

# Sistemas de Aprendizaje Automático

*Conforme a contenidos del «Curso de Especialización  
en Inteligencia Artificial y Big Data»*



Sistemas de  
**Aprendizaje\_Automático**

**Universidad de Castilla-La Mancha**

Escuela Superior de Informática  
Ciudad Real



# 6

## Capítulo

# Aprendizaje no supervisado

---

Ricardo García Ródenas  
José Ángel Martín Baos

## 6.1. Introducción

En el aprendizaje no supervisado no se dispone de la etiqueta  $y_i$  del individuo (objeto)  $i$  que aparecía en el aprendizaje supervisado. Los datos que se disponen del individuo (objeto)  $i$  vienen almacenados en un vector  $x_i$ . Esto es, tenemos información del mismo conjunto de atributos para todos los individuos. A partir de estos datos el aprendizaje no supervisado resuelve uno de los siguientes problemas:

- **Análisis cluster**, también llamado **análisis de conglomerados**, realiza un particionamiento de los objetos (individuos) en grupos. Cada uno de estos grupos se llama *cluster* o conglomerado. El objetivo es encontrar clusters cuyos objetos sean lo mas homogéneos entre sí y los objetos de clusters diferentes lo mas heterogéneos posible. Cuando los registros de la base de datos están asociados a individuos (en lugar de objetos) este problema también se le conoce como segmentación (de mercados).
- **Detección de anomalías**. Este problema busca los individuos (objetos) que no encaja en los grupos conformados.
- **Reducción de la dimensionalidad de la base de datos**. Este problema consiste en simplificar la base de datos agregando variables con atributos similares. Hay técnicas específicas para abordarlo, como el análisis de componentes principales, pero también puede ser visto como un problema de análisis cluster en el que en lugar de agrupar individuos se agrupan las variables, intentando encontrar aquellas que sean similares entre sí y por tanto repre-

sentables por una única variable (por ejemplo, como una suma de ellas o una media ponderada). Esta visión considera este problema como un análisis cluster dual en el que **en lugar de agrupar filas de la base de datos se agrupan columnas**.

El segundo problema es un aspecto del primer problema y el tercer problema puede enfocarse como un caso particular del primer problema. Por estos motivos nos centraremos en el **análisis cluster**, destacando en cada momento las cuestiones a tener en cuenta para abordar los dos últimos problemas.

Existen **multitud de aplicaciones** del análisis cluster en minería de datos, estadística, análisis de imágenes, bioinformática, etc. De hecho, la **clasificación** es uno de los objetivos fundamentales de la ciencia y el análisis cluster nos proporciona los medios técnicos para realizarla.

Ya desde Linneo (1707 – 1778) las clasificaciones y taxonomías fueron piezas claves en las investigaciones biológicas. Los trabajos de Robert R. Sokal y Peter H. A. Sneath en **1963** ([SS63]) marcan el inicio de las técnicas de clusterización que poco a poco se han ido extendiendo a todos los ámbitos científicos y económicos, debido a una imparable digitalización de las sociedades que crean enormes cantidades de datos que demandan técnicas para resumirlos a través de la definición de un conjunto de patrones a los que se adaptan los datos.

Los algoritmos de análisis cluster se pueden **clasificar** de acuerdo al **tipo de datos, al criterio de agrupamiento o a la teoría que lo fundamenta** (estadística, teoría lógica borrosa, etc). El primer criterio se basa en el nivel de **incertidumbre** que se asume en los datos. Bajo este criterio se diferencian dos grandes grupos:

- \* **Los métodos particionales** (*crisp clustering*) consideran que los objetos sólo pueden pertenecer a **un grupo**. Dentro de esta amplia categoría destacan los basados en el concepto de **distancia** (por ejemplo, algoritmo *K – means* y algoritmos jerárquicos) y los que emplean el concepto de **densidad** (como el DBSCAN y el DPC). Una tercera vía no analizada en estas notas son los métodos clusters basados en **distribuciones de probabilidad**. Estos métodos se fundamentan en la estadística y se basan en distribuciones de probabilidad. Los objetos se agrupan de tal modo que lo más verosímil (probable) es que los objetos de un mismo grupo pertenezcan a la misma distribución de probabilidad.
- \* **Los métodos borrosos** consideran que hay **superposición de grupos** y los objetos pueden ser clasificados en más de un grupo. El resultado de estos algoritmos es la obtención del **grado de pertenencia** de un objeto a cada cluster. El algoritmo más difundido de esta categoría es el Fuzzy C-means (FCM), una versión borrosa del algoritmo *K – means* ([BEF84]).

## 6.2. El problema de clustering

### 6.2.1. El concepto de distancia

A continuación definimos varios conceptos matemáticos empleados en el análisis cluster.

**Definición 1 (Distancia)** Dado un conjunto  $X$ , una distancia sobre  $X$ , es una aplicación  $d : X \times X \mapsto \mathbb{R}$  que a cada par de puntos  $x, y \in X$  le asocia un número real  $d(x, y)$ , que cumple las siguientes propiedades:

- i)  $d(x, y) \geq 0$  para todo  $x, y \in X$  y  $d(x, y) = 0$  si y sólo si  $x = y$ .
- ii)  $d(x, y) = d(y, x)$  para todo  $x, y \in X$ .
- iii)  $d(x, y) \leq d(x, z) + d(z, y)$  para todo  $x, y, z \in X$ .

Los ejemplos más usuales de distancia se muestran en la tabla 6.1.

Calcular empleando la distancia euclídea, de Manhattan y de Chebyshev la distancia entre los puntos  $x = (1, 0, 1)$  e  $y = (-2, -3, 0)$ .



$$d^1(x, y) = |1 - (-2)| + |0 - (-3)| + |1 - 0| = 3 + 3 + 1 = 7$$

$$d^2(x, y) = \sqrt{(1 - (-2))^2 + (0 - (-3))^2 + (1 - 0)^2} = \sqrt{9 + 9 + 1} = \sqrt{19}$$

$$d^\infty(x, y) = \max\{|1 - (-2)| + |0 - (-3)| + |1 - 0|\} = \max\{3, 3, 1\} = 3$$



**Ejercicio.** Una circunferencia se define como el lugar geométrico de puntos que son equidistantes respecto a otro punto (centro de la circunferencia). Por tanto el aspecto dependerá de la distancia empleada. En la figura 6.1 se muestran el aspecto de tres circunferencias obtenidas eligiendo las distancias euclídeas, de Manhattan y la de Chebyshev. ¿Qué distancia corresponde a cada una de las circunferencias?



**Figura 6.1:** Aspectos de circunferencias en función de la distancia empleada.

Tabla 6.1: Ejemplos de distancias empleadas en el análisis cluster

| Distancia   | Definición matemática   |
|---|---|
| <ul style="list-style-type: none"> <li>• <b>Distancia de Minkowski o distancia <math>L_p</math></b> <ul style="list-style-type: none"> <li>◦ Distancia euclídea (<math>p = 2</math>)</li> <li>◦ Distancia Manhattan (<math>p = 1</math>)</li> <li>◦ Distancia Chebyshev (<math>p = \infty</math>)</li> </ul> </li> <li>• <b>Formas cuadráticas</b> <ul style="list-style-type: none"> <li>◦ Distancia de Mahalanobis</li> </ul> </li> </ul> | $d^p(x, y) = \left( \sum_{j=1}^P (x_j - y_j)^p \right)^{\frac{1}{p}}$ $d^2(x, y) = \sqrt{\sum_{j=1}^P (x_j - y_j)^2}$ $d^1(x, y) = \sum_{j=1}^P  x_j - y_j $ $d^\infty(x, y) = \max_{1 \leq j \leq P}  x_j - y_j $ $d_Q(x, y) = [(x - y)^T Q (x - y)]^{\frac{1}{2}}; \text{ donde } Q \text{ es una matriz cuadrada definida positiva.}$ $d_M(x, y) = [(x - y)^T V^{-1} (x - y)]^{\frac{1}{2}}; \text{ donde } V \text{ es la matriz de varianza-covarianza de } x \text{ y } y.$ |

En ocasiones se relaja la condición de la desigualdad triangular y se trabaja con las llamadas funciones de **similitud**. Estas funciones  $d : X \times X \mapsto \mathbb{R}$  se le exigen las propiedades: i)  $d(x, y) \leq d_0$  para todo  $x, y \in X$ , ii) simetría  $d(x, y) = d(y, x)$  y iii)  $d(x, x) = d_0$  para todo  $x \in X$ . El valor  $d_0$  es un número real arbitrario.

**Definición 2 (Espacio métrico)** Un espacio métrico es un par  $(X, d)$ , donde  $X$  es un conjunto y  $d$  es una distancia definida en  $X$ .

Un espacio métrico no tiene una estructura algebraica asociada y por tanto no se puede sumar objetos o multiplicarlos por un escalar. Esto ha motivado que en ciertas aplicaciones, como las del análisis cluster, se suele trabajar con **espacios normados**.

**Definición 3 (Norma)** Una norma sobre un conjunto  $X$  es una aplicación definida de  $\|\cdot\| : X \mapsto \mathbb{R}$ , cumpliendo:

- i)  $\|x\| \geq 0$  para todo  $x \in X$  y  $\|x\| = 0$  si y sólo si  $x = 0$ .
- ii)  $\|\lambda x\| = |\lambda| \|x\|$  para todo  $x \in X$  y  $\lambda \in \mathbb{R}$ .
- iii)  $\|x + y\| \leq \|x\| + \|y\|$  para todo  $x, y \in X$ .

