IIC 2433 Minería de Datos

https://github.com/marcelomendoza/IIC2433

Cierre de la clase 2 – t-SNE y UMAP

t-SNE y UMAP:

¿Qué es lo que preserva t-SNE del espacio original?

¿Para qué se usa el kernel Gaussiano en t-SNE?

¿Cuál es la función objetivo que usa t-SNE?

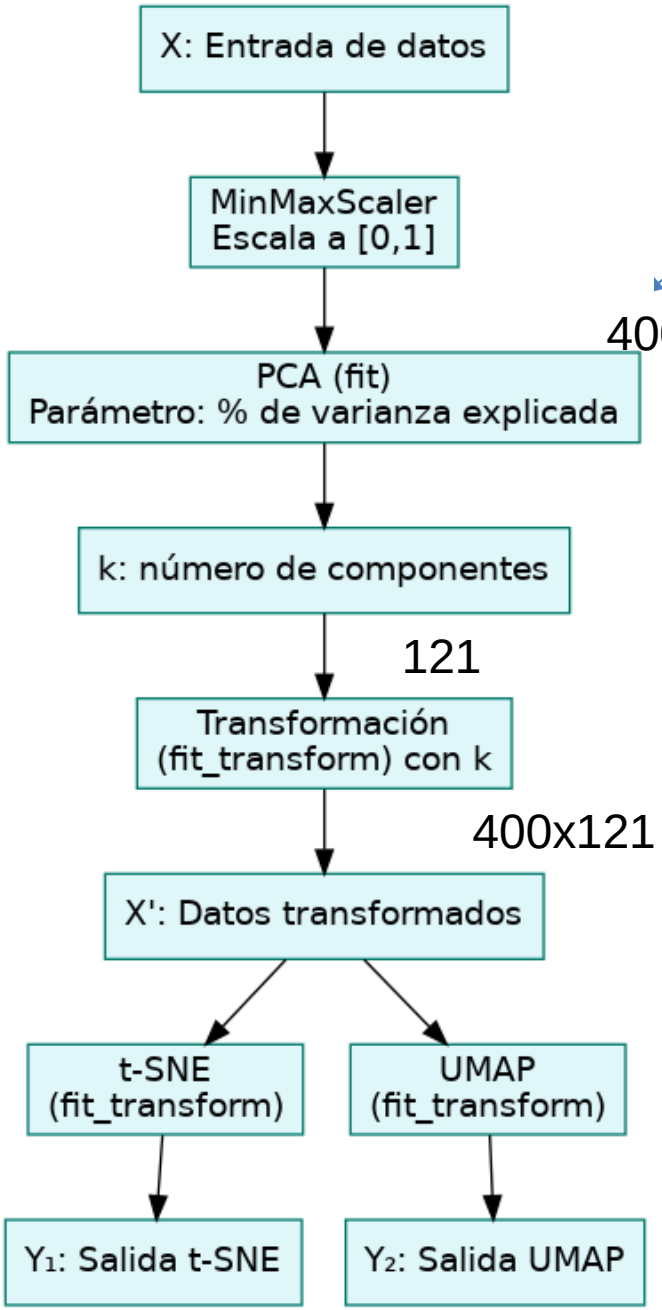
Intuitivamente ¿Qué es la perplejidad de un modelo como t-SNE?

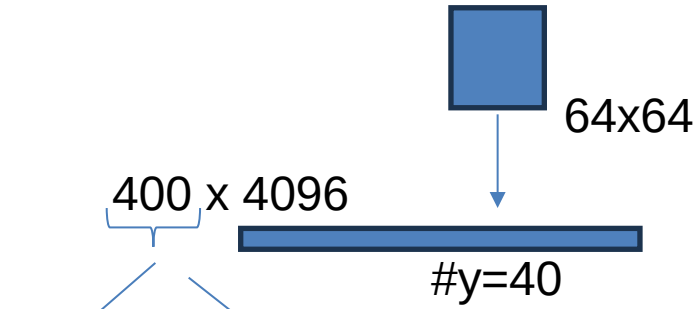
Comparación con PCA:

PCA es una técnica de reducción de dimensionalidad. Sin embargo, t-SNE y UMAP también reducen dimensionalidad. ¿En qué se diferencian según su uso?

Cierre de la clase 2 – actividad formativa

Trabajamos en una actividad formativa con PCA, t-SNE y UMAP.
Lo que hicieron fue definir un pipeline:





P >= 10

- VECINOS CERCANOS -

Vecinos cercanos

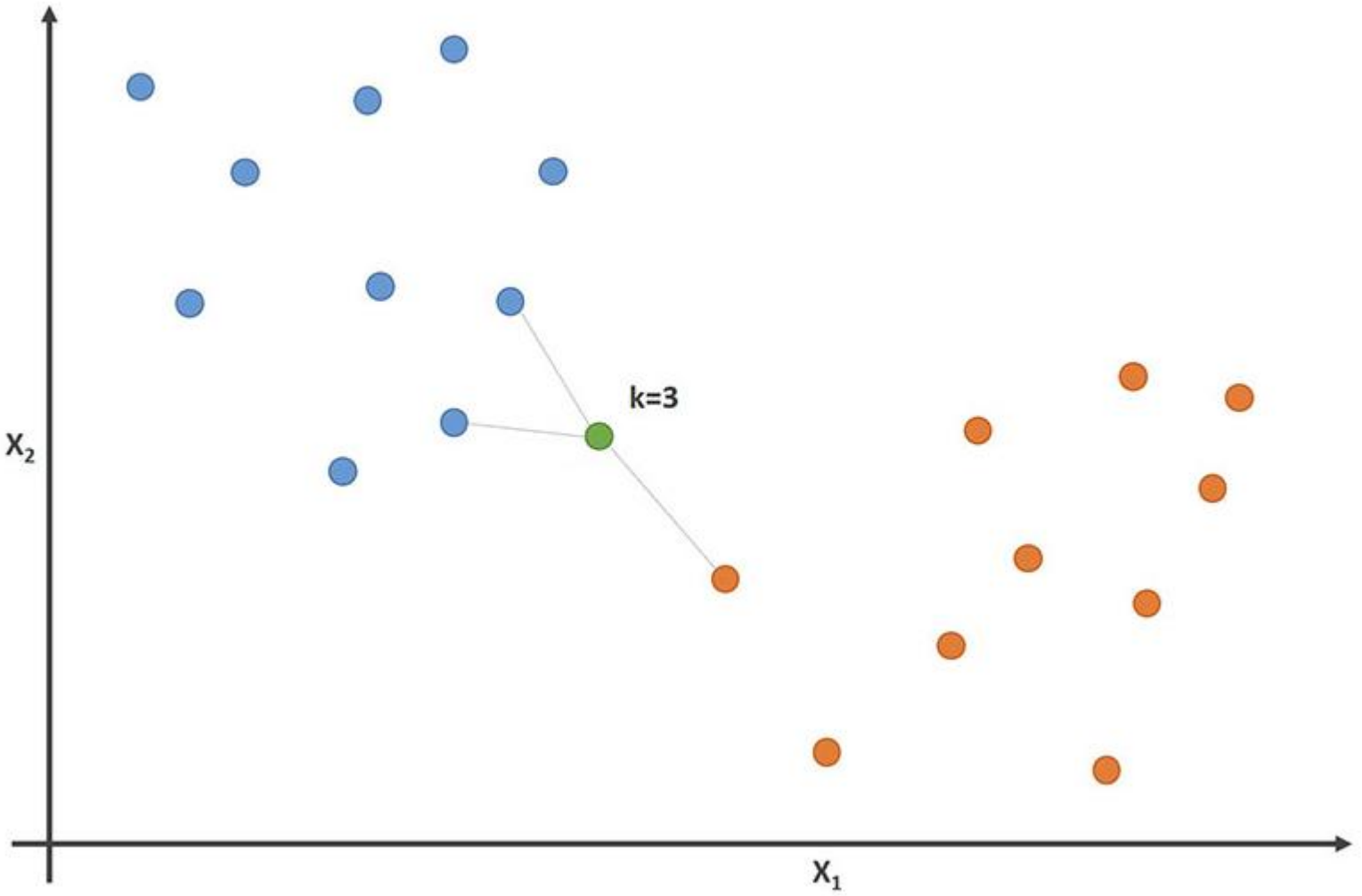
Los métodos de vecinos cercanos son la base de muchos otros métodos de aprendizaje.

