que he comentado previamente.

Para el ajuste de hiperparámetros en la grid search he usado los siguientes valores:

- **Clasificación**: Los parámetros que tenemos que ajustar son:
  - **C**: Inversa de la fuerza que aplicamos a la regularización. Para el valor he escogido entre el rango [0.1,0.9]. Se ha elegido con el objetivo de no aumentar mucho la fuerza de la regularización, ya que puede estropear el ajuste.
  - **Penalty**: Penalización por la complejidad de la clase de funciones. L1 es Lasso y L2 es Ridge. He probado las dos con el objetivo de ver cual es la diferencia entre seleccionar características (usando Lasso) o tenerlas en cuenta.
- **Regresión**: Los parámetros que tenemos que ajustar son:
  - α: Parámetro que acompaña al termino de regularización. El rango en que he escogido es [0.01,0.9]. Este rango se ha escogido para ver como afecta en el ajuste una mayor regularización frente a una menor regularización.

He usado Lasso para seleccionar que características usar para luego usar Ridge, pero la diferencia entre usar Lasso y luego Ridge no es significativa si la comparamos con usar solo Ridge. Esto se puede explicar debido a que el dataset no tiene muchas características.

6 Resultados y conclusiones

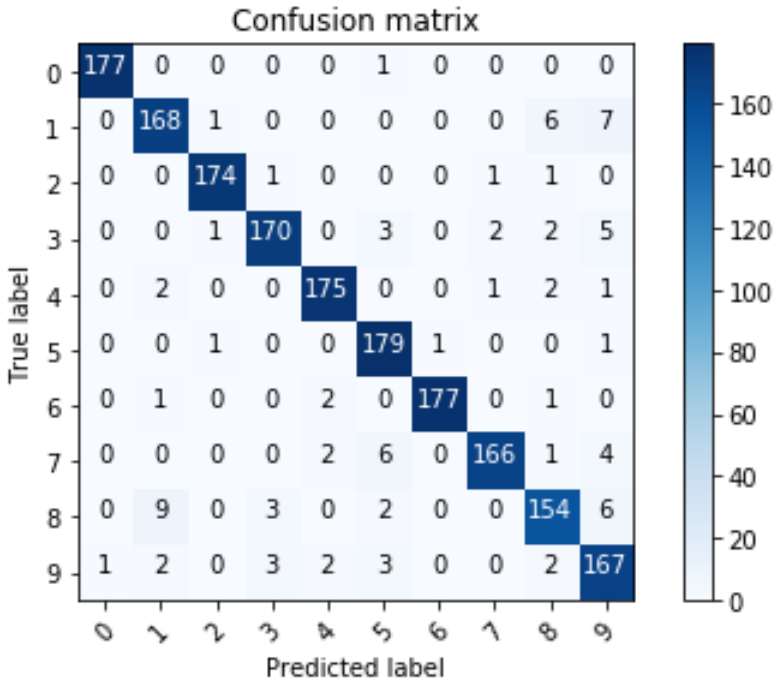
Tengo que destacar que para regresión he ampliado el numero de características de forma polinomial. He decidido esto debido a que los datos no son linealmente separables y me salia un error bastante alto (50% de error). He ido incrementando el grado del polinomio hasta el punto donde en test comenzaba a fallar más. Finalmente he dejado el grado del polinomio con un valor de 6.

Para ver como se han comportado los distintos modelos he usado las siguientes métricas:

- **Clasificación**: A la hora de seleccionar los mejores parámetros he usado la métrica accuracy y con las pruebas del test he usado la matriz de confusión. También he generado un clasification-report. Usando esta función obtenemos las métricas precission,recall y f1-score para cada clase usando el test. La columna support es el numero de ocurrencias de cada clase.
- **Regresión**: A la hora de seleccionar los mejores parametros he usado neg MSE y para ver el acierto en test he usado la métrica r^2.

Los resultados obtenidos han sido los siguientes:

- **Clasificación**: Los mejores parámetros han sido: C=0.9 y penalty=l1 con un valor de accuracy de 0.963. Moviéndonos a test, tenemos una f1-score de 0.95 y obtenemos también la siguiente matriz de confusión.



Estos resultados indican que el ajuste ha sido bastante bueno y que el modelo predice muy bien los datos. Cabe destacar que la penalización que mejor resultado a dado ha sido Lasso. Seguramente sea por que haya bastantes características que no influyan para nada en el ajuste (ruido).

- **Regresión**: El mejor valor para α en Lasso ha sido de 0.01 con un valor de neg MSE de -11.828. Para Ridge el mejor valor para α en Ridge ha sido de 0.5 con un valor de neg MSE de -10.51.

  En este punto intente pasando los coeficientes de Lasso a Ridge para que eliminase características innecesarias, esto provocaba que el error en test aumentase por lo que finalmente me decidí por usar solo Ridge. Los resultados en test para han sido los siguientes:

  - r^2=0.87

  Como vemos, los resultados en regresión también son buenos. Si nos fijamos en r^2 (coeficiente de determinación) podemos decir que hemos explicado el 87% de la varianza, por lo que podemos decir que el ajuste es bueno.