Trabajo 3 Programación BBDD 1

Víctor Manuel Arroyo Martín
Aprendizaje Automático

1 Comprender el problema a resolver. Identificar los elementos X, Y and f del problema.

Siguiendo la información que aporta la base de datos en el archivo optdigits.name, tenemos que cada dato es un conjunto de 64 valores que representan la suma de unos (pixel en negro) de cada pequeña región de la imagen. Estas regiones se sacan de un bitmap de 32x32 dividido en regiones de 4x4, quedando 8x8 regiones en total en la imagen, de ahí que sean 64+1 atributos. El último atributo representa el número que está escrito.

Así, los elementos X son los 5620 datos de los 64 primeros enteros y la Y, la variable de clase, es el último valor que decíamos que representaba el número, es decir, la etiqueta, el atributo de clase.
Se trata por tanto de un problema de aprendizaje supervisado pues tenemos las etiquetas reales de cada dato para entrenar. También, es un problema de clasificación pues ante un bitmap nuevo manuscrito, nuestro objetivo es aproximar la función f clasificatoria que nos diga qué numero está representando ese dato.

2 Selección de las clase/s de funciones a usar. Identificar cuáles y porqué.

Usaré las clases de funciones lineal y cuadrática, la lineal es siempre recomendable usarla al principio ya que es la clase de funciones más simple. La cuadrática también la uso pues es rápida de calcular, a diferencia de poilinomios de mayor grado o funciones de otro tipo, y abre la posibilidad de ajustar mejor que el modelo lineal la base de datos.

3 Fijar conjuntos de training y test que sean coherentes.

Los conjunto de train y test ya vienen divididos en un 70 y 30% respectivamente y con el de train he hecho crossvalidation dividiéndolo en 5. Esto quiere decir que se compararán los modelos con cada uno de los hiperparámetros que le indiquemos, lo cual lleva a 1 minuto de ejecución pero resultados más precisos.

4 Preprocesado los datos

Los datos obtenidos de la base de datos ya han sido preprocesados para disminuir la dimensionalidad de los datos de 32x32 a 8x8 tal y como expliqué en la sección 1.

Para el preprocesado de datos también he añadido las siguientes funciones de sklearn:

-Variancethreshold: Con el parámtero 0.01 indico que quiero eliminar las variables que tengan menos de esa varianza ya que una varianza tan baja no va a aportar nada en el aprendizaje de nuestra función ibjetivo.

-PolynomialFeatures: para generar un vector w de pesos con forma cuadrática.

-StandardScaler: Estandariza las características eliminando la media y escalando a la varianza unidad. Si u es la media y s la desviación típica de nuestros datos de entrenamiento, standardScaler transforma un dato x en

$$z=\frac{(x-u)}{s}$$

También en una segunda versión de preprocesado, he incuido a las anteriores tectinas Lasso que puede actuar como selector de características o como regularización. Para el preprocesado lo he usado con su primera versión.

5 Fijar la métrica de error a usar. Discutir su idoneidad para el problema.

Para la métrica he usado Accuracy (precisión) pues lo que fomenta esta métrica es el clasificado correcto. En general, lo que queremos es que cuando se detecte un número, se lea bien el que es.

6 Discutir la técnica de ajuste elegida.

Ya que sklearn da tantas posibilidades, he usado modelos con técnica SGD y modelos con pseudoinversa, pues de los que hemos dado, se ajustan a cualquier situación y dan buenos resultados en poco tiempo lo cual es idóneo para una base de datos extensa como esta.

7 Discutir la necesidad de regularización y en su caso la justificar la función usada para ello.

Debido a que muchos modelos (sobre todo polinómicos) tienden al overfitting, es necesario usar técnicas de regularización para que el E_{test} y E_{out} no sean excesivamente grandes en comparación al pequeño E_{in} que tiene el modelo cuando ocurre overfitting. He usado dos técnicas de regularización que sklearn permite y he seleccionado la mejor al entrenar los

modelos:
- Ridge (l2): Esta técnica se centra en hacer pequeños los pesos que causan el sobreajuste aunque no los elimina del todo. Lo hace añadiendo a la función de ajuste de w en cada paso una constante λ multiplicada por la suma de los cuadrados de los pesos. Esto lleva a que la varianza baje y suba el sesgo.

- Lasso (l1): Al igual que en la regresión de Ridge, el lasso también reduce los coeficientes estimados a cero, pero la penalización hará que los coeficientes sean igual a cero si el parámetro de ajuste λ es lo suficientemente grande. Esto lo hace sumando el prámetro por el valor absoluto de los pesos en cada ajuste de w.

