Clustering

Clustering

Los algoritmos de clustering agrupan automáticamente (sin supervisión) grupos de elementos en “clusters”.

El objetivo de un algoritmo de clustering es que cada cluster sea internamente coherente pero diferente a los demás clusters.

Los elementos pertenecientes a un mismo cluster deben ser muy similares y al mismo tiempo muy distintos a los elementos en los demás clusters.

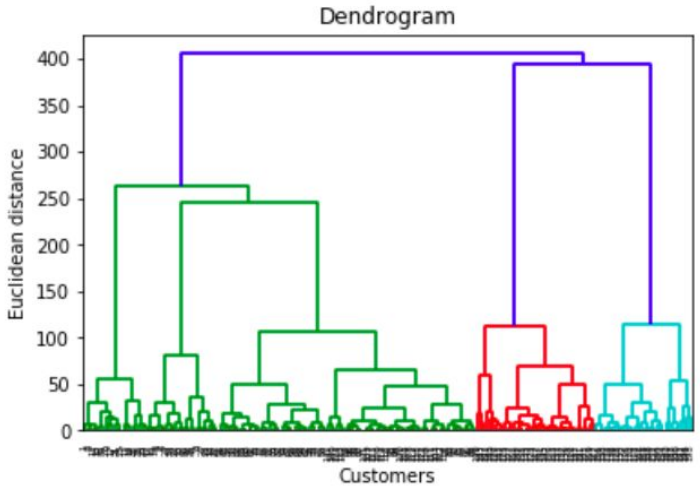
Clustering jerárquico

- Suele generar soluciones muy buenas
- No escala
- Bonus: da una jerarquía entre clusters

Clustering Jerárquico

1. Partimos de que cada punto es un cluster
2. En cada paso unimos los dos **clusters más cercanos**
3. Repetir hasta que haya un único cluster, u obtengamos la cantidad que queríamos.

Anotando la distancias a las cuales unimos los clusters, podemos armar un **dendrograma**.

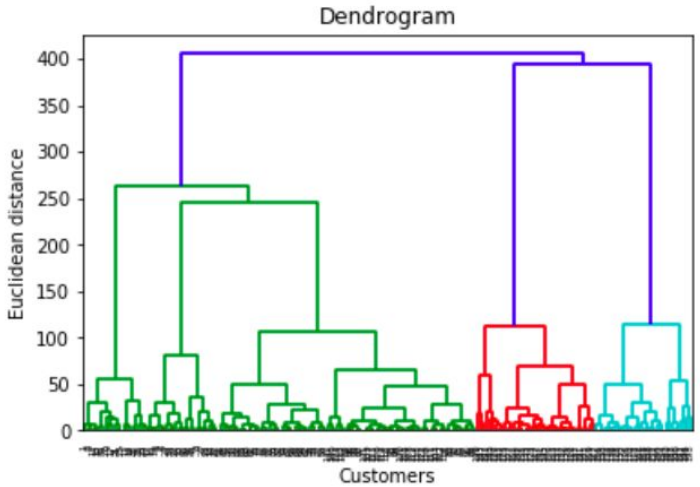


Clustering Jerárquico

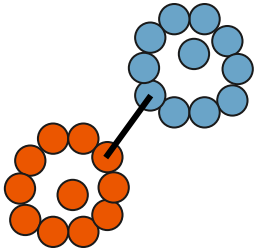
1. Partimos de que cada punto es un cluster
2. En cada paso unimos los dos **clusters más cercanos**
3. Repetir hasta que haya un único cluster, u obtengamos la cantidad que queríamos.

Anotando la distancias a las cuales unimos los clusters, podemos armar un **dendrograma**.

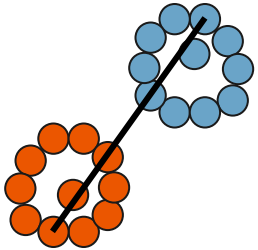
¿cómo definimos esto?



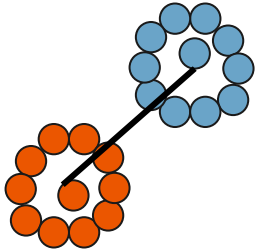
Estrategias para unir clusters



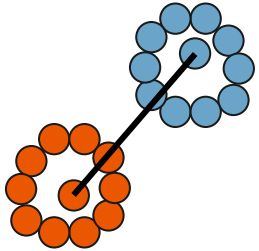
single-linkage



complete-linkage

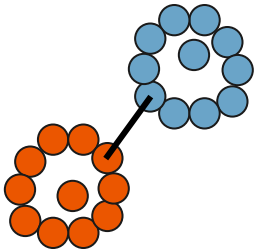


average-link



median-link (?)

Optimización con LSH para single-linkage



single-linkage

- Es una aproximación
- Construimos la estructura de LSH (r, b, etc)
  - En cada bucket habrá como máximo *un* punto de cada cluster
- Si dos puntos en un bucket tienen distancia menor a *t*, unimos los clusters

k-Means

- Dado un set de puntos en R^d

- Particionar los puntos en K sets (K<d)

- De forma tal de minimizar la sumatoria de las distancias de cada punto al centro del cluster

$$\sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \left\| \mathbf{x} - \boldsymbol{\mu}_i \right\|$$

Complejidad del problema

Es NP-Hard en R^d, incluso para k=2

Es NP-Hard en R^2, para cualquier k

Estas son malas noticias!