**Definición 4 (Espacio normado)** Un espacio normado es un par  $(X, \|\cdot\|)$  en el que  $(X, +, \cdot)$  es un espacio vectorial y  $\|\cdot\|$  es una norma en  $X$ .

Un espacio normado induce un espacio métrico en el que la distancia se define:

$$d(x, y) = \|x - y\| \text{ para todo } x, y \in X.$$

Las definiciones anteriores proporcionan un marco conceptual lo suficientemente general para describir multitud de problemas de análisis cluster. El espacio normado empleado usualmente en el análisis cluster es el espacio euclideo  $(\mathbb{R}^P, +, \cdot)$  y la distancia es la derivada de la norma  $L_p$ . Esta distancia y otros ejemplos se muestran en la tabla 6.1. Notar que  $P$  es el número de variables del problema y que  $p$  es un número arbitrario elegido por el usuario.

### 6.2.2. Formulación matemática del problema de clustering

A continuación formularemos matemáticamente este problema.

**Definición 5 (Partición)** Diremos que la colección de subconjuntos  $P = \{C_k\}_{k \in \mathcal{K}}$  es una partición de un conjunto  $C$  si cumple las siguientes propiedades:

- i)  $C_k \neq \{\emptyset\}$  para todo  $k \in \mathcal{K}$ .
- ii)  $C_k \cap C_{k'} = \{\emptyset\}$  para todo  $k, k' \in \mathcal{K}$ .
- iii)  $\bigcup_{k \in \mathcal{K}} C_k = C$ .

Al conjunto de todas las particiones de  $C$  lo denotaremos como  $\mathcal{P}(C)$ .

El **problema de análisis cluster** se formula mediante el siguiente modelo de optimización:

$$\begin{aligned} & \text{maximizar} && J(P) \\ & \text{sueto a:} && P \in \mathcal{P}(C) \end{aligned} \tag{6.1}$$

donde  $J(P)$  es una función que calcula la *calidad* de la partición  $P$ , en ocasiones recibe el nombre de función de *mérito*. La definición matemática de la función objetivo  $J(P)$  requiere trabajar con el concepto de distancia. El conjunto  $C$  que se quiere particionar, consta de un número  $n$  de elementos (vectores de  $\mathbb{R}^P$ ) y se desea encontrar un número  $K$  de cluster.

En este caso particular en el que el número de objetos es finito y el número de clusters es  $K$ , el conjunto factible del problema de optimización (6.1) se puede expresar del siguiente modo:

$$\sum_{k=1}^K w_{ik} = 1; \quad i = 1, \dots, n \tag{6.2}$$

$$\sum_{i=1}^n w_{ik} \geq 1; \quad k = 1, \dots, K \tag{6.3}$$

$$\begin{aligned} w_{ik} &\in \{0, 1\}; && i = 1, \dots, n \\ &&& k = 1, \dots, K; \end{aligned} \tag{6.4}$$

donde la variable de decisión  $w_{ik}$  tomará el valor 1 si el objeto  $i$  es asignado al cluster  $k$  y 0 en caso contrario. La ecuación (6.2) indica que todos los objetos deben ser asignados a un único cluster. La ecuación (6.3) impone que no existan clusters vacíos y la ecuación (6.4) establece la naturaleza binaria en las variables.

El problema de particionar  $n$  objetos en  $K$  subconjuntos disjuntos es un problema que no se puede abordar con una estrategia de enumeración completa de todas las posibles particiones. El número de estas particiones se calcula mediante los llamados números de Stirling de segunda clase:

$$S_{n,K} = \frac{1}{n!} \sum_{i=1}^n (-1)^{n-i} \binom{n}{i} n^i \quad (6.5)$$

El número total de particiones que se puede realizar en un conjunto de  $n$  elementos, esto es, el cardinal del conjunto de particiones  $\mathcal{P}(\mathcal{C})$ , viene definido por el llamado número de Bell

$$B_n = \sum_{K=1}^n S_{n,K} \quad (6.6)$$

Por ejemplo, si queremos agrupar 25 objetos en 4 clusters, existen aproximadamente  $4,69 \times 10^{13}$  particiones diferentes. [KR90] mostraron que ese número de Stirling  $S_{n,K}$  puede ser aproximado por  $K^n/K!$ , lo que muestra un crecimiento exponencial. Para pequeños valores de  $n$  entre 20 y 30 se ha empleado la llamada programación dinámica para encontrar agrupamientos óptimos. Debido a que es un problema  $\mathcal{NP}$ -completo se han diseñado métodos heurísticos para obtener buenas particiones en un tiempo computacional razonable. El método que quizás haya sido el más empleado y estudiado es el algoritmo de las  $K$ -medias.

### 6.3. Algoritmo de las $K$ -medias

Desde el inicio del análisis cluster se puso de manifiesto que los conglomerados deberían tener un cierto grado de homogeneidad interna y heterogeneidad entre los grupos. Históricamente muchos investigadores hicieron operativa esta definición mediante la minimización de la variación dentro del grupo, [Cox57]. El criterio más ampliamente usado para este fin es el de minimizar la suma de los errores al cuadrado (SSE), cuya formulación matemática es la siguiente.

Supongamos que queremos particionar  $n$  objetos en  $K$  clusters. Cada objeto  $i$  viene definido por el vector  $P$ -dimensional:

$$\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{iP})^T, \quad i = 1 \dots n; \quad (6.7)$$

y cada cluster  $k$  viene definido por el denominado **centroide**:

$$\bar{\mathbf{x}}_k = \frac{1}{n_k} \sum_{i \in C_k} \mathbf{x}_i; \quad (6.8)$$



donde  $n_k$  es el número de objetos en el grupo  $k$ . El centroide es la media aritmética de los objetos y hace el papel del representante del grupo (*patrón*).

La expresión de la suma de los errores al cuadrados dentro de cada cluster viene dada por:


$$SSE = \sum_{j=1}^P \sum_{k=1}^K \sum_{i \in C_k} (x_{ij} - \bar{x}_{kj})^2. \quad (6.9)$$

El índice SSE define la función objetivo del problema (6.1), dando lugar a la siguiente formulación utilizada en el análisis cluster:

$$\begin{aligned} \text{Minimize} \quad & SSE = \sum_{i=1}^n \sum_{k=1}^K w_{ik} [d^2(\mathbf{x}_i, \bar{\mathbf{x}}_k)]^2 \\ \text{subject to:} \quad & \sum_{k=1}^K w_{ik} = 1; \quad i = 1, \dots, n \\ & \sum_{i=1}^n w_{ik} \geq 1; \quad k = 1, \dots, K \\ & w_{ik} \in \{0, 1\} \end{aligned} \quad (6.10)$$

Notar que el problema (6.1) está formulado mediante un problema de maximización mientras que en el problema (6.10) se representa mediante uno de minimización. Esto es así porque  $J(P)$  representaba una medida genérica de *calidad* que cuando aumenta su valor mejor es el agrupamiento encontrado. El índice SSE operan en sentido contrario, cuando menor es su valor mejor es el agrupamiento.

El algoritmo de las  $K$ -medias fue desarrollado por [Mac67], este algoritmo puede ser considerado como un **procedimiento heurístico para resolver el problema de optimización** (6.10). La tabla 6.2 muestra un pseudocódigo del algoritmo de las  $K$ -medias. En la etapa de inicialización se eligen  $K$  individuos (registros, filas de la matriz de datos) como centroides. En la siguiente etapa se asigna cada dato al centroide más cercano respecto a la distancia euclídea. Esto produce una partición de los datos en grupos. Ahora se actualiza el centroide de cada nuevo grupo. Como posiblemente los grupos de la etapa anterior habrán cambiado se debe actualizar el centroides en cada grupo. Si algún centroide ha cambiado se repite el procedimiento. En otro caso se para el procedimiento ya que el algoritmo seguiría repitiendo indefinidamente los grupos obtenidos y por tanto sus centroides.

En el siguiente enlace  Enlace: <http://www.naftaliharris.com/blog/visualizing-k-means-clustering/> se puede visualizar el funcionamiento del algoritmo  $K$ -means para diferentes conjuntos de datos, mostrando que es adecuado para problemas en los que los grupos no estén engarzados uno dentro de otros.

**Tabla 6.2:** El algoritmo de las  $K$ -medias

---

**Paso 0. (Inicialización).** Elegir aleatoriamente  $k$  objetos (individuos)  $\{\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_K\}$  como los centroides iniciales para los  $K$  grupos.

**Paso 1. (Agrupación).** Calcular la distancia euclídea al cuadrado,  $[d^2(\mathbf{x}_i, \mathbf{r}_k)]^2$  entre el  $i$ -ésimo objeto y el  $k$ -ésimo centroide

$$[d^2(\mathbf{x}_i, \mathbf{r}_k)]^2 = \sum_{j=1}^P (x_{ij} - r_{kj})^2 \quad (6.11)$$

Asignar el individuo  $i$  al grupo  $k$  que minimice (6.11).

**Paso 2. (Actualización de centroides).** Calcular los nuevos centroides empleando la fórmula (6.8). Denotarlos  $\{\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_K\}$ .

**Paso 3. (Criterio de paro).** Chequear si ha habido algún cambio en los centroides. Si es que si, ir al paso 1. En caso contrario parar, el algoritmo ha encontrado los mejores  $K$ -grupos definidos por los centroides  $\{\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_K\}$ .

---

### 6.3.1. Funciones objetivo alternativas

El algoritmo de las  $K$ -medias se puede aplicar con distancias diferentes a la distancia euclídea, en dicho caso, esta nueva versión se interpretaría como un algoritmo heurístico para resolver el problema (6.10) asociado a dicha distancia. En esta versión del algoritmo de las  $K$ -medias asigna cada objeto al grupo que minimiza la nueva distancia en lugar de la distancia euclídea.

