IIC 2433 Minería de Datos

https://github.com/marcelomendoza/IIC2433

- VECINOS CERCANOS -

Vecinos cercanos

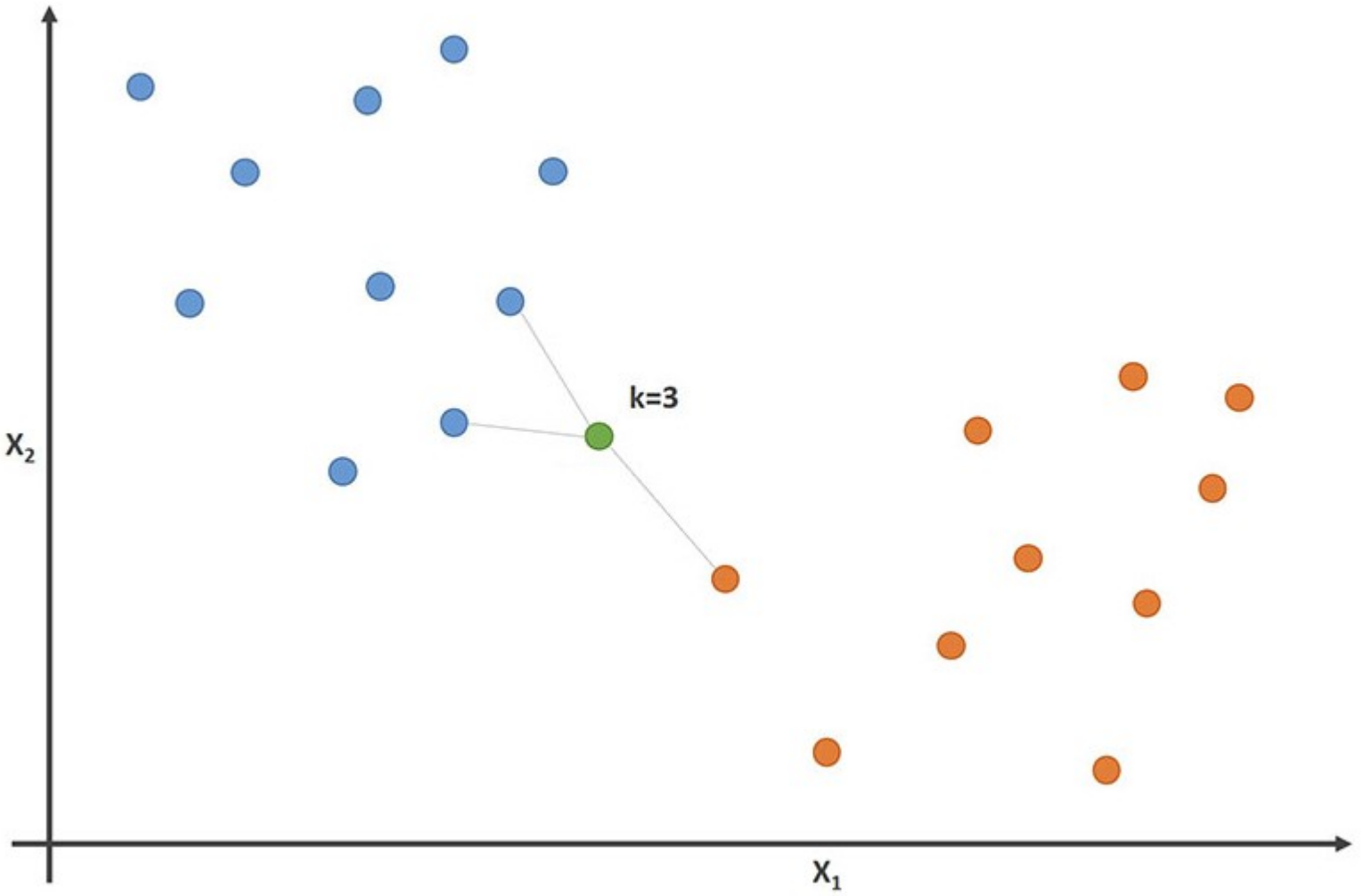
Los métodos de vecinos cercanos son la base de muchos otros métodos de aprendizaje.

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)

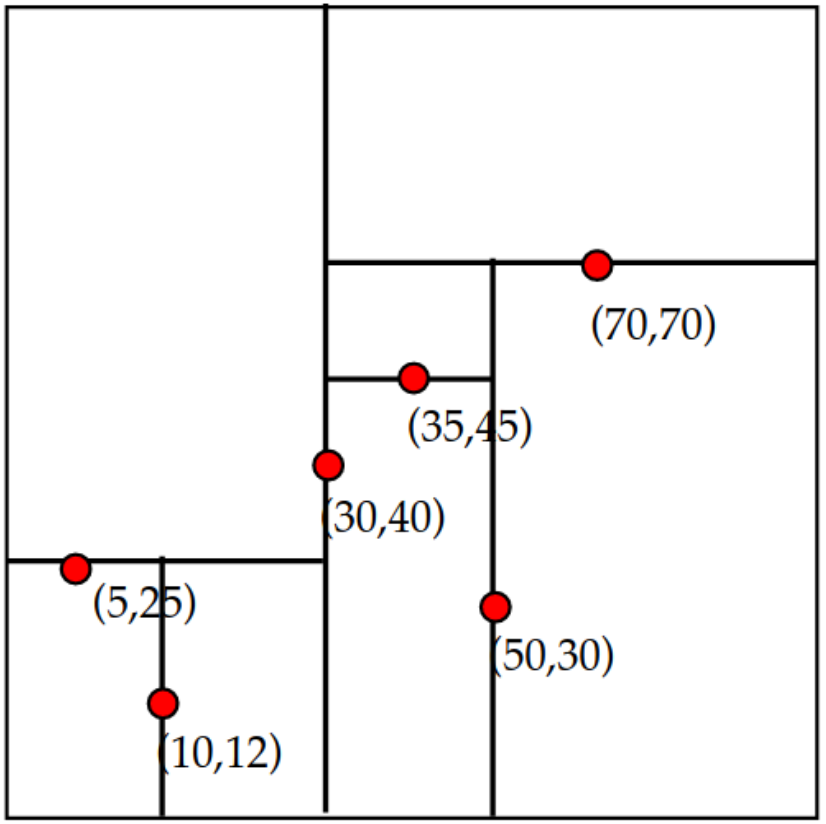
Vecinos cercanos (pairwise metric)

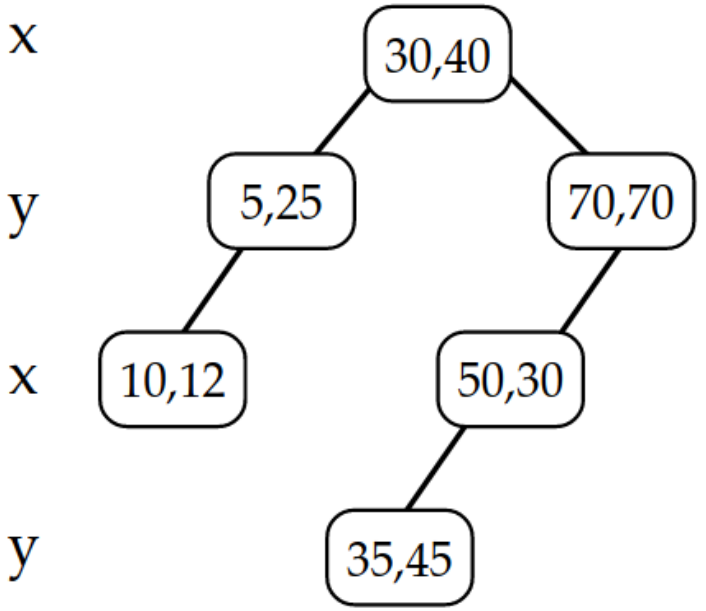
1.5															
1.4	1.6														
1.6	1.4	1.3													
1.7	1.4	1.5	1.5												
1.3	1.4	1.4	1.5	1.4											
1.6	1.3	1.4	1.4	1.5	1.8										
1.5	1.4	1.6	1.3	1.7	1.6	1.4									
1.4	1.3	1.4	1.5	1.2	1.4	1.3	1.5								
2.3	2.4	2.5	2.3	2.6	2.7	2.8	2.7	3.1							
2.9	2.8	2.9	3.0	2.9	3.1	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)





