Modelos no supervisados

PID_00284572

Jordi Gironés Roig

Tiempo mínimo de dedicación recomendado: 2 horas

Universitat Oberta de Catalunya

Jordi Gironés Roig

Licenciado en Matemáticas por la Universidad Autónoma de Barcelona y diplomado en Empresariales por la Universitat Oberta de Catalunya. Ha desarrollado la mayor parte de su carrera profesional en torno de la solución SAP, en sus vertientes operativas con S4-HANA y estratégica con SAP-BI. Actualmente trabaja en la industria químico-farmacéutica como responsable de aplicaciones corporativas para Esteve Pharmaceuticals y colabora con la UOC en asignaturas relacionadas con la analítica de datos.

El encargo y la creación de este recurso de aprendizaje UOC han sido coordinados por el profesor: Julià Minguillón Alfonso

Primera edición: septiembre 2021

Av. Tibidabo, 39-43, 08035 Barcelona
Autoría: Jordi Gironés Roig
Producción: FUOC

Índice general

Introducción

La clasificación no supervisada persigue la obtención de un modelo válido para clasificar objetos a partir de la similitud de sus características. Más formalmente podríamos decirlo del siguiente modo:

> A partir de un conjunto de objetos descritos por un vector de características y a partir de una métrica que nos defina el concepto de similitud entre objetos, se construye un modelo o regla general que nos va a permitir clasificar todos los objetos.

No se trata de modelos predictivos, sino de modelos de descubrimiento de patrones. Existen dos grandes familias de algoritmos de clasificación no supervisada:

1) Algoritmos jerárquicos: construyen nodos de forma jerárquica, unos a partir de otros. La representación de los resultados se hace habitualmente mediante dendogramas. Estos se dividen en dos subcategorías:

- Aglomerativos o *bottom up*. Esta aproximación parte del supuesto que cada objeto es un nodo o clúster y, a medida que evolucionan los pasos del algoritmo, los nodos se van agrupando hasta conseguir un número de nodos aceptable.

- Divisivos o *top down*. Es la aproximación opuesta, es decir, parte del supuesto que existe un único nodo y, a medida que avanza el algoritmo, este nodo se va subdividiendo en nuevos nodos, y así sucesivamente.

  Ejemplos de algoritmos jerárquicos pueden ser el método del mínimo o *single linkage* y el método del máximo o *complete linkage*.

2) Algoritmos particionales: también llamados algoritmos de optimización, obtienen los nodos a partir de la optimización de una función adecuada para el propósito del estudio. Esta función suele estar relacionada con la métrica seleccionada para establecer el concepto de similitud entre objetos.

1. Conceptos preliminares

Como hemos visto, muchos modelos no supervisados basan su lógica en el concepto de *similitud* o *distancia*, de modo que merece la pena dedicar tiempo a su comprensión, ya que nos ayudará a entender el mecanismo de algoritmos como el *k-means* y el *clustering*, entre otros.

1.1. Distancia o similitud

Cuando hablamos de *distancia*, en realidad estamos refiriéndonos a una forma de cuantificar cómo de similares son dos objetos, dos variables o dos puntos. Planteado de esta forma, el concepto de *distancia* es muy abstracto y etéreo; por este motivo, los científicos quieren ponerle algunos límites o condiciones. Para un conjunto de elementos X, se considera distancia cualquier función

$$(XxX) \rightarrow R \tag{1}$$

que cumpla las tres condiciones siguientes:

- La no negatividad: la distancia entre dos puntos siempre debe ser positiva.

$$d(a,b) \geq 0 \quad \forall a,b \in X \tag{2}$$

- La simetría: la distancia entre un punto a y un punto b debe ser la misma que la distancia entre el punto b y el punto a.

$$d(a,b) = d(b,a) \quad \forall a,b \in X \tag{3}$$

- La desigualdad triangular: la distancia debe coincidir con la idea intuitiva que tenemos de ella en cuanto a que es el camino más corto entre dos puntos.

$$d(a,b) \leq d(a,c) + d(c,b) \quad \forall a,b,c \in X \tag{4}$$

Con estas tres condiciones, vamos a centrarnos en tres definiciones de distancia muy peculiares: la euclidiana o clásica, la estadística o de Gauss y la que propuso Mahalanobis.

1.1.1. Euclides

La distancia euclidiana coincide plenamente con lo que nuestra intuición entiende por *distancia*. Podríamos llamarla *distancia ordinaria*.

Su expresión matemática para un espacio, por ejemplo, de dos dimensiones y para los puntos $A(x_1,y_1)$, $B(x_2,y_2)$, sería:

$$d(A,B) = d((x_1,x_2),(y_1,y_2)) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (5)$$

Utilizar esta distancia en un proceso de segmentación presenta un inconveniente. **No tiene en cuenta las escalas** en las que pueden estar expresadas las variables X e Y.

Ejemplo

Si la variable X representa la edad de dos ratones, entonces tomará valores de entre 0 y 3. Si la variable Y representa la longitud de la cola del ratón en milímetros, puede tomar valores entre 90 y 120.

Parece claro que cuando queramos comparar y calculemos la distancia entre dos individuos, pesará injustamente mucho más la longitud de la cola que la edad.

1.1.2. Gauss

Para superar la distorsión provocada por las diferentes unidades de medida usadas en las distintas variables estudiadas tenemos la distancia estadística, que simplemente normaliza las variables para situarlas a todas bajo la misma escala.