En la literatura se han propuesto funciones objetivo del problema (6.10) diferentes a SSE (asociada a la distancia euclídea), para describirlas, consideramos dos matrices adicionales: a) la matriz de pertenencia  $M = \{w_{ik}\}_{n \times K}$  donde  $w_{ik}$  vale 1 si el objeto  $i$  pertenece al cluster  $k$  y 0 en caso contrario y b) la matriz de representación de cluster que viene definida por:

$$R_{K \times P} = \begin{pmatrix} r_1^T \\ r_2^T \\ \vdots \\ r_K^T \end{pmatrix}$$

donde la fila  $k$ -ésima  $r_k^T$  es el vector centroide del cluster  $k$  sobre las  $P$  variables. La matriz de suma de cuadrados y productos cruzados intragrupos  $W$  se expresa como el siguiente producto de matrices  $W = (X - MR)^T \cdot (X - MR)$ , donde  $X$  es la matriz  $n \times P$  de datos,  $^T$  la transpuesta de una matriz y  $\cdot$  el producto de matrices.

Tabla 6.3: Funciones objetivos para el análisis cluster

| Referencia                    | Definición matemática   |
|-------------------------------|---|
| • Estadístico lambda de Wilk  | minimizar $ W / T $<br>minimizar $ W $                                      |
| • Criterio traza de Hotelling | maximizar el mayor valor propio $W^{-1}B$<br>maximizar $\text{tr}(W^{-1}B)$ |
| • [Win87]                     | minimizar $\sum_{k=1}^K  W_k ^{1/P}$  |
| • [SS71]                      | minimizar $\prod_{k=1}^K  W_k ^{n_k}$                                       |
| • [Sym81]                     | minimizar $\sum_{k=1}^K (n_k \log  W_k  - 2n_k \log n_k)$                   |

La función objetivo SSE se puede reescribir empleando estas matrices como:

$$SSE = \text{tr}(W) = \sum_{k=1}^K \text{tr}(W_k),$$

(6.12)

donde  $tr(\cdot)$  es la traza de una matriz, suma de los elementos de una diagonal, y  $W_k$  está definida por

$$W_k = \frac{1}{2n_k} \sum_{i \in C_k} \sum_{i^* \in C_k} (x_i - x_{i^*}) \cdot (x_i - x_{i^*})^T.$$

(6.13)

Aunque la SSE es la función objetivo más utilizada en el análisis cluster también han sido sugeridas otras alternativas. Para describirlas consideramos la siguiente relación:

$$T = W + B,$$

(6.14)

donde  $T$  es la matriz de suma de cuadrados y productos cruzados totales y  $B$  es la matriz de suma cuadrados y productos cruzados entre clusters. La tabla 6.3 recoge alguna de estas funciones objetivo que surgen de la adaptación de estadísticos clásicos del análisis estadístico multivariante al análisis cluster.

6.3.2. Un ejemplo ilustrativo

En esta sección analizaremos un ejemplo sencillo para ilustrar alguno de los anteriores problemas del análisis cluster. Generamos dos muestras aleatorias de una distribución normal bivalente,  $x_i \sim N_2(\mu_i, I_2)$  con  $i = 1, 2$ , donde  $\mu_1 = (1, 1)$ ,  $\mu_2 = (-1, -1)$  y la matriz de varianza-covarianza  $I_2$  es la matriz identidad de orden 2; el tamaño muestral es  $n = 20$  para cada muestra. Sobre estos  $n =$

40 datos vamos a realizar un análisis cluster. Por la construcción de los mismos, sabemos que existen dos clusters conteniendo cada uno de ellos a 20 individuos. En el siguiente código (Listado 6.1) se puede observar como se implementaría el algoritmo de las  $k$ -medias en Python para agrupar los datos generados por las dos muestras aleatorias anteriores.

Listado 6.1: Ejemplo del algoritmo de las  $K$ -medias en Python

```

1 import numpy as np
2 from sklearn.cluster import KMeans
3
4 # Generamos las dos muestras aleatorias (x1 y x2)
5 np.random.seed(10) # Fijamos la semilla para la reproducibilidad de los resultados
6 x1 = np.random.standard_normal((20, 2)) + np.ones((20, 2)).mean()
7 x2 = np.random.standard_normal((20, 2)) - np.ones((20, 2)).mean()
8 X = np.concatenate((x1, x2), axis=0) # Dataset conteniendo ambas muestras
9
10 # Aplicamos el algoritmo de las k-medias para obtener dos grupos
11 kmeans = KMeans(n_clusters=2, random_state=10)
12 kmeans.fit(X)
13 pred = kmeans.predict(X) # Obtenemos un vector de números naturales que indica el grupo
    al que pertenece cada elemento

```

La figura 6.2 muestra los datos obtenidos y los clusters que encuentra el algoritmo de las  $K$ -medias. La primera observación que se muestra es la dificultad de determinar, en base exclusivamente a la figura, el número  $K$  de cluster que contienen los datos, visualmente parecería que existen dos o tres grupos.



**Atención.** Notar que los resultados obtenidos por el lector serán diferentes en función de la semilla utilizada para la generación de los datos e incluso para la generación de los centroides en el primer paso del algoritmo de las  $K$ -medias.

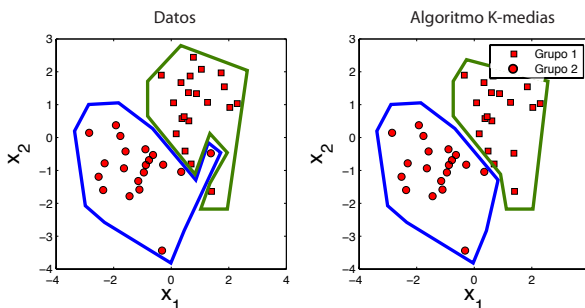


Figura 6.2: Ejemplo ilustrativo del algoritmo de las  $K$ -medias.

Tabla 6.4: Centroides obtenidos con el algoritmo de  $K$ -medias

|                      | Grupo 1         |                 | Grupo 2         |                 |
|----------------------|-----------------|-----------------|-----------------|-----------------|
|                      | $\bar{x}_{1,1}$ | $\bar{x}_{1,2}$ | $\bar{x}_{2,1}$ | $\bar{x}_{2,2}$ |
| Centroides obtenidos | 0.8939          | 0.8933          | -1.3173         | -0.9381         |
| Centroides datos     | 0.8696          | 0.9618          | -1.1824         | -0.9151         |

Se ve que el algoritmo de las  $K$ -medias es capaz, conociendo el valor verdadero de  $K = 2$ , de reconstruir los grupos originales. En el ejemplo mostrado en la figura anterior, únicamente un dato (*outlier*) del grupo 2 es clasificado como grupo 1. La tabla 6.4 muestra el valor del centroide de los grupos originales y de los grupos reconstruidos, observándose diferencias del mismo orden de magnitud que respecto a los centroides teóricos  $\mu_1$  y  $\mu_2$ .

Aunque los resultados son satisfactorios, este problema exhibe las dificultades teóricas anteriormente comentadas. Para comprobarlo aplicamos 10 veces el algoritmo de las  $K$ -medias, inicializando aleatoriamente los centroides de los cluster cada una de las veces. La tabla 6.5 muestra en la primera columna el número de iteraciones realizadas por el algoritmo y en la segunda columna el valor de la suma de los errores al cuadrado (SSE). Se observa que el algoritmo converge a tres soluciones con valores de SSE diferentes, lo que implica la existencia de mínimos locales, ya que en algunas repeticiones el algoritmo converge a valores SSE peores que en otras repeticiones.

6.3.3. Consideraciones importantes

Óptimos locales

Uno de los problemas más importantes en la resolución de los problemas de optimización es la existencia de los llamados *óptimos locales*. Estos puntos son los mejores candidatos relativos a un subconjunto próximo de soluciones factibles, de modo que pequeños cambios de las variables de decisión, no consiguen mejorar la función objetivo. Muchos de los algoritmos empleados en la teoría de optimización convergen a este tipo de soluciones. El análisis cluster también exhibe esta problemática ya que como se muestra en (6.1) se formula como un problema de optimización (discreta). La figura 6.3 ilustra el concepto de mínimo local en un problema de optimización continua. La solución  $\underline{x}$  es un mínimo local ya que si introdujésemos pequeños desplazamientos suyos en un cierto entorno o vecindad se incrementaría el valor de la suma de errores al cuadrado (SSE). La solución  $\mathbf{x}^*$  es el óptimo global porque tiene la imagen menor. Notar que  $\mathbf{x}^*$  también es un mínimo local. El algoritmo de las  $K$ -medias acaba convergiendo a mínimos locales.

A continuación formalizaremos el concepto de mínimo local en este tipo de problemas.

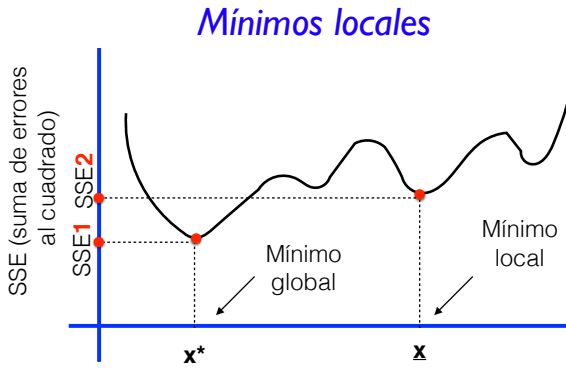


Figura 6.3: Mínimos locales.

