IIC 2433 Minería de Datos

https://github.com/marcelomendoza/IIC2433

- VECINOS CERCANOS -

Vecinos cercanos

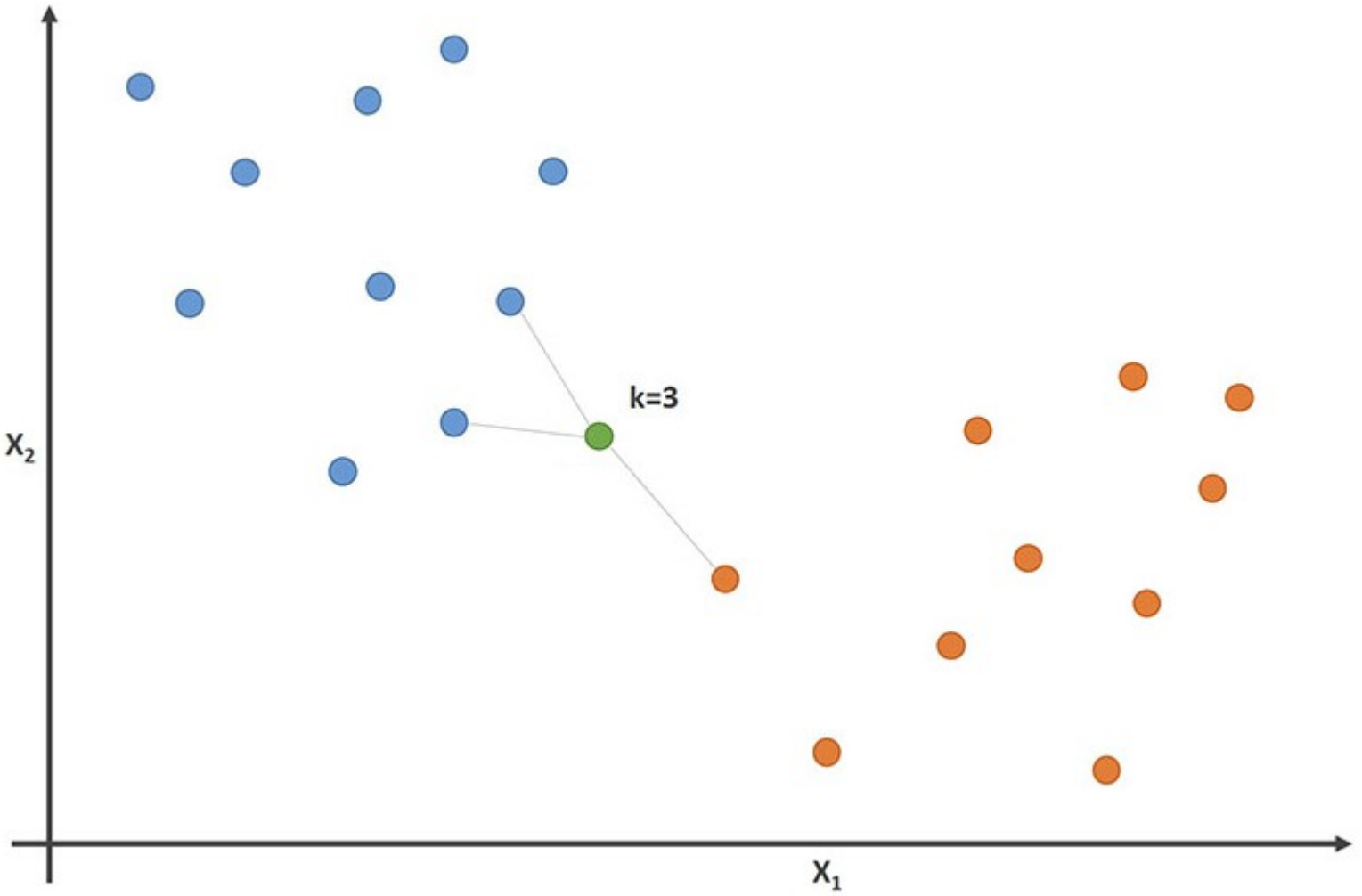
Los métodos de vecinos cercanos son la base de muchos otros métodos de aprendizaje.

El principio de los métodos de vecinos cercanos consiste en encontrar un número predefinido de muestras de entrenamiento que están más próximas en distancia a un nuevo punto. El número de muestras puede ser una constante definida por el usuario (k-vecinos más cercanos) o variar según la densidad local de puntos (aprendizaje basado en un radio determinado).

Aunque la distancia puede ser medida con cualquier métrica, la distancia euclidiana es la más utilizada.

A pesar de su simplicidad, los vecinos más cercanos han demostrado ser eficaces en una amplia variedad de problemas, incluidos la **búsqueda**, la **clasificación** y la **detección de anomalías**.

Vecinos cercanos



Vecinos cercanos

Para disponer de una estructura de vecinos cercana que podamos consultar, se usa alguno de los siguientes tres algoritmos:

- Pairwise metric (fuerza bruta, pocos datos)

- BallTree (alta dimensionalidad)

- kd-trees (baja dimensionalidad)

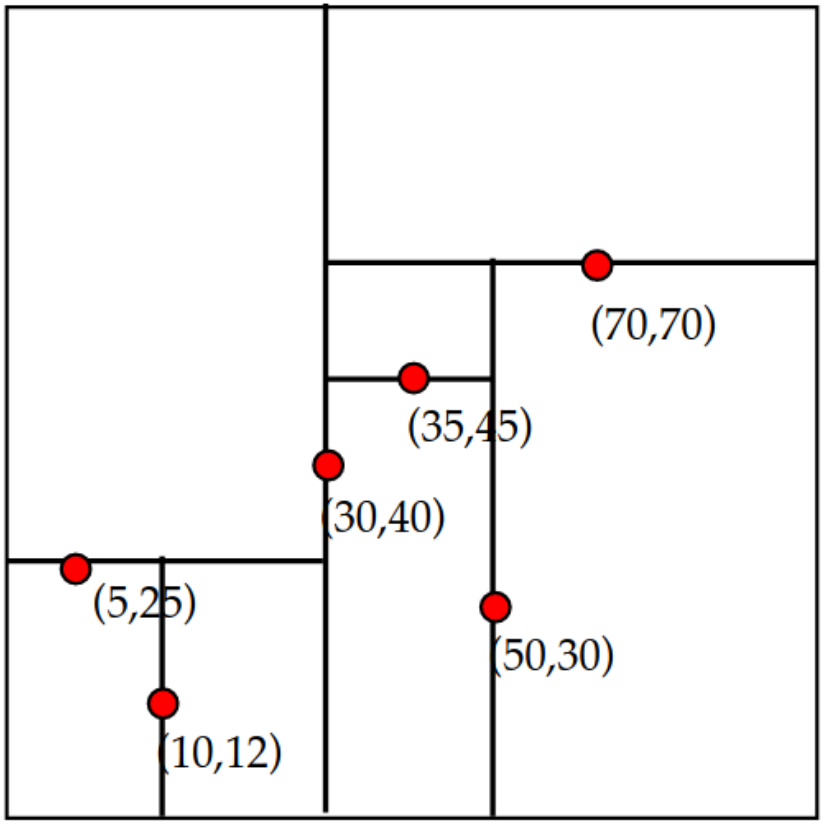
Vecinos cercanos (pairwise metric)