Distancia de Gauss

Debemos este avance al brillante matemático alemán Carl Friedrich Gauss (1777 – 1855)

Su expresión analítica para un espacio, por ejemplo, de dos dimensiones y para los puntos $A(x_1,y_1)$, $B(x_2,y_2)$, en la que nuestra distribución de puntos en estas dimensiones tuviera una desviación estándar σ:

$$d(A,B) = \sqrt{\left(\frac{x_2 - x_1}{\sigma(X)}\right)^2 + \left(\frac{y_2 - y_1}{\sigma(Y)}\right)^2} \quad (6)$$

Nuevamente, este concepto de distancia tiene problemas. **No tiene en cuenta la correlación** entre las variables, es decir, si nuestras variables fueran totalmente independientes no habría ningún problema. Sin embargo, si tienen algún tipo de correlación, una influye sobre la otra y esta influencia no queda bien reflejada si usamos la distancia estadística.

Ejemplo

Las variables *entrenar* y *rendimiento* están correlacionadas, de modo que más entrenamiento siempre implica más rendimiento, pero esta regla no se cumple infinitamente ya que entrenamiento infinito no implica rendimiento infinito (lo vemos continuamente en el deporte). Si no tenemos en cuenta esta correlación y queremos comparar dos deportistas, podemos llegar a conclusiones erróneas.

1.1.3. Mahalanobis

Prasanta Chandra Mahalanobis (India) en 1936 se dio cuenta de esta carencia y propuso corregir la distorsión provocada por la correlación de las variables mediante la siguiente expresión:

La versión simplificada para un espacio, por ejemplo, de dos dimensiones, con un conjunto de puntos de varianza $\sigma^2(X)$,$\sigma^2(Y)$ y covarianza $\text{cov}(X,Y)$ sería:

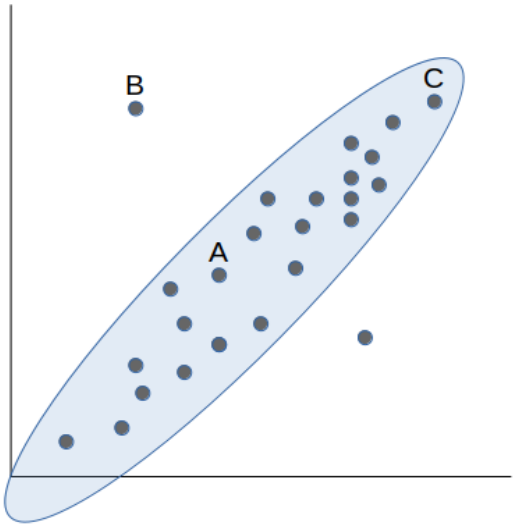
$$\sqrt{(x_1-y_1,x_2-y_2)\begin{bmatrix}\sigma^2(X) & \text{cov}(X,Y)\\ \text{cov}(Y,X) & \sigma^2(Y)\end{bmatrix}^{-1}(x_1-y_1,x_2-y_2)} \qquad (7)$$

La definición de *distancia* propuesta por Mahalanobis responde a la idea intuitiva de que los puntos que se encuentran en una zona densamente poblada deberían considerarse más cercanos entre ellos que con respecto a puntos fuera de esta zona de mayor densidad.

Si tomamos como ejemplo la figura 1, vemos como una versión clásica de distancia nos diría que el punto *A* está más cerca de *B* que de *C*. Sin embargo, Mahalanobis, entendiendo qué puntos en zona de densidad son más similares, nos dirá que el punto *A* está más cerca de *C* que de *B*.

Que Mahalanobis generalice a Gauss y Gauss a Euclides forma parte de lo que se ha llamado *la belleza de las matemáticas*. Veámoslo.

Figura 1. Distancia de Mahalanobis



Si en nuestra distribución de puntos, las dos dimensiones son totalmente independientes su covarianza será cero, de forma que la distancia de Mahalanobis coincide con la distancia estadística.

$$\sqrt{(x_2 - x_1, y_2 - y_1)\begin{bmatrix}\sigma^2(X) & 0\\ 0 & \sigma^2(Y)\end{bmatrix}^{-1}(x_2 - x_1, y_2 - y_1)} = \tag{8}$$

$$= \sqrt{\left(\frac{x_2 - x_1}{\sigma(X)}\right)^2 + \left(\frac{y_2 - y_1}{\sigma(Y)}\right)^2} \tag{9}$$

Si en la misma distribución con variables independientes tenemos que su distribución está normalizada, es decir, que las dos variables tienen la misma escala, entonces su varianza será 1, de forma que la distancia estadística coincide con la distancia de Euclides.

$$\sqrt{(x_2 - x_1, y_2 - y_1)\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}^{-1}(x_2 - x_1, y_2 - y_1)} = \tag{10}$$

$$= \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{11}$$

2. *Clustering* y segmentación

Clustering y *segmentation*, traducidos como *agrupamiento* y *segmentación*, constituyen el ámbito de conocimiento correspondiente a las técnicas no supervisadas, ya que no tienen como objetivo predecir una etiqueta que marca cada observación del juego de datos.

Su objetivo es describir el juego de datos encontrando patrones a partir de la identificación de grupos similares

De este modo, los conceptos de distancia y similitud serán claves para entender este tipo de algoritmos.

2.1. Agrupación jerárquica

Dentro de esta familia, distinguiremos dos tipos de algoritmo: los aglomerativos y los divisivos.

2.1.1. Algoritmos de tipo aglomerativo

Los algoritmos de agrupación jerárquica son de **tipo aglomerativo** cuando, partiendo de una fragmentación completa de los datos, estos se van fusionando hasta conseguir una situación contraria, es decir, todos los datos se unen en un solo grupo. En este caso hablaremos de ***clustering*** o agrupamiento.

