

# APRENDIZAJE NO SUPERVISADO

Profesor: Félix José Fuentes Hurtado

MÁSTER EN INTELIGENCIA ARTIFICIAL

Módulo IV. Aprendizaje autónomo

**viu**

**Universidad  
Internacional  
de Valencia**

Este material es de uso exclusivo para los alumnos de la Universidad Internacional de Valencia. No está permitida la reproducción total o parcial de su contenido ni su tratamiento por cualquier método por aquellas personas que no acrediten su relación con la Universidad Internacional de Valencia, sin autorización expresa de la misma.

**Edita**

Universidad Internacional de Valencia

Máster en  
**Inteligencia Artificial**

---

**Aprendizaje no supervisado**

Módulo IV. Aprendizaje autónomo  
6 ECTS

---

**Autor:** Jerónimo Hernández-González  
**Profesor:** Félix José Fuentes Hurtado

## Leyendas

---



Enlace de interés



Ejemplo



Importante



**abc** Los términos resaltados a lo largo del contenido en color **naranja** se recogen en el apartado **GLOSARIO**.

---

# Índice

<b>CAPÍTULO 1. INTRODUCCIÓN A LA MINERÍA DE DATOS Y EL APRENDIZAJE AUTOMÁTICO.....</b>	<b>7</b>
1.1. Aprendizaje automático supervisado .....	9
1.1.1. Aprendizaje semisupervisado .....	11
1.2. Aprendizaje automático no supervisado.....	11
1.2.1. Análisis de agrupamiento .....	11
<b>CAPÍTULO 2. ANÁLISIS DE AGRUPAMIENTO.....</b>	<b>13</b>
2.1. Medidas de similitud y distancia.....	15
2.1.1. Disimilitud por variable.....	17
2.1.2. Matriz de distancia o disimilitud.....	18
2.2. Algoritmos de agrupamiento basados en particiones .....	19
2.2.1. K-means .....	19
2.2.2. Elección de K .....	20
2.2.3. Selección inicial de centros avanzada.....	22
2.2.4. K-medoides .....	23
2.3. Agrupamiento jerárquico.....	24
2.3.1. Aproximación aglomerativa .....	25
2.3.2. Aproximación divisiva .....	27
2.3.3. Adecuación de la jerarquía a la realidad de los datos.....	29
2.4. Agrupamiento espectral .....	30
2.4.1. Grafo de similitud.....	31
2.4.2. Matriz laplaciana.....	31
2.4.3. Transformación de los datos en un espacio alternativo.....	32
2.5. Agrupamiento basado en densidad.....	35
2.5.1. Algoritmo DBSCAN.....	36
2.5.2. Algoritmo de desplazamiento de media .....	38
2.5.3. Propagación de afinidad.....	42
2.6. Agrupamiento basado en modelos probabilísticos .....	45
2.6.1. Modelos de mixturas .....	45
2.6.2. Algoritmo esperanza-maximización.....	48
2.7. Evaluación de un agrupamiento.....	51
2.7.1. Medidas de evaluación extrínseca.....	53
2.7.2. Medidas de evaluación intrínseca .....	55

---

<b>CAPÍTULO 3. ANÁLISIS DE COMPONENTES .....</b>	57
3.1. Análisis de componentes principales.....	58
3.2. Análisis de componentes independientes .....	63
<b>CAPÍTULO 4. ANÁLISIS DE COAGRUPAMIENTOS .....</b>	65
4.1. Algoritmo de Cheng y Church .....	69
4.2. Algoritmo de firma iterativa .....	70
4.3. Algoritmo de coagrupamiento espectral.....	71
4.4. Estrategia de agrupamiento de doble sentido.....	72
<b>CAPÍTULO 5. APRENDIZAJE SEMISUPERVISADO .....</b>	74
5.1. Estrategia EM en aprendizaje semisupervisado.....	76
5.2. Aprendizaje semisupervisado basado en grafos .....	79
5.3. Coentrenamiento .....	83
<b>CAPÍTULO 6. OTRAS TÉCNICAS NO SUPERVISADAS .....</b>	86
6.1. Análisis de grafos .....	87
6.1.1. Algoritmo PageRank.....	88
6.2. Análisis de reglas de asociación .....	90
6.2.1. Algoritmo Apriori .....	92
<b>GLOSARIO.....</b>	94
<b>ENLACES DE INTERÉS .....</b>	99
<b>BIBLIOGRAFÍA.....</b>	100



# Capítulo 1

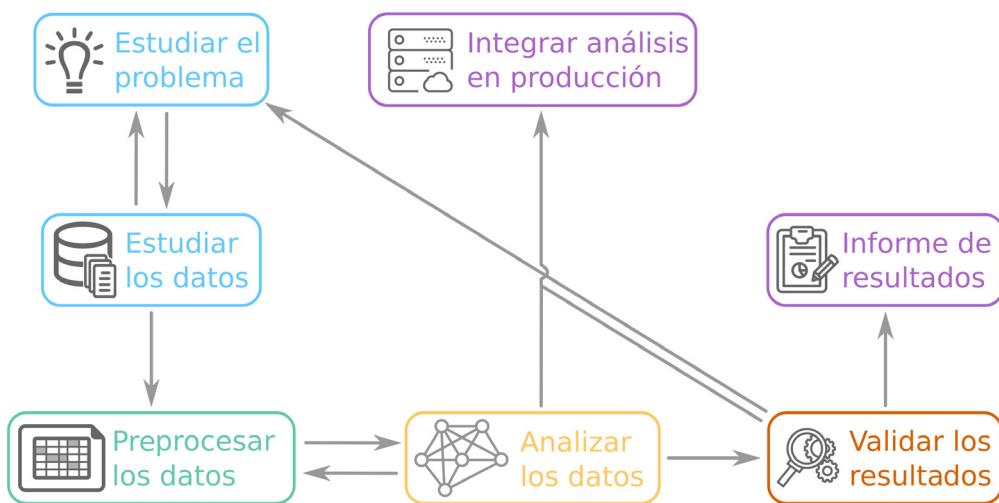
## Introducción a la minería de datos y el aprendizaje automático

Cada día con mayor incidencia, vivimos en un mundo monitorizado donde se generan datos continuamente en casi todos los ámbitos de la vida, desde el más cotidiano (uso de dispositivos electrónicos, coches y otros aparatos repletos de sensores, etc.) a otros más profesionalizados (medicina, industria, etc.). La **minería de datos** es un campo de conocimiento a medio camino entre la estadística y las ciencias de la computación que desarrolla técnicas de análisis de datos y explora su uso para la obtención de información a partir de cualquier fuente, nueva o antigua, de datos.

Los datos, que con frecuencia no han sido generados con el propósito de ser analizados, pueden esconder información relevante para quien los tiene a su disposición. Las técnicas de minería tratan de descubrir esa información, en forma de patrones o relaciones en los datos, y transformarla de tal manera que sea comprensible por el experto o propietario para su aprovechamiento posterior. El uso de estas técnicas requiere de habilidades interdisciplinares: aparte de conocer cómo funcionan, es necesario saber qué se busca para discriminar la información relevante.

El **ciclo de vida** de la minería de datos (véase la Figura 1) se describe mediante un conjunto de etapas que transcurren desde que se accede a la fuente de datos del problema y se plantea un problema que hay que resolver, hasta la presentación de los resultados del análisis (por ejemplo, informes técnicos) y, posiblemente, el despliegue del producto o modelo predictivo o descriptivo resultante. Conlleva la captura y gestión de datos y bases de datos, su preprocesamiento, modelado (análisis) de los datos y validación, así como su presentación y puesta en producción del modelo.

En esta asignatura, nos centramos en la fase de análisis de datos, donde se enmarcaría el proceso de aprendizaje automático, en este caso, no supervisado.



**Figura 1.** Ciclo de vida de la minería de datos.

El **aprendizaje automático** es otra rama de la inteligencia artificial estrechamente relacionada con el análisis de datos. En este caso, el objetivo es el desarrollo de métodos que posibiliten que los ordenadores “aprendan”. El concepto de *aprendizaje*, en este contexto, está estrechamente relacionado con el de identificación y generalización de comportamientos observados en los datos. El aprendizaje automático, tradicionalmente insertado en la etapa de análisis del ciclo de vida de la minería de datos (véase la Figura 1), trata de crear programas informáticos que sean capaces de generalizar, mayoritariamente mediante el uso de técnicas matemáticas (estadísticas), los comportamientos observados en un conjunto de ejemplos previos del problema estudiado.

Existen tres grandes **ramas del aprendizaje automático**: el aprendizaje por refuerzo (basado en el concepto de prueba y error), el aprendizaje supervisado (eminente predictivo) y el aprendizaje no supervisado (mayormente descriptivo o de descubrimiento de información). En esta última rama se centra esta asignatura y las técnicas que estudiaremos.



Por convención, en este documento se usa el siguiente **código de estilo**:

Las **variables aleatorias** se representan usando la letra mayúscula (ej.,  $X_v$  o  $Y$ ). Los valores que toma una variable aleatoria se representan con los mismos caracteres en minúscula (ej.,  $x_v$  o  $y$ ). La asignación  $X_v = x_{v1}$  se usa para denotar que la variable  $X_v$  toma el primero de sus valores posibles,  $x_{v1}$ , cuando el conjunto de posibles valores de  $X_v$  es numerable, o simplemente  $Y = y$  cuando no lo es. Los vectores se representan en negrita, ya sean tanto de variables aleatorias,  $\mathbf{X} = (X_1, \dots, X_v)$ , como de valores,  $\mathbf{x} = (x_1, \dots, x_v)$ . Las matrices suelen también denotarse con letra mayúscula, aunque para evitar confusión se suele usar otro rango de letras. Por ejemplo, la letra  $D$  la usaremos para denotar un conjunto de datos, que no es otra cosa que una matriz con  $n$  filas (tantas como casos) y  $v$  columnas (tantas como variables aleatorias).

## 1.1. Aprendizaje automático supervisado

El **aprendizaje automático supervisado**, también conocido como clasificación supervisada o regresión según el tipo de predicción, es un tipo de aprendizaje que trata de aprender la relación existente entre los datos descriptivos de un problema de interés y una variable especial. El valor de esa variable especial, de difícil recolección por diferentes motivos, es necesario anticiparlo. Es decir, dado un conjunto de casos previos donde se conocen tanto el valor de las variables descriptoras como el de la variable de interés, se busca inferir el comportamiento que relaciona los valores de unas y otra variables.

Concretamente, el objetivo es crear un modelo que, entrenado (configurado) a partir del conocimiento inferido de esa relación, sea capaz de predecir el valor de la variable especial para un ejemplo del cual solo se disponen los valores de las variables descriptoras. Es decir, el aprendizaje supervisado tiene un **objetivo predictivo**.



### Ejemplo

En una fábrica, es necesario detectar qué piezas producidas en una cadena de producción son defectuosas. Sin embargo, dado el volumen de piezas producidas, la inspección manual pieza a pieza no es factible. Por eso, la empresa ha decidido implementar un sistema de aprendizaje automático para solucionar el problema. Dada una foto de una pieza, la empresa quiere saber qué pieza es defectuosa y cuál no.

Para ello, recoge las fotos de las piezas producidas en un período concreto y pide a un experto que etiquete (como "buena" o "defectuosa") cada una de las fotos de piezas. Con esas fotografías etiquetadas, se aprende un modelo de predicción que es capaz de identificar automáticamente las características de una pieza defectuosa y, dada una nueva pieza no etiquetada, el modelo será capaz de dar una predicción fiable de la validez de la pieza fotografiada.

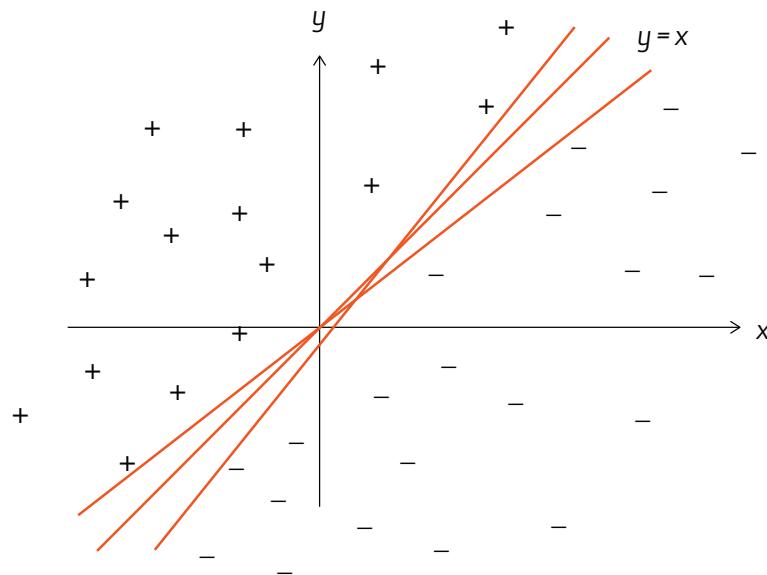
Un **problema de clasificación supervisada** se define dado un conjunto de variables aleatorias, denominadas **variables descriptoras o predictivas**,  $(X_1, \dots, X_v)$ , que describen los casos del problema que se quiere abordar. En clasificación supervisada existe una variable especial: la variable clase o respuesta,  $Y$ . Así, un ejemplo del problema es un vector de tamaño  $v + 1$  que asigna un valor a cada una de las variables descriptoras y a la **variable respuesta**,  $(x_1, \dots, x_v, y)$ . Es necesario disponer de un conjunto de datos  $D$ , donde cada fila es un ejemplo pasado del problema de interés, a partir del cual entrenaremos el modelo,  $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$ .

Dado un conjunto de **ejemplos** de **entrenamiento**, el objetivo es aprender un modelo que sea capaz de predecir el valor de la variable  $Y$  dado un nuevo ejemplo  $(x, ?)$ .



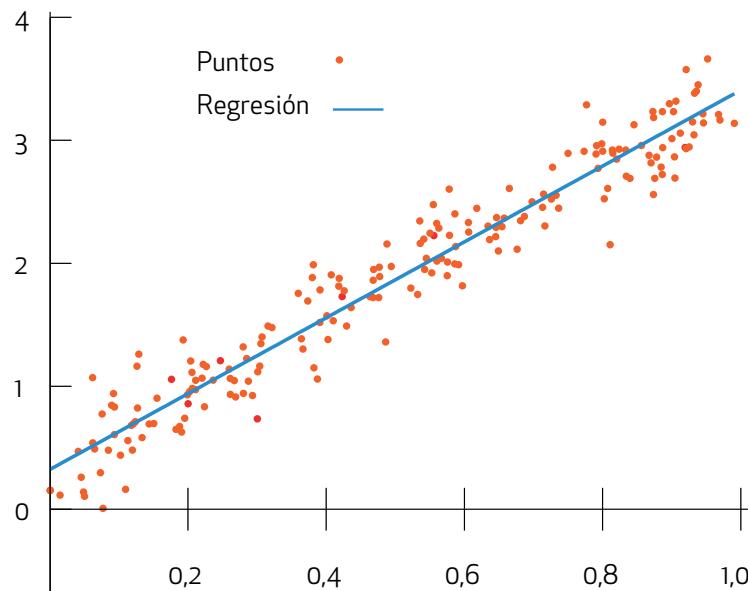
Cuando la variable dependiente u objetivo es **categórica**, decimos que nuestro problema es un problema de clasificación **supervisada**. En este caso, esta variable aleatoria especial se llama **variable clase** y cada uno de los valores posibles de la variable se llama **etiqueta**.

En la Figura 2 se puede ver una muestra, de dimensiones reducidas, de este tipo de problemas, donde cada línea roja representa un posible clasificador lineal que separa los subconjuntos de ejemplos que pertenecen a cada etiqueta o categoría:



**Figura 2.** Tres posibles clasificadores (líneas rojas) que separan el subconjunto de casos con etiqueta positiva del subconjunto con etiqueta negativa. Por Silenius bajo licencia CC BY-SA 3.0, 2.5, 2.0, 1.0. Recuperado de [https://commons.wikimedia.org/wiki/File:Plusieurs\\_separatrice\\_lineaire.svg](https://commons.wikimedia.org/wiki/File:Plusieurs_separatrice_lineaire.svg)

Cuando la variable dependiente u objetivo es **continua**, decimos que estamos ante un problema de regresión. En este caso, esta variable especial también recibe el nombre de *variable respuesta* y, en este sentido, sus posibles valores (numéricos) se conocen también como **valores respuesta**. En la Figura 3 se muestra un ejemplo, de dimensiones reducidas, de este tipo de problemas:



**Figura 3.** Regresión lineal simple con una única variable descriptora (eje horizontal) y la variable respuesta (eje vertical). Dominio público. Recuperado de <https://commons.wikimedia.org/wiki/File:LinearRegression.svg>

### 1.1.1. Aprendizaje semisupervisado

Entre otras suposiciones, el **aprendizaje automático supervisado** asume que los datos de entrenamiento están completos. Concretamente, la información más sensible de este tipo de aprendizaje es la de **supervisión**, tanto que incluso complementa el nombre del paradigma. Así pues, en el escenario tradicional de aprendizaje supervisado se asume que la información de supervisión, es decir, el valor de la variable respuesta  $Y$  de cada ejemplo del conjunto de entrenamiento, es conocida. Todos los ejemplos del conjunto de entrenamiento están etiquetados (tienen un valor asignado para la variable de interés que hay que predecir). Aunque esta suposición es central en este paradigma, no debe olvidarse que obtener el valor de la variable respuesta  $Y$  es habitualmente costoso.