Afortunadamente existen heurísticas muy buenas que son las que se conocen como k-means.

Algoritmo

1. Elegir *k* puntos, uno para cada cluster
    a. Inicialmente, los centroides de cada cluster son esos puntos.
2. Asignarle a cada punto el cluster con centroide más cercano
3. Re-calcular los centroides como el promedio de los puntos
4. Volver a 2

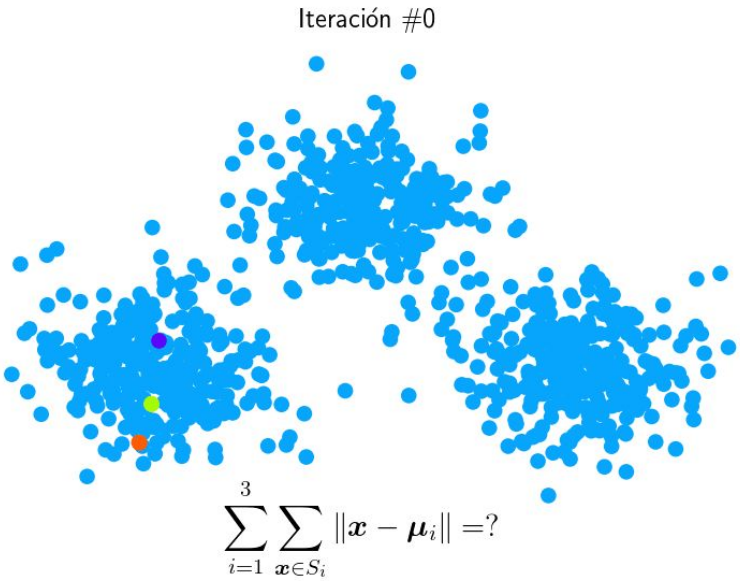
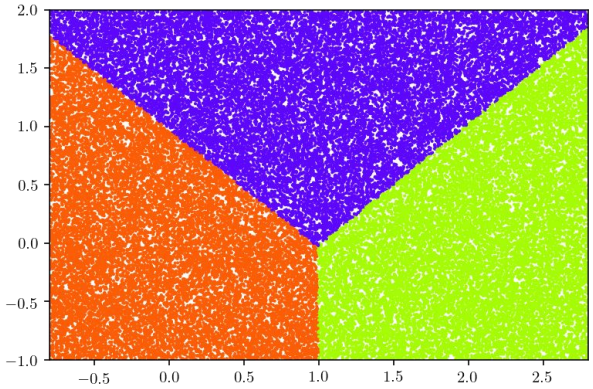


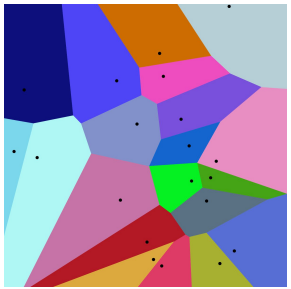
Diagrama de Voronoi

La decisión de a dónde agrupamos un punto está dada por cuál centroide está más cercano.

¿Qué pasa en el límite?

Esto nos muestra que k-means separa los clusters con rectas.





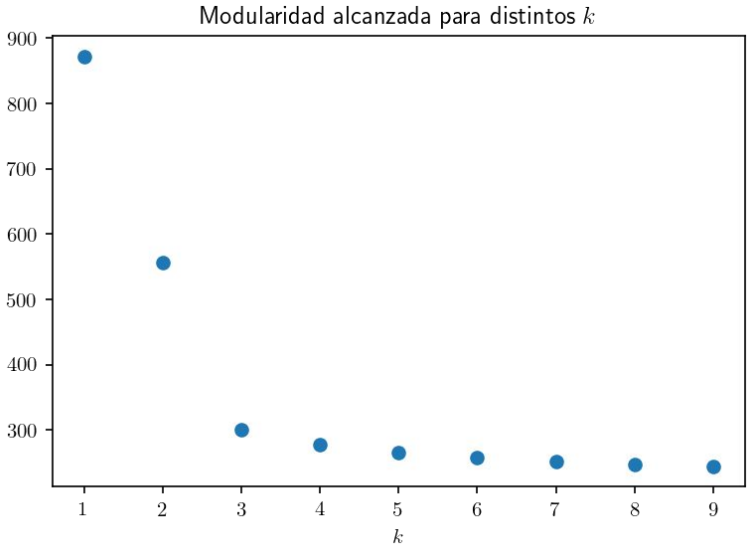
¿De dónde sale K?

Es necesario probar valores y analizar alguna métrica de calidad de clusters.

Por ejemplo, promedio de la sumatoria de distancias dentro de cada cluster.

