
Introducción a la Minería de Datos

J. Mario García Valdez

Análisis de Clusters

El análisis de clusters o grupos, es la tarea de aglutinar conjuntos de objetos similares (en cierto sentido). Los objetos que pertenecen a un cluster formado como resultado del análisis, deben ser más similares entre sí que con los objetos de otros grupos, incluso se desea que la distancia entre los grupos sea lo mayor posible. Es una de las principales tareas de la minería de datos, y el aprendizaje no supervisado.

Podemos hacer análisis de grupos sin necesidad de programar algoritmos o utilizar métodos matemáticos ya que el *agrupar objetos* es una de las tareas, que sin pensar, realizamos todos los días. ¿Cómo agruparías a los objetos de la siguiente figura?. Recuerda que se deben agrupar objetos similares. Podemos agrupar a los objetos, considerando su (a) ubicación, (b) tamaño, (c) color o algún otro atributo, incluso podemos considerar varias características, por ejemplo, el color y la figura. En todos estos casos hemos realizado un *análisis de grupos* sobre las figuras. Una de las preguntas que debemos hacernos al realizar este tipo de análisis es ¿Cuál de las agrupaciones es la correcta?, para ayudarnos a contestarla se han propuesto varias métricas, pero mucho depende del contexto de aplicación.

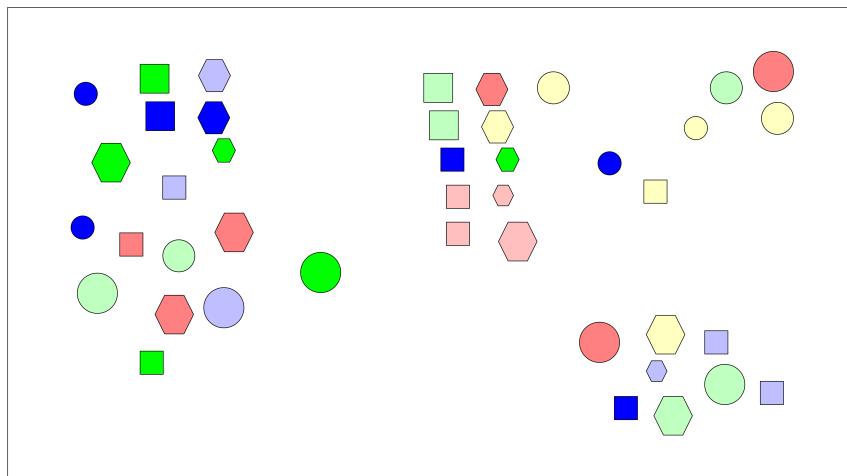


Figura 1: Conjunto de objetos con distinta posición, figura, color y tamaño.

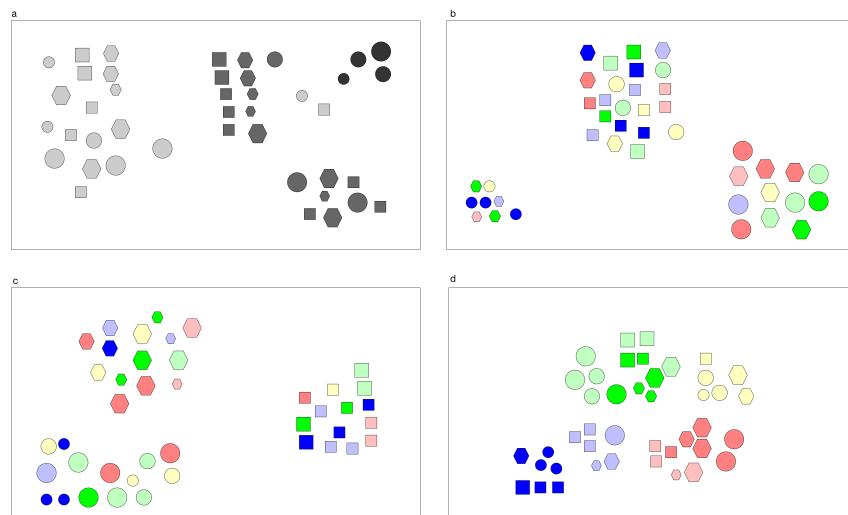


Figura 2: Distintos grupos extraídos de un análisis de grupos realizado sobre el conjunto de datos de ejemplo.

Si te fijas, esta misma actividad la realizan los científicos para clasificar o entender distintos fenómenos. Por ejemplo, el estudio de la evolución de las especies requiere de agrupar animales que tienen características similares. En la figura siguiente, se muestra un dendograma, donde se agrupan las relaciones evolutivas entre animales del orden carnívora (imagen del Instituto de Biología Canina), donde podemos ver datos interesantes, como que el oso negro es más parecido a una morsa que a un perro.