Para construir grupos, necesitan de un concepto de distancia entre objetos y de un criterio de enlace para establecer la pertenencia a un grupo u otro. Algunos de los criterios más utilizados para medir la distancia entre dos grupos A y B son los siguientes:

- Enlace simple o *simple linkage*. Tomaremos como criterio la distancia mínima entre elementos de los grupos:

$$min\{d(x,y) \quad |x \in A, y \in B\} \tag{12}$$

Puede ser apropiado para encontrar grupos de forma no elíptica, pero es muy sensible al ruido en los datos y puede llegar a provocar el efecto cadena. Este consiste en el hecho de que puede llegar a forzar la unión de dos grupos, que *a priori* deberían permanecer bien diferenciados, por el hecho de que estos compartan algún elemento muy próximo.

- Enlace completo o *complete linkage*. Tomaremos como criterio la distancia máxima entre elementos de los grupos:

$$max\{d(x,y) \quad |x \in A, y \in B\} \tag{13}$$

No produce el efecto cadena, pero es sensible a los valores *outliers*. Sin embargo, suele dar mejores resultados que el criterio simple.

- Enlace medio o *average linkage*. Tomaremos como criterio la distancia media entre elementos de los grupos:

$$\frac{1}{|A||B|}\sum_{x\in A}\sum_{y\in B} d(x,y) \tag{14}$$

Se trata de un criterio que intenta mitigar los inconvenientes de los dos anteriores sin acabar de resolverlos por completo.

- Enlace centroide o *centroid linkage*. La distancia entre dos grupos será la distancia entre sus dos centroides. Presenta la ventaja de que su coste computacional es muy inferior al de los criterios anteriores, de modo que está indicado para juegos de datos de gran volumen.

Para construir un dendograma aglomerativo, deberemos inicialmente establecer con qué métrica desearemos trabajar (distancia euclidiana, de Gauss, de Mahalanobis...) y qué criterio de enlace de grupos o segmentos utilizaremos (*simple linkage*, *complete linkage*, *average linkage*, *centroid linkage*...).

El siguiente paso será considerar cada observación del juego de datos como un grupo o segmento en sí mismo, y a partir de aquí empezaremos a calcular distancias entre grupos. En este punto entraremos en un proceso iterativo en el que en cada repetición fusionaremos los grupos más cercanos.

2.1.2. Algoritmos de tipo divisivo

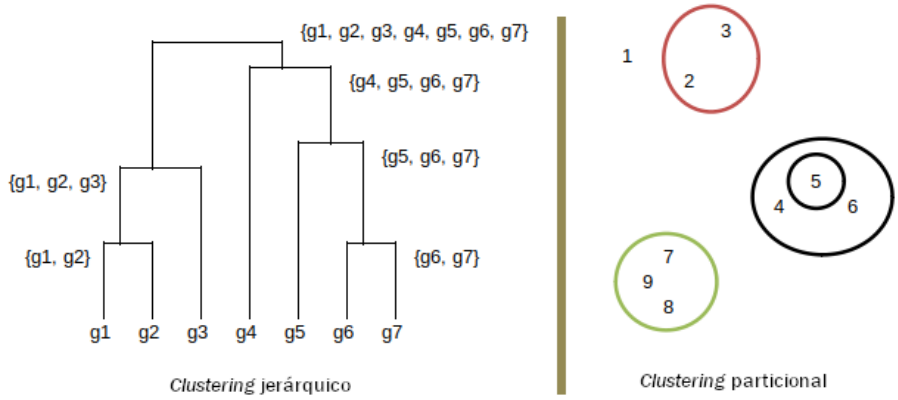
Diremos que son de **tipo divisivo** cuando, partiendo de un grupo que contiene todos los datos, se procede a una división progresiva hasta conseguir tener un grupo para cada observación. En este caso hablaremos de **segmentación**.

2.2. Algoritmos particionales

Los algoritmos particionales o no jerárquicos reciben este nombre porque los segmentos que acaban produciendo no responden a ningún tipo de organización jerárquica. En esta categoría de algoritmos encontraríamos al ***k-means***.

En la figura 2 podemos distinguir de una forma visual la diferencia entre el *clustering* jerárquico y el particional.

Figura 2. Tipos de *clustering*



2.2.1. *k-means*

El algoritmo *k-means*, o *k-medias* en español, está considerado como un algoritmo de clasificación no supervisada. Requiere que de antemano se fijen los k grupos que quieren obtenerse.

Supongamos que disponemos de un juego de datos compuesto por n observaciones. Por ejemplo, cada caso podría ser un cliente del que hemos seleccionado m atributos que lo caracterizan.

Llamaremos X a este juego de datos $X = \{x_1,x_2,...,x_n\}$ donde cada x_i podría ser un cliente con m atributos $x_i = \{x_{i1},x_{i2},...,x_{im}\}$ como pueden ser, por ejemplo, ventas, promociones, distancia al centro de distribución logística, etc.

Para clasificar nuestro juego de datos X mediante el algoritmo *k-means*, seguiremos los siguientes pasos:

1) De entre las n observaciones seleccionaremos k, al que llamaremos semillas, y denotaremos por c_j donde $j = 1,...,k$. Cada semilla c_j identificará su clúster C_j.

2) Asignaremos la observación x_i al clúster C_t cuando la distancia entre la observación x_i y la semilla c_t sea la menor entre todas las semillas.

$$d(x_i,c_t) = min\{d(x_i,c_j)\},\ j = 1,...,k \tag{15}$$

3) Calcularemos los nuevos centroides a partir de las medias de los clústeres actuales.

