Recuperación de Información Multimedia

Búsquedas por Similitud

Juan Manuel Barrios
https://juan.cl/

Búsquedas por similitud en sistemas RIM

- Los documentos multimedia se representan por sus descriptores de contenido
- La similitud entre documentos corresponde a calcular similitud entre descriptores
- Se necesitan métodos eficientes para localizar descriptores cercanos
- A continuación se presentan definiciones básicas sobre búsquedas por similitud en sistemas de RIM

Búsquedas por similitud

- Sea $\mathcal{D}$ el espacio de los objetos (el dominio)
- Sea $\mathcal{R} \subseteq \mathcal{D}$ un conjunto de objetos, que llamaremos el espacio de búsqueda
- Sea $q \in \mathcal{D}$ un objeto de consulta
- Se definen las búsquedas:
  - Búsqueda exacta
  - Búsqueda por rango (range query)
  - Búsqueda del vecino más cercano (nearest neighbors query)

Búsqueda exacta

- Determinar si el objeto *q* está en el espacio de búsqueda *R*
- Algoritmo secuencial (linear scan):

```
foreach u_i ∈ R do
    if u_i = q then
        return true ;
    end
end
return false ;
```

- Se puede hacer más eficiente si los objetos se pueden ordenar de menor a mayor (ver TDA Diccionario)

Búsqueda por rango

- Recuperar los objetos del espacio de búsqueda a distancia menor o igual a *r* de *q*
  - *r* es el radio de tolerancia

$$\mathrm{R}(q, r) = \{u \in \mathcal{R}, d(u, q) \leq r\}$$

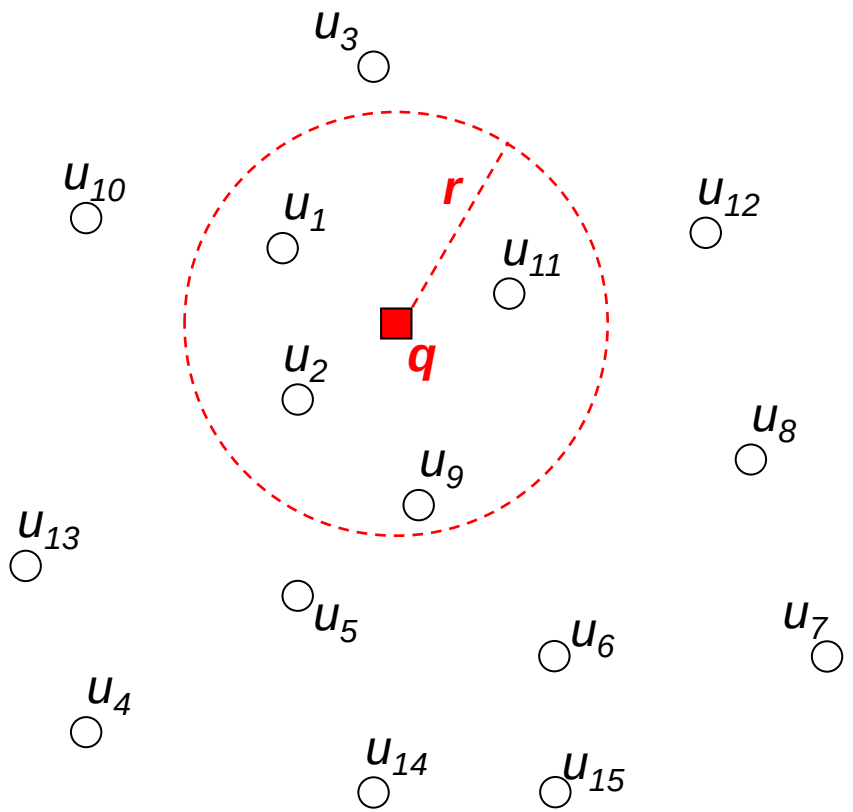
- Bola de consulta

$$\mathrm{B}(q, r) = \{x \in \mathcal{D}, d(x, q) \leq r\}$$

- La búsqueda por rango entrega los objetos de *R* que están dentro de la bola de consulta:

$$\mathrm{R}(q, r) = \mathcal{R} \cap \mathrm{B}(q, r)$$

Búsqueda por rango



$$\mathrm{R}(q, r) = \{u_1, u_2, u_9, u_{11}\}$$

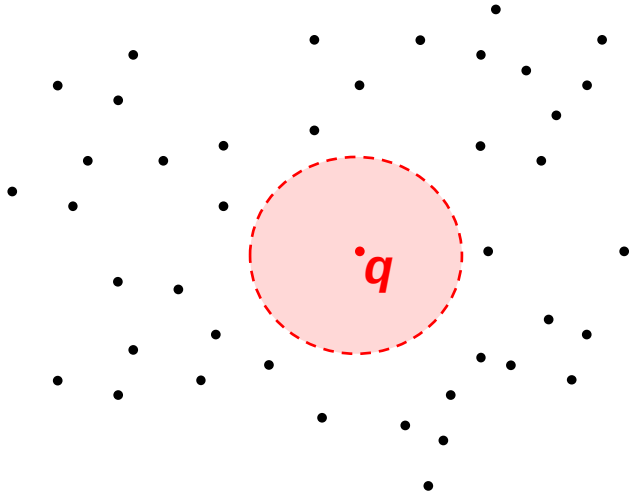
Búsqueda por rango

- Algoritmo secuencial (linear scan):

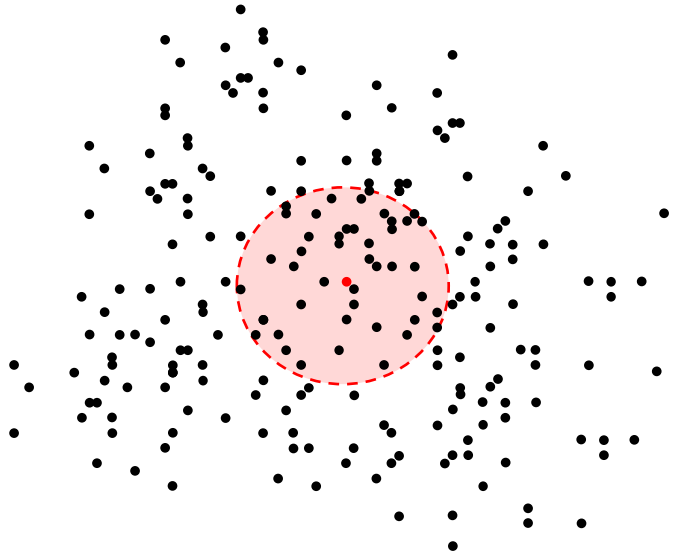
```
queue ← ∅ ;
foreach u_i ∈ R do
    if d(u_i, q) ≤ r then
        queue.Add(u_i) ;
    end
end
Print(queue);
```

Búsqueda por rango

- ¿Cómo fijar el radio de tolerancia?