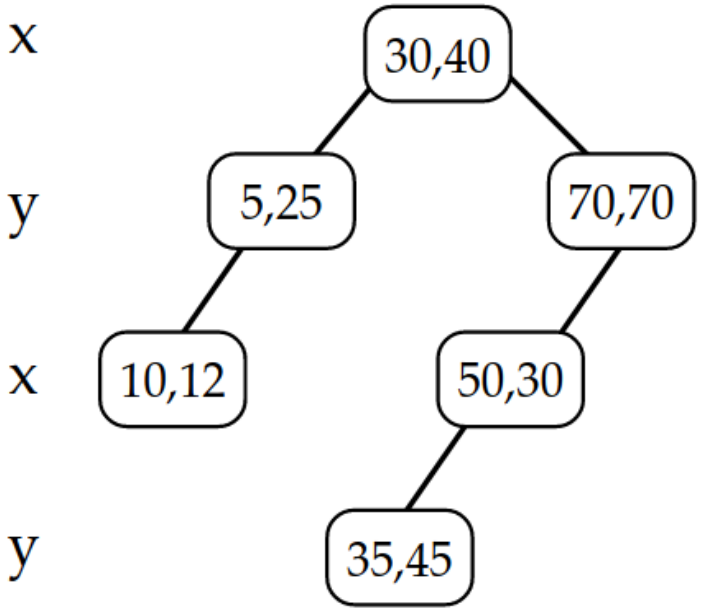
1.5															
1.4	1.6														
1.6	1.4	1.3													
1.7	1.4	1.5	1.5												
1.3	1.4	1.4	1.5	1.4											
1.6	1.3	1.4	1.4	1.5	1.8										
1.5	1.4	1.6	1.3	1.7	1.6	1.4									
1.4	1.3	1.4	1.5	1.2	1.4	1.3	1.5								
2.3	2.4	2.5	2.3	2.6	2.7	2.8	2.7	3.1							
2.9	2.8	2.9	3.0	2.9	3.1	2.9	3.1	3.0	1.5						
3.2	3.3	3.2	3.1	3.3	3.4	3.3	3.4	3.5	3.3	1.6					
3.3	3.4	3.2	3.2	3.3	3.4	3.2	3.3	3.5	3.6	1.4	1.7				
3.4	3.2	3.5	3.4	3.7	3.5	3.6	3.3	3.5	3.6	1.5	1.8	0.5			
4.2	4.1	4.1	4.1	4.1	4.1	4.1	4.1	4.1	4.1	1.7	1.6	0.3	0.5		
4.1	4.1	4.1	4.1	4.1	4.1	4.1	4.1	4.1	4.1	1.6	1.5	0.4	0.5	0.4	
5.9	6.2	6.2	5.8	6.1	6.0	6.1	5.9	5.8	6.0	2.3	2.3	2.5	2.3	2.4	2.5
6.1	6.3	6.2	5.8	6.1	6.0	6.1	5.9	5.8	6.0	3.1	2.7	2.6	2.3	2.5	2.6	3.0

Distancia Euclideana

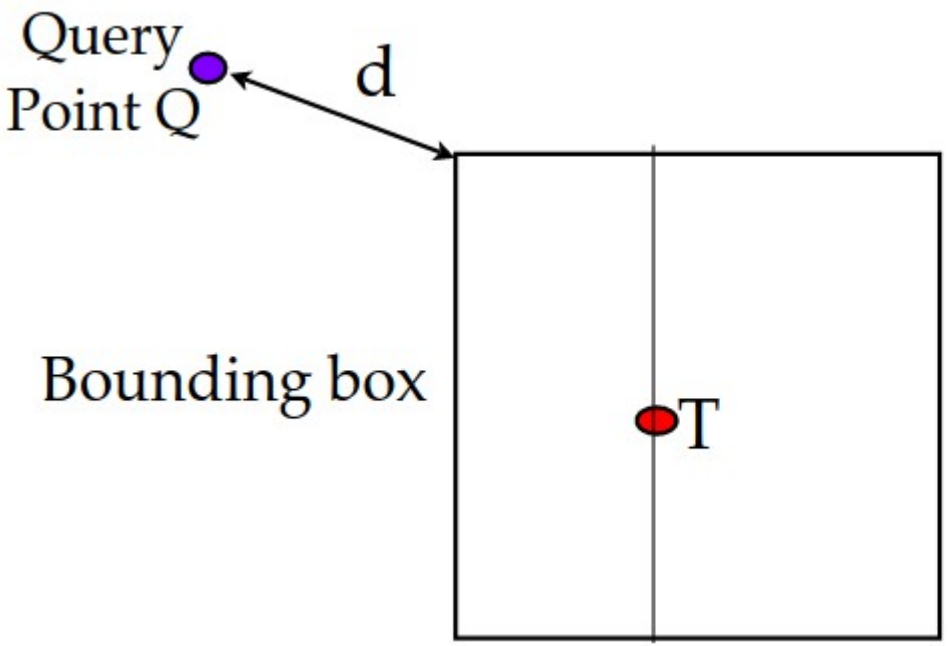
kd-trees

insert: (30,40), (5,25), (10,12), (70,70), (50,30), (35,45)