Ball-trees

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

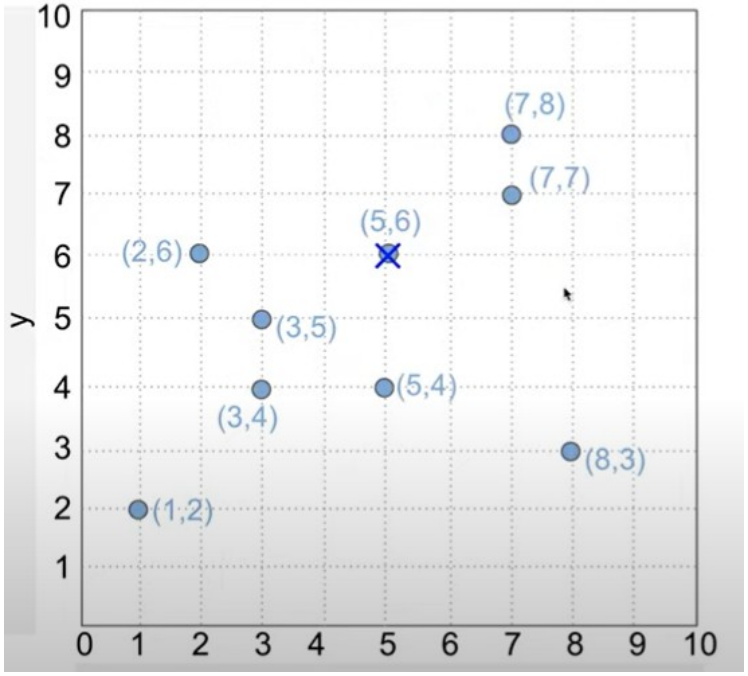
Cambiamos por un método basado en radios.

- 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:

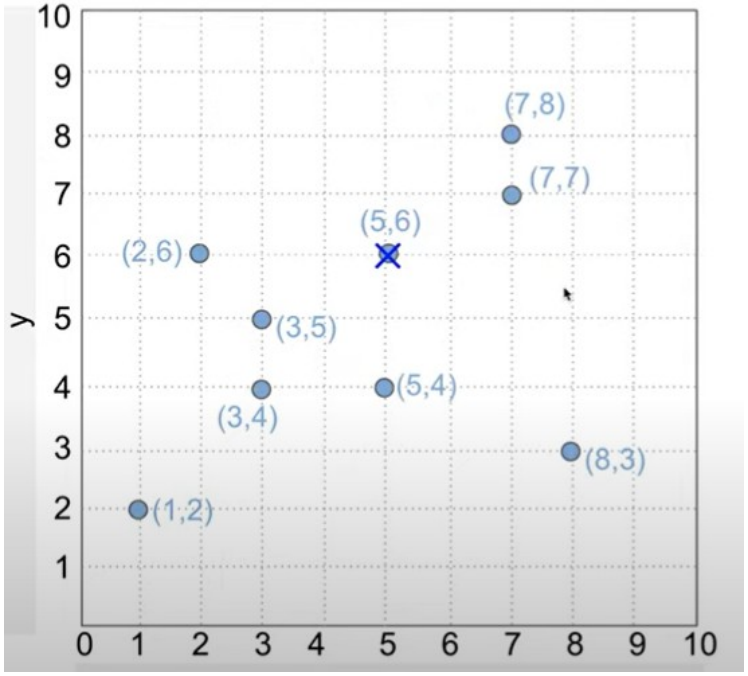
> Dado un punto externo *t* a una esfera B, su distancia a cualquier punto de B será mayor o igual que la distancia a la superficie de B.

Ball-trees (construcción)

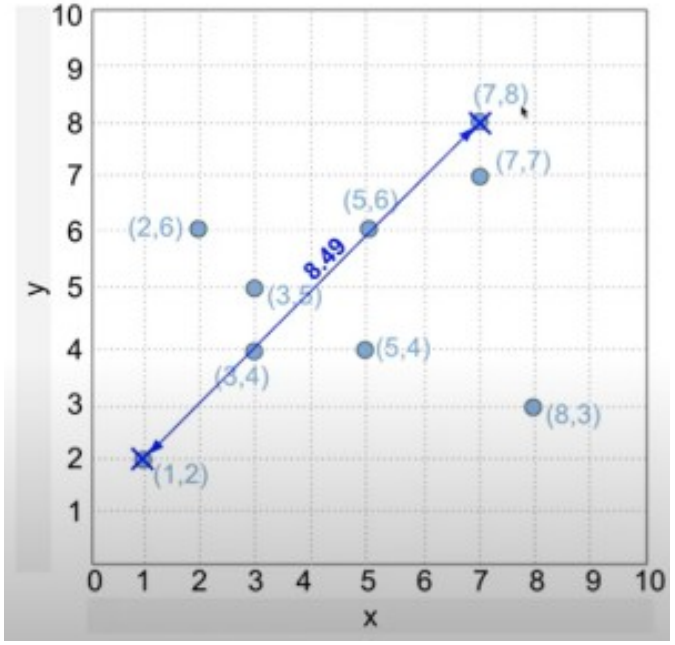


(a) Seleccionamos una semilla

Ball-trees (construcción)

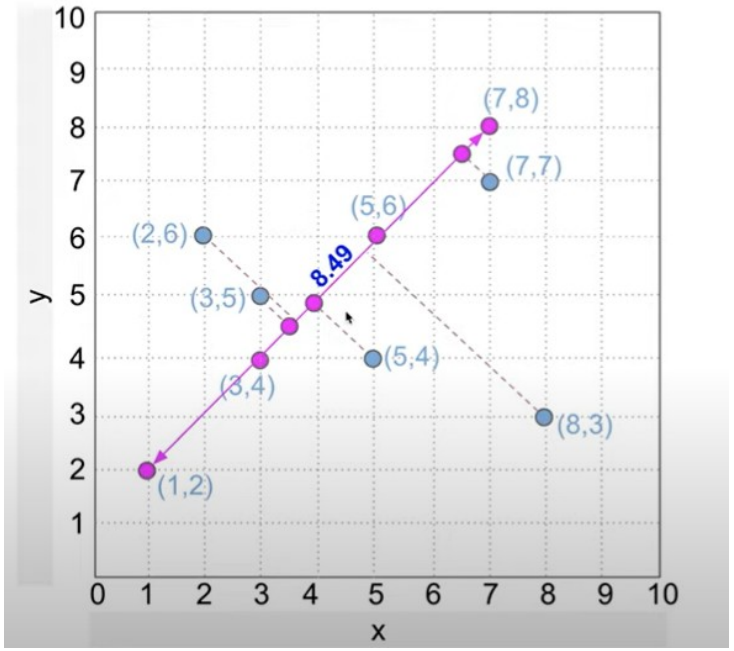


(a) Seleccionamos una semilla



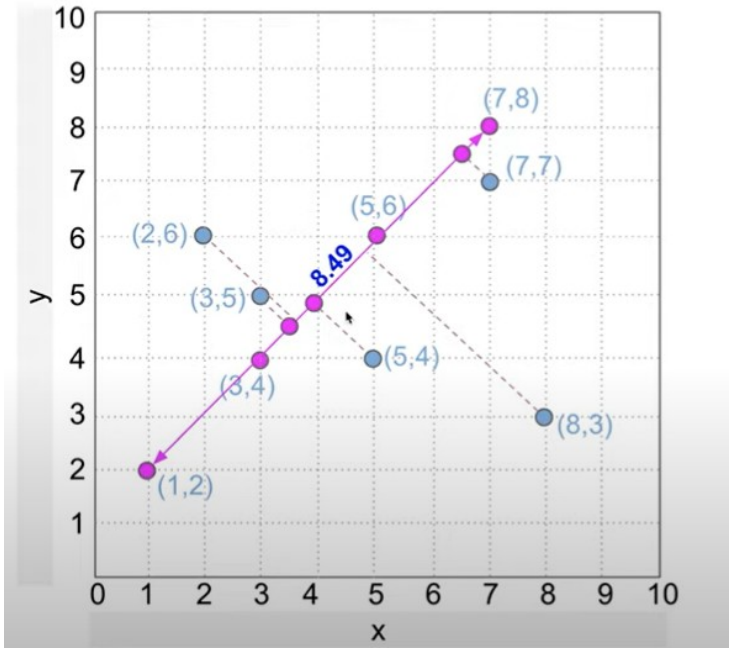
(b) Buscamos el par de puntos más lejanos a la semilla