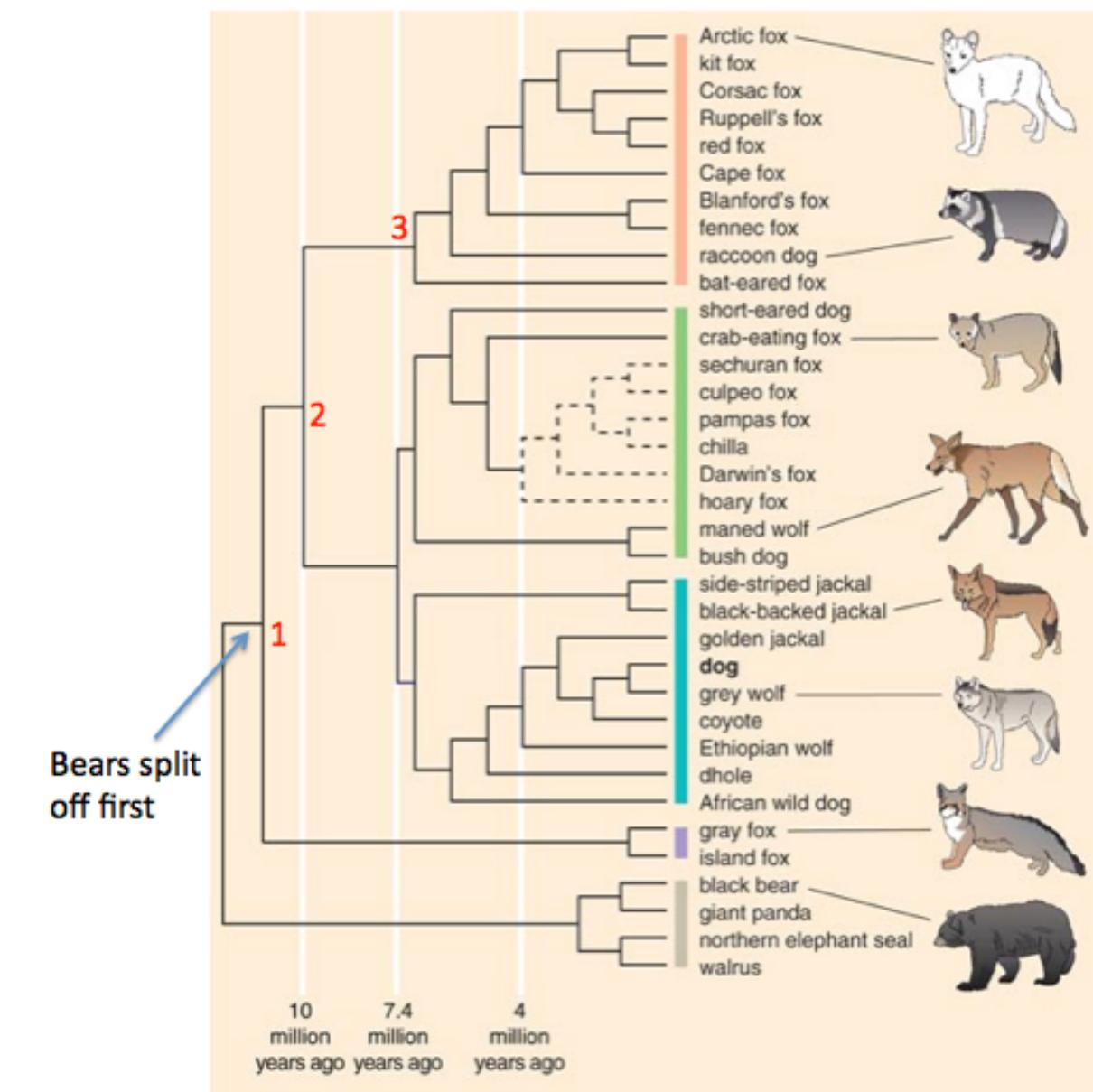


Figura 3: Imagen del Instituto de Biología Canina

Los métodos matemáticos de análisis de grupos, pueden realizar la misma tarea, pero en lugar de objetos reales como los animales o figuras, trabajan sobre datos que representan a los objetos. La representación más común es una matriz con vectores de datos continuos, categóricos u ordinales. Por ejemplo, las figuras analizadas anteriormente, pueden ser representadas por la siguiente matriz:

| id | Figura | Posición | Color | Tamaño |
|----|----------|----------|--------|--------|
| 1 | Círculo | 100, 230 | 0000FF | 50, 50 |
| 2 | Cuadrado | 430, 450 | CCFFCC | 70, 70 |
| 3 | Círculo | 600, 230 | FFFFCC | 30, 30 |
| 4 | Hexágono | 300, 330 | 99DDFF | 30, 30 |
| 5 | Hexágono | 700, 530 | 0000FF | 70, 70 |
| 6 | Cuadrado | 200, 230 | FFFFCC | 70, 70 |

Los grupos encontrados a partir del análisis de clusters, son potencialmente *clases o categorías* de objetos: Podemos sugerir que el análisis de grupos es el descubrimiento de categorías sin necesidad de entrenamiento ya que las clases pueden extraerse de los datos sin necesidad de que sean asignados previamente. Cuidado, los algoritmos de análisis de grupos no reemplazan a los algoritmos de clasificación, ya que las categorías no solo dependen de la similaridad de ciertas características. Antes de pasar a los algoritmos veamos primero algunas de las aplicaciones de las técnicas de agrupamiento:

- **Video Juegos** Para los desarrolladores empresas es útil, identificar usuarios con comportamientos similares. Por ejemplo, a partir de capturar el movimiento y actividades de los usuarios, se han identificado tipos de jugadores como: agresivos, cazadores, exploradores, etc. Esta información se utiliza después para hacer cambios en la mecánica del video juego o para dar promociones a otros usuarios.
- **Recuperación de Información** Para mejorar el desempeño de la recuperación de la información, los documentos similares pueden almacenarse cerca unos de otros, mismo servidor, archivo, máquina, etc.
- **Datos Representativos** En lugar de trabajar con todos los datos para realizar algún análisis, se puede trabajar con objetos representativos de los distintos clusters generados previamente.
- **Cuantificación Vectorial** A partir de los clusters se pueden obtener vectores característicos. Existen técnicas de compresión como la cuantificación vectorial, que se basan en encontrar vectores característicos para un grupo de objetos similares, así en lugar de almacenar todos los datos de estos objetos, solo se almacena el identificador de su vector característico.

Clustering basado en prototipos

En este tipo de algoritmo, se tiene un objeto prototipo para cada uno de los grupos. Así la membresía de los objetos a su grupo depende de la similaridad con los prototipos. Para conjuntos de datos

continuos, el prototipo es el centroide de todos los objetos que pertenecen al grupo.

Clustering basado en densidades

Los grupos se definen como una región de alta densidad rodeada de regiones de baja densidad.

Clustering basado en grafos

Los objetos dentro de la estructura de un grafo tienen conexiones intra-grupales pero no conexiones con otros grupos.

k -medias

El algoritmo de k -medias es un algoritmo muy sencillo basado en prototipos. Solo requerimos especificar el parámetro k donde indicamos el número de clusters que deseamos obtener. En este algoritmo cada centroide define a un clúster. El algoritmo es el siguiente:

1. Se elige de manera aleatoria la posición de los k centroides.

Repetir estos pasos hasta que los centroides no se muevan:

1. Cada objeto se asigna al centroide más cercano.
2. Se Recalculan de nuevo los centroides a partir de los objetos que pertenecen a ellos.
3. Detener si ninguno cambia de posición, de otro modo continuar el ciclo.