**Definición 6 (Bola abierta)** Dado  $(S, d)$  un espacio métrico, llamaremos bola abierta de centro  $x$  y radio  $\delta$  al conjunto de puntos

$$B(x, \delta) = \{y \in S : d(x, y) < \delta\} \quad (6.15)$$

**Definición 7 (Mínimo local)** Diremos que  $\underline{x}$  es un mínimo local del problema

$$\underset{x \in X}{\text{minimizar}} f(x)$$

si existe una bola abierta  $B(\underline{x}, \delta)$ , con  $\delta > 0$  que cumple:

$$f(\underline{x}) \leq f(x) \text{ para todo } x \in B(\underline{x}, \delta) \cap X \quad (6.16)$$

Vamos a ilustrar estas definiciones en el contexto de análisis cluster para entender que representa un entorno o vecindad de una partición (solución)  $w$ . En el problema de análisis cluster, la región factible viene definida por:

$$X = \{w \in \mathcal{M}_{n \times K} : w \text{ cumpliendo las ecuaciones (6.2) – (6.4)}\} \quad (6.17)$$

donde  $\mathcal{M}_{n \times K}$  es el conjunto de todas la matrices de dimensiones  $n \times K$  ( $n$  filas asociadas a los objetos y  $K$  columnas asociadas a los grupos). Si consideramos la distancia de Manhattan  $d^1$  y  $\delta = 3$ , el entorno  $B(w, \delta) \cap X$  contiene a todas las matrices (particiones) que son como  $w$ , a excepción de dos de sus elementos (¿Por qué?). La diferencia entre una matriz  $w' \in B(w, 3)$  y  $w$  es que un elemento de  $w'$  contiene un 1 y ese mismo elemento para la matriz  $w$  es un 0 y al contrario con otro elemento. Este hecho se interpreta en que ambas particiones son idénticas a excepción de un solo elemento que es asignado en grupos diferentes. Para ilustrarlo, consideremos el siguiente ejemplo:

**Ejemplo 1 (Entornos en análisis cluster)** Queremos asignar tres objetos a dos cluster. Las variables de decisión  $w_{ik}$  toman el valor 1 si el objeto  $i$  es asignado al cluster  $k$ . Las variables de decisión  $w$  las representamos mediante matrices binarias, donde las filas están asociadas a los objetos y las columnas a los clusters. El conjunto de soluciones factibles es:

$$X = \left\{ \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} \right\}$$

Consideremos la solución

$$w = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix}$$

el entorno de soluciones (bola abierta) para  $\delta = 3$  es:

$$B(w, \delta) \cap X = \left\{ \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} \right\}$$

[GG92] proponen que después de aplicar el algoritmo de las  $K$ -medias se realice una inspección final entre todos los objetos y centroides. Si existe un objeto  $i$  dentro de un cluster  $k$  cumpliendo

$$\frac{n_k}{n_k - 1} d^2(x_i, \bar{x}_k) > \frac{n_{k^*}}{n_{k^*} + 1} d^2(x_i, \bar{x}_{k^*}) \quad (6.18)$$

donde  $\bar{x}_k$  y  $\bar{x}_{k^*}$  son respectivamente los centroides de los grupos  $k$  y  $k^*$ , entonces el elemento  $i$ -ésimo se reasignaría del cluster  $k$  al  $k^*$  y el valor de SSE se reduciría, [Spa80]. Si realizamos de manera reiterada este último paso, hasta no obtener ningún cambio, se garantiza que hemos finalizado, obteniendo un óptimo local. El algoritmo de las  $K$ -medias finaliza en un óptimo local y esta partición depende de los centroides elegidos en la etapa de inicialización. Además no existe un criterio para saber si el óptimo local encontrado es o no un óptimo global.

Para evitar los óptimos locales, algunos autores sugieren realizar varias veces el algoritmo de las  $K$ -medias y elegir aquella solución que sea mejor en términos de SSE. [Ste03] mostró que el número de óptimos locales para un conjunto de datos de tamaño moderado pueden alcanzar las centenas, conduciendo a que un número pequeño de reinicializaciones del algoritmo de las  $K$ -medias no conduce a un óptimo global. La existencia de mínimos locales es un problema más persistente de lo que en un principio se podría pensar. No obstante, se ha demostrado que el algoritmo de las  $K$ -medias usualmente exhibe buenas propiedades de reconstrucción de los clusters, [DDW02].

El algoritmo de las  $K$ -medias puede ser visto como un **algoritmo heurístico voraz** en el que en cada iteración hace una búsqueda en el entorno de la solución actual. Este tipo de algoritmos, por tanto, también finalizan en mínimos locales.

**Tabla 6.5:** Valor de SSE y número de iteraciones obtenidas con el algoritmo de las  $K$ -medias

| Repetición | Iteraciones $K$ – means | SSE          |
|------------|-------------------------|--------------|
| 1          | 3                       | 69,58        |
| 2          | 4                       | 68,25        |
| 3          | 5                       | 69,58        |
| 4          | 3                       | <b>67,93</b> |
| 5          | 4                       | 69,58        |
| 6          | 4                       | 69,58        |
| 7          | 2                       | <b>67,93</b> |
| 8          | 2                       | <b>67,93</b> |
| 9          | 3                       | <b>67,93</b> |
| 10         | 2                       | 68,25        |

### Métodos de inicialización del algoritmo de las $K$ -medias

La elección de los valores iniciales del algoritmo de las  $K$ -medias es crucial para evitar los numerosos óptimos locales del análisis cluster. Se han propuesto numerosas técnicas para determinar estos valores, que pasan desde tomar los  $K$  primeros datos de la base de datos, elegirlos aleatoriamente, incluso basados en la experiencia anterior de un experto. [Mil80] propone inicializar el algoritmo de las  $K$ -medias a partir de los resultados obtenidos de un análisis cluster jerárquico. [Ste03] particiona aleatoriamente los datos en  $K$  clusters y emplea los centroides como puntos iniciales. Se han propuesto métodos más sofisticados, uno de ellos se basa en la llamada densidad (número de datos que distan menos de una distancia  $d$  respecto a un determinado dato). Los puntos iniciales del algoritmo son elegidos atendiendo a la máxima densidad y dispersión entre ellos. Otros algoritmos son los de tipo *bootstrap* que se basan en muestreos sobre la base de datos.

Ilustremos esta problemática en el ejemplo anterior. Vamos a reinicializar el métodos desde varios puntos y comprobaremos que el valor SSE de las diferentes soluciones encontradas puede variar, lo que indica la existencia de mínimos locales. El resultado de ejecutar el script mostrado en el listado 6.2 se muestra en la tabla 6.5 y se observa que el algoritmo de las  $K$ –medias converge a tres soluciones (con diferentes valores de SSE). Esto motiva replicar el algoritmo varias veces, con la opción de Python `n_init` y elegir la mejor. Notar que el resultado obtenido por el lector será diferente dependiendo de la semilla utilizada para generar los datos e inicializar los centroides en el algoritmo de las  $K$ -medias.

#### Listado 6.2: Ejemplo de los efectos de las distintas inicializaciones de los centroides

```
1 kmeans = KMeans(n_clusters=2, n_init=10, random_state=100, verbose=1)
2 kmeans.fit(datos)
```



### Métodos para estimar el hiperparámetro $K$

Uno de los problemas mas complejos e importantes del análisis cluster es determinar el número de conglomerados. Se han desarrollado diversos procedimientos basados en esquemas algorítmicos, en métodos gráficos y aplicación de fórmulas. El esquema general es plantear un índice de calidad del agrupamiento y optimizar respecto al hiperparámetro  $K$ . Formalmente, se elige el valor  $K^*$  que resuelve el siguiente problema de optimización

$$\underset{K=2,3,\dots,K_{max}}{\text{maximizar}} \quad \Phi(K) \quad (6.19)$$

[MC85] realizaron un estudio de las diversas técnicas propuestas para determinar el valor de  $K$  sobre datos generados aleatoriamente y recomendaron entre todos los métodos, en el contexto del algoritmo de las  $K$ -medias,

$$\text{maximizar } \Phi(K) = \left\{ \frac{tr(B)}{K-1} \right\} / \left\{ \frac{tr(W)}{n-k} \right\}. \quad (6.20)$$

Este tema tiene un creciente interés, los autores [TWH01] propusieron el estadístico *Gap* donde

$$Gap(K) = \frac{1}{B} \sum_b \log(SSW_b(K)) - \log(SSW(K)) \quad (6.21)$$

siendo  $B$  el número de conjuntos de datos uniformemente distribuidos en el mismo rango que los datos originales y  $SSW_b(K)$  es la suma de cuadrados intracusters del  $b$ -ésimo conjunto de datos uniformes. Entonces el valor de  $K$  es elegido cumpliendo la ecuación:

$$Gap(K) \geq Gap(K+1) - s_{k+1} \quad (6.22)$$

donde  $s_k$  es una estimación de la desviación estándar de  $\log(SSW_b(K))$ .

Este problema es un caso particular de la validación de agrupamientos, [HBV01]. Muchos de los índices y métodos dependen del problema en consideración, lo que hace que no exista hoy por hoy un procedimiento que domine al resto en todos los problemas, [VCH10].