Búsqueda en kd-trees



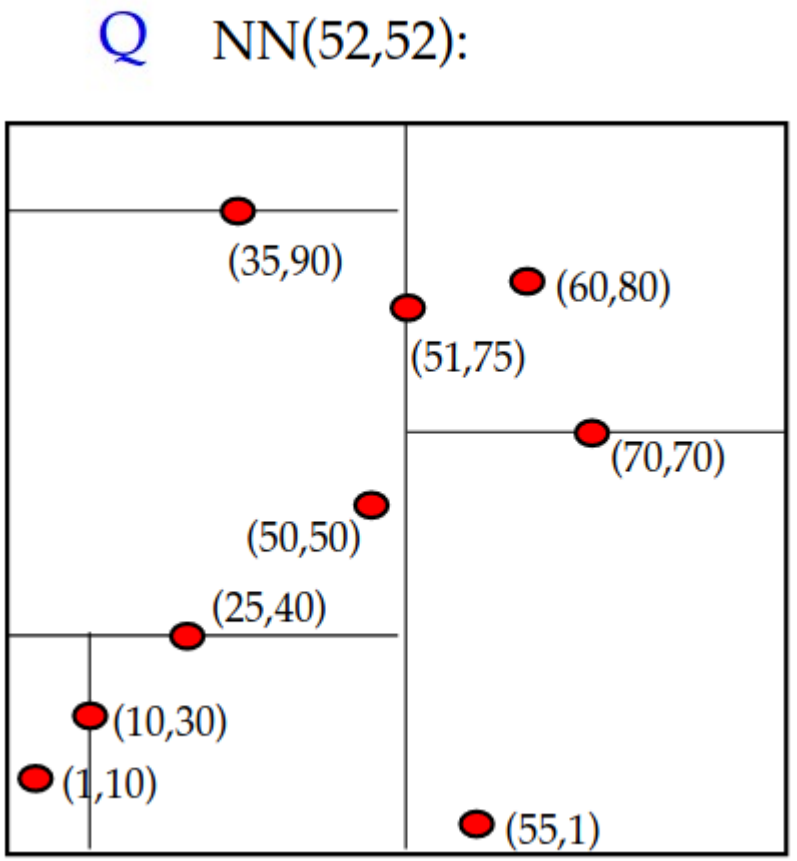
T

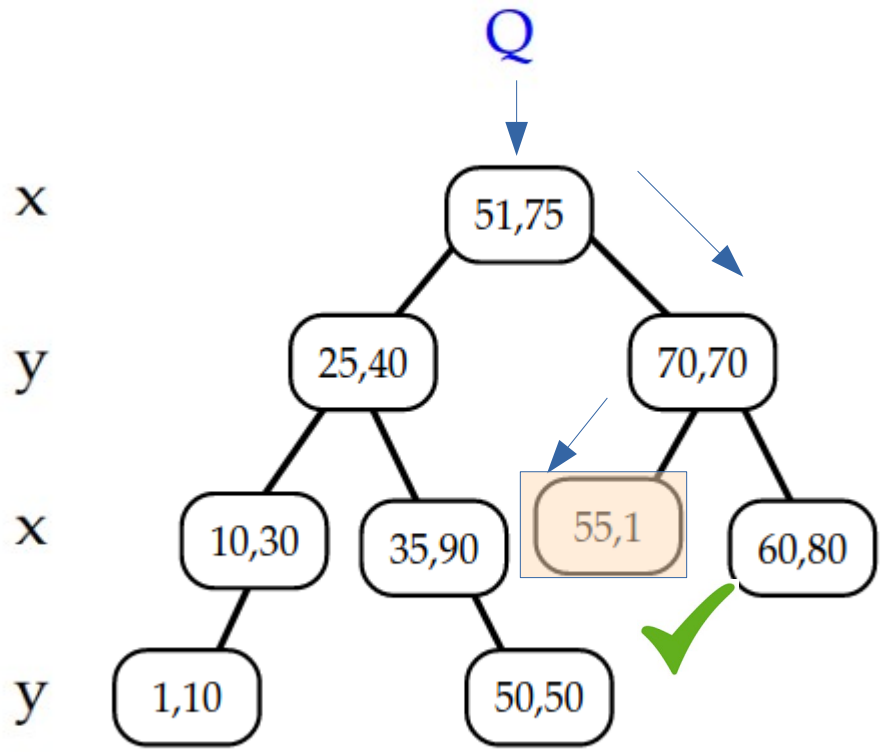
Si $d > dist(C, Q)$

descartamos el subárbol
en dirección contraria a Q

Búsqueda en kd-trees

1. Busca el punto usando el algoritmo de inserción.



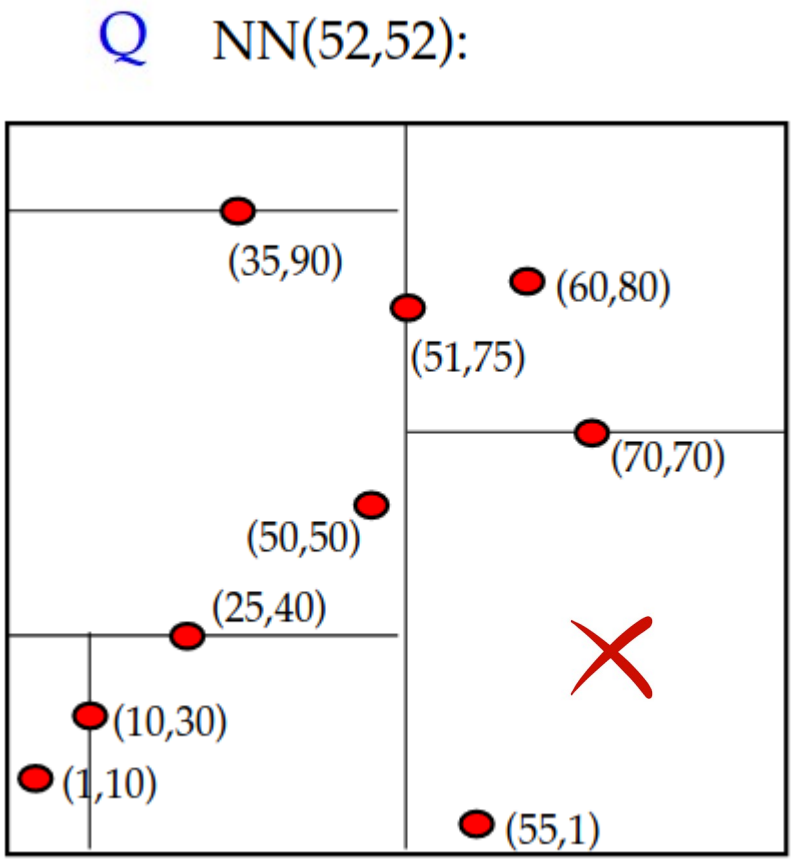


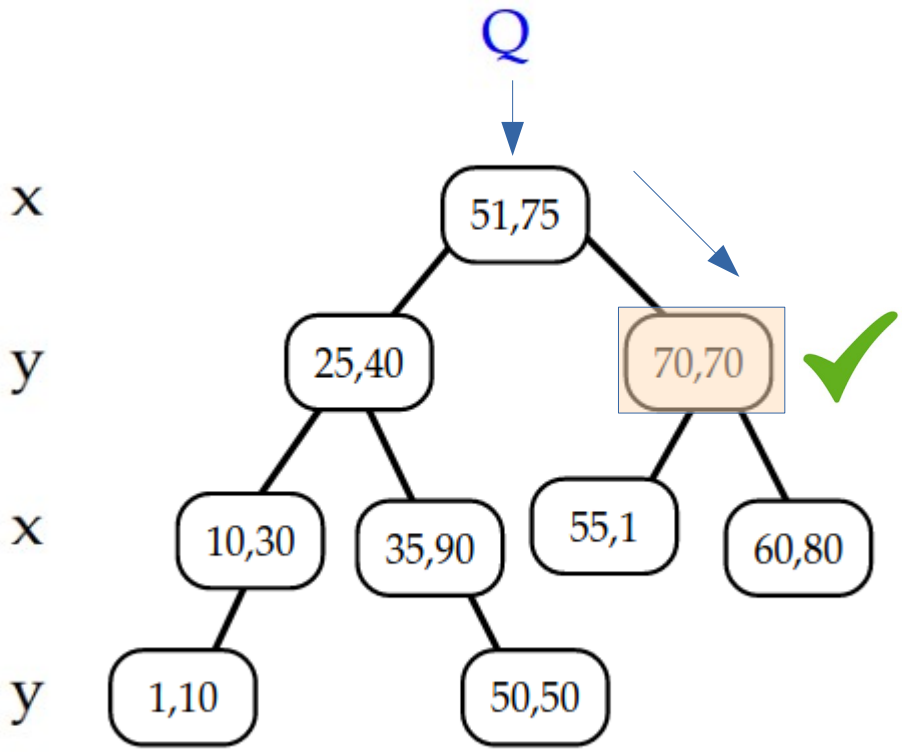
d((52,52),(55,1)) = 51

Búsqueda en kd-trees

2. Usar regla de poda

d((52,52),(55,1)) = 51



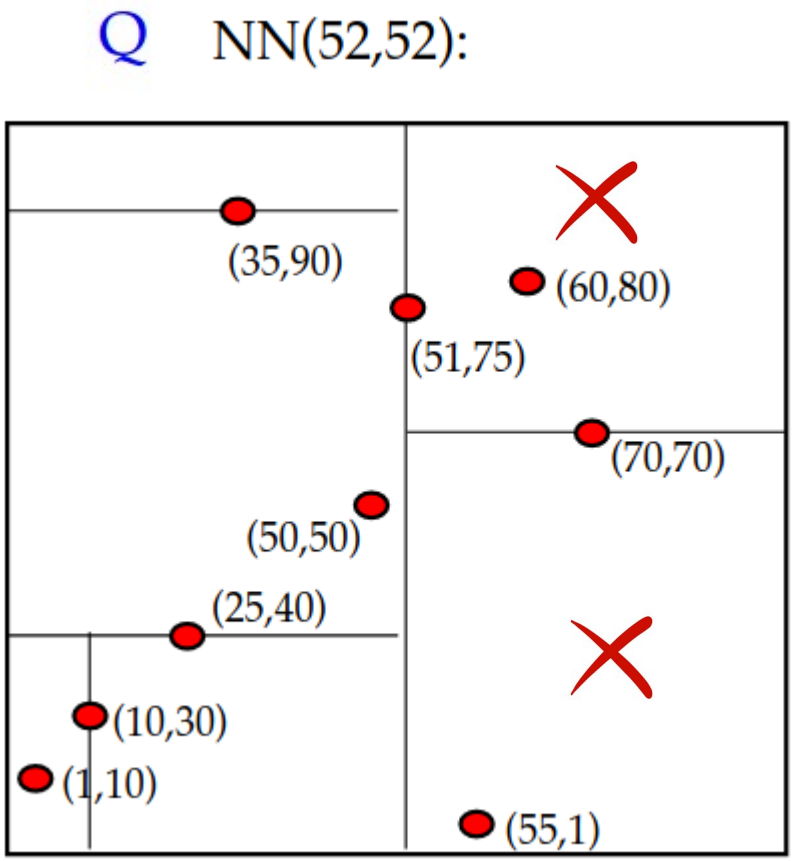