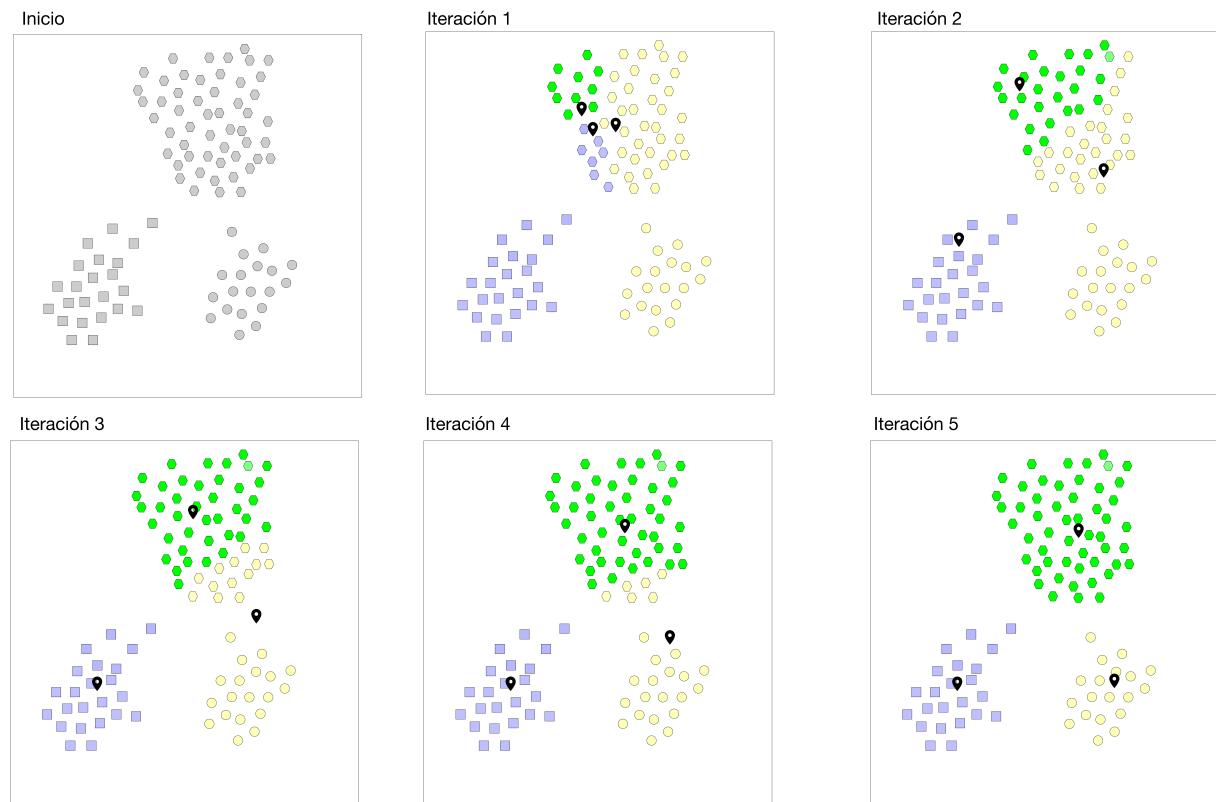


Figura 4: Ejemplo del algoritmo k -medias

Veamos los detalles del algoritmo sobre los datos de la figura x. Podemos observar que hay tres grupos **bien formados**, designados como círculos, rectángulos y hexágonos. Al inicio se posicionan los tres centroides, indicados por el ícono de lugar (●). El siguiente paso es calcular la distancia o similitud entre los objetos y los centroides. En este caso utilizamos la distancia euclídea, ésta es muy común ya que es fácil de interpretar y Digamos que para cada objeto se calcula la distancia con cada centroide y se asigna al centroide más cercano, en este caso aquí lo indicamos con el color. Antes de iniciar la siguiente iteración se vuelven a calcular los centroides y como su posición cambió se inicia la iteración siguiente. En la figura podemos ver como después de cinco iteraciones y se ven pintados correctamente los grupos.

Incorrecta selección de las posiciones iniciales de los centroides

Debemos tener cuidado al ejecutar el algoritmo ya que la ubicación inicial aleatoria puede tener un efecto adverso. Como ejemplo, veamos el posicionamiento sugerido en la figura 2. Como vemos los clusters generados no son buenos ya que claramente vemos que en lugar de dos grupos en la parte superior debería de ser uno, y en la parte inferior dos en lugar de uno.

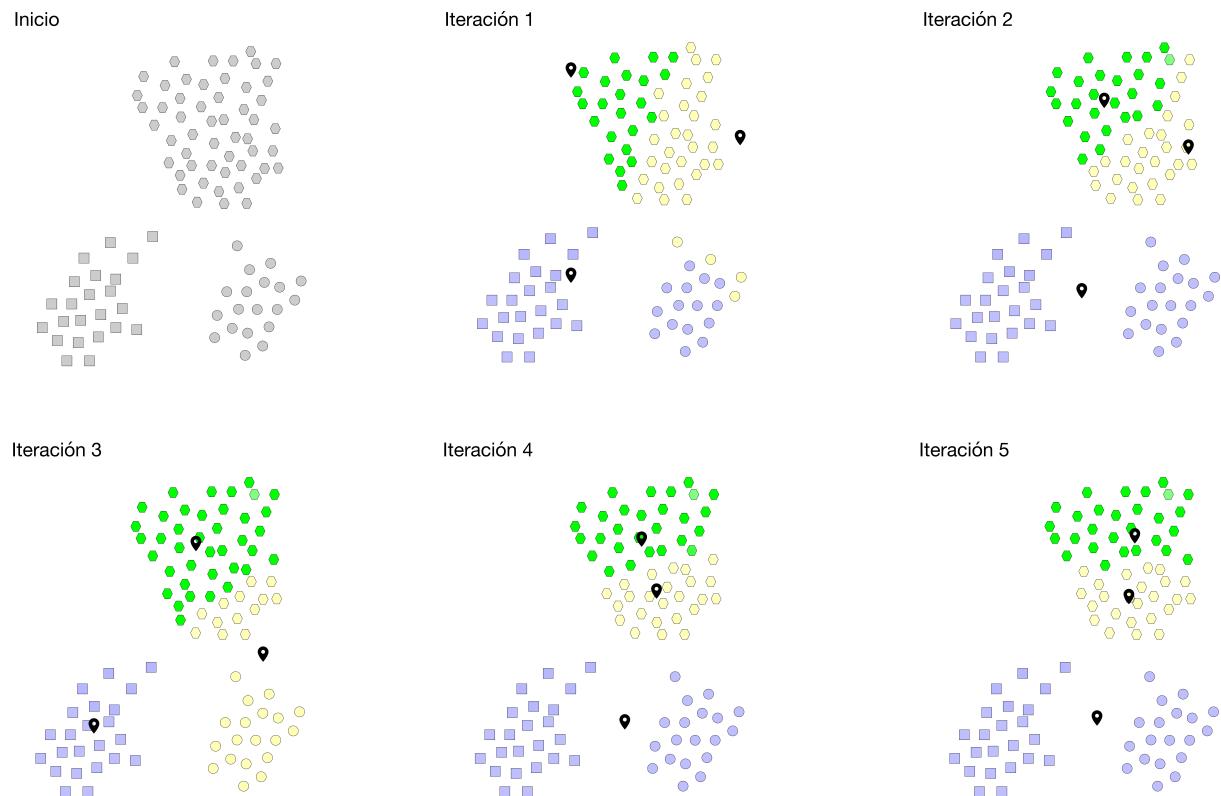


Figura 5: Incorrecta selección de posiciones iniciales de los centroides

¿Cómo evaluamos la calidad de los clusters generados?

Recordemos que lo que buscamos en minimizar la distancia interna de los miembros de un grupo y maximizar la distancia entre grupos. Como observamos en el clustering insuficiente de la figura x, en la parte superior hay dos clusters muy juntos y la parte inferior cluster con objetos muy lejanos entre sí. Una métrica muy utilizada es la de Suma del Error Cuadrático o SSE (del inglés Sum of Square Error), esta métrica considera precisamente la distancia de cada objeto a su centroide como un error, de tal manera que por lo menos eliminamos el problema de la parte inferior ya que al hacer el cálculo seguro dará un error mayor al de la agrupación correcta. La formula es esta:

$$SSE = \sum_{i=1}^k \sum_{x \in C_i} dist(m_i, x)^2$$

Donde,

- k es el número de centroides
- C_i es el conjunto de los miembros del centroide actual
- x es el objeto actual
- m_i es el centroide actual

- *dist* es la función de distancia (en este caso es el error)

Aunque no hay distancias negativas y por lo tanto no hay errores negativos, se mantiene la costumbre de elevar al cuadrado. También podemos observar que si aumentamos el número de clusters k el error es muy posible que disminuya ya que el error se distribuye. El error que suman dos miembros es probable que sea menor que el que generan 3.