Ball-trees (construcción)

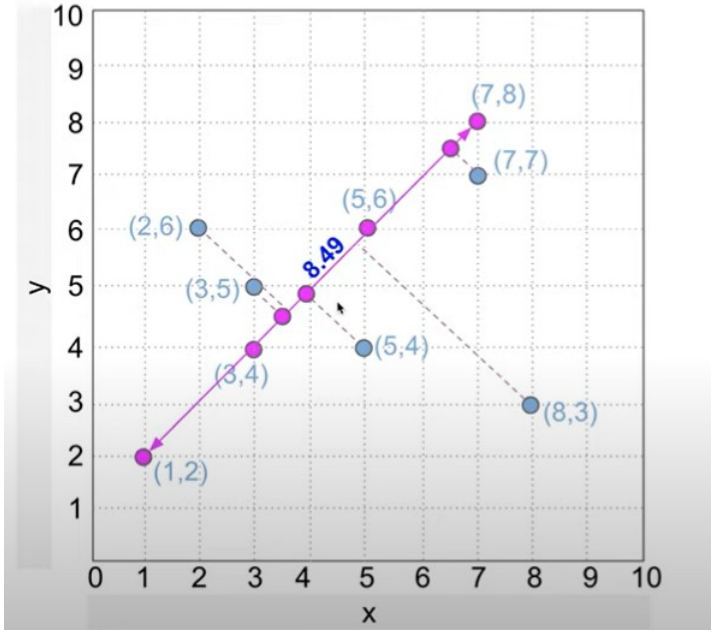


(c) Proyectamos los puntos a la recta que une los puntos más lejanos

Ball-trees (construcción)

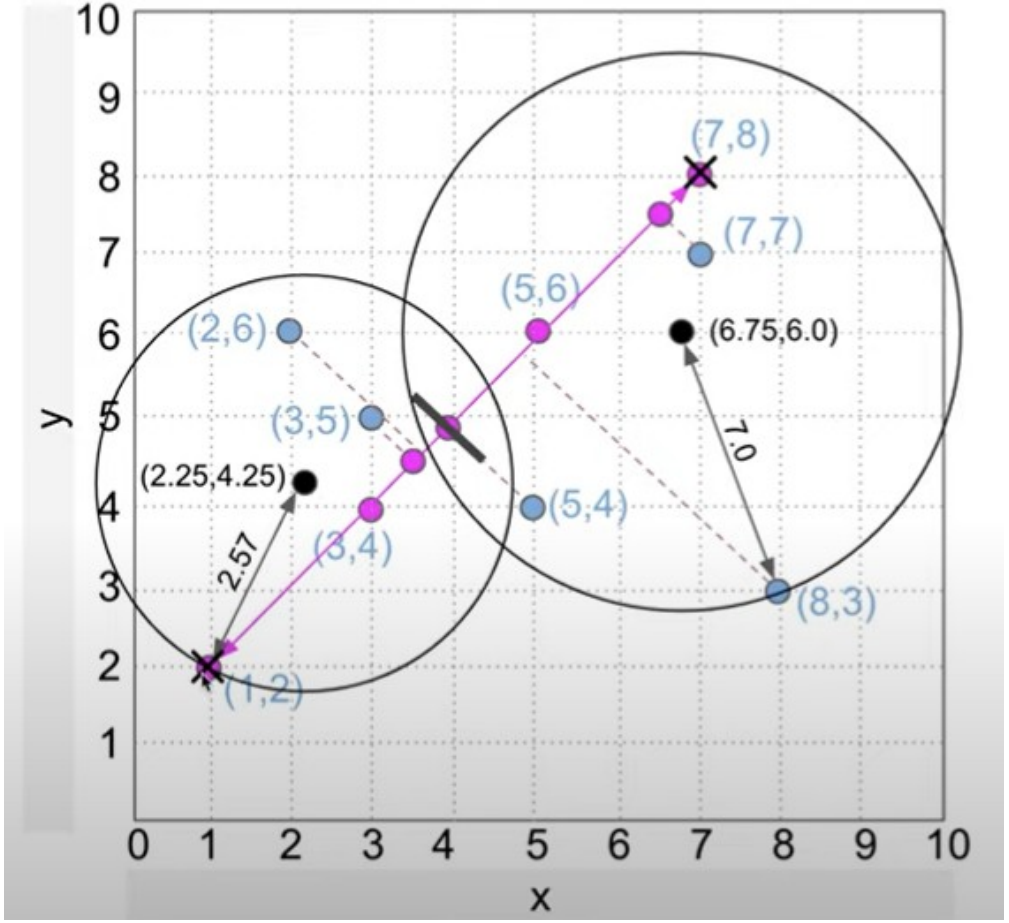


(c) Proyectamos los puntos a la recta que une los puntos más lejanos



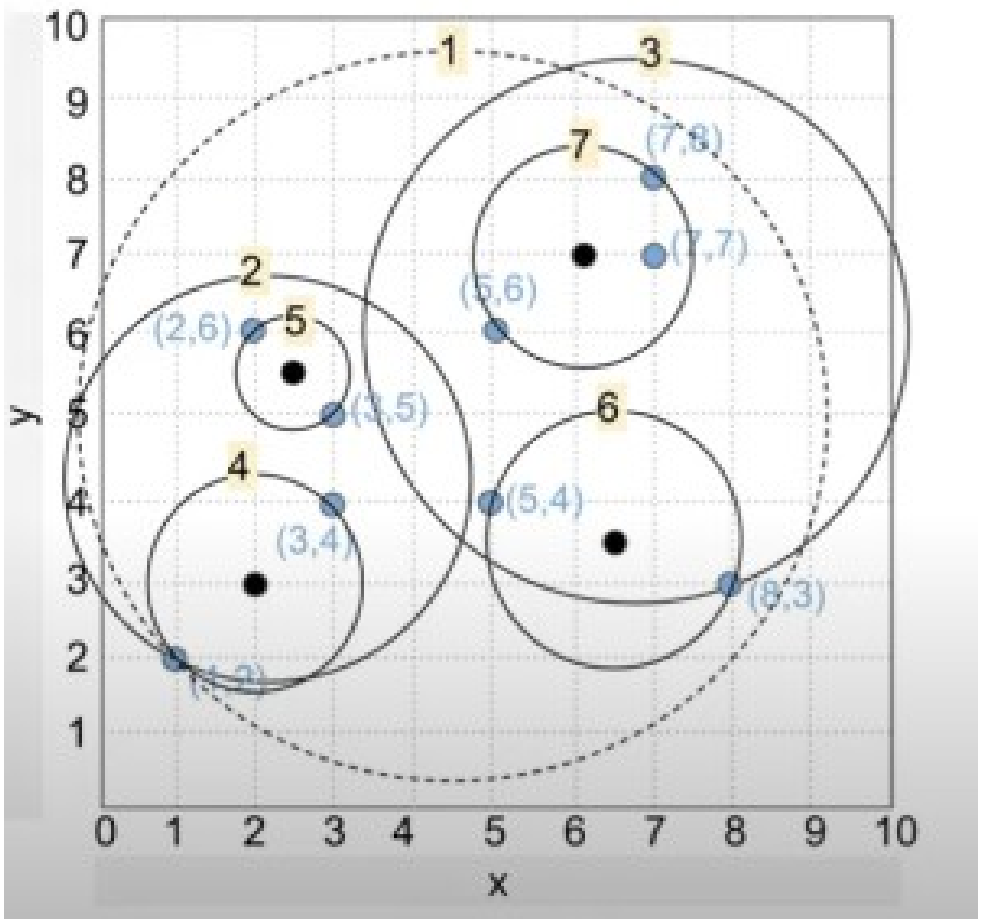
(d) Dividimos el espacio en torno de la mediana