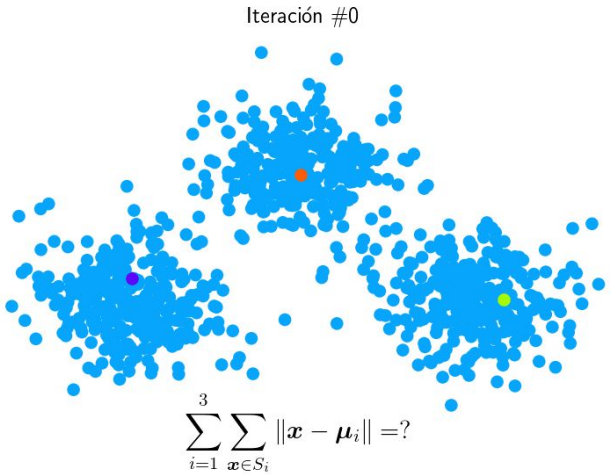
Método del codo

$$\sum_{i=1}^{k} \sum_{\boldsymbol{x} \in S_i} \|\boldsymbol{x} - \boldsymbol{\mu}_i\|$$



Problema de la inicialización

- K-means es bastante sensible a los puntos iniciales.
  - Tal vez llega a la solución óptima pero tarda mucho más de lo esperado
- ¿Podemos elegir más inteligentemente los centroides iniciales?
  - No
  - Bueno sí



Problema de la inicialización: k-means++

1. Elegir un punto al azar como centroide
2. Calcular la distancia de cada punto contra los centroides y quedarse con la mínima.
3. Calcular la probabilidad de cada punto, como la distancia dividida por la suma de todas las distancias de los puntos.
4. Elegir un nuevo punto al azar, utilizando la probabilidad calculada.
5. Repetir hasta tener *k* centroides

k-Means online

- O(k) en memoria
- Puede procesar puntos ilimitados => streams!
- Algoritmo:
    - 1. Inicializar k centroides al azar
    - 2. Para cada punto que viene:
        - Asignar al centroide más cercano
        - Aumentar el contador del cluster
        - Mover proporcionalmente el centroide

k-Means online: algoritmo

```
contador[0..k-1] = 1
centroide[0..k-1] = k-puntos iniciales

cuando llega un punto x:
     i <- índice_cluster_más_cerca(x)
     # .. registrar/asignar x a ese cluster (si es que nos interesa hacerlo)
     contador[i] += 1
     centroide[i] = centroide[i] + 1/contador[i] (x - centroide[i])
```

k-Means como algoritmo de compresión

Cuanto más detalle tiene una imagen, es más difícil de comprimir. Si pudiéramos reducir el detalle podríamos obtener una compresión mejor.

Podemos reducir el detalle utilizando menos colores.

Idea de “puntos representativos”.

k-Means trick

K-means puede usarse para reducir las dimensiones de un set de puntos.

- Correr k-means con algún k.

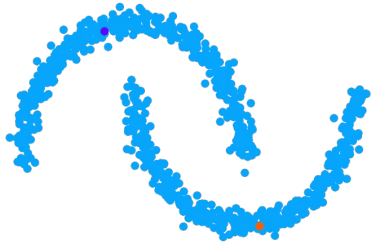
- Y ahora computar la distancia de cada punto a cada uno de los k-centroides.

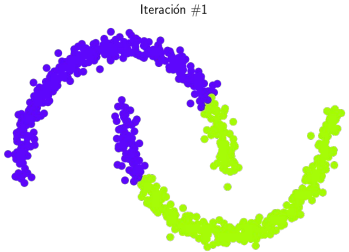
- El resultado es que cada punto queda convertido a k dimensiones!

Muchos problemas de clasificación pueden resolverse aplicando k-means trick y luego un simple clasificador lineal (!).

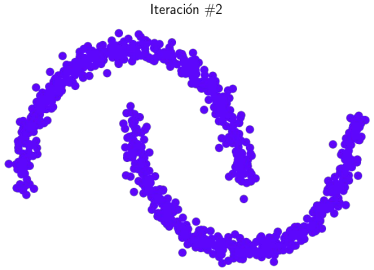
Clustering espectral

- Esa palabra nos recuerda a..
- Laplacian eigenmaps!
  - Nos servía para reducir dimensiones de cosas no-lineales
  - “Desenrollaba” las superficies hiper-dimensionales (*manifolds*)
- Clustering espectral se trata de aplicar k-Means sobre Laplacian Eigenmaps





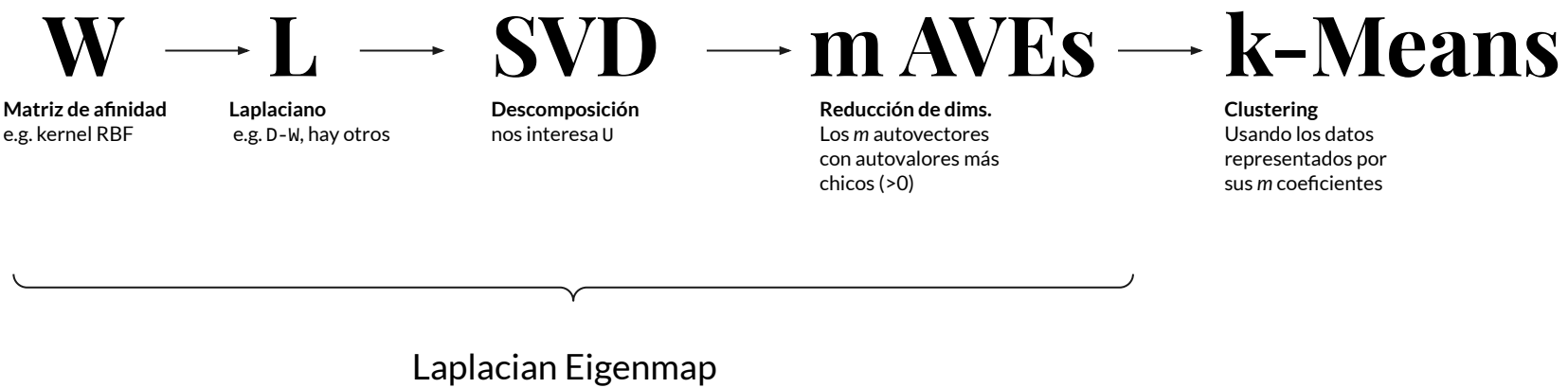
$$\sum_{i=1}^{3} \sum_{\boldsymbol{x} \in S_i} \|\boldsymbol{x} - \boldsymbol{\mu}_i\| = 548.27$$