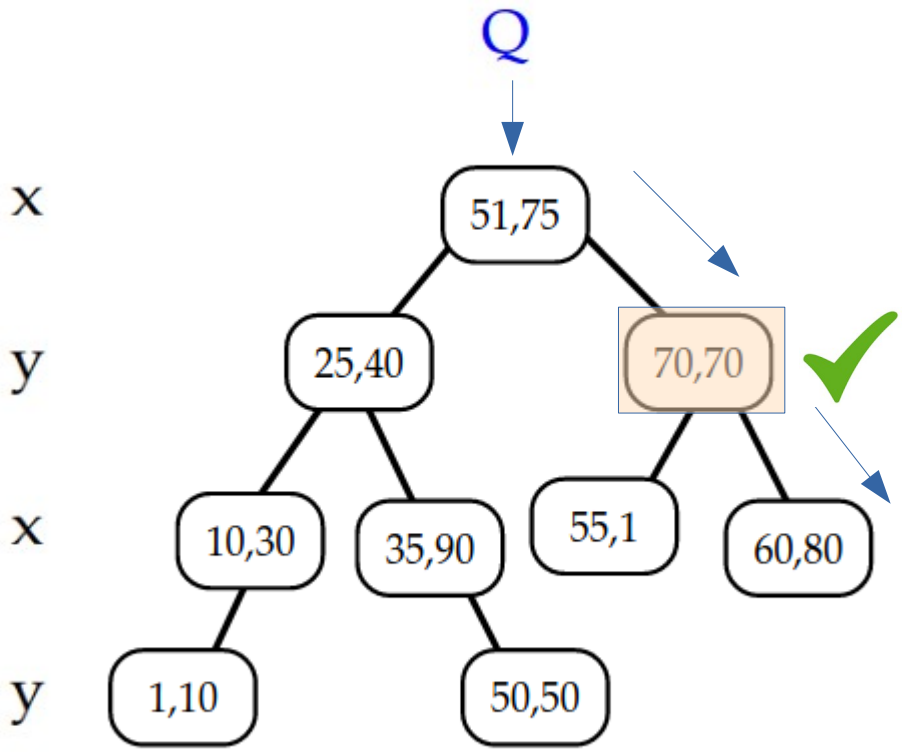
d((52,52),(70,70)) = 25

Búsqueda en kd-trees

2. Usar regla de poda

d((52,52),(70,70)) = 25



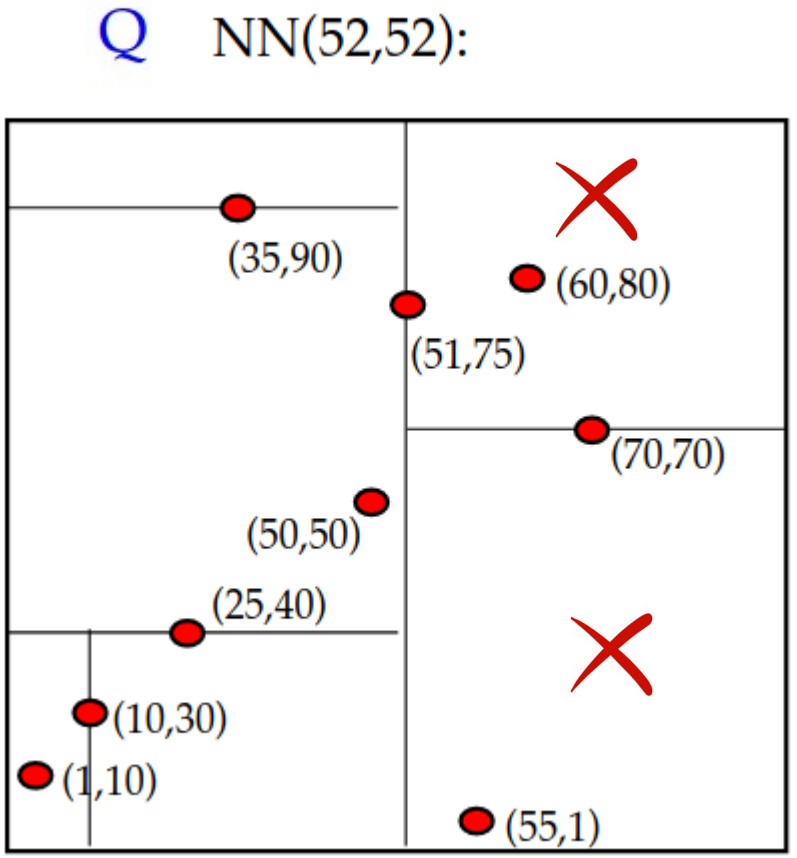


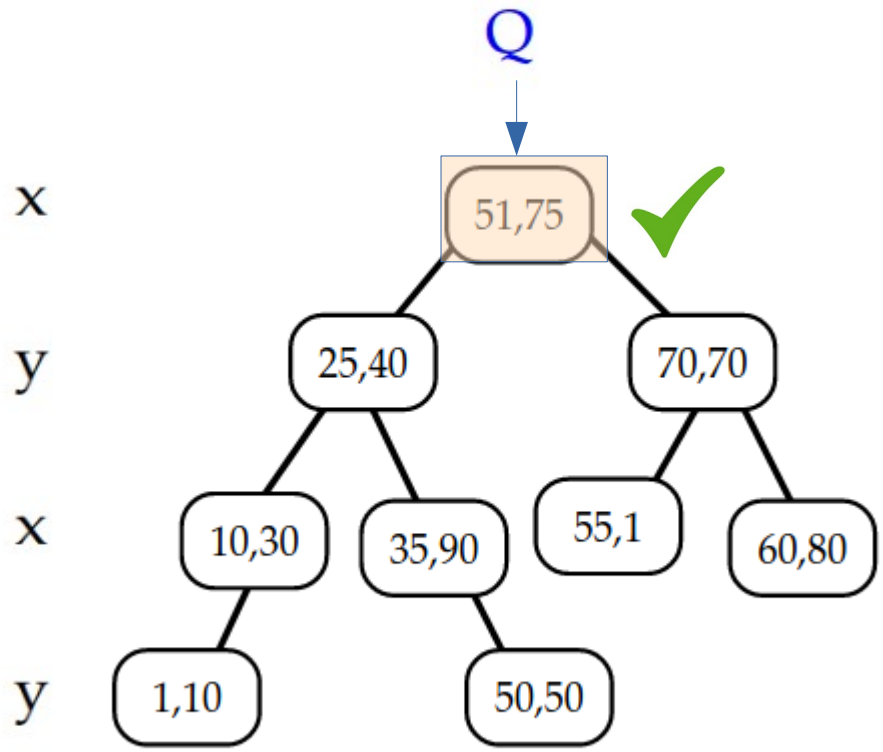
d((52,52),(60,80)) = 29

Búsqueda en kd-trees

2. Usar regla de poda

d((52,52),(70,70)) = 25



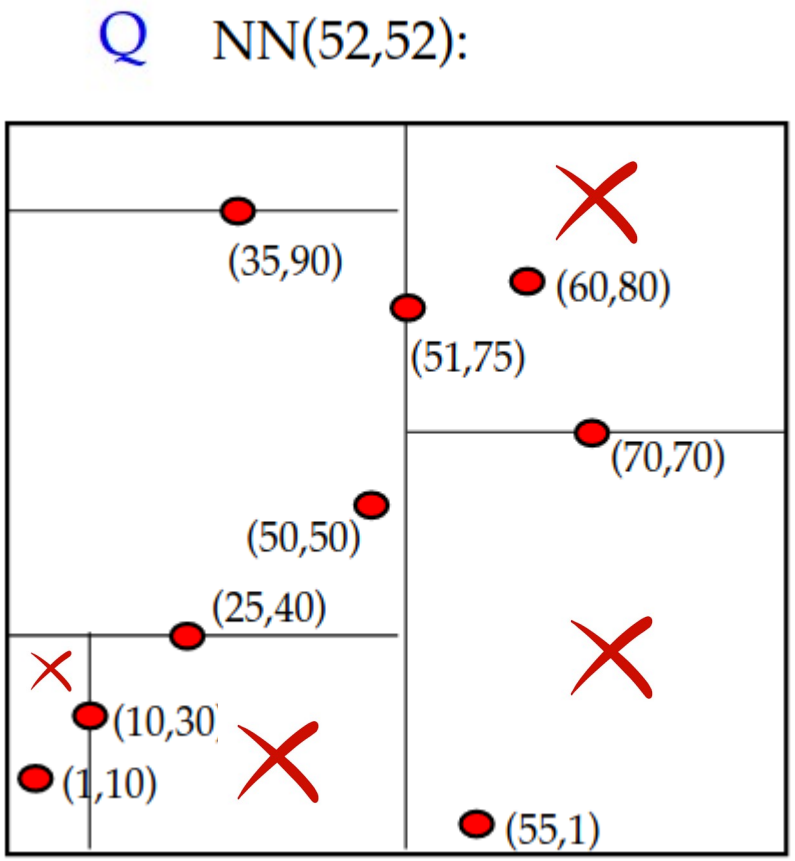


d((52,52),(51,75)) = 23

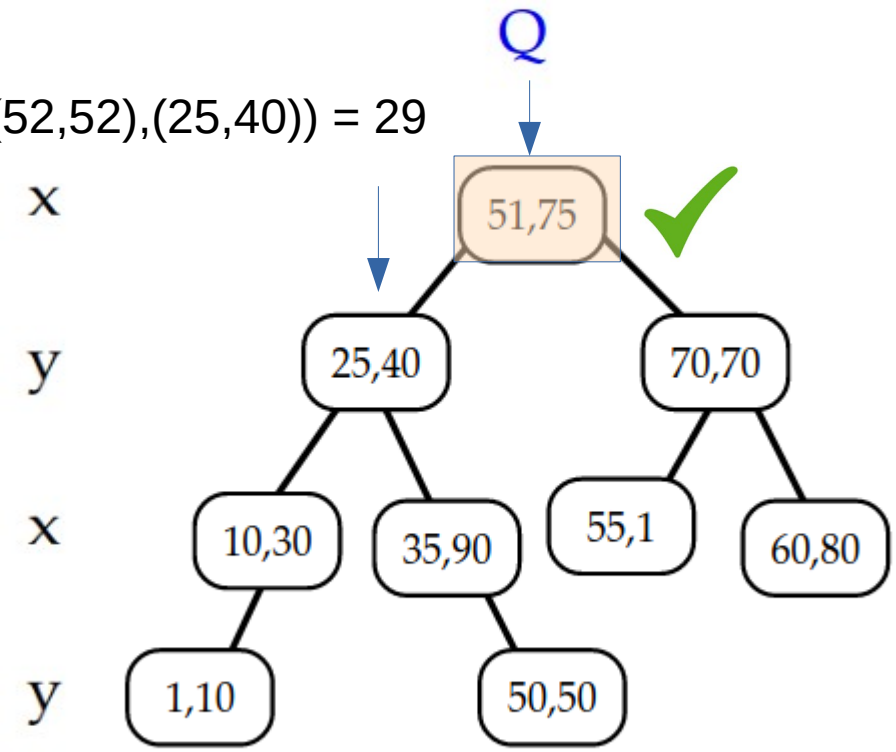
Búsqueda en kd-trees

2. Usar regla de poda

d((52,52),(51,75)) = 23



