IMT2200

INTRODUCCIÓN A CIENCIA DE DATOS

2022-2

CLUSTERING

CLASE 24

APRENDIZAJE NO SUPERVISADO

Algoritmos de aprendizaje de máquina donde **no existe un output conocido**, sino que se pide al algoritmo extraer conocimiento a partir de la data de entrada.

En general, hay **dos grandes clases** de algoritmos de aprendizaje no supervisado:

1. Clustering ➔ algoritmos para particionar la data en grupos disímiles pero con ítems similares.

2. Transformaciones ➔ algoritmos que crean una nueva representación de los datos que puede ser más fácil de entender para humano o máquinas en comparación a la representación original.

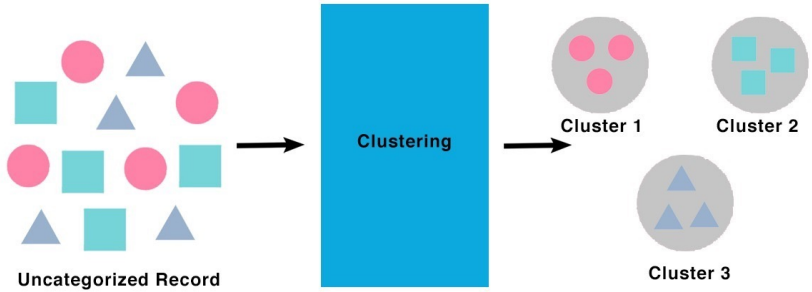
- Reducción de dimensionalidad.
- Extracción de tópicos

Desafío: evaluar si el algoritmo aprendió lo que se buscaba. Muchas veces se requiere inspeccionar o evaluar manualmente los resultados.

- Aprendizaje no supervisado se utiliza muchas veces en una etapa de exploración, o como parte del preprocesamiento para algoritmos supervisados.

CLUSTERING

- Algoritmos para particionar la data en grupos disímiles pero con ítems similares.
- Cada observación del dataset es asignada a sólo una categoría según el valor de las variables consideradas para la clasificación.
- Los clusters se definen de manera que cada uno agrupe observaciones similares entre sí, pero difieran entre grupos.
- El análisis se simplifica enfocándose sólo en los perfiles de cada cluster, que permiten interpretar la estructura de data multi-variable.

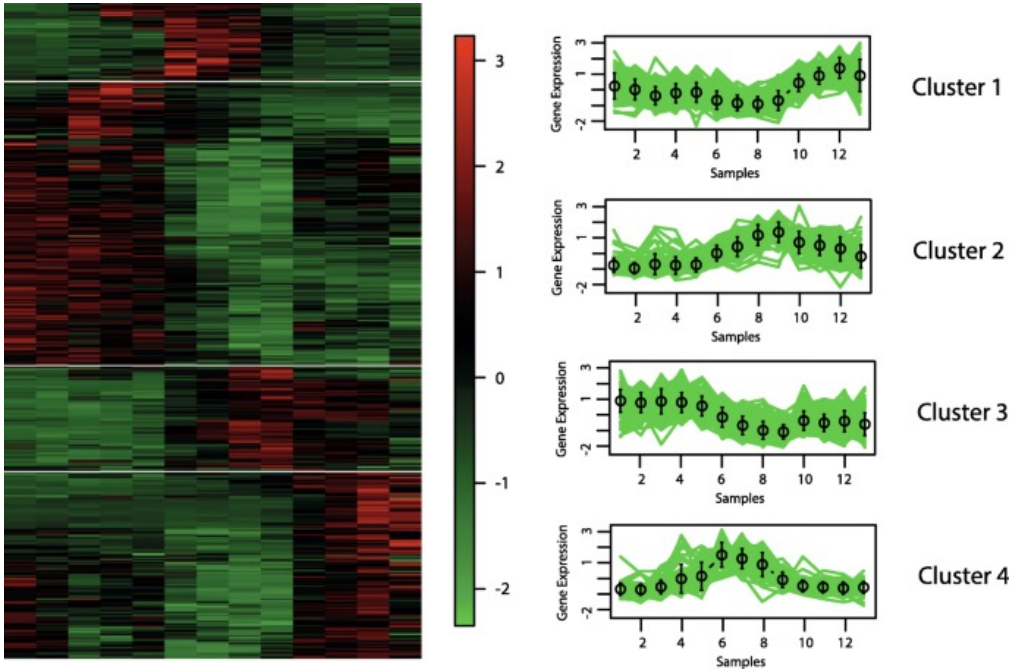


CLUSTERING: EJEMPLOS

A multi-objective gene clustering algorithm guided by apriori biological knowledge with intensification and diversification strategies