Ball-trees (construcción)



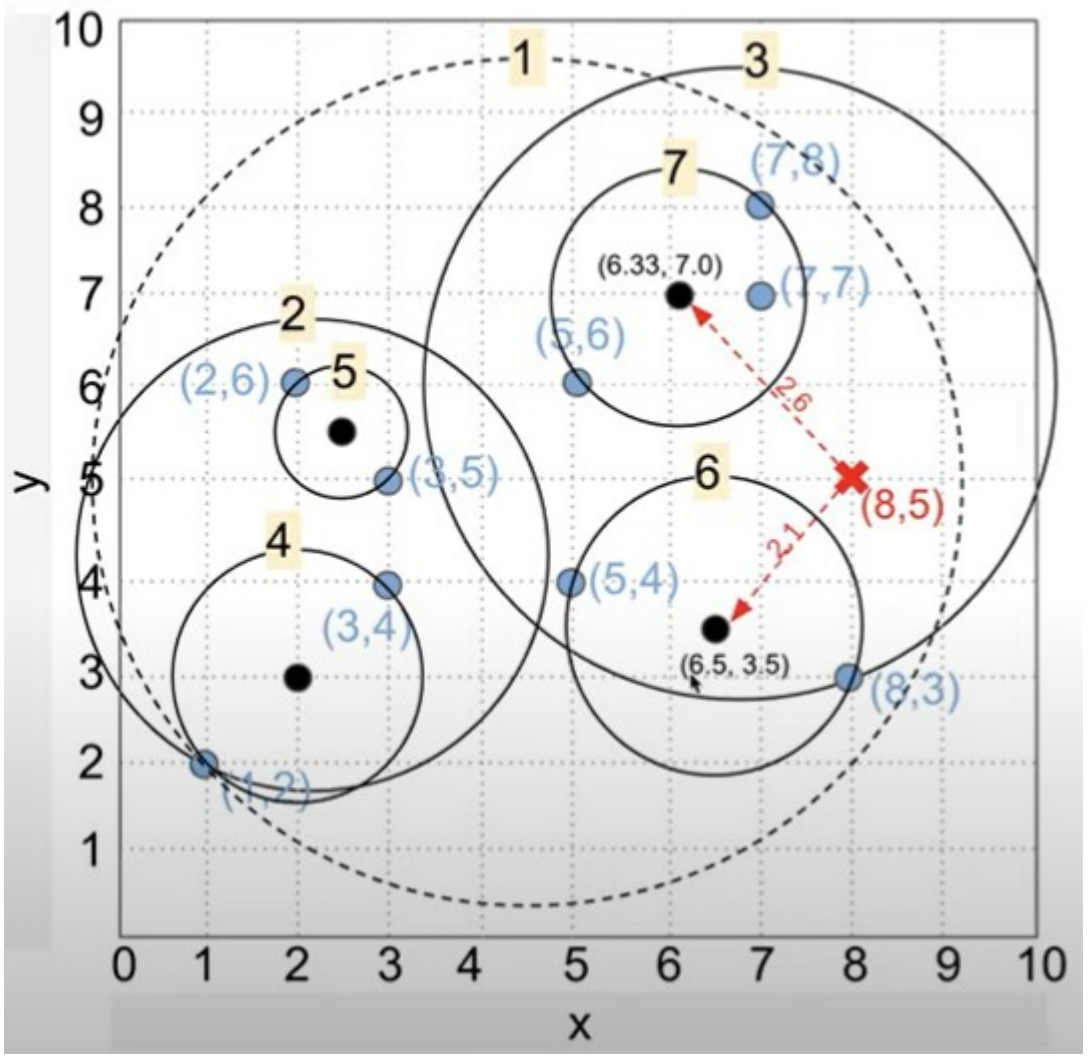
(e) En cada mitad, calculamos el centroide. Desde el centroide buscamos el punto más lejano de la mitad.

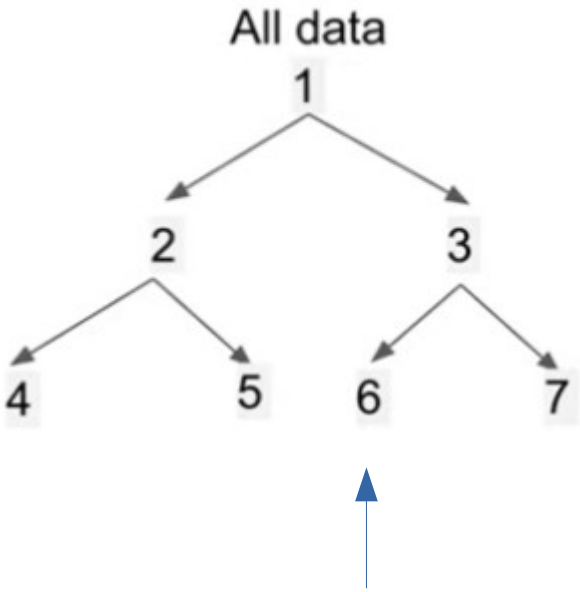
Ball-trees (construcción)



(f) Repetimos el procedimiento

Búsqueda en Ball-trees



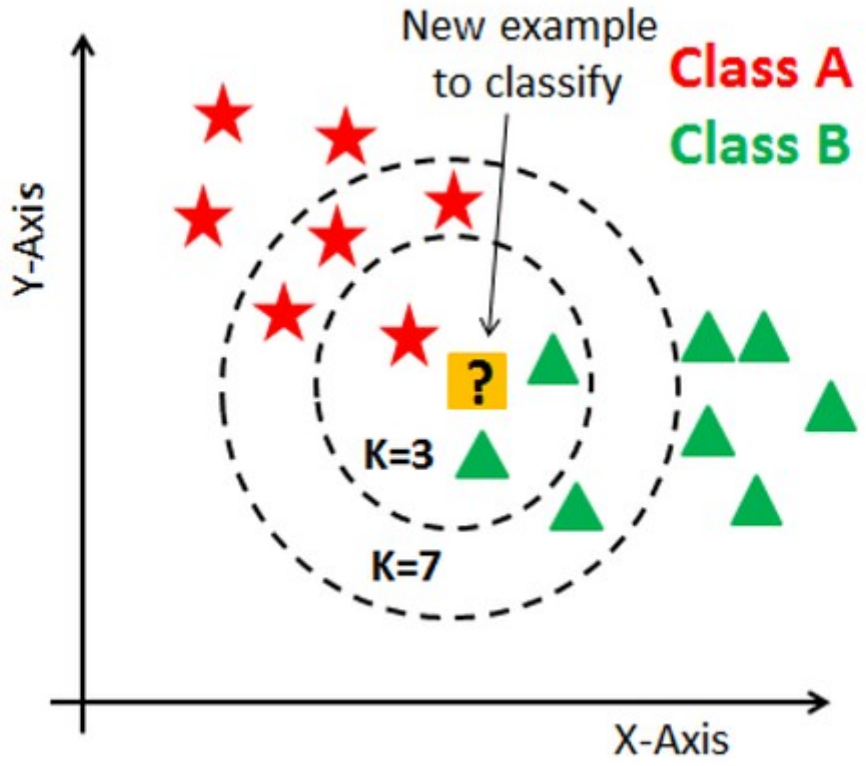


Se busca la esfera de centroide más cercano.

- 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.

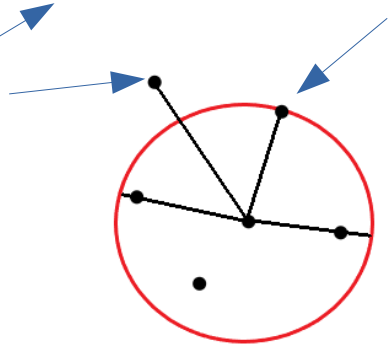
- Se compara su densidad con la densidad de los vecinos.

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

Luego definimos alcanzabilidad:

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

Parametriza la distancia según el tamaño del 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 se minimiza si el área es densa

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

… pero como es el inverso, LRD aumenta si el área es densa.