El método que emplearemos en la aplicación práctica se basa en el llamado **Ba-yesian Information Criterion (BIC)**. Este criterio compara el ajuste de diferentes modelos pero penalizando la complejidad del modelo (número de parámetros). Para entender la filosofía de este criterio considerar la siguiente situación. Si se planteara como criterio de calidad la suma de errores al cuadrado SSE, el valor elegido para el hiperparámetro sería  $K^* = n$ , donde  $n$  es el número de datos. Esto es, cada dato sería su propio grupo porque de esa forma los objetos (solo hay uno) dentro de cada grupo son completamente homogéneos y  $SSE = 0$ . Esta solución con los clusters excelentemente conformados tiene como desventaja que debemos estimar exacta-

mente  $n$  centroides (vectores de  $P$  dimensiones). Tendríamos que estimar muchos parámetros  $n * P$ . El criterio BIC quiere sopesar ambos aspectos, quiere introducir un nuevo grupo (con la estimación de su centroide), si el decrecimiento del SSE es significativo.

El valor del índice BIC para una solución del problema cluster con  $K$  grupos se define:

$$BIC(K) = -2 \sum_{k=1}^K \xi(k) + 2 \cdot K \cdot P \cdot \ln(n) \quad (6.23)$$

donde:

$P$  : número de variables

$K$  : número de clusters

$n$  : número de datos

$\xi(k)$  : es una medida de la varianza estimada dentro del grupo  $k$  definida por

$$\xi(k) := -n_k \left( \sum_{j=1}^P \frac{1}{2} \ln(\hat{\sigma}_j^2 + \hat{\sigma}_{kj}^2) \right) \quad (6.24)$$

donde  $n_k$  es el número de elementos en el cluster  $k$ ,  $\hat{\sigma}_j^2$  y  $\hat{\sigma}_{kj}^2$  son respectivamente una estimación de la varianza del atributo  $j$  en toda la muestra y dentro del grupo  $k$ .

El criterio BIC consta de dos sumandos. El primero está asociado a la variabilidad dentro de los grupos y el segundo es una penalización por los parámetros estimados. Si se aumenta el valor de  $K$  el primer sumando decrece pero el segundo sumando aumenta. Se iría aumentando el valor de  $K$  mientras que el incremento provocado por los parámetros sea absorbido por la mejor conformación de los clusters. O dicho matemáticamente se tomaría como  $K^*$  el valor que minimice el BIC.

En el código mostrado en el listado 6.3 se implementa en Python el cálculo del índice BIC para análisis cluster.

#### Listado 6.3: Implementación del índice BIC en Python

```

1 def BIC(K, grupos, X):
2     """
3     K: Número de grupos (clusters)
4     grupos: Vector que contiene los grupos de los datos
5     X: Matriz de datos
6     """
7     N = X.shape[0] # Número de datos
8     P = X.shape[1] # Número de variables
9     xi = np.zeros((1,K)) # Vector xi
10

```

**Tabla 6.6:** Algoritmo de [XC07] para determinar el valor de  $K$ 

- 
- **Etapla 1.** Calcular el valor de  $BIC(K)$  mientras que se cumpla  $BIC(K + 1) < BIC(K)$ , esto es, que vaya decreciendo su valor. Denotar por  $\hat{K}$  este último valor que cumple  $BIC(\hat{K} + 1) \geq BIC(\hat{K})$
  - **Etapla 2.** Calcular el cociente de cambio

$$R(K) = \frac{S(K-1)}{S(K)} \text{ para } K = 2, \dots, \hat{K} \quad (6.25)$$

donde  $S(K) = L(K) - L(K+1)$  y para cada solución de  $K$  grupos se calcula  $L(K) = \sum_{k=1}^K \xi(k)$ .

- **Etapla 3** (Elección del número de clusters.) Sea  $K_1$  y  $K_2$  los números de clusters donde se alcanza el mayor cociente de cambio, esto es:

$$K_1 = \text{Arg maximizar } R(K)_{K \in \{2, \dots, \hat{K}\}} \quad (6.26)$$

$$K_2 = \text{Arg maximizar } R(K)_{K \in \{2, \dots, \hat{K}\} - K_1} \quad (6.27)$$

$$\begin{aligned} \text{Si } \frac{R(K_1)}{R(K_2)} > 1,15 &\Rightarrow K^* = K_1 \\ \text{En otro caso} &\Rightarrow K^* = \max\{K_1, K_2\} \end{aligned}$$


---

```

11 # Calculamos el sumario de xi en la fórmula
12 for k in range(0, K):
13     suma = 0
14     for j in range(0, P):
15         sigma = np.square(np.std(X[:, j]))
16         sigma_j = np.square(np.std(X[grupos==k, j]))
17         suma += 0.5*np.log(sigma + sigma_j)
18
19     n_k = sum(grupos==k) # Número de elementos en el grupo k
20     xi[0, k] = -n_k*suma
21
22 bic = -2*np.sum(xi) + 2*K*P*np.log(N)
23 return bic

```

---

Un refinamiento de la minimización del índice BIC es el método propuesto por [XC07]. Este algoritmo es es una heurística basada en el BIC y se muestra en la tabla 6.6.

En el código mostrado en el listado 6.4 se generan  $n$  datos de  $K = 4$  grupos. Con este código se representa gráficamente los datos generados y la evaluación del valor del índice BIC en función del número de cluster realizado. Se observa que inicialmente el valor de BIC es decreciente hasta que a partir de un cierto valor,  $K^*$  elegido, empieza a crecer. Es interesante para entender el funcionamiento del BIC emplear varios valores de  $n$  y  $\sigma$  para ver cómo influye en la detección del número correcto de clusters.

Listado 6.4: Ejemplo de la elección del hiperparámetro  $K$  usando el índice BIC

```

1 # Generamos datos simulados (K=4)
2 sigma = 0.5 # Desviación típica de la variable normal empleada en la generación de datos
3 n = 20 # Número de datos por grupo
4 Kmax = 10 # Número máximo de clusters a analizar
5
6 np.random.seed(100) # Fijamos la semilla para la reproducibilidad de los resultados
7 x1 = sigma*np.random.standard_normal((n, 2)) + np.ones((n, 2))
8 x2 = sigma*np.random.standard_normal((n, 2)) - np.ones((n, 2))
9 x3 = sigma*np.random.standard_normal((n, 2)) + np.concatenate((np.ones((n, 1)), -np.ones
    ((n, 1))), axis=1)
10 x4 = sigma*np.random.standard_normal((n, 2)) + np.concatenate((-np.ones((n, 1)), np.ones
    ((n, 1))), axis=1)
11 X = np.concatenate((x1, x2, x3, x4), axis=0)
12
13 # Dibujamos los datos simulados, elegiremos un color para cada muestra
14 color = ['b','r','g','k']
15 forma = ['o','s','*','v']
16 for k in range(0, 4):
17     plt.plot(X[k*n:(k+1)*n, 0], X[k*n:(k+1)*n, 1], color[k]+forma[k])
18 plt.title("Datos generados", fontsize=16)
19 plt.xlabel(r"$x_1$", fontsize=14)
20 plt.ylabel(r"$x_2$", fontsize=14)
21 plt.show()
22
23 # Calculo del BIC
24 BIC_array = []
25 for k in range(2, Kmax):
26     kmeans = KMeans(n_clusters=k, n_init=10, random_state=100)
27     grupos = kmeans.fit_predict(X)
28     BIC_array.append(BIC(k, grupos, X))
29
30 # Dibujamos el BIC obtenido para cada valor de k
31 plt.plot(np.arange(2, Kmax), BIC_array, "ko-")
32 plt.title("Valor del BIC en función de K", fontsize=16)
33 plt.xlabel("K", fontsize=14)
34 plt.ylabel("BIC(K)", fontsize=14)
35 plt.show()

```

### Detección de observaciones influyentes en análisis cluster

El análisis cluster puede estar fuertemente influido por unas determinadas observaciones (las llamadas **observaciones influyentes**) que poseen valores extremos en determinadas variables. Estas observaciones, por estar muy alejadas del resto, **conformarán su propio grupo**. En algunos problemas tienen un **significado especial**, pueden representar incidentes en una red de tráfico o intrusiones en un sistema de videovigilancia y por ello es importante detectarlas, y en su caso eliminarlas del análisis.

El método empleado es la **técnica de jackknife** descrita en el capítulo 2. Si es usada en combinación con el algoritmo de  $K$ -means el índice adecuado es tomar como  $\psi_i$  la suma de los errores al cuadrado  $SSE_i$ . Este método realizaría  $n$  análisis clusters, eliminando en cada una de las veces un dato. Finalmente se detectaría si la muestra  $\{\psi_i\}$  contiene outliers basándonos por ejemplo en el método de bandas:

$$\text{Si } |SSE_i - \mu_{SSE}| > k * \sigma_{SSE} \Rightarrow i \text{ es observación inluyente} \quad (6.28)$$

donde  $\mu_{SSE}, \sigma_{SSE}$  son respectivamente la media y desviación típica de la muestra  $\{SSE_i\}$ . El valor  $k$  es el umbral elegido.

Listado 6.5: Ejemplo de la eliminación de outliers usando jackknife en K-medias

```

1 # Lo primero que haremos es añadir dos observaciones influyentes a un dataset.
2 # Por ejemplo, vamos a modificar las observaciones nº 18 y 38 de X
3 # (Posiciones 17 y 37 si empezamos a contar en 0)
4 X[17,:] = [-3,3]
5 X[37,:] = [3,-3]
6
7 N = X.shape[0] # Número de observaciones
8 K = 2 # Número de clusters
9 SSE = []
10 for i in range(0, N):
11     X_sin_i = np.delete(X, i, axis=0) # Eliminamos la observación i
12     # Aplicamos K-medias a X_sin_i y obtenemos el índice SSE
13     kmeans = KMeans(n_clusters=K, n_init=10, random_state=100).fit(X_sin_i)
14     SSE.append(kmeans.inertia_)
15
16 # Detección analítica de outliers
17 sigma = np.std(SSE) # Desviación típica
18 mu = np.mean(SSE) # Media
19 umbral = 2 # Umbral: 2 para distribuciones normales y 3 para cualquier otra distribución
20 outliers = []
21 for i in range(0, N):
22     if np.abs(SSE[i]-mu) > umbral*sigma:
23         outliers.append(i)
24 print(outliers)

```

Partiendo de los datos generados en el listado 6.1 y tras ejecutar el código de ejemplo mostrado en el listado 6.5 obtendríamos que las observaciones influyentes son las que ocupan las posiciones número 17 y 37 (empezando a contar en cero).