$$\sum_{i=1}^{3} \sum_{\boldsymbol{x} \in S_i} \|\boldsymbol{x} - \boldsymbol{\mu}_i\| = 832.69$$

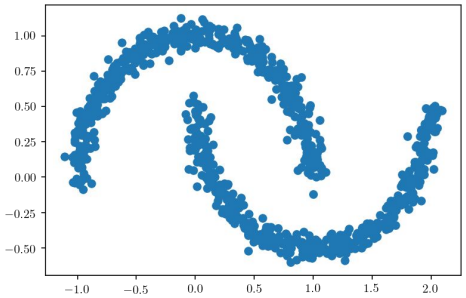
Clustering espectral

Los pasos:

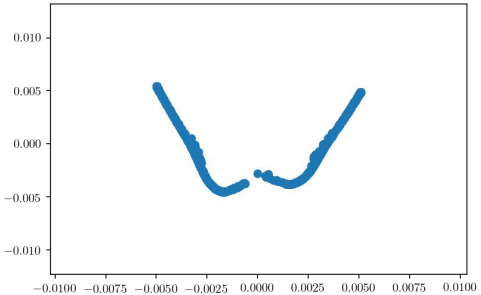


Clustering espectral

Visualicemos:

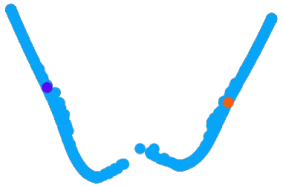


Laplacian Eigenmap
W=nearest neighbors
m=2

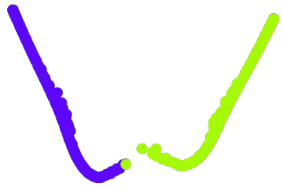


Clustering espectral

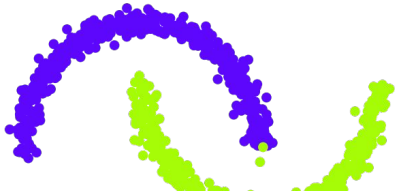
Visualicemos:



k-Means
Una iteración, k=2



$$\sum_{i=1}^{3} \sum_{\boldsymbol{x} \in S_i} \|\boldsymbol{x} - \boldsymbol{\mu}_i\| = 3.07$$



$$\sum_{i=1}^{3} \sum_{\boldsymbol{x} \in S_i} \|\boldsymbol{x} - \boldsymbol{\mu}_i\| = 654.73$$