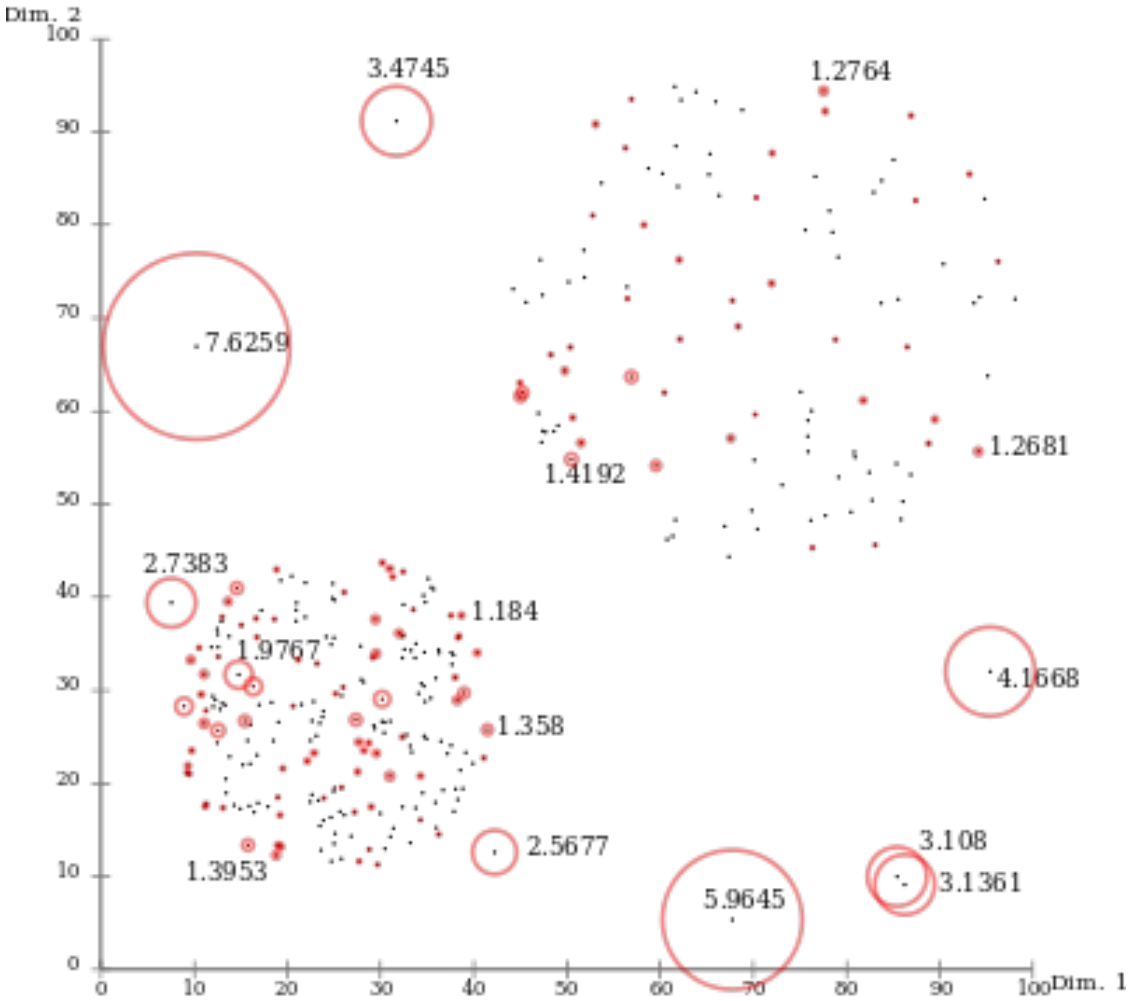
Y luego el Local Outlier Factor (LOF):

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

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

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

Vecinos cercanos para detección de outliers



- KMEANS -

Clustering permite entender como se agrupan los datos

Clustering con k-means

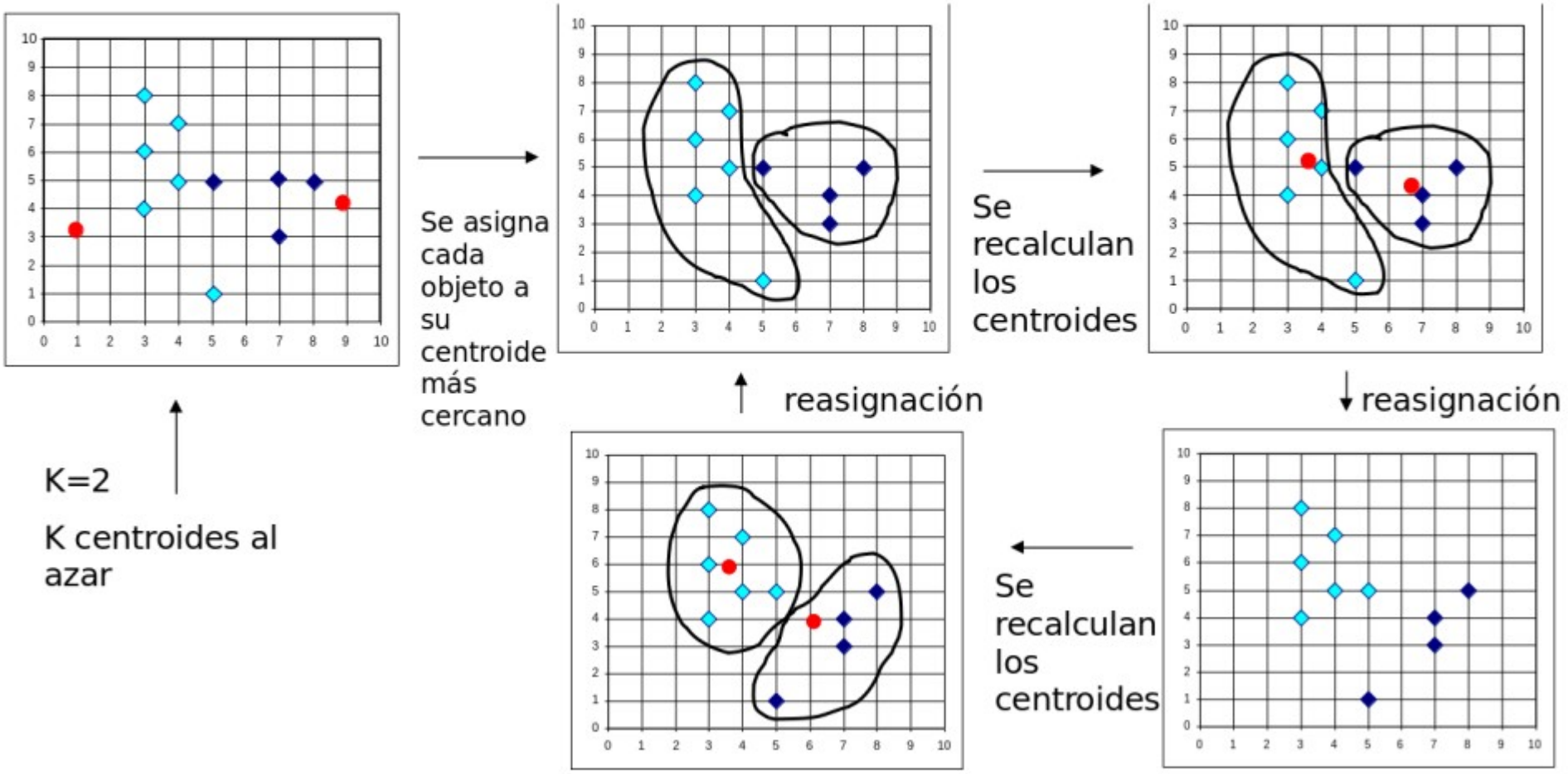
- Cada cluster en K-means es definido por un centroide.
- Objetivo: optimizar alguna noción de distancia:
  1. Intra-cluster: (Minimizar) distancia entre objetos de un cluster a su centroide.
  2. Inter-cluster: (Maximizar) distancia entre objetos de clusters distintos.
- Centroide:

$$c_i = \frac{1}{m_i} \sum_{x \in C_i} x$$

  donde C_i denota un cluster.
- Idea del algoritmo:
  - Asignación inicial: `k` centroides al azar.
  - Reasignación: asignar cada objeto a su centroide más cercano (algoritmo avaro).
  - Recomputación: recalcular los centroides. □

Clustering con k-means

Ejemplo



Clustering con k-means

Hechos importantes:

- K-means converge. (McQueen, 67)
- Criterios de parada
  1. Iteraciones: (Máximo) número de iteraciones.
  2. Error tolerado: (Optimizar) alguna noción de distancia entre objetos.
- Complejidad:
  1. K-means es $\texttt{NP} - \texttt{hard}$ en cualquier espacio d-dimensional con distancia Euclideana o coseno.
  2. K-means es $\texttt{NP} - \texttt{hard}$ para cualquier valor de k.