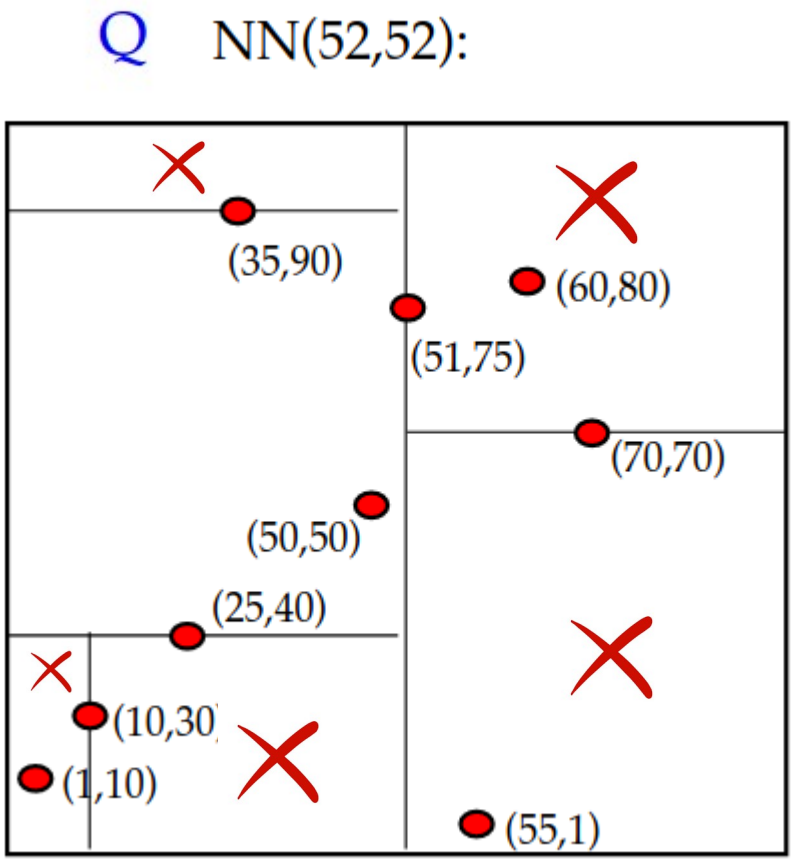
d((52,52),(25,40)) = 29

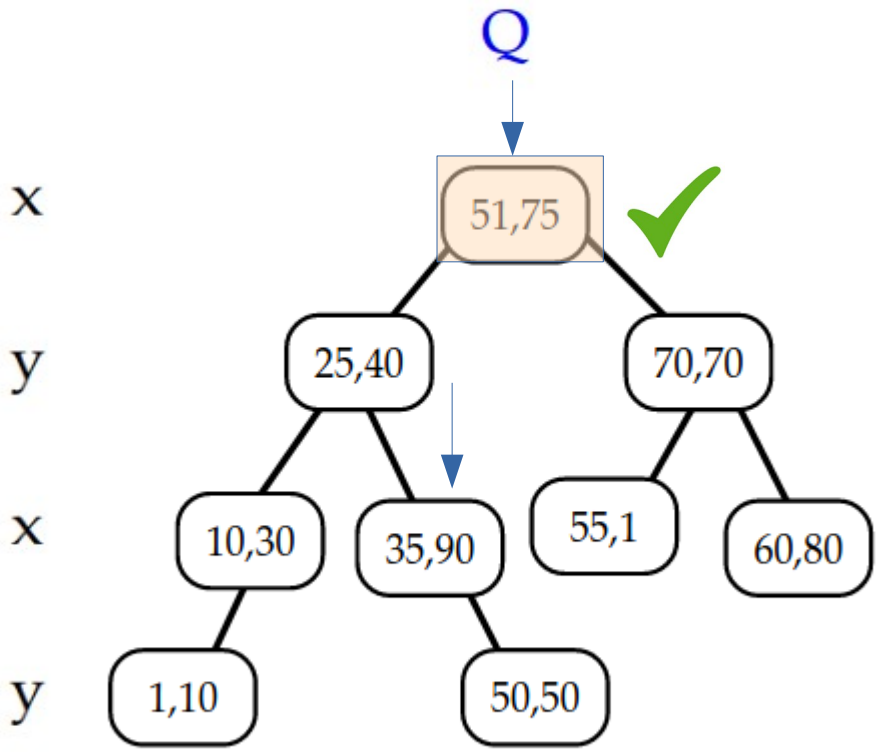


Búsqueda en kd-trees

2. Usar regla de poda

d((52,52),(51,75)) = 23



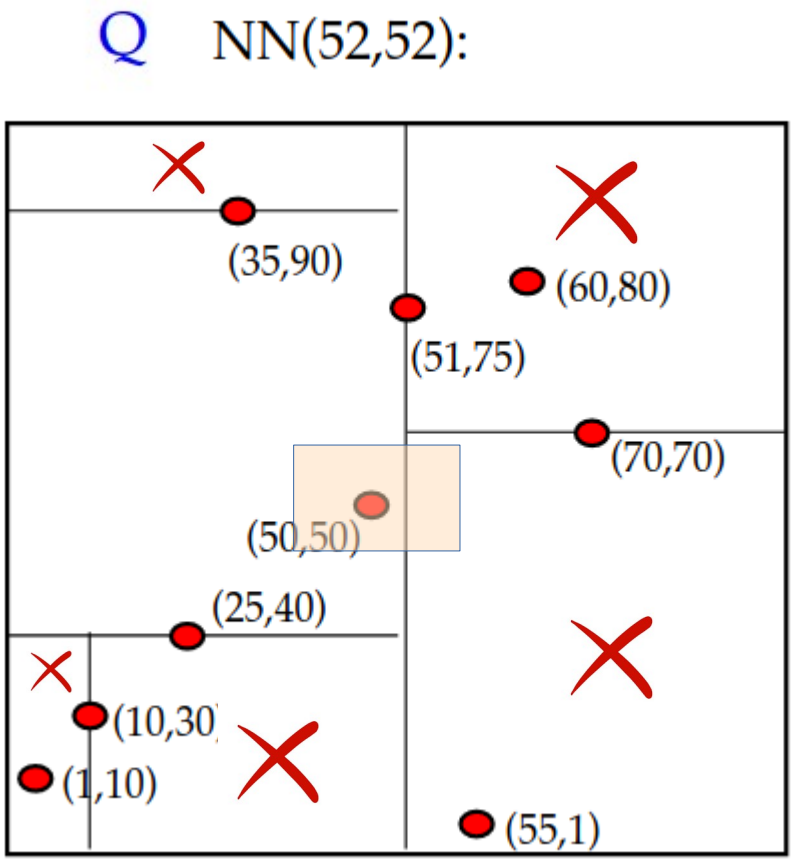


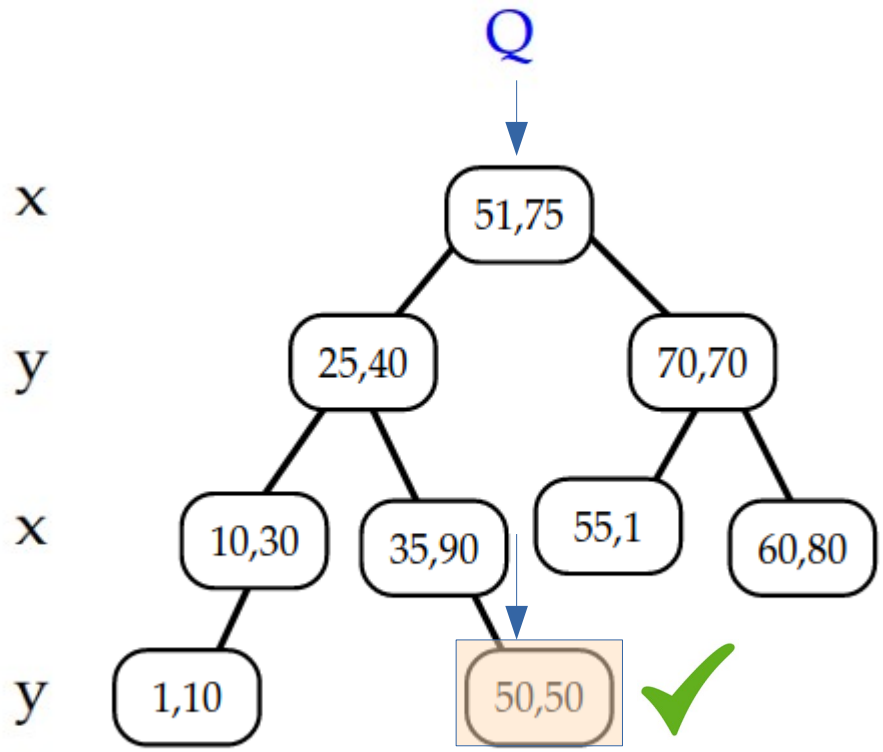
d((52,52),(35,90)) = 41

Búsqueda en kd-trees

2. Usar regla de poda

d((52,52),(51,75)) = 23





d((52,52),(50,50)) = 2.8

Ball-trees

Las búsquedas en kd-trees se hacen muy ineficientes si aumenta la dimensionalidad.