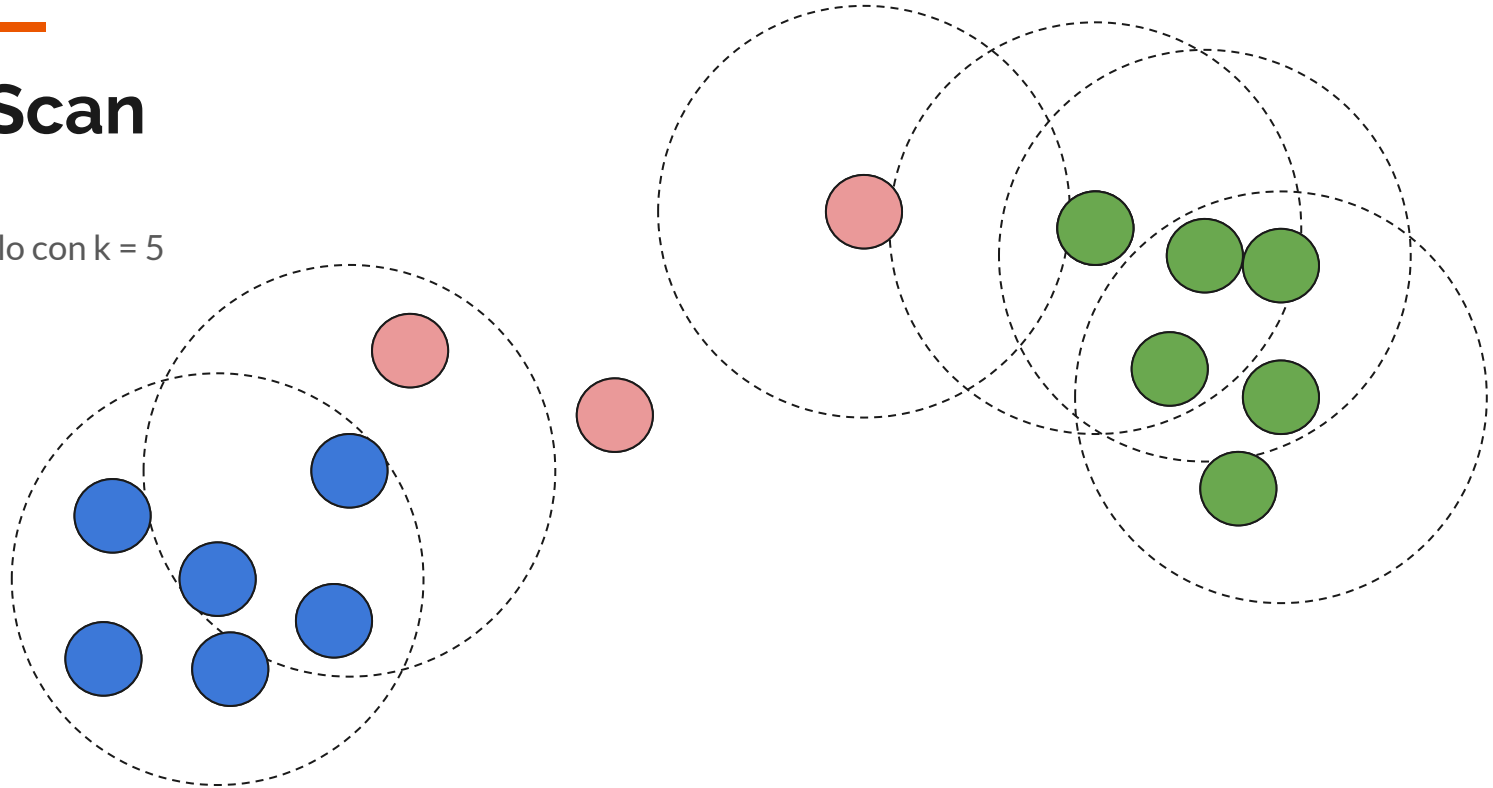
DBScan

- Density Based Spatial Clustering of Applications with Noise
- Concepto distinto de cluster -- grupo **denso**
- Permite tener puntos que no son parte de algún cluster (**noise**)
- Tenemos que definir **densidad**
- Un punto está en una región densa ⇔ hay más de **k** puntos con distancia menor a **ε** del punto (incluimos al punto)

⇒ tenemos dos hiperparámetros: k y ε

DBScan

- Ejemplo con k = 5



DBScan

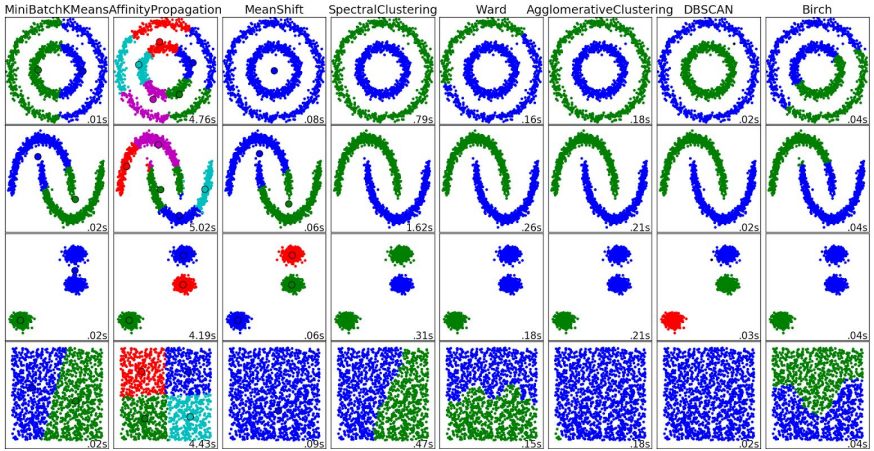
Input: X, ϵ, k
Output: Output

$C \leftarrow 0$;
for $x \in X$ **do**
  if cluster$(x) \neq$ *undefined* **then**
    continue;
  $V \leftarrow$ Vecinos(X, x, ϵ);
  if $|V| < k - 1$ **then**
    cluster$(x) \leftarrow$ outlier;
    continue;
  $C \leftarrow C + 1$;
  cluster$(x) \leftarrow C$;
  $Q \leftarrow cola()$;
  $Q.encolar(\{v \in V\})$;
  while $v \leftarrow Q.desencolar()$ **do**
    if $cluster(v) =$ outlier **then**
      cluster$(v) \leftarrow C$;
    if cluster$(v) \neq$ *undefined* **then**
      continue;
    cluster$(v) \leftarrow C$;
    $V \leftarrow$ Vecinos(X, v, ϵ);
    if $k - 1 \leq |V|$ **then**
      $Q.encolar(\{v' \in V\})$;

BFS!