SSE: Suma de errores al ²

Clustering con k-means

k-means minimiza el SSE: implícitamente

$$\mathrm{SSE} = \sum_{i=1}^{K} \sum_{x \in C_i} (c_i - x)^2$$

SSE: Suma de errores al ²

Clustering con k-means

k-means minimiza el SSE: implícitamente

$$\mathrm{SSE} = \sum_{i=1}^{K} \sum_{x \in C_i} (c_i - x)^2$$

$$\begin{aligned} \frac{\partial}{\partial c_k}\mathrm{SSE} &= \frac{\partial}{\partial c_k} \sum_{i=1}^{K} \sum_{x \in C_i} (c_i - x)^2 \\ &= \sum_{i=1}^{K} \sum_{x \in C_i} \frac{\partial}{\partial c_k} (c_i - x)^2 \\ &= \sum_{x \in C_k} 2 * (c_k - x_k) = 0 \end{aligned}$$

SSE: Suma de errores al ²

Clustering con k-means

k-means minimiza el SSE: implícitamente

$$\mathrm{SSE} = \sum_{i=1}^{K} \sum_{x \in C_i} (c_i - x)^2$$

$$\begin{aligned} \frac{\partial}{\partial c_k}\mathrm{SSE} &= \frac{\partial}{\partial c_k} \sum_{i=1}^{K} \sum_{x \in C_i} (c_i - x)^2 \\ &= \sum_{i=1}^{K} \sum_{x \in C_i} \frac{\partial}{\partial c_k} (c_i - x)^2 \\ &= \sum_{x \in C_k} 2 * (c_k - x_k) = 0 \end{aligned}$$

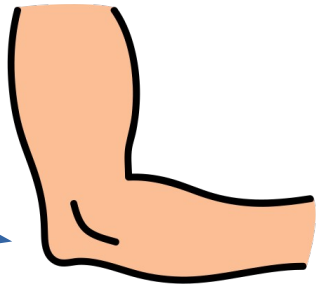
$$\sum_{x \in C_k} 2 * (c_k - x_k) = 0 \Rightarrow m_k c_k = \sum_{x \in C_k} x_k \Rightarrow c_k = \frac{1}{m_k} \sum_{x \in C_k} x_k$$

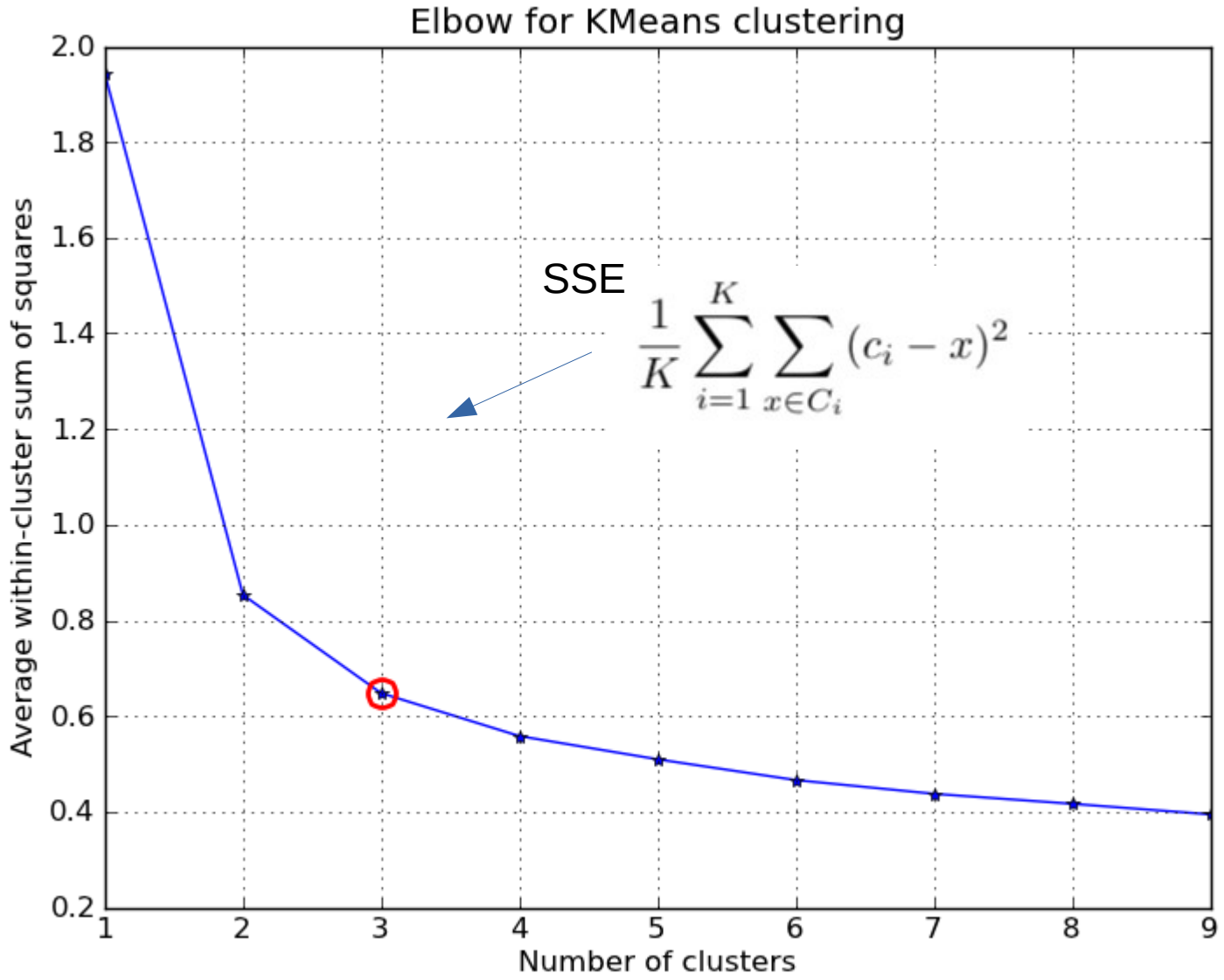
m_k: # elementos en el clúster

¿Cuántos prototipos usamos?

ELBOW (codo):

Variar k buscando el codo





SSE

$$\frac{1}{K}\sum_{i=1}^{K}\sum_{x \in C_i}(c_i - x)^2$$

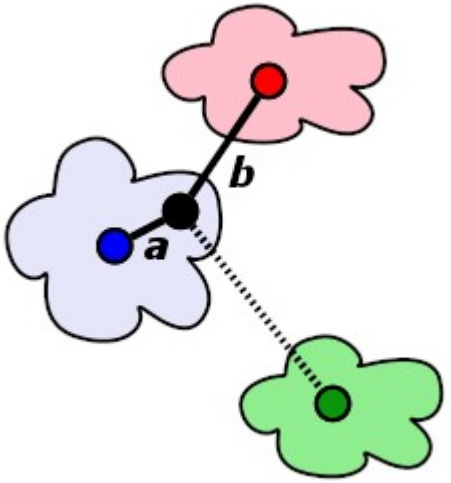
Esto es mala idea ya que SSE es monótono decreciente con k

¿Cuántos prototipos usamos?