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

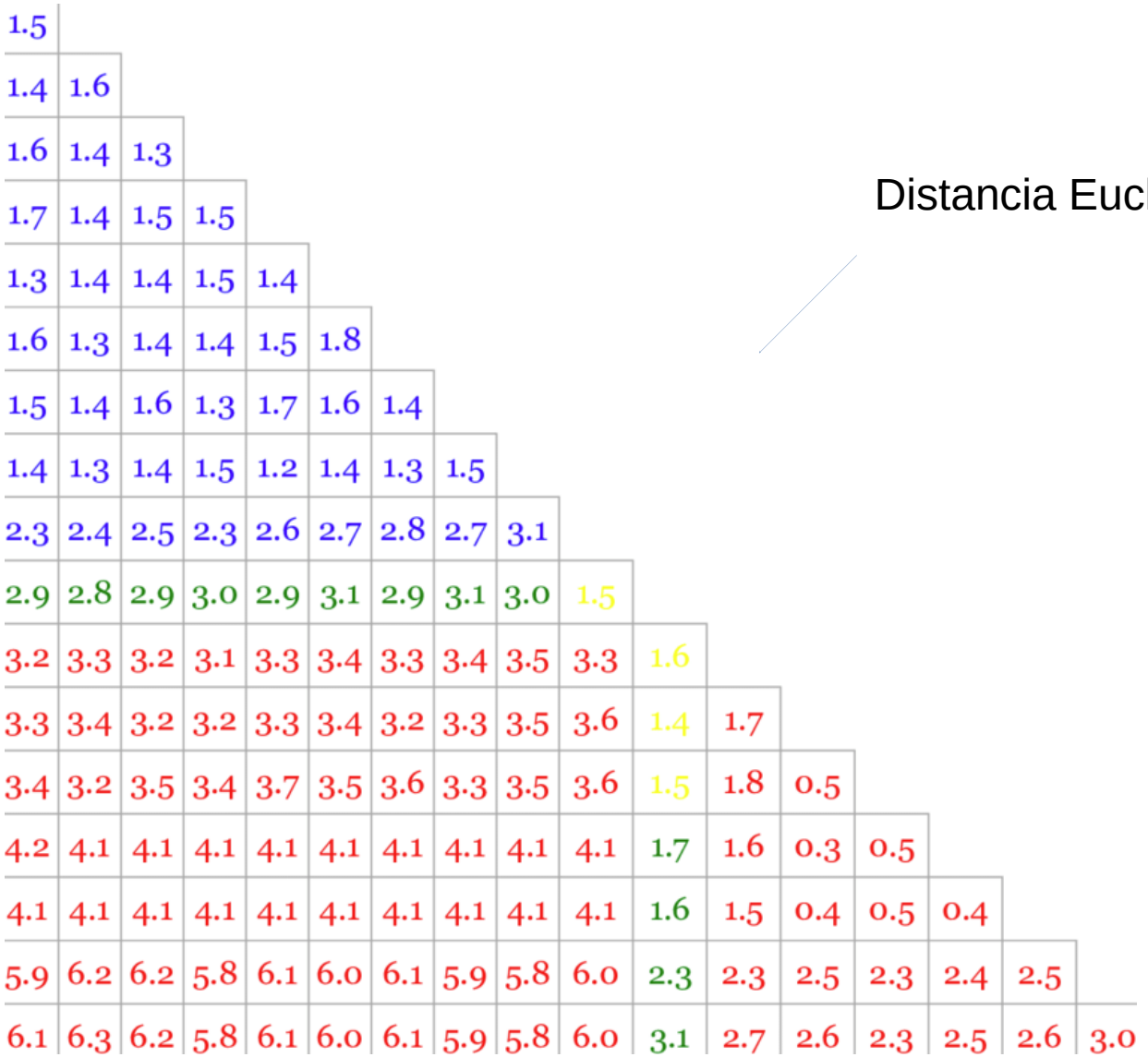
Para disponer de una estructura de vecinos cercana que podamos consultar, se usa alguno de los siguientes tres algoritmos:

- Pairwise metric (fuerza bruta, pocos datos)

- BallTree (alta dimensionalidad)

- kd-trees (baja dimensionalidad)

Vecinos cercanos (pairwise metric)