DBScan

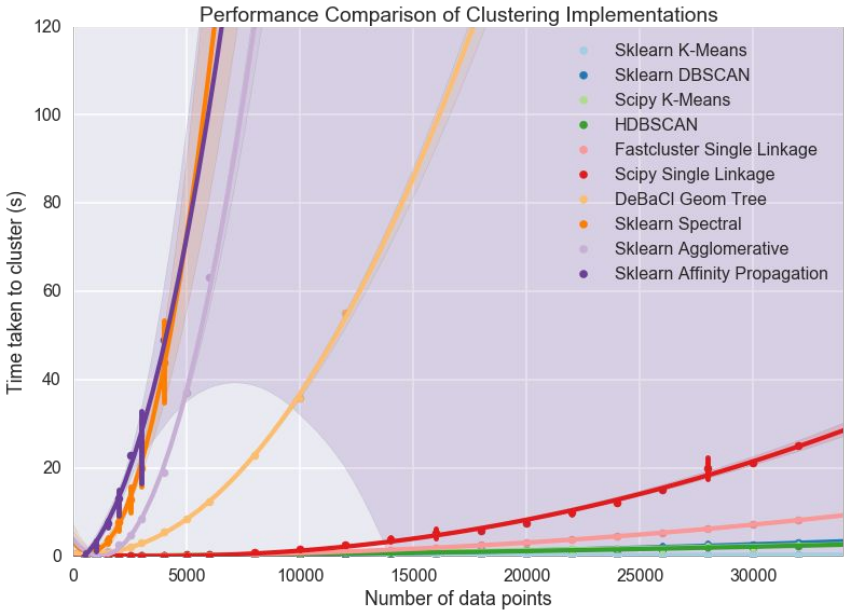
· Puede encontrar grupos de cualquier forma siempre y cuando se mantenga la densidad



DBScan

P: ¿Pero por qué usaríamos DBScan si Clustering Espectral ya nos permite encontrar formas no lineales?

R: Tiempo!



DBScan

- Problema fundamental de DBScan: es bastante sensible a los parámetros
- Es particularmente difícil estimar ε
- Como la densidad se define a priori, no maneja bien clusters con densidades muy distintas.
- Como es ‘plano’, no puede encontrar clusters más densos dentro de otros más grandes.

HDBScan

- Basado en DBScan y en clustering jerárquico
- Soluciona el problema de densidades diferentes entre los clusters
- Primero analicemos DBScan y luego veamos qué modifica esta versión

HDBScan

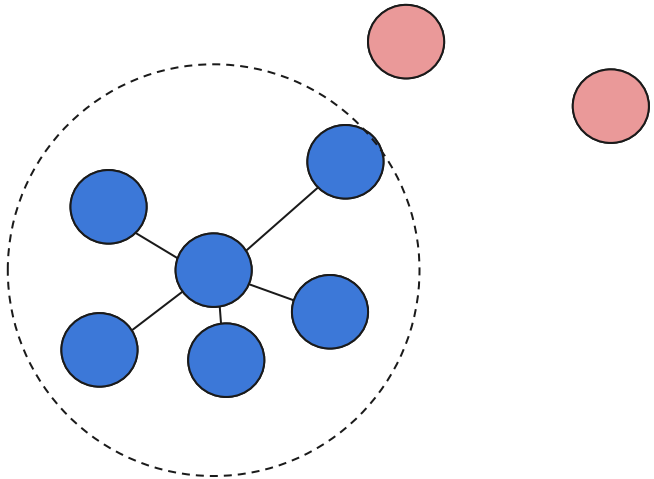
Podemos interpretar a DBSCAN como que arma un grafo con aristas:

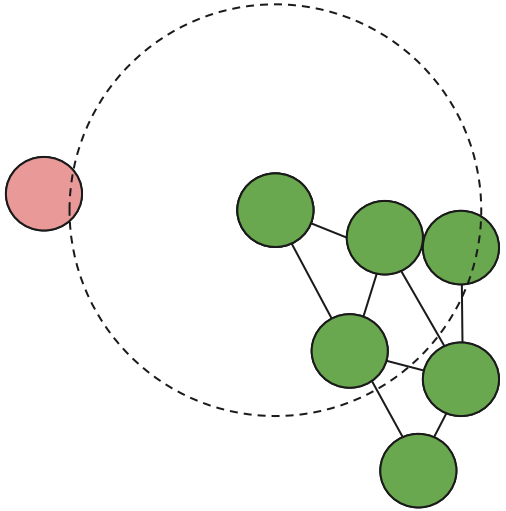
$$w_{ij} = \begin{cases} 1 \ si \ \ (|V(x_i, \epsilon)| \geq k - 1 \vee |V(x_j, \epsilon)| \geq k - 1) \wedge x_j \in V(x_i, \epsilon) \\ 0 \text{ en otro caso} \end{cases}$$

y los clusters son las componentes conexas.

DBScan

- Ejemplo con k = 5





HDBScan

- Lo que hace HDBScan es evitar el parámetro ε que no permite analizar densidades distintas.
- Para analizar densidades distintas, es necesario empujar los outliers fuera de los puntos densos.