¿Cómo evitamos el error provocado por la inicialización?

- Múltiples ejecuciones
- Inicialmente selecciona un número mayor de k centroides y luego de ellos.
- Utiliza otra clustering jerárquico para determinar los centroides.

C-medias difuso

En un algoritmo de agrupamiento difuso, un objeto es asignado a más de un grupo o incluso a todos, con distinto grado de membresía. El algoritmo de agrupamiento difuso más utilizado es *Fuzzy C-Means* (FCM) (????). El algoritmo es muy parecido a k -means, pero los componentes cambian para considerar el aspecto difuso:

Los objetos

En otros algoritmos, cada objeto tiene una etiqueta indicando a qué cluster pertenece. En el caso difuso, cada objeto x debe tener un coeficiente que indique su grado de pertenencia a cada cluster k . El grado de pertenencia se especifica como un peso $w_k(x)$.

El centroide

El centroide difuso considera a todos los puntos para calcularse. Al calcularse, aquellos objetos que tienen mayor membresía tienen mayor peso, mientras que otros solo van a contribuir muy poco al centroide:

$$c_k = \frac{\sum_x w_k(x)^m x}{\sum_x w_k(x)^m},$$

donde m es un parámetro que indica que tan *fuzzy* o *crisp* será la agrupación. Mientras mayor sea, más difusa será la agrupación. Típicamente tiene un valor de 2.

El algoritmo

Para un conjunto de datos $X = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$, el algoritmo regresa como resultado una lista de centroides $C = \{\mathbf{c}_1, \dots, \mathbf{c}_n\}$ y una matriz de pertenencia $W = w_{i,j} \in [0, 1]$, $i = 1, \dots, n$, $j = 1, \dots, c$, en la cual cada elemento w_{ij} , especifica el grado de pertenencia de cada elemento, \mathbf{x}_i , a cada cluster \mathbf{c}_j .

El algoritmo tiene como objetivo minimizar la distancia (o maximizar la similaridad) dentro de cada cluster, como vimos anteriormente esta es una métrica de calidad de los algoritmos de agrupación:

$$\arg \min_C \sum_{i=1}^n \sum_{j=1}^c w_{ij}^m \|\mathbf{x}_i - \mathbf{c}_j\|^2,$$

donde:

$$w_{ij} = \frac{1}{\sum_{k=1}^c \left(\frac{\|\mathbf{x}_i - \mathbf{c}_j\|}{\|\mathbf{x}_i - \mathbf{c}_k\|} \right)^{\frac{2}{m-1}}}.$$

Agrupamiento Jerárquico

Esta técnica busca crear una jerarquía de agrupamientos y se representa típicamente mediante un dendrograma. La representación nos permite ver como se han ido formado los grupos paso a paso, iniciando con objetos individuales en el primer nivel hasta llegar al nivel más alto que incluye a todos los objetos. A diferencia del algoritmo k -medias el cual crea un número determinado de agrupaciones en una agrupación jerárquica elegimos el nivel de agrupación deseado mediante la selección de un punto de corte en el árbol.

Primero vamos a ver el resultado final de un agrupamiento jerárquico.

| | Tijuana | Ensenada | Mexicali | San Diego | Los Angeles | San Francisco | Las Vegas | San Jose | Barstow |
|---------------|---------|----------|----------|-----------|-------------|---------------|-----------|----------|---------|
| Tijuana | - | - | - | - | - | - | - | - | - |
| Ensenada | 104 | - | - | - | - | - | - | - | - |
| Mexicali | 181 | 240 | - | - | - | - | - | - | - |
| San Diego | 32 | 144 | 198 | - | - | - | - | - | - |
| Los Angeles | 221 | 332 | 367 | 124 | - | - | - | - | - |
| San Francisco | 836 | 947 | 980 | 808 | 614 | - | - | - | - |
| Las Vegas | 560 | 669 | 515 | 532 | 432 | 915 | - | - | - |
| San Jose | 768 | 879 | 912 | 740 | 340 | 77 | 848 | - | - |
| Barstow | 309 | 420 | 394 | 176 | 185 | 664 | 254 | 597 | - |

Figura 6: Ejemplo de agrupamiento jerárquico

Antes de aplicar el método de agrupamiento debemos tener una matriz de similaridad como la mostrada en la figura anterior. En este ejemplo básico, la similaridad es igual a la distancia que hay por carretera entre las ciudades, según Google Maps. En otros casos la similaridad deberá calcularse y de ser necesario utilizando datos normalizados.

El método de agrupamiento que utilizaremos recibe el nombre de **método de grupo-par sin pesos, utilizando promedios aritméticos**, en inglés se le conoce por su abreviación UPGMA (???). El método genera un dendrograma el cual refleja la estructura de la matriz de similaridad.