Jorge Parraga-Alava, Marcio Dorn & Mario Inostroza-Ponta

BioData Mining **11**, Article number: 16 (2018) | Cite this article

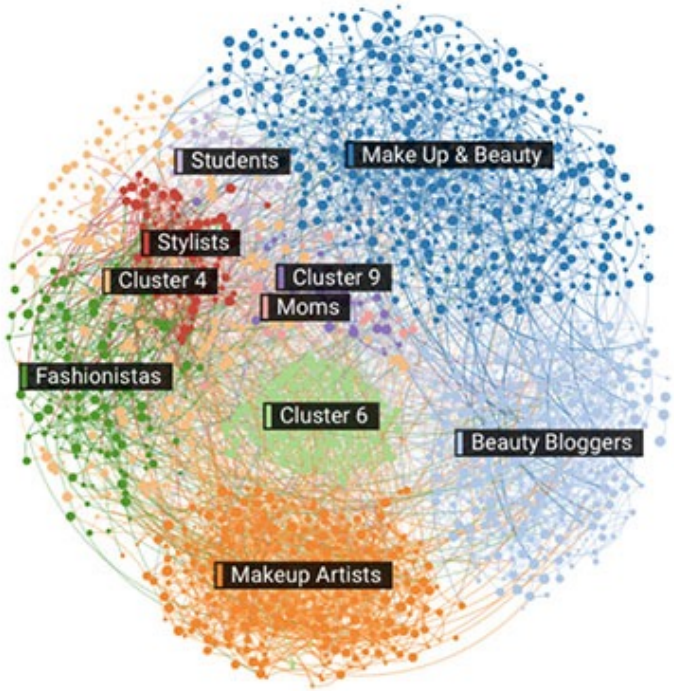


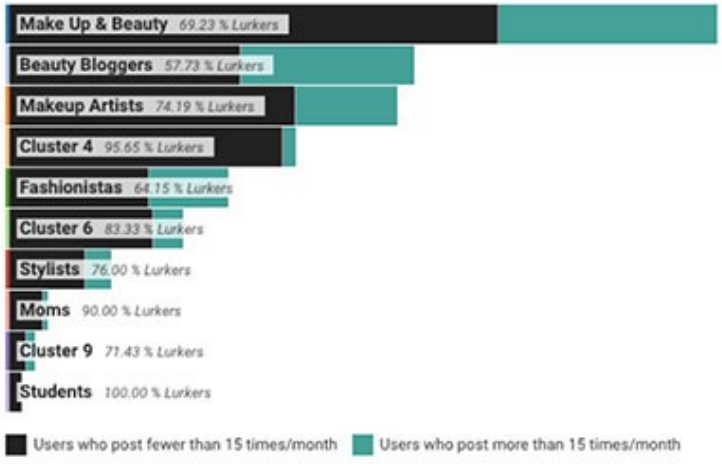
09:56

CLUSTERING: EJEMPLOS

Affinio Deep Learning Social Network Analysis Clustering

by Mariya Yao | Mar 5, 2018





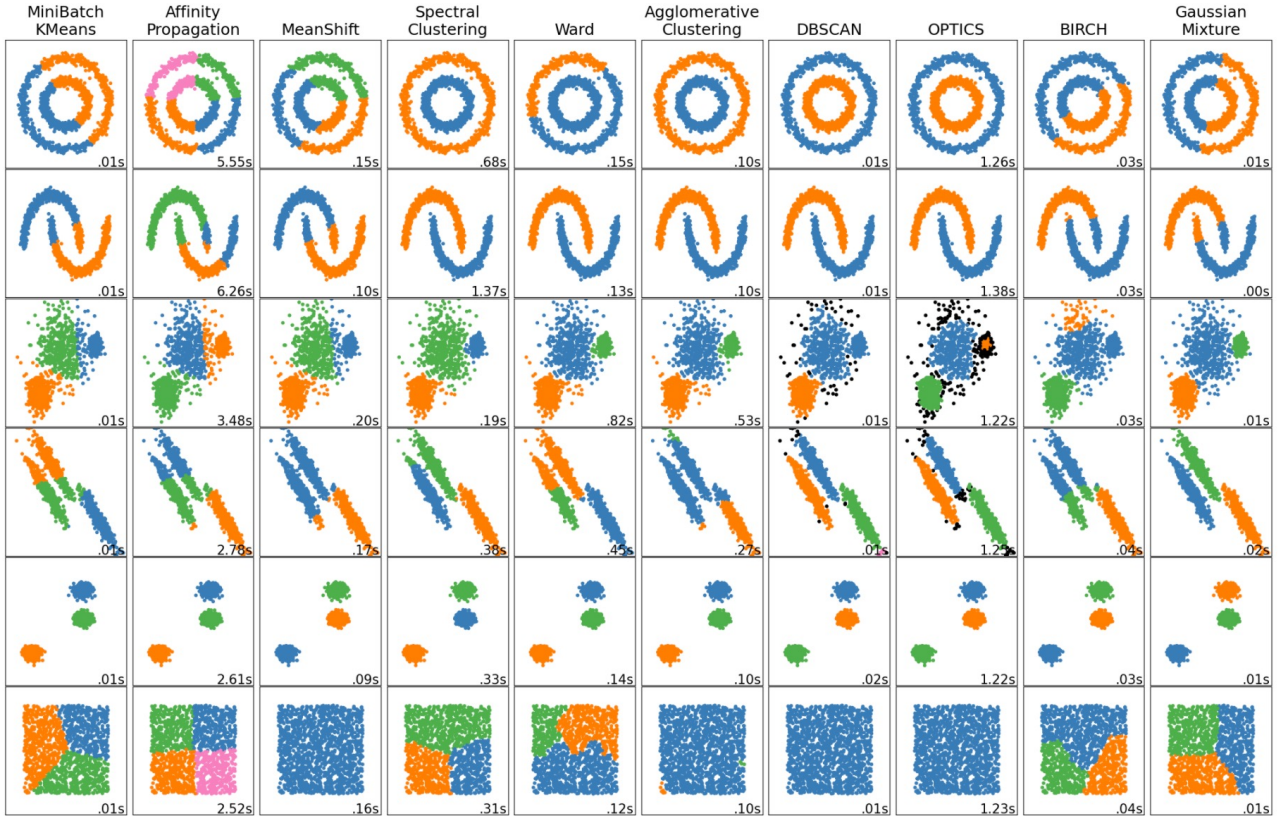
09:56

MÉTODOS DE CLUSTERING

Existen **múltiples tipos de clustering** que difieren en:

- Cómo se define la similaridad entre observaciones y disimilaridad entre grupos

- Operacionalización del algoritmo de clasificación



A comparison of the clustering algorithms in scikit-learn

09:56

MÉTODOS DE CLUSTERING

Algunos de los métodos más usados son:

- **Métodos de partición :** requieren que el usuario especifique de antemano el número de clusters.

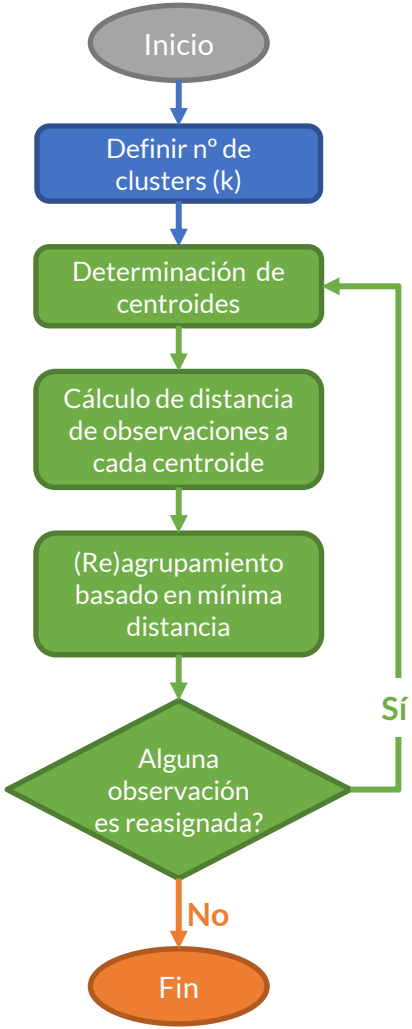
- **Métodos jerárquicos :** se busca crear una jerarquía para la fusión de observaciones fusionadas en clusters. No requiere que se pre-especifique el número de clusters.
  - Aglomerativos: el agrupamiento se inicia con todas las observaciones separadas, cada una formando un cluster individual. Los clusters se van combinado a medida que la estructura crece hasta converger en uno solo.
  - Divisivo: Se inicia con todas las observaciones contenidas en un mismo cluster y se suceden divisiones hasta que cada observación forma un cluster individual.

- **Métodos basados en densidad:** forma de identificar clusters siguiendo el modo intuitivo en el que lo hace el cerebro humano, identificando regiones con alta densidad de observaciones separadas por regiones de baja densidad.

CLUSTERING k-MEANS

- **Método de partición:** cada observación as asignada a uno de ***k* clusters (predefinido)**, de manera que la observación que está más cerca del centro de su cluster, que de cualquier otro.

- El problema se resuelve mediante un **proceso iterativo** de asignación y actualización:
  - Primero, se genera un centro aleatorio para cada cluster.
  - Se calcula la media multivariada (centroide) de cada cluster.
  - Se calcula la distancia entre los centros y observaciones para asignar cada dato al cluster más cercano.
  - Se recalcula el centro de los clusters
  - El proceso de continúa hasta que no son necesarias nuevas reasignaciones.

