EST-297

Métodos de Clustering basados en K-Means

Juan Zamora O.

Mayo, 2024.

Estructura de la Presentación

1. Aprendizaje No-Supervisado
2. Clustering basado en representantes
3. Algoritmo Kernel K-Means
4. Clustering Spectral

Aprendizaje No-Supervisado

- Existen muchas bases de datos no etiquetadas
  - Muchos registros $\rightarrow$ dificultad para etiquetado manual
  - Subjetividad en etiquetado humano es otro problema
  - Costo de tener expertxs etiquetando es alto
- No-Supervisado significa que no hay "**Instructor**" en forma de etiquetas de clase para cada observación o registro

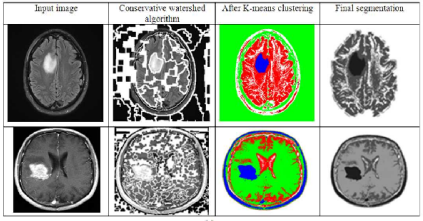
El problema de Clustering

El objetivo de la tarea de Clustering consiste en agrupar objetos similares y separarlos de aquellos “disimiles”

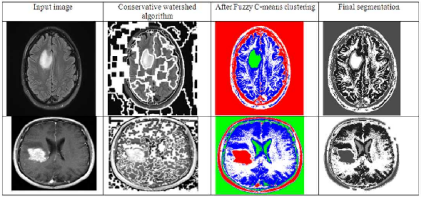
Enfoque empleado exitosamente en diversos dominios tales como:

- Genómica
- Imágenes médicas
- Sistemas recomendadores
- Segmentación de mercado

Ejemplo 1: Identificación de ROI



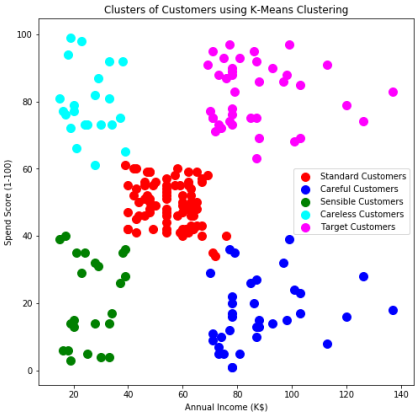
(a)



(b)

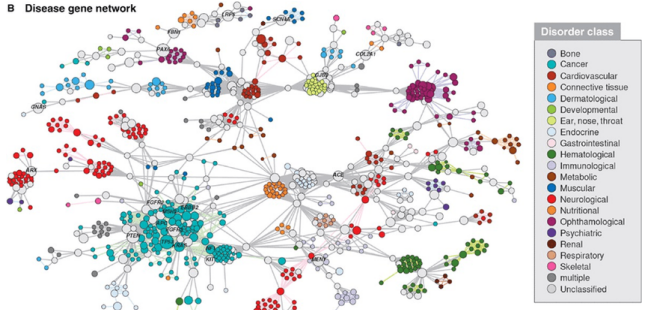
Extraído desde "Segmentation of Medical Image using Clustering and Watershed Algorithms" DOI:10.3844/AJASSP.2011.1349.1352

Ejemplo 2: Segmentación de clientes



https://analyticsindiamag.com/comparison-of-k-means-hierarchical-clustering-in-customer-segmentation/

Ejemplo 3: Grupos en red de enfermedades



https://studentwork.prattsi.org/infovis/visualization/networks-human-disease/

Métodos de Clustering

- Sin etiquetas, es necesario hacer algunos supuestos para definir qué propiedades son deseables en un grupo y cuando dos objetos serán considerados como similares
- Existen dos tipos de métodos de Clustering: Particionales y Jerárquicos.
- En este Curso haremos un mayor incapié en los particionales

