AJUSTE DE PREDICTOR PARA IMAGE SEGMENTATION DATA SET

Antonio José Blánquez Pérez, Diego Navarro Cabrera

Proyecto final Aprendizaje Automático - Universidad de Granada

1 Definición del problema a resolver y enfoque elegido

La base de datos seleccionada es Image Segmentation Data Set, creada por el Vision Group perteneciente a la Universidad de Massachusetts. Está formada por varias fotografías recogidas de 7 bases de datos que contienen distintos elementos al aire libre, los cuales son losa, cielo, follaje, cemento, ventana, camino y césped. Cada imagen ha sido segmentada a mano de manera que cada instancia de nuestra base de datos es una región de 3x3 píxeles de alguna de ellas, teniendo cada una 19 atributos que se detallarán más adelante.

Con esta información se pretende generar un modelo mediante técnicas de aprendizaje supervisado que prediga correctamente, dada una sección de 3x3 píxeles de una imagen, a que clase de las 7 antes mencionadas pertenece.

2 Argumentos a favor de la elección de los modelos

Se han elegido tres modelos candidatos para el problema que nos ocupa, uno lineal y dos no lineales. El modelo lineal ha sido seleccionado mediante validación cruzada entre dos posibilidades: Regresión Logística y Perceptrón, siendo esta última técnica la que mejor resultado proporciona y por tanto la seleccionada para comparar con los dos modelos no lineales. El modelo Perceptrón es un gran clasificador, sobre todo cuando los datos son linealmente separables, ya que si esto ocurre nos asegura una clasificación perfecta de los datos de la muestra de entrenamiento; siendo además, ayudado por el algoritmo Pocket, capaz de dar buenos resultados aún sin ser los datos linealmente separables.

3 Codificación de los datos de entrada para hacerlos útiles a los algoritmos

En principio los datos no necesitan ningún tipo de codificación para ser útiles en proceso de aprendizaje, no obstante se han convertido los datos a combinaciones cuadráticas de estos. Así damos potencia al ajuste ya que, aunque estén ajustados por un modelo lineal, los datos podrán estar clasificados como funciones cuadráticas. Para ello se ha hecho uso de la función *Polynomial_Features* del módulo *Preprocessing* de Scikit-Learn, pasándole como parámetro el exponente 2 para obtener la susodicha combinación.

4 Valoración del interés de la variables para el problema y selección de un subconjunto (en su caso)

En cuanto a las variables medidas en cada instancia, todas parecen aportar información que uno podría considerar valiosa para identificar una imagen, y puesto que no es un número demasiado elevado de atributos no tenemos razones para intentar reducir la dimensionalidad del problema deshechando algunas de ellas o usando una técnica de reducción como PCA.

Si que hay una variable de la que nos vamos a deshacer, pero esto es debido a que funciona como una constante que toma el mismo valor en todas las instancias, esta variable es el número de píxeles por región, que es 9 en todos los casos. Al ser una variable que no cambia de una instancia a otra no nos aporta ninguna información que pueda ser útil para nuestro modelo, por lo que eliminarla no afecta a la solución, pero si al tiempo de ejecución, que se ve reducido al tener que tratar con menos datos.

5 Normalización de las variables(en su caso)

Puesto que el rango de los atributos varia de unos a otros es importante reescalar cada uno de forma independiente para poder compararlos mejor y no darle más o menos peso a cada atributo en función de su tamaño. Para esto usaremos una escala min-max que para cada atributo x_i perteneciente a la muestra i calcula:

$$x_i' = \frac{x_i - min}{max - min} \tag{1}$$

Donde min y max son los valores mínimo y máximo del atributo en toda la muestra.

Este tipo de escala mantiene la distribución de los valores, pero los ajusta a un rango entre 0 y 1. Si supiéramos que los datos se siguen un tipo de distribución como la distribución normal podríamos usar otras formas de normalización, pero este no parece ser el caso, por lo que usar estas otras técnicas modificaría la distribución de la muestra y provocaría un peor ajuste.

Para evitar mezclar la información del conjunto de entrenamiento con la del de prueba aplicaremos este ajuste en cada conjunto de forma independiente, ya que si no no podríamos confiar en el valor de E_{test} para estimar E_{out}.

6 Justificación de la función de pérdida usada

7 Selección de las técnica (paramétrica) y valoración de la idoneidad de la misma frente a otras alternativas

7.1 Modelo Lineal

Para ajustar el modelo lineal vamos a usar la función de la biblioteca sklearn del gradiente descendente estocástico $SGDClassifier$. Vamos a comparar entre 2 modelos diferentes, y decidiremos cual es mejor en tiempo de ejecución. Los modelos que vamos a comparar son regresión logística y la implementación que usa la biblioteca del Perceptrón.

Usamos SGD como técnica de ajuste porque es un algoritmo de optimización ámpliamente usado para ajustar todo tipo de modelos, y su implementación en scikit-learn nos facilitará su uso y el ajuste de todos sus posibles parámetros.