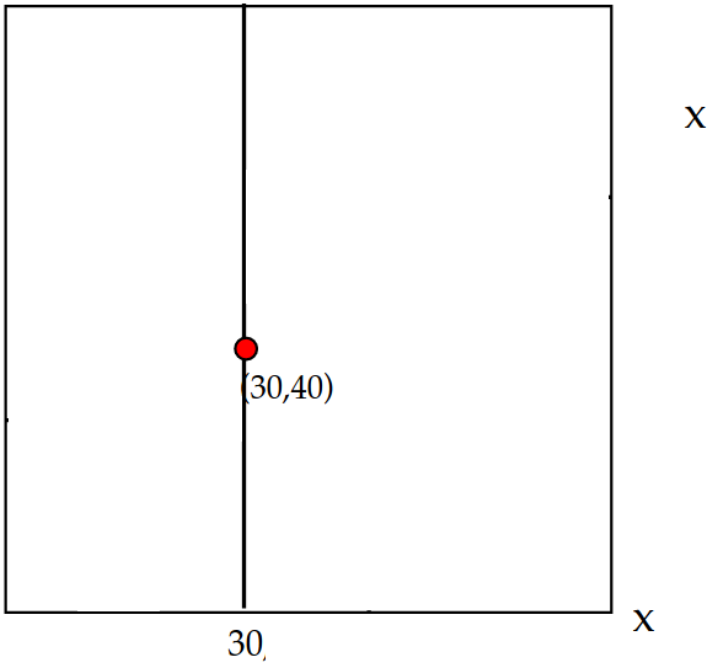


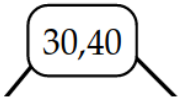
Distancia Euclideana

kd-trees

-Se usan datos para pivotear (dividir) el espacio de representación.

-Si estamos en 2D, el kd-tree alterna la dimensión en cada nivel. Por ejemplo, en el primer nivel, el dato divide el espacio según el eje x.

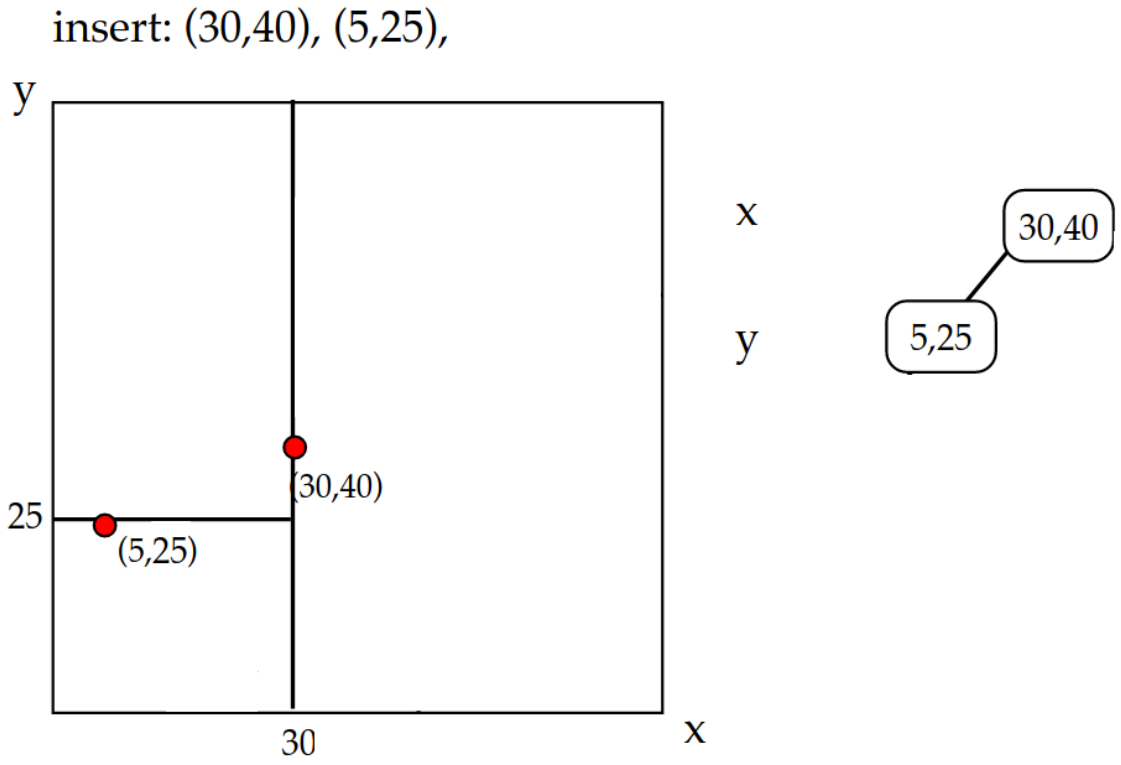




kd-trees

-Se usan datos para pivotear (dividir) el espacio de representación.

-Si estamos en 2D, el kd-tree alterna la dimensión en cada nivel. Por ejemplo, en el primer nivel, el dato divide el espacio según el eje x. En el segundo nivel, pivoteamos en el eje y.

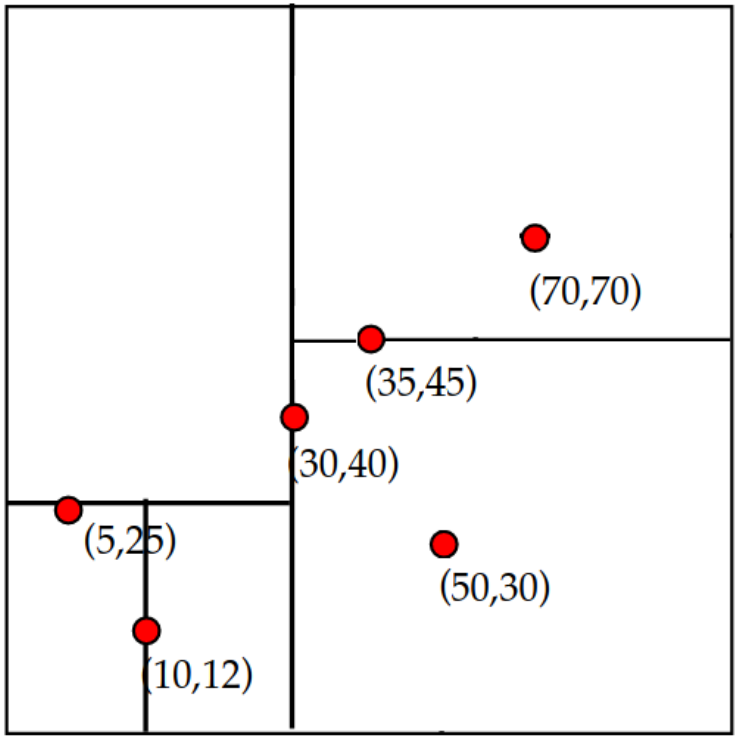


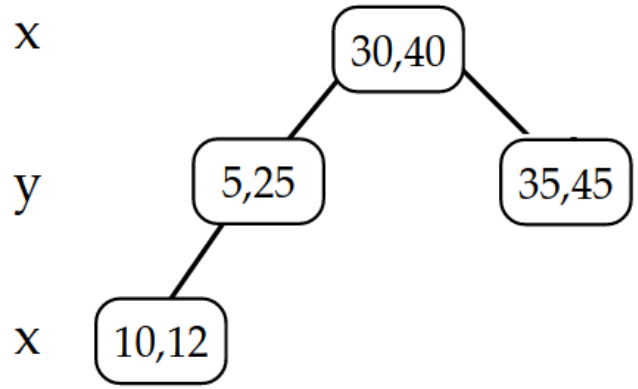
kd-trees

Siempre se divide usando la mediana del sub-espacio:

1) Eje x: Mediana {5, 10, **30**, 35, 50, 70} → (**30**, 40)
2) Eje y: Mediana {12, **25**, 40} →(5, **25**)
3) Eje x: Mediana {5, **10**, 30} → (**10**, 12)
4) Eje y: Mediana {30, 45, 70} →(35, **45**)
5) ...

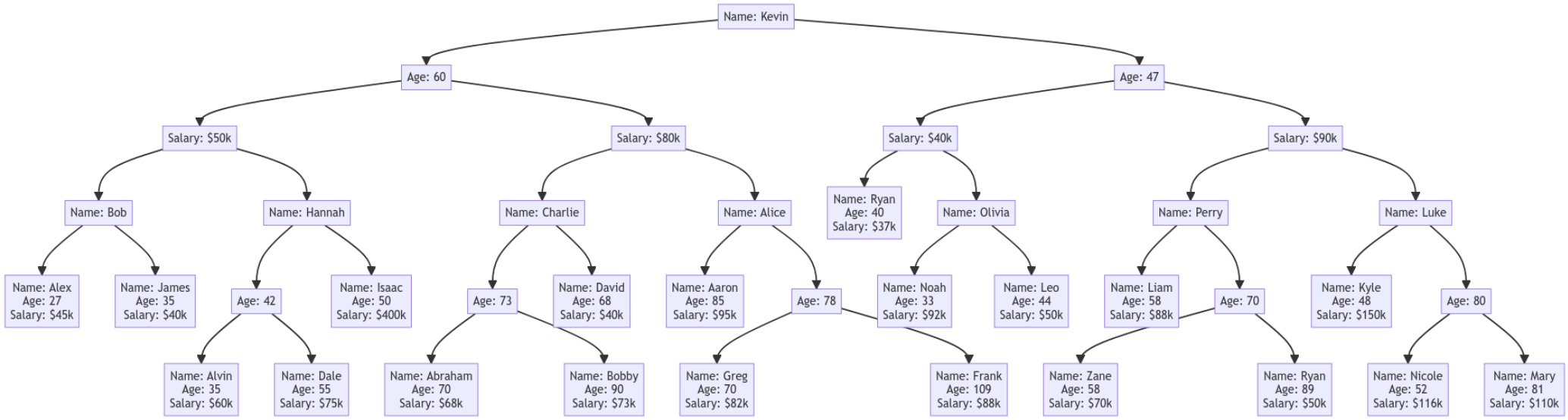
insert: (30,40), (5,25), (10,12),





kd-trees

Los kd-trees son una extensión **k d**imensional de los árboles de búsqueda binaria para datos k dimensionales.



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

Ball-trees

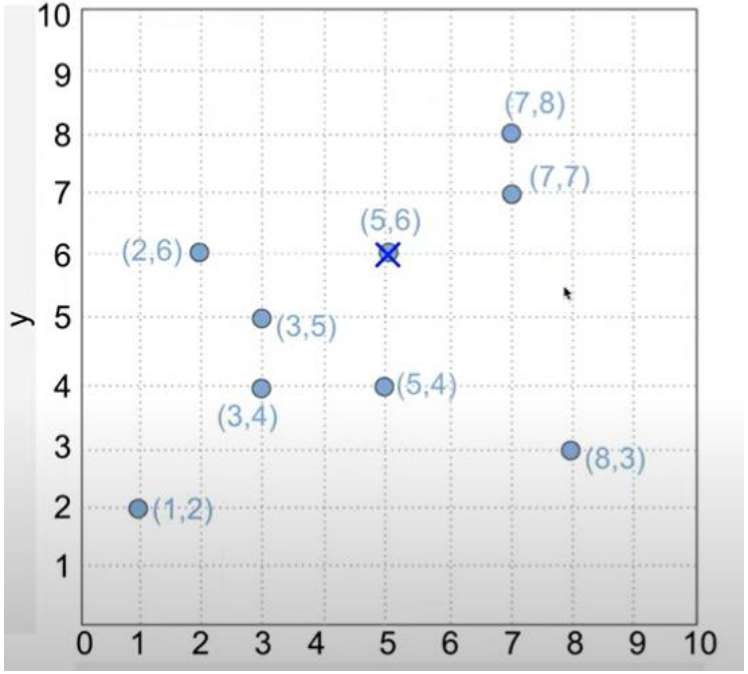
Cambiamos por un método basado en radios.

- 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:

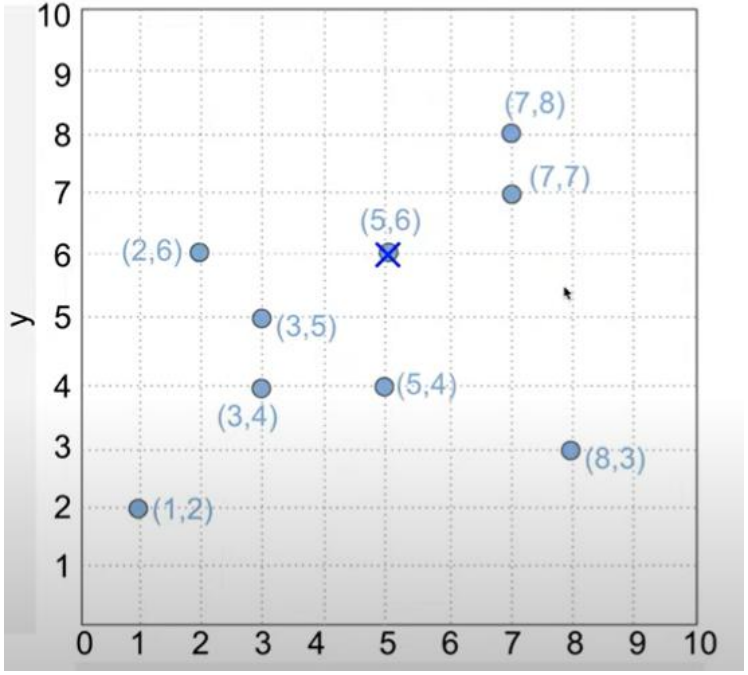
> Dado un punto externo *t* a una esfera B, su distancia a cualquier punto de B será mayor o igual que la distancia a la superficie de B.

