



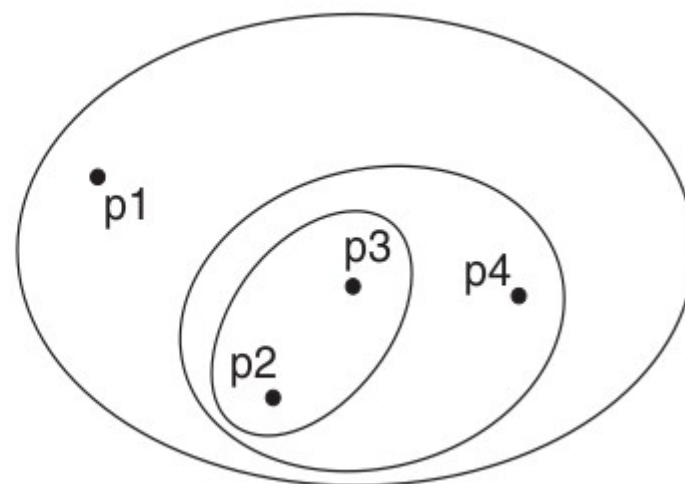
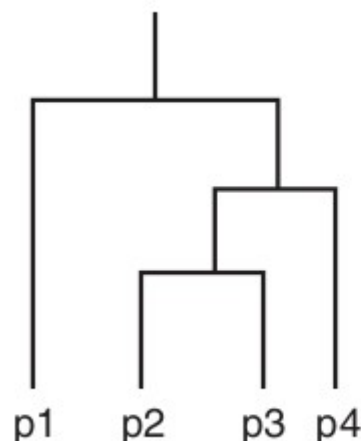
# IIC 2433 Minería de Datos

<https://github.com/marcelomendoza/IIC2433>

- HAC -

# Clustering Jerárquico

Idea:



---

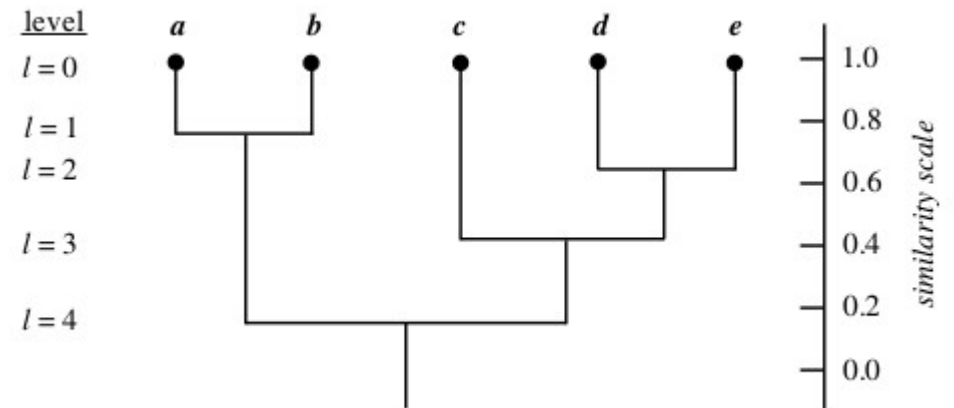
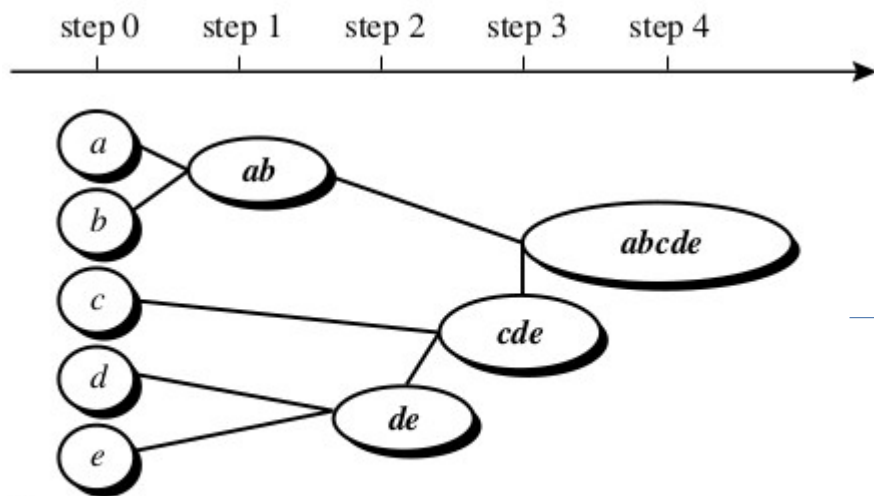
**Algorithm**    Basic agglomerative hierarchical clustering algorithm.

---

- 1: Compute the proximity matrix, if necessary.
  - 2: **repeat**
  - 3:    Merge the closest two clusters.
  - 4:    Update the proximity matrix to reflect the proximity between the new cluster and the original clusters.
  - 5: **until** Only one cluster remains.
-