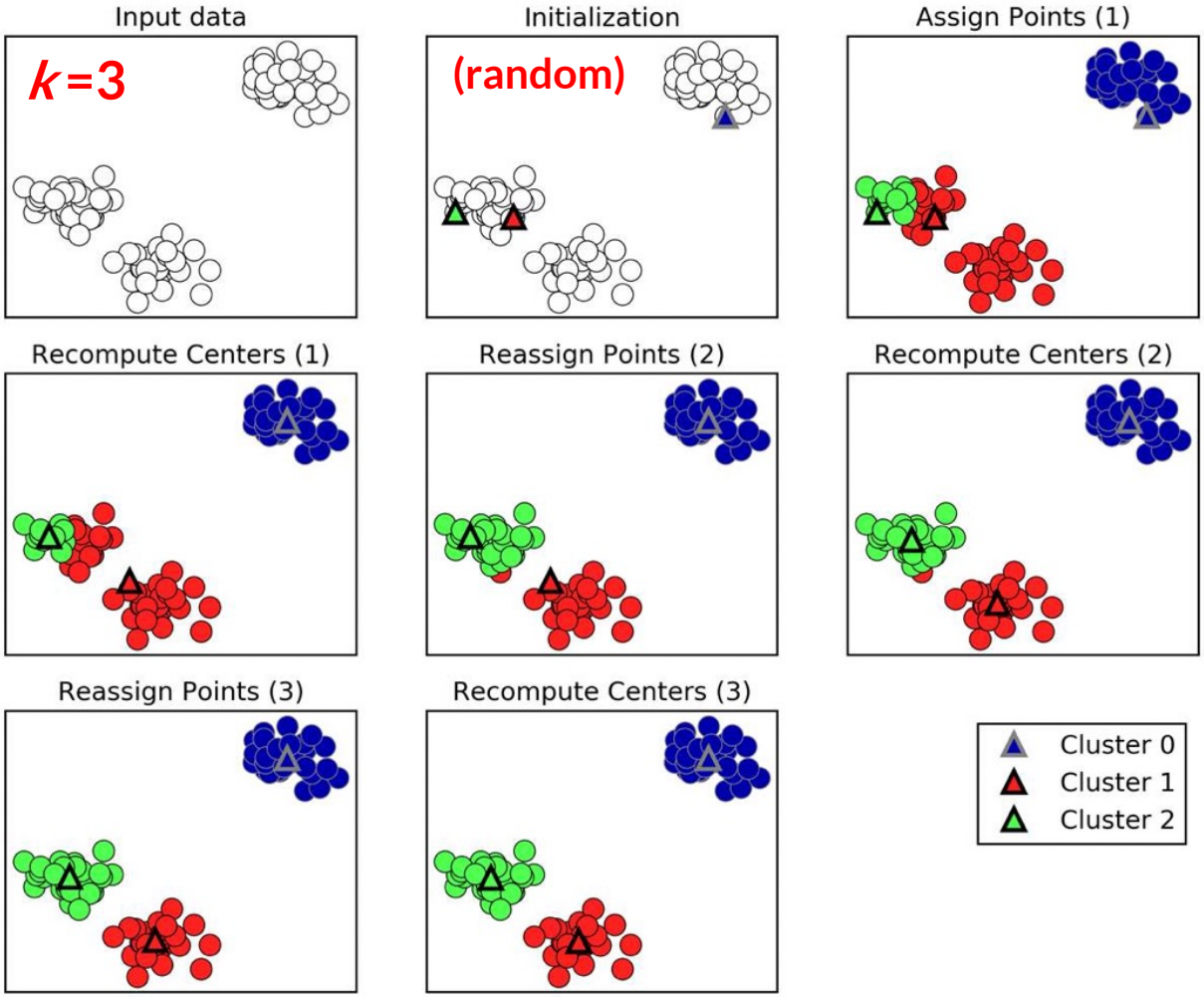


09:56

CLUSTERING
k-MEANS

El proceso continúa hasta que no son necesarias nuevas reasignaciones.

Inicialización aleatoria puede llevar a distintos resultados para un mismo conjunto de datos ➔ se repite el proceso varias veces, y se conserva el mejor resultado.



CLUSTERING k-MEANS

¿Cuándo termina el proceso?

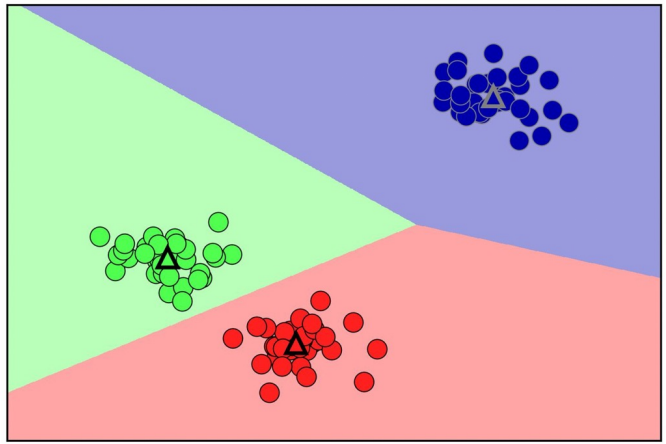
- El objetivo es lograr una agrupación que **minimice la disimilaridad dentro de cada cluster**.
- **Disimilaridad** ➔ suma de distancias euclideanas o errores cuadráticos (SSE) entre cada observación y el centroide de su cluster. También se denomina **inercia** de clusters.

$$SSE = \sum_{i=1}^{n} \sum_{j=i}^{k} w^{i,j} ||\boldsymbol{x}^i - \boldsymbol{\mu}^j||^2$$

Δ μ^j: centroide del cluster j

o x^i: observación i

$$w^{i,j} = \begin{cases} 1 \text{ si la observación } x^i \text{ está en el cluster } j \\ 0 \text{ en otro caso} \end{cases}$$



➔ El algoritmo continúa hasta que ya no es posible reducir SSE.