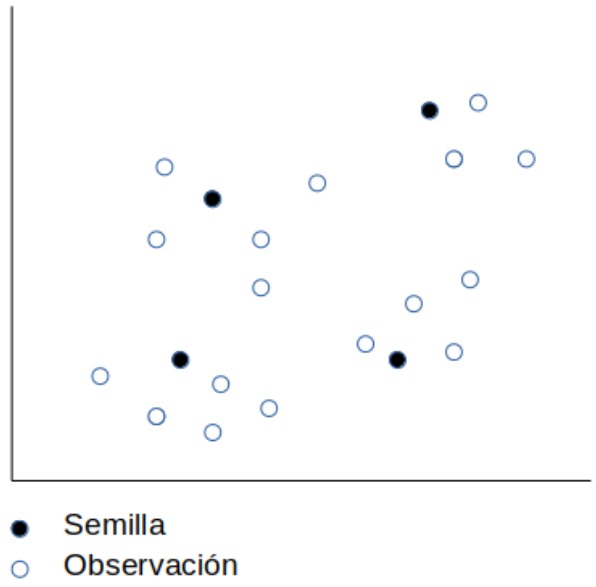
4) Como criterio de parada, calcularemos la mejora que se produciría si asignáramos una observación a un clúster al que no pertenece actualmente. Entendiendo por mejora, por ejemplo, la minimización de la distancia de las distintas observaciones a sus respectivos centros.

5) Haremos el cambio que mayor mejora proporciona.

6) Repetiremos los pasos 3, 4 y 5 hasta que ningún cambio sea capaz de proporcionar una mejora significativa.

En la figura 3 podemos ver un ejemplo en dos dimensiones, es decir para $m = 2$.

Figura 3. Semillas *k-means*

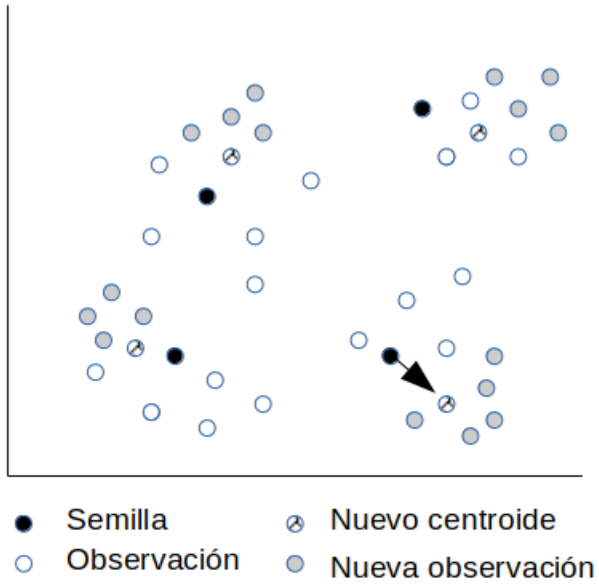


En la figura 4 podemos apreciar de una forma visual el mecanismo de generación de nuevos centroides.

2.2.2. Criterios para seleccionar *k*

Uno de los inconvenientes que tiene *k-means* es el hecho de que requiere que se le especifique de antemano el valor *k* (número de clústeres).

Figura 4. Nuevos centroides *k-means*

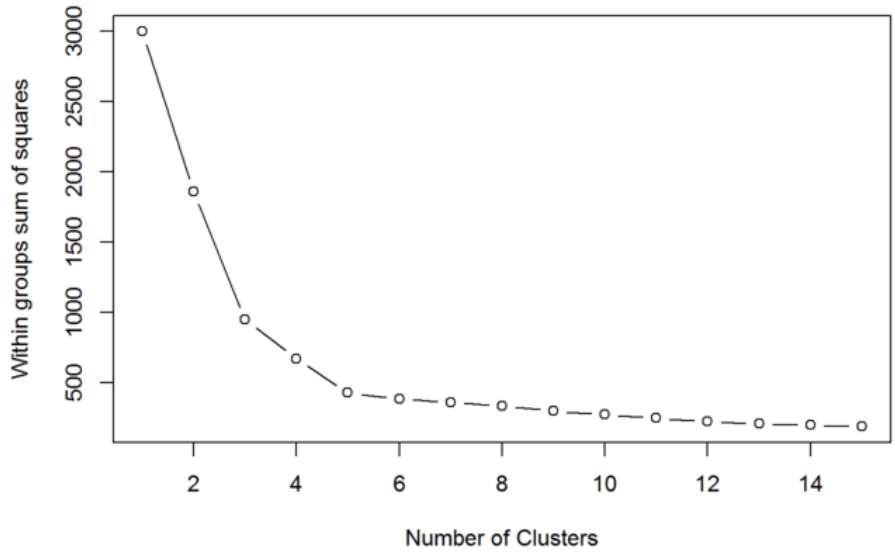


La minimización de distancias intragrupo o la maximización de distancias intergrupo pueden usarse como criterios para establecer un valor adecuado para el parámetro *k*.

Los valores *k* para los que ya no se consiguen mejoras significativas en la homogeneidad interna de los segmentos o la heterogeneidad entre segmentos distintos, deberían descartarse.

En la figura 5 hemos generado un gráfico con la suma de las distancias intragrupo que obtenemos para cada valor de *k*.

Figura 5. Elección de *k*



Observamos cómo, a partir de cinco segmentos, la mejora que se produce en la distancia interna de los segmentos ya es insignificante. Este hecho debería ser indicativo de que cinco segmentos es un valor adecuado para *k*.

2.3. *Canopy clustering*

Podemos pensar esta técnica como una generalización de los algoritmos particionales.

La idea brillante que subyace a esta técnica es que podemos reducir drásticamente el número de cálculos que requieren los algoritmos particionales como *k-means*, introduciendo un proceso previo de generación de **grupos superpuestos** o (*canopies*) a partir de una **métrica más sencilla** de calcular, (*cheapest metric*).

De esta forma, solo calcularemos distancias con la métrica inicial, más estricta y pesada en cálculos, para los puntos que pertenecen al mismo *canopy*.

Podríamos resumirlo diciendo que, previamente, mediante una métrica simple, decidimos qué puntos están definitivamente lejos y, en consecuencia, para estos puntos alejados ya no valdrá la pena malgastar más cálculos con una métrica más exigente.