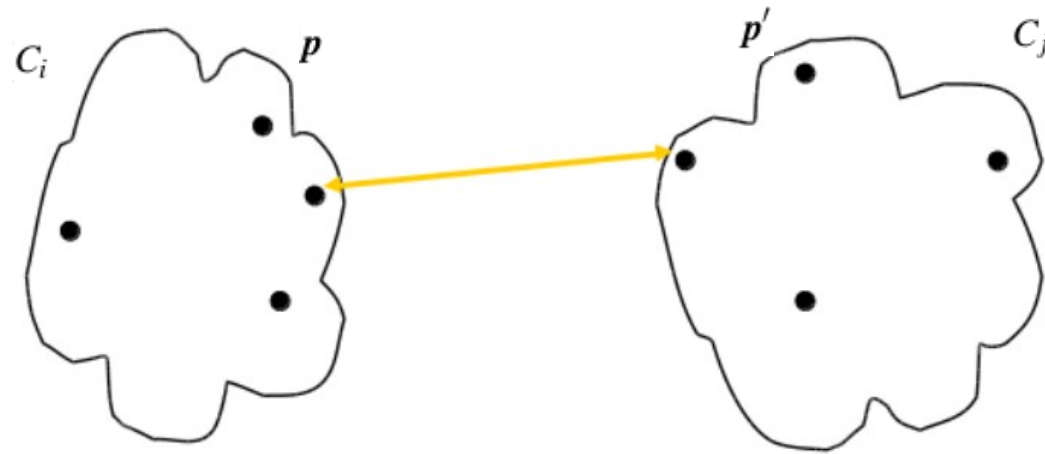
# Clustering Jerárquico

aglomerativo



# Clustering Jerárquico Aglomerativo

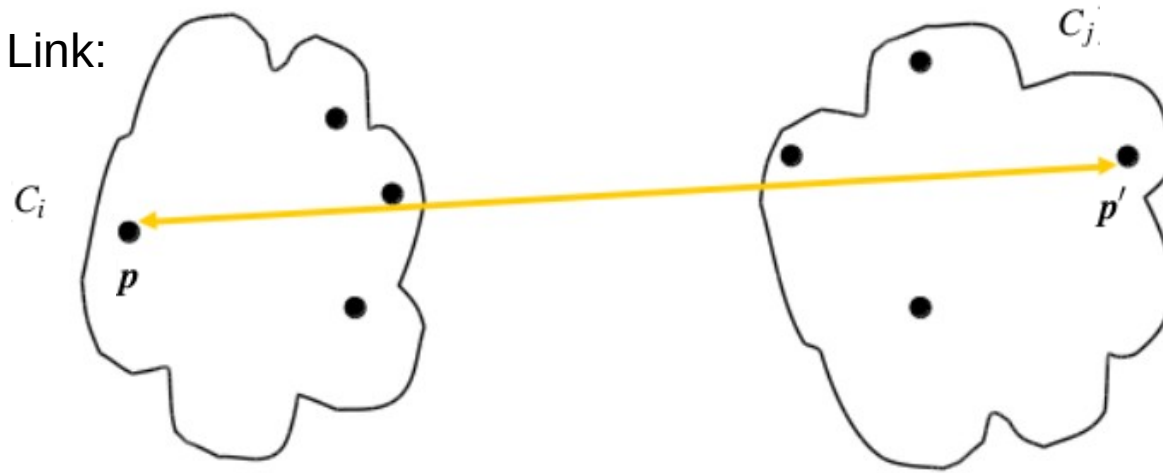
Single Link:



$$d_{min}(C_i, C_j) = \min_{p \in C_i, p' \in C_j} |p - p'|$$

# Clustering Jerárquico Aglomerativo

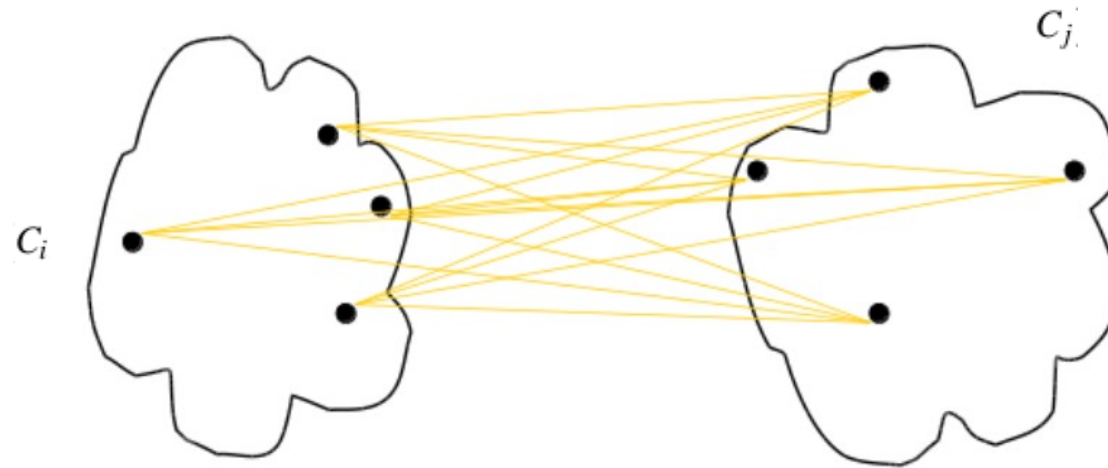
Complete Link:



$$d_{max}(C_i, C_j) = \max_{p \in C_i, p' \in C_j} |p - p'|$$

## Clustering Jerárquico Aglomerativo

Average Link:



$$d_{avg}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{p \in C_i} \sum_{p' \in C_j} |p - p'|$$

# Clustering Jerárquico Aglomerativo

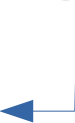
Método de Ward:

#datos de cada cluster



$$\Delta(A, B) = \sum_{i \in A \cup B} \|\vec{x}_i - \vec{m}_{A \cup B}\|^2 - \sum_{i \in A} \|\vec{x}_i - \vec{m}_A\|^2 - \sum_{i \in B} \|\vec{x}_i - \vec{m}_B\|^2 = \frac{n_A n_B}{n_A + n_B} \|\vec{m}_A - \vec{m}_B\|^2$$

Centroide del nuevo cluster





# Clustering Jerárquico Aglomerativo

Método de Ward:

#datos de cada cluster



$$\Delta(A, B) = \sum_{i \in A \cup B} \|\vec{x}_i - \vec{m}_{A \cup B}\|^2 - \sum_{i \in A} \|\vec{x}_i - \vec{m}_A\|^2 - \sum_{i \in B} \|\vec{x}_i - \vec{m}_B\|^2 = \frac{n_A n_B}{n_A + n_B} \|\vec{m}_A - \vec{m}_B\|^2$$

Centroide del nuevo cluster



$$\sim d_{mean}(C_i, C_j) = |\mathbf{m}_i - \mathbf{m}_j|$$

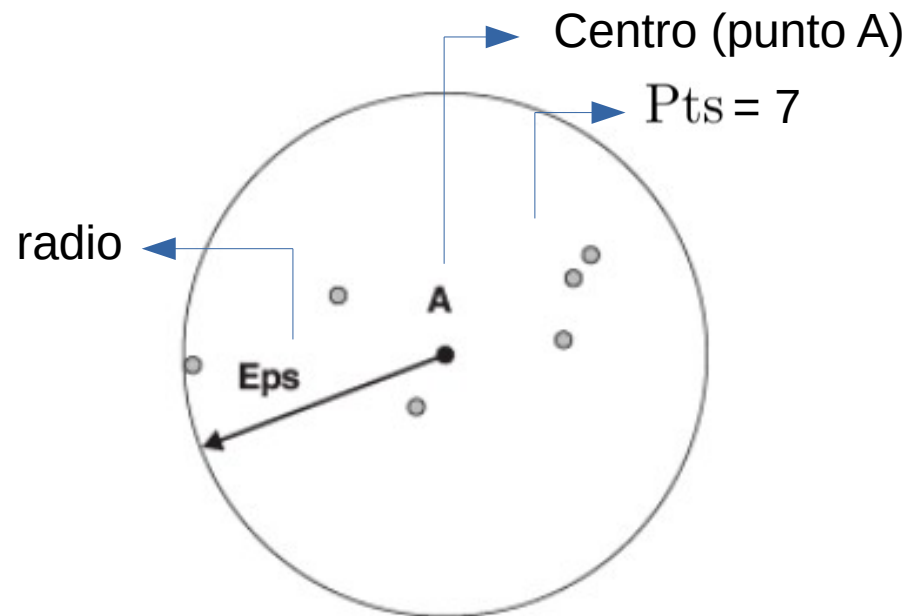
- DBSCAN Y OPTICS -

# DBSCAN

## Density-based clustering

Idea: Interpretar regiones de alta densidad como clusters.