Ball-trees (construcción)

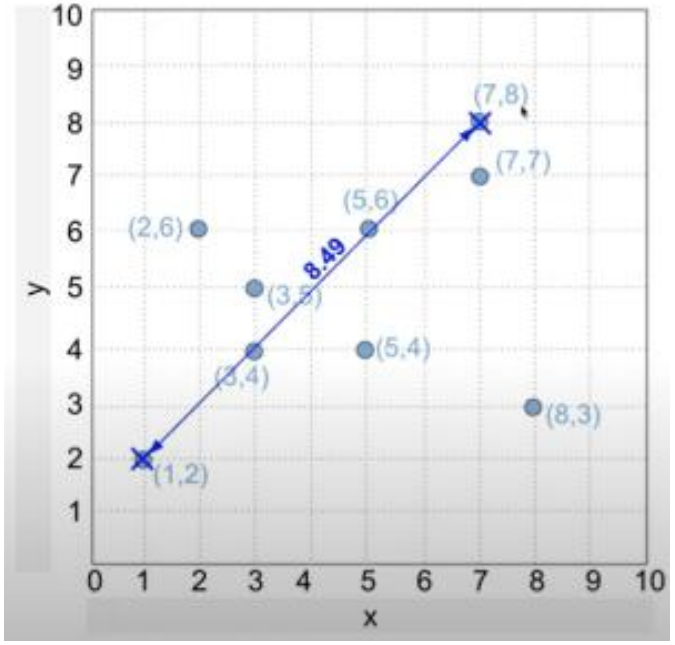


(a) Seleccionamos una semilla

Ball-trees (construcción)

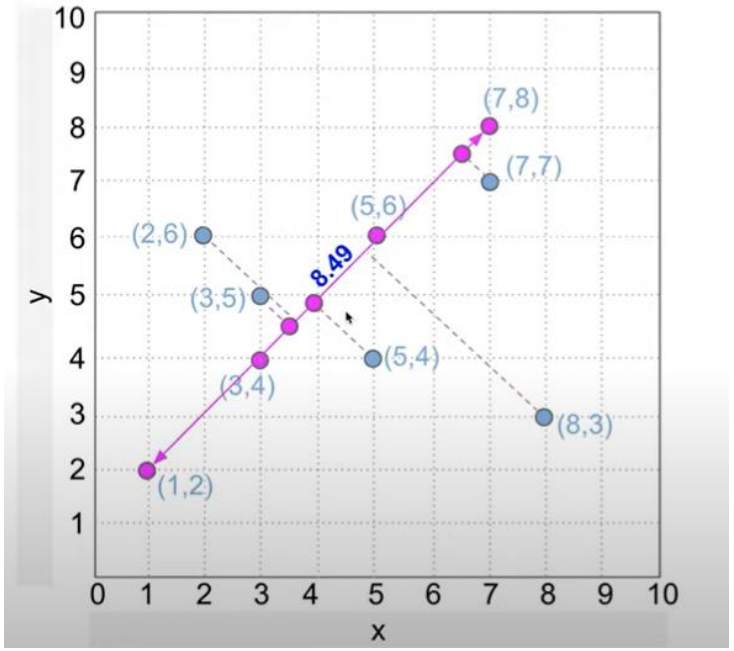


(a) Seleccionamos una semilla



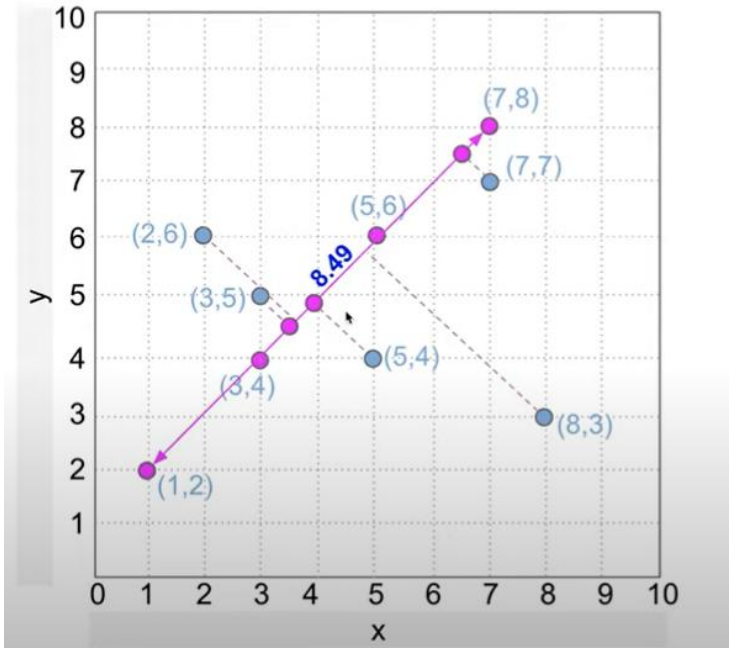
(b) Buscamos el par de puntos más lejanos a la semilla

Ball-trees (construcción)

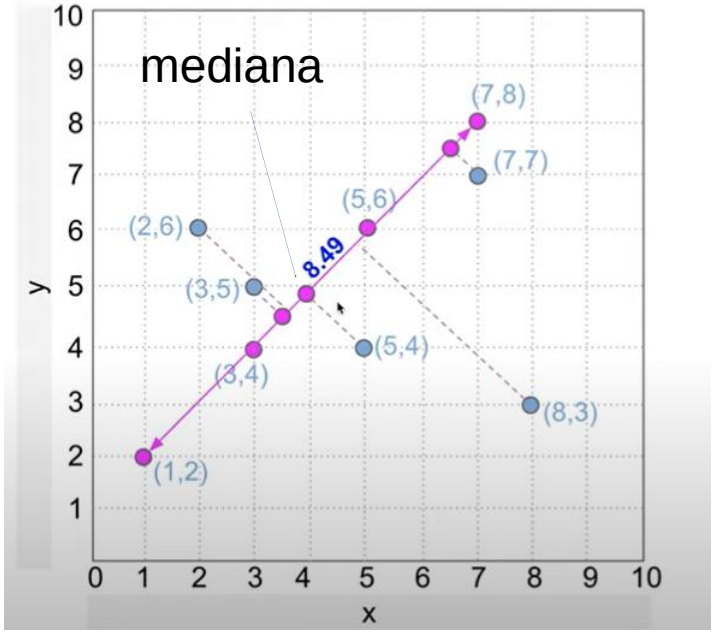


(c) Proyectamos los puntos a la recta que une los puntos más lejanos

Ball-trees (construcción)

