Universidad De Granada

Apredizaje Automático

Proyecto Final

Pedro Bonilla Nadal
Antonio Martín Ruiz

12 de junio de 2020

Índice

1. Compensión del problema a resolver

Este dataset contiene datos de la oficina del censo[1] relativos a censos de 1995, obtenidos del repositorio UCI de bases de datos [2]. Un total de 48842 personas han sido encuestadas para este censo. De estas personas tenemos una serie de variables con información de carácter socioeconómico. En particular:

- tenemos 6 variables de tipo numérico u ordinal, con valores en rangos distintos.
- 8 variables de tipo categórico.
- Una variable de clase que toma como valores < 50K y > 50k.
- Es un problema de clasificación, ya que no tenemos información para hacer una regresión sobre la ganancia.

El código utilizado para la resolución de la práctica se encuentra en el archivo `main.py`. Del mismo modo, los datos limpiados se encuentran en el archivo `adults.data` y su información relativa procesada en el archivo `adult.names`.

2. División y codificación de los datos

A la hora de obetener los datos en el respositorio se encuentran tres archivos: `adult.data`, `adult.test` y `adult.names`. Estos datos fueron procesados en el año 1996. Nosotros realizamos una operación de formato de los datos para ajustarlos a convenciones más recientes.

- Cambiamos la variable de clase por valores categóricos 0 para < 50K y 1 para > 50K
- Eliminamos el espaciado después de la coma para facilitar la lectura por parte de librerías actuales.
- Añadimos la información del archivo `adult.test` al archivo `adult.data` con el objetivo de tener la información procesada de manera conjunta.
- Añadimos el flag `@attribute` a la lista de atributos provista en `adult.names`.

Una vez leídos los datos haremos una división train/test de 80 %/20 %. Elegimos esta proporción aprovechando que dada la gran cantidad de datos de los que poseemos. Obtenemos por lo tanto un conjunto de train con 39074 instancias, así como un conjuto de test con 9768 instancias.

Además, a la hora de validar la selección de hiperparámetros utilizaremos la técnica cross validation, para la cual diviremos el conjunto de training en 5 subconjuntos. Se utilizará para esta la función `cross_validate`[4].

Como ultimo detalle de codificación de los datos, durante la parte de preprocesado realizamos la técnica de dummy variables. Una variable dummy es aquella que toma sólo el valor 0 o 1 para indicar la ausencia o la presencia de algún efecto categórico que se pueda esperar que cambie el resultado. Esta técnica ayudará a codificar la entrada para algunos algoritmos, así como poder hacer un estudio de cada categoría como una variable separada a la hora de hacer una valoración del interés de cada variable.

3. Preprocesado

Mostraremos a continuación las siguientes técnicas de preprocesado realizadas. Justificaremos individualmente su uso.

- Normalización de las variables: realizaremos una normalización de los datos, para evitar que su escala afecte a la relevancia de estos. Para ello usaremos la función `StandardScaler`[3] provista en la librería `sklearn`.
- Valoración del interés de la variables: Para esto vamos a realizar dos técnicas:
  - El conjunto de datos presenta una altisima cantidad de columnas (105) tras la realización de las codificación por dummy variabels. Por ello vamos a intentar reducir aquellas que presenten poca o ningúna información haciendo un análisis por varianza, y eliminando aquellas columnas cuya varianza sea inferior a 0.01. Esto lo realizaremos con el algoritmo `varianceThreshold`[9].
  - Realizaremos un análisis de componentes principales PCA[10] para intentar reducir aun más la dimensionalidad del conjunto.

Proceso	Dimensionalidad Resultante
`Dummy_Variables`	105
`varianceThreshold`	57
`PCA`	55

Tabla 1: Dimensionalidad de la muestra tras cada proceso.

Podemos observar como tenemos una gran cantidad de variables que aportan poca o ninguna información. Observando la muestra atentamente veremos que se debe a variables como `nacionality`, que tras ser codificadas como dummy variables quedan como columnas nacionalidades con muy pocos positivos. Esto es negativo porque da pie a que algunos algoritmos (en particular los basados en árboles) sobreajusten a esta submuestra en particular.

- Polynomial Freatures: con especial interés en el caso lineal, el uso de variaciónes polinómicas de los datos puede ser util para permitir aproximaciones a clases de funciones nuevas. Probaremos variación cuadrática sobre las variables numéricas cuando realizemos un ajuste lineal. Para Random Forest no lo consideramos necesario por la ya conocida complejidad de los arboles, y para SVM utilizaremos variaciones de kernell.

- Valoración de las variables de interés: Para realizar un estudio de las variables de interés. Este ya lo hemos visto mostrado en
- Datos incompletos y valores perdidos: en relación a los valores perdidos encontramos en tres variables de tipo categórico.