En cada paso se deben seleccionar los dos clusters más similares y combinarlos en uno solo. En caso de empate se selecciona uno de los empatados de manera aleatoria.

En el ejemplo los objetos más similares son Tijuana-San Diego con 28 kilómetros. Estos dos clusters serán reemplazados en la matriz por un cluster al que llamaremos *Merge-1*. Es necesario calcular la distancia entre *Merge-1* y los otros objetos, esto se hace calculando el promedio de las distancias de Tijuana y San Diego a los objetos, ver la Figura. Una vez hecho esto, especificamos la primera conexión en el dendrograma.

| | Merge 1 | Ensenada | Mexicali | Los Angeles | San Francisco | Las Vegas | San Jose | Barstow |
|---------------|-----------|----------|----------|-------------|---------------|-----------|----------|---------|
| Merge 1 | - | - | - | - | - | - | - | - |
| Ensenada | 124 | - | - | - | - | - | - | - |
| Mexicali | 190 | 240 | - | - | - | - | - | - |
| Los Angeles | 207 | 332 | 367 | - | - | - | - | - |
| San Francisco | 822 | 947 | 980 | 614 | - | - | - | - |
| Las Vegas | 546 | 669 | 515 | 432 | 915 | - | - | - |
| San Jose | 754 | 879 | 912 | 340 | 77 | 848 | - | - |
| Barstow | 296 | 420 | 394 | 185 | 664 | 254 | 597 | - |
| | | | | | | | | |
| | | | | | | | | |
| Tijuana | San Diego | Promedio | | | | | | |
| Ensenada | 104 | 144 | 124 | | | | | |
| Mexicali | 181 | 198 | 189.5 | | | | | |
| Los Angeles | 221 | 193 | 207 | | | | | |
| San Francisco | 836 | 808 | 822 | | | | | |
| Las Vegas | 560 | 532 | 546 | | | | | |
| San Jose | 768 | 740 | 754 | | | | | |
| Barstow | 309 | 283 | 296 | | | | | |

Figura 7: Matriz después de la combinación de Tijuana-San Diego



Figura 8: Primer paso de la construcción del dendrograma

Se repite el paso hasta que queden solo dos grupos en la matriz y ambos se atan a la raíz del dendrograma.

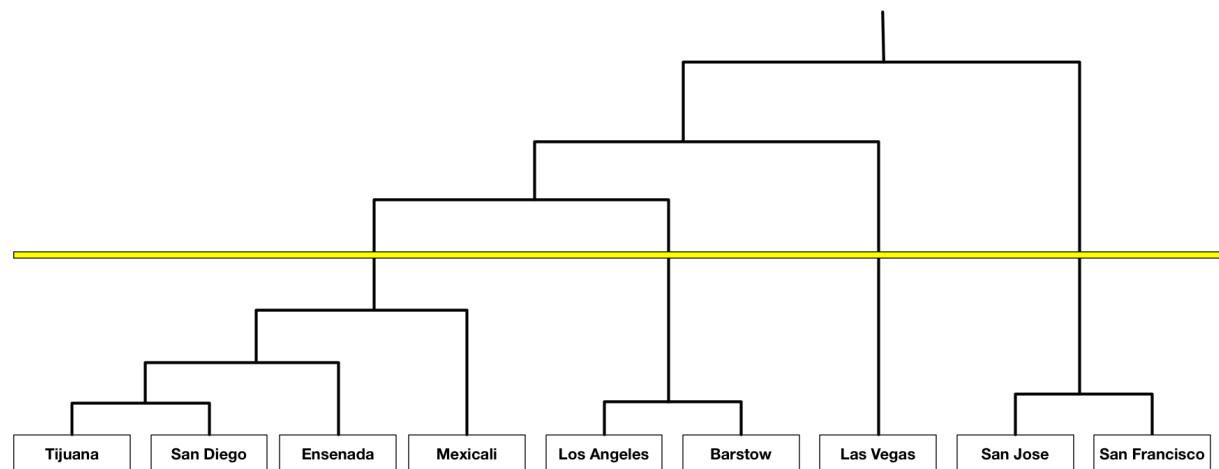


Figura 9: Ejemplo de agrupamiento jerárquico

En este caso estipulamos un corte para obtener cuatro grupos. El resultado también se puede expresar gráficamente como se muestra en la figura.