En realidad, el método del *canopy clustering* divide el proceso de segmentación en dos etapas:

- En la primera, usaremos una métrica sencilla en cálculos con el objetivo de generar los *canopies* o subgrupos superpuestos de puntos. Además, lo haremos de modo que cada punto pueda pertenecer a más de un *canopy* y, a su vez, todos los puntos tengan que pertenecer al menos a un *canopy*.

- En la segunda, utilizaremos un método de segmentación tradicional, como por ejemplo el método *k-means*, pero lo haremos con la siguiente restricción: no calcularemos la distancia entre puntos que no pertenecen al mismo *canopy*.

Para facilitar la comprensión del mecanismo del algoritmo, vamos a situarnos en los dos casos extremos:

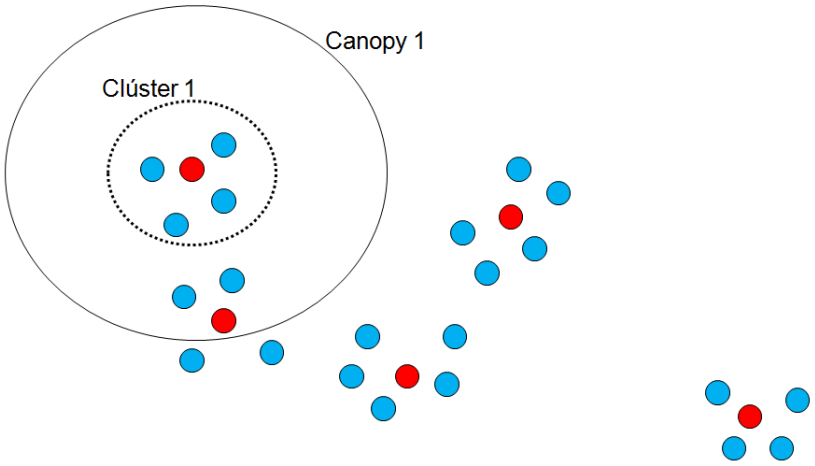
1) Supongamos que, como consecuencia de la primera etapa, nuestro universo de puntos cae por completo en un solo *canopy*. Entonces, el método de segmentación por *canopies* sería exactamente igual al del método de segmentación tradicional seleccionado, es decir, *k-means* en nuestro ejemplo.

2) Supongamos que, como resultado de la primera etapa, generamos *canopies* relativamente pequeños y con muy poca superposición. En este caso, al aplicar la técnica tradicional solo dentro de cada *canopy*, habremos ahorrado un gran número de cálculos.

Para ilustrar de un modo gráfico el proceso de construcción de *canopies* a partir de una métrica simple y el proceso de construcción de clústeres a partir de una métrica más exigente, proporcionamos una serie de tres figuras.

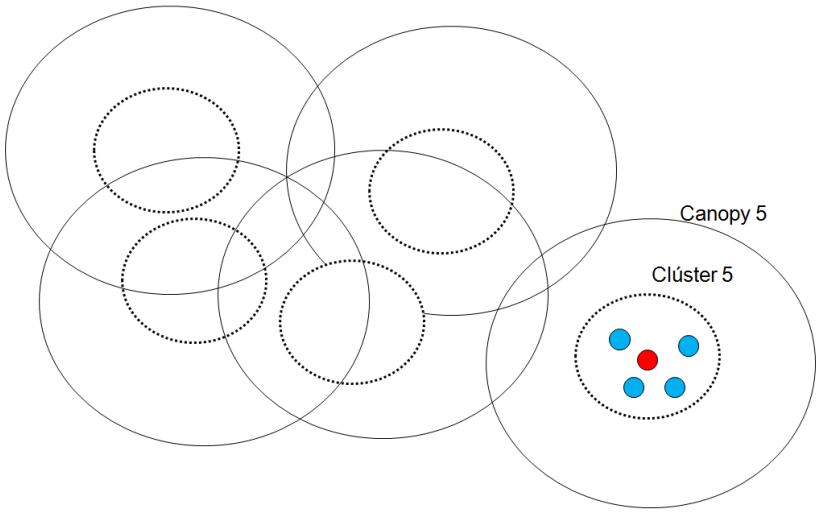
Inicialmente, en la figura 6 se aprecia como los *canopies* se construyen de una forma amplia y con superposiciones.

Figura 6. Construcción del *canopy* 1



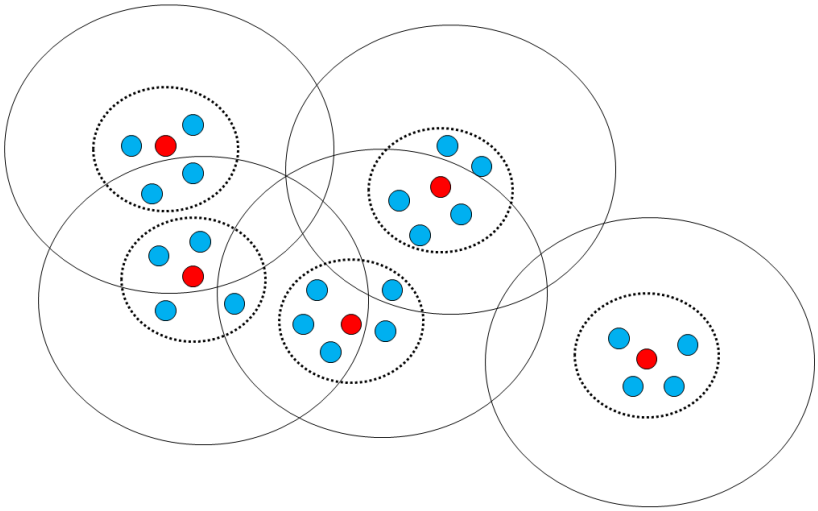
En la figura 7 apreciamos como los clústeres se calculan sin superposiciones y siempre dentro de su *canopy* correspondiente.