Enfoque: Center-based density



Noción de densidad: Circunferencia centrada en **A** de radio **Eps** tal que contiene al menos **MinPts** vecinos.

# DBSCAN

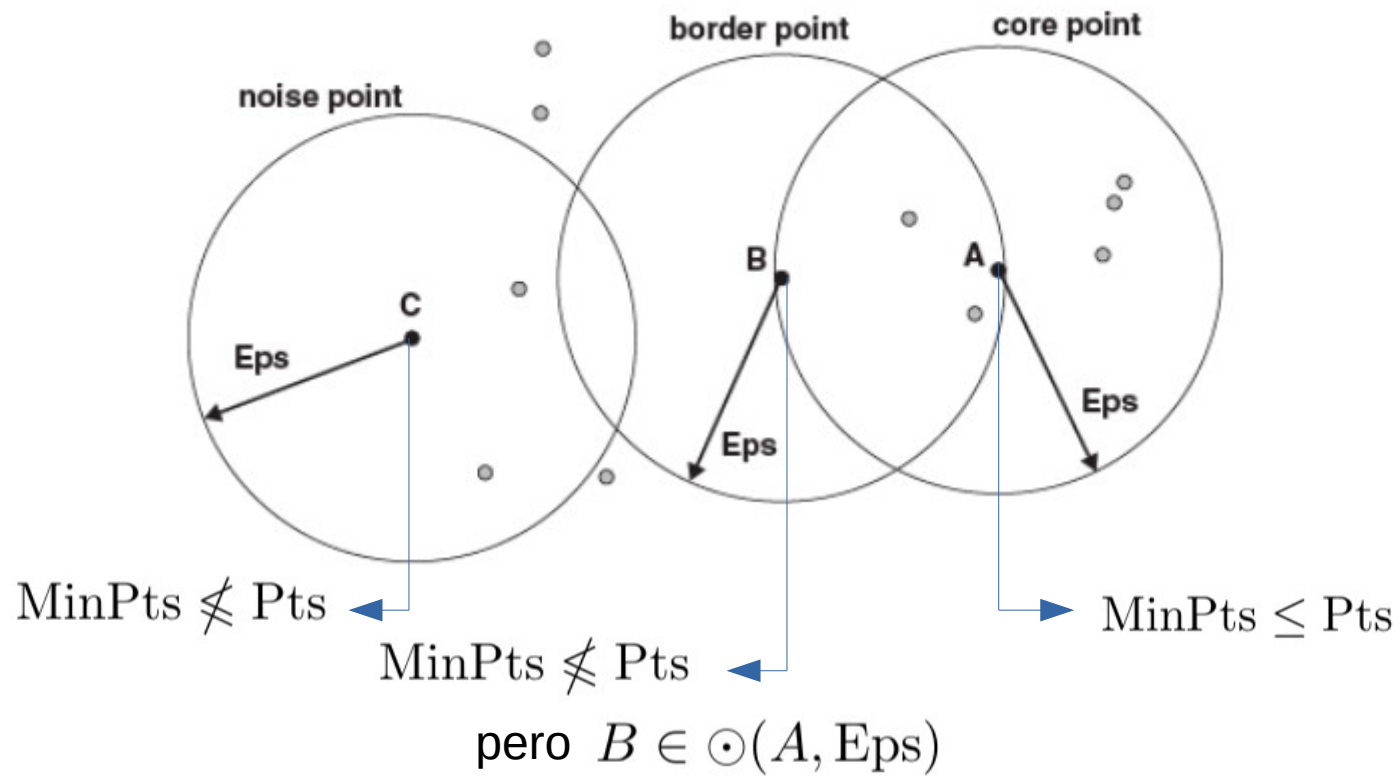
La noción de densidad centrada en puntos nos permite clasificar los datos en tres categorías:

Dado  $\text{MinPts}$  y  $\text{Eps}$  :  hiperparámetros

- **Core point**: un dato es un **core point** si la circunferencia de radio  $\text{Eps}$  centrada en torno del dato cumple que  $\text{MinPts} \leq \text{Pts}$
- **Border point**: un dato es un **border point** si no es un core point pero pertenece al vecindario de un core point.
- **Noise point**: Un dato es un **noise point** si no cumple con ninguna de las definiciones anteriores.

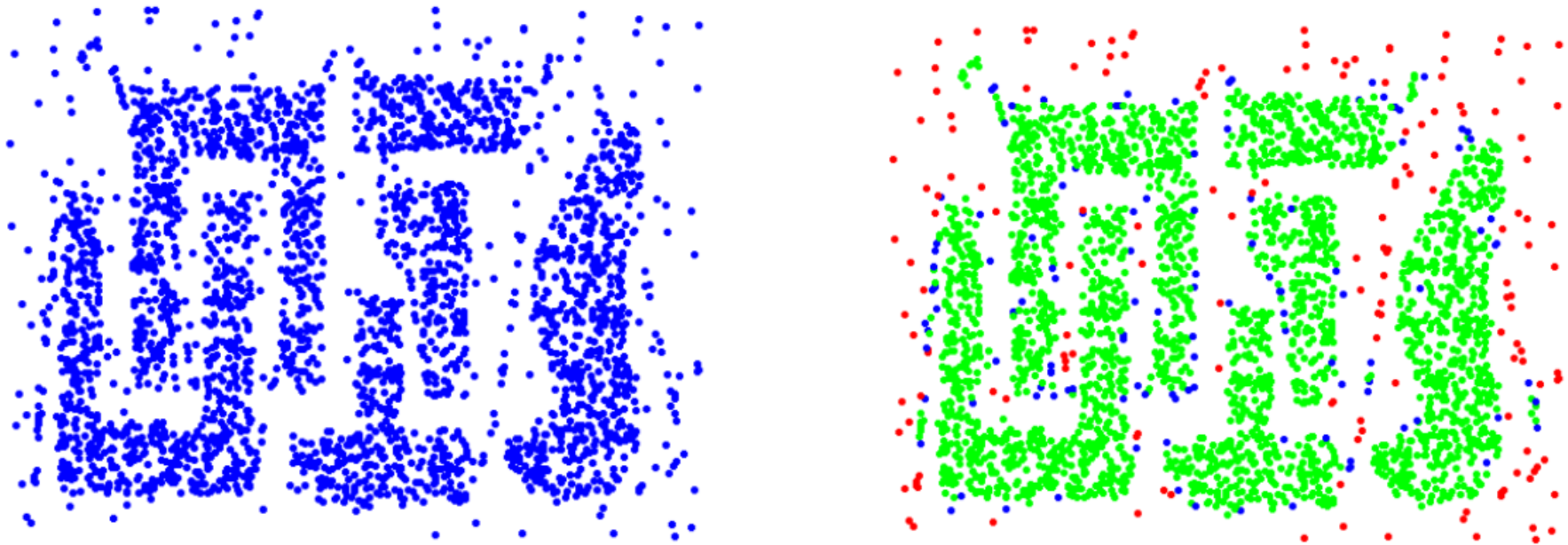
# DBSCAN

MinPts = 7



# DBSCAN

Ejemplo:



Core, border y noise points (verde, azul y rojo, resp.)


