**PRÁCTICA 3: ALGORITMOS DE CLASIFICACIÓN**

*INGENIERÍA DEL CONOCIMIENTO*

*Iván Pisonero Díaz*

*Sergio Vázquez carbajo*

Índice

1.- Estimación paramétrica Bayes……………………………………………4

1.1.- Detalles de implementación…………………………………………………….4

1.2.- Resultados de la ejecución……………………………………………………….5

1.3.- Detalles de implementación transformación de entrada con C++..................6

1.4.- Manual de usuario………………………………………………………………………………….6

2.- Agrupamiento borroso (k-medias)...........................................7

2.1.- Detalles de implementación…………………………………………………….8

2.2.- Resultados de la ejecución……………………………………………………….9

2.3.- Manual de usuario………………………………………………………………….10

1.- Estimación paramétrica Bayes:

Hemos realizado una implementación del método de estimación paramétrica de bayes usando Python y una librería muy conocida para este lenguaje llamada numpy, que hemos usado para trabajar con matrices. También hemos hecho un pequeño programa en C++ para poder tomar el fichero de entrada original, Iris2Clases.txt, y convertirlo en varios ficheros con la información estructurada de diferente forma, para que fuese más fácil de cara al uso de dicha librería, que requiere de una sintaxis específica. Esto también nos permitió sacar las clases de antemano, y poder así inicializar los diccionarios / conjuntos de Python fácilmente.

1.1.- Detalles de implementación:

El algoritmo está implementado en Python. En primer lugar lee las clases e inicializa el conjunto que contiene a las clases y el diccionario en el que, para cada clave, crearemos una cadena con todos los casos correspondientes a esa clave (a esa clase).

En primer lugar, lee cada caso y lo introduce en la cadena de ejemplos de la clase a la que pertenece. Esta cadena tiene un formato específico para que pueda ser transformada a matriz fácilmente usando numpy. Esta transformación es la que se hace después.

A continuación, se calcula el vector de media de cada clase, usando el método mean, y dándole como parámetro 0, que indica que se deben obtener las medias por columnas. Se almacena dicho vector en un diccionario donde la clave es el nombre de la clase.

Una vez calculado esto, se pasa al cálculo de las matrices C inversas. En primer lugar, se inicializa la matriz C a una matriz de ceros. Después, se recorren los ejemplos de la clase cuya matriz C inversa se está calculando, llevando a cabo la operación (para la operación transpuesta se usa el método transpose de numpy):

Y acumulando el resultado en la matriz en un principio inicializada a ceros, sumando ambas matrices. Obtenemos así el sumatorio de todas estas matrices, cuyo resultado se divide entre el número de ejemplos. Con esto obtenemos la matriz C. Por último, se calcula su inversa usando la función getI() que ofrece numpy, y se almacena en un diccionario donde la clave es el nombre de la clase.

Una vez calculada la matriz C inversa de cada clase, procedemos a valorar cada caso. Los casos se leen del fichero casos.txt. Después, iteramos sobre cada clase del conjunto de clases. Para cada clase c, en primer lugar hacemos:

Después, calculamos:

Tras usar el método item(0) sobre el resultado de esa operación, que usamos porque el resultado es una matriz 1x1 y de esa forma obtenemos el elemento como número, procedemos a comparar el resultado con el menor resultado hasta ahora. Si es menor, actualizamos el menor resultado hasta ahora y la clase que lo ha conseguido, hasta haber iterado por todas las clases. Esto únicamente cambia en la primera iteración, en la que el menor resultado hasta ahora se inicializa al resultado actual.

Por último, imprimimos la estimación que se ha realizado (la clase que ha conseguido la menor distancia).

1.2.- Resultado de la ejecución

Exponemos en la ilustración 1 el resultado de la ejecución de bayes.py. Los casos son los propuestos en TestIris01, TestIris02 y TestIris03.



Ilustración 1: Resultado de la ejecución de bayes.py

1.3.- Detalles de implementación transformación de entrada con C++:

Para el programa de C++ que se encarga de procesar la entrada y transformarla en los ficheros que usa el programa de Python, usamos los siguientes elementos:

* Un conjunto de clases, en el que se va insertando cada clase que se procesa, y que al no permitir duplicados, nos aseguramos de que contendrá todas las clases que aparezcan en el fichero, y solo una vez.
* Un vector de ejemplos, que es un vector de pares cuya primera componente es un vector de valores (double), y su segunda componente es la clase a la que pertenece.
* Un mapa que asocia cada clase con un vector de casos (cada caso es un vector de valores double).

En primer lugar se lee la entrada y se inserta en el conjunto de clases y en el vector de ejemplos. Una vez leída, se inicializan las entradas del mapa para cada clase procesada a un vector vacío. Después se itera sobre cada ejemplo y su vector de valores se inserta en la entrada correspondiente a su clase en el mapa.

Al ser este mapa iterable, podemos generar la salida que necesitamos para que sea fácil convertirla en una matriz, para cada clase, desde el programa de Python.

1.4.- Manual de usuario:

Para ejecutar la práctica se necesitará python, así como la librería numpy que se puede instalar usando la herramienta pip que se distribuye junto con python (por ejemplo, ejecutando ‘pip install numpy’ o ‘python -m install numpy’ desde línea de comandos).