Realizamos la sustitución de estos datos mediante la función `replace_lost_categorical_values`. En esta, para cada columna, calculamos su distribución de probabilidad, esto es, sumamos las ocurrencias de cada categoría y dividimos entre el total de valores con valores no perdidos. Obtenemos así la probabilidad de cada categoría. A continuación calculamos las probabilidades acumuladas sumando para categoría su probablidad y la de todas las

Variable	Valores distintos	Valores Perdidos
`workclass`	9	2799
`occupation`	15	2809
`native-country`	42	857

Tabla 2: Representación de valores perdidos.

anteriores. Para cada dato perdido generamos un número aleatorio mediante una distribución uniforme en el intervalo $[0, 1]$. La clase por la que el valor perdido será sustituida será la de cuyo intervalo contenga al valor generado, siendo el intervalo de cada clase el comprendido entre la probabilidad acumulada de la anterior y su probabilidad acumulada (incluyendo en cada uno su extremo inferior, pero no su extremo superior).

- Datos inconsistentes: no encontramos datos inconsistentes.
- Balanceo de clases: nos encontramos ante una situación con un desbalanceo notable. Sin embargo, dada la cantidad de datos provista por la base de datos creemos que no es necesario realizar modificaciones de los datos ni en el conjunto de train ni en el de test.

4. Métricas

Hemos decidido conseiderar, en este contexto de clasificación, dos funciones con objetivo de medir el error, ambas halladas en [6].

- *accuracy*: Decidimos utilizar esta medida por su simplicidad y expresividad. Esta métrica nos propocionará una idea general de la bondad de nuestro modelo en un caso general, así como nos permitirá comparar, por ejemplo, con el clasificador modal.
- *f1-score*: Al estar en un modelo de clases desbalanceadas, consideramos que debiamos utilizar una métrica que penara comportamientos de 'asignación modal'. La medida F_1 se define como[7]:

$$F_1 = \frac{2}{\text{recall}^{-1} + \text{precision}^{-1}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

Donde precisión es (en términos de clase positiva/negativa) el número de verdaderos positivos entre los positivos escogidos, y recall es el número de positivos escogidos entre los positivos totales.

5. Modelos Considerados

5.1. Modelo Lineal

Como modelo lineal, dado que estamos en clasificación, podemos escoger de los modelos explicados en clase entre Regresión Logística y PLA. Decidimos utilizar regresión logística por una serie de motivos.

- Al estar en un dataset que proviene de un censo sucede que encontraremos una gran cantidad de outliers y datos algo ruidosos. Esto hará que el perceptron sea contraindicado, dado que dificilmente podremos conseguir convergencia de este debido al ruido, y en el caso de que uno de estos elementos ruidosos sea además un outlier podría variar notable (e indebidamente) el plano clasificador.

- Nos resulta interesante el hecho de que podamos obtener la confianza de una predicción y no solo la predicción.

Para el uso de la regresión logistica haremos uso de la función de sklearn.

5.2. Random Forest

5.3. SVM

Consideramos que la técnica SVM-soft margin puede ser muy útil para este problema debido a que podemos fijar a placer el grado de compromiso entre busqueda del plano óptimo y la tolerancia al ruido, ambas cosas resultando de mucha utilidad a la hora de trabajar con datos potencialmente ruidosos.

Otro motivo para decidir ajustar este modelo, aunque de carácter menos importante, es la curiosidad suscitada por este modelo durante el desarrollo teórico de la asignatura.

6. Técnica de Ajuste para el Modelo Lineal

Para este conjunto utilizaremos dos técinas de ajuste, ambas definidas como variaciones especializadas sobre el algoritmo SGD.:

- `lbfgs`: Utiliza una variación del método Newton visto en el bonus de la práctica 1 que en lugar de la matriz Hessiana utiliza una aproximación a esta. Es eficiente en comparación con el método de newton estandard y se podría definir como una variación del gradiente descendente estocástico. Como desventajas tiene que se podría acercar a puntos críticos que no fuese mínimos. Esta técnica no admite regularización `l1`.

- `liblinear`: es el ganador ICML 2008 large-scale learning challenge. Lo utilizamos para probar las regularizaciónes dado que funciona tanto con `l1` como con `l2`.

  Liblinear utiliza técnicas de descenso coordinado[8] donde minimizamos la función en torno a cada eje, resolviendo un problema de minimización unidimensional en bucle.

Estas dos técnicas se pueden entender como variaciones del gradiente descendiente estocastico en esecia, dado que surgen de su filosofía de realizar aproximaciones sucesivas al mínimo. Pese a ello se debe comprender que se parecen en poco más que filosofía. Son técnicas mucho más eficientes que el algoritmo de la pseudoinversa, cosa que nos puede resultar interesante, aunque trabajamos con un dataset que no es particularmente grande. Las compararemos y comentaremos sus resultados en la sección de hiperparámetros y selección del modelo, pero elegiremos lbfgs dado que para el tamaño de nuestro dataset no es relevante la mejora en eficiencia de `liblinear` y `lbfgs` consigue un resultado ligeramente mejor.