Silhouette:

Congruencia de x_i a C_i: $$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i,j)$$

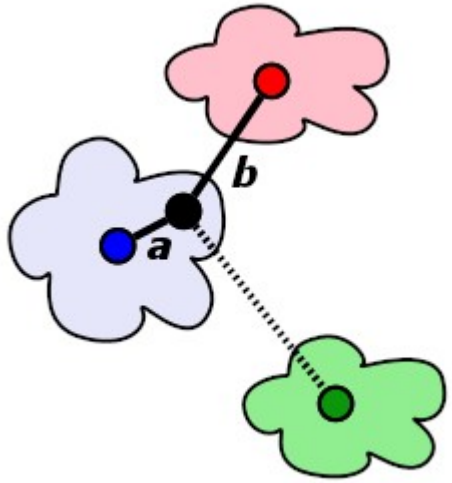
Congruencia de x_i a otros clusters: $$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i,j)$$



¿Cuántos prototipos usamos?

Silhouette:

Congruencia de x_i a C_i: $$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i,j)$$



Congruencia de x_i a otros clusters: $$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i,j)$$

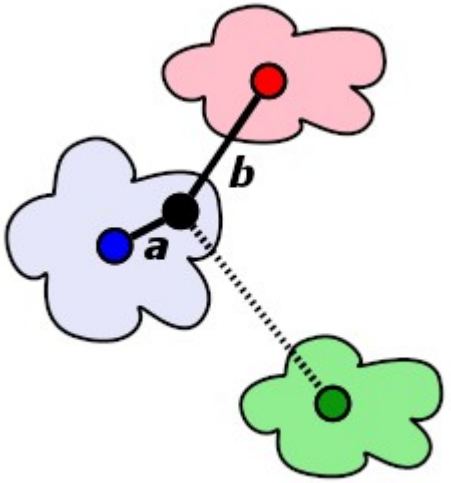
Silhouette Coef.: $$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \quad \text{si} \quad |C_i| > 1,$$

$$s(i) = 0, \quad \text{si} \quad |C_i| = 1.$$

¿Cuántos prototipos usamos?

Silhouette:

Congruencia de x_i a C_i: $$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i,j)$$



Congruencia de x_i a otros clusters: $$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i,j)$$

Silhouette Coef.: $$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \quad \text{si} \quad |C_i| > 1,$$

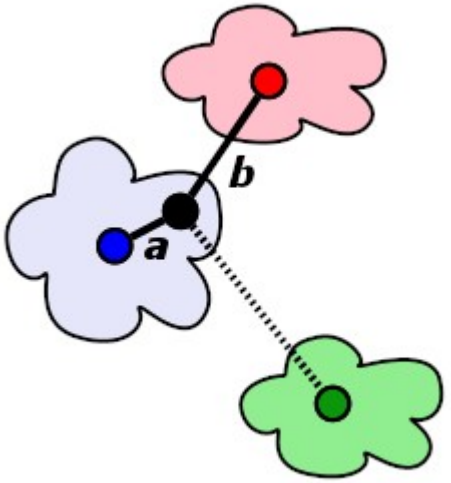
$$s(i) = 0, \quad \text{si} \quad |C_i| = 1.$$

¿Intervalo?

¿Cuántos prototipos usamos?

Silhouette:

Congruencia de x_i a C_i: $$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i,j)$$



Congruencia de x_i a otros clusters: $$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i,j)$$

Silhouette Coef.: $$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \quad \text{si} \quad |C_i| > 1,$$

$$s(i) = 0, \quad \text{si} \quad |C_i| = 1.$$

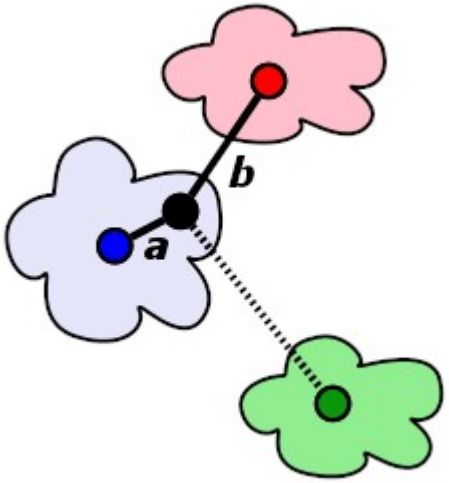
[-1, 1]

¿Cuántos prototipos usamos?

Silhouette:

Un valor bajo indica alta congruencia con su cluster

Congruencia de x_i a C_i: $$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i,j)$$



Congruencia de x_i a otros clusters: $$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i,j)$$

Un valor alto indica alta congruencia con su cluster

Silhouette Coef.: $$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \quad \text{si} \quad |C_i| > 1,$$

$$s(i) = 0, \quad \text{si} \quad |C_i| = 1.$$

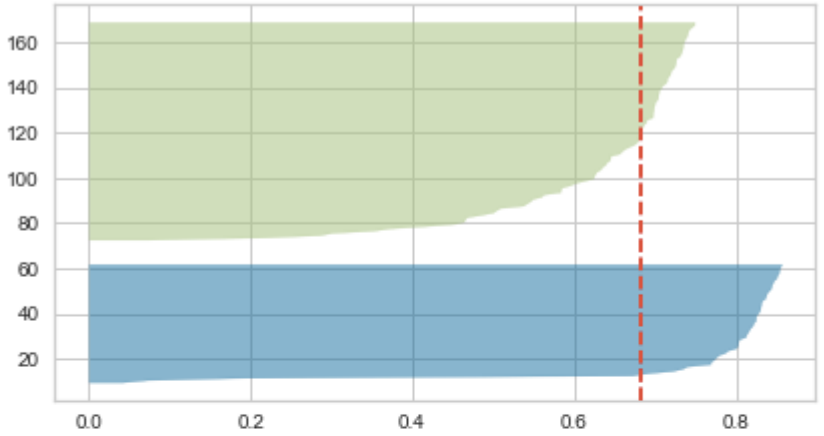
Un valor alto indica alta congruencia con su cluster

[-1, 1]

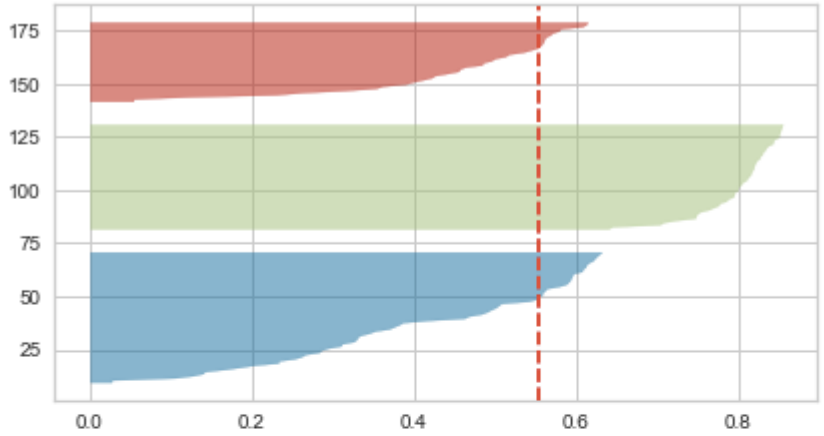
¿Cuántos prototipos usamos?