Clustering basado en representantes

- Dado un conjunto de datos con n puntos en un espacio d-dimensional, $\mathcal{D} = \{\mathbf{x}_j\}_{j=1}^{n}$
- Dado también $k \in \mathbb{Z}$, el número de grupos a encontrar en $\mathcal{D}$
- El **objetivo** de la tarea es particionar $\mathcal{D}$ en k grupos o *clusters*, representados como $\mathcal{C} = \{C_1, C_2, \dots, C_k\}$
- Luego, para cada cluster C_i existe un punto representativo μ_i
  - Comúnmente se le denomina centroide y corresponde a

$$\mu_i = \frac{1}{|C_i|} \sum_{\mathbf{x}_j \in C_i} \mathbf{x}_j$$

Método de Fuerza bruta

Una posible solución a este problema consiste en

1. Generar todos las posibles particiones de n puntos en k grupos, $\binom{n}{k}$
2. Para cada una, evaluar alguna puntuación que favorezca *ciertas propiedades*
   - Podría usarse

$$SSE(\mathcal{C}) = \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in C_i} \|\mathbf{x}_j - \mu_i\|^2$$

3. Reportar aquella partición con la mejor puntuación, es decir

$$\mathcal{C}^* = \underset{\mathcal{C}}{\texttt{argmin}}\, SSE(\mathcal{C})$$

Problema $\mathcal{O}(k^n/k!)$ posibles soluciones …

Algoritmo K-Means

- K-Means utiliza una estrategia *Greedy* e iterativa para encontrar $\mathcal{C}^*$
  1. Inicializa los centroides al azar, generando k puntos en $\mathbb{R}^d$
  2. Comienza a iterar y en cada iteración realiza dos pasos: Asignación a un grupo y actualización de centroides
  3. Cada punto es asignado al grupo asociado al centroide más cercano.
  4. Se actualizan los centroides, calculando nuevamente las medias sobre los puntos asignados en el paso anterior.
  5. Al alcanzarse una cierta cantidad de iteraciones, no haber cambios en las asignaciones o bien, no encontrarse cambios significativos, el método termina.

K-MEANS (D, k, ϵ):

$t = 0$

Randomly initialize k centroids: $\boldsymbol{\mu}_1^t, \boldsymbol{\mu}_2^t, \ldots, \boldsymbol{\mu}_k^t \in \mathbb{R}^d$

repeat

  $t \leftarrow t + 1$

  $C_j \leftarrow \emptyset$ for all $j = 1, \cdots, k$

  `// Cluster Assignment Step`

  foreach $\mathbf{x}_j \in \mathbf{D}$ **do**

    $j^* \leftarrow \arg\min_i \left\{ \left\| \mathbf{x}_j - \boldsymbol{\mu}_i^t \right\|^2 \right\}$ `// Assign` $\mathbf{x}_j$ `to closest centroid`

    $C_{j^*} \leftarrow C_{j^*} \cup \{\mathbf{x}_j\}$

  `// Centroid Update Step`

  foreach $i = 1$ *to* k **do**

    $\boldsymbol{\mu}_i^t \leftarrow \frac{1}{|C_i|} \sum_{\mathbf{x}_j \in C_i} \mathbf{x}_j$

until $\sum_{i=1}^{k} \left\| \boldsymbol{\mu}_i^t - \boldsymbol{\mu}_i^{t-1} \right\|^2 \leq \epsilon$

Algoritmo Kernel K-Means

- K-means realiza una separación en el espacio original de características de los datos
- El uso del *Kernel Trick* puede ayudar a construir grupos usando transformaciones no-lineales de este espacio