7. Hiperparámetros y selección del modelo

En esta sección estudiaremos la splicación de las técnicas especificando claramente que algoritmos se usan en la estimación de los parámetros, los hiperparámetros y el error de generalización

7.1. Modelo Lineal

Como hiperparámetro vamos a considerar la prueba de las dos distintas técnicas de ajuste, las dos técnicas de regularización, la selección del parámetro `max_iter` así como la fuerza de regularización. Veamos los resultados:

solver	Regularización	Fuerza Regularización	max_iter	E_{in}	E_{cv}
liblinear	l1	0.1	100	0.85289	0.85227
liblinear	l1	1	100	0.85276	0.85181
liblinear	l1	10	100	0.85281	0.85186
liblinear	l1	0.1	200	0.85289	0.85227
liblinear	l1	1	200	0.85276	0.85181
liblinear	l1	10	200	0.85281	0.85186
liblinear	l1	0.1	1000	0.85289	0.85227
liblinear	l1	1	1000	0.85276	0.85181
liblinear	l1	10	1000	0.85281	0.85186
lbfgs	l2	0.1	100	0.85268	0.85207
lbfgs	l2	1 y	100	0.85276	0.85181
lbfgs	l2	10	100	0.85281	0.85191
lbfgs	l2	0.1	200	0.85268	0.85207
lbfgs	l2	1	200	0.85276	0.85181
lbfgs	l2	10	200	0.85281	0.85189
lbfgs	l2	0.1	1000	0.85268	0.85207
lbfgs	l2	1	1000	0.85276	0.85181
lbfgs	l2	10	1000	0.85281	0.85186
liblinear	l2	0.1	100	0.85271	0.85209
liblinear	l2	1	100	0.85281	0.85186
liblinear	l2	10	100	0.85281	0.85186
liblinear	l2	0.1	200	0.85271	0.85209
liblinear	l2	1	200	0.85281	0.85186
liblinear	l2	10	200	0.85281	0.85186
liblinear	l2	0.1	1000	0.85271	0.85209
liblinear	l2	1	1000	0.85281	0.85186
liblinear	l2	10	10000	0.85281	0.85186

Tabla 3: Resultados con variables lineales.

8. Error de Generalización

9. Argumentar sobre la idoneidad de la función regularización usada

10. Conclusiónes

Valoración de los resultados y justificación (gráficas, métricas de error, análisis de residuos, etc)
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.

solver	Regularización	Fuerza Regularización	max_iter	E_{in}	E_{cv}
liblinear	l1	0.1	100	0.85289	0.85227
liblinear	l1	1 y	100	0.85276	0.85181
liblinear	l1	10	100	0.85281	0.85186
liblinear	l1	0.1	200	0.85289	0.85227
liblinear	l1	1	200	0.85276	0.85181
liblinear	l1	10	200	0.85281	0.85186
liblinear	l1	0.1	1000	0.85289	0.85227
liblinear	l1	1 y	1000	0.85276	0.85181
liblinear	l1	10	1000	0.85281	0.85186
lbfgs	l2	0.1	100	0.85268	0.85207
lbfgs	l2	1	100	0.85276	0.85181
lbfgs	l2	10	100	0.85281	0.85191
lbfgs	l2	0.1	200	0.85268	0.85207
lbfgs	l2	1	200	0.85276	0.85181
lbfgs	l2	10	200	0.85281	0.85189
lbfgs	l2	0.1	1000	0.85268	0.85207
lbfgs	l2	1	1000	0.85276	0.85181
lbfgs	l2	10	1000	0.85281	0.85186
liblinear	l2	0.1	100	0.85271	0.85209
liblinear	l2	1	100	0.85281	0.85186
liblinear	l2	10	100	0.85281	0.85186
liblinear	l2	0.1	200	0.85271	0.85209
liblinear	l2	1	200	0.85281	0.85186
liblinear	l2	10	200	0.85281	0.85186
liblinear	l2	0.1	1000	0.85271	0.85209
liblinear	l2	1	10000	0.85281	0.85186
liblinear	l2	10	1000	0.85281	0.85186

Tabla 4: Resultados con variables cuadráticas.

Referencias

[1] Página oficial de la Oficina del censo: `http://www.census.gov/ftp/pub/DES/www/welcome.html`.

[2] Página del Centro para el Apredizaje Automático y Sistemas Inteligentes:`http://archive.ics.uci.edu/ml/datasets/Adult`.

[3] Función `StandardScaler` de la librería `sklearn`: `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html`

[4] Función `cross_validate` de la librería `sklearn`: `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_validate.html`

[5] Artículo donde explica el concepto de dummy variable: `https://en.wikipedia.org/wiki/Dummy_variable_(statistics)`

[6] Métricas de sklearn `https://scikit-learn.org/stable/modules/model_evaluation.html#balanced-accuracy-score`

[7] Métrica f1-score `https://en.wikipedia.org/wiki/F1_score`

[8] Código RL: https://en.wikipedia.org/wiki/Coordinate_descent

[9] Uso del algoritmo varianceThreshold: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.VarianceThreshold.html

[10] Técnica de PCA: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html#sklearn.decomposition.PCA