Curso académico: 2018 - 2019

Estudiante: José Javier Alonso Ramos

DNI: 77766199-W

e-mail: jjavier.ar98@gmail.com

Grupo: 2

Horario prácticas: Miércoles 17:30 - 19:30

Práctica 1b:

Técnicas de Búsqueda Local y Algoritmos Greedy para el Problema del Aprendizaje de Pesos en Características

---

Universidad de Granada

Figure 1: Universidad de Granada

Índice

Aloritmos considerados

- Greedy RELIEF
- Local Search
- 1-NN

Descripción del problema

Dada una **población**, nos encontramos una serie de n **elementos** por los que está formada, teniendo cada uno de ellos x **características** según las cuales se les **clasifica** en una determinada clase c perteneciente a un conjunto de clases C.

Nuestro problema comprende la situación de clasificación de un nuevo elemento insertado en la población con la tara que supone no saber el criterio, según el cual, se clasifica.

Para llevar a cabo esta tarea crearemos un sistema clasificador automático que analizará un subconjunto de la población escogido de manera que sea representativo respecto a la población al que llamaremos **muestra**. Aprenderemos a partir de las características de los elementos que conforman la muestra y sus clasificaciones una manera de clasificar el nuevo elemento correctamente con una fiabilidad suficientemente alta.

Es un problema de **aprendizaje supervisado** ya que cuando entrenamos a nuestro sistema clasificador corroboraremos los resultados obtenidos con la clase real a la que pertenece el elemento, pudiendo así variar nuestras decisiones en pos de mejorar nuestro clasificador.
El método empleado para aprender a clasificar será decisivo en la calidad, coste requerido y velocidad de la clasificación.

Elementos comunes a todos algoritmos implementados

La práctica ha sido realizada en python. Usar python3 como intérprete; si es posible python3.6.

Módulos importados y para qué han sido utilizados:

- **arff** (de *scipy.io*): este módulo nos permite leer los archivos con extensión *.arff* de datos de población.
- **numpy**: nos permite trabajar de forma cómoda con vectores y matrices a parte de proporcionar una serie de funciones matemáticas como *valor absoluto*, *media*, *distancia euclídea*, *generación de números aleatorios*, *distribución normal y uniforme*, *truncamiento de valores a un determinado rango*, y un largo etcétera.
- **pandas**: nos permite trabajar de forma cómoda con conjuntos de datos y crear tablas entre otras cosas. En la práctica lo usamos para transformar los datos leídos del fichero arff a una matriz.
- **KDTree** (de *sklearn.neighbors*): creamos un arbol con un determinado conjunto de datos (*conjunto1*) al que, pasándole un segundo conjunto (*conjunto2* - que puede ser el mismo que el primero), nos devuelve k elementos del conjunto1 en orden del más cercano al más lejano a cada elemento del conjunto2.
- **StratifiedKFold** (de *sklearn.model_selection*): en la práctica se especifica que particionemos los datos leídos en 5 conjuntos del 20% de manera que las etiquetas se mantengan proporcionales al conunto original. Este módulo nos permite hacer esto además de manera aleatoria y mezclando los datos en los subconjuntos.
- **MinMaxScaler** (de *sklearn.preprocessing*): nos permite escalar los valores de las carácterísticas de los elementos al rango [1,0]
- **time** (de *time*): nos permite calcular el tiempo de ejecución de los ditintos algoritmos.
- **PrettyTable** (de *prettytable*): para imrimir los resultados obtenidos por los algoritmos de forma ordenada en tablas.

Funciones auxiliares:

- **byte2string(x)**: cuando leemos el conjunto de datos x transformamos todas las etiquetas (clases) en strings para poder trabajar con ellas de

manera uniforme. Esto permite que si leemos un conjunto de datos y sus etiquetas, en un principio, son númericas trabajaremos con él de igual forma que lo haríamos con un conjunto de etiquetas alfabéticas.

- **read_arff(name_of_file)**: lee el contenido de un archivo *.arff* situado en el directorio *../data/* relativo al directorio donde se encuentra nuestro script. Transforma los datos a un formato más amigable.
  Devuelve dos objetos: datos y metadatos.

```
read_arff(name_of_file):
    datos, metadatos = read(../data/name_of_file.arff)
    transform(datos)

    return datos, metadatos
```

- **get_tags(data)**: del conjunto de datos *data* devolvemos la última columna, es decir, las etiquetas.

```
get_tags(data):
    return last_column(data)
```

- **get_only_data(data)**: del conjunto de datos *data* devolvemos todas las columnas excepto la última, es decir, sólo las características de los datos.

```
get_only_data(data):
    return ( data - last_column(data) )
```

Elemento solución

Clasificador k-NN