09:56

CLUSTERING k-MEANS: PREPROCESAMIENTO

- La función objetivo o **disimilaridad es sensible a la escala de las variables**:
  - Variables de un dataset pueden tener unidades no-comparables (Ej: dimensiones, m, $)
  - Variables con las mismas unidades pueden tener muy distintas escalas de valores y varianzas

- Si una variable tiene una escala mucho mayor que el resto, determinará en gran medida el valor de distancia/similitud obtenido al comparar las observaciones, dirigiendo así la agrupación final.

- Para evitar que las variables con mucha varianza dominen la función objetivo, se requiere **estandarizar las variables** antes de clusterizar:

  - **Estandarización z** ➔ restar la media y dividir por la desviación estándar $z = \frac{x - \mu}{\sigma}$

09:56

CLUSTERING k-MEANS: IMPLEMENTACIÓN

sklearn.cluster.KMeans

class sklearn.cluster.KMeans(*n_clusters=8*, ***, *init='k-means++'*, *n_init=10*, *max_iter=300*, *tol=0.0001*, *verbose=0*, *random_state=None*, *copy_x=True*, *algorithm='auto'*) [source]

Parameters::

n_clusters : ***int, default=8***

The number of clusters to form as well as the number of centroids to generate.

init : ***{'k-means++', 'random'}, callable or array-like of shape (n_clusters, n_features), default='k-means++'***

Method for initialization:

'k-means++' : selects initial cluster centroids using sampling based on an empirical probability distribution of the points' contribution to the overall inertia. This technique speeds up convergence, and is theoretically proven to be $\mathcal{O}(\log k)$-optimal. See the description of `n_init` for more details.

'random': choose `n_clusters` observations (rows) at random from data for the initial centroids.

If an array is passed, it should be of shape (n_clusters, n_features) and gives the initial centers.

If a callable is passed, it should take arguments X, n_clusters and a random state and return an initialization.

n_init : ***int, default=10***

Number of time the k-means algorithm will be run with different centroid seeds. The final results will be the best output of n_init consecutive runs in terms of inertia.

09:56

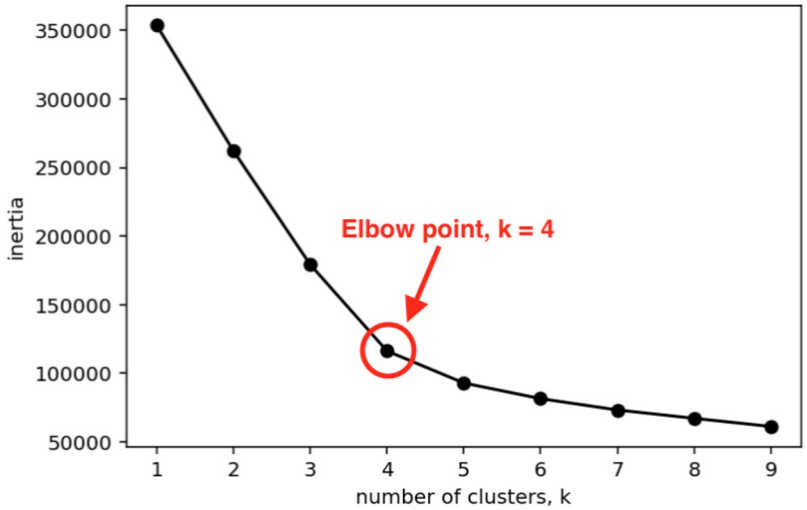
CLUSTERING k-MEANS: SELECCIÓN DE *k*

El algoritmo requiere definir a priori el **número de clusters (k):** ¿cómo lo elegimos?

Típicamente, se consideran varios valores para k , y se comparan los clusters resultantes en términos de la función objetivo (SSE).

Método del codo:

- Se grafica la función objetivo vs. valores crecientes de k
- Mientras no se alcance el número óptimo de clusters, la función objetivo sigue mejorando sustancialmente,
- Cuando se excede k óptimo, la curva se aplana.



09:56

CLUSTERING k-MEANS: LIMITACIONES

- El algoritmo requiere definir a priori el **número de clusters (k)** ➔ no siempre lo sabremos.