Figura 7. Construcción del *canopy* 5



Finalmente en la figura 8 podemos ver el resultado de un proceso de segmentación mediante la técnica del *canopy clustering*.

Figura 8. *Canopy clustering*



3. Modelos basados en la densidad

Los modelos de *clustering* basados en la densidad constituyen una familia de algoritmos que han dado muy buenos resultados y, en consecuencia, han despertado el interés de muchos analistas. Este tipo de algoritmos se especializan en identificar zonas de alta concentración de observaciones separadas entre sí por zonas con menor densidad de observaciones.

Los dos algoritmos más conocidos son DBSCAN y OPTICS.

3.1. Algoritmo DBSCAN

Density-based Spatial Clustering of Applications with Noise, un nombre complejo para un algoritmo que en el fondo es muy sencillo. Veamos cómo funciona.

Inicialmente, requiere que se le informe de dos parámetros:

- El valor ϵ (**epsilon**): máximo radio de vecindad.

  Consideraremos que dos puntos u observaciones están cercanos si la distancia que los separa es menor o igual a ϵ.

- El valor **minPts**: mínimo número de puntos en la ϵ-vecindad de un punto.

  Podemos pensarlo como el valor que marcará nuestro criterio de qué consideramos como denso.

De este modo, DBSCAN irá construyendo esferas de radio ϵ que al menos incluyan *minPts* observaciones.

La lógica que sigue el algoritmo para construir los clústeres o zonas densamente pobladas es la siguiente:

- Se considera que un punto p es un **punto núcleo**, *core point*, si al menos tiene *minPts* puntos a una distancia menor o igual a ϵ. Dicho de otro modo, contiene *minPts* en la ϵ-vecindad.

- Un punto q es **alcanzable** desde p, (*p-reachable)*, donde p es núcleo, si la distancia entre ambos es inferior o igual a ϵ. Dicho de otro modo, si está dentro de la ϵ-vecindad de p.

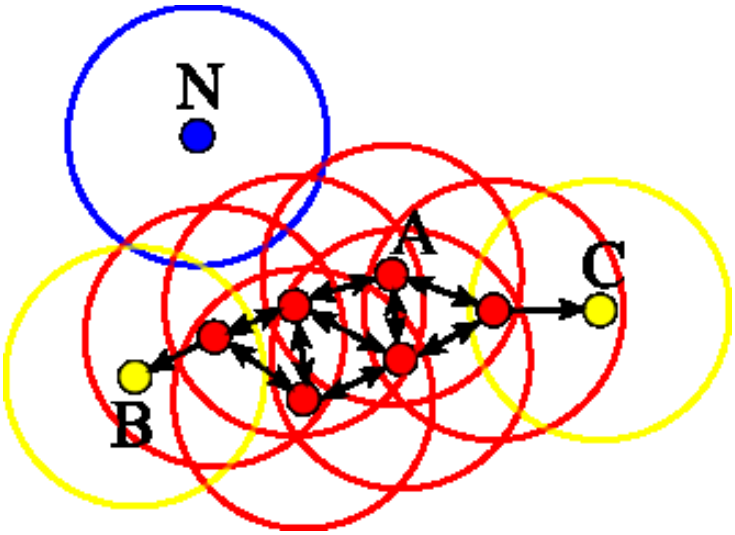
- Un punto q es alcanzable desde p, si existe un camino de puntos núcleo que los conecta.

  Explicado más formalmente, si existe $p_1,...,p_n$, con $p_1 = p$ y $p_n = q$, donde cada p_{i+1} es alcanzable por p_i y todos los $p_1,...,p_{n-1}$ son puntos núcleo.

- Cualquier punto no alcanzable se considerará punto extremo o *outlier*.

La figura 9 nos muestra de un modo esquemático, el proceso de construcción de zonas de densidad. En este ejemplo se toma *minPts* = 4.

Figura 9. Proceso de construcción de zonas de densidad



Fuente: By Chire - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=17045963

Los puntos *B* y *C* corresponden a la frontera del clúster, es decir, son puntos alcanzables desde un punto núcleo, pero ellos mismo no son punto núcleo porque no incluyen *minPts* en su ϵ-vecindario.

Los puntos *A* son puntos núcleo ya que como mínimo cada uno de ellos tiene 4 puntos en un radio ϵ pre-fijado.

Finalmente, el punto *N* se considera extremo o *outlier* puesto que no es alcanzable desde ningún punto del juego de datos.

3.1.1. Ventajas e inconvenientes

La principal ventaja de DBSCAN es que es capaz de identificar **clústeres de cualquier forma geométrica**, no solo circular, ya que solo necesita que exista la combinación de zonas con alta y baja densidad de concentración de puntos. Es especialmente bueno **identificando valores extremos**. Contrariamente a otros algoritmos, para DBSCAN no supone ningún inconveniente trabajar con un juego de datos con este tipo de valores.

DBSCAN **tampoco requiere que le prefijemos el número de clústeres** que queremos que identifique. Lo único que necesita es que haya zonas de baja densidad de puntos para así poder marcar bien las fronteras entre clústeres.

En cuanto a inconvenientes, el principal es el hecho de tener que fijar como parámetros de entrada los valores ϵ y *minPts*. **Acertar con el valor óptimo de estos parámetros** requiere de experiencia y conocimiento tanto sobre el algoritmo en sí como sobre el propio juego de datos.