HDBScan

Grafo → **Arbol** → **Jerarquías** → **Simplificar** → **Aplanar**

Completo y pesado
con distancia MRD

Tendido mínimo
también llamado MST

Dendrograma

Eliminación de ruido
Distinguimos al *núcleo* del cluster del r*uido*

(Opcional)
Si no queremos la jerarquía, podemos obtener los clusters más conservados

Hiperparámetros:

- K para $core_k$
- Tamaño mínimo de cluster

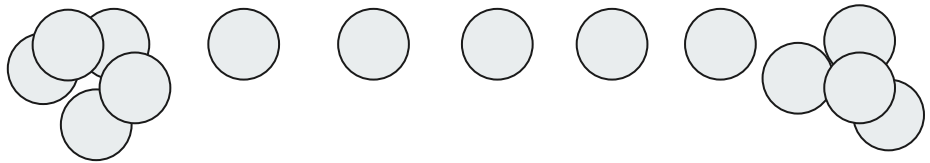
HDBScan

Definimos una distancia nueva:

$$\mathrm{mrd}(a,b) = \max\{\mathrm{core_k}(a), \mathrm{core_k}(b), d(a,b)\}$$

donde $\mathrm{core}_k(a)$ es la distancia al k-vecino más cercano.

K: vecinos



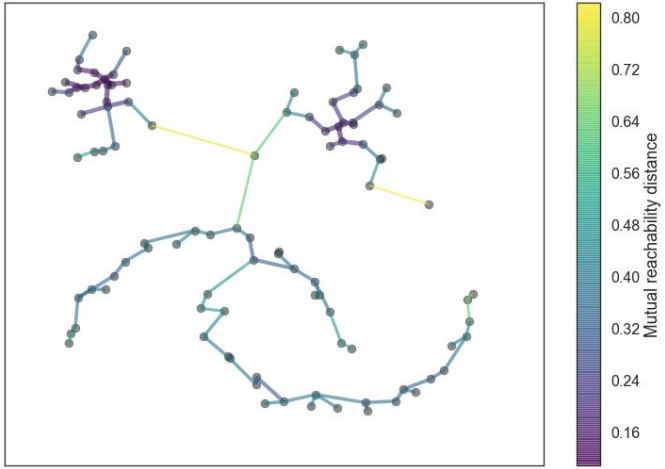
HDBScan

Armamos el grafo completo:

$$w_{ij} = \mathrm{mrd}(x_i, x_j)$$

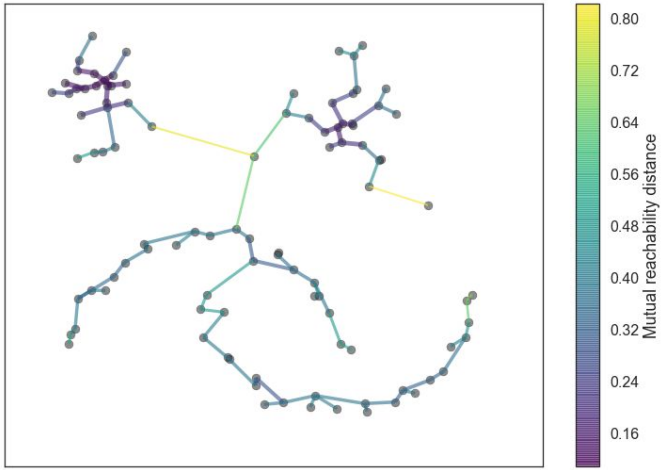
HDBScan

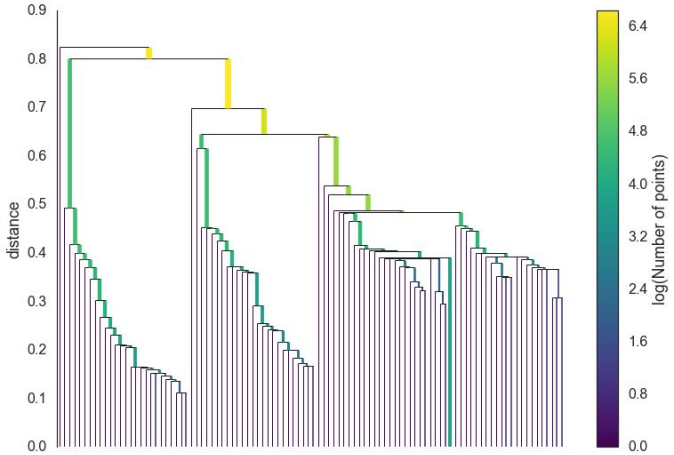
Armamos el árbol de tendido mínimo.



HDBScan

Podemos armar una jerarquía:



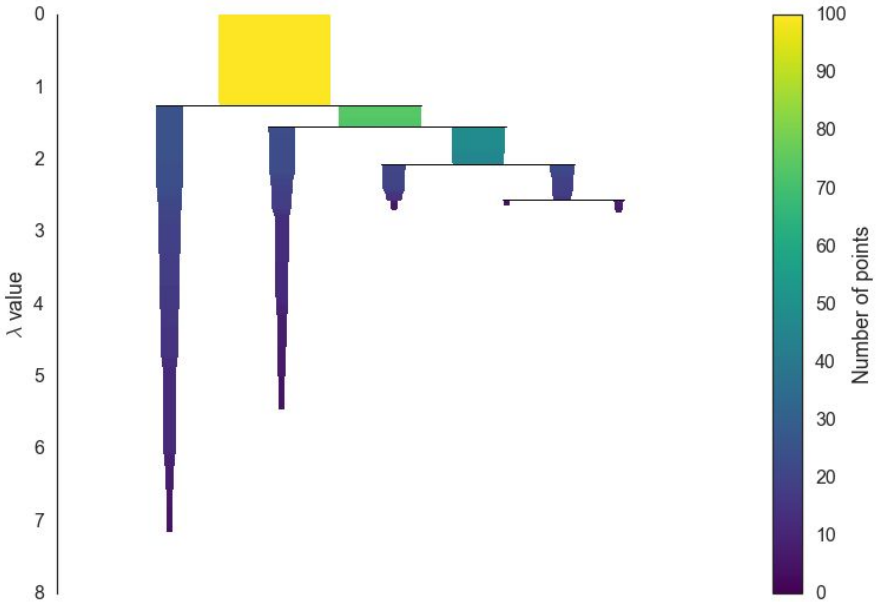


HDBScan

Simplificamos teniendo en cuenta un tamaño mínimo de cluster.

En cada split pueden pasar tres cosas:

- Los dos sub componentes son muy chicos (desaparecen)
- Los dos sub componentes perduran
- Un sub componente perdura y el otro desaparece



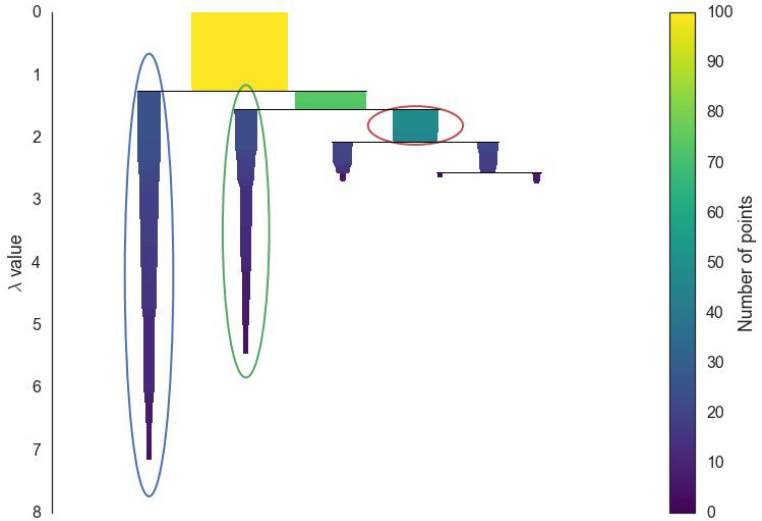
HDBScan

Si queremos un clustering plano, analicemos la estabilidad de los clusters.

Definimos λ = 1/distancia:

- Cada cluster tiene λ_{nace} y λ_{muere}
- Cada punto tiene λ_p cuando se cae

$$\text{Estabilidad}(\text{cluster}) = \sum_{p \in \text{cluster}} (\lambda_p - \lambda_{nace}(\text{cluster}))$$

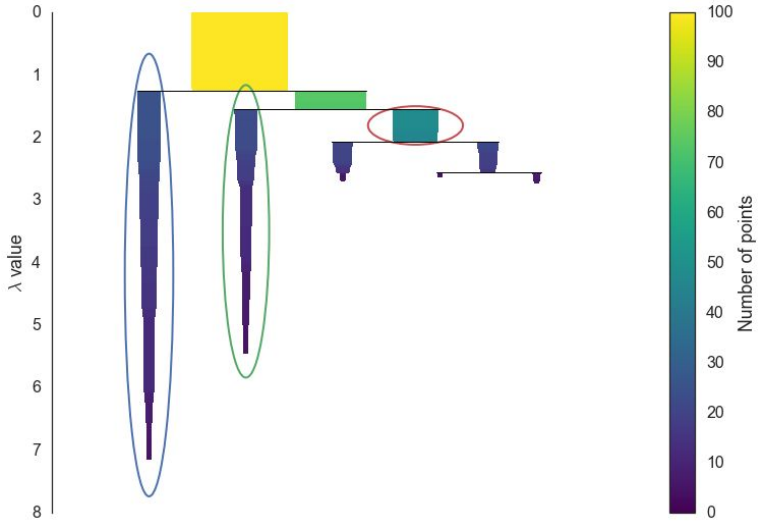


HDBScan

$$\text{Estabilidad}(\text{cluster}) = \sum_{p \in \text{cluster}} (\lambda_p - \lambda_{nace}(\text{cluster}))$$

Partimos desde las hojas y decimos que son los clusters elegidos.

El padre es elegido si y solo si su estabilidad es mayor a la de sus hijos.



HDBScan - Performance

Links a extras

Notebook con animaciones:
https://drive.google.com/file/d/1mRzyuX719dSUn6MwkzvAjREX-VuW5Xsf/view?usp=sharing

Notebook con comparación de métodos:
https://nbviewer.jupyter.org/github/lmcinnes/hdbscan/blob/master/notebooks/Comparing%20Clustering%20Algorithms.ipynb

Documentación de HDBScan: https://hdbscan.readthedocs.io/en/latest/index.html

Paper de LSH para Clustering Jerárquico:
http://web-ext.u-aizu.ac.jp/labs/is-se/conference_proceedings/iwait-15/33.pdf