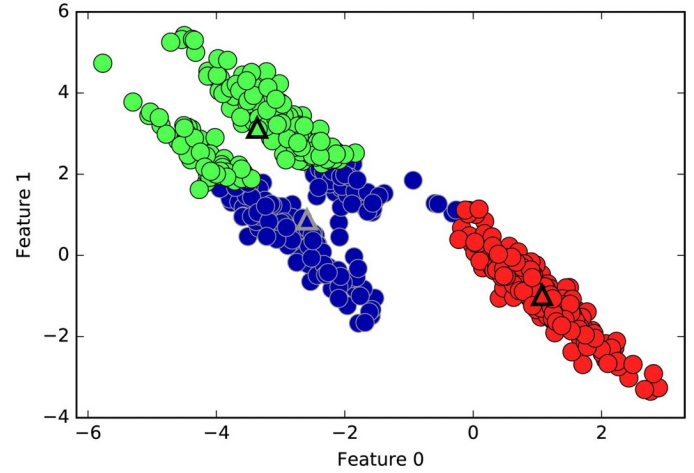
- Los clusters se inicializan aleatoriamente ➔ puede que el algoritmo no converja, o que el resultado sea inestable.

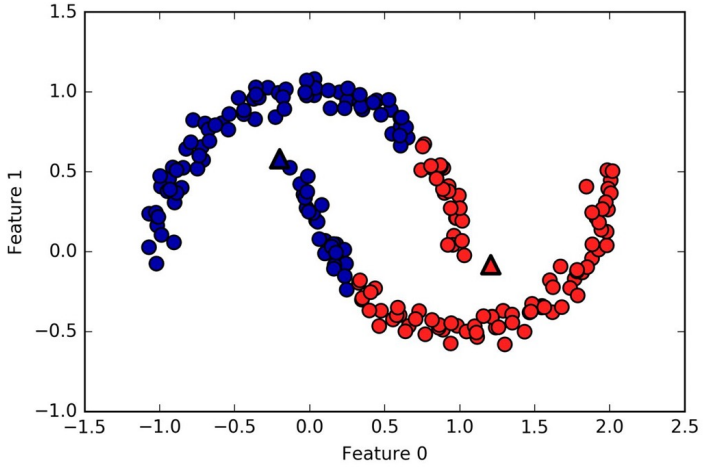
  ➔ `sklearn` realiza 10 iteraciones por defecto

- El algoritmo asume que cada punto pertenece al cluster más cercano, y que todas las direcciones son igualmente importantes

  ➔ Permite identificar clusters esféricos, pero **no funciona bien para datos con geometrías complejas**.

- En sensible a outliers.
- Puede ser lento para grandes cantidades de observaciones, ya que en cada paso hay que calcular todos los pares de distancias.

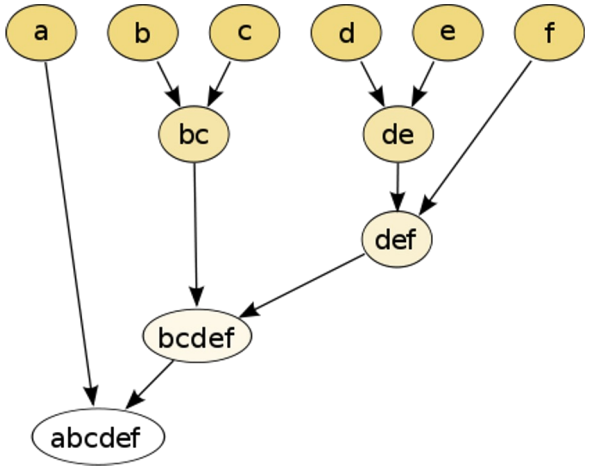




09:56

CLUSTERING AGLOMERATIVO (JERÁRQUICO)

Familia de algoritmos en que se construyen clusters anidados a partir de **fusiones sucesivas** de clusters.