- Se mapea cada $\mathbf{x_j} \in \mathcal{C}$ mediante $\phi(\mathbf{x_j})$
- Una función de kernel $\mathbf{K}$ es aplicada sobre cada par de puntos, $\mathbf{K}(\mathbf{x_i}, \mathbf{x_j}) = \phi(\mathbf{x_i})^T \phi(\mathbf{x_j})$
- Se aplica K-Means en este nuevo espacio
- La nueva función objetivo en el espacio de características es:

$$\min_{\mathcal{C}} SSE(\mathcal{C}) = \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in C_i} \|\phi(\mathbf{x}_j) - \mu_i^{\phi}\|^2$$

$$\min_{\mathcal{C}} SSE(\mathcal{C}) = \sum_{j=1}^{n} \mathbf{K}(\mathbf{x}_j, \mathbf{x}_j) - \sum_{i=1}^{k} \frac{1}{|C_i|} \sum_{\mathbf{x}_p \in C_i} \sum_{\mathbf{x}_q \in C_i} \mathbf{K}(\mathbf{x}_p, \mathbf{x}_q)$$

KERNEL-KMEANS($\mathbf{K}, k, \epsilon$):

$t \leftarrow 0$

$\mathcal{C}^t \leftarrow \{C_1^t, \ldots, C_k^t\}$ // Randomly partition points into k clusters

repeat

  $t \leftarrow t+1$

  foreach $C_i \in \mathcal{C}^{t-1}$ **do** // Squared norm of cluster means

    $\text{sqnorm}_i \leftarrow \frac{1}{n_i^2} \sum_{\mathbf{x}_a \in C_i} \sum_{\mathbf{x}_b \in C_i} K(\mathbf{x}_a, \mathbf{x}_b)$

  foreach $\mathbf{x}_j \in \mathbf{D}$ **do** // Average kernel value for $\mathbf{x}_j$ and C_i

    foreach $C_i \in \mathcal{C}^{t-1}$ **do**

      $\text{avg}_{ji} \leftarrow \frac{1}{n_i} \sum_{\mathbf{x}_a \in C_i} K(\mathbf{x}_a, \mathbf{x}_j)$

  // Find closest cluster for each point

  foreach $\mathbf{x}_j \in \mathbf{D}$ **do**

    foreach $C_i \in \mathcal{C}^{t-1}$ **do**

      $d(\mathbf{x}_j, C_i) \leftarrow \text{sqnorm}_i - 2 \cdot \text{avg}_{ji}$

    $j^* \leftarrow \arg\min_i \{d(\mathbf{x}_j, C_i)\}$

    $C_{j^*}^t \leftarrow C_{j^*}^t \cup \{\mathbf{x}_j\}$ // Cluster reassignment

  $\mathcal{C}^t \leftarrow \{C_1^t, \ldots, C_k^t\}$

until $1 - \frac{1}{n} \sum_{i=1}^{k} |C_i^t \cap C_i^{t-1}| \leq \epsilon$

Vista general del método

1. Pre-procesamiento: Construir una representación de grafo (matriz)
2. Descomposición:
   - Calcular los valores y vectores propios de la matriz
   - Mapear cada punto en $\mathcal{D}$ a una representación vectorial basado en uno o más vectores propios
3. Agrupar
   - Asignar todos los puntos a 2 o más grupos, usando la nueva representación vectorial

Particionamiento espectral de un grafo

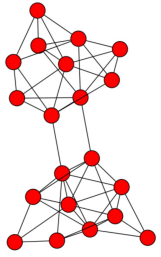
- Dado un grafo $G = (V, E)$, con $V = \{1, 2, ..., n\}$ y $E : V \times V \rightarrow \{0, 1\}$
- A: Matriz de adyacencia de un grafo no dirigido que codifica los vecindarios de cada nodo en G
  - $A_{ij} = 1$ cuando existe conexión entre instancias i y j. 0 en otro caso.
  - Simétrica
  - Tiene n valores propios
  - Vectores propios reales y ortogonales
- D: Matriz de grados
  - $D = [d_{ii}]$, donde d_{ii} es el grado del nodo i
  - Es diagonal

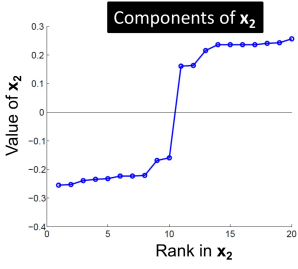
Laplaciano

- $L = D - A$: Matriz Laplaciano
- Todos sus valores propios son ≥ 0
- $xLx = \sum_{ij} L_{ij} x_i x_j \geq 0$ para todo x
- Multiplicidad del primer valor propio es igual al número de componentes conectados
- Si vértices i y j están conectados en el grafo, en algún vector propio asociado al primer valor propio las conmponentes i y j serán 1