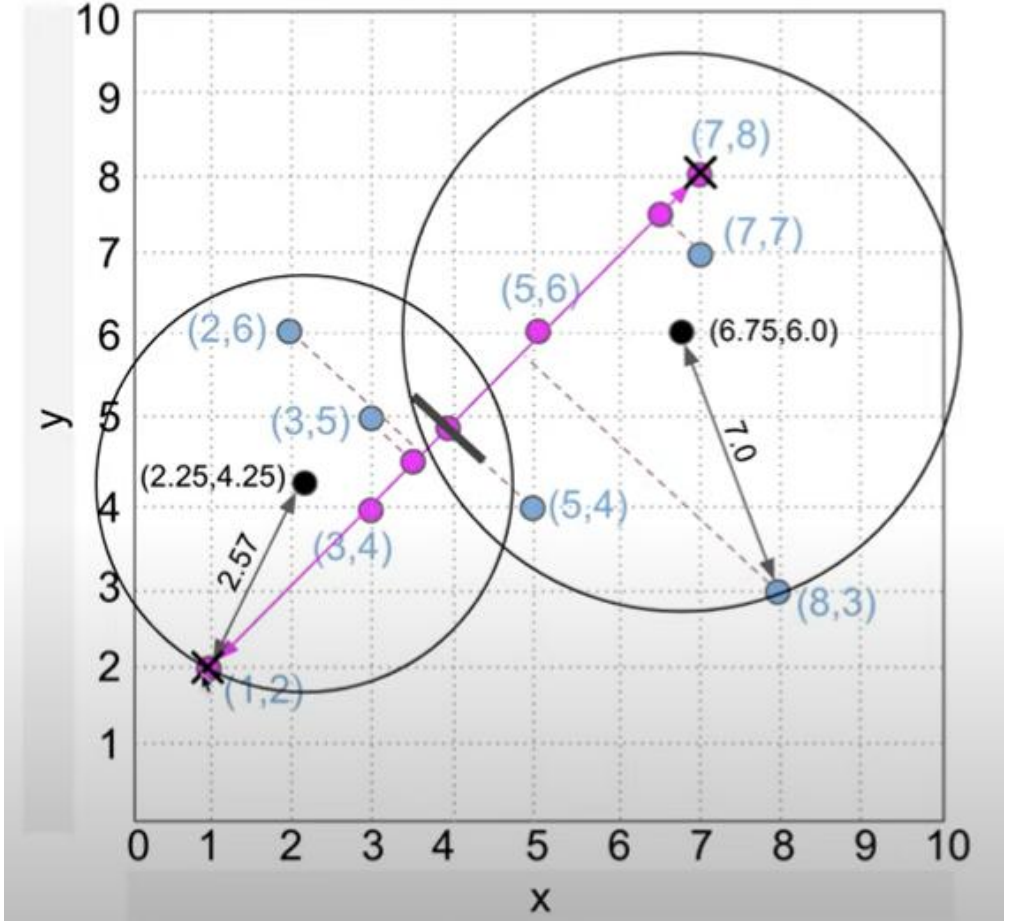


(c) Proyectamos los puntos a la recta que une los puntos más lejanos



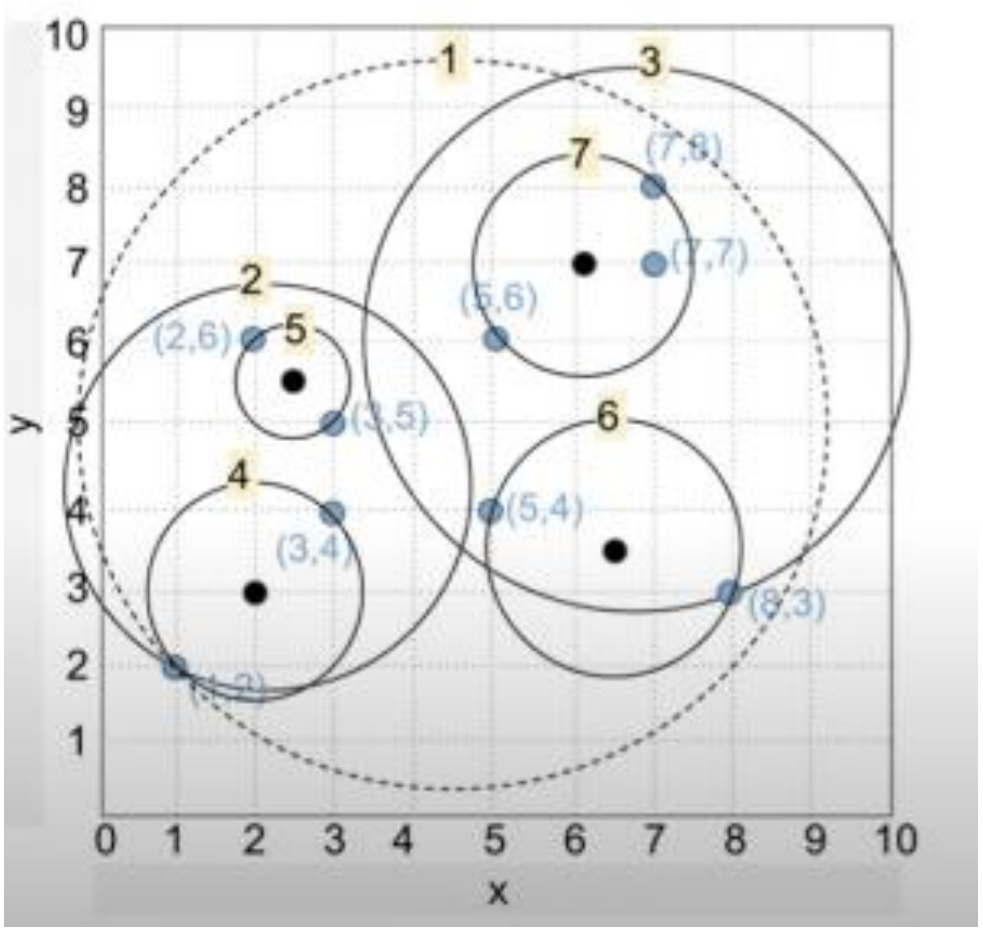
(d) Dividimos el espacio en torno de la mediana

Ball-trees (construcción)