## 6.4. Fuzzy C-means (FCM)

Si observamos el problema (6.10) vemos que las variables  $w_{ik}$  sólo pueden tomar los valores 0/1, esto es, un objeto  $i$  pertenece o no al cluster  $k$ . Se puede interpretar el significado de estas variables como **grado de pertenecía**, en este caso estas variables toman valores en el intervalo  $[0, 1]$ , es decir valores  $0 \leq w_{ik} \leq 1$ . El valor 1 representa que seguro que pertenece al cluster y el valor 0 que seguro que no pertenece. Un individuo pertenece de alguna manera a todos los grupos pero en diferente grado, determinado por las variables  $w_{ik}$ .

Por tanto este problema deja de ser de optimización discreta (sobre una región factible de soluciones finita) y pasa a ser un problema de optimización no lineal continua (el conjunto de soluciones factible es infinito) que tienen algoritmos de resolución basados en el gradiente. De hecho el algoritmo *fuzzy c-means* es simplemente uno de estos algoritmos de resolución.

La función objetivo en este caso es

$$J_m = \sum_{i=1}^n \sum_{k=1}^K w_{ik}^m [d^2(\mathbf{x}_i, \bar{\mathbf{x}}_k)]^2 \quad \text{con } 1 \leq m < \infty \quad (6.29)$$

donde  $d^2(\cdot, \cdot)$  es la distancia euclídea y por tanto el problema fuzzy de análisis cluster se formula:

$$\begin{aligned} \text{Minimize} \quad & J_m = \sum_{i=1}^n \sum_{k=1}^K w_{ik}^m [d^2(\mathbf{x}_i, \bar{\mathbf{x}}_k)]^2 \\ \text{subject to:} \quad & \sum_{k=1}^K w_{ik} = 1; \quad i = 1, \dots, n \\ & \sum_{i=1}^n w_{ik} > 0; \quad k = 1, \dots, K \\ & 0 \leq w_{ik} \leq 1; \quad i = 1, \dots, n, k = 1, \dots, K. \end{aligned} \quad (6.30)$$

donde  $m$  es un **hiperparámetro** del modelo y toma valores en  $m \in [1, +\infty)$ . Este hiperparámetro controla la cantidad de solapamiento entre grupos. Si su conjunto de datos es disperso, con mucho solapamiento entre los clusters potenciales, los centros de los clusters calculados pueden estar muy cerca unos de otros. En este caso, cada individuo tiene aproximadamente el mismo grado de pertenencia a todos los clusters. Esa situación se modela con valores altos de  $m$ . Un valor usual por defecto es  $m = 2$ . Para mejorar los resultados del agrupamiento, se puede disminuir este valor, limitando así la cantidad de solapamiento entre los grupos.

Listado 6.6: Ejemplo del algoritmo Fuzzy C-means en Python

```
1 from skfuzzy.cluster import cmeans
2 import matplotlib.pyplot as plt
3
4 K = 2 # Número de clusters
5 m = 2 # Parámetro de FCM, 2 es el defecto
6 tolerancia = 1e-5 # Tolerancia (criterio de parada)
```

Tabla 6.7: El algoritmo FCM

**Paso 0. (Inicialización).** Elegir una matriz de pertenencia inicial  $\mathbf{U}^{(0)} = (w_{ik})$

**Paso 1. (Actualización de centroides).** Calcular la matriz de centroides  $\mathbf{C}^{(t)}$  en la  $t$ -ésima iteración a partir de la matriz  $\mathbf{U}^{(t)}$ :

$$\bar{\mathbf{x}}_k = \frac{\sum_{i=1}^n w_{ik}^m \mathbf{x}_i}{\sum_{i=1}^n w_{ik}^m} \quad (6.31)$$

**Paso 2. (Actualización de los pesos).** Calcular  $\mathbf{U}^{t+1}$  a partir de  $\mathbf{U}^t$  mediante la expresión:

$$w_{ik} = \frac{1}{\sum_{s=1}^K \left( \frac{d^2(\mathbf{x}_i, \bar{\mathbf{x}}_k)}{d^2(\mathbf{x}_i, \bar{\mathbf{x}}_s)} \right)^{\frac{2}{m-1}}} \quad (6.32)$$

**Paso 3. (Criterio de paro).** Si  $d(\mathbf{U}^{(t+1)}, \mathbf{U}^{(t)}) < \text{Tolerancia}$  parar. En caso contrario  $t = t + 1$  e ir al Paso 1.

```

7 maxiter = 100 # Número máximo de iteraciones
8
9 cntr, u, u0, d, jm, p, fpc = cmeans(X.T, K, m, tolerancia, maxiter, seed=100)
10 # Parámetros de salida:
11 # - cntr: Centroides
12 # - u: Matriz de pertenencia de los individuos a los clusters
13 # - u0: Matriz u en la iteración 0
14 # - d: Matriz de distancias Euclideas en la última iteración
15 # - jm: Historico del valor de la función objetivo
16 # - p: Número de iteraciones
17 # - fpc: Coeficiente de partición difusa final
18
19 # Obtener a que grupo pertenece cada observación
20 grupos = np.argmax(U, axis=0) # Devuelve los índices del valor máximo sobre el eje 0
    (filas)
21 # Obtener el grado de pertenencia al grupo con mayor pertenencia
22 maxU = np.amax(U, axis=0) # Devuelve el valor máximo sobre el eje 0 (filas)
23
24 # Representar las observaciones asociadas con cada grupo (cluster)
25 for k in range(0, K):
26     plt.scatter(X[grupos==k, 0], X[grupos==k, 1], label="Grupo {}".format(k))
27
28 # Representar junto a cada observación el grado de pertenencia al grupo
29 for i, maxU_i in enumerate(np.round(maxU, 2)):
30     plt.annotate(maxU_i, (X[i,0], X[i,1]))
31
32 # Dibujamos los centroides
33 plt.plot(cntr[:, 0], cntr[:, 1], 'k*', markersize=14, label="Centroides")
34
35 plt.title("Algoritmo Fuzzy c-means", fontsize=14)
36 plt.xlabel(r"$x_1$", fontsize=14)

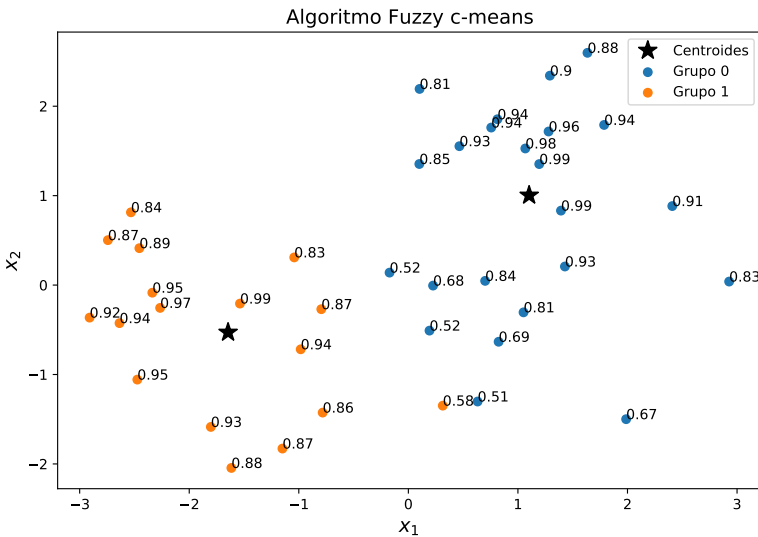
```

```

37 plt.ylabel(r"$x_2$", fontsize=14)
38 plt.legend()
39 plt.show()

```

La figura 6.4 muestra los resultados de ejecutar el listado 6.6. Con este script se realiza un análisis cluster con el algoritmo FCM, se dibujan los grupos con diferentes colores y se muestra el grado de pertenencia de cada elemento al cluster asignado. El algoritmo detecta diferentes niveles de pertenencia, variando desde valores alrededor de 0,5 cerca de la frontera entre clusters hasta valores superiores a 0,9 cerca de los centroides. Se ha utilizado la implementación de C-means del paquete skfuzzy. Notar que esta implementación considera que los datos de entrada son una matriz de tamaño (S, N), donde S es el número de características de los datos y N es el número de observaciones. Por lo tanto, se toma la traspuesta de la matriz de datos  $X$ .



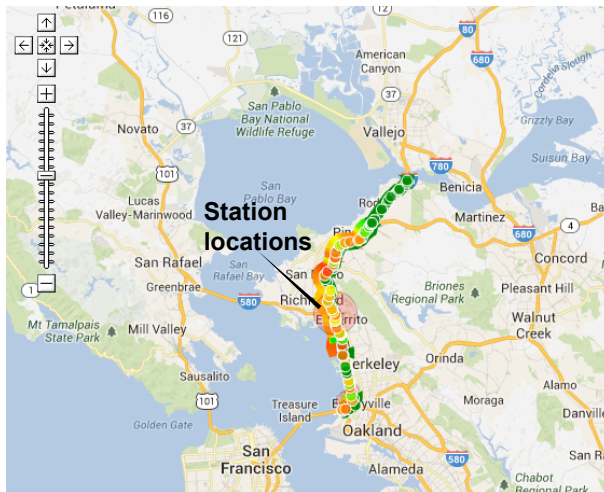
**Figura 6.4:** Ejemplo ilustrativo del algoritmo FCM.



