Fundamentos de Aprendizaje Automático 2014/2015

PRÁCTICA Nº 1

Objetivo

El objetivo de la práctica es implementar el algoritmo de clasificación estándar *Naive-Bayes*. El algoritmo *Naive-Bayes* clasifica cada ejemplo en la clase cuya probabilidad posterior es máxima, suponiendo que todos los atributos son independientes. Además, vamos a implementar de forma genérica el procedimiento de *validación cruzada* para poder evaluar los clasificadores creados.

Preliminares

TAREA
La clasificación es una de las tareas más utilizadas en el campo conocido como minería de datos (*datamining*). Cada instancia o ejemplo pertenece a una clase que se representa mediante un atributo de clase. El resto de los atributos de la instancia se utilizan para predecir la clase.

EVALUACIÓN
La clasificación, al igual que otros métodos de aprendizaje, permite construir modelos a partir de un conjunto de datos. El modelo puede ayudar a resolver un problema pero es necesario evaluar su calidad antes de utilizarlo en un entorno real. Es necesario un proceso de evaluación que nos ofrezca alguna medida objetiva de la precisión del modelo creado.

Para evaluar un modelo se podría utilizar el propio conjunto de entrenamiento como referencia pero esta aproximación conduce normalmente a lo que se conoce como *sobreajuste*, es decir, la tendencia del modelo a trabajar bien con conjuntos de datos similares al de entrenamiento pero poco generalizable a otros datos. Lo más habitual por tanto es utilizar conjuntos diferentes para crear el modelo (*datos de entrenamiento*) y para evaluarlo (*datos de validación* o *prueba*).

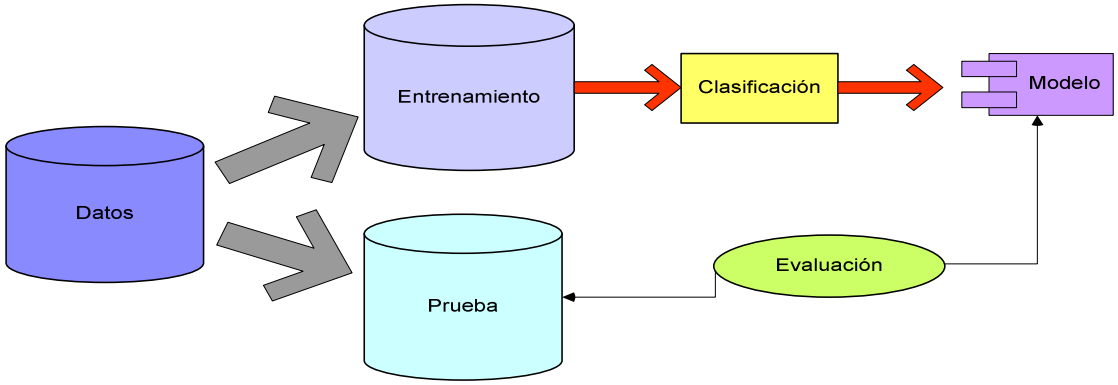

Figura 1: Evaluación de un modelo mediante conjuntos de entrenamiento y prueba

La utilización de dos conjuntos de datos distintos puede ayudar a resolver problemas como el sobreajuste, pero no está exenta de otro tipo de problemas, como la disponibilidad de pocos datos o la dependencia del modo en que se realice la partición. Una manera de evitar los problemas de dependencia consiste en utilizar la técnica de *validación cruzada (cross-validation)*. Con este método los datos se dividen en **k** grupos, uno de ellos se reserva para el conjunto de prueba y los **k-1** restantes se usan como conjunto de entrenamiento para construir el modelo. El modelo se evalúa entonces para predecir el resultado sobre los datos de prueba reservados. Este proceso se repite **k** veces, dejando cada vez un grupo diferente como conjunto de pruebas y el resto como conjunto de entrenamiento. De esta manera se obtienen **k** tasas de error, a partir de las que puede obtenerse el promedio, lo que nos dará la estimación de la precisión del modelo.

Actividades

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

- *1º semana*: Implementar la estructura de la aplicación necesaria para leer los datos del fichero de entrada, organizar los clasificadores y crear las particiones para los conjuntos de entrenamiento y prueba de un clasificador. Se deben implementar dos métodos de particionado distintos pero el diseño debería permitir añadir cualquier otro tipo de particionado de forma flexible. Una posible estructura de implementación se comenta posteriormente.

- *2º semana:* Implementar el clasificador de *Naive-Bayes*. Clasificar los patrones de los conjuntos de datos ***car*** (http://archive.ics.uci.edu/ml/datasets/Car+Evaluation) y ***lenses*** (http://archive.ics.uci.edu/ml/datasets/Lenses)**.** Obtener el error promedio de clasificación para los conjuntos de entrenamiento y de test. Clasificar los mismos conjuntos de datos aplicando la corrección de Laplace.

- *3º semana*: Clasificar los patrones de los conjuntos de datos ***iris*** (http://archive.ics.uci.edu/ml/datasets/Iris) y ***spam*** (http://archive.ics.uci.edu/ml/datasets/Spambase) mediante el método de *Naive-Bayes* suponiendo que todos los atributos siguen una distribución normal dada la clase. Para ello se debe calcular la media y la varianza de cada atributo dada la clase.