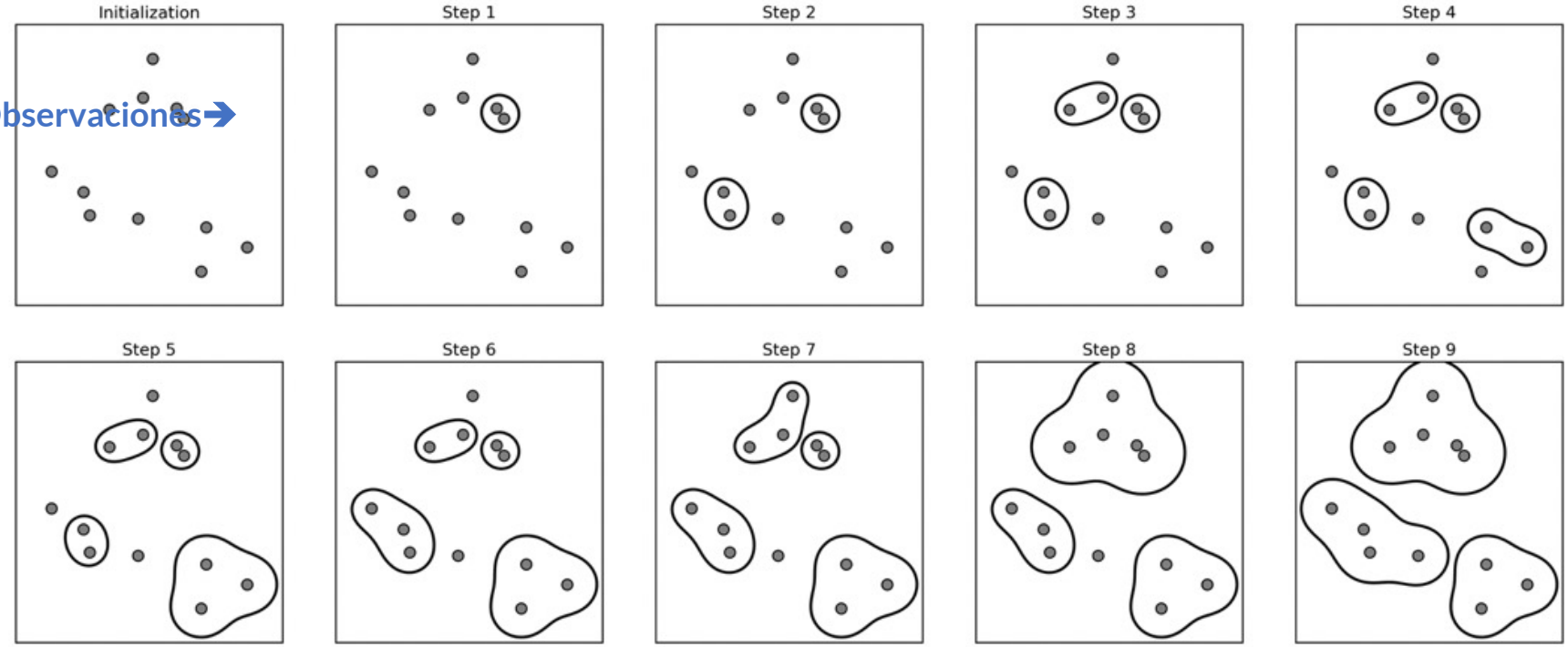


- Primero, todos los puntos son considerados como clusters individuales
- Se calculan las distancias entre todos los pares de centroides (matriz de distancias) de clusters y **se unen los dos más cercanos.**
- Se recalcula el centro del cluster fusionado.
- El proceso de unión de clusters continua hasta llegar al número deseado de clusters.

CLUSTERING AGLOMERATIVO

Clusters individuales

Observaciones ➔

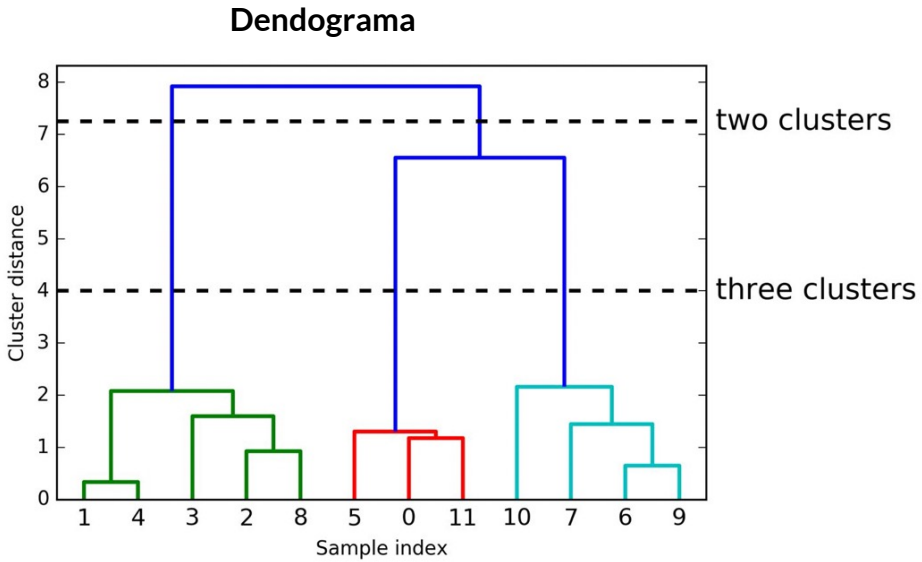


Se calculan las distancias entre todos los pares de centroides de clusters y se fusionan los más cercanos.

09:56

CLUSTERING AGLOMERATIVO (JERÁRQUICO)

- Para datasets multidimensionales, el proceso de fusión de clusters puede visualizarse mediante un dendograma:



- Cada observación (leaf) es inicialmente un cluster.
- Los nodos muestra fusiones de clusters.
- La distancia vertical (branch) representa la distancia entre los clusters que se fusionan.