3.2. Algoritmo OPTICS

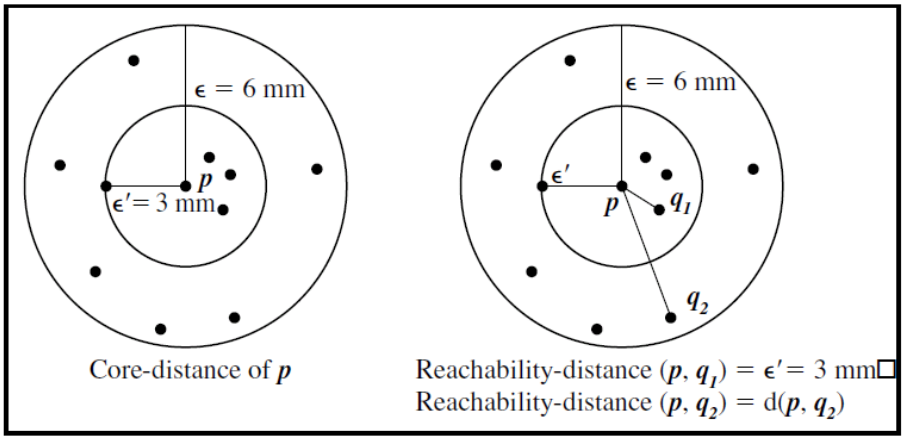
Ordering Points to Identify Cluster Structure es un algoritmo que de algún modo generaliza DBSCAN y resuelve su principal inconveniente: los parámetros iniciales.

OPTICS requiere un radio ϵ y un criterio de densidad *minPts* igual que DBSCAN, pero en el caso de OPTICS el valor de radio ϵ no determinará la formación de clústeres sino que servirá para ayudar a reducir la complejidad de cálculo en el propio algoritmo.

En realidad OPTICS no es un algoritmo que genere una propuesta de clústeres a partir de un juego de datos de entrada, como DBSCAN. De hecho, lo que hace es ordenar los puntos del juego de datos en función de su distancia de alcanzabilidad, o ***reachability distance***, en inglés.

Para entender bien este concepto nuevo, nos basaremos en la figura 10, donde hemos tomado *minPts* = 5.

Figura 10. *Reachability-distance*



La ***core-distance*** del punto p es el radio ϵ' mínimo tal que su ϵ'-vecindad contiene al menos *minPts* = 5 puntos.

La ***reachability-distance*** de un punto q respecto de un punto núcleo (*core-point*) p será la mayor de las dos distancias siguientes:

- *core-distance* del punto p,
- distancia euclidiana entre los puntos p y q, que denotaremos por $d(p,q)$.

Siguiendo con el ejemplo de la figura 10, vemos cómo la *reachability-distance* de los puntos p y q_1 es la *core-distance* del punto p, porque esta es mayor que la distancia euclidiana entre los puntos p y q.

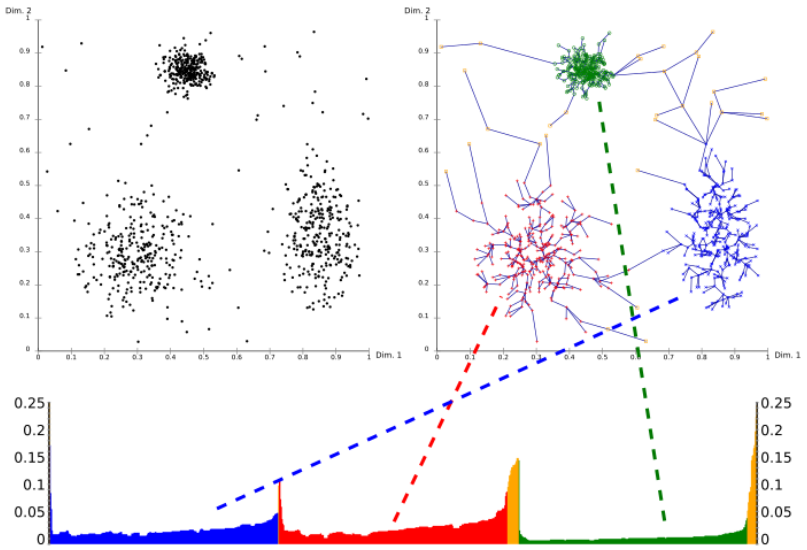
Por otro lado, la *reachability-distance* de los puntos p y q_2 es la distancia euclidiana entre ellos, porque esta es mayor que la *core-distance* del punto p.

OPTICS como algoritmo lo que nos va a hacer es asignar a cada punto del juego de datos una *reachability-distance*.

Aclarados estos conceptos básicos, podemos avanzar en la comprensión de la utilidad de disponer de dicha ordenación. Para ello usaremos un tipo de gráfico específico para este algoritmo, el *reachability plot*.

Para entender bien qué es un *reachability plot* veamos la figura 11. En el gráfico inferior vemos la *reachability-distance* asignada a cada punto y apreciamos como hay zonas con valores altos que se corresponden con los puntos *outliers* y zonas con valores muy bajos que se corresponden con puntos ubicados en zonas densas.

Figura 11. *Reachability-plot*



Fuente: By Chire - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=10293701

Fijémonos que a la hora de generar los clústeres podremos decidir cuál es la *reachability-distance* límite que nos marca qué consideramos como clúster.

Podremos calibrar o ajustar este valor límite hasta conseguir una generación de clústeres adecuada.

La posibilidad de calibrar la *reachability-distance* límite, hace que OPTICS en realidad lo que nos dé es una ordenación de puntos por *reachability-distance* y en consecuencia será el propio analista quien podrá generar múltiples combinaciones de clústeres en función del límite que se quiera fijar.

3.2.1. Ventajas e inconvenientes

DBSCAN presupone que la densidad que encontrará en todos los clústeres es un valor constante. Sin embargo, OPTICS permite que el valor de densidad sea variable en un juego de datos, precisamente por la habilidad de fijar el límite en el punto que queramos.

4. *Affinity propagation*

Se trata de un algoritmo de *clustering* que basa su lógica en el intercambio de mensajes entre los distintos puntos del juego de datos.

