Sistemas de Gestión de Datos y de la Información

Algoritmos de Minería de Datos

[Práctica 3]

Javier Martín Moreno-Manzanaro

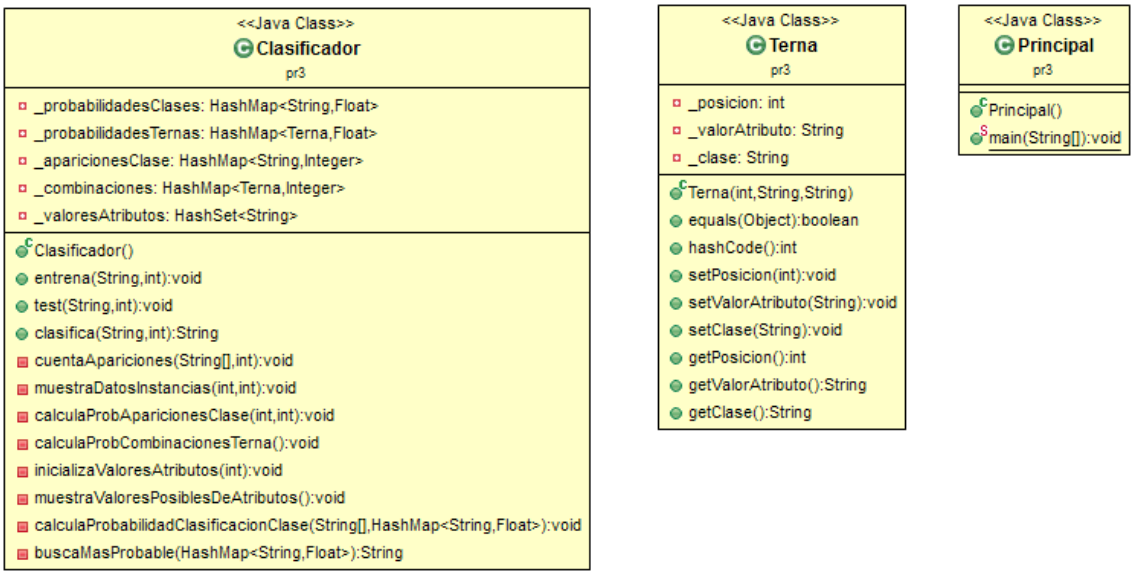
2014

Universidad Complutense de Madrid

Índice

Diseño de la Aplicación

El diseño seguido para la implementación de esta práctica es el especificado en el enunciado de la misma, es decir una clase que implementa el clasificador. En un primer momento y debido a la escasez de clases, sopesé la posibilidad de evitar poner el diagrama de clases, sin embargo, dado que ofrece a golpe de vista, los atributos y métodos implementados de las clases, lo adjunto a continuación.



Principal

Clase que contiene el método main, que establece el punto de inicio de la aplicación.

Lo primero se crea un objeto de clase 'Clasificador', y luego vamos invocando sus métodos para ir probándolos. En primer lugar, practicamos un poco llamando al método entrena con el fichero de entrenamiento (una variación del "car.data" subido al Campus Virtual consistente en una selección de un conjunto de líneas al azar, agrupadas en bloques no consecutivos, contando un total de 401 líneas).

Después lo probamos llamando al método test, con el fichero destinado a tal efecto. Dicho fichero consta de 201 de las últimas líneas del fichero "car.data".

Por último se invoca, al método clasifica de manera individual (ya se habrá utilizado internamente en la llamada al método test), para realizar una traza aislada y mostrarla por consola. Para efectuar esta traza, he cogido la última línea del fichero "car.data" (quitando el nombre de la clase, tal y como se propone en el enunciado).

Además de estas invocaciones, se añaden también algunas trazas por consola para dotar de mayor claridad a la salida escrita.

Terna

Esta clase representa la terna de datos formada por el valor de un atributo, su posición (nº de campo que ocupa en la línea) y la clase a la que pertenece. Dichos atributos son simples Strings y un entero para la posición.

En cuanto a los métodos con que se ha dotado a la clase, se incluyen un constructor con tres parámetros para dar valor a sus atributos y accedentes y mutadores para éstos últimos. Asimismo es importante destacar la necesidad de sobrescribir la implementación de los métodos hasCode y equals, para el adecuado funcionamiento de los HashMap utilizados en la clase Clasificador, y que explicaremos en el apartado siguiente. Baste decir, que se considera que dos Ternas son iguales, si sus atributos son iguales uno a uno; y que el cálculo del hashCode se realiza como la suma de la posición, la longitud del String del valor del atributo, y la longitud del nombre de la clase.

Clasificador

En esta clase se implementa el *Clasificador Bayesiano Simple,* que incluye los métodos 'entrena', 'test' y 'clasifica', y los atributos necesarios para su implementación. Dichos atributos son los siguientes:

```
private HashMap<String, Float> _probabilidadesClases;
private HashMap<Terna, Float> _probabilidadesTernas;
private HashMap<String, Integer> _aparicionesClase;
private HashMap<Terna, Integer> _combinaciones;
private HashSet<String>[] _valoresAtributos;
```

En ellos guardamos, respectivamente, las probabilidades de aparición de las clases, las probabilidades de aparición de las ternas en las clases de las propias ternas, el número de apariciones de las clases, el número total de combinaciones de las ternas, y los conjuntos de posibles valores de los atributos.

En cuanto a los métodos que contiene, los repaso a continuación.