Introducción a la Minería de Datos

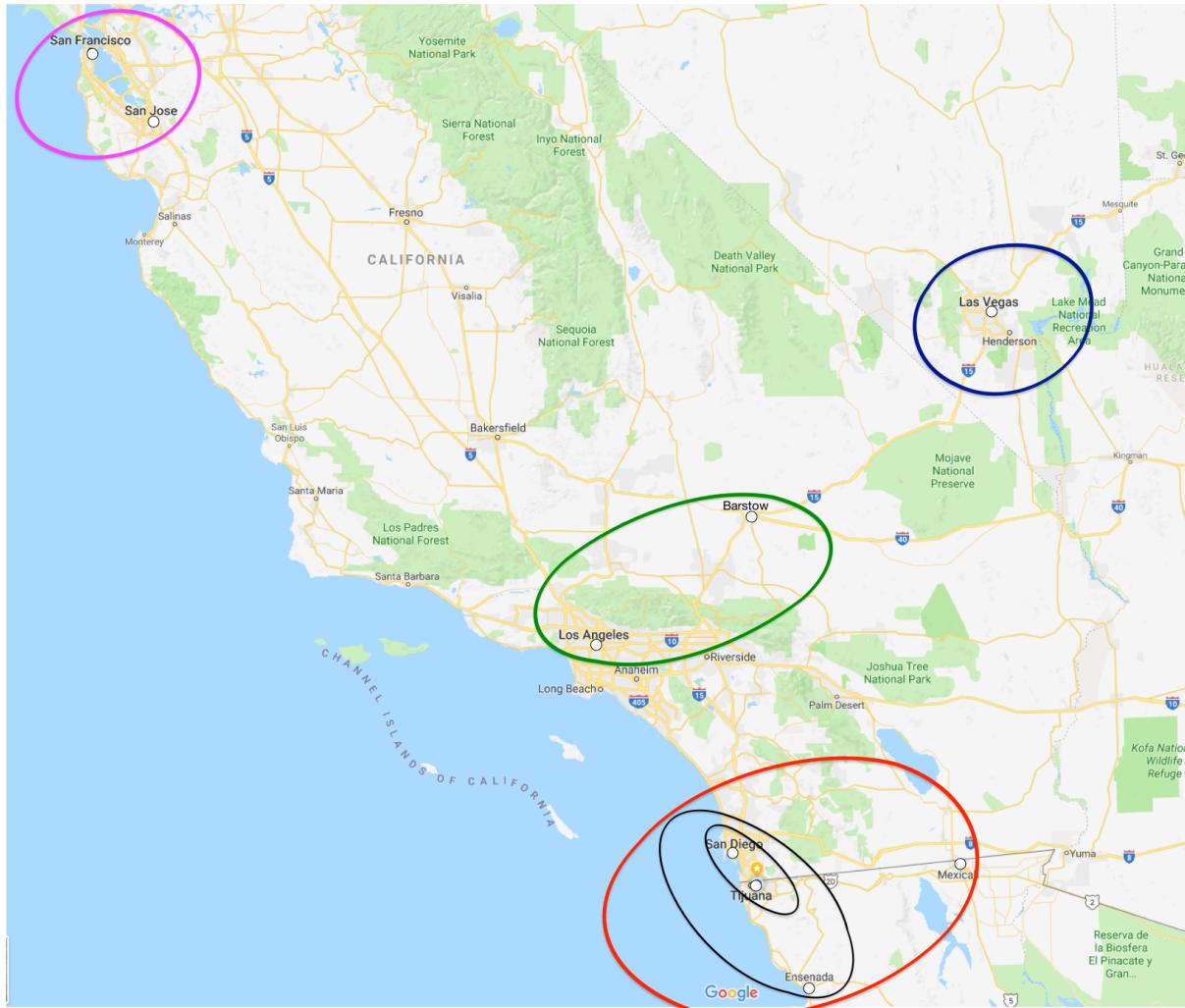


Figura 10: Agrupación jerárquica

Una de las desventajas del método UPGMA es que tiene una complejidad en tiempo de $O(n^3)$. Esto significa que es muy lento para grandes cantidades de datos.

Ejercicio

Busca algún otro algoritmo de agrupamiento jerárquico que tenga una complejidad menor que $O(n^3)$. ¿Qué otras opciones hay para calcular la distancia a un aglomerado además de el promedio? ¿Cómo cambiaría el dendrograma del ejercicio el uso de otra técnica?

DBSCAN

Los algoritmos basados en densidad, como su nombre lo dice, no solo consideran la similitud (o distancia) entre los objetos para definir a los grupos. Estos algoritmos también consideran que haya un mínimo número de objetos en cierta región para considerar que existe un grupo ahí. El objetivo que se busca es evitar que el ruido o los objetos atípicos afecten la agrupación. Algoritmos como k -medias, no discriminan a los objetos que están fuera de cierta frontera que de manera natural forman ciertos objetos. En esas ocasiones cuando los objetos no forman la típica nube que surge de una distribución normal, más bien tiene una forma de anillo o son incluso formas rectangulares (ver figura), es mejor utilizar un algoritmo basado en densidad.

