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,*** en donde cada imagen es representada por un vector característico de 4 dimensiones que recoge información 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 C_i in the collection
3.      dist = ED(Q, C_i)
4.      if dist < r
5.          append(result, C_i )
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 qué 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}
$r1 = 1$	1	1	0.5
$r2 = 3$	0.9074074074074074	0.3769230769230769	0.5
$r3 = 5$	0.362962962962963	0.3310810810810811	0.3333333333333333

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 cercano (KNN) variando el k entre {2, 4, 8, 16, 32}.

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

**** Otra forma de implementación es gestionando la lista de resultado como una cola de prioridad máxima ¿Habría alguna mejora?**

Si, ya que no se tendría que ordenar la lista para poder devolverla sino que se van ordenando mientras se van insertando, lo que reduce el costo computacional.

PR	Q_{15}	Q_{82}	Q_{121}
$k = 2$	1	1	1
$k = 4$	1	1	1
$k = 8$	1	1	0.875
$k = 16$	1	1	0.625
$k = 32$	1	1	0.5

Preguntas:

1- ¿Cuál de los dos métodos de búsqueda es más fácil de implementar si tengo todos los parámetros definidos? ¿Y cuál es más eficiente?

En mi opinión, la búsqueda por rango fue más fácil de implementar porque lo único que se usa es la ED y se compara con un r fijo, en cambio, en el KNN tuve que buscar la forma de ordenar la lista donde cada elemento es un par. Gracias a los resultados, pude notar que el KNN es más eficiente porque nos devuelve resultados con alta precisión en la mayoría de los casos

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

Aunque el KNN sea más eficaz al momento de recuperar data relevante, me parece que podría caer en el error de devolver elementos que no quiero en el caso de que cada elemento esté bien separado del otro. Siento que con la búsqueda de rango, limito a mi gusto el espacio de búsqueda y así recuperar elementos con la similaridad que yo quiera y no elementos que, si bien son los más cercanos, pueden estar a una distancia considerable.