Práctica 6

Ieltzu Irazu, Mikel De Velasco Y María Inés Fernandez

December 2, 2015

1 Introducción:

1.1 Exposición de la práctica:

En la práctica presentada se nos pide implementar el algoritmo K-Means (2) para el conjunto de datos colon.arff. Para ello hemos desarrollado nuestro código en el lenguaje de programación Java y después hemos creado nuestro ejecutador .jar. Además hemos desarrollado este documento para plasmar las partes más importantes de la práctica.

1.2 Objetivos:

Los objetivos son claros; conseguir un algoritmo eficiente y efectivo para clusterizar un conjunto de datos del que no se sabe la clase. El código que hemos desarrollado no es dependiente a un solo conjunto de datos,ya que es posible utilizarlo en más de un conjunto de datos.

Como hemos recibido el archivo colon.arff para desarrollar la práctica vamos a analizar el archivo para ver su contenido. En el archivo de datos podemos encontrar 2000 variables predictoras. Como este archivo sí tiene clase y nosotros estamos implementando un clusterizador, lo primero que haremos en la páctica sera borrar la clase para no tenerla en cuenta. Como son tantas variables nos limitaremos a decir que todas ellas son de tipo numérico, por lo tanto no habrá que discretizar ninguna variable y tampoco habrá que eliminar ninguna para poder implementar el algoritmo K-Means, ya que este solamente acepta variables de este tipo.

Para clusterizar nuestras instancias vamos a aplicar distintos grupos y haremos mediciones. Empezaremos dándole entrada del parámetro k a valor 2, el cual nos dice que cantidad de grupos vamos a tener para clusterizar las instancias. Una vez calculadas las particiones de datos calcularemos las métricas SSE y Silhouette para evaluarlo.

2 Pseudocódigo del algoritmo:

Figure 1: K-Means algorithm

Algorithm 1 *KMeans*

```
1:  procedure K-Means(Instances, K)
2:      Centroides ← escogerAleatoriamenteKInstancias(Instances, K)
3:      CentroidesTMP ← ∅
4:      while Centroides ≠ CentroidesTMP do
5:          CentroidesTMP ← Centroides
6:          Centroides ← ∅
7:          for Ins ∈ Instances do
8:              centroide ← CalcularCentroideMásCercano(Ins)
9:              MeterloEnElClusterDel(centroide, ins)
10:         end for
11:         for grupo ∈ Clusters do
12:             Centroides.add(RecalcularCentroide(grupo))
13:         end for
14:     end while
15: end procedure
end
```

3 Experimentación y Resultados:

Tras haber implementado el K-Means, y haber hecho algunas pruebas, hemos decidido sacar varios datos que hemos trasladado a los gráficos 3 (SSE y Silhouette) y 3 (Tiempo y Vueltas) para después analizar y sacar conplusiones.

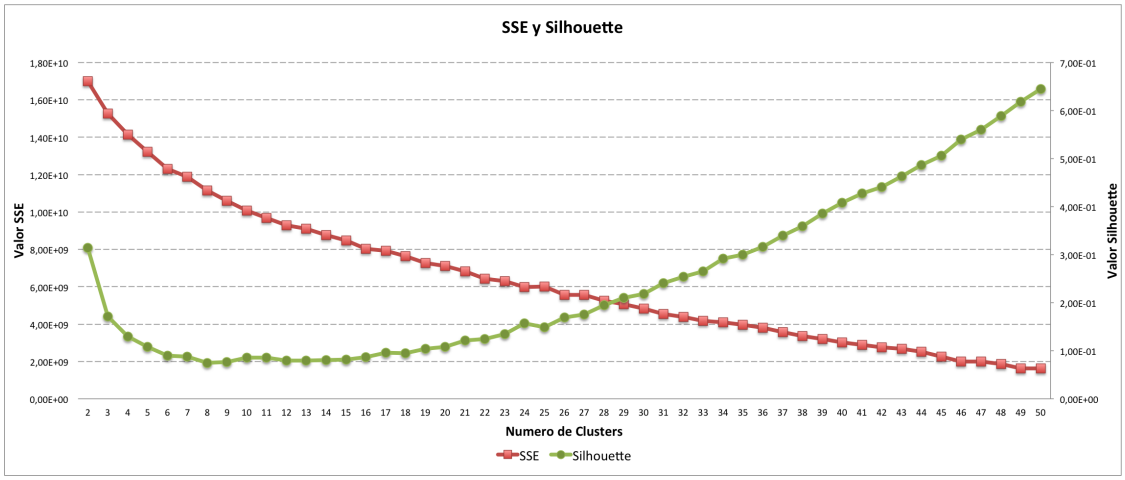

Figure 2: SSE y Silhouette

Como podemos ver, según el parámetro k va en aumento, las métricas van obteniendo distintos valores importantes. Son importantes porque están altamente correlacionados. Según el gráfico de arriba 3, podemos ver que las dos medidas tanto el SSE como el Silhouette, son inversamente proporcionales, ya que una de ellas mide la bondad (Silhouette) y la otra (SSE) mide el error.