El **aprendizaje semisupervisado** suaviza la suposición de la existencia de un conjunto de datos de entrenamiento completo con respecto a la variable especial  $Y$ . En este caso, el conjunto de entrenamiento puede dividirse en dos subconjuntos:

- El conjunto de datos completo, donde todos los ejemplos tienen un valor asignado para la variable dependiente o respuesta  $Y$ .
- El conjunto de datos no etiquetado, donde los ejemplos no tienen un valor asignado para la variable  $Y$ .

El objetivo de este paradigma de aprendizaje automático sigue siendo el entrenamiento de un modelo predictivo que, dado un nuevo ejemplo no etiquetado, sea capaz de predecir su valor para la variable  $Y$ . Es habitual suponer que el número de ejemplos del primer subconjunto (completo) es bastante inferior al número de ejemplos no etiquetados. Aprender en estas circunstancias requiere de técnicas específicas capaces de extraer el conocimiento disponible de ambos subconjuntos.

## 1.2. Aprendizaje automático no supervisado

Al contrario que el aprendizaje supervisado y su naturaleza predictiva, el **aprendizaje no supervisado** se caracteriza por buscar en los datos relaciones ocultas que permitan describir los datos del mismo conjunto de entrenamiento. Es decir, el aprendizaje no supervisado tiene un objetivo eminentemente descriptivo y de descubrimiento de información, estructuras o patrones en los datos. En este sentido, en el aprendizaje no supervisado no existe una variable especial que haya que predecir.

### 1.2.1. Análisis de agrupamiento

El núcleo principal del aprendizaje no supervisado está compuesto por las técnicas de **agrupamiento automático** (*clustering* en inglés). En este paradigma, se asume que los ejemplos del conjunto de entrenamiento se agrupan de manera natural en subconjuntos. El objetivo de este tipo de técnicas es conocer e identificar esos subconjuntos. Diferentes suposiciones sobre el origen de los datos llevan a las diferentes técnicas para descubrir diferentes agrupaciones en los datos. La finalidad de estas técnicas es básicamente **descriptiva**, aunque, una vez identificado un agrupamiento, es posible incorporar nuevos ejemplos de manera sencilla al subconjunto que corresponda.



## Ejemplo

En una fábrica es necesario identificar diferentes defectos en las piezas producidas en una cadena de producción porque, presumiblemente, tendrán diferente tratamiento (tipo de reparación o destrucción). Sin embargo, dado el volumen de piezas producidas, la inspección manual pieza a pieza no es factible y no se sabe con certeza cuántos tipos de defectos existen y cómo se diferencian entre ellos. Por eso, la empresa ha decidido implementar un sistema de aprendizaje automático para solucionar el problema. Dada una foto de una pieza, la empresa quiere saber qué tipo de defecto tiene la pieza.

Para ello, recoge las fotos de las piezas producidas en un período concreto y se aprende un agrupamiento que es capaz de identificar diferentes características de piezas con diferentes defectos, lo cual devuelve el conjunto de entrenamiento agrupado por los defectos que se observan en cada una de las piezas, y, por ende, un listado de tipos de defecto. Será tarea del experto saber qué tratamiento debe darse a cada tipo de defecto identificado por el algoritmo de agrupamiento.

Un **problema de análisis de agrupamiento** se define dado un conjunto de variables descriptoras,  $(X_1, \dots, X_v)$ , que describen los casos del problema que se quiere abordar. Un ejemplo de problema es un vector de tamaño  $v$  que asigna un valor a cada variable descriptora,  $(x_1, \dots, s_v)$ . A partir de un conjunto de datos,  $D = (x_1, x_2, \dots, x_n)$ , donde cada fila es un ejemplo del problema de interés, se aprende el agrupamiento, es decir, una serie de subconjuntos (clústeres) de ejemplos que divide los ejemplos del conjunto de entrenamiento  $D$ . El objetivo es aprender la mejor división del conjunto de entrenamiento  $D$  en subconjuntos de acuerdo con cierto criterio.



## Capítulo 2

### Análisis de agrupamiento

El **análisis de agrupamiento** (*clustering* en inglés) tiene por objetivo principal conseguir subgrupos homogéneos de ejemplos que manifiestan diferencias relevantes con los ejemplos de otros subgrupos. Cuando el concepto de **homogeneidad** de los ejemplos en el mismo agrupamiento y el de diferencia con los ejemplos de otros agrupamientos se difuminan, es común recurrir a una jerarquía de agrupamientos. Esta jerarquía, aparte de la información de **disimilitud** o **similitud** entre elementos, también da información sobre las diferencias entre agrupamientos o clústeres.

La mayoría de algoritmos de agrupamiento asignan cada uno de los casos del conjunto de entrenamiento a un agrupamiento **sin tener en cuenta la existencia de un modelo probabilístico** que describa la generación de los casos. Dado un número de agrupamientos,  $K$ , cada caso se asigna a una única partición. Se asume la existencia de una función  $C^*$  (mapeo  $n$  a  $1$ ) que asigna cada ejemplo a un agrupamiento,  $k = C^*(x_i)$ .

A partir de esto, se puede definir el problema de agrupamiento como un **problema de optimización**. El objetivo es encontrar una función  $C$  que aproxime la función de asignación original (desconocida),  $C^*$ , minimizando cierta función de pérdida. Por ejemplo, tal y como se comentaba anteriormente, se puede estar interesado en minimizar la **distancia** entre los elementos de un mismo agrupamiento. Es decir, se buscan clústeres compactos. En este caso, la función que hay que minimizar, conocida como **dispersión intraclúster**, es:

$$I(C) = \frac{1}{2} \sum_{k=1}^K \sum_{i:C(x_i)=k} \sum_{i':C(x_{i'})=k} d(x_i, x_{i'})$$

De manera alternativa, podríamos buscar que los diferentes agrupamientos estén separados entre sí. En este caso, el objetivo sería maximizar la función de **dispersión interclúster**:

$$O(C) = \frac{1}{2} \sum_{k=1}^K \sum_{i:C(x_i)=k} \sum_{i':C(x_{i'}) \neq k} d(x_i, x_{i'})$$

De hecho, ambos objetivos son **equivalentes**. Nótese que, dada una función  $C$ , la distancia total entre todos los pares de elementos del conjunto de entrenamiento (constante) puede expresarse de la siguiente manera:

$$T = \frac{1}{2} \sum_{i=1}^n \sum_{i'=1}^n d(x_i, x_{i'}) = \frac{1}{2} \sum_{k=1}^K \sum_{i:C(x_i)=k} \left( \sum_{i':C(x_{i'})=k} d(x_i, x_{i'}) + \sum_{i':C(x_{i'}) \neq k} d(x_i, x_{i'}) \right)$$

$$T = I(C) + O(C)$$

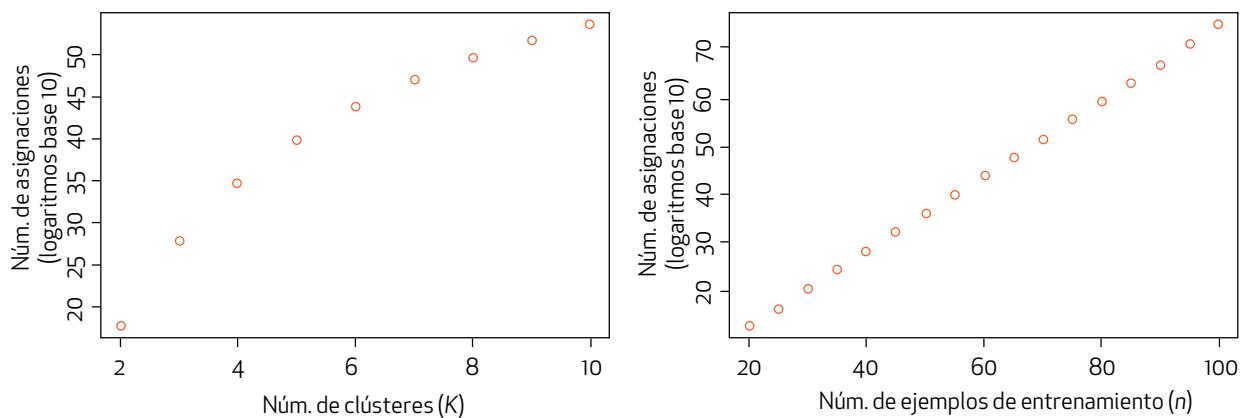
Así, dado que  $I(C) = T - O(C)$ , minimizar la función  $I$  es equivalente a maximizar la función  $O$ .



El problema radica en que la búsqueda de la función  $C$  que optimiza la función de pérdida elegida no se puede realizar mediante una exploración exhaustiva de todas las posibles asignaciones de elementos a clústeres. El número de posibles asignaciones es un **número de Stirling de segunda especie**:

$$S(n, K) = \frac{1}{K!} \sum_{k=1}^K (-1)^{K-k} \binom{K}{k} k^n$$

Este número es exponencial en el número de agrupamientos y de ejemplos de entrenamiento (véase la Figura 4). Por ello, la **búsqueda exhaustiva** se convierte en inalcanzable incluso para conjuntos de entrenamiento pequeños. Ante este problema, es habitual recurrir a técnicas **heurísticas** que simplifiquen la búsqueda.



**Figura 4.** Número de asignaciones posibles según el número de ejemplos ( $n$ ) y el número de clústeres ( $K$ ) en escala logarítmica.

En la figura de la izquierda, fijado el número de ejemplos en  $n = 60$ , se varía  $K$ . En la figura de la derecha, fijado el número de clústeres en  $K = 6$ , se varía  $n$ .

Como se puede inferir de la discusión previa, los conceptos de **similitud** y **disimilitud** son básicos a la hora de definir el objetivo de un problema de agrupamiento. De hecho, la elección de la métrica de distancia (disimilitud) determina el resultado del agrupamiento.

## 2.1. Medidas de similitud y distancia

A la hora de medir y comparar la **similitud o disimilitud** entre objetos y/o grupos, existe una amplia bibliografía sobre diferentes medidas con diferentes características que responden a diferentes necesidades. En esta sección, se hará un repaso de las medidas de disimilitud o similitud más habitualmente empleadas en problemas de análisis de agrupamiento.

Probablemente, la medida de distancia entre dos puntos más común es la **distancia euclíadiana**:

$$d(x_1, x_2) = \sqrt{\sum_{j=1}^v (x_{1j} - x_{2j})^2} = \sqrt{(x_1 - x_2)^T (x_1 - x_2)}$$

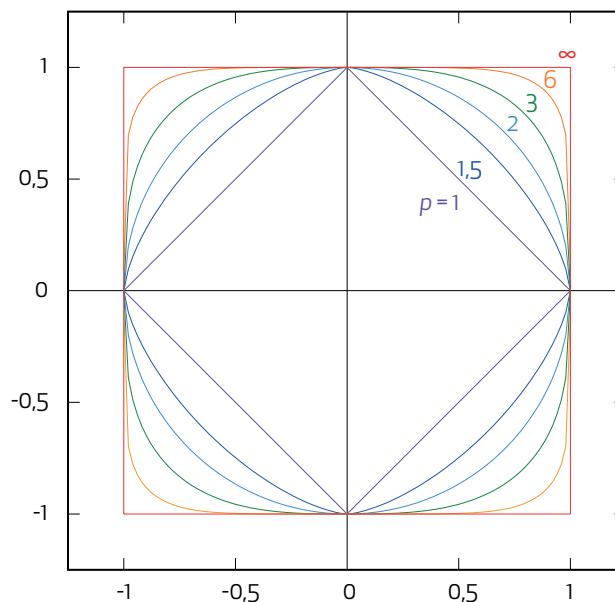
La distancia euclíadiana, que mide la distancia en línea recta entre los dos puntos, se integra en la familia de distancias conocida como la **norma  $p$** , que generaliza la distancia euclíadiana para todos los valores de  $p \geq 1$ :

$$d_p(x_1, x_2) = \|x_1 - x_2\|_p = \left( \sum_{j=1}^v |x_{1j} - x_{2j}|^p \right)^{1/p}$$

Esta generalización produce un número infinito de medidas de disimilitud, que han sido usadas en diferentes aplicaciones según su idoneidad. Aparte de la distancia euclíadiana, **otras distancias derivadas de la norma  $p$**  son habituales en esta área:

- Rectilínea ( $p = 1$ ):  $d(x_1, x_2) = \sum_{j=1}^v |x_{1j} - x_{2j}| = \|x_1 - x_2\|_1$
- Euclíadiana ( $p = 2$ ).
- Máximo ( $p = \infty$ ):  $d(x_1, x_2) = \max_{j \in \{1, v\}} |x_{1j} - x_{2j}| = \|x_1 - x_2\|_\infty$

En la Figura 5 se ilustra cómo interpretan el espacio de distancias las diferentes instanciaciones (con diferentes valores de  $p$ ) de la norma  $p$ :

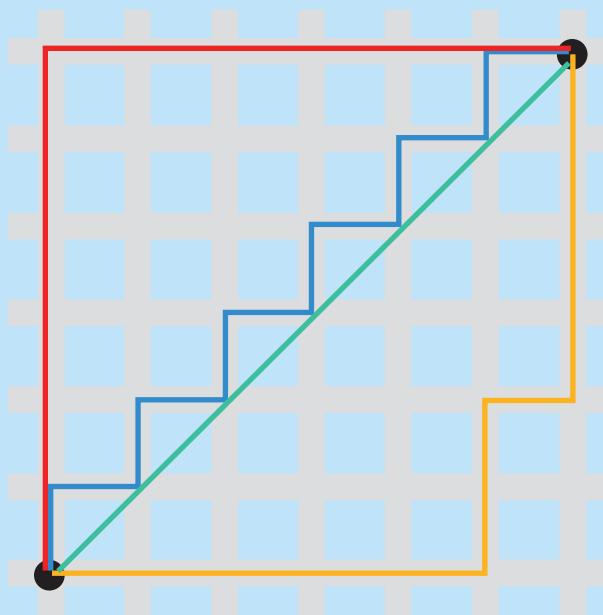


**Figura 5.** Circunferencia unitaria en diferentes instanciaciones de la norma  $p$  en un escenario de dos componentes o variables. Por Cmglee bajo licencia CC BY-SA 3.0. Recuperado de [https://commons.wikimedia.org/wiki/File:Vector-p-Norms\\_qtl1.svg](https://commons.wikimedia.org/wiki/File:Vector-p-Norms_qtl1.svg)



### Ejemplo

Desplazarse en una ciudad no es normalmente sencillo debido a múltiples motivos. Para llegar a un punto desde otro, la línea recta no es siempre una posibilidad real. Este problema se lleva al extremo en lugares como Manhattan, en la ciudad de Nueva York, o el Eixample, en la ciudad de Barcelona, donde la distribución urbana en forma de manzanas (bloques de edificios) rectangulares del mismo tamaño obliga a quien quiere desplazarse en diagonal a dibujar ángulos rectos (véase la Figura 6). Es conocido que los taxistas de Manhattan usan la distancia rectilínea para sus cálculos de rutas. Por eso, dicha distancia recibe también los nombres de Taxicab o Manhattan.



**Figura 6.** Tres caminos (rojo, azul y amarillo) entre dos puntos en una cuadrícula con la misma distancia rectilínea. La distancia euclidiana se incluye también en color verde. Dominio público. Recuperado de [https://commons.wikimedia.org/wiki/File:Manhattan\\_distance.svg](https://commons.wikimedia.org/wiki/File:Manhattan_distance.svg)

La **distancia Mahalanobis** es especialmente útil cuando se quiere tener en cuenta la dependencia entre variables:

$$d(x_1, x_2) = \sqrt{(x_1 - x_2)^T \Sigma^{-1} (x_1 - x_2)}$$

Incluyendo la matriz de **covarianzas**,  $\Sigma$ , se consigue tener en cuenta la **varianza** de cada variable y las relaciones entre estas. Si solo se contempla la varianza de cada variable de manera independiente, desestimando posibles dependencias entre variables, la distancia Mahalanobis se puede expresar de la siguiente manera:

$$d(x_1, x_2) = \sqrt{\sum_{j=1}^v \left( \frac{x_{1j} - x_{2j}}{\sigma_j} \right)^2} = \sqrt{(x_1 - x_2)^T S^{-1} (x_1 - x_2)}$$

En la fórmula anterior,  $\sigma_j$  es la desviación típica de la  $j$ -ésima variable y  $S$  es la matriz diagonal que tiene  $S_{jj} = \sigma_j, \forall j = \{1, \dots, v\}$  (y  $S_{jj'} = 0, \forall j \neq j'$ ). Esta distancia, concretamente, es la distancia euclíadiana estandarizada. Nótese que, si se usa  $\sigma_j = 1$  para todas las variables, es decir, la matriz  $S$  es la matriz identidad, la distancia de Mahalanobis se convierte en la distancia euclíadiana. En otras palabras, la distancia euclíadiana es equivalente a la de Mahalanobis si se asume que todas las variables son independientes y tienen la misma varianza.