# DBSCAN


Algoritmo:

---

**Algorithm** DBSCAN algorithm.

---

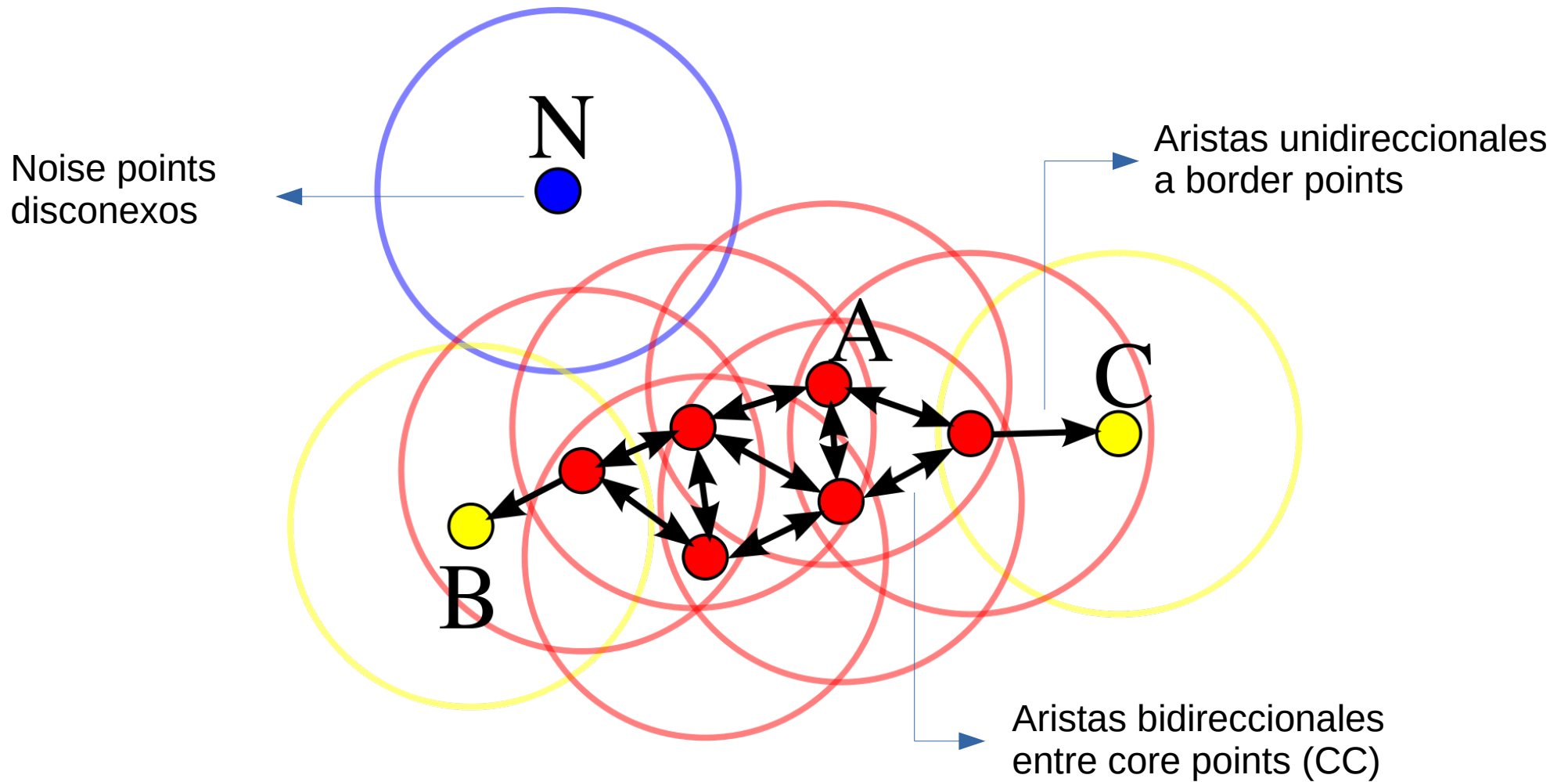
- 1: Label all points as core, border, or noise points.
  - 2: Eliminate noise points.
  - 3: Put an edge between all core points that are within  $Eps$  of each other. 
  - 4: Make each group of connected core points into a separate cluster.
  - 5: Assign each border point to one of the clusters of its associated core points.
- 



DBSCAN construye un grafo de vecinos cercanos y lo colorea usando componentes conexas

# DBSCAN

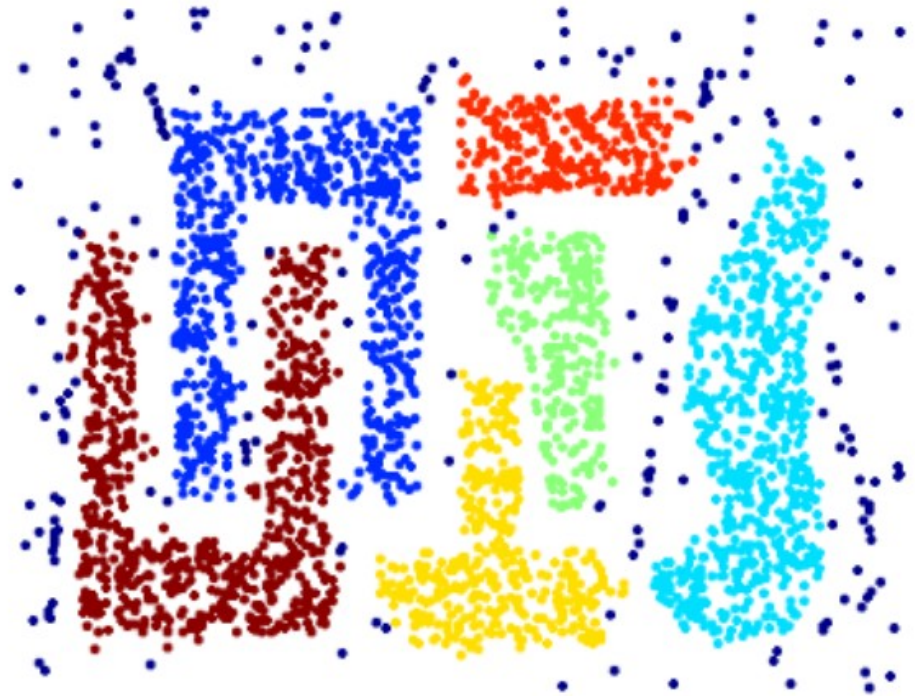
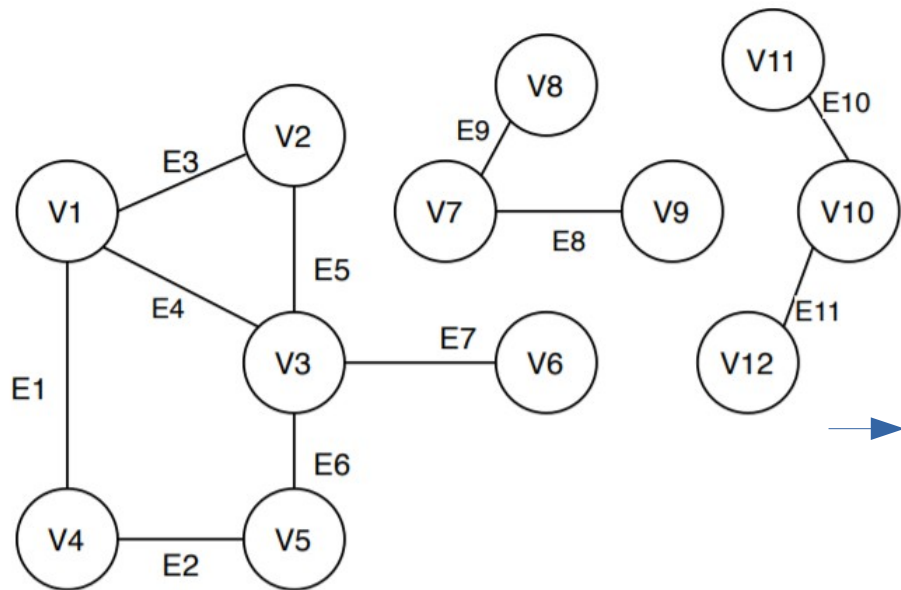
Grafo dirigido construido conectando core points y border points:





# DBSCAN

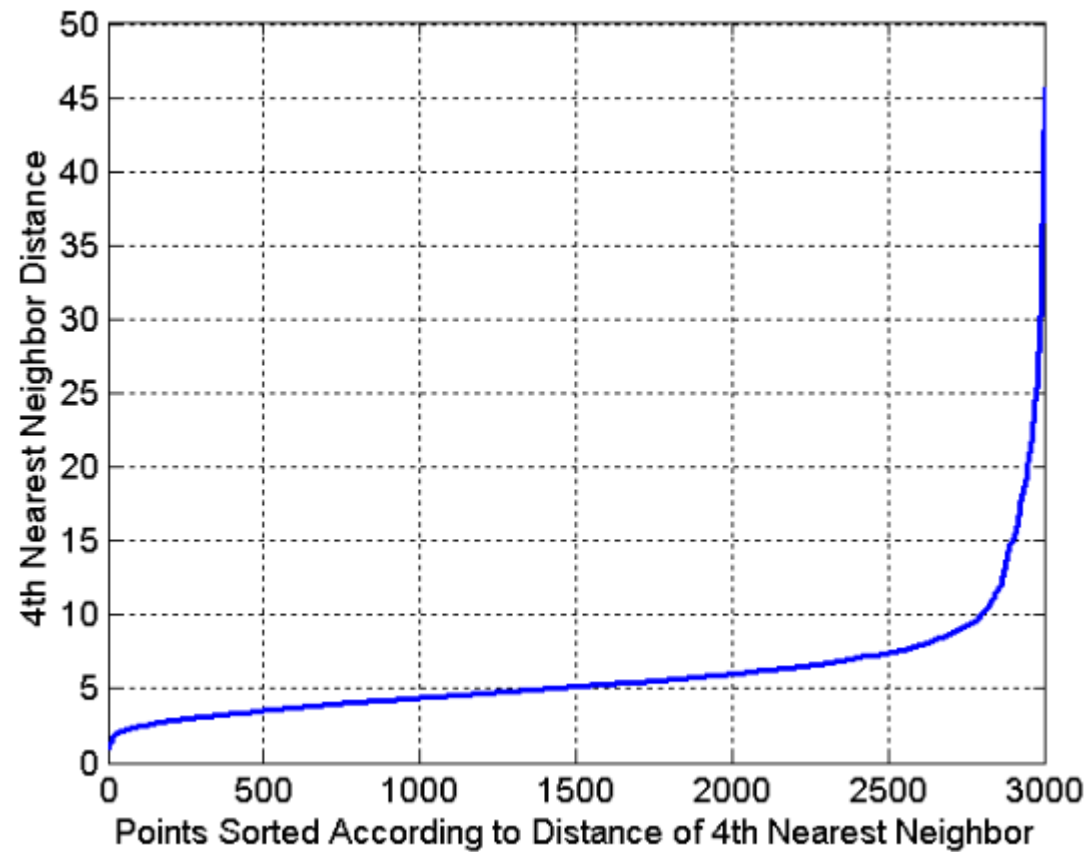
Componentes conexas en DBSCAN:



# DBSCAN

Sintonización del algoritmo

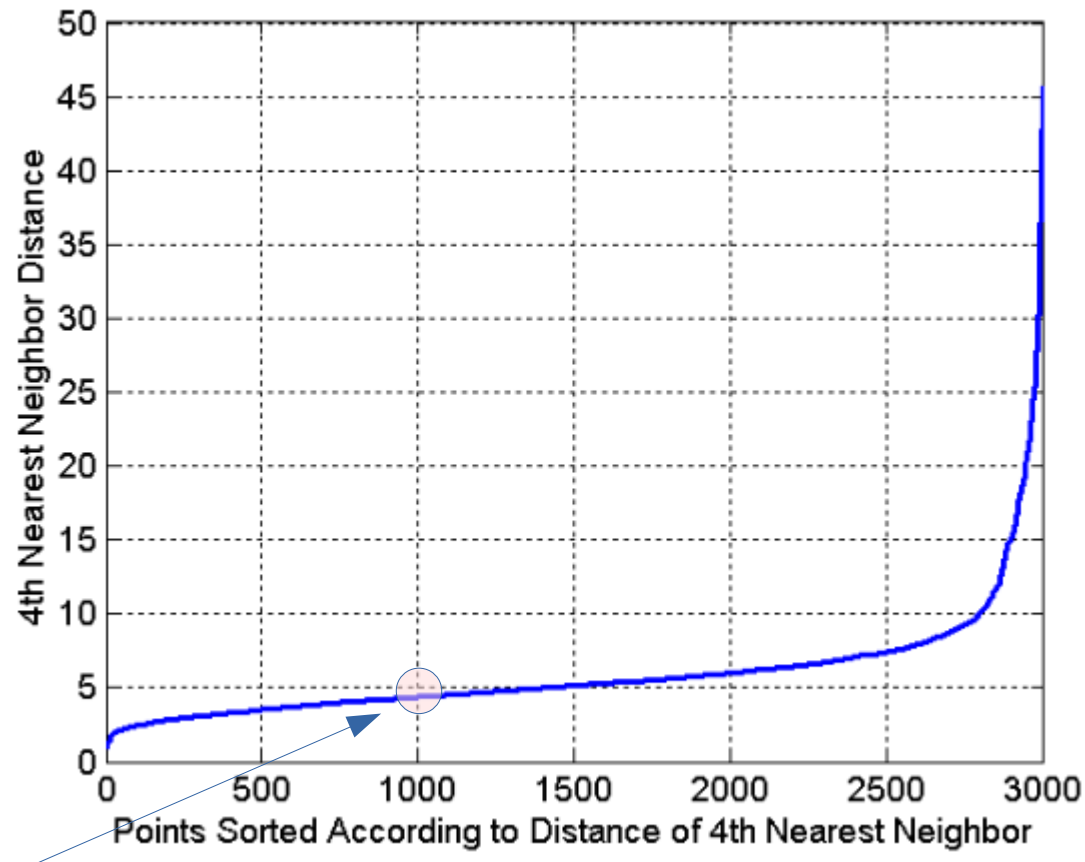
$k$ -dist plot ( $k=4$ ) ← MinPts (candidato)



# DBSCAN

## Sintonización del algoritmo

$k$ -dist plot ( $k=4$ ) ← MinPts (candidato)



1000 puntos tienen a lo más  
distancia = 5 a su 4º vecino

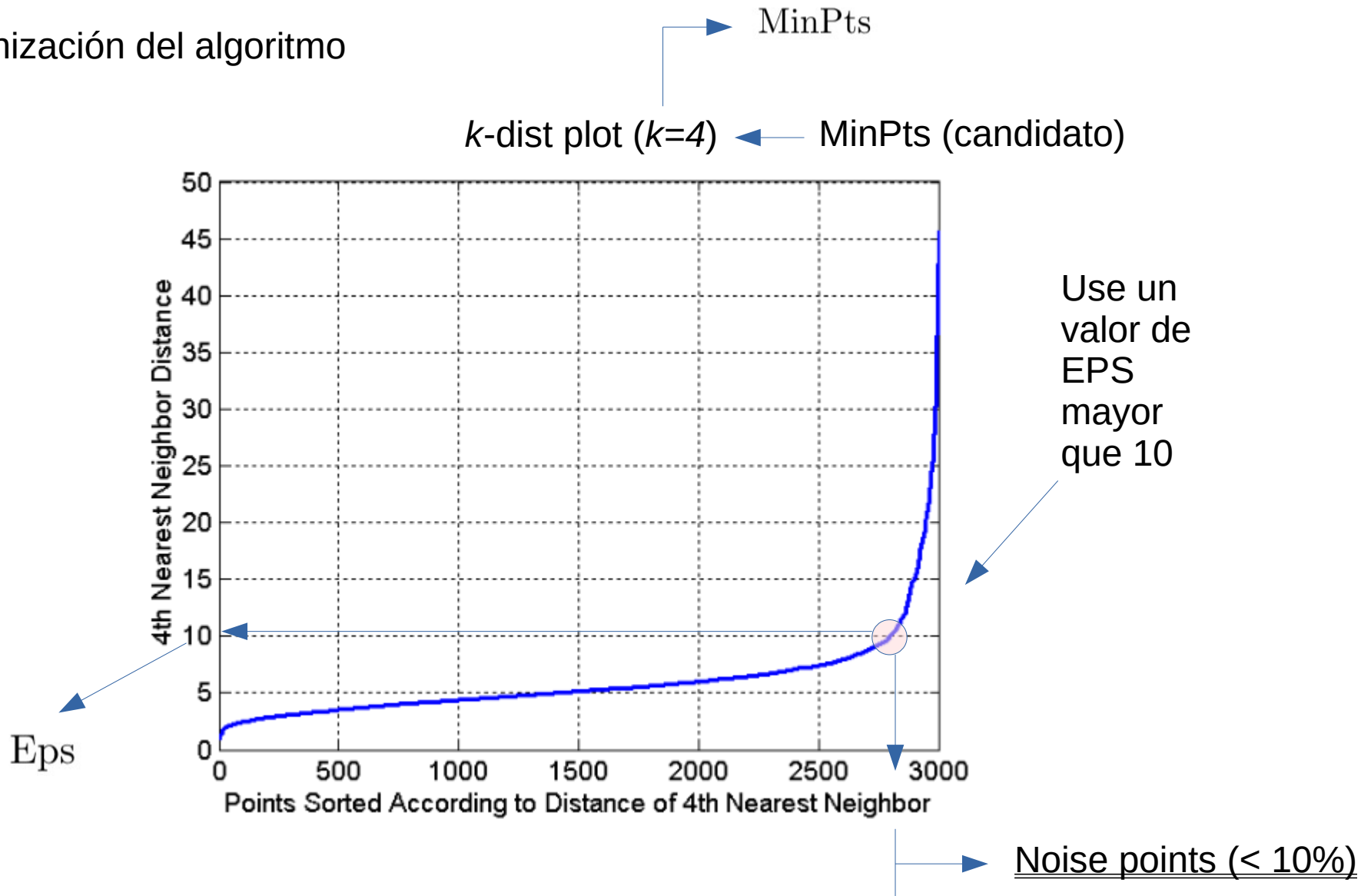
- UC - M. Mendoza -

→ Si  $EPS = 5$  y  $MinPts = 5 \rightarrow 1000$  core points

Notar que si aumento EPS, los clusters son menos densos

## DBSCAN

Sintonización del algoritmo



# HDBSCAN

Intuición: separar las regiones de baja y alta densidad en base a comparaciones de  $k$ -distances. Le denominan core distance.

Luego definen la mutual reachability distance entre dos puntos como:

$$d_{\text{mreach-}k}(a, b) = \max\{\text{core}_k(a), \text{core}_k(b), d(a, b)\}$$

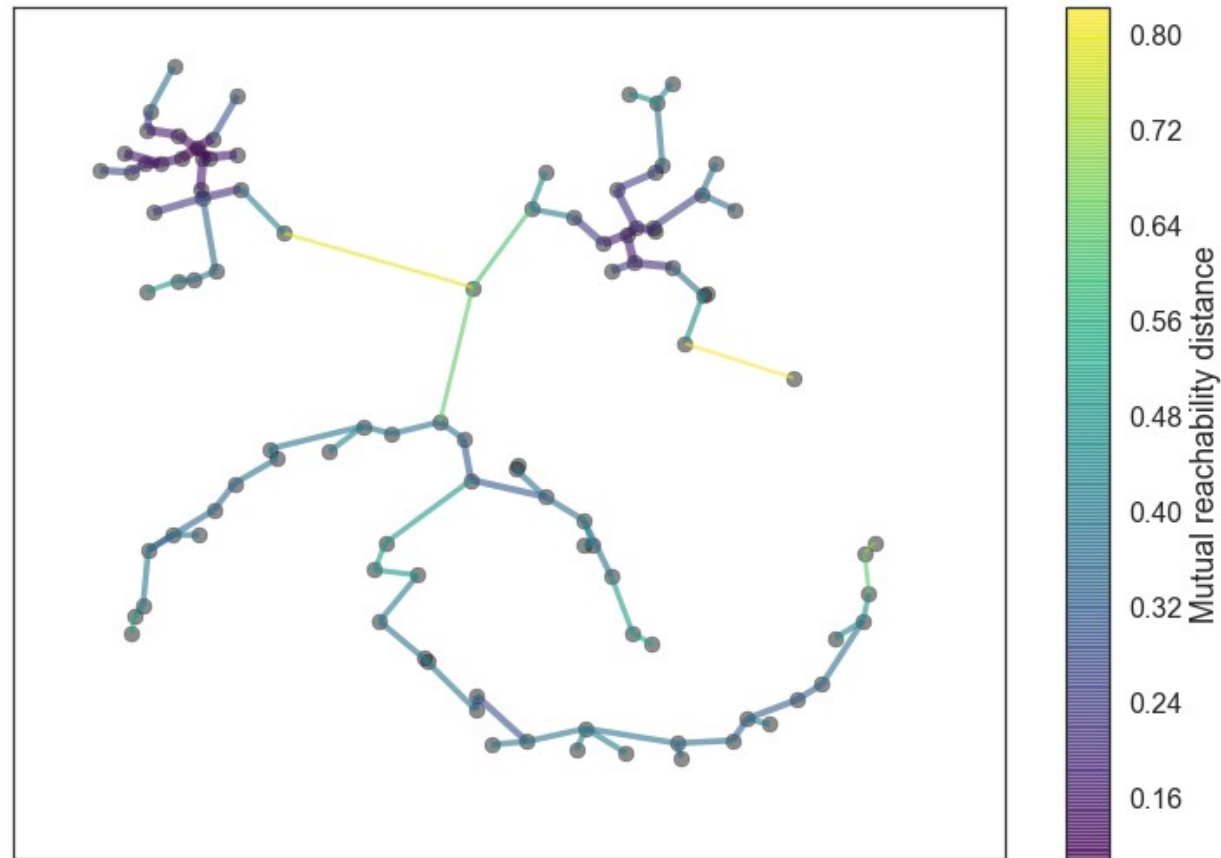
Bajo esta métrica los puntos de áreas densas se mantienen a la misma distancia (core distance) pero los puntos en áreas sparse son empujados para estar al menos a su core distance del resto de los puntos.

La distancia ayuda a separar los puntos de áreas *sparse*, como el efecto de bajar el nivel del mar para que surjan islas (clusters):



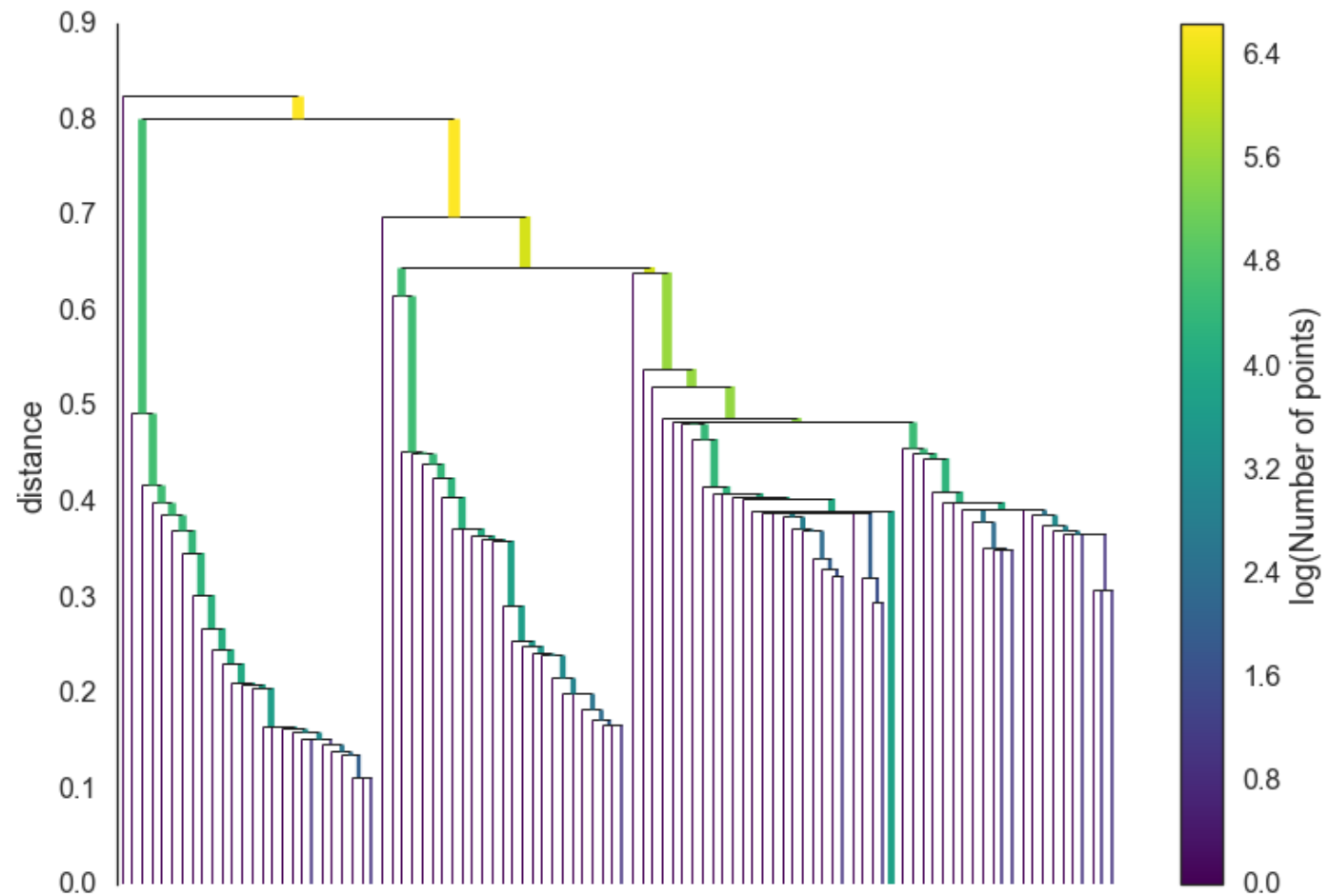
# HDBSCAN

- Tenemos el grafo de distancias entre los datos en base a la mutual reachability distance.
- Vamos a construir el minimum spanning tree usando el algoritmo de Prim. El invariante del algoritmo es agregar la arista de menor peso que conecta al árbol actual un nodo que no está conectado.



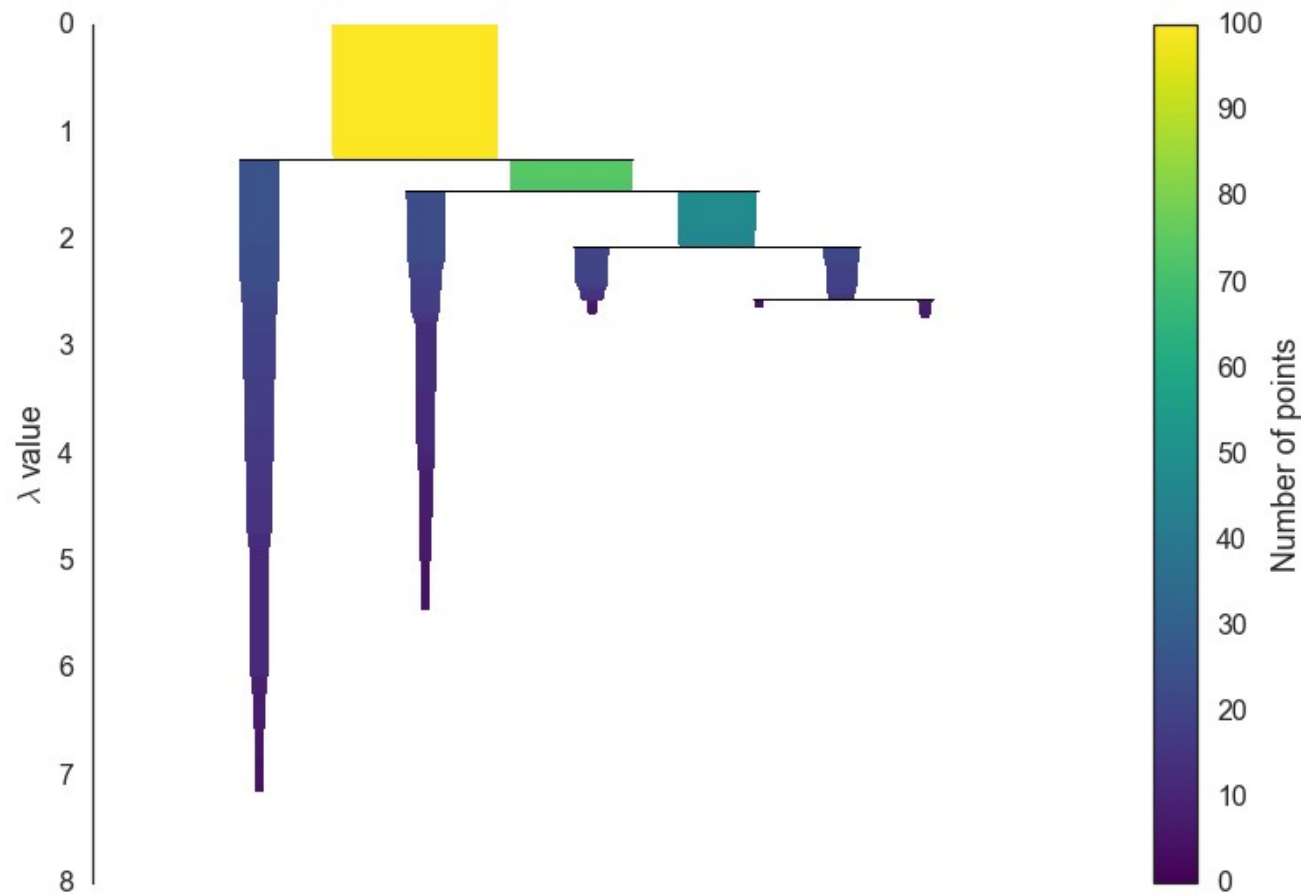
# HDBSCAN

- Ahora construiremos una jerarquía de componentes conexas, ordenando las aristas del árbol de forma creciente (bottom-up, HAC).



# HDBSCAN

- Finalmente condensaremos el dendrograma con una condición de **minimum cluster size**. Si un cluster tiene menos datos, se mezcla con su cluster padre (merge-up). Si un split produce dos clusters que cumplen con la condición de tamaño, el split se mantiene.





# HDBSCAN

Se usa el concepto de **cluster estable** para indicar cuando un cluster sobrevive a los splits definidos por un umbral de distancia en el dendrograma. Es inverso a la distancia, por lo que se define la variable lambda como  $1/d$ .

Hacemos un recorrido top-down del dendrograma. Vamos a tener un lambda de nacimiento del cluster, y en la medida que  $d$  disminuya (cortamos más abajo) en algún momento el cluster va a morir. Para cada dato del cluster vamos medir cuando sale del cluster (lambda del dato), y calcularemos la **estabilidad** del cluster según:

$$\sum_{p \in \text{cluster}} (\lambda_p - \lambda_{\text{birth}})$$

# HDBSCAN

Ahora recorremos el dendrograma bottom-up. Si la suma de las estabilidades de los hijos es mayor que la de un cluster, definimos la estabilidad del cluster como la suma de las estabilidades de sus hijos. En otro caso, si el cluster es más estable que la suma de sus hijos, seleccionamos el cluster y deseleccionamos a sus hijos.