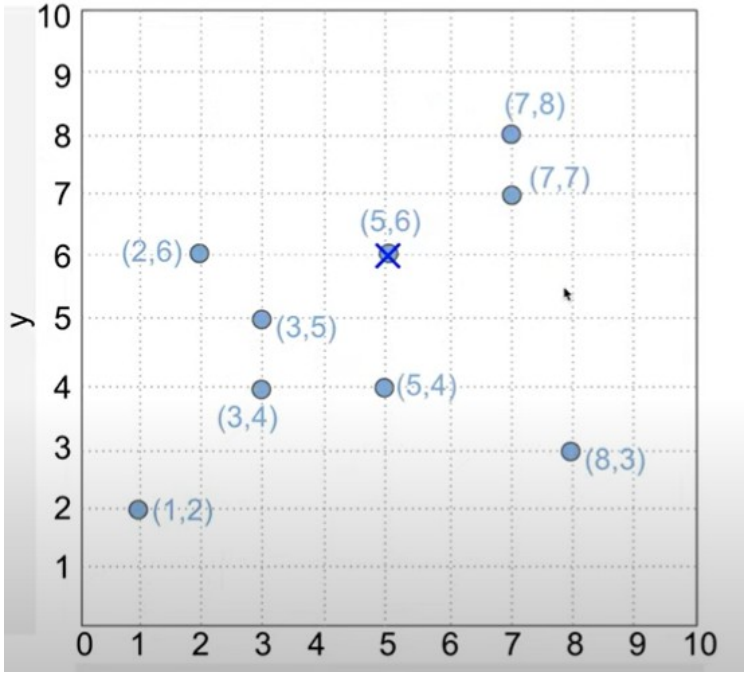
Cambiamos por un método basado en radios.

- Se crea un árbol binario. Cada nodo define la esfera más pequeña que contiene los puntos de su subárbol.

- El criterio de construcción da lugar a un invariante que usaremos en búsqueda:

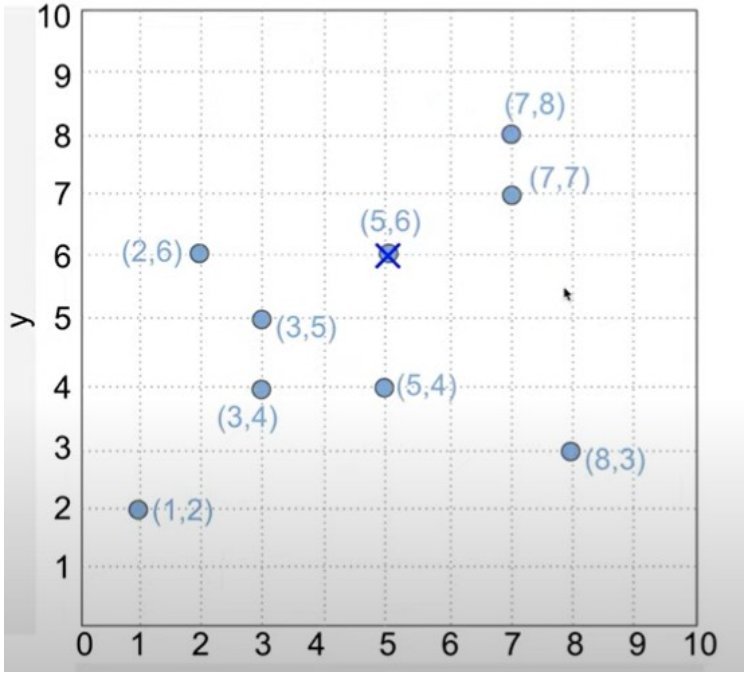
> Dado un punto externo *t* a una esfera B, su distancia a cualquier punto de B será mayor o igual que la distancia a la superficie de B.

Ball-trees (construcción)

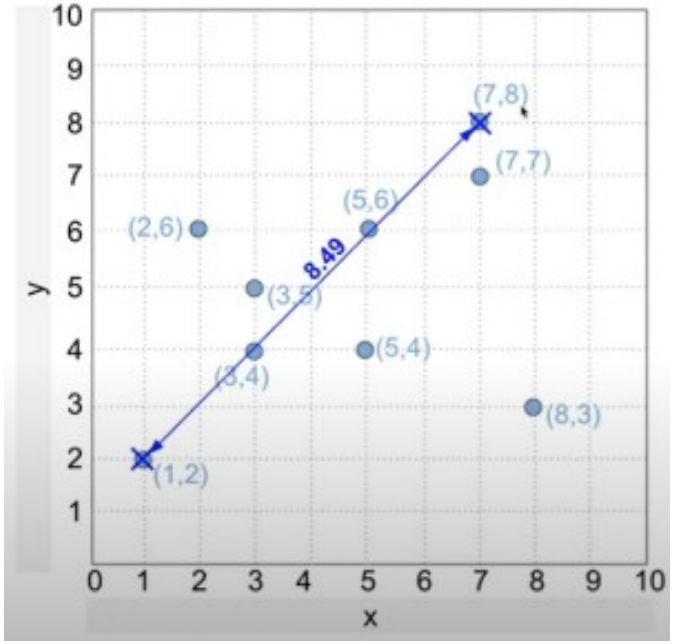


(a) Seleccionamos una semilla

Ball-trees (construcción)

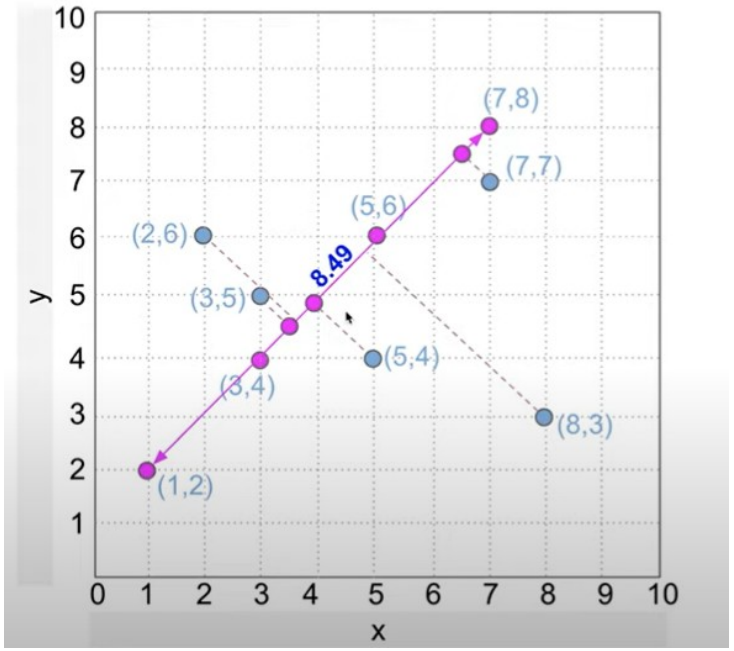


(a) Seleccionamos una semilla



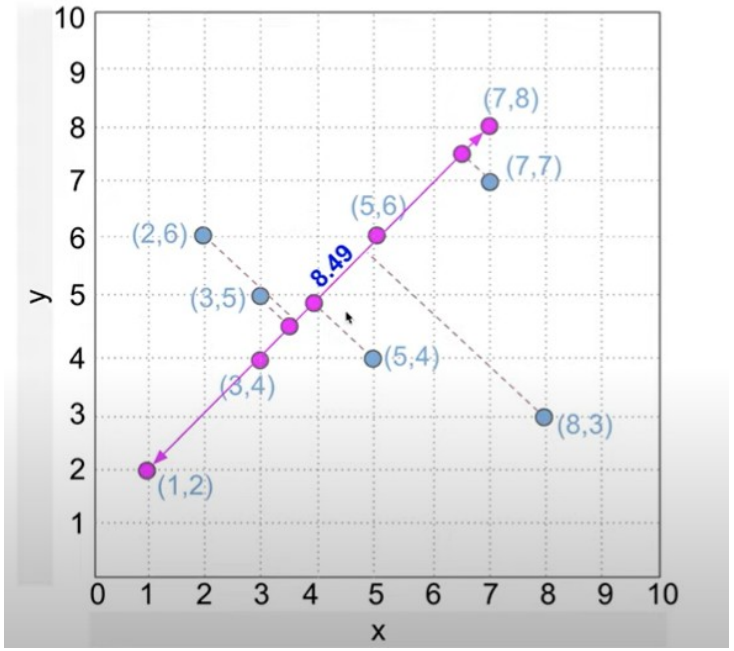
(b) Buscamos el par de puntos más lejanos a la semilla