Cuando se quiere establecer una distancia entre casos que se describen por medio de **variables categóricas**, las medidas de distancia que se han explicado hasta ahora no pueden aplicarse. En este caso, y aunque pueda parecer una opción extrema, es habitual en esta área del conocimiento considerar la mínima distancia (0) cuando el valor es coincidente y la máxima (1) en cualquier otro caso. La medida se definiría, para una variable, de la siguiente manera:

$$d_j(x_{ij}, x_{i'j}) = \begin{cases} 0 & \text{si } x_{ij} = x_{i'j} \\ 1 & \text{si } x_{ij} \neq x_{i'j} \end{cases}$$

Cuando las variables son binarias, existen otras muchas alternativas. Una de las más populares es el **coeficiente de Jaccard**, una medida de similitud entre vectores de variables binarias que aplica el ratio entre el número de variables donde el valor (positivo) coincide y el número de variables donde al menos uno de los ejemplos toma valor positivo:

$$s(x_1, x_2) = \frac{| \{j \in \{1, \dots, v\} : x_{1j} = 1 \wedge x_{2j} = 1 \} |}{| \{j \in \{1, \dots, v\} : x_{1j} = 1 \vee x_{2j} = 1 \} |}$$

### 2.1.1. Disimilitud por variable

Hasta ahora se ha hecho una revisión de las principales métricas de distancias, asumiendo que todas las variables descriptivas son del mismo tipo. Sin embargo, existe la posibilidad de establecer una medida de **disimilitud por variable** —especialmente apropiada cuando el conjunto de variables descriptivas es heterogéneo— y **combinarlas**:

$$d(x_i, x_{i'}) = \sum_{j=1}^v d_j(x_{ij}, x_{i'j})$$

La combinación de la disimilitud en todas las variables con el mismo peso no implica necesariamente que todas las variables tienen la misma importancia en el cálculo de la disimilitud global. Factores como el rango de cada variable, la función de disimilitud usada, etc., condicionan la influencia de cada variable en el cálculo global. Para corregir esta eventualidad, se pueden considerar **pesos diferentes para cada variable**:

$$d(x_i, x_{i'}) = \sum_{j=1}^v w_j \cdot d_j(x_{ij}, x_{i'j})$$

En este caso,  $\sum_{j=1}^V w_j = 1$ . Hastie, Tibshirani y Friedman (2008) proponen un **método para calcular los pesos de cada variable** a partir de su influencia en el cálculo de las disimilitudes en todo el conjunto de datos:

$$w_j = \frac{1}{\bar{d}_j}, \quad \bar{d}_j = \frac{1}{N^2} \sum_{i=1}^n \sum_{i'=1}^n d_j(x_{ij}, x_{i'j})$$

Si consideramos que todas las variables son numéricas y que la distancia  $d_j$  es la distancia euclíadiana al cuadrado,  $d_j(x_{ij}, x_{i'j}) = (x_{ij} - x_{i'j})^2$ , el peso que se obtendría tiene una fórmula cerrada que se reduce a lo siguiente:

$$w_j = \frac{1}{2 \cdot \text{var}_j}$$

En la fórmula anterior,  $\text{var}_j$  es la estimación de la varianza obtenida con este conjunto de datos. El resultado obtenido con este peso de las variables es equivalente a estandarizar las variables (véase la distancia de Mahalanobis). Aunque es normalmente recomendado, el uso de la estandarización puede desdibujar la separación original de los datos en agrupamientos, por lo que se recomienda un uso prudente.

### 2.1.2. Matriz de distancia o disimilitud

La mayoría de técnicas de agrupamiento pueden trabajar con **dos tipos de matrices**:

- La **matriz original de datos**, con  $n$  filas (ejemplos) y  $v$  columnas (variables descriptivas), donde cada celda guarda el valor original que toma una variable en un caso concreto.
- La **matriz de distancias entre casos  $M$** , con  $n$  filas y columnas (cuadrada), donde cada celda  $M_{ij}$  guarda la distancia entre el ejemplo  $x_i$  y el ejemplo  $x_j$ . Los elementos de la diagonal principal de la matriz de distancias son 0 (distancia entre un vector y él mismo) y el resto son valores no negativos. Es común asumir que la matriz es simétrica, ya que es una propiedad común en una función de disimilitud  $d(x_i, x_j) = d(x_j, x_i)$ . Si no lo fuese, la matriz  $M$  suele ser reemplazada por la media de ambos valores,  $(d(x_i, x_j) + d(x_j, x_i))/2$ , que en forma matricial se denota de la siguiente forma:  $(M + M^T)/2$ .

Una **matriz de similitud** es equivalente a una matriz de distancias o disimilitud, tal y como se define anteriormente, con una única diferencia: cada celda define la similitud entre dos ejemplos (y no la distancia o disimilitud). Así, la matriz de similitud  $S$  es una matriz cuadrada con  $n$  filas y columnas donde cada celda  $S_{ij}$  guarda la similitud entre los ejemplos  $x_i$  y  $x_j$ . Las celdas de la diagonal principal  $S_{ii}$  tienen valor 1 (máxima similitud, iguales). Entre el resto de pares de ejemplos, los valores cercanos a 1 denotan gran similitud. Y, cuanto menor sea el valor (más cercano a 0), menor será la similitud entre el par de ejemplos en cuestión (mayor distancia). Entre las muchas técnicas para expresar un **valor de distancia en términos de similitud**, una de las más habituales se expresa como sigue ( $c$  es un parámetro de escala):

$$S_{ij} = e^{(-M_{ij}^2/c)}$$

## 2.2. Algoritmos de agrupamiento basados en particiones

Muchos algoritmos de agrupamiento hacen uso de un **centro** o **centroide** para representar cada agrupamiento. En este contexto, el **agrupamiento** es la asignación de cada ejemplo en el conjunto de entrenamiento a uno de los centros o centroides o representantes. A lo largo de esta sección se abordan diferentes algoritmos de agrupamiento basados en particiones. Entre ellos, se estudiará el algoritmo K-means (K-medias en español), que es probablemente el algoritmo de agrupamiento más popular.

### 2.2.1. K-means

El popular **algoritmo K-means** es un método iterativo que inicialmente crea  $K$  clústeres y reconsidera la asignación de ejemplos a los  $K$  clústeres en cada iteración hasta que se llega a la convergencia (cuando ningún ejemplo cambia de clúster). Concretamente, está diseñado para trabajar con variables descriptivas continuas y usa la distancia euclidiana al cuadrado para calcular la disimilitud entre elementos. Está basado en la idea de que, dado un conjunto de vectores medios,  $\{\bar{x}_1, \dots, \bar{x}_K\}$ , la dispersión intraclúster (véase la página 14) puede expresarse como se muestra a continuación:

$$I(C) = \sum_{k=1}^K N_k \cdot \sum_{x_i: C(x_i)=k} \|x_i - \bar{x}_k\|^2$$

En la fórmula anterior,  $\bar{x}_k$  es el vector medio (centro) asociado al  $k$ -ésimo clúster y  $N_k$  es el número de ejemplos asignados a ese clúster. Así, el objetivo de optimización puede reinterpretarse como la asignación de los ejemplos del conjunto de entrenamiento a los  $K$  agrupamientos, de tal manera que la disimilitud media de los ejemplos con respecto al centro de sus respectivos clústeres se minimice. En ese sentido, el **objetivo** se puede expresar de la siguiente manera:

$$\operatorname{argmin}_{C, \{\bar{x}_1, \dots, \bar{x}_K\}} \sum_{k=1}^K N_k \cdot \sum_{x_i: C(x_i)=k} \|x_i - \bar{x}_k\|^2$$

Nótese que son dos los parámetros que se obtienen de este proceso de optimización: la función de asignación,  $C$ , y el conjunto de centros final,  $\{\bar{x}_1, \dots, \bar{x}_K\}$ . El método K-means propone un algoritmo **iterativo** que busca los valores que optimizan, por separado aunque de manera consecutiva, los dos parámetros. Como puede verse en el pseudocódigo de la Tabla 1, se alterna la asignación de los ejemplos del conjunto de entrenamiento a cada centro con la reestimación de los nuevos centros:

**Tabla 1**  
Pseudocódigo del algoritmo K-means

---

#### K-means

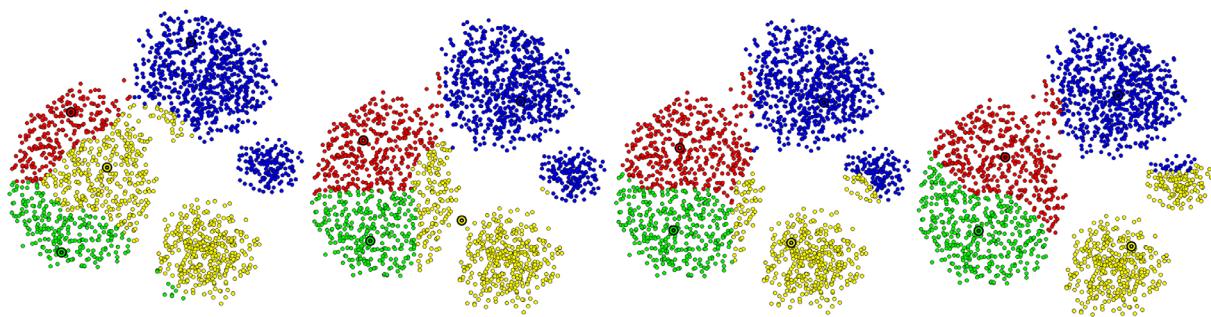
Recibe: conjunto de entrenamiento,  $\{x_1, \dots, x_n\}$ ; número de clústeres,  $K$ .

1. Elegir (aleatoriamente)  $K$  puntos del conjunto de entrenamiento como centros,  $\{\bar{x}_1, \dots, \bar{x}_K\}$ .
2. Asignar cada ejemplo  $x_i$  al clúster del centro más cercano:  $C(x_i) = \operatorname{argmin}_{k \in \{1, \dots, K\}} \|x_i - \bar{x}_k\|^2$ .
3. Para cada clúster  $k$ , recalcular su centro:  $\bar{x}_k = \operatorname{argmin}_x \sum_{x_i: C(x_i)=k} \|x_i - x\|^2$ .
4. Iterar los pasos 2 y 3 hasta que los centros no cambian.

Devuelve: conjunto de centros,  $\{\bar{x}_1, \dots, \bar{x}_K\}$ ; asignación,  $C$ .

---

En la Figura 7 se muestra un ejemplo de ejecución del algoritmo sobre un conjunto de datos sintético:



**Figura 7.** Ejecución del algoritmo K-means ( $K = 4$ ) sobre un conjunto de datos sintético, desde la primera elección aleatoria de centros (izquierda) a la asignación final (derecha).

A pesar de su popularidad, K-means tiene una serie de **desventajas**. Por un lado, el problema más obvio es que este algoritmo usa  $K$ , el número de agrupamientos a formar, como un **parámetro**. Muchas veces, en estudios reales, este valor se desconoce o es precisamente uno de los valores que se espera estimar. En secciones posteriores se estudiará un popular procedimiento para la selección del valor de  $K$ .

Por otro lado, al ser un algoritmo de búsqueda local inicializado de manera aleatoria, puede quedar **atrapado en óptimos locales**. Es decir, K-means es sensible a la elección inicial de centros. Para contrarrestarlo, se suele aconsejar la reejecución del algoritmo un número determinado de veces y quedarse con la mejor solución, es decir, la que minimiza la dispersión intraclúster. Este método es también altamente sensible a valores raros (**outliers** en inglés). Es necesario, en la medida de lo posible, detectar y eliminar estos valores extraños en un proceso previo a la ejecución del algoritmo. Además, K-means sufre para encontrar una **agrupación satisfactoria** cuando los clústeres originales son de distinto tamaño y/o densidad, o no son **convexos**.

Finalmente, K-means ha sido diseñado para trabajar en el **espacio continuo**. El simple hecho de considerar que el centro de un clúster se puede calcular como el ejemplo que el valor medio de los elementos del clúster toma para cada variable es una consecuencia directa de esta decisión. A continuación se estudiará una generalización de este algoritmo que permite lidiar con otros tipos de variables descriptivas.



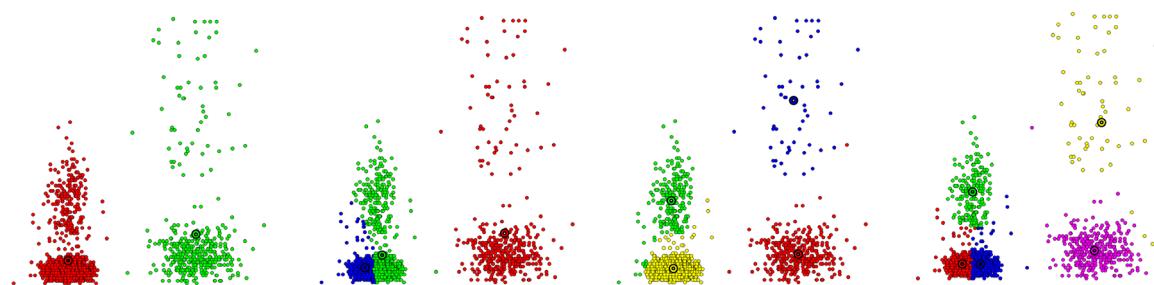
### Enlace de interés

Documentación del algoritmo K-means implementado en la librería scikit-learn para el lenguaje de programación Python:

<https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html>

## 2.2.2. Elección de $K$

La **elección del valor de  $K$**  (número de clústeres) adecuado es clave para obtener un buen resultado de agrupamiento. En la Figura 8 se pueden observar los diferentes agrupamientos obtenidos al ejecutar el algoritmo K-means con diferentes valores de  $K$  sobre un conjunto de datos de ejemplo.



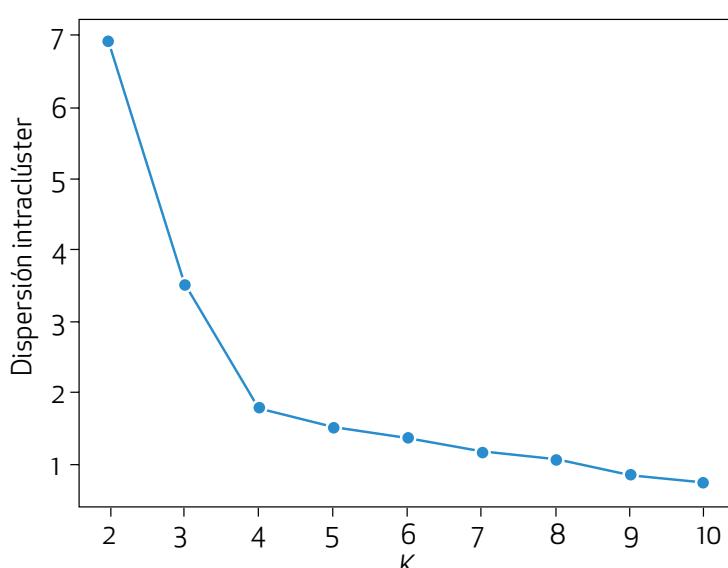
**Figura 8.** Ejecución del algoritmo K-means sobre un conjunto de datos sintético usando diferentes valores para el número de agrupamientos,  $K = \{2, 3, 4, 5\}$ .

Saber cuál es el valor de  $K$  adecuado no es un ejercicio trivial. Una solución analítica se puede obtener mediante el uso de la **validación cruzada** para identificar el mejor valor de  $K$  evaluando sobre subconjuntos de datos no usados para entrenar el algoritmo. Sin embargo, es una práctica habitual usar la inspección visual de los resultados para elegir dicho valor, el **método del codo**.



El **procedimiento del codo** consiste en realizar diferentes ejecuciones del algoritmo K-means con diferentes valores (crecientes) de  $K$ . El mejor agrupamiento resultante con cada valor de  $K$  se evalúa usando la medida de dispersión intraclúster. En este caso, cuanto menor sea el valor obtenido, mejor es *a priori* el agrupamiento. Sin embargo, se da la paradoja de que esta medida siempre decrece a medida que se aumenta el valor de  $K$ . El valor de  $K$  que minimiza la dispersión intraclúster no es, por lo tanto, una buena elección.

Tal y como se observa en el ejemplo de la Figura 9, donde se muestra gráficamente el valor de dispersión obtenido (eje Y) dado un valor de  $K$  (eje X) sobre un conjunto de datos de muestra, es habitual observar un rápido decrecimiento de la dispersión con los primeros valores de  $K$ , el cual se atenúa a medida que  $K$  toma valores mayores:



**Figura 9.** Representación gráfica de la idoneidad del agrupamiento para distintos valores de  $K$  e identificación del codo en el valor de  $K = 4$ .

La gráfica suele dibujar lo que se ha interpretado como un brazo semiflexionado. La línea dibuja, en la primera parte, una **pendiente pronunciada** y, en la segunda parte, una **pendiente suavizada**. El punto de flexión o **codo** representa el último valor de  $K$  que implica una mejora sustancial en términos de dispersión intraclúster con respecto al valor anterior. El valor que toma  $K$  en el codo suele considerarse un valor adecuado y, en ese sentido, suele ser el valor recomendado.

### 2.2.3. Selección inicial de centros avanzada

Como se ha explicado anteriormente, K-means es **sensible a la selección inicial de centros** que, de acuerdo con el algoritmo original, se eligen por muestreo aleatorio del conjunto de aprendizaje. Así, el algoritmo encalla fácilmente en óptimos locales. La solución más sencilla consiste en repetir el algoritmo múltiples veces con diferentes inicializaciones.