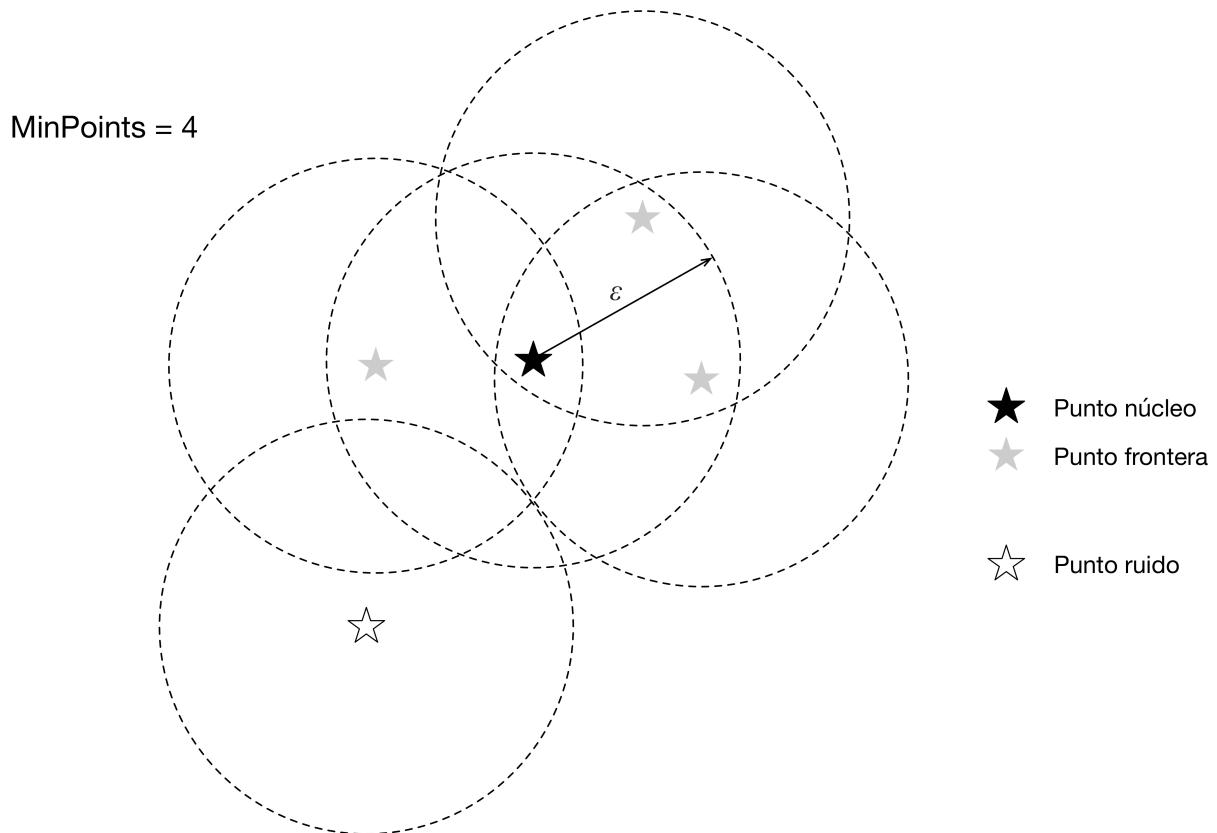


Figura 11: Agrupación DBSCAN

DBSCAN (del inglés *Density-based spatial clustering of applications with noise*) (???), es un algoritmo basado en densidad, es muy popular, es sencillo de entender y lo veremos a continuación.

El algoritmo toma como entrada dos parámetros:

1. El radio Eps , el cual define el *campo de visión* de cada objeto.

2. *MinPoints*, el número mínimo de objetos que debe haber dentro del círculo definido por *Eps*, para que el objeto (o punto) en el centro, sea considerado el núcleo de un cluster.

DBSCAN considera tres tipos de puntos, los cuales se ilustran en la figura:

1. Puntos núcleo (Core points). De nuevo, son aquellos puntos los cuales tienen un número igual o mayor que *MinPoints* en el área definida por el radio *Eps*. El propio punto se contabiliza. Estos puntos son considerados como la parte interna del cluster.
2. Puntos frontera (Border points). Estos puntos no tienen el número mínimo para ser considerados *core* pero se encuentra dentro del área de uno. Estos puntos forman parte del cluster aunque no de su parte interna.
3. Puntos ruido (Noise points). Todos los otros puntos son considerados ruido.

El pseudocódigo de DBSCAN en python es el siguiente:

```

1 def DBSCAN(Puntos, eps, minPts, calcula_distancia):
2     C = 0          # Contador de clusters
3     for punto in Puntos:
4         if punto.etiqueta != "indefinido":
5             continue                      # Este punto ya se visitó
6             vecinos = encuentra_vecinos(Puntos, calcula_distancia, punto, eps)
7             ) # encuentra los vecinos + punto actual
8             if len(vecinos) < minPts:           # Hay
9                 suficientes puntos?
10                punto.etiqueta = "ruido"        # No hay,
11                entonces es ruido
12                continue
13                C+= 1                         # Si hay, entonces
14                es un cluster
15                punto.etiqueta = str(C)        # Se
16                etiqueta al punto inicial
17                vecinos.remove(punto)          # Removemos
18                a punto de los vecinos para procesarlos
19                for vecino in vecinos:          # Revisamos
20                    cada vecino
21                    if vecino.etiqueta == "ruido":  # El punto
22                        vecino.etiqueta = str(C)
23                        ruido se cambia a frontera
24                    if vecino.etiqueta != "indefinido":  # Este punto ya
25                        continue
26                        se visitó
27                        vecino.etiqueta = str(C)        # Se etiqueta
28                        como parte del cluster

```

```
19     otros_vecinos = encuentra_vecinos(Puntos, calcula_distancia,
20         vecino, eps)
21     if len(otros_vecinos) >= minPts:                      # Se revisa, si
22         es_nucleo
23         vecinos.extend(otros_vecinos)                      # Si es, se
24         agrega_a_vecinos
```

El método *calcula_distancia* puede variar dependiendo del problema.