Clustering

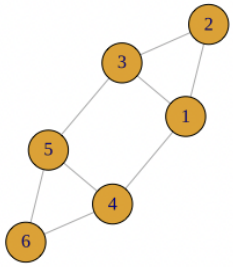
- El segundo valor propio es clave
- El vector propio de este segundo valor será usado para encontrar los grupos
- **Obj.** Minimizar la cantidad de arcos de una partición a otra.
- Este vector es real, no tiene etiquetas $\{-1, +1\}$
  - Puede usarse la función signo





¿Cómo generar más de 2 clusters?

- Tomar los k vectores propios asociados a los valores propios más pequeños
- Esta matriz de $n \times k$ es entregada a otro método (e.g. K-Means)



- **Adyacencia**

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

- **Grado**

$$\begin{bmatrix} 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

Las otras matrices …

- $L = D - A$

- **Laplaciano**

$$\begin{bmatrix} 3 & -1 & -1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ -1 & -1 & 3 & 0 & -1 & 0 \\ -1 & 0 & 0 & 3 & -1 & -1 \\ 0 & 0 & -1 & -1 & 3 & -1 \\ 0 & 0 & 0 & -1 & -1 & 2 \end{bmatrix}$$

- **Valores propios**

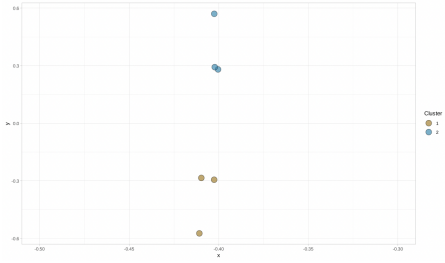
$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 \end{bmatrix}$$

- **Vectores propios**

$$\begin{bmatrix} -0.408 & 0.289 & 0 & -0.577 & -0.408 & 0.5 \\ -0.408 & 0.577 & 0.5 & 0.289 & 0.408 & 0 \\ -0.408 & 0.289 & -0.5 & 0.289 & -0.408 & -0.5 \\ -0.408 & -0.289 & 0 & -0.577 & 0.408 & -0.5 \\ -0.408 & -0.289 & -0.5 & 0.289 & 0.408 & 0.5 \\ -0.408 & -0.577 & 0.5 & 0.289 & -0.408 & 0 \end{bmatrix}$$

Datos proyectados sobre los vectores propios

$$\left(\begin{array}{cc|cccc} -0.408 & 0.289 & 0 & -0.577 & -0.408 & 0.5 \\ -0.408 & 0.577 & 0.5 & 0.289 & 0.408 & 0 \\ -0.408 & 0.289 & -0.5 & 0.289 & -0.408 & -0.5 \\ -0.408 & -0.289 & 0 & -0.577 & 0.408 & -0.5 \\ -0.408 & -0.289 & -0.5 & 0.289 & 0.408 & 0.5 \\ -0.408 & -0.577 & 0.5 & 0.289 & -0.408 & 0 \end{array}\right)$$



$$\left(\begin{array}{|ccc|ccc}
-0.408 & 0.289 & 0 & -0.577 & -0.408 & 0.5 \\
-0.408 & 0.577 & 0.5 & 0.289 & 0.408 & 0 \\
-0.408 & 0.289 & -0.5 & 0.289 & -0.408 & -0.5 \\
-0.408 & -0.289 & 0 & -0.577 & 0.408 & -0.5 \\
-0.408 & -0.289 & -0.5 & 0.289 & 0.408 & 0.5 \\
-0.408 & -0.577 & 0.5 & 0.289 & -0.408 & 0
\end{array}\right)$$