El algoritmo **K-means++** de Arthur y Vassilvitskii (2007) viene a lidiar con este problema. Estos autores proponen tomar  $K$  ejemplos, uno por uno, a través de un muestreo sobre el conjunto de datos de entrenamiento que sigue una **distribución de probabilidad no uniforme**. Es decir, no todos los casos del conjunto de entrenamiento tienen la misma probabilidad de ser escogidos. Las  $K$  muestras serán usadas como los  $K$  centros para la inicialización del algoritmo K-means.



La intuición detrás de este algoritmo es que los **centros** deben estar distribuidos por todo el espacio y no concentrados en zonas con, tal vez, mayor densidad de puntos. Los centros se eligen uno por uno y se van incluyendo en un conjunto de centros  $S$ . Dado un conjunto de centros ya elegidos,  $S$  ( $|S| < K$ ), cuanto mayor sea la distancia de un punto del conjunto de entrenamiento con respecto a los centros en  $S$ , mayor probabilidad debería tener dicho punto de ser considerado el siguiente centro.

Arthur y Vassilvitskii (2007) decidieron llevar a cabo un muestreo donde la probabilidad de elegir un punto es proporcional al cuadrado de la distancia mínima de ese punto a los elementos de  $S$ . El pseudocódigo de este algoritmo está disponible en la Tabla 2:

**Tabla 2**

Pseudocódigo del algoritmo K-means++ para la selección de los centros iniciales

**K-means++ (inicialización)**

Recibe: conjunto de entrenamiento,  $\{x_1, \dots, x_n\}$ ; número de clústeres,  $K$ .

1. Elegir (aleatoriamente) 1 punto del conjunto de entrenamiento como primer centro,  $S = \{\bar{x}_1\}$ .
2. Mientras  $|S| < K$ , repetir.

2.1. Para todos los ejemplos de entrenamiento, calcular  $D(x_i, S)$ , la distancia al centro más cercano:  

$$D(x_i, S) = \min_{k \in \{1, \dots, |S|\}} \|x_i - \bar{x}_k\|^2.$$

2.2. Muestrear un nuevo caso  $x'$  del conjunto de entrenamiento, donde el caso  $x$  tiene probabilidad  $D(x, S)^2 / \sum_{i=1}^n D(x_i, S)^2$  y añadir a  $S$ :  $S = S \cup \{x'\}$ .

Devuelve: conjunto de centros,  $\{\bar{x}_1, \dots, \bar{x}_K\}$ .

El **algoritmo K-means++** supone una mejora en la inicialización de K-means que previene, en gran medida, y al contrario que el algoritmo original, de quedar atrapado en óptimos locales lejos de la partición original u óptima. Nótese que el algoritmo no elige, a cada paso, simplemente el punto con mayor distancia mínima a un centro de  $S$ . Aunque en esencia el objetivo es el mismo, este proceder conllevaría una serie de problemas, como la tendencia a seleccionar outliers como centros.



### Enlace de interés

El algoritmo K-means implementado en la librería scikit-learn para el lenguaje de programación Python usa por defecto esta inicialización. El parámetro “init” de dicha función permite fichar el tipo de inicialización usada.

<https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html>

## 2.2.4. K-medoides

Una de las principales ventajas de trabajar solo en el espacio continuo es que, por definición, un vector medio  $\bar{x}_k$  calculado como la media de los ejemplos asignados al  $k$ -ésimo clúster es un **ejemplo válido**. Es decir, aunque no lo sea,  $\bar{x}_k$  podría ser un elemento del conjunto de entrenamiento. En cambio, cuando las variables no son (todas) continuas, este comportamiento no tiene por qué darse. Además, la definición de media no existe en todos los casos. Por ejemplo, dado un conjunto de 10 casos de entrenamiento que toman en su  $j$ -ésima variable los valores  $\{a, b, a, a, a, b, c, c, b\}$ , ¿cuál sería el valor medio? La media no está definida para variables categóricas.

De manera alternativa, se podría considerar el uso de la moda (el valor más frecuente) para reemplazar la media (y en dicho ejemplo saldría elegido el valor  $a$ ). En este caso, usando la medida estándar de disimilitud para variables categóricas (véase la página 17), el primer caso estaría a distancia 0 — $d_j(a, a)$ — de la moda, mientras que el segundo caso estaría a distancia 1 — $d_j(b, a)$ —.



El **algoritmo K-medoides** (véase el pseudocódigo de la Tabla 3) soluciona este problema añadiendo una restricción a la búsqueda de los centros: un centro debe ser un elemento del conjunto de entrenamiento. Para hacer evidente que los puntos seleccionados no responden a la definición de centro, en este contexto los puntos de referencia de cada clúster se conocen como **medoides**.

**Tabla 3**

Pseudocódigo del algoritmo K-medoides

#### K-medoides

Recibe: conjunto de entrenamiento,  $\{x_1, \dots, x_n\}$ ; número de clústeres,  $K$ .

1. Elección (aleatoria) de  $K$  puntos del conjunto de entrenamiento como medoides,  $\{\check{x}_1, \dots, \check{x}_K\}$ .
2. Asignar cada ejemplo  $x_i$  al clúster del medoide más cercano:  $C(x_i) = \operatorname{argmin}_{k \in \{1, \dots, K\}} d(x_i, \check{x}_k)$ .
3. Para cada clúster  $k$ , recalcular su medoide:  $\check{x}_k = \operatorname{argmin}_{x: C(x)=k} \sum_{x_i: C(x_i)=k} d(x_i, x)$ .
4. Los pasos 2 y 3 se iteran hasta que los centros no cambian.

Devuelve: conjunto de centros,  $\{\check{x}_1, \dots, \check{x}_K\}$ ; asignación,  $C$ .

K-medoides se puede considerar una **generalización** de K-means en la medida en que no restringe el uso de una medida de disimilitud concreta para elegir los medoides y la asignación de elementos a los clústeres. Por otro lado, la búsqueda del medoide entre los elementos del clúster se concreta en el paso 3 (véase el pseudocódigo en la Tabla 3), que restringe la búsqueda de los posibles vectores que minimizan la suma de distancias a aquellos que pertenecen al clúster,  $x : C(x) = k$ .

Kaufman y Rousseeuw (1987) proponen una variante de este algoritmo que busca obtener el máximo rendimiento del hecho de que los centroides solo pueden ser elementos del clúster (y, por ende, del conjunto de entrenamiento). El algoritmo, conocido como **PAM** por sus siglas en inglés (de *partitioning around medoids*, es decir, agrupar alrededor de medoides), selecciona del conjunto de entrenamiento  $K$  puntos  $\{\check{x}_1, \dots, \check{x}_K\}$  como medoides y asigna el resto de ejemplos a los clústeres respectivos. Tras esta inicialización similar a la de K-medoides y K-means, considera cada elemento  $x_i \notin \{\check{x}_1, \dots, \check{x}_K\}$  como si fuese el medoide de su clúster y recalcula la medida a minimizar: la dispersión intraclúster. El cambio que más mejore la función objetivo se aplica (dado  $k : C(x_i) = k, \check{x}_k = x_i$ ). Este proceso se itera hasta que ningún cambio mejora la dispersión intraclúster.

Este algoritmo tiene varios **inconvenientes** característicos del algoritmo K-means. La  $K$  es un parámetro del método y, como tal, debe ser fijada al mejor valor posible. La discusión al respecto de la sección “2.2.2. Elección de  $K$ ” es también válida para este algoritmo. También es un algoritmo altamente sensible a la elección de los medoides iniciales. La repetición del proceso múltiples veces para seleccionar la mejor solución es una solución sencilla. Una inicialización avanzada como la de K-means++ (véase la sección “2.2.3. Selección inicial de centros avanzada”) sería posible si se usara una medida de disimilitud no sólo apta para datos continuos.

## 2.3. Agrupamiento jerárquico

En las secciones previas se hacía hincapié en la dificultad de seleccionar el número de clústeres (valor de  $K$ ) óptimo. El **agrupamiento jerárquico** no necesita especificar un número de clústeres, ya que no crea una simple partición de los ejemplos de entrenamiento. Como su nombre indica, se busca una jerarquía de clústeres, que puede entenderse como una evolución o secuencia ordenada del agrupamiento desde  $K=1$  (todos los casos se agrupan en un único clúster) hasta  $K=n$  (cada ejemplo tiene su propio clúster).

Paso a paso y mediante una búsqueda voraz, dos clústeres del paso anterior se agrupan en un único clúster (aglomeración) o bien un único clúster se separa en dos subgrupos (división). Estas decisiones no se combinan, sino que de manera consistente se aplica durante todo el proceso la misma decisión. Así se crean las dos aproximaciones existentes al agrupamiento jerárquico: **la aglomerativa y la divisiva**. En ambos casos, cada nivel de la jerarquía representa una partición concreta de los datos. Queda a la elección del usuario obtener, si así lo desea, un agrupamiento concreto (el más apropiado), seleccionando las agrupaciones del nivel de la jerarquía deseado.

Existen herramientas gráficas para examinar el resultado de un algoritmo de agrupamiento jerárquico. La más común es el **dendrograma** (véase la Figura 10), que aporta información relevante a la hora de tomar la decisión, si se desea, sobre el mejor agrupamiento. Se trata de un árbol binario donde cada nodo representa un clúster. Los dos hijos de cada nodo se reparten los ejemplos que se incluyen en el nodo padre. Se representa gráficamente de tal manera que la distancia entre dos nodos y su padre es proporcional a la disimilitud interclúster de los dos nodos.

La principal elección de un algoritmo de agrupamiento jerárquico radica en la **medida de disimilitud** usada para agrupar o dividir clústeres. Tanto en el caso de la aproximación aglomerativa como en el caso de la divisiva, la decisión de qué dos clústeres combinar o qué clúster dividir (y cómo) se toma de acuerdo con la función que hay que optimizar, la cual, a su vez, se basa en una medida de disimilitud. A continuación, se estudiarán de manera separada ambas aproximaciones.

### 2.3.1. Aproximación aglomerativa

La **aproximación aglomerativa** (*bottom-up* en inglés) comienza creando  $K = n$  clústeres y asignando cada ejemplo de entrenamiento a uno de ellos. A cada paso, se eligen los dos clústeres con la menor disimilitud interclúster entre ellos y se combinan. Así, mientras que en el paso inicial ( $t = 0$ ) existen  $K = n$  clústeres, en el paso  $t$  existirán  $K = n - t$  clústeres. Este proceso se repite hasta que se fusionan los dos últimos clústeres y  $K = 1$ . Nótese que el número de pasos (fusiones) es exactamente  $n - 1$ .

La simpleza de este algoritmo entraña un subprocesso complejo que tiene distintas soluciones: el **cálculo de la disimilitud** entre dos clústeres. Diferentes medidas de disimilitud producen distintas jerarquías de agrupamientos. Dadas dos particiones,  $S_A$  y  $S_B$ , del conjunto de entrenamiento ( $S_A \subset D \wedge S_B \subset D : S_A \cap S_B = \emptyset$ ), la disimilitud entre los clústeres puede medirse teniendo en cuenta todos los ejemplos de cada clúster o solo un representante de cada uno. A continuación, entre la gran variedad de medidas de disimilitud interclúster disponibles, se presentan las más comunes:

- **Disimilitud mínima:** la disimilitud entre dos clústeres viene determinada por el par de ejemplos (uno de cada clúster) con **mayor similitud**.

$$d(S_A, S_B) = \min_{x_a \in S_A; x_b \in S_B} d(x_a, x_b)$$

- **Disimilitud máxima:** el par de ejemplos con **mayor disimilitud** determina la disimilitud entre dos clústeres.

$$d(S_A, S_B) = \max_{x_a \in S_A; x_b \in S_B} d(x_a, x_b)$$

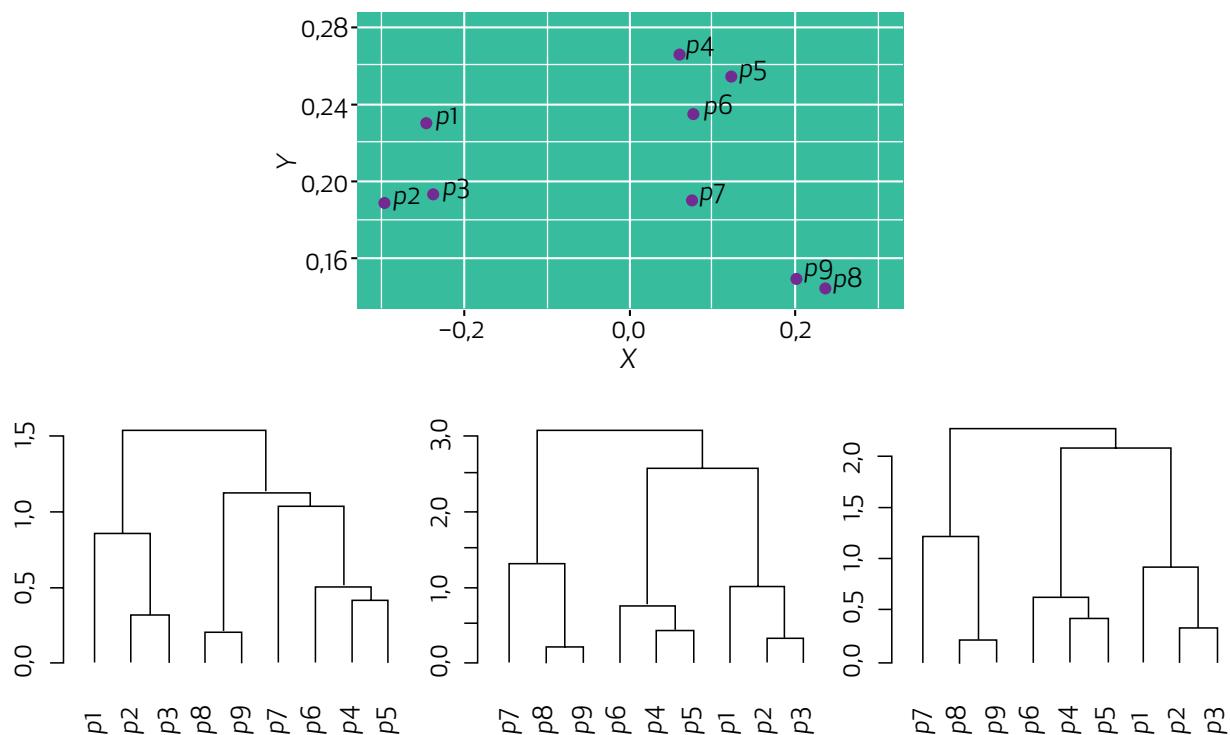
- **Disimilitud media:** la disimilitud entre dos clústeres se define como la **disimilitud media** entre todos los pares de ejemplos (tomando uno de cada clúster).

$$d(S_A, S_B) = \frac{1}{|S_A| + |S_B|} \sum_{x_a \in S_A} \sum_{x_b \in S_B} d(x_a, x_b)$$



Siempre que los datos de entrenamiento están originalmente distribuidos en clústeres bien definidos y compactos, todas estas medidas de disimilitud interclúster usadas para decidir qué par de clústeres se fusionan son **equivalentes**. A medida que estas características ideales se diluyen, los resultados obtenidos con cada una de las métricas divergen.

En la Figura 10 se muestran tres dendrogramas construidos usando las tres medidas presentadas sobre un conjunto de datos simple de ejemplo:



**Figura 10.** Conjunto de datos de prueba y dendrogramas obtenidos con un agrupamiento aglomerativo de disimilitud mínima (izquierda), disimilitud máxima (centro) y disimilitud media (derecha).

El ejemplo etiquetado como  $p_7$  no pertenece claramente a ninguno de los subconjuntos y, por ello, diferentes aproximaciones lo agrupan de manera distinta.

Queda claro de esta manera que, dependiendo del escenario, es decir, dependiendo del tipo de datos de entrenamiento, puede funcionar mejor una u otra medida. Para establecer distintos escenarios, definamos primero el diámetro de un clúster como la disimilitud máxima entre dos elementos del clúster  $S_k$ :

$$d(S_k) = \max_{x_i, x_j \in S_k} d(x_i, x_j)$$

La medida de **disimilitud máxima intraclúster** tiende a producir clústeres compactos con diámetro reducido, ya que es precisamente esa la medida que se busca minimizar. Nótese que la disimilitud máxima intraclúster se convierte, automáticamente después de la fusión de los clústeres, en el diámetro del nuevo clúster. La **disimilitud mínima**, en cambio, tiende a unir clústeres cercanos, aunque no formen una unidad compacta. Se puede pensar en una cadena —donde cada eslabón se une tangencialmente con el siguiente— como situación extrema de una agrupación producto del uso de la disimilitud mínima. En este caso, los casos cercanos tenderán a juntarse en el mismo clúster a costa de un mayor diámetro. Por el contrario, se da la circunstancia de que la disimilitud máxima puede llegar a separar en clústeres diferentes ejemplos muy cercanos. Como se puede apreciar, el uso de una u otra medida lleva a resultados opuestos. En el medio, con la capacidad de generar clústeres relativamente compactos, aunque con casos no necesariamente cercanos, se sitúa la **disimilitud media**.



### Ejemplo

Una empresa que quiere segmentar a su clientela usa un algoritmo de agrupamiento aglomerativo. A la hora de plantearse qué estrategia (medida) de fusión de clústeres emplear, se da cuenta de que se trata de comparar los clientes de los diferentes subgrupos. Los clientes se describen por el número de artículos a la venta que compran de cada tipo.