8 Identificar los modelos a usar.

Como modelos he decidido usar los siguientes:
- Regresión Logística: Lo he usado por sus ventajas, la salida LGR puede considerarse como una probabilidad de clasificación y nos permite cierta flexibilidad para asignar muestras a las etiquetas. En general se puede decir que es un buen clasificador.
- Perceptrón + pocket: el perceptron de sklearn usa también pocket con lo que es posible usarlo en esta base de datos ya que sabemos que el algoritmo acabará. A persar de que no es muy eficiente en tiempo, el hecho de usar pocket asegura que encontraremos una buena solución, más aún si le ponemos una tolerancia baja (en este caso por defecto está a 1e-3).
- SGD Clasiffier: Son un conjunto de modelos que usan SGD como técnica de ajuste. Hay modelos como SVM que usa los vectores de soporte para crear un hiperplano que separe la muestra, regresión logística etc.

9 Estimación de hiperparámetros y selección del mejor modelo.

En regresión logística he usado los siguientes parámetros:
- L2 o lasso para regresión pues los parámtros lbfgs y newton-c sólo soportan este tipo.
- C: Es la inversa de la fuerza de refularización, cuanto más pequeño es su valor (siempre positivo) más afecta la regularización al modelo. Por ello, he usado 5 valores repartidos uniformemente en la escala -4, 4 logarítmica, para tener varios valores pequeños con los que comparar en crossvalidation.
- El parámetro solver es el que indica qué algoritmo a utilizar en el problema de optimización. Para una base de datos multiclase como esta, sklearn recomienda usar el método de newton y lbfgs (que es el método de newton mejorado en tiempo).
- Poly dregree: Indica el grado del polinomio que estamos usando para aproximar, he puesto las opciones 1 y 2.

Para perceptron he usado:
- penalty: esta vez no solo tenemos l2 lasso sino también l1 ridge, para dar más opciones a la hora de comparar modelos.

- alpha: es equivalente a C de regresión logística y he usado un rango de números más pequeños para asegurar que no hay sobreajuste.

Para SDGClasiffier:
- Como funciones de pérdida he puesto hinge que es la de SVM y log que es la de regresión logística.No he puesto más porque tarda bastante en ejecutarse y ya hay bastantes opciones.
- Como en perceptron, he usado l1 y l2 de regresión.
Como mejor modelo, he seleccionado regresión logística pues es el que menor error me daba tanto en el crossvalidation como en el train y test. En concreto, es el mejor con las opciones C=10000, 500 iteraciones máximo, newton-cg como solver y tolerancia de 0'0001. En el siguiente apartado hablaré de los errores.
Con lasso en preprocesamiento, he obtenido también regression logística pero esta vez con C=1, l2 de regresión y lbfgs.

10 Estimación por validación cruzada del error E out del modelo. Compárela con E test , ¿que conclusiones obtiene?

Por validación cruzada, la precisión media que devuelve sklearn es de un 98.300% siendo en el conjunto completo de training de un 100%, esto quiere decir que se ajusta muy bien a la muestra. De hecho la precisión para el cojnunto test da 96'828%, lo que quiere decir que los errores en las predicciones serán muy bajos.
Al añadir lasso en el preprocesamiento de datos empeora un poco ya que lasso lo que hace es eliminar algunas características que influyen poco en el aprendizaje. Aún con ello, las precisiones son:
Accuracy en CV: 96.312%
Accuracy en training: 99.738%
Accuracy en test: 93.489%
También muy buenas.

11 Suponga que Ud ha sido encargado de realizar este ajuste para una empresa. ¿Qué modelo les propondría y que error E out les diría que tiene?. Justifique las decisiones.

Les propondría el modelo mencionado en el apartado 9, pues es el que mejores resultados ha lanzado. Para justificarlo, diría que he hecho uso de las opciones de sklearn y he comparado varios modelos con múltiples hiperparámetros y varios preprocesamientos de los datos.
No diría que mi modelo es el mejor de todos los posibles porque eso es imposible, pero haría énfasis en la alta precisión y el bajo error que da este modelo. Para el E_{out} usaría el E_{test} pues del conjunto test no tiene información el modelo y se comporta como un conjunto de

fuera de la muestra de entrenamiento.
Añadiría que a la vista de los resultados y de la gran variedad de modelos comparados mediante crossvalidation, el modelo resultante se ajusta fielmente a los datos.