## 6.5. Un caso práctico: determinación de patrones de tráfico

### 6.5.1. Introducción

En esta sección se analizarán datos de tráfico recogidos con el sistema PeMS (California Freeway Performance Measurement System). Este sistema cuenta con 25000 sensores individuales. En este estudio se consideran exclusivamente dos sensores y 100 días del año 2013 mostrados en la figura 6.5. El objetivo es establecer los patrones de tráfico presentes en la zona de estudio.



**Figura 6.5:** Localización de los dos sensores.

### 6.5.2. Datos usados para este problema

Los tres parámetros esenciales que caracterizan el tráfico son: **flujo**, **densidad** y **velocidad**. El flujo es el número de vehículo que pasan por un punto de la vía por unidad de tiempo, se mide en vehículos/hora. La densidad se mide en número de vehículos por unidad de longitud, por ejemplo número de vehículos por kilómetro. Finalmente con respecto a la velocidad, como los datos corresponden a una zona en Estados Unidos, la unidad de longitud es la milla (mi), midiéndose la velocidad en millas por hora. En lugar de la densidad los sensores registran la ocupación de la vía, es decir, la proporción de la vía ocupada por vehículos, la cual se registra en tantos por uno. Los sensores realizan un registro cada 5 minutos. Por lo tanto, se tienen 288 instantes al día.

La figura 6.6 muestra los datos por tipo y sensor. Hemos simplificado este problema, considerado solamente las observaciones de velocidad. Estos datos se han preprocesado en una matriz  $X$ , la cual contiene los datos necesarios para realizar el análisis cluster. Las filas están asociadas a los días y las columnas a todas las mediciones realizadas en un día (en cada instante de tiempo) por ambos sensores, primero los 288 instantes del primer sensor y después los 288 instantes del segundo sensor. Por lo tanto, la matriz resultante  $X$  tiene 100 filas (días) y  $288 \times 2$  sensores = 576 columnas.

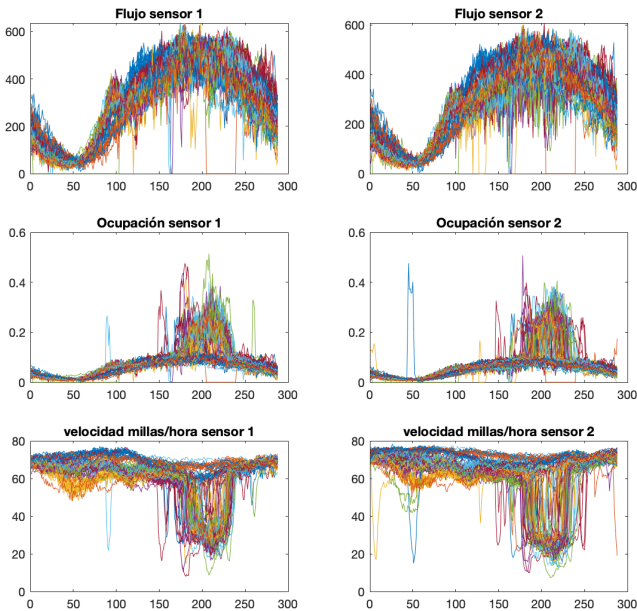


Figura 6.6: Representación gráfica de los datos obtenidos por tipo de parámetro y sensor.

6.5.3. Detección del número de clusters

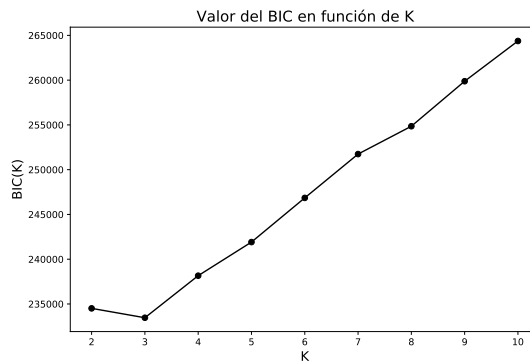
Por tratarse de un caso real el número de clusters  $K$  es desconocido. Vamos a aplicar el criterio basado en el BIC para su cálculo, para ello haremos uso de la función BIC (listado 6.3). El código mostrado en el listado 6.7 hace esta tarea y observamos la gráfica resultante 6.7 observando que el BIC decrece hasta  $K = 3$  y luego crece. Por lo tanto, este será el valor elegido.

Listado 6.7: Determinación del hiperparámetro  $K$  usando el índice BIC

```
1 Kmax = 10
2 BIC_array = []
3 for k in range(2, Kmax+1):
```

## 6.5. Un caso práctico: determinación de patrones de tráfico

```
4 kmeans = KMeans(n_clusters=k, n_init=15, random_state=100)
5 grupos = kmeans.fit_predict(X)
6 BIC_array.append(BIC(k, grupos, X))
7
8 # Dibujamos el BIC obtenido para cada valor de k
9 plt.plot(np.arange(2, Kmax+1), BIC_array, "ko-")
10 plt.title("Valor del BIC en función de K", fontsize=16)
11 plt.xlabel("K", fontsize=14)
12 plt.ylabel("BIC(K)", fontsize=14)
13 plt.show()
```



**Figura 6.7:** Resultado de la ejecución de casoestudioDeteccionK .

### 6.5.4. Detección de outliers (anomalías)

Vamos a aplicar el método de Jackknife para detectar las observaciones influyentes. El listado 6.8 detecta los outliers y el listado 6.9 los elimina del análisis y los representa visualmente. Notar que si el número de réplicas en el número de veces que se realiza el algoritmo de las k-medias es pequeño el procedimiento detecta diferentes conjunto de outliers ya que llega a diferentes soluciones. En este caso en la figura 6.8 detectamos 3 observaciones influyentes. La representación visual de estos outliers está dada en la gráfica 6.9. Cuando detectemos los patrones podemos volver a analizar los outliers.

#### Listado 6.8: Determinación de las observaciones influyentes mediante el método Jackknife

```
1 # Detección de outliers
2 N = X.shape[0] # Número de observaciones
3 K = 3 # Número de clusters
4 SSE = []
5 for i in range(0, N):
6     X_sin_i = np.delete(X, i, axis=0)
7     kmeans = KMeans(n_clusters=K, n_init=30, random_state=100).fit(X_sin_i)
```

```

8     SSE.append(kmeans.inertia_)
9
10 # Detección visual de outliers
11 plt.plot(np.arange(0, N), SSE, "ko-")
12 plt.title("Valor del índice SSE eliminando el dato i", fontsize=16)
13 plt.xlabel("Dato i", fontsize=14)
14 plt.ylabel("SSE", fontsize=14)
15 plt.show()
16
17 # Detección analítica de outliers
18 sigma = np.std(SSE) # Desviación típica de SSE
19 mu = np.mean(SSE) # Media
20 umbral = 2
21 outliers = []
22 for i in range(0, N):
23     if np.abs(SSE[i]-mu) > umbral*sigma:
24         outliers.append(i)
25 print(outliers)

```

#### Listado 6.9: Representación y eliminación de las observaciones influyentes

```

1 fig, axs = plt.subplots(len(outliers), 2)
2 fig.set_size_inches(10, 9)
3 axs = axs.ravel()
4 for i in range(len(outliers)):
5     for j in range(2): # Para cada uno de los dos sensores
6         ini_seg_sensor = X.shape[1]//2
7         axs[i*2 + j].plot(np.arange(0, ini_seg_sensor), X[outliers[i], j*ini_seg_sensor:(j+1)
8                             *ini_seg_sensor])
9         axs[i*2 + j].set_title("Observación {}".format(outliers[i]), fontsize=14)
10        axs[i*2 + j].set_xlabel("Intervalo del día", fontsize=14)
11        axs[i*2 + j].set_ylabel("vel. (mi./h)", fontsize=14)
12 plt.tight_layout()
13 # Eliminación de los outliers
14 X_new = np.delete(X, outliers, axis=0)

```

### 6.5.5. Determinación de patrones y estudio

Finalmente calculamos los patrones mediante el algoritmo  $K$ -means, los representamos gráficamente y calculamos los intervalos de congestión en cada patrón. Fijamos como existencia de congestión que la velocidad se reduzca por debajo de los 50 millas a la hora. En la gráfica 6.10 muestra tres patrones. El **Patrón 3** es una situación de tráfico sin congestión, el **Patrón 2** es el más congestionado y el **Patrón 1** es una congestión moderada. En la tabla 6.8 se calcula el número de días que se ha repetido cada patrón. Notar que los sensores recolectan datos de 100 días. Dado que una semana tiene 7 días, tenemos datos de 14 semanas. Estas cantidades pueden indicar que el **Patrón 3** corresponde a sábados y domingos, ya que corresponde con 14 semanas  $\times$  2 días por semana e indica que no hay congestión; el **Patrón 2** puede indicar días laborables entre semana; finalmente el **Patrón 1** puede indicar viernes y otros días laborales con menos afluencia de tráfico.