Una medida de distancia apropiada para comparar dos clientes podría ser la **distancia Manhattan**. Al medir la diferencia entre dos clústeres, la estrategia más natural sería comparar todos los pares posibles de clientes de ambos grupos y calcular la distancia media. En vez de hacer la media, podríamos decidir coger la distancia máxima, es decir, ¿cuál es la distancia entre los dos clientes con un comportamiento de compra más diferente entre ellos? De igual manera, podríamos decidir quedarnos solo con la distancia mínima, es decir, ¿cuál es la distancia entre los dos clientes de cada clúster que muestran un comportamiento de compra más similar?

Así, se definen tres maneras de calcular una **distancia entre dos grupos**. Si en cada paso se van a fusionar los dos subgrupos de clientes con menor distancia entre ellos, el uso de una distancia entre grupos u otra puede resultar en diferentes segmentaciones de la clientela. Si quisiésemos grupos de clientes homogéneos, optaría por una distancia máxima, ya que impone que todos los clientes sean lo más parecidos posible. Si no importa la homogeneidad, pero interesa que en los grupos cada cliente tenga siempre otro de gran parecido, la elección sería la distancia mínima.

Aunque en este documento no entraremos en detalle, otros criterios populares son las **aproximaciones basadas en centroides**, donde cada clúster de la jerarquía es representado por medio de un centroide y las disimilitudes o similitudes entre centroides guían la sucesión de fusiones. Se suele diferenciar entre ponderados y no ponderados según consideren o no el número de ejemplos en cada clúster para relativizar las distancias entre clústeres.



### Enlace de interés

Documentación del algoritmo de agrupamiento jerárquico aglomerativo implementado en la librería scikit-learn para el lenguaje de programación Python:  
<https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html>

## 2.3.2. Aproximación divisiva

La **aproximación divisiva o disociativa** (*top-down* en inglés) comienza creando un único  $K = 1$  clúster, al cual se asignan todos los ejemplos de entrenamiento. A cada paso, se elige un clúster y se divide en dos subgrupos. Solo los clústeres con más de un ejemplo pueden ser divididos. Así, mientras que en el paso inicial ( $t = 0$ ) existen  $K = 1$  clústeres, en el paso  $t$  existen  $K = t + 1$  clústeres. Este proceso se repite hasta que se divide el último clúster de tamaño 2 y  $K = n$ . El número de pasos (divisiones) es igualmente  $n - 1$ . Este valor solo podría variar (reducirse) si en el conjunto de datos de entrenamiento existieran varios casos idénticos (la disimilitud entre ellos sería 0).



Mientras que en la aproximación aglomerativa existía una única pregunta (¿qué dos clústeres se fusionan?), en esta aproximación las **decisiones** a tomar son dos:

- ¿Qué clúster dividir?
- ¿Cómo dividir el clúster seleccionado?

La respuesta a la primera pregunta se puede responder fijando como criterio de selección **el clúster que maximiza la disimilitud media intraclúster**:

$$k = \operatorname{argmax}_{k \in \{1, K^t\}} \frac{1}{|S_k|^2} \sum_{x_i \in S_k} \sum_{x_j \in S_k} d(x_i, x_j)$$

$S_k$  es un clúster y  $K^t$  es el número de clústeres en el nivel  $t$  de la jerarquía. Otra alternativa consiste en usar el clúster de mayor diámetro:

$$k = \operatorname{argmax}_{k \in \{1, K^t\}} \max_{x_i, x_j \in S_k} d(x_i, x_j)$$



La segunda decisión, la que se refiere a **cómo dividir el clúster seleccionado**, es la que confiere a las técnicas divisivas su dificultad característica. La literatura al respecto es relativamente escasa debido a la complejidad que conlleva. Aun así, existen varias propuestas.

Por un lado, se propone usar un algoritmo de agrupamiento basado en particiones ( $K$ -means,  $K$ -medoides, etc.) con  $K = 2$  sobre los casos del clúster que se ha elegido para dividir. De esta manera se obtiene una agrupación de los datos, aunque el coste de ejecutar un algoritmo de este tipo en cada nivel de la jerarquía es alto y muchas veces impráctico. Además, este tipo de algoritmos tiene como inconveniente añadido, tal y como se ha visto en la sección “2.2.1.  $K$ -means”, que son sensibles a la selección inicial de centros o centroides.

Macnaughton-Smith, Williams, Dame y Mockett (1965) propusieron una alternativa. Dado un clúster, se divide inicialmente de tal manera que en el primer nuevo clúster,  $S_A$ , solo hay un ejemplo (el que tiene mayor disimilitud media con el resto de elementos del clúster) y en el otro,  $S_B$ , el resto de ejemplos del clúster original. A partir de ahí, se selecciona iterativamente el elemento  $x$  de  $S_B$  que, en media, está más cercano a los casos de  $S_A$  que al resto de casos de  $S_B$ :

$$x = \operatorname{argmax}_{x_i \in S_B} \left( \frac{1}{|S_B|-1} \sum_{x_j \in S_B: x_i \neq x_j} d(x_i, x_j) - \frac{1}{|S_A|} \sum_{x_j \in S_A} d(x_i, x_j) \right)$$

El caso seleccionado  $x$  se transfiere a  $S_A$ . La transferencia de ejemplos se detiene cuando la diferencia de medias que se busca maximizar es negativa. En este instante, el clúster original habrá sido dividido en dos clústeres nuevos,  $S_A$  y  $S_B$ , que se incluyen en el siguiente nivel de la jerarquía.



### Ejemplo

Siguiendo con el ejemplo anterior de la segmentación de la clientela, la empresa ha decidido usar un algoritmo divisorio. Para ello, a la hora de decidir cómo separar un clúster en dos, aplica la siguiente estrategia:

Para empezar, busca cuál es el cliente que tiene mayor distancia con el resto de clientes del subgrupo (el que tiene un patrón de compras más diferente del resto) y lo separa del resto. Posteriormente, busca si existe algún otro cliente que, en media, esté más cercano al cliente separado que al resto del subgrupo original. Si existe, ese segundo cliente también se separa formando grupo con el primero. Iterativamente, se van buscando clientes que tengan un patrón de compras similar al de los clientes que se han ido separando. Al final, se obtienen dos subgrupos, donde todos los clientes se parecen, en media, más a los otros clientes de su subgrupo que a los del otro subgrupo.

Este proceso se repite hasta que todos los casos están ubicados en su propio clúster o todos los elementos de un clúster tienen disimilitud 0 entre ellos.

### 2.3.3. Adecuación de la jerarquía a la realidad de los datos

En algunos casos, es pertinente estudiar hasta qué punto la jerarquía de clústeres generadas por un algoritmo de agrupamiento jerárquico representa la realidad de los datos. Para estudiar esta adecuación, se utiliza el **coeficiente de correlación cofenético**. Este coeficiente mide la correlación de las disimilitudes calculadas para todos los pares de ejemplos del conjunto de entrenamiento ( $D_{ij} \equiv d(x_i, x_j), \forall x_i, x_j \in D$ ) con la correspondiente disimilitud cofenética ( $F_{ij}$ ).

La disimilitud cofenética se calcula sobre el dendrograma y se define, para dos ejemplos  $x_i$  y  $x_j$ , como la disimilitud interclúster de los nodos hijos de la relación parental en la que  $x_i$  y  $x_j$  pasan a compartir, por primera vez, clúster.

Sin embargo, la medida de disimilitud cofenética es muy restrictiva y genera **muchísimos empates**, lo que ensombrece el cálculo de correlaciones.

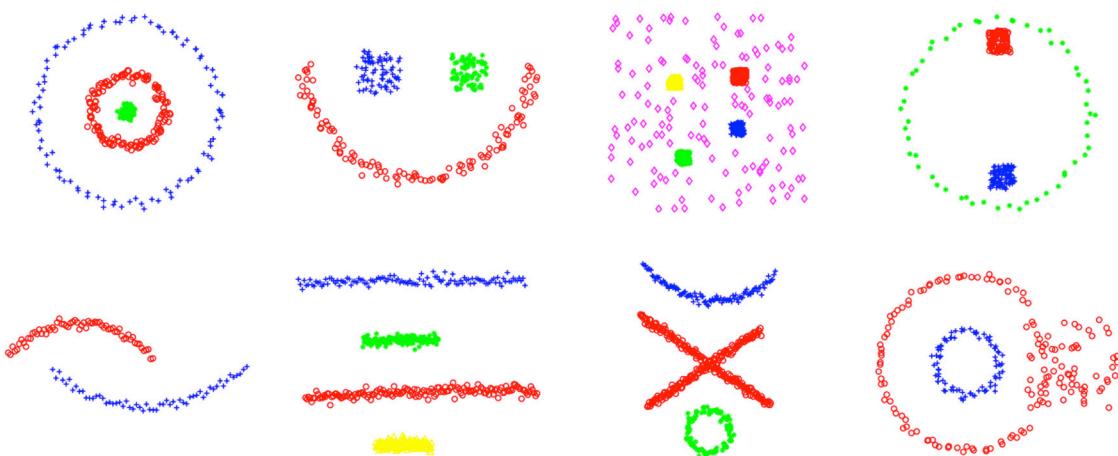
Por eso, este cálculo solo se suele recomendar en casos específicos donde la adecuación de la jerarquía a los datos es importante para el análisis.

En el escenario de estudio habitual, el dendrograma debe presentarse simplemente como la **descripción de la evolución de las particiones** de los datos obtenidas de la aplicación de un algoritmo concreto. A pesar de todo, cuando en la práctica se han aplicado diferentes algoritmos de agrupamiento jerárquico y es necesario quedarse con un único dendrograma, algunos autores recomiendan usar el coeficiente de correlación cofenético para esta elección y seleccionar aquella jerarquía que maximice el coeficiente.

## 2.4. Agrupamiento espectral

Los **algoritmos de agrupamiento tradicionales**, tales como K-means o las mixturas gaussianas, consideran por definición la existencia de clústeres con forma convexa. Es decir, se trata de conjuntos esféricos o elípticos agrupados en torno a un centro o centroide. Las **técnicas de agrupamiento espectral** (*spectral clustering* en inglés), en cambio, transforman el conjunto de datos de entrenamiento y lo representan en espacios alternativos donde se simplifica la identificación de los distintos clústeres sea cual sea su forma.

Así, este tipo de técnicas novedosas permite identificar agrupamientos con formas y densidades tan variadas como las observadas en los ejemplos simples de la Figura 11. Es necesario darse cuenta de que la identificación de los agrupamientos en estos conjuntos de datos de ejemplos mediante técnicas de agrupamiento tradicionales sería difícil o imposible.



**Figura 11.** Diferentes conjuntos de datos para ilustrar el tipo de problemas resolvibles mediante las técnicas de agrupamiento espectral. Recuperado de “Self-Tuning Spectral Clustering”, por L. Zelnik-Manor y P. Perona, 2004, *Advances in Neural Information Processing Systems*, 17, pp. 1601-1608. Disponible en <http://papers.nips.cc/paper/2619-self-tuning-spectral-clustering.pdf>

Un algoritmo de agrupamiento espectral suele constar, a grandes rasgos, de **tres pasos**:

1. Se construye un **grafo de similitudes**, donde cada nodo es un ejemplo de entrenamiento, el cual se une, mediante arcos, a los nodos con mayor similitud, de lo que se obtiene la **matriz laplaciana**.
2. Usando los vectores propios de la matriz laplaciana, los **ejemplos** son transformados a otro **espacio de baja dimensionalidad** donde los clústeres se hacen más evidentes.
3. Se aplica un **algoritmo tradicional de agrupamiento** (por ejemplo, K-means) sobre los datos en este nuevo espacio.



El término **espectral** hace referencia al análisis del espectro de la matriz laplaciana para obtener el agrupamiento. La **matriz laplaciana** es una representación matricial de un grafo que habilita el estudio espectral de este. Entre sus múltiples usos, destaca su utilización para encontrar espacios de dimensionalidad reducida donde representar de manera alternativa un conjunto de datos.

## 2.4.1. Grafo de similitud

Entrando más en detalle, se describe a continuación el **proceso de agrupamiento de los algoritmos espectrales**. Partiendo del conjunto de datos de entrenamiento, se construye una matriz de similitud. Tal y como se explica en la sección “2.1.2. Matriz de distancia o disimilitud”, una matriz de similitud  $S$  es una matriz cuadrada con tantas filas (y columnas) como casos en el conjunto de entrenamiento y donde el valor de cada celda  $S_{ij}$  denota la similitud entre los ejemplos  $x_i$  y  $x_j$ . De manera equivalente, un **grafo de similitud** ( $V, E$ ) no dirigido se define de tal manera que cada ejemplo del conjunto de entrenamiento se representa mediante un nodo en el grafo  $x_i \in V$ . En dicho grafo, dos nodos comparten un arco no dirigido que los conecta,  $e_{ij} \in E$ , si los ejemplos correspondientes,  $x_i$  y  $x_j$ , tienen cierto grado de similitud  $w_{ij} > 0$ . Dado un grafo de similitud deseado, el problema de agrupamiento se puede expresar como un problema de partición de un grafo.

Partiendo de un conjunto de datos y su matriz de similitud  $S$ , existen diferentes **estrategias para crear un grafo de similitud**.

- **Grafo completo.** Una vez que  $V$  contiene un nodo por cada ejemplo de entrenamiento, todo par de nodos-ejemplos ( $x_i, x_j$ ) que tiene un valor positivo en la correspondiente celda de la matriz de similitud,  $S_{ij} > 0$ , se une con un arco,  $e_{ij} \in E$ . El arco  $e_{ij} \in E$  se pesa con el valor de similitud  $w_{ij} = S_{ij}$ .
- **Grafo con poda basada en un umbral.** Dado un valor umbral  $t$  de similitud mínima y el conjunto  $V$  inicializado con un nodo por cada ejemplo de entrenamiento, todo par de nodos-ejemplos ( $x_i, x_j$ ) que tiene un valor mayor que  $t$  en la correspondiente celda de la matriz de similitud,  $S_{ij} > t$ , se une con un arco,  $e_{ij} \in E$ . El arco  $e_{ij} \in E$  se pesa con el valor de similitud  $w_{ij} = S_{ij}$ .
- **Grafo con K vecinos más cercanos** (K-NN, por sus siglas en inglés). Dado un valor  $K$  (número de vecinos) y el conjunto  $V$  inicializado con un nodo por cada ejemplo de entrenamiento, se añade un arco,  $e_{ij} \in E$ , para todo par de nodos-ejemplos ( $x_i, x_j$ ) tal que  $x_i$  es uno de los  $K$  vecinos más cercanos de  $x_j$  o  $x_j$  lo es de  $x_i$ . El arco  $e_{ij} \in E$  se pesa con el valor de similitud  $w_{ij} = S_{ij}$ . En la práctica, esta es probablemente la técnica más habitual para obtener el grafo a partir de la matriz de similitud.

## 2.4.2. Matriz laplaciana

A la hora de **construir la matriz laplaciana**, es necesario introducir el concepto de matriz de pesos de los arcos,  $W$ . En este caso, cada celda toma como valor el peso del arco correspondiente,  $w_{ij}$ , y toma valor nulo si el arco no existe,  $e_{ij} \notin E \rightarrow w_{ij} = 0$ . En un grafo, el grado de un nodo se puede definir como el número de arcos que lo conectan a otros nodos del grafo. En el caso de un grafo con pesos, como en este razonamiento, el grado de un nodo es la suma de los pesos de los arcos que lo conectan a otros nodos del grafo,  $d_i = \sum_{i': e_{ii'} \in E} w_{ii'}$ .  $D$  se define como la matriz diagonal que tiene en cada elemento de su diagonal principal el grado de un nodo,  $g_i$ . Dados  $W$  y  $D$ , definimos la matriz laplaciana de la siguiente forma:

$$L = D - W$$

Hablamos de la matriz laplaciana **normalizada** cuando  $L$  se estandariza con respecto al grado de los nodos:

$$\tilde{L} = I - D^{-1}W$$

En la fórmula anterior,  $I$  es la matriz identidad.

**Tabla 4**

Pseudocódigo del algoritmo de agrupamiento espectral con matriz laplaciana no normalizada

**Agrupamiento espectral**

Recibe: conjunto de entrenamiento,  $\{x_1, \dots, x_n\}$ ; número de clústeres,  $K$ ; número de vectores propios,  $m$ .

1. Construir el grafo de similitud y calcular las matrices  $W$  y  $D$ .
2. Obtener la matriz laplaciana no normalizada:  $L = D - W$ .
3. Obtener los  $m$  vectores propios con menores valores propios y construir la matriz  $Z$ .
4. Aplicar un algoritmo tradicional de agrupamiento (como K-means) sobre  $Z$  para identificar los  $K$  clústeres.

Devuelve: agrupamiento resultante del punto 4.