Una diferencia significativa entre otro algoritmo de *clustering* como *k-means* y *affinity propagation* es que en *k-means* empezamos con un número predefinido de clústeres y una propuesta inicial de centros potenciales, mientras que en *affinity propagation* cada punto del juego de datos se trata como un centro potencial o *exemplar*.

Como entrada, el algoritmo requiere que se le pasen dos juegos de datos:

- Una matriz de similaridades S donde suele tomarse la distancia euclidiana $s(i,k) = -(x_i - x_k)^2$ y donde quedará representado cómo es de adecuado que dos puntos $\{i,k\}$ pertenezcan al mismo clúster.

- Una relación de preferencias donde reflejaremos qué puntos consideraremos más apropiados para jugar el papel de ejemplares.

A partir de aquí, *affinity propagation* centrará su procedimiento en dos matrices:

- La matriz de responsabilidades R: $r(i,k)$ nos va a indicar cómo de bien encaja el punto k como punto ejemplar de i.

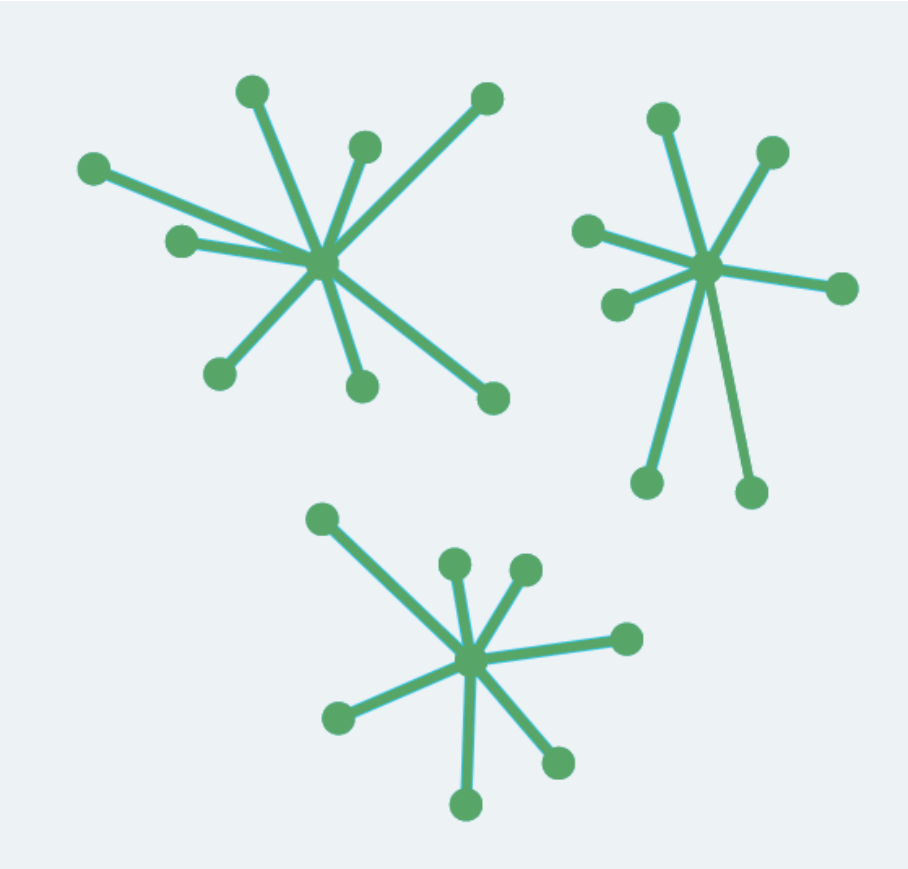
- La matriz de disponibilidad A: $a(i,k)$ nos indicará cómo de adecuado es para el punto i considerar k como su ejemplar.

Ambas matrices pueden ser interpretadas como probabilidades logarítmicas y es por este motivo que consideraremos su valores en negativo.

En la figura 12 vemos cómo los distintos puntos del juego de datos se relacionan con su punto ejemplar de referencia.

El mecanismo interno del algoritmo consiste en un intercambio de mensajes entre puntos. Estos se alinean entre ellos formando clústeres locales de modo que, a medida que avanza el proceso iterativo, siguen intercambiando mensajes hasta que se llegan a formar clústeres más grandes y estables.

Figura 12. *Affinity propagation*



En la expresión formal de la matriz de responsabilidades R (ecuación 16), observamos cómo esta refleja la evidencia acumulada de cómo es de adecuado el punto k para servir como ejemplar para el punto i, teniendo en cuenta otros posibles ejemplares para el punto i.

$$r(i,k) = s(i,k) - max_{k\prime \notin \{i,k\}}\{a(i,k\prime) + s(i,k\prime)\} \tag{16}$$

Por otro lado, vemos cómo la expresión formal de la matriz de disponibilidad *A availability matrix* (ecuación 17) refleja la evidencia acumulada de cómo de apropiado sería para el punto i elegir el punto k como su ejemplar, teniendo en cuenta el apoyo de otros puntos para que el punto k fuera un ejemplar.

Está dividida en dos partes:

$$a(i,k) = min\{0,r(k,k) + \sum_{i\prime \notin \{i,k\}} max\{0,r(i\prime,k)\}\} \tag{17a}$$

$$a(k,k) = \sum_{i\prime \neq k} max\{0,r(i\prime,k)\} \tag{17b}$$

- La primera para los puntos fuera de la diagonal de A, es decir, para los mensajes que van de un punto a otro.

- La segunda para los puntos en la diagonal de A, es decir, para el mensaje de disponibilidad que un punto se envía a sí mismo.

Affinity propagation, desde su aparición en 2007, ha ido ganando adeptos como uno de los mejores algoritmos en tareas de *clustering* y por su buen rendimiento en multitud de situaciones.