Ball-trees (construcción)

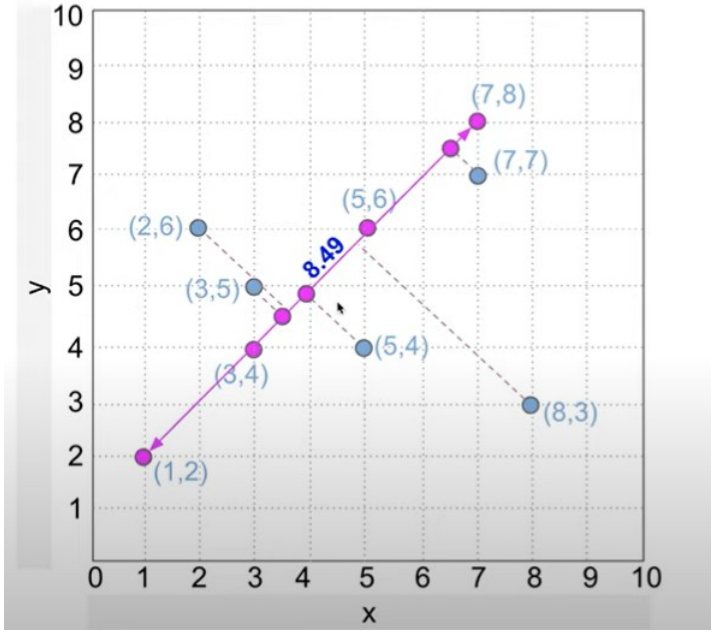


(c) Proyectamos los puntos a la recta que une los puntos más lejanos

Ball-trees (construcción)

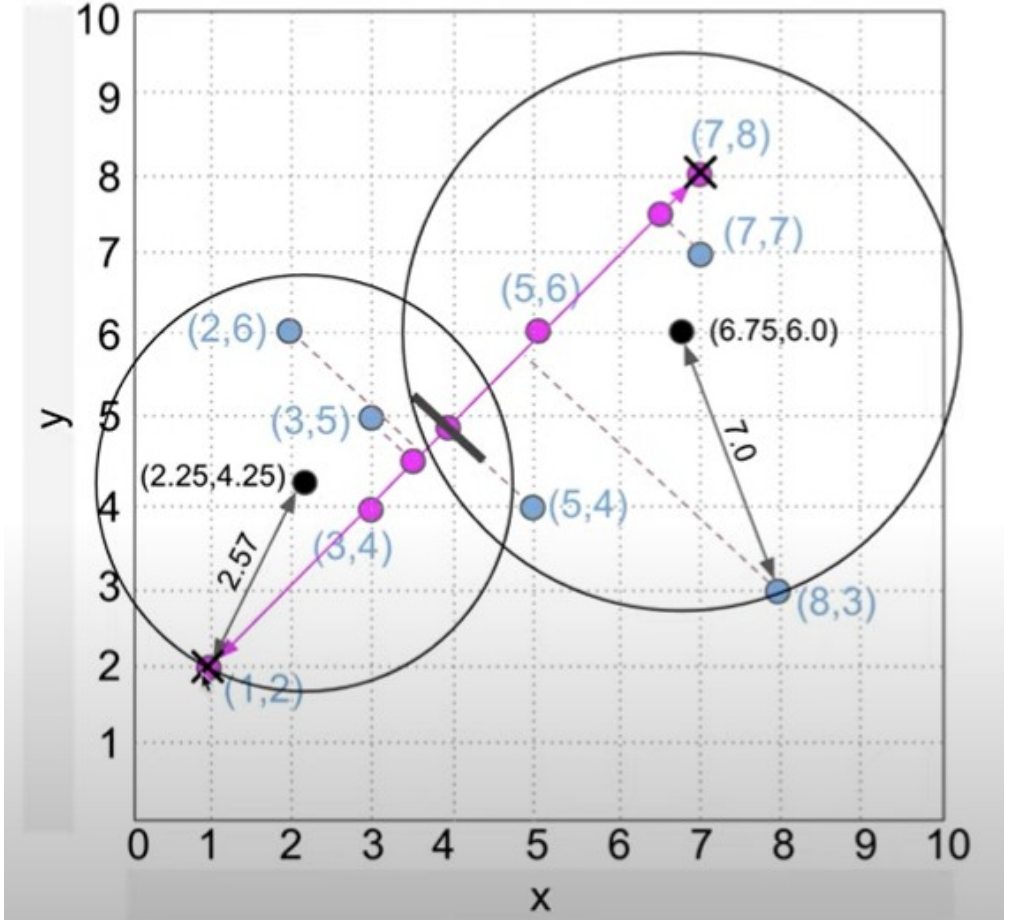


(c) Proyectamos los puntos a la recta que une los puntos más lejanos



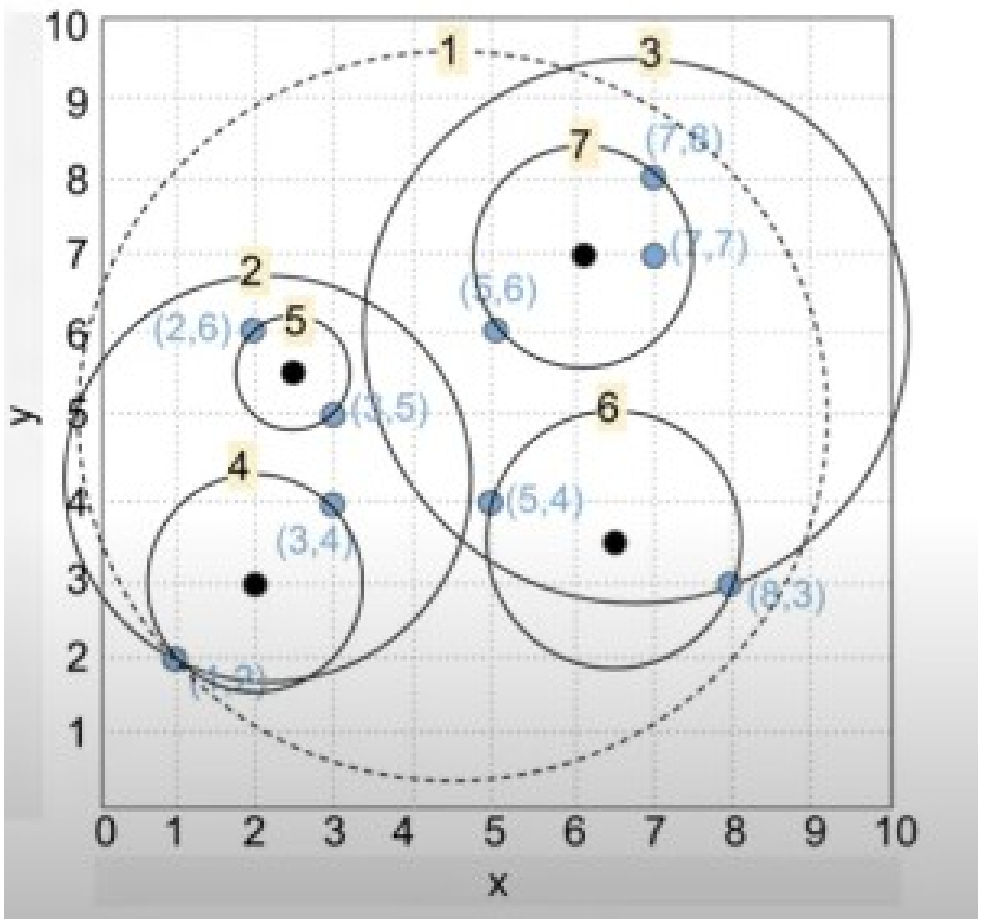
(d) Dividimos el espacio en torno de la mediana

Ball-trees (construcción)