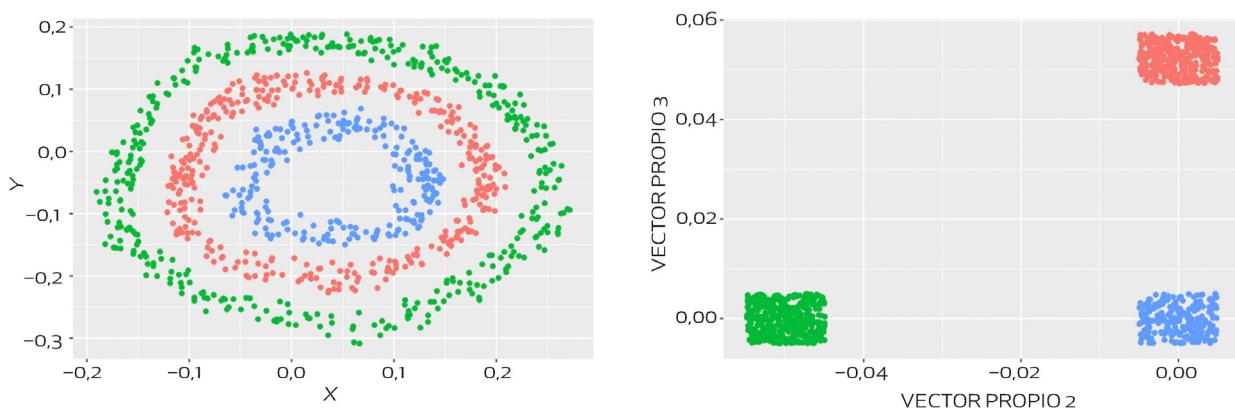
### 2.4.3. Transformación de los datos en un espacio alternativo

El siguiente paso consiste en obtener los **valores y vectores propios de la matriz laplaciana**, que se ordenan por valor creciente de valor propio.

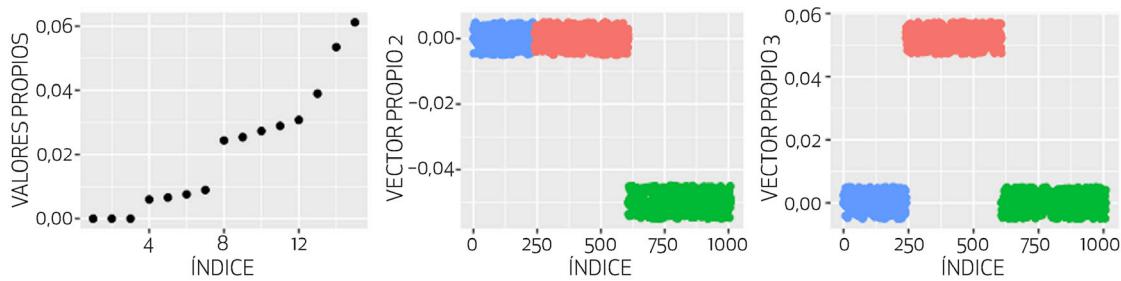
El primer valor y vector propio se obvia porque no aporta información (el vector propio es constante).

El algoritmo selecciona los siguientes  $m$  vectores propios, aquellos con los valores propios más pequeños. Estos vectores propios se combinan formando una matriz  $Z$  de tamaño  $n \times m$ , que puede considerarse una transformación del conjunto de datos original. Se espera que esta transformación conduzca a una distribución del conjunto de datos en la que sea más sencillo separar los elementos de los diferentes clústeres.

Se puede ver un ejemplo de transformación ( $m = 2$ ) sobre un conjunto de datos sintéticamente generado para la ocasión en la Figura 12 (en la que se usan tres colores diferentes para resaltar los clústeres originales), y la visualización por separado de los vectores propios considerados en este proceso en la Figura 13.



**Figura 12.** Conjunto de datos para ilustrar la capacidad del agrupamiento espectral (izquierda) y su visualización tras la transformación usando los dos primeros vectores propios (sin contar el nulo) de la matriz laplaciana no normalizada (derecha).



**Figura 13.** Detalles del proceso de transformación de la Figura 12: primeros 15 valores propios (izquierda), división del conjunto por medio del segundo vector propio (centro) y división del conjunto por medio del tercer vector propio (derecha).

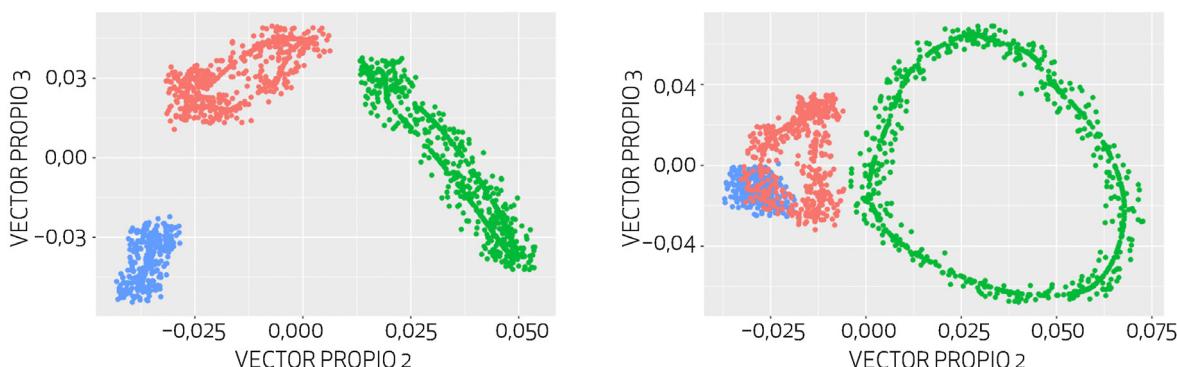
Sobre la matriz  $Z$  (el conjunto de datos representado en otro espacio) se aplica, finalmente, un algoritmo de agrupamiento clásico (normalmente, K-means) para obtener el agrupamiento final. En la Tabla 4 se muestra el pseudocódigo completo del algoritmo. En el ejemplo simplista de la Figura 12, se puede observar que la transformación ofrece un conjunto  $Z$  mucho más apropiado para la identificación de clústeres por medio del algoritmo K-means que el conjunto original, sobre el cual el algoritmo K-means no sería capaz de separar correctamente los clústeres.

Cuando se usa la matriz laplaciana normalizada, este método se puede interpretar desde la **teoría de los saltos aleatorios** (*random walks* en inglés). Un camino de saltos aleatorios es una cadena de Markov caracterizada por una matriz  $P$  cuadrada ( $n \times m$ ) donde cada celda  $P_{ij}$  denota la probabilidad, en cada momento, de saltar del nodo  $i$  al nodo  $j$ . En el caso del agrupamiento espectral, se puede definir esta matriz  $P$  tal que el valor de cada celda es  $P_{ij} = w_{ij}/d_i$  (matricialmente,  $P = D^{-1}W$ ). Si considerásemos dos subconjuntos de ejemplos,  $A$  y  $\bar{A}$ , la probabilidad de transitar entre subconjuntos se podría calcular de la siguiente manera:  $P(A, \bar{A}) = P_{A\bar{A}} + P_{\bar{A}A}$ .

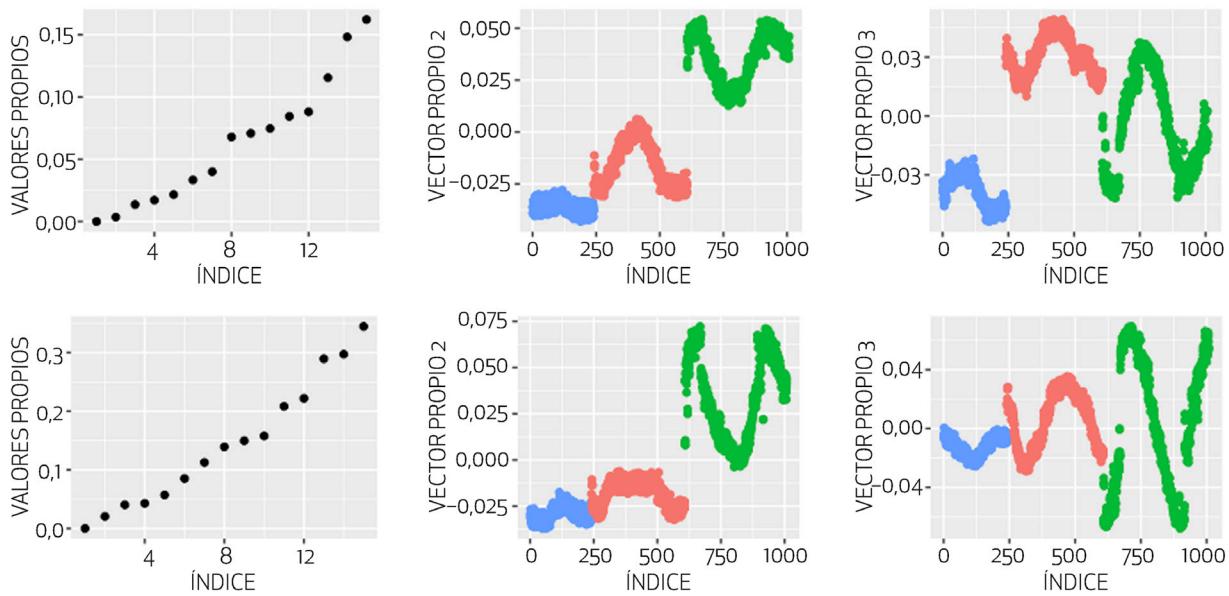
En la fórmula anterior:

$$P_{A\bar{A}} = \sum_{x_i \in A, x_j \in \bar{A}} w_{ij} / \sum_{x_i \in A} d_i$$

Así pues, el objetivo de un algoritmo de agrupamiento espectral podría replantearse como la **búsqueda de un grupo de subconjuntos de nodos** (que, en nuestro caso, representan ejemplos del conjunto de entrenamiento) **tal que la probabilidad de transitar entre ellos sea mínima**.



**Figura 14.** Transformaciones obtenidas a partir del conjunto de datos mostrado en la Figura 12 usando diferentes valores de  $K$  en el subproceso K-NN: izquierda,  $K = 9$ ; derecha,  $K = 12$ ; subfigura 12 derecha,  $K = 6$ .



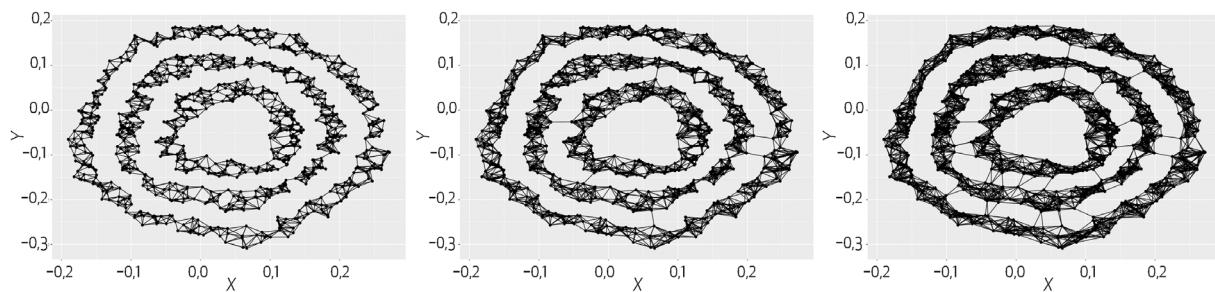
**Figura 15.** Detalles del proceso de transformaciones de la Figura 14: primeros 15 valores propios (izquierda), división del conjunto por medio del segundo vector propio (centro) y división del conjunto por medio del tercer vector propio (derecha). Las subfiguras de la primera fila corresponden al proceso con  $K=9$ , las de la segunda fila al proceso con  $K=12$ , y las de la Figura 13 al proceso con  $K=6$ .

Son varios los **problemas y decisiones** que se deben afrontar a la hora de aplicar técnicas de agrupamiento espectral. Por un lado, se debe elegir una función de **similitud y su parametrización** (por ejemplo, el valor de  $c$  en la transformación de  $M$  en  $S$  propuesta en la sección “2.1.2. Matriz de distancia o disimilitud”).

También se ha de seleccionar el **procedimiento para obtener un grafo de similitud y sus respectivas parametrizaciones**: el valor del umbral  $t$  si se considera una similitud mínima en la matriz de similitudes, o el número de vecinos  $K$  si se aplica el método basado en el K-NN. En las Figuras 14 y 15, se puede observar cómo resultaría el agrupamiento destallado en las Figuras 12 y 13 si se usase otro número de vecinos (véanse también los grafos de adyacencia subyacentes a este proceso en la Figura 16).

Otra elección característica de estas técnicas es la del **número de vectores propios  $m$  de la matriz laplaciana** considerados para generar el conjunto transformado  $Z$ . Una propiedad de la matriz laplaciana es que el número de valores propios iguales a 0 es igual al número de subconjuntos conectados (existe un camino en el grafo que une dos puntos cualesquiera del subconjunto). Este comportamiento se puede ver en la subfigura izquierda de las Figuras 13 y 15, donde 3 subconjuntos conectados se traducen en 3 valores propios a 0. Sin embargo, esta separación ideal no suele observarse en los problemas de agrupamiento con datos reales. En estos casos, una técnica habitual consiste en observar si en la ilustración de los valores propios más pequeños (véase la subfigura izquierda de las Figuras 13 y 15) se encuentra un salto pronunciado entre dos valores consecutivos y establecer a partir de ello el valor de  $m$ .

Finalmente, como en la mayoría de técnicas de agrupamiento, se debe seleccionar el **número de clústeres o cualquier otro parámetro** que requiera el método que se haya elegido para realizar el agrupamiento sobre los datos transformados.



**Figura 16.** Grafos de adyacencias obtenido tras aplicar el subproceso K-NN sobre el conjunto de datos de la Figura 12 con el uso de diferentes números de vecinos: 6, 9 y 12, de izquierda a derecha.



### Enlace de interés

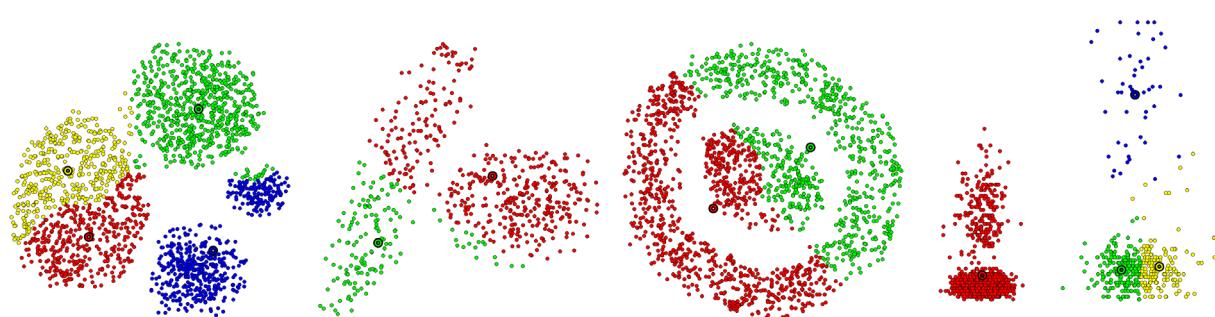
Documentación del algoritmo de agrupamiento espectral implementado en la librería scikit-learn para el lenguaje de programación Python:

<https://scikit-learn.org/stable/modules/generated/sklearn.cluster.SpectralClustering.html>

## 2.5. Agrupamiento basado en densidad

La mayoría de **algoritmos de agrupamiento** asumen la existencia de una distribución de probabilidad desde la cual se han muestreado los ejemplos del conjunto de entrenamiento. Tanto el algoritmo esperanza-maximización (véase la sección “2.6.2. Algoritmo esperanza-maximización”), de manera explícita, como el K-means (véase la sección “2.2.1. K-means”), de una forma más implícita, asumen la existencia de una distribución de probabilidad (mixtura) gaussiana. Con esta distribución en concreto, por definición, **se asume que los clústeres son esféricos o elípticos**.

Suposiciones como esta se alejan de la **realidad, donde los clústeres suelen mostrar formas arbitrarias**: desde clústeres con otro tipo de siluetas hasta escenarios donde la frontera entre agrupaciones se difumina. La Figura 17 ilustra algunos ejemplos sencillos de conjuntos de datos de dos dimensiones donde el algoritmo K-means no es capaz de detectar los clústeres que se observan.



**Figura 17.** Cuatro conjuntos de datos donde el algoritmo K-means no es capaz de detectar los clústeres originales.



Los **algoritmos basados en densidad** surgieron de la necesidad de encontrar agrupamientos de formas diversas. Se basan en la idea de descubrir subconjuntos de datos que, sea cual sea la forma que proyectan, tienen cierta continuidad a lo largo del espacio. La continuidad se define en términos de densidad de ejemplos: un clúster es un conjunto denso representado en un área del espacio de los datos y rodeado por áreas menos densamente pobladas. Esta idea permite construir técnicas, tanto paramétricas como no paramétricas, en que el número de clústeres no se designa *a priori* sino que se descubre.