Después, navegamos hasta el directorio que contenga el ejecutable y los ficheros usando la línea de comandos (por ejemplo, usando: cd ‘ruta-al-directorio’).

A continuación, usamos ‘python bayes.py’ para ejecutar la práctica, que imprimirá los tres casos que se proponen y la clasificación que realiza el algoritmo de ellos, es decir, la clase a la que se ha estimado que pertenece el caso (los ficheros que necesita el programa de python ya están generados, luego no hace falta ejecutar el programa de C++ antes).

Si se quieren probar otros casos bastará con añadirlos al fichero casos.txt, poniendo un ejemplo por línea y siguiendo el siguiente formato:

valor\_1, valor\_2, valor\_3, valor\_4

Para ejecutar el programa de C++ que usamos para generar la entrada, podemos usar Visual Studio. Creamos un proyecto, y movemos genera-entrada.cpp e Iris2Clases.txt a la carpeta de ese proyecto (si no es la misma), y los añadimos como archivos de recursos.

Podemos usar la opción ‘Depurador local’ para ejecutarlo.

Se nos imprimirá en pantalla el contenido generado, que es el que volcamos la primera vez en el resto de ficheros cuando los generamos. Esto se puede hacer desde el programa usando ofstream de la siguiente manera (no incluido para que no se escriba lo mismo dos veces en los ficheros):

ofstream fichero = open(‘ruta-fichero-donde-escribir’);

fichero << entrada;

fichero.close();

2.- Agrupamiento borroso (k-medias):

Hemos realizado la implementación del algoritmo k-medias en lenguaje c++, usando Visual Studio. En el que leemos el archivo Iris2Clases.txt y luego unos de los siguientes archivos: TestIris01.txt, TestIris02.txt, TestIris03.txt. Y en función de esos datos realizamos el algoritmo k-means.

2.1.- Detalles de implementación:

Cada archivo de texto (TestIris01.txt, TestIris02.txt, TestIris03.txt.) contiene los cuatro parámetros de cada flor y su nombre.

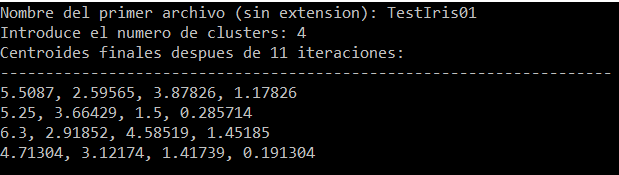
En nuestro programa recogemos los siguientes datos a tener en cuenta:

* **struct flor:** estructura de datos que recoge los datos a tener en cuenta de cada archivo de texto, los cuales contiene cuatro variables de tipo double y un string, para recoger los datos de el valor de los parámetros de el primer dato, el segundo, el tercero, el cuarto y la especie de la flor sucesivamente.
* **vector<flor> leerDatos(const string& archivo):** función que devuelve un vector de estructura “flor” de los datos leídos por archivo.
* **double distancia(const flor& a, const flor& b):** función que devuelve la distancia euclídea entre dos estructuras “flor”.
* **vector<flor>inicializarCentroides(vector<flor>& flores, int k):** función que inicializa los centroides y devuelve una estructura “flor” con esos centroides inicializados.
* **void borroso(vector<flor>& flores, int k):** algoritmo que implementa el algoritmo borroso (k-means).
* **vector<flor> combinarDatos(const vector<flor>& datos1, const vector<flor>& datos2):** función que combina los datos de dos estructuras “flor” y devuelve los datos de el resultado de la función.
* **main():** es la función principal, es dónde se leen los distintos archivos de texto para posteriormente realizar el algoritmo y las funciones.

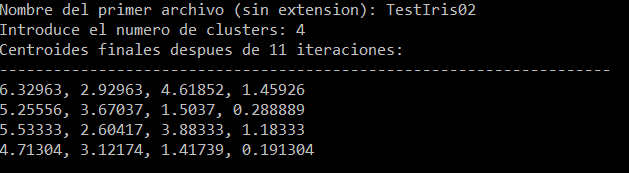
2.2.- Resultado de la ejecución:

A continuación se muestran los resultados de la ejecución para los tres archivos que se pueden proporcionar:

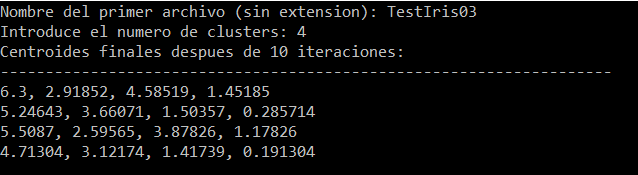
Caso expuesto con el archivo TestIris01.txt:



Caso expuesto con el archivo TestIris02.txt:



Caso expuesto con el archivo TestIris03.txt:



2.3.- Manual de usuario:

Para ejecutar el algoritmo primero es necesario tener Visual Studio 2019. Una vez cumplido este requisito, el programa nos pedirá un archivo a ingresar entre los siguientes; TestIris01.txt, TestIris02.txt, TestIris03.txt, el cual contendrá las clases para poder realizar nuestro algoritmo.Después procederá a leer el archivo Iris2Clases.txt, que es donde van a estar la matriz con todos los datos. Por último nos pedirá el número de clusters para especificar el tamaño.