Fundamentos de Aprendizaje Automático 2021/2022

PRÁCTICA Nº 1: Naive Bayes

1. Objetivo

El objetivo de esta práctica, en primer lugar, es implementar el algoritmo de clasificación *Naive-Bayes*. El algoritmo *Naive-Bayes* clasifica cada ejemplo en la clase cuya probabilidad a posteriori es máxima, suponiendo que todos los atributos con los que se representan las instancias son independientes. Tras nuestra implementación utilizaremos las librerías proporcionadas por Scikit Learn para Naive Bayes, de manera que podamos comparar los resultados de ambas alternativas.

2. Actividades

La planificación temporal sugerida y las actividades a llevar a cabo son las siguientes:

- *1ª semana:* Implementar el clasificador *Naive-Bayes* y aplicar este algoritmo a los conjuntos de datos **tic-tac-toe** y **german**, para obtener los porcentajes de error en clasificación en cada caso, tanto con validación cruzada como con validación simple. En todos los casos se deben hacer los cálculos sin aplicar y aplicando la *corrección de Laplace*. Tened en cuenta que el conjunto de datos **german** incluye atributos continuos, por lo que en ese caso supondremos que estos atributos siguen una distribución normal dada la clase. Para ello se debe calcular la media y la varianza de cada atributo continuo. Pueden ser de utilidad las funciones para la distribución normal del módulo `scipy.stats.norm`.

- *2ª semana*: Comparar los resultados obtenidos en nuestra implementación con los que proporciona la librería de aprendizaje automático Scikit-learn (http://scikit-learn.org/stable/). Reproducir el diseño experimental utilizado anteriormente para la validación simple y la validación cruzada. Algunos aspectos sobre el uso de Scikit-learn se comentan en el apartado 3. Evaluar y comentar las hipótesis obtenidas con el clasificador Naive-Bayes. Los detalles se encuentran en el apartado 4.

3. Scikit-learn

Scikit-learn proporciona diferentes implementaciones para el algoritmo Naive Bayes (http://scikit-learn.org/stable/modules/naive_bayes.html). En esta práctica, consideraremos las siguientes funciones:

- **MultinomialNB**: se utiliza el conteo del número de veces que un atributo toma un determinado valor dada la clase. El valor a priori de cada clase se puede asumir uniforme (`fit_prior=False`) o se puede obtener a partir de los datos (`fit_prior=True`). Esta función también permite especificar un parámetro aditivo de suavizado `alpha`. `alpha=0` no realiza ningún tipo de suavizado y `alpha=1` implementa la corrección de Laplace.

- **`GaussianNB`**: la verosimilitud de las variables se asume que sigue una distribución normal.

- **`CategoricalNB`**: es la versión Naive Bayes para datos categóricos. Asume que cada característica tiene su propia distribución categórica.

Un aspecto importante a tener en cuenta cuando se utilizan funciones de la librería Scikit-learn es que muchos de los métodos no admiten atributos categóricos como tal y es necesario realizar un preprocesado de los datos con el fin de poder utilizar determinados algoritmos. Para ello, scikit-learn proporciona la clase **`OneHotEncoder`** (http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html) contenida en el módulo `preprocessing`. Esta clase proporciona métodos para codificar valores enteros[1] en bits de forma que si una variable toma **m** posibles valores enteros diferentes, ésta se codifica en m bits con solo uno de ellos activos. Por ejemplo, en el conjunto de datos *tic-tac-toe* donde cada una de las nueve variables toma tres posibles valores (`x,b,o`), tras la codificación *one-hot* (o *one-of-K*), el dataset pasa a tener 9x3=27 atributos. Si usamos Pandas, existe el método **get_dummies** que realiza funciones similares. Se recomienda analizar previamente las ventajas y desventajas de cada alternativa.

En cuanto al particionado de los datos, el submódulo **`model_selection`** de Scikit-learn implementa diferentes estrategias (http://scikit-learn.org/stable/model_selection.html). En lo que se refiere a las estrategias de validación simple y validación cruzada a implementar en esta práctica, estas son algunas de las funciones más relevantes:

- **Validación simple**: La función **`train_test_split`** hace transparente la gestión de índices ya que devuelve directamente los conjuntos de entrenamiento y validación para un conjunto de datos dado. Por su parte, la instanciación de la clase **`ShuffleSplit`** proporciona los índices para dividir los datos en subconjuntos de training y test. Los parámetros de estas funciones/constructores permiten indicar la proporción de patrones a utilizar en test o el número de particiones a generar, entre otros.

- **Validación cruzada**: Las funciones **`cross_val_score`** y **`cross_valpredict`** devuelven directamente los errores por validación cruzada por cada *fold* y las predicciones por cada patrón, respectivamente. Por su parte, la instanciación de la clase **`KFold`** genera los índices de las particiones. Entre otras posibilidades, los parámetros de estas funciones/constructores permiten indicar el número de *folds* a generar o imponer la permutación aleatoria de los datos antes de generar la partición.

4. Fecha de entrega y entregables

Semana del 11 de Octubre al 15 de Octubre de 2021. La entrega debe realizarse antes del comienzo de la clase de prácticas correspondiente. Se deberá entregar un fichero comprimido .zip con nombre **`FAAP1_<grupo>_<pareja>.zip`** (ejemplo `FAAP1_1461_1.zip`) y el siguiente contenido:

[1] Recordar que en el constructor de la clase Datos se habían codificado las variables categóricas como enteros

1. **Ipython Notebook (.ipynb)**. El Notebook debe estructurarse para contener los siguientes apartados:

Apartado 1	**Naive-Bayes** • Tabla con los resultados de la ejecución para los conjuntos de datos analizados (tic-tac-toe y german). Considerar los dos tipos de particionado. Los resultados se refieren a las tasas de error y deben mostrarse tanto con la corrección de Laplace como sin ella. Se debe incluir tanto el promedio de error para las diferentes particiones como su desviación típica. Es importante mostrar todos los resultados agrupados en una tabla para facilitar su evaluación. • Breve análisis de los resultados anteriores. Discutir el efecto Laplace
Apartado 2	**Scikit-Learn** • Incluir los mismos resultados que en el apartado 1 pero usando los métodos del paquete scikit-learn: **MultinomialNB, GaussianNB** y **CategoricalNB**. • Analizar las diferencias de los resultados obtenidos para SKL en función del método empleado: **MultinomialNB, GaussianNB** y **CategoricalNB**. ¿Cuál crees que es mejor en cada dataset? ¿Por qué? • Analizar el efecto de una codificación **oneHotEncoder** en **MultinomialNB**.
Apartado 3	**Conclusión** • Comparar y analizar los resultados propios con los de Scikit-Learn.

2. Ipython Notebook exportado como html.

3. Código Python (**ficheros.py**) necesario para la correcta ejecución del Notebook.

5. Puntuación de cada apartado

La valoración de cada apartado será la siguiente:

- ✓ **Naive-Bayes propio**: 4,5 puntos
- ✓ **Naive-Bayes con Scikit-Learn**: 3 puntos
- ✓ **Presentación y discusión de los resultados**: 2,5 puntos

6. Referencias

[1] J. H. Orallo, M. J. Ramírez Quintana, C. F. Ramírez. *Introducción a la Minería de Datos*. Pearson, Prentice Hall