### 2.5.1. Algoritmo DBSCAN

El algoritmo de agrupamiento basado en densidad más conocido es DBSCAN (Ester, Kriegel, Sander y Xu, 1996). El concepto de densidad usado por el algoritmo se define por medio de **dos parámetros,  $\mathcal{E}$  y  $M$** . Una parametrización concreta de estos fija la densidad mínima deseada en los clústeres resultantes. El parámetro  $\mathcal{E}$  fija un umbral que permite definir el **vecindario** de un ejemplo  $x_i$ : cualquier otro ejemplo  $x_j$  del conjunto de datos  $D$  cuya distancia con el ejemplo  $x_i$  sea menor que  $\mathcal{E}$ :

$$\text{neigh}_{\mathcal{E}}(x_i) = \{x_j \in D : \text{dist}(x_i, x_j) \leq \mathcal{E}\}$$

Un par de ejemplos se dice que están conectados si cada uno de ellos pertenece al vecindario del otro. De manera equivalente, un ejemplo del conjunto de entrenamiento es un **punto nuclear** si dentro de su vecindario hay al menos  $M$  ejemplos. Cualquier punto  $x_j$  en el vecindario de un punto nuclear  $x_i$  se dice que es **directamente densoalcanzable** desde  $x_i$  (definido a partir de  $\mathcal{E}$ ). Hay dos condiciones implícitas en esta definición:

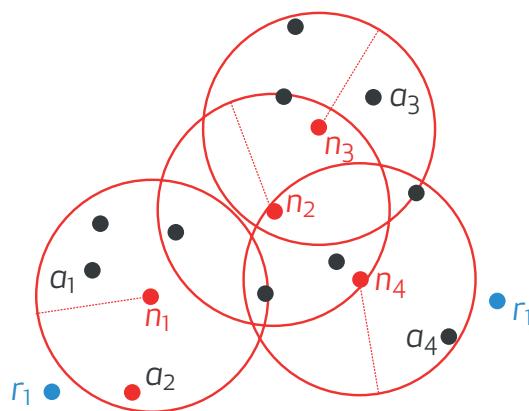
$$d^2\text{Reach}_{x_i}(x_j) \equiv (x_j \in \text{neigh}_{\mathcal{E}}(x_i)) \wedge (|\text{neigh}_{\mathcal{E}}(x_i)| \geq M)$$

Un punto  $x_j$  es **densoalcanzable** desde  $x_i$  si se puede encontrar un camino de puntos  $x_1, \dots, x_p$  tal que  $x_{k+1}$  sea directamente densoalcanzable desde  $x_k$  (para todo  $k \in \{1, \dots, p\}$ ) donde  $x_1 = x_i$  y  $x_p = x_j$ . Para que este camino sea posible, todos los puntos aparte de  $x_j$  deben ser necesariamente puntos nucleares. Precisamente este hecho hace que la relación densoalcanzable no **sea necesariamente simétrica**: aunque  $x_j$  sea **densoalcanzable** desde  $x_i$ , si  $x_j$  no es un punto nuclear, por definición el punto  $x_i$  no es **densoalcanzable** desde  $x_j$ .

En su lugar, se dice que dos puntos,  $x_i$  y  $x_j$ , están **densoconectados** si se puede encontrar un camino  $x_1, \dots, x_p$  tal que  $x_i$  sea directamente densoalcanzable desde  $x_1$ ,  $x_j$  lo sea desde  $x_p$ , y  $x_{k+1}$  lo sea desde  $x_k$  (para todo  $k \in \{1, \dots, p\}$ ).

Nótese que, en este caso, el concepto de densoconectado **sí es simétrico**. Así, un clúster se define como un subconjunto de ejemplos densoconectados que es máximo (es decir, no existe ningún punto densoconectado con los elementos del grupo y que no pertenezca a este).

La Figura 18 trata de ilustrar gráficamente todos los conceptos claves de este algoritmo:

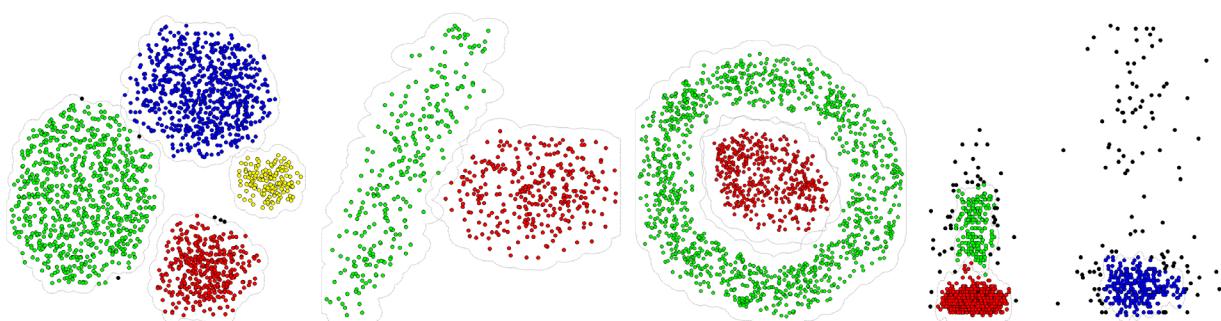


**Figura 18.** Ilustración de los conceptos básicos del algoritmo DBSCAN ( $M = 5$ ).

Los ejemplos rojos son puntos nucleares y las circunferencias representan el área de su vecindario ( $\mathcal{E}$ ). Los puntos azules son ruido: no son densoalcanzables desde ningún núcleo. El punto nuclear  $n_1$  no es densoalcanzable desde ningún otro núcleo, lo que genera dos clústeres: uno alrededor de  $n_1$  y otro alrededor de los puntos nucleares  $n_2$ ,  $n_3$  y  $n_4$ , que sí son densoalcanzables entre sí. Los puntos negros son puntos que pertenecen a algún clúster pero no son núcleos. Por ejemplo,  $a_3$  y  $a_4$  (o  $a_1$  y  $a_2$ ) están densocorrelados, pero no así  $a_2$  y  $a_4$ .



Dado un conjunto de datos, el algoritmo DBSCAN identifica clústeres que cumplen estas características fijadas los parámetros  $\mathcal{E}$  y  $M$ . Una característica de este algoritmo es que aquellos **puntos alejados del resto**, en zonas de baja densidad, **pueden no quedar asignados a clústeres**. A diferencia de la mayoría de métodos de agrupamiento, se considera que esos puntos son producto del ruido y que no forman parte realmente de ningún agrupamiento.



**Figura 19.** Los cuatro conjuntos de datos de la Figura 17 tras la aplicación de DBSCAN.

Tal y como puede observarse en la Figura 19, DBSCAN es un algoritmo capaz de identificar, con la parametrización adecuada, clústeres de diferentes formas. En este caso, su rendimiento solo es pobre para el conjunto de la derecha, donde no es capaz de detectar subconjuntos de diferente densidad.

Uno de los problemas principales de DBSCAN es su **debilidad a la hora de tratar con clústeres de diferente densidad**. En la subfigura 19 del extremo derecho se puede observar que, ante un conjunto de datos con 4 clústeres de diferente densidad, el algoritmo acaba detectando como ruido la mayoría de los ejemplos de los clústeres más dispersos.

Por otro lado, el proceso de **cálculo de los vecindarios** puede también ser complejo y lastrar el rendimiento del algoritmo. Existen diferentes implementaciones que agilizan el cómputo según las características de los datos. La implementación más sencilla puede llevarse a cabo siguiendo el pseudocódigo de la Tabla 5:

**Tabla 5**

Pseudocódigo del algoritmo de agrupamiento basado en densidad DBSCAN

---

### DBSCAN

---

Recibe: conjunto de entrenamiento,  $\{x_1, \dots, x_n\}$ ; distancia  $\mathcal{E}$ ; número mínimo de vecinos  $M$ .

---

1.  $C = 1$ .

2. Para todo ejemplo,  $x_i$ :

2.1. Si  $x_i$  ya está asignado, continuar.

2.2. Calcular vecindario  $V$  de  $x_i$  (dado  $\mathcal{E}$ ).

2.3. Si  $|V| < M$ , asignar  $x_i$  como ruido y continuar.

2.4. Crear el clúster número  $C$  y asignarle  $x_i$ .

2.5. Para todo ejemplo  $x_j \in V$ .

2.5.1. Si  $x_j$  está asignado como ruido, asignarlo al clúster  $C$  y continuar.

2.5.2. Si  $x_j$  tiene otra asignación, continuar.

2.5.3. Asignar  $x_j$  al clúster  $C$ .

2.5.4. Calcular el vecindario  $V'$  de  $x_j$  (dado  $\mathcal{E}$ ).

2.5.5. Si  $|V'| \geq M$ , entonces  $V = V \cup V'$ .

2.6.  $C = C + 1$ .

---

Devuelve: agrupamiento resultante.

---

Las dos **condiciones básicas** de este algoritmo, las que definen un punto nuclear, pueden ser **generalizadas a conceptos más abstractos**. Así, podrían considerarse otros conceptos de vecindario de un punto  $x_i$  (no solo según la distancia y el umbral  $\mathcal{E}$ ). Igualmente, la cardinalidad del vecindario podría calcularse de otra manera (por ejemplo, con la suma ponderada respecto a la distancia al núcleo).



#### Enlace de interés

Documentación del algoritmo DBSCAN implementado en la librería scikit-learn para el lenguaje de programación Python:

<https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html>

## 2.5.2. Algoritmo de desplazamiento de media

El algoritmo de agrupamiento conocido como **desplazamiento de media** (*mean-shift* en inglés) es una técnica que busca las modas (máximos locales) de la función de densidad que se observa a partir del conjunto de datos.

Se trata de un procedimiento que iterativamente substituye cada punto del conjunto de entrenamiento por el **punto medio (ponderado) de sus vecinos** (Comaniciu y Meer, 2002). Tanto el vecindario como los pesos que se asignan a cada vecino vienen determinados por la **función núcleo** (*kernel* en inglés) que el usuario haya elegido.

Dependiendo del núcleo elegido, los clústeres resultantes tenderán a ser mayores o menores. El método tiene un parámetro  $h$  (radio) que permite controlar la anchura del núcleo. Un núcleo estrecho genera muchos clústeres pequeños, y a medida que se usa un núcleo más ancho, el número de clústeres se reduce y su tamaño se incrementa.

Se puede demostrar que este procedimiento iterativo converge al punto estacionario más cercano de la función de densidad subyacente. El método no necesita que se le especifique el número de clústeres  $K$ , ya que este es producto del aprendizaje.

La propuesta original (Comaniciu y Meer, 2002) usa el **estimador de densidad tipo núcleo de Parzen**. El estimador tipo núcleo de Parzen de la función de densidad  $f(x)$  se define matemáticamente de la siguiente manera:

$$\hat{f}(x) = \frac{1}{n \cdot h^v} \sum_{i=1}^n k\left(\frac{x - x_i^2}{h^2}\right)$$

En la fórmula anterior,  $a^2 = \sum_{j=1}^v |a_j|^2$ . Desde cada punto  $x_i$ , se aplica un método de ascenso de gradiente (u otro método de optimización equivalente) para buscar los puntos máximos. Ya que la media se desplaza siempre hacia el punto de mayor incremento de la densidad, se dibuja un camino hacia un punto estacionario de máxima (local) densidad. Estos puntos, donde el algoritmo converge, los máximos locales de la función de densidad, son los representantes de cada clúster. Aquellos ejemplos cuyo camino conduce al mismo máximo local pertenecerán al mismo clúster.

En la Tabla 6 se presenta el pseudocódigo del algoritmo:

**Tabla 6**

Pseudocódigo del algoritmo de agrupamiento de desplazamiento de media

**Desplazamiento de media**

Recibe: conjunto de entrenamiento,  $\{x_1, \dots, x_n\}$ ; radio  $h$ ; kernel  $k$ .

1. Repetir hasta convergencia:  $m(x_i) = x_i, \forall x_i$ .
  - 1.1. Para todo ejemplo  $x_i$ :  $x_i \leftarrow m(x_i)$ .
2. Identificar modas: puntos a los que varios casos originales convergen.
3. Asignar todos los ejemplos que convergen a la misma moda al mismo clúster.

Devuelve: agrupamiento resultante.

La función  $m(x)$  devuelve la media de los puntos del vecindario de  $x$ . La diferencia  $m(x) - x$  es el desplazamiento de la media que da nombre al algoritmo. En cada iteración  $t$ , cada ejemplo  $x$  del conjunto de entrenamiento se actualiza:  $x^{(t)} \leftarrow m(x^{(t-1)})$ . El algoritmo se detiene (es decir, la convergencia se alcanza) cuando  $m(x^{(t)}) = x^{(t)}$ .

La clave del algoritmo se encuentra, como se puede entrever, en el **cálculo de la media**. Un ejemplo  $x$  se actualiza (desplaza) con la media ponderada de su vecindario,  $m(x)$ , que se calcula de la siguiente manera:

$$m(x) = \frac{\sum_{i=1}^n k\left(\frac{x - x_i^2}{h^2}\right) \cdot x_i}{\sum_{i=1}^n k\left(\frac{x - x_i^2}{h^2}\right)}$$

La **función núcleo**  $k$  permite asignar un peso a cada uno de los vecinos. Las funciones núcleo más habituales son:

- Núcleo plano (donde, normalmente, el umbral toma valor  $\lambda = 1$ ):

$$k(x) = \begin{cases} 1, & x \leq \lambda \\ 0, & x > \lambda \end{cases}$$

- Núcleo gaussiano:

$$k(x) = e^{-x}$$

El efecto del **parámetro radio**,  $h$ , es diferente según el núcleo considerado. En el caso del núcleo plano (con  $\lambda = 1$ ),  $h$  determina el umbral, ya que:

$$x \leq \lambda = 1; \frac{x - x_i^2}{h^2} \leq 1; x - x_i^2 \leq h^2$$

Por lo tanto, se podría definir el **vecindario** de  $x$  con respecto a  $h$ :

$$V_h(x) = \{x_j \in D : x - x_j^2 \leq h^2\}$$

Así, es posible reducir el **cómputo de la función**  $m(x)$  a lo siguiente:

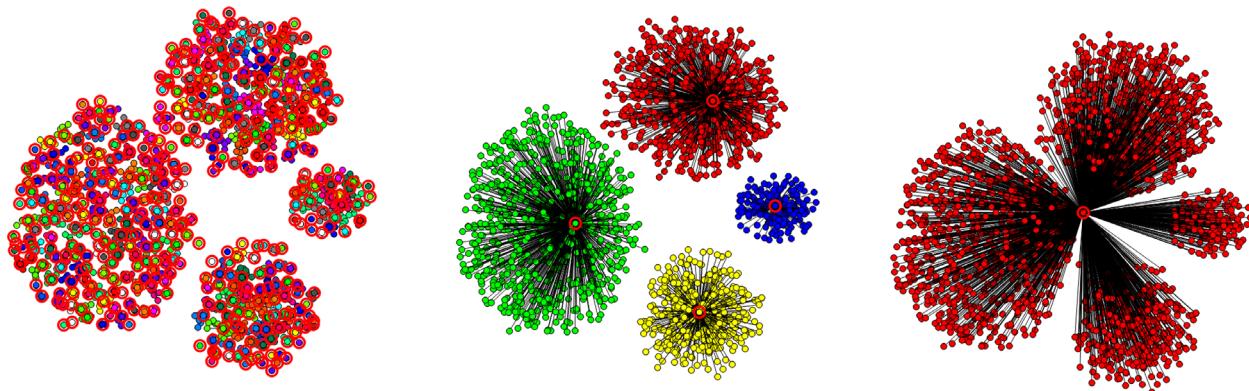
$$m(x) = \frac{\sum_{x_i \in V_h(x)} k\left(\frac{x - x_i^2}{h^2}\right) \cdot x_i}{\sum_{x_i \in V_h(x)} k\left(\frac{x - x_i^2}{h^2}\right)} = \frac{\sum_{x_i \in V_h(x)} x_i}{|V_h(x)|}$$

En la Figura 20 se puede observar el **efecto del parámetro**  $h$  en los resultados. Si se usa el núcleo gaussiano, el parámetro radio,  $h$ , funciona como la desviación típica,  $h = \sqrt{2}\sigma$ :

$$e^{-x} = e^{-\frac{x_j - x_j^2}{h^2}} = e^{-\frac{x_j - x_j^2}{2\sigma^2}}$$

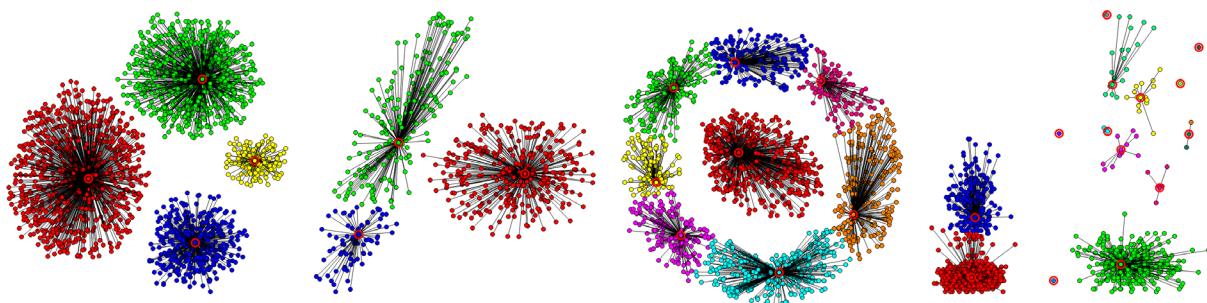
En este caso, cualquier punto  $x'$  del conjunto de entrenamiento cuenta en el cálculo de la función  $m(x)$ , pero su contribución a la media decrece de manera proporcional a su distancia con respecto al punto  $x$ .

La rapidez de la caída depende del parámetro  $h = \sigma$ .



**Figura 20.** Efecto de distintos parámetros  $h$  del núcleo en un mismo conjunto de datos.

En la izquierda, un ancho pequeño provoca la selección de múltiples modas (clústeres). En el centro, un ancho adecuado permite identificar los 4 clústeres. En la derecha, un ancho demasiado grande no da pie a la separación en clústeres.



**Figura 21.** Los cuatro conjuntos de datos de la Figura 17 agrupados según el algoritmo de desplazamiento de media.

Como se puede observar, el algoritmo divide repetidamente clústeres naturales en dos o varios clústeres. Por el contrario, es capaz de detectar algún tipo de estructura en el clúster menos denso del conjunto de la derecha.