6.5. Un caso práctico: determinación de patrones de tráfico

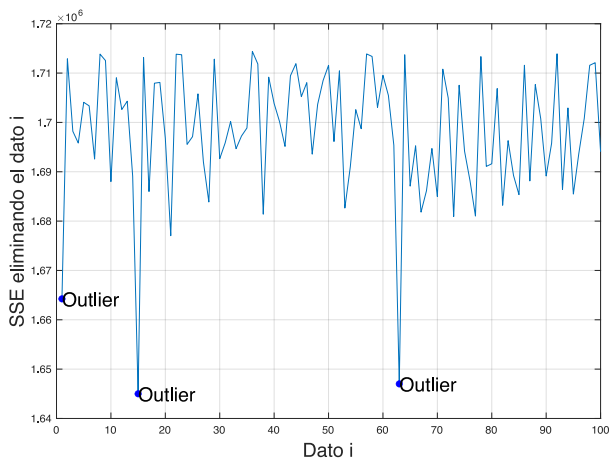


Figura 6.8: Resultado de la ejecución del listado 6.8.

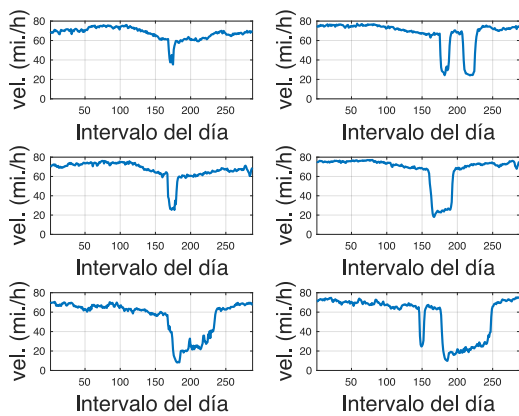


Figura 6.9: Resultado de la ejecución del listado 6.9.

Además, se ha calculado los intervalos de congestión definidos como los instantes de tiempo que la velocidad desciende de las 50 millas/hora. Se observa que este efecto de la congestión aparece cuando la gente vuelve del trabajo. Se ha medido la velocidad media en dichos intervalos observando que el patrón **Patrón 3** no aparece la congestión y en los otros dos está presente con diferente intensidad.

Listado 6.10: Representación gráfica de los patrones encontrados

```
1 fig, axs = plt.subplots(len(outliers), 2)
2 fig.set_size_inches(10, 9)
```

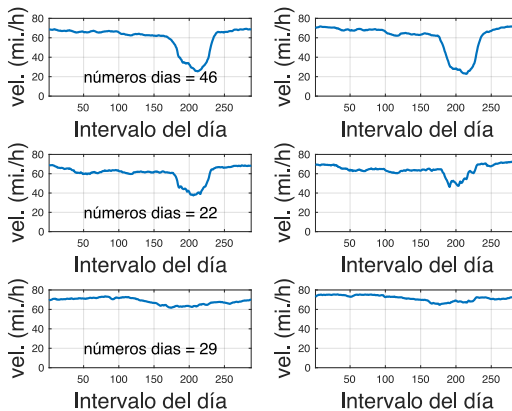
**Tabla 6.8:** Intervalos de congestión y velocidad media para patrones

| Patrón          | números días | Sensor 1        |                 |              | Sensor 2        |                 |              |
|-----------------|--------------|-----------------|-----------------|--------------|-----------------|-----------------|--------------|
|                 |              | Velocidad media | instante inicio | instante fin | Velocidad media | instante inicio | instante fin |
| <b>Patrón 1</b> | 20           | 43.48           | 15.25           | 18.58        | 42.50           | 15.75           | 16.92        |
| <b>Patrón 2</b> | 49           | 33.45           | 15.17           | 19.08        | 33.15           | 15.25           | 19.08        |
| <b>Patrón 3</b> | 28           | > 50            | -               | -            | > 50            | -               | -            |

```

3  axs = axs.ravel()
4  for i in range(K):
5      for j in range(2): # Para cada uno de los dos sensores
6          ini_seg_sensor = X.shape[1]//2
7          axs[i*2 + j].set_ylim([0, 80]) # Establecer limites del eje y
8          axs[i*2 + j].plot(np.arange(0, ini_seg_sensor), centroides[i, j*ini_seg_sensor:(j+1)*
              ini_seg_sensor])
9          axs[i*2 + j].set_title("Patrón {}".format(i), fontsize=14)
10         axs[i*2 + j].set_xlabel("Intervalo del día", fontsize=14)
11         axs[i*2 + j].set_ylabel("vel. (mi./h)", fontsize=14)
12
13 plt.tight_layout()

```

**Figura 6.10:** Resultado de la ejecución del listado 6.10.

### 6.5.6. Otras consideraciones

El estado del tráfico viene definido por la combinación de los tres parámetros y éstos se relacionan a través de la ecuación fundamental

$$q = v \cdot k \quad (6.33)$$

## 6.5. Un caso práctico: determinación de patrones de tráfico

---

donde  $q$  es el flujo,  $v$  la velocidad y  $k$  la densidad. Esta **relación fundamental** pone de manifiesto que puede existir dos estados diferenciados de tráfico con el mismo valor en un único de éstos parámetros. Por ejemplo, se puede circular a una velocidad elevada pero sin embargo la densidad en la vía puede variar de estar vacía a un nivel significativo de tráfico. O por ejemplo, existir dos situaciones donde el flujo de vehículos fuese similar pero en una situación hubiera congestión con tráfico lento y en otro hubiera una menor densidad pero circularan a mayor velocidad. Esta consideraciones nos llevaría a introducir simultáneamente varios parámetros de tráfico en el análisis. Esta combinación de magnitudes conduciría a un escalamiento de la variable ya que si consideramos por ejemplo flujo (medido en miles de vehículos) con ocupación (medidos en tantos por uno, proporciones) estas últimas variables serían irrelevantes en el análisis.





# Bibliografía

---

- [Aro50] N. Aroszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404, 1950.
- [BEF84] J.C. Bezdek, R. Ehrlich, and W. Full. Fcm: The fuzzy c-means clustering algorithm. *Computers and Geosciences*, 10(2-3):191–203, 1984.
- [Cox57] D. R. Cox. Note on grouping. *Journal of the American Statistical Association*, 52:543–547, 1957.
- [DDW02] E. Dimitriadou, S. Dolnicar, and A. Weingessel. An examination of indexes for determining the number of clusters in binary data sets. *Psychometrika*, 67(1)(1):137–160, 2002.
- [EKSX96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD’96, page 226–231. AAAI Press, 1996.
- [GG92] A. Gersho and R.M. Grey. *Vector quantization and signal compression*. Kuwer Academic, Boston, 1992.
- [GH10] J. González-Hernández. *Representing Functional Data in Reproducing Kernel Hilbert Spaces with Applications to Clustering, Classification and Time Series Problems*. PhD thesis, Department of Statistics, Universidad Carlos III, Getafe, Madrid, 2010.
- [HBV01] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. On clustering validation techniques. *J. Intell. Inf. Syst.*, 17(2-3):107–145, December 2001.

- [HPK11] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.
- [JWHT13] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.
- [KR90] L. Kaufman and P. Rousseeuw. *Finding groups in data: An introduction to cluster*. Wiley, New York, 1990.
- [LT18] Z. Li and Y. Tang. Comparative density peaks clustering. *Expert Systems with Applications*, 95:236–247, 2018.
- [LWY18] R. Liu, H. Wang, and X. Yu. Shared-nearest-neighbor-based clustering by fast search and find of density peaks. *Information Sciences*, 450:200–226, 2018.
- [Mac67] J. MacQueen. Some methods of classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1. L. M. Le Cam and J. Neyman (Eds.)*. Berkeley, CA: University of California Press, pages 281–297, 1967.
- [MC85] G.W. Milligan and M.C. Cooper. An examination of procedures for determining the numbers of clusters in a data set. *Psychometrika*, 50:159–179, 1985.
- [MC12] F. Murtagh and P. Contreras. Algorithms for hierarchical clustering: An overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(1):86–97, 2012.
- [Mer09] J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London. Series A*, 209:415–446, 1909.
- [Mil80] G.W. Milligan. An examination of the effect of six types of error perturbation on fifteen clustering algorithms. *Psychometrika*, (45):181–204, 1980.
- [RL14] A. Rodríguez and A. Laio. Clustering by fast search and find of density peaks. *Science*, 344(6191):1492–1496, 2014.
- [Spa80] *Cluster analysis algorithms for data reduction and classification of objects*. Wiley, New York, 1980.
- [SS63] R. Sokal and P.H.A. Sneath. *Principles of numerical taxonomy*. San Francisco: W.H. Freeman., 1963.
- [SS71] A.J. Scott and M.J. Symons. Clustering methods based on likelihood ratio criteria. *Biometrics*, (27):387–398, 1971.
- [Ste03] D. Steinley. Local optima in k-means clustering: What you don’t know may hurt you. *Psychological Methods*, 8(3):294–304, 2003.

- [Sym81] M.J. Symons. Clustering criteria and multivariate normal mixtures. *Biometrics*, (37):35–43, 1981.
- [TSK16] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to data mining*. Pearson Education India, 2016.
- [TWH01] R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society B*, (63):411–423, 2001.
- [VCH10] L. Vendramin, R.J.G.B. Campello, and E.R. Hruschka. Relative clustering validity criteria: A comparative overview. *Statistical Analysis and Data Mining*, 3(4):209–235, 2010.
- [Win87] M.P. Windham. Parameter modification for clustering criteria. *Journal of Classification*, (4):191–214, 1987.
- [XC07] J. Xia and M. Chen. A nested clustering technique for freeway operating condition classification. *Computer-Aided Civil and Infrastructure Engineering*, 22(6):430–437, 2007.