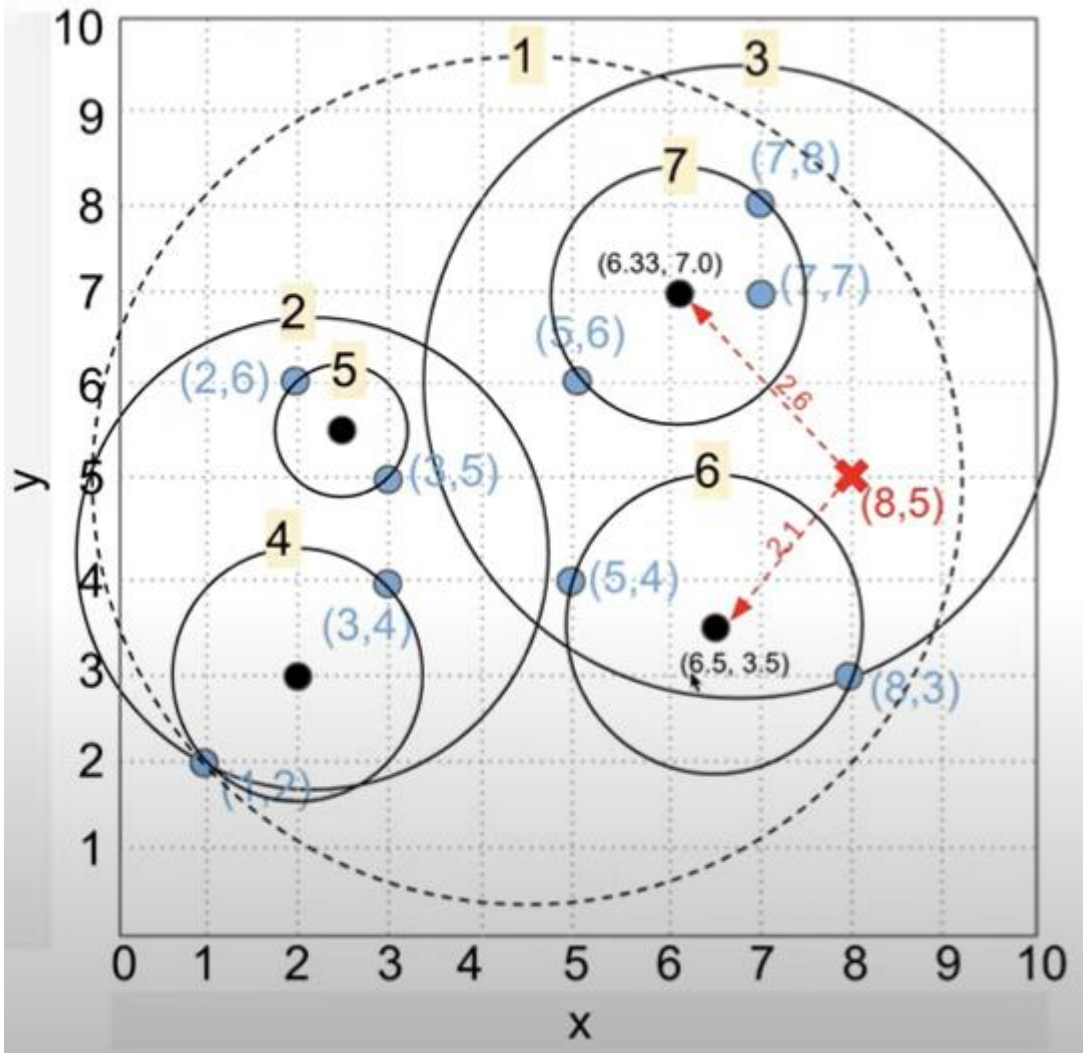
(e) En cada sub-espacio, calculamos el centroide. Desde el centroide buscamos el punto más lejano del sub-espacio. Calculamos el radio desde el centroide al punto lejano y se forma la circunferencia.

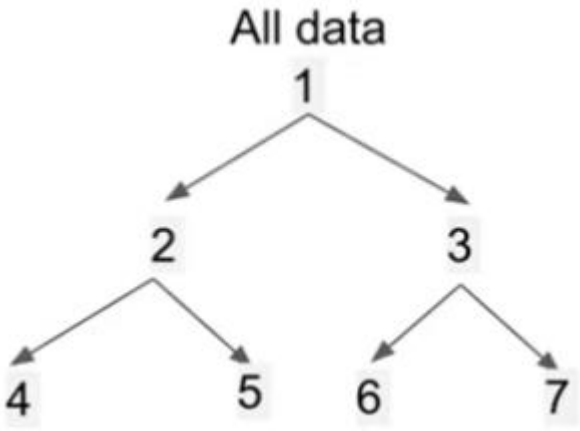
Ball-trees (construcción)



(f) Repetimos el procedimiento

Búsqueda en Ball-trees



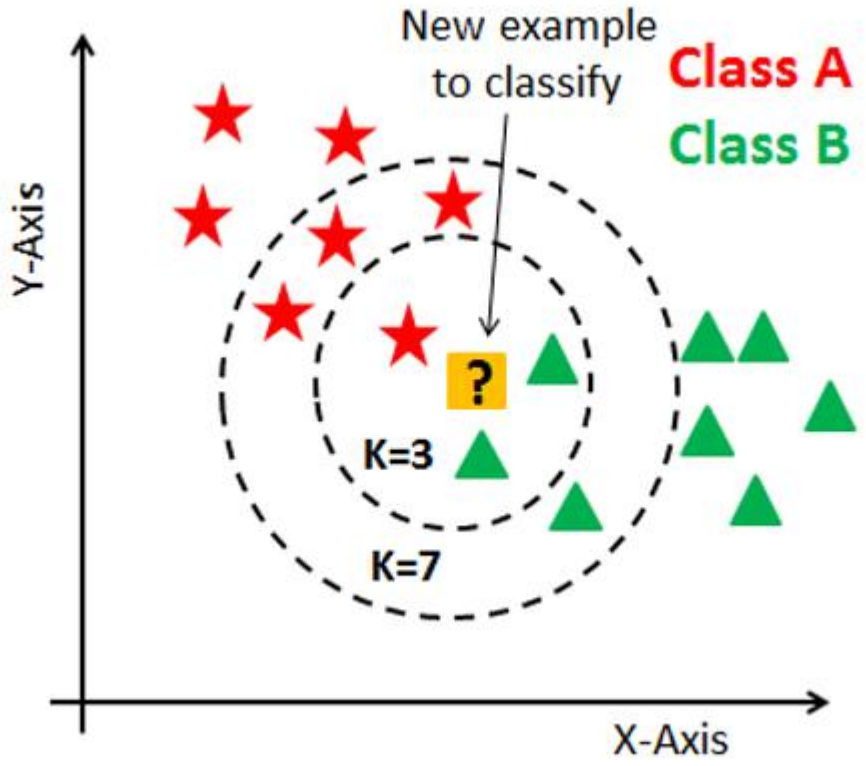


Se busca la esfera de centroide más cercano.