7.2 Boosting

Como técnica de boosting vamos a usar gradient boosting, que aunque no use funciones stamp como se recomienda en el guión, funciona mucho mejor que AdaBoost u otras funciones que si las usen. Esto se debe a que puesto que nuestro conjunto de datos de entrenamiento es muy pequeño y simple, estimadores muy débiles no son lo suficientemente buenos como para contribuir de forma tan significativa como en casos más complejos, y sobre todo, un conjunto de entrenamiento pequeño requiere de un tiempo de aprendizaje mucho menor y

por lo tanto es viable usar métodos más exigentes como gradient boosting, que requiere de significativamente más tiempo que AdaBoost para el ajuste.

7.3 Random Forest

En el caso de Random Forest vamos a usar la versión estándar de scikit-learn para clasificación: $RandomForestClassifier$, del módulo $ensemble$. Se plantean otras alternativas nombradas como árboles extremadamente aleatorizados, sin embargo en este caso no nos interesan ya que este conjunto de datos es lo suficientemente simple como para poder generar un número de árboles que nos garantice una buena varianza en un tiempo razonable. Es por esto por lo que no nos interesa una técnica que fuerza más la aleatoriedad en pos de generar varianza a costa de sesgo.

8 Aplicación de la técnica especificando claramente que algoritmos se usan en la estimación de los parámetros, los hiperparámetros y el error de generalización.

Para ajustar los hiperparámetros de todos los modelos usaremos la función $GridSearchCV$ de la biblioteca scikit-learn, que recibe un conjunto de valores posibles para los distintos parámetros que se quieren ajustar y usa validación cruzada para determinar cual es la mejor combinación de dichos valores.

8.1 Modelo Lineal

Para el ajuste de los modelos lineales nos centraremos en 2 parámetros que ajustaremos con $GridSearchCV$: loss, que indicará el modelo ajustado, que podrá ser log (regresión logística) o perceptron, como ya se ha comentado antes; y el factor de regularización, que provocará una regularización más o menos agresiva en función de lo alto que sea.

A parte de estos parámetros, es importante determinar el tipo de regularización usada, que en este caso será la regularización Lasso, ya que no parece que los atributos estén muy correlados y algunas características podría interesarnos que algunas características influyesen más en el modelo que otras.

El resto de parámetros que usa la función pueden dejarse tal y como están puesto que sirven más para definir el comportamiento del algoritmo para alcan-

zar la solución lo más rápido posible o determinar cuando parar la búsqueda, pero los valores por defecto ya son bastante buenos y modificarlos no nos llevaría a mejores soluciones, si no quizás a reducir los tiempos de ejecución.

8.2 Boosting

El principal parámetro que determinará la calidad de nuestro modelo es el número de estimadores, que indicará el número de etapas de la fase de aprendizaje. Puesto que este método es bastante resistente al sobreajuste, no hará falta hacer muchas pruebas para ajustar este valor, puesto que podemos elegir un número que consideremos lo suficientemente alto sin miedo a pasarnos y obtener un mal ajuste.

El resto de parámetros podemos decir que al igual que con $SGDClassifier$, modificarlos no afectaría significativamente a la calidad del resultado, por lo que podemos dejarlos con sus valores por defecto.

8.3 Random Forest

Uno de los principales parámetros a tener en cuenta en Random Forest es el número de árboles que se usarán, que en nuestro caso hemos comprobado que el más optimo es 100 debido a que el resultado deja de mejorar a partir de ese punto, entrenando aún así en un tiempo razonable. Otro parámetro a tener en cuenta es el criterio usado al elegir los atributos, teniendo las opciones del criterio Gini y el de la entropía.

El segundo es el que hemos estudiado junto al algoritmo ID3, mientras que el primero tiene en cuenta la probabilidad de que un elemento elegido aleatoriamente quede mal etiquetado si lo hacemos también de manera aleatoria. Como a simple vista no queda claro que técnica es mejor en este caso, se han sometido a validación cruzada, quedando seleccionado el criterio de la entropía.

También existen una serie parámetros relacionados con el criterio de parada para establecer un conjunto de hoja. Como hemos mencionado anteriormente, el entrenamiento no tiene un gran coste en tiempo, por lo que no tiene sentido recortar la generación de ramas. Por ello hemos creído conveniente dejar estos valores de manera que no limiten el proceder del algoritmo, coincidiendo con los valores por defecto.

Adicionalmente es posible establecer cuantos atributos participan al ramificar el árbol. Tampoco es fácil de determinar cuántos darán el mejor resultado, por lo que se han sometido a validación cruzada las posibilidades de la raíz cuadrada del total, el logaritmo en base 2 y los porcentajes de 60, 70, 80 y 90%,

que son los que hemos creído más razonables. El valor finalmente elegido ha sido el 80%.

El resto de parámetros que proporciona la función usada no son interesantes para este estudio, por lo que se dejan por defecto.

9 Argumentar sobre la idoneidad de la función regularización usada (en su caso)

10 Valoración de los resultados (gráficas, métricas de error, análisis de residuos, etc)

11 Justificar que se ha obtenido la mejor de las posibles soluciones con la técnica elegida y la muestra dada. Argumentar en términos de los errores de ajuste y generalización