r es muy pequeño: no encuentra nada



r es muy grande: encuentra demasiado

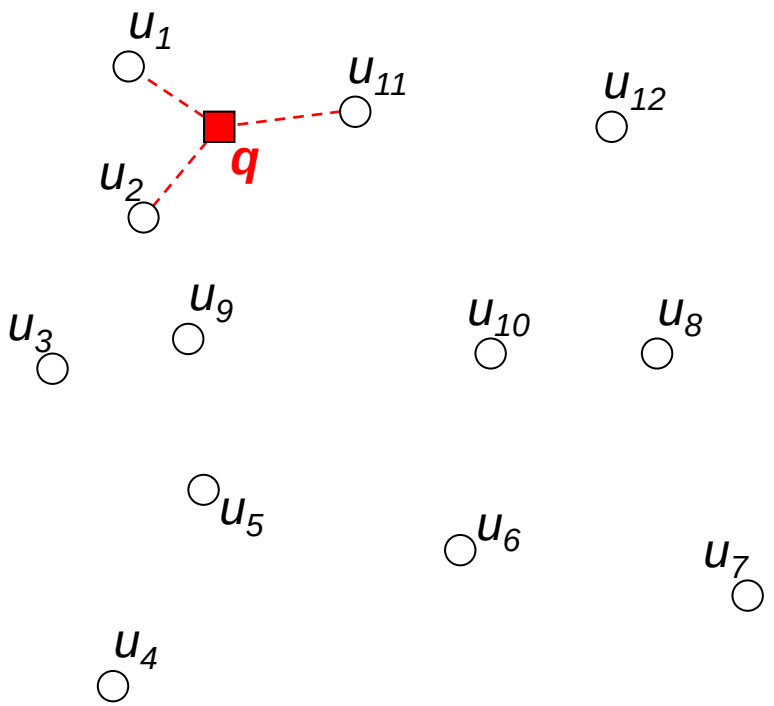
Búsqueda del vecino más cercano

- K Nearest Neighbors (k-NN)
  - Recuperar los *k* objetos del espacio de búsqueda con menor distancia a *q*

$$kNN(q) = \{\mathcal{C} \subseteq \mathcal{R}, |\mathcal{C}| = k \ \wedge \\ \forall x \in \mathcal{C}, y \in \mathcal{R} - \mathcal{C}, \ d(x,q) \leq d(y,q)\}$$

  - Existe más de una respuesta válida cuando hay varios objetos a la misma distancia de *q*

Búsqueda del vecino más cercano



$$3\mathrm{NN}(q) = \{u_1, u_2, u_{11}\}$$

Búsqueda del vecino más cercano

- Algoritmo secuencial 1-NN (linear scan):

```
candidate ← null ;
candidate_dist ← +∞ ;
foreach u_i ∈ R do
    dist ← d(u_i, q) ;
    if dist < candidate_dist then
        candidate ← u_i ;
        candidate_dist ← dist;
    end
end
Print(candidate);
```

Búsqueda k vecinos más cercanos

- Algoritmo secuencial k-NN (linear scan):

```
candidates ← ∅ ; // Priority Queue (Max-Heap)
foreach u_i ∈ R do
    dist ← d(u_i, q) ;
    if candidates.Size() < k then
        candidates.Add(dist, u_i) ;
    else if candidates.Get-Max().priority > dist then
        candidates.Remove-Max() ;
        candidates.Add(dist, u_i) ;
    end
end
Print(candidates);
```

Otros tipos de búsquedas

- **Combinaciones de criterios**
  - Consulta por rango + k-NN
  - Consulta k-NN con región de búsqueda (constrained k-NN)
- **Reverse nearest neighbor**
  - Buscar los objetos de *R* para los cuales *q* es uno de sus k-NN
- **Similarity join** (all nearest neighbors search)
  - Para dos conjuntos *Q* y *R* resolver una búsqueda por similitud (rango o knn) en *R* para cada elemento de *Q*
    - Cada q_i de *Q* se une con sus k-NN en *R* o con los objetos a distancia ≤ *r*
- **Self similarity join**
  - Resolver un similarity join cuando Q=R

Agenda

- **Repaso de estructuras de datos** (TDAs cola de prioridad y diccionario)
- **Índices multidimensionales:**
  - Indexar vectores (usar los valores de sus coordenadas para crear agrupaciones)
  - **Árboles**: Agrupan vectores en regiones espaciales ordenadas jerárquicamente
  - **Locality Sensitive Hashing**: Asignan vectores a una o más tablas de tamaño fijo
  - **Filling curves**: Convierten el espacio multi-dimensional en un espacio unidimensional
  - **Reducción de dimensionalidad**: PCA
- **Índices métricos:**
  - Indexar cualquier tipo de objeto (usar los valores dados por una función de distancia)
  - **Funciones de distancia**: Distintas funciones y sus propiedades métricas
  - **Tablas de pivotes**: Seleccionar objetos de referencia y compararlos con el resto
  - **Multi-métricas**: combinaciones lineales de distancias
- **Espacios de alta dimensionalidad**