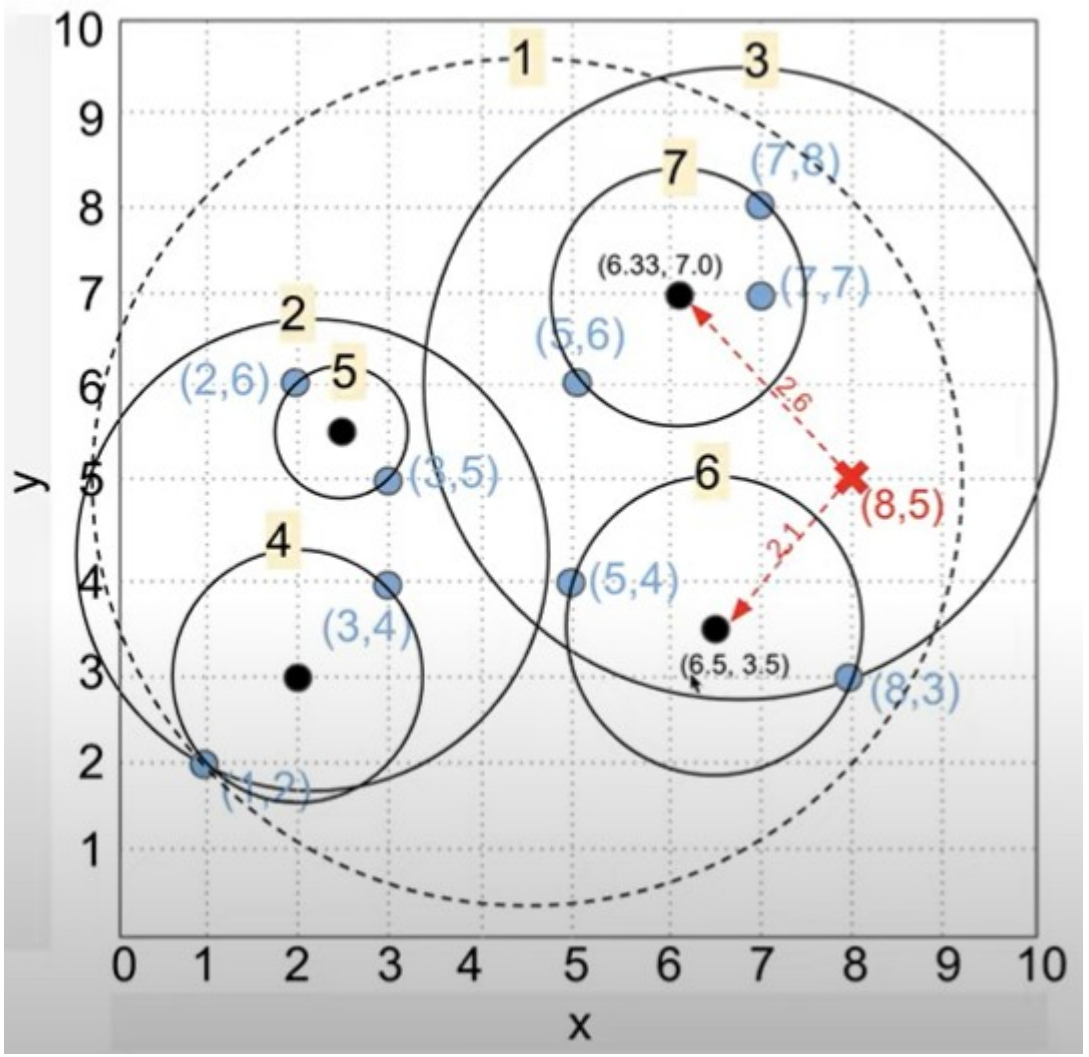
(e) En cada mitad, calculamos el centroide. Desde el centroide buscamos el punto más lejano de la mitad.

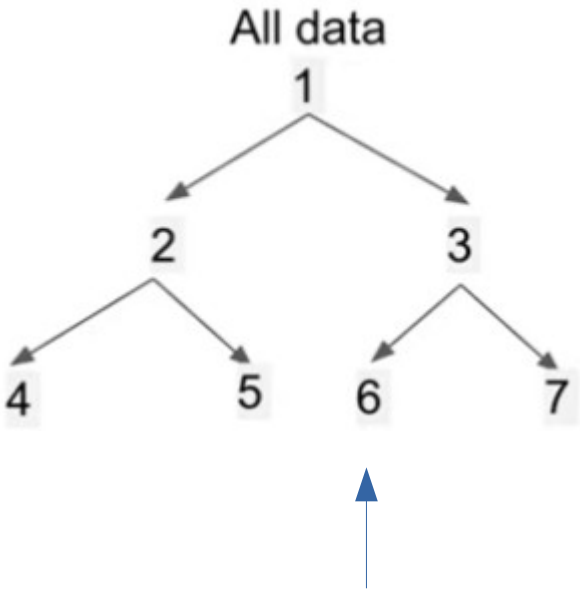
Ball-trees (construcción)



(f) Repetimos el procedimiento

Búsqueda en Ball-trees



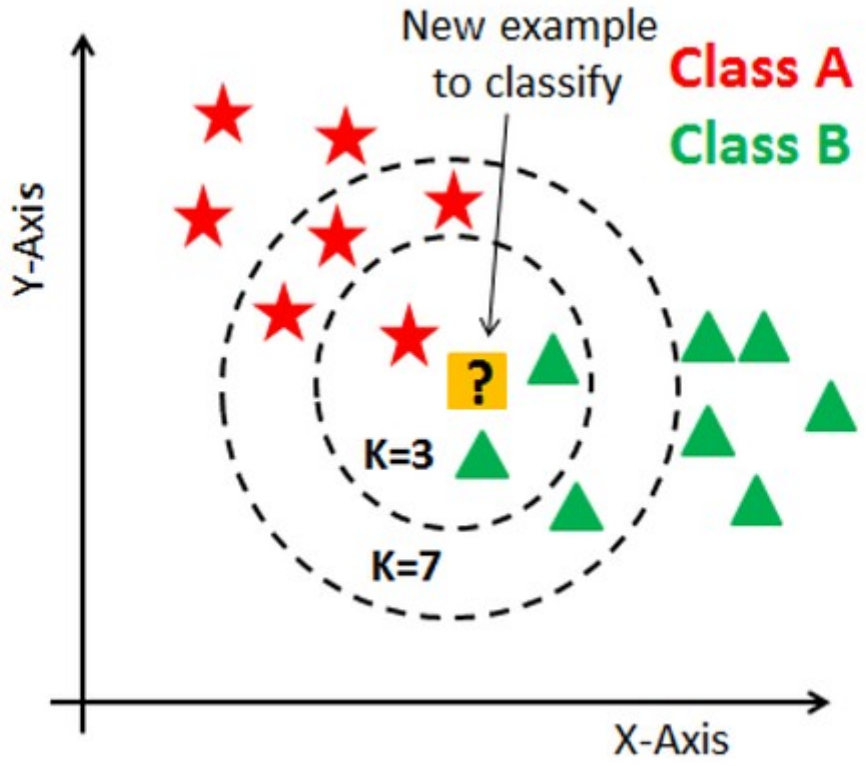


Se busca la esfera de centroide más cercano.

- APLICACIONES DE VECINOS CERCANOS -

Vecinos cercanos para clasificación

- Dado un nuevo ejemplo, buscamos sus k vecinos más cercanos y lo asignamos a la clase mayoritaria.



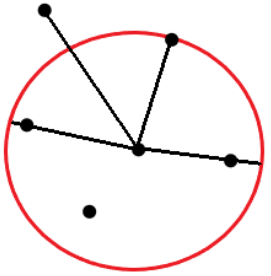
Vecinos cercanos para detección de outliers

- Dado un nuevo ejemplo, buscamos sus k vecinos más cercanos.
- Su distancia a los vecinos es usada para estimar su densidad.
- Se compara su densidad con la densidad de los vecinos.

Definimos k-distance(B): distancia de B a su kNN.

Luego definimos alcanzabilidad:

$$\text{reachability-distance}_k(A,B)=\max\{k\text{-distance}(B),\ d(A,B)\}$$



Vecinos cercanos para detección de outliers

Definimos la densidad de alcanzabilidad local:

$$LRD(p) = 1/\left(\frac{\sum_{q \epsilon knn(p)} reach-dist(p.q)}{||k-neighborhood||}\right)$$

Y luego el Local Outlier Factor (LOF):

$$LOF(p) = \left(\frac{\sum_{q\ \epsilon\ knn(p)} \frac{LRD(q)}{LRD(p)}}{||k-neighborhood||}\right)$$

Vecinos cercanos para detección de outliers