Al igual que DBSCAN, el desplazamiento de media no considera ninguna forma predeterminada para los clústeres. Aparte del núcleo, el único parámetro del método es el **valor del radio**,  $h$ . Al contrario que el parámetro  $K$  (número de clústeres) en algoritmos como K-means,  $h$  es un parámetro intuitivo que permite definir el vecindario y el peso de los vecinos en el cálculo de la media.

Sin embargo, este algoritmo tiene algunos **inconvenientes**. En primer lugar, la elección del **valor de  $h$**  no es trivial y puede conducir a resultados muy dispares (véase la Figura 20).

Además, de acuerdo con los resultados ilustrados en la Figura 21, se observa que el desplazamiento de media es capaz de encontrar los **clústeres**, pero estos no son máximos, es decir, separa en varios clústeres lo que realmente es solo uno (véanse las dos subfiguras centrales de la Figura 21).

Por otro lado, muchos autores consideran que el algoritmo de desplazamiento de media **no es un método basado en densidad**. Argumentan que la asignación de ejemplos a clústeres no es absoluta (pertenece o no) sino que es ponderada (pertenece a cierto clúster con cierto peso) y, sobre todo, no existe garantía de que los clústeres sean conjuntos de puntos conectados.

Otro punto cuestionable es que no existe una **demostración formal de la convergencia** del algoritmo usando un núcleo general en entornos multidimensionales. Solo existen pruebas de convergencia en entornos controlados (espacio unidimensional o número finito de puntos estacionarios).

#### Enlace de interés

Documentación del algoritmo de desplazamiento de media implementado en la librería scikit-learn para el lenguaje de programación Python:

<https://scikit-learn.org/stable/modules/generated/sklearn.cluster.MeanShift.html>

### 2.5.3. Propagación de afinidad

Otro algoritmo de agrupamiento recientemente propuesto (Frey y Dueck, 2007) que se está abriendo un hueco en el estado del arte es el **algoritmo de propagación de afinidad** (*affinity propagation*, AP). Está basado en el concepto de pase de mensaje y tiene la ventaja de que no necesita conocer el número de clústeres  $K$  para operar.

Trabaja sobre la **matriz de similitud** (véase la sección “2.1.2. Matriz de distancia o disimilitud”), donde cada celda de la matriz,  $s(x_i, x_k)$ ,  $\forall i, k \in \{1, \dots, n\}$ , mide la similitud entre dos puntos del conjunto de datos de entrenamiento. La matriz de similitudes se modifica parcialmente para acomodar las **preferencias**, es decir, la evidencia de que un punto cualquiera  $x_k$  pueda ser el representante del clúster al que pertenezca.

Concretamente, las celdas de la diagonal principal,  $s(x_k, x_k)$ —por definición, todos los valores son 1—, se modifican para que tomen un valor proporcional a la posibilidad de que el punto  $x_k$  pueda ser un representante (concepto equivalente a centroide). Normalmente, es el usuario quien decide cómo calcular las preferencias.



La **magnitud de las preferencias** tiene también su **impacto en el algoritmo**: valores altos (como la mediana de las similitudes del ejemplo  $x_k$ ) inducen un mayor número de clústeres en el agrupamiento resultante, mientras que valores más bajos (como el valor mínimo de similitud de  $x_k$ ) tienden a producir una agrupación en pocos clústeres.

El algoritmo de propagación de afinidad se basa en dos conceptos teóricos con una definición matemática basada en la similitud: la responsabilidad y la disponibilidad. El primer concepto, la **responsabilidad**, mide, para un punto cualquiera  $x_i$ , la evidencia acumulada de que otro punto  $x_k$  es el representante ideal de  $x_i$  (con respecto a otros posibles representantes  $x_{k'}$ ).

Por otro lado, la **disponibilidad** mide la evidencia acumulada de lo apropiado que sería que  $x_i$  escogiese  $x_k$  como su representante, teniendo en cuenta que otros puntos también consideran que  $x_k$  debe ser representante. Matemáticamente, estos conceptos se calculan de la siguiente manera. La responsabilidad es:

$$r(x_i, x_k) = s(x_i, x_k) - \max_{k' \neq k} (a(x_i, x_{k'}) + s(x_i, x_{k'}))$$

Por otro lado, la disponibilidad se obtiene aplicando la siguiente fórmula:

$$a(x_i, x_k) = \begin{cases} \min\left(0; r(x_k, x_i) + \sum_{i' \in [1, \dots, n]: i' \neq i} \max(0; r(x_{i'}, x_k))\right), & i \neq k \\ \sum_{\substack{i' \in [1, \dots, n]: \\ i' \neq k}} \max(0; r(x_{i'}, x_k)), & i = k \end{cases}$$

Se puede ver que un concepto se define en términos del otro, lo que nos transporta a un **entorno necesariamente iterativo**. El algoritmo se inicializa con la matriz de similitud adaptada para acomodar las preferencias, y todas las disponibilidades se ponen a cero:  $a(x_i, x_k) = 0, \forall i, k \in \{1, \dots, n\}$ . A continuación, las dos ecuaciones de estimación de la responsabilidad y la disponibilidad se iteran alternativamente. Un caso con mayor preferencia inicial tiene más posibilidades de convertirse en representante de su clúster. Si dos casos similares se consideran a sí mismos buenos candidatos para representantes,  $r(x_k, x_k)$ , se establece una competición entre ambos hasta que en cierta iteración  $t$  uno de los dos se impone.

Los representantes de cada clúster son los puntos que tienen **responsabilidad y disponibilidad hacia ellos mismos positiva**:

$$R \subset \{1, \dots, n\}: \forall x_k \in R, (r(x_k, x_k) + a(x_k, x_k)) > 0$$

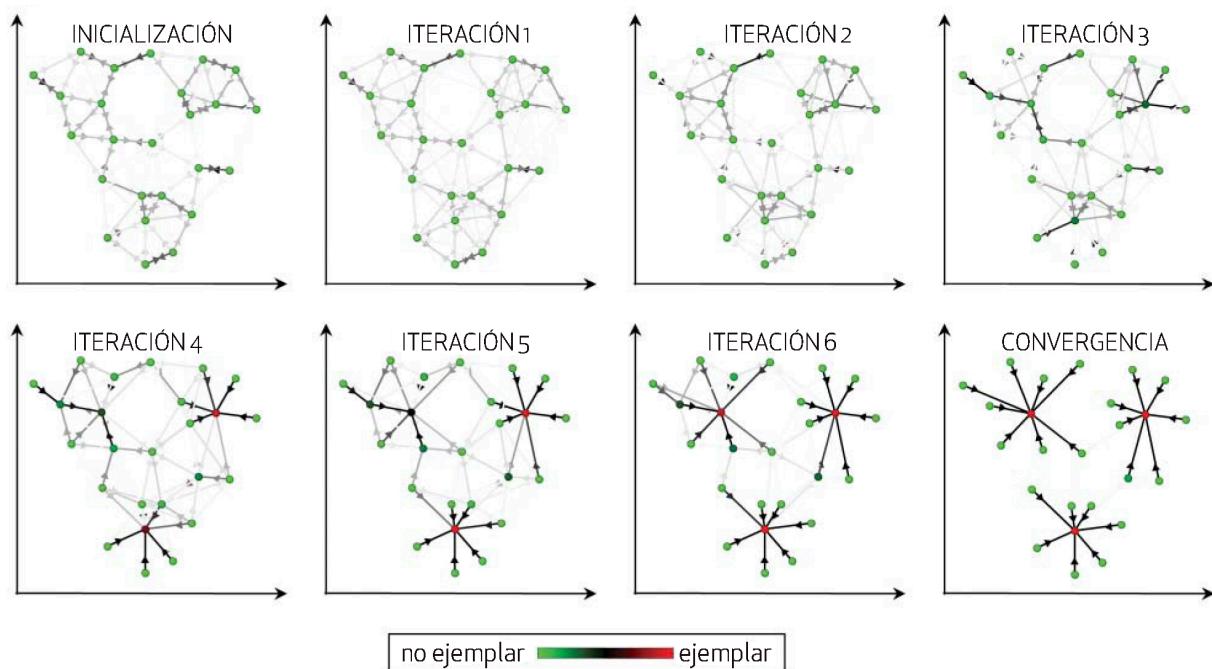
El **número de clústeres** viene determinado por el número de representantes encontrados. En cualquier momento, se puede calcular a qué clúster  $k$  (representado por  $x_k$ ) está asignado un ejemplo  $x_i$  como aquel cuyo representante  $x_k$  maximiza la suma de la responsabilidad y la disponibilidad:

$$C(x_i) = \operatorname{argmax}_k (r(x_i, x_k) + a(x_i, x_k))$$

El **procedimiento iterativo se detendrá** cuando se cumpla una de estas condiciones:

- Se alcanza un número de iteraciones máximo.
- La diferencia entre los valores de responsabilidad y disponibilidad de dos iteraciones consecutivas cae por debajo de un umbral  $U$ .
- La asignación a clústeres no cambia durante varias iteraciones consecutivas.

- En la figura siguiente puede observarse un ejemplo de ejecución del algoritmo sobre un conjunto de datos de ejemplo:



**Figura 22.** Ejecución del algoritmo de propagación de afinidad. Adaptado de "Clustering by Passing Messages Between Data Points" por B. J. Frey y D. Dueck, 2007, *Science*, 314(5815), pp. 972-976.

Como se ha comentado anteriormente, la principal ventaja de este algoritmo es la no necesidad de especificar de antemano un número de clústeres,  $K$ . Sin embargo, existen otros **parámetros que se deben fijar** y cuyo valor puede condicionar el resultado del agrupamiento.

Por ejemplo, la magnitud del valor de las preferencias es proporcional al número de clústeres que tienden a formarse. La explicación de esta relación radica en el hecho de que los valores altos de preferencia otorgan confianza en sí mismos a los casos en el sentido de que pueden ser los representantes.

Así pues, es difícil que a través de este proceso iterativo un representante se imponga a otro con preferencia inicial alta. Fijar estos valores de preferencia puede requerir un procedimiento (como la validación cruzada) donde se consideren diferentes valores y, de acuerdo con algún criterio de selección, quedarse con el que lleva al mejor agrupamiento resultante.



### Enlace de interés

Documentación del algoritmo de propagación de afinidad implementado en la librería scikit-learn para el lenguaje de programación Python:

<https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AffinityPropagation.html>

## 2.6. Agrupamiento basado en modelos probabilísticos

El uso de **técnicas de agrupamiento basadas en modelos probabilísticos** ha dado históricamente resultados prometedores. En este caso, se asume la existencia de un modelo probabilístico a partir del cual se han generado los datos observados. Un algoritmo de agrupamiento de este tipo intenta encontrar, a partir de un conjunto de datos de entrenamiento (datos observados), el mejor ajuste del modelo generador que se esté asumiendo.



De esta manera, las técnicas de agrupamiento de este tipo se podrían definir como un procedimiento de **estimación de los parámetros del modelo** que se asume. La aproximación más común es asumir que el modelo probabilístico generador de los datos es una mixtura de distribuciones de probabilidad (normalmente, una mixtura gaussiana).

Cada **componente** (distribución) de la mixtura representaría un **clúster** y, de esta manera, se pueden separar los ejemplos del conjunto de entrenamiento en clústeres. Diremos que un ejemplo pertenece al clúster cuya componente le otorga mayor probabilidad.

De hecho, las técnicas probabilísticas de agrupamiento permiten considerar **pertenencias probabilísticas** (no deterministas) de ejemplos a clústeres, es decir, un ejemplo podría pertenecer a la vez a diferentes clústeres con diferente probabilidad.

En esta sección, primero se definirá formalmente el concepto de mixtura de distribuciones para, posteriormente, explicar el algoritmo esperanza-maximización, usado mayoritariamente para aprender los parámetros de este tipo de modelos en presencia de incertidumbre. La incertidumbre, en este caso, se encuentra en el desconocimiento de la asignación de los casos del conjunto de entrenamiento a uno u otro clúster.

### 2.6.1. Modelos de mixturas

En aprendizaje automático no supervisado, los **modelos de mixturas de distribuciones** han sido usados para representar de manera probabilística la asignación de ejemplos a clústeres. Se asume la existencia de un modelo generador, una mixtura de distribuciones, y que cada ejemplo del conjunto de entrenamiento (datos observados) ha sido muestreado de una de las componentes de la mixtura. El objetivo es aprender los parámetros del modelo de mixtura que mejor se ajuste a los datos observados. Cada clúster está representado por una componente y compuesto por todos los ejemplos que fueron probablemente generados por esa distribución de probabilidad.

Formalmente, el conjunto de datos,  $\{x_1, \dots, x_n\}$ , consta de  $n$  muestras de una variable aleatoria v-dimensional. Esta variable aleatoria está distribuida de acuerdo con una mixtura de  $K$  componentes, donde cada componente representa una distribución de probabilidad. La **función de densidad de una mixtura** se puede expresar de la siguiente manera:

$$p(x) = \sum_{k=1}^K \pi_k \cdot p(x; \theta_k)$$

En la fórmula anterior,  $\theta_k$  representa el conjunto de parámetros de la  $k$ -ésima componente,  $p(x; \theta_k)$  es la distribución de probabilidad de la  $k$ -ésima componente parametrizada mediante  $\theta_k$ , y los valores  $\{\pi_1, \dots, \pi_K\}$ , conocidos como coeficientes o pesos de mezcla, denotan la probabilidad de que una muestra sea la realización de cada una de las componentes. Por eso, al ser probabilidades, el conjunto  $\{\pi_1, \dots, \pi_K\}$  debe cumplir las siguientes propiedades:

$$0 \leq \pi_k \leq 1, \forall k \in \{1, \dots, K\} \wedge \sum_{k=1}^K \pi_k = 1$$

Tal y como se ha explicado, se asume que cada ejemplo es una muestra de una de las componentes (distribuciones) de la mixtura. El proceso generador de una muestra puede, así, escalonarse en **dos pasos**:

1. Seleccionar la componente  $k$  desde la que se generará la muestra.
2. Muestrear la distribución de probabilidad seleccionada,  $p(x; \theta_k)$ .

Desde este punto de vista, se puede inferir directamente la existencia de una **variable aleatoria latente** que determina la selección de la componente a muestrear: en concreto, una distribución de probabilidad categórica distribuida de acuerdo con las probabilidades  $\{\pi_1, \dots, \pi_K\}$ ,  $z \sim \text{Cat}(\{\pi_k\}_{k=1}^K)$ . Es decir, la probabilidad de seleccionar la  $k$ -ésima componente es  $p(z = k) = \pi_k$ . En otras palabras, existe una variable aleatoria  $Z$  no observada en el conjunto de datos cuyo valor en cada ejemplo identificaría la distribución de probabilidad (componente) a la que pertenece. En otras palabras, la variable  $Z$  desvelaría la asignación de los ejemplos muestreados a su clúster. Nótese también que la probabilidad  $p(x; \theta_k)$  puede considerarse la probabilidad condicionada del caso  $x$ , dado que se ha seleccionado la componente  $k$ ,  $p(x | z = k) = p(x; \theta_k)$ .

Por conveniencia y de manera totalmente equivalente, se puede considerar que la variable latente sigue una distribución de probabilidad multinomial con un único intento,  $z \sim \text{Mult}(1, \{\pi_k\}_{k=1}^K)$ . Una variable latente es una variable que, a pesar de no ser observada, se asume que existe. En este caso, una muestra  $z$  es un vector  $K$ -dimensional que toma  $K - 1$  valores 0 y un único 1, que indica cuál es la componente seleccionada ( $z_k \in \{0, 1\} \wedge \sum_{k=1}^K z_k = 1$ ). La distribución marginal  $p(z)$  en este caso se expresa de la siguiente manera:

$$p(z) = \prod_{k=1}^K \pi_k^{z_k}$$

La distribución condicional del caso  $x$  dado  $z$  es:

$$p(x | z) = \prod_{k=1}^K p(x; \theta_k)^{z_k}$$

De ambas se puede obtener la distribución marginal de  $x$  integrando la distribución conjunta  $p(x, z)$  sobre todos los posibles valores de  $z$ :

$$p(x) = \sum_z p(x, z) = \sum_z p(x | z)p(z) = \sum_{k=1}^K \pi_k \cdot p(x; \theta_k)$$

Usando la **regla de Bayes** se puede obtener la probabilidad de que un caso provenga de una componente concreta, es decir, la probabilidad condicionada de  $z$  dado  $x$ :

$$p(z_k = 1 | x) = \frac{p(x | z_k = 1)p(z_k = 1)}{\sum_{k'=1}^K p(x | z_{k'} = 1)p(z_{k'} = 1)} = \frac{\pi_k \cdot p(x; \theta_k)}{\sum_{k'=1}^K \pi_{k'} \cdot p(x; \theta_{k'})}$$

En un contexto probabilístico, la **verosimilitud** se define como la plausibilidad de un conjunto de parámetros  $\Theta$  de un modelo dado un conjunto de datos observados. Se calcula como la probabilidad asignada al conjunto de datos observado por el modelo parametrizado mediante  $\Theta$ :

$$\mathcal{L}(\Theta | \{x_1, \dots, x_n\}) = \prod_{i=1}^n p(x_i; \Theta)$$

donde, en el caso de un modelo de mixtura