- 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.

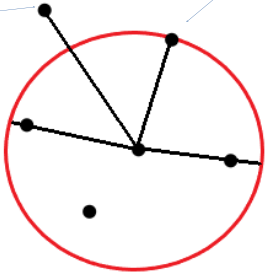
- Se compara su densidad con la densidad de los vecinos.

Definimos *k*-distance(B): distancia de B a su *k*-ésimo vecino más cercano.

Luego definimos alcanzabilidad:

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

Parametriza la distancia según el vecindario de B



Si A está afuera del vecindario de B, nos quedamos con ésta

Si A está dentro del vecindario, se reemplaza por la distancia al k vecino de B.

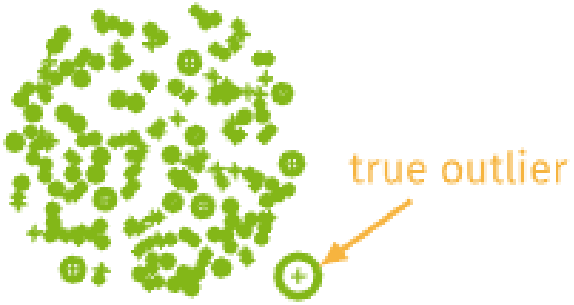
Vecinos cercanos para detección de outliers

Definimos la densidad de alcanzabilidad local:

La sumatoria aumenta su valor si el área el dato es un outlier

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

… pero como es el inverso, LRD disminuye si el dato es un outlier.



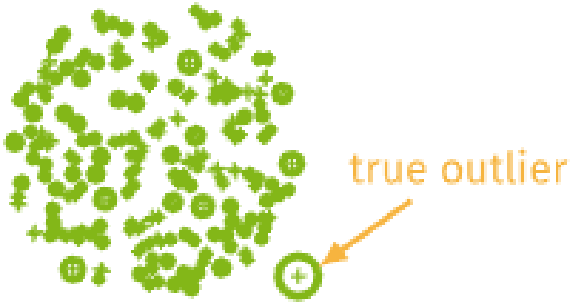
Vecinos cercanos para detección de outliers

LRD es pequeño si el dato es un outlier.

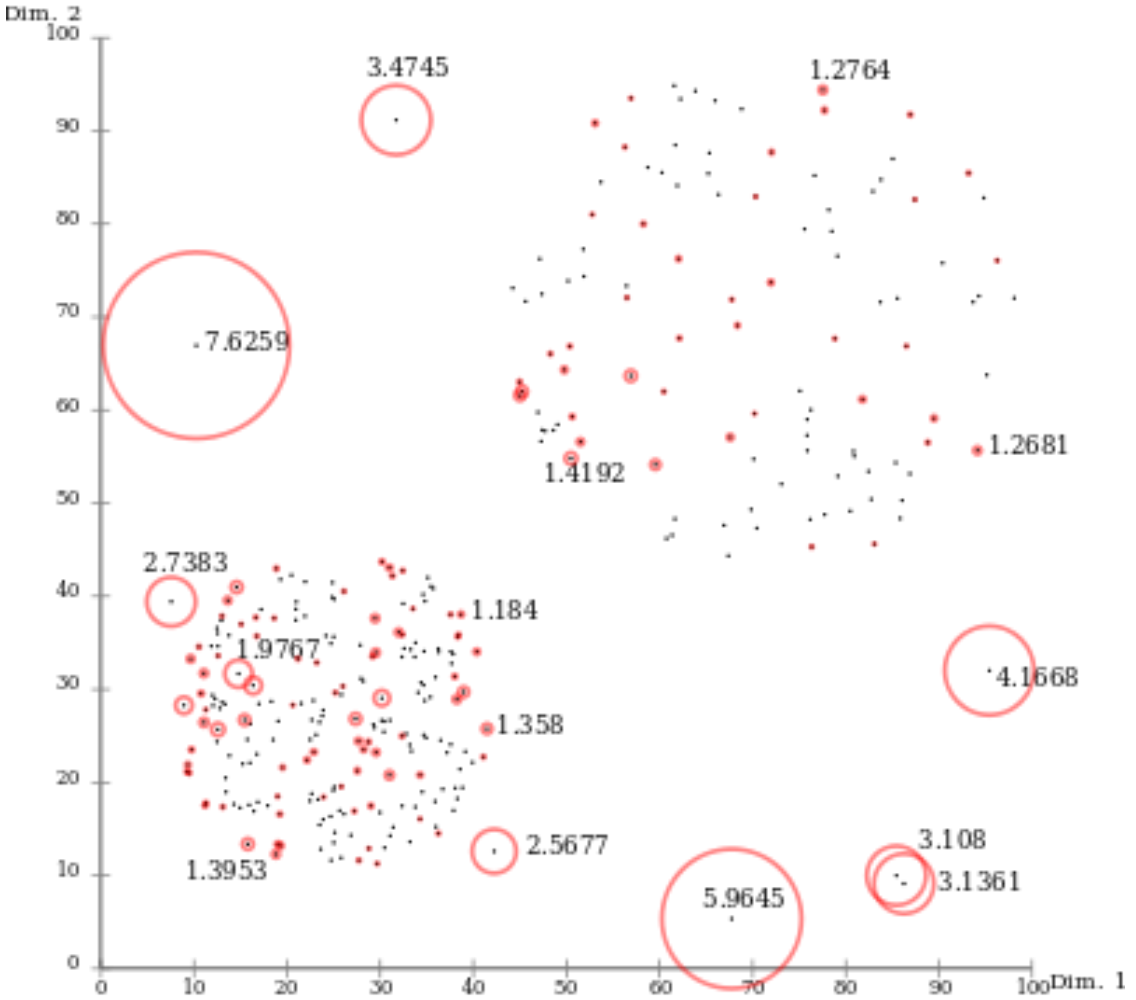
Local Outlier Factor (LOF):

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

Si p es un outlier, su LRD va a ser menor que el de sus vecinos (q), por lo que LOF(p) va a ser grande.



Vecinos cercanos para detección de outliers



Obs.: En la librería que usaremos, aparece un signo -. Lo importante es el valor absoluto de LOF.