Entrena

Además de lo necesario para leer el fichero, sacar trazas por consola y otros añadidos, en este método damos los pasos que se explican en el enunciado de la práctica, en el siguiente orden:

1. Contamos las apariciones tanto de las clases como de las ternas
2. Contamos las instancias totales y erróneas.
3. Mostramos los datos de las instancias
4. Calculamos las probabilidades de aparición de las clases
5. Calculamos las probabilidades de las combinaciones de las ternas
6. Mostramos los posibles valores de los atributos.

Los puntos 1 y 2 los vamos haciendo a medida que vamos leyendo líneas del fichero de entrada, mientras que los puntos 3 a 6 se implementan en sendos métodos privados, haciendo uso de los valores de los atributos que hemos ido rellenando en el primer y segundo punto. En

resumen, primero llevamos a cabo una fase de conteo y después una fase de cálculo de probabilidades.

Clasifica

Esta función sirve para clasificar una instancia con una clase, para ello, dividimos el proceso en dos partes:

1. Calculamos la probabilidad, para cada clase, de que se corresponda con una instancia. Para ello, vamos acumulando el producto de las probabilidades de las ternas de cada atributo que tengan como clase la clase que estamos mirando; y repetimos el proceso para cada clase.
2. Una vez calculadas las probabilidades del punto anterior, simplemente comprobamos cuál es la mayor, y esta será la predicción que devolveremos.

Nuevamente, cada uno de los puntos anteriores está codificado en su respectivo método privado, para tener una implementación más estructurada.

Así pues, en resumidas cuentas, calculamos las probabilidades de que una instancia tenga una clase, y devolvemos la clase más probable.

Test

De nuevo, al margen del código necesario para leer el fichero de entrada, sacar las trazas por consola y demás “extras”, lo que se realiza en este procedimiento se recoge en los siguientes pasos:

1. Realizamos una predicción con el método clasifica (leyendo línea a línea el fichero de entrada, pero sin considerar las clases) y vamos contando los aciertos y fallos tras compararla con la instancia original.
2. Contamos el número de instancias totales y erróneas.
3. Mostramos los datos de las instancias del punto anterior (usando el mismo procedimiento que el utilizado en el apartado 3 del método entrena, explicado un poco mas arriba).
4. Calculamos la tasa de aciertos

Así pues, recapitulando, primero se realiza una predicción basándonos en las probabilidades calculadas al entrenar, contando los aciertos y fallos para luego calcular la tasa de aciertos.

Requisitos e Instrucciones

Entorno

La configuración del entorno que he utilizado para la realización de la práctica ha sido:

- **Eclipse Juno (4.2)**
- **Plugin ObjectAid UML v1.14**
- **JavaSE1.7**
- **Windows 7**.

Lanzando tareas

Para lanzar las tareas de minería de datos de la práctica y probar su funcionamiento, simplemente se ha creado un objeto de clase clasificador y se invocan los métodos pertinentes.

Para cambiar los ficheros que se deben cargar en cada método, es necesario hacer dos cosas:

- Cambiarlos en el main, en los métodos en que se quieran utilizar, escribiendo el string con el nombre del fichero.
- Asegurarse de que el fichero que vamos a leer esté en la carpeta del proyecto (en mi caso en “Clasificador_Bayesiano_Simple”).

Como lo más sencillo es verlo, dejo el método main de la práctica aquí debajo:

```
public static void main(String[] args) {

Clasificador c = new Clasificador();

System.out.println();
System.out.println();
System.out.println("UTILIZAMOS LA FUNCIÓN DE ENTRENAMIENTO Y MOSTRAMOS
ALGUNOS DATOS");
System.out.println();
c.entrena("car_half.data", 7);


System.out.println();
System.out.println();
System.out.println("UTILIZAMOS LA FUNCIÓN DE TEST, MOSTRANDO LAS PREDICCIONES
PARA CADA INSTANCIA");
System.out.println();
c.test("test.data", 7);


System.out.println();
```

```
System.out.println();
System.out.println("POR ÚLTIMO, MOSTRAMOS EL RESULTADO DE UNA EJECUCIÓN
AISLADA DE LA FUNCIÓN CLASIFICA, CON UNA INSTANCIA AL AZAR (CABLEADA), COMO
HEMOS HECHO EN LA FUNCIÓN ANTERIOR");
System.out.println();
System.out.println("La clase de la instancia introducida es:   ---" +
c.clasifica("low,low,5more,more,big,high", 7) + "---");
}
```

Nótese que la instancia que usamos para probar de manera aislada la función clasifica, es la última línea del fichero car.data, cableada en el código en un String de entrada de la función.

Algunas notas y consideraciones

- En el código se incluyen comentarios que ayudan a seguir el proceso de entrenamiento, test y clasificación, así como comentarios que aclaran la implementación que yo he hecho.

- Para cada método que se debe implementar se trazan todos los pasos y resultados que se van dando, ofreciendo información en todo momento de lo que va sucediendo.

- Al calcular la probabilidad de que una instancia sea de una clase, se hace multiplicando las probabilidades de que cada terna de cada atributo sea de esa clase, por lo que si la probabilidad de una terna (un atributo en una posición de una clase concreta) es cero, la probabilidad de que la instancia sea de esa clase será cero. Esto es lo más lógico ya que si un atributo nunca va a tener un valor determinado en una clase concreta, las instancias con ese atributo que tengan ese valor, nunca van a ser de esa clase.