Búsqueda por Similitud

Profesor Heider Sanchez

El objetivo del laboratorio es aplicar la búsqueda por rango y la búsqueda de los k vecinos más cercano sobre un conjunto de vectores característicos.

Se toma como referencia la colección de imágenes de flores ***Iris*** *(https://archive.ics.uci.edu/ml/datasets/iris)*, en donde cada imagen es representada por un vector característico de 4 dimensiones que recoge información del ancho y largo del sépalo y del pétalo. Además, las imágenes están agrupadas en tres categorías: *versicolor, setosa y virginica*.

Iris Versicolor **Iris Setosa** **Iris Virginica**

P1. Búsqueda por Rango

Implementar en cualquier lenguaje de programación el algoritmo lineal de búsqueda por rango, el cual recibe como parámetro el objeto de consulta y un **radio de cobertura**. Luego usando la distancia Euclidiana (ED) se retorna todos los elementos que son cubiertos por el radio.

```
Algorithm RangeSearch(Q, r)
1.  result = [ ]
2.  for all objects Ci in the collection
3.      dist = ED(Q, Ci)
4.      if dist < r
5.          append(result, Ci )
6.      endif
7.  endfor
8.  return result
```

- Aplique la búsqueda para 3 elementos de la colección (Q_{15}, Q_{82}, Q_{121}) y para tres valores de radio ($r1 < r2 < r3$).
- El objeto de consulta debe ser retirado de la colección antes de aplicar la búsqueda.
- Para saber que valores de radio seleccionar, debe primero realizar un análisis de la distribución de las distancias computando N veces la distancia entre dos elementos aleatorios de la colección.
- Para evaluar la efectividad del resultado se debe usar la medida de Precisión ¿Cuántos de los objetos recuperados pertenecen a la misma categoría de la consulta?:

$$PR = \frac{\#ObjetosRelevantesRecuperados}{\#ObjetosRecuperados}$$

A continuación, se proporciona el cuadro que debe ser llenado por el alumno.

PR	Q_{15}	Q_{82}	Q_{121}
$r1 = 0.5$	1	1	1
$r2 = 1.5$	1	0.78	0.45
$r3 = 2.5$	1	0.56	0.47

P2. Búsqueda KNN

Usando los mismos objetos de consulta del ejercicio anterior, implementar y aplicar el algoritmo lineal de búsqueda de los k vecinos más cercanos (KNN) variando el k entre {2, 4, 8, 16, 32}.

```
Algorithm KnnSearch(Q, k)
1.  result = [ ]
2.  for all objects C_i in the collection
3.      dist = ED(Q, C_i)
4.      append(result, {C_i , dist})
5.  endfor
6.  orderByDist(result)
7.  return result[1:k]
```

**** La mejor forma de implementación es gestionando la lista de resultado como una cola de prioridad máxima. Analice la complejidad.**

PR	Q_{15}	Q_{82}	Q_{121}
$k = 2$	1	1	1
$k = 4$	1	1	1
$k = 8$	1	1	0.88
$k = 16$	1	1	0.62
$k = 32$	1	1	0.5

Preguntas:

1- ¿Cuál es la complejidad computacional de ambos métodos de búsqueda en función de cálculos de la ED?

La complejidad de ambos métodos, en función de cálculos de la ED, es de O(n) siendo n el número de objetos en la colección ya que ambos algoritmos calculan la ED entre la consulta y cada objeto en la colección.

2- ¿Cuál de los dos métodos de búsqueda usted usaría en un ambiente real de recuperación de la información? Sustente su respuesta.

En un ambiente real usaría el range search ya que al usar el KNN search se corre el riesgo de que si la consulta está muy alejada de los objetos en la colección retorne k objetos que no son similares a la consulta, cosa que podría considerarse un error teniendo en cuenta que el objetivo es obtener aquellos objetos que sean similares a la consulta. Si se usa range search en este caso, el resultado sería 0 objetos, lo que significa que no hay objetos en la colección similares a la consulta, la cual es una respuesta válida. Por otra parte, la desventaja del range

search es que, si no definimos correctamente el radio de búsqueda, el algoritmo puede retornar o muchos o muy pocos objetos. Sin embargo, esto se puede solucionar realizando un análisis previo de la distribución de distancias con el objetivo de escoger un radio que nos retorne un porcentaje estimado de los objetos en la colección dependiendo de lo que queramos.