Lenguaje y Diseño

El lenguaje a utilizar para el desarrollo de la práctica será Java. Con el objetivo de desarrollar una aplicación lo más flexible y general posible se plantea un diseño basado en patrones cuyos detalles más importantes se especifican a continuación.

PARTICIONADO

Para conseguir un particionado de los datos con la mayor flexibilidad posible se debe crear un patrón de diseño denominado *EstrategiaParticionado*, que separe la estrategia de particionado del particionado en sí mismo. La figura muestra un esquema de este patrón.

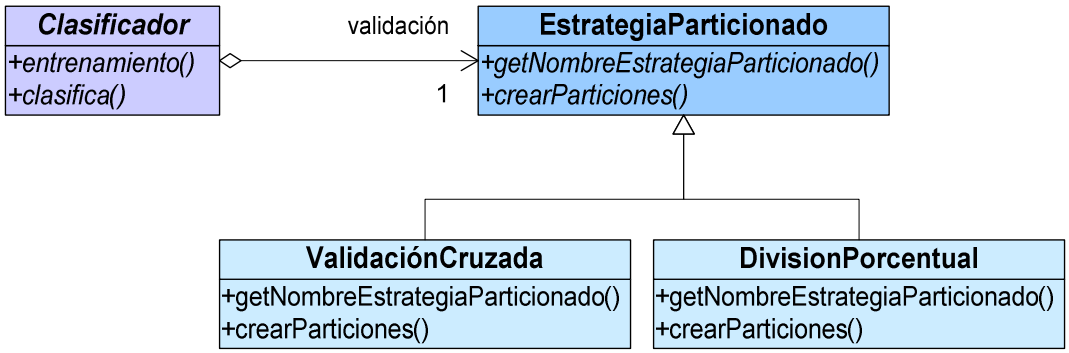

Figura 2: Esquema del patrón de particionado

La clase *Clasificador* utiliza una estrategia de particionado determinada para llevar a cabo su algoritmo de clasificación, empleando el conjunto de entrenamiento para construir el modelo y el conjunto de prueba para evaluarlo. Cada estrategia de particionado creará los conjuntos de entrenamiento y prueba de diferente forma según el algoritmo de particionado. En concreto, la *validación cruzada* dividirá los datos del fichero de entrada de forma aleatoria entre el número de particiones especificado, tal y como se ha explicado anteriormente. Por su parte, la *división porcentual* dividirá los datos de entrada en dos conjuntos, uno para entrenamiento y otro para prueba en función del porcentaje que se especifique. Con el esquema propuesto es sencillo ir añadiendo diferentes estrategias de particionado en función de las necesidades de cada clasificador.

CLASIFICADORES

Como el objetivo de la asignatura es implementar y trabajar con diferentes clasificadores la idea es crear una jerarquía de clasificadores, como se muestra en la figura.

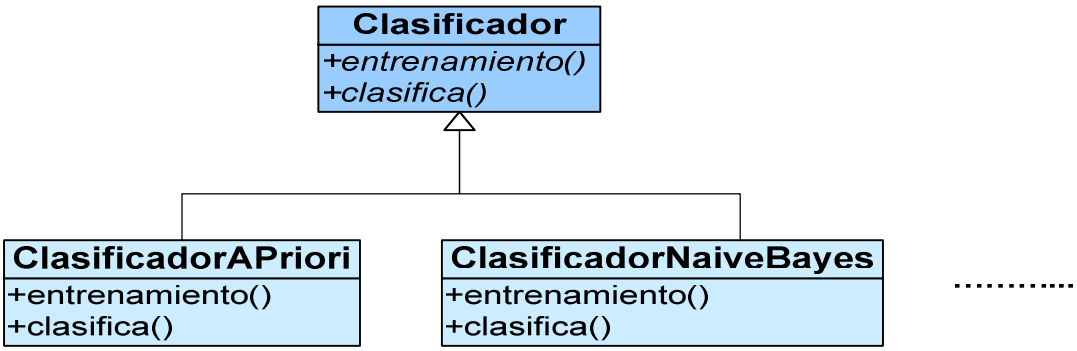

Figura 3: Jerarquía de clasificadores

Los métodos entrenamiento y clasifica son auto-explicativos y cada uno utilizará el conjunto de entrenamiento y prueba correspondiente que se crea según el tipo de particionado.

DATOS
Los conjuntos de datos propuestos pueden encontrarse en el repositorio de la Universidad de California. Además de los datos se incluyen una descripción de los mismos. Cada fichero debe tratarse para que siga la siguiente estructura:

- 1ª Fila: Número de datos del conjunto
- 2ª Fila: Nombres de los campos
- 3ª Fila: Tipos de los atributos: Nominal o Continuo
- Resto de filas: Conjunto de datos, uno por fila y campos separados por comas

PLANTILLAS
En moodle se encuentran plantillas para las clases *Datos*, *Clasificador y ClasificadorAPriori*, así como la del patrón de diseño *EstrategiaParticionado* que implementará las estrategias de particionado. La clase *ClasificadorAPriori* se puede usar para probar la validación cruzada. Como fichero de datos se puede usar el conjunto de datos ***iris***.

Fecha de entrega y entregables

Jueves 9 de Octubre: Fichero comprimido con el código de las clases Java y una memoria resumida explicando los resultados obtenidos. En la clase de prácticas previa a la entrega se podría solicitar una prueba de ejecución.