Otra característica es que parece que cuanto mayor sea el número de clusters a generar, aparentemente mejor clusterización conseguimos. Esto se debe a que solamente tenemos 62 instancias en el dataset, y al generar un número de clusters grande, como pude ser 50, muchos de los clusters son únicamente de una instancia, dando a parecer que son buenos. Pero si nos fijamos en el Silhouette, vemos que se rompe la regla de que es mejor clusterización cuando generamos mas clusters, ya que de 2 a 10 clusters la tendencia del Silhouette es contraria. Por eso creemos que es mas significativo los valores de K pequeños que los mayores.

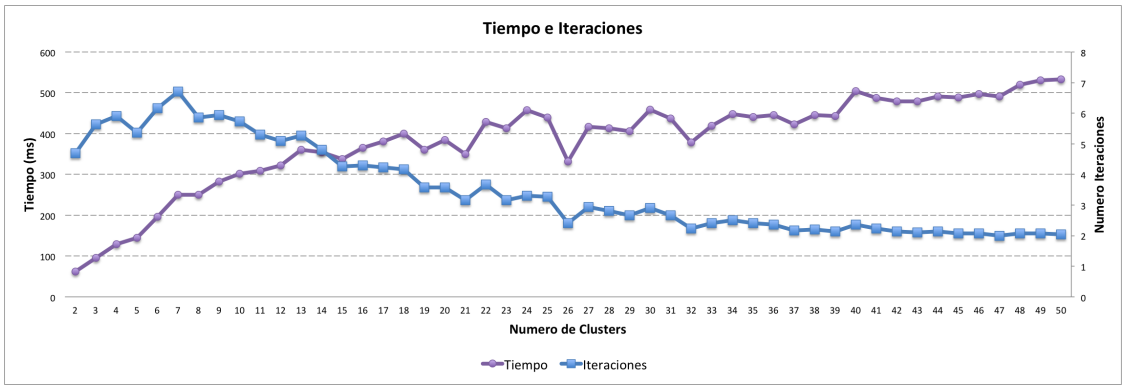

Figure 3: Tiempo y Vueltas

En cuanto al gráfico del tiempo e iteraciones, vemos que cuanto más grande es el parámetro k el tiempo de ejecución para cada partición es mayor. Esto quiere decir, que cuantos más clusters existan, más distancias se deben de calcular (distancia por cada instancia para cada cluster), por lo tanto, un mayor tiempo de ejecución.

También se aprecia que la línea de tiempo no es uniforme, lo que nos empujó a sacar el número de iteraciones que se efectuaban para cada valor de K. Como se puede ver, aunque sean proporcionalmente inversas, cosa que no dice mucho, los picos los tienen en los mismos valores y para el mismo sentido. Por ejemplo, cuando intentamos clusterizar con K 26, tenemos un pico bajo, se ejecuta más rápido que sus alrederores, pero esto se debe a que necesita menos iteraciones para converger.

Con esta gráfica, también vemos que necesita menos iteraciones a medida que la K aumenta. Pero al igual que pasaba con el Silhouette, esa regal no se cumple cuando la K varia entre 2 y 10. De donde podemos intuir que converge más rápido porque las instancias están distribuidas en 2 grupos mayormente.

4 Conclusiones:

El algoritmo K-Means tiene una gran limitación, y es que es muy dependiente del conjunto de datos. Es un algoritmo sencillo de implementar pero pese a su simplicidad es bastante eficiente. Los algoritmos de evaluación interna SSE(p) y Silhouette son bastante completos, aunque a nuestro parecer el Silhouette es el mejor de los dos.

En cuanto al dataset que se nos entregó (colon.arff) y los datos que hemos analizado en el punto anterior con los gráficos, podemos concluir que lo mejor para este dataset es clusterizar en dos grupos. Ya que el Silhouette es bastante alto, el tiempo de ejecución es corto y mirando el número de iteraciones vemos que converge de manera rápida. Esto no quiere decir que cogiendo otro dataset se comporte igual.

También se podría decir que al tener la limitación de instancias, los clusterizados con muchos grupos no son representativos ya que se necesitarían bastantes más instancias para sacar conclusiones más deterministas. Y por ello preferimos coger los valores de cuando K es bajo.

5 Valoración Subjetiva:

Ieltzu: Ha sido una práctica bastante interesante en cuanto a programación. Al principio pensábamos que iba a ser más difícil implementar el algoritmo, pero una vez que empezamos salió todo bastante fluido. Nuestros mayores problemas fueron entender los algoritmos SSE(p) y el Silhouette. Por lo demás muy fácil. Al crear los gráficos tenemos una visión mucho mejor de como cambian los valores SSE y Silhouette según sus parámetros.

Mikel: Me ha parecido una práctica bastante interesante. Por un lado, al implementar el K-Means, te das cuenta de cómo funciona este método, por otro lado, al sacar las conclusiones tras las medidas de los gráficos, aprendes a sacar dichas conclusiones y comprendes como funcionan. Cabe decir, que lo más difícil de esta práctica ha sido implementar tanto el Silhouette como el SSE.

Maria:

Bibliografia

- https://es.wikipedia.org/wiki/K-means
- https://exceltotal.com/como-crear-un-grafico-en-excel/
- https://en.wikipedia.org/wiki/Silhouette_(clustering)