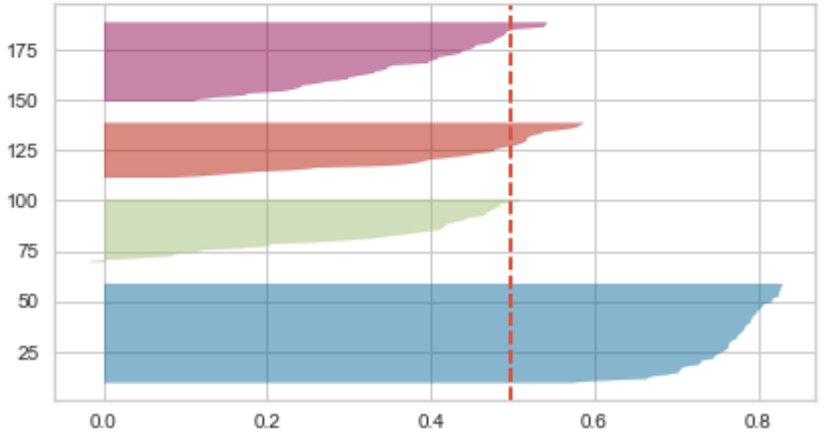
Silhouette promedio:

k=2

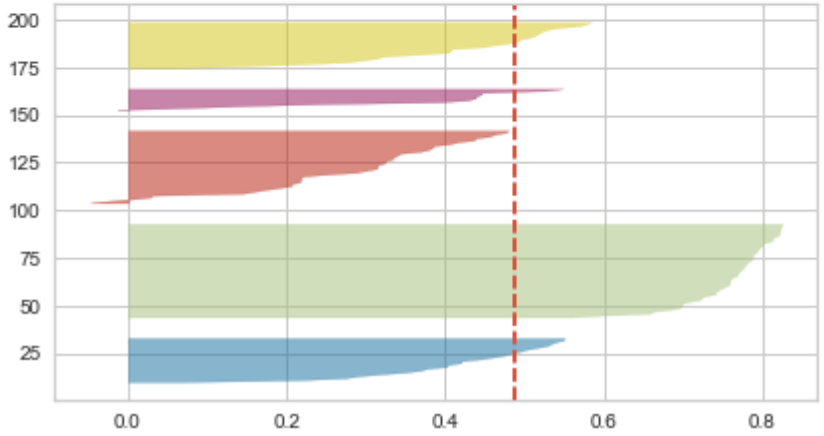


k=3





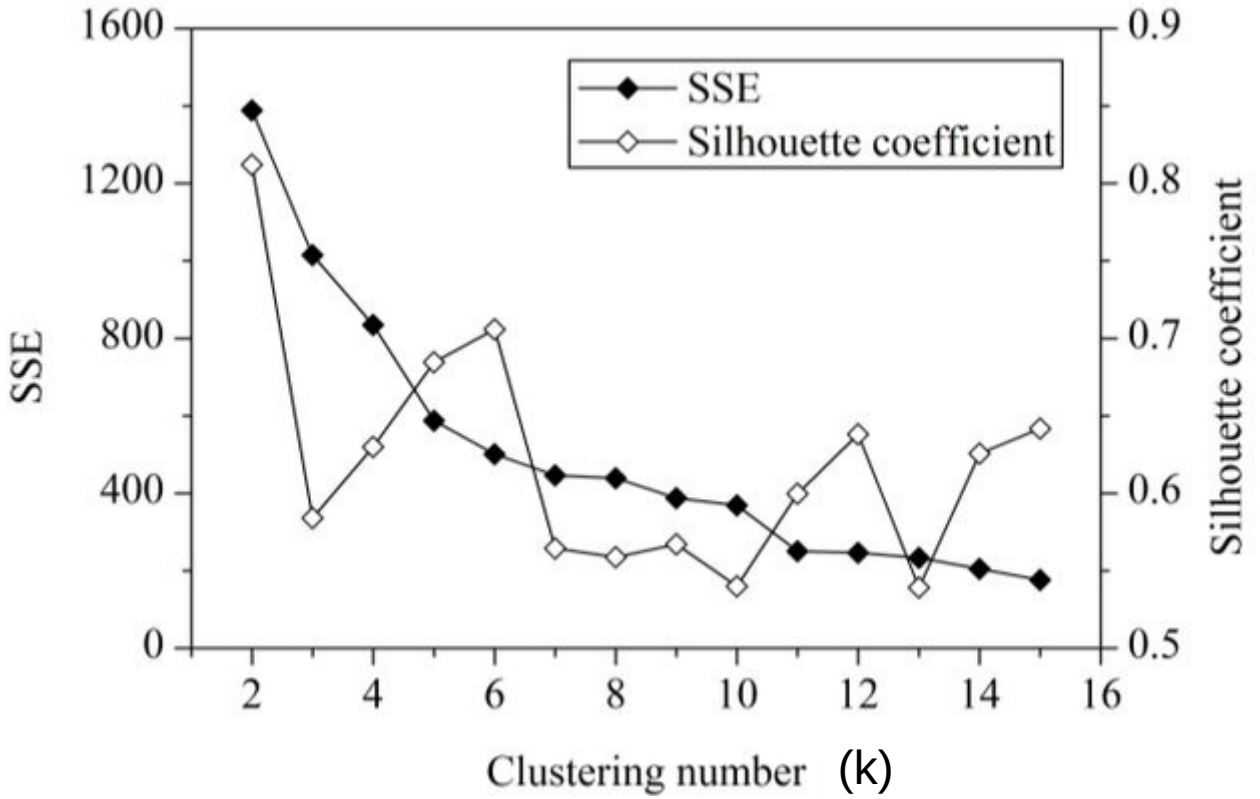
k=4



k=5

¿Cuántos prototipos usamos?

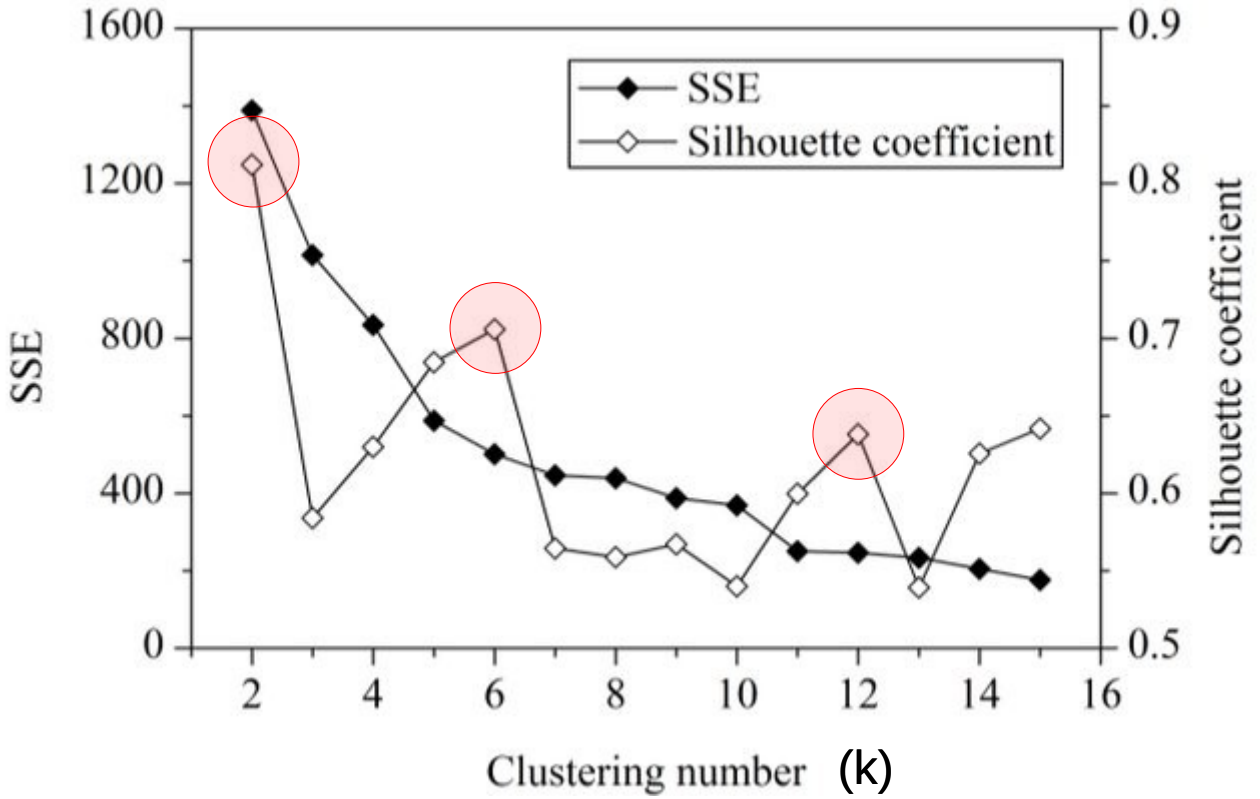
Silhouette v/s ELBOW:



¿Con cuál **k** se quedan?

¿Cuántos prototipos usamos?

Silhouette v/s ELBOW:



¿Con cuál **k** se quedan?