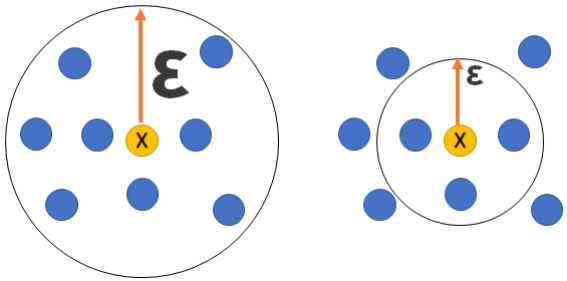
CLUSTERING BASADO EN DENSIDAD

DBSCAN: Density-based cluster algorithm

- No requiere definir el número de clusters a priori.
- Descubre clusters de distintas formas y tamaños para grandes cantidades de data que contiene también ruido y outliers.
- Puede identificar puntos que no pertenecen a ningún cluster.

¿Cómo funciona?

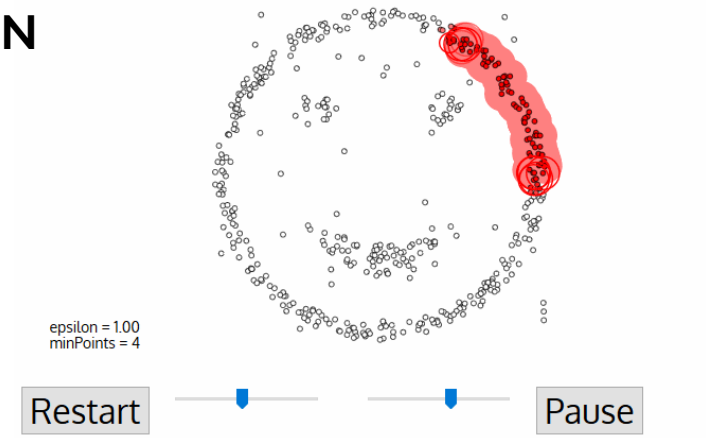
- El principio del algoritmo es que los clusters forman zonas de alta densidad de datos, separados por zonas relativamente vacías.
- Utiliza **2 parámetros**:
  - `min_samples`: el mínimo número de puntos aglomerados para que una región sea considerada "densa"
  - `eps (ε)`: una medida de distancia para ubicar los puntos en el vecindario de otro punto. La medida de distancia **depende de las unidades y escalas de los features** a utilizar en el clustering.



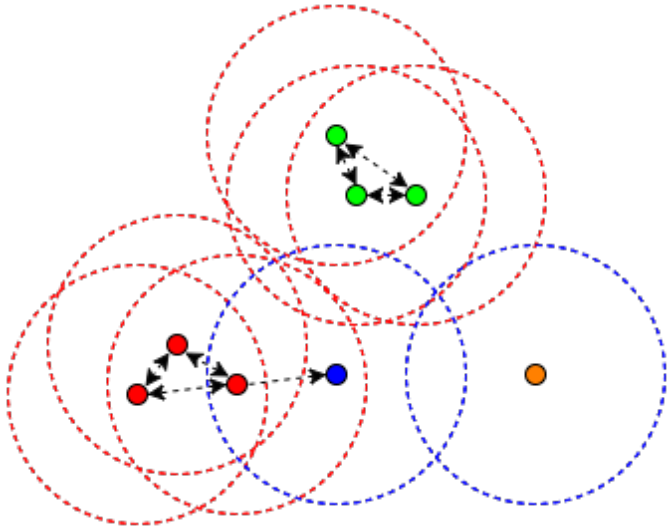
09:56

CLUSTERING BASADO EN DENSIDAD: DBSCAN

- Se elige un punto arbitrario.
- Si hay al menos **`min_samples`** dentro de un radio **`eps`** del punto:
  - El punto pasa a ser el centro de un cluster ➔ **core point**.
  - Todos los vecinos a **`d<eps`** del core point, son asignados al cluster.
- El cluster se expande recursivamente haciendo el cálculo para cada punto vecino.



- Una vez que no hay más puntos a distancia **`d<eps`** del cluster, se pasa a otro punto que no ha sido visitado antes.
- Al final hay 3 tipos de puntos:
  - **Core points:** puntos para los cuales hay al menos min_samples a distancia eps (incluyéndose)
  - **Border points:** puntos al alcance de un core point, pero que no tienen **`min_samples`** en su vecindad.
  - **Noise:** puntos que no están al alcance de ningún core point.



CLUSTERING BASADO EN DENSIDAD: DBSCAN

Para distintas realizaciones de DBSCAN sobre un mismo dataset:

- Puntos clasificados como “core” y “noise” son siempre los mismos.
- Un punto de borde que está al alcance de dos core points, puede ser asignado a uno u otro cluster.

➔ Pertenencia de puntos de borde depende del orden en que se visiten los puntos (no es muy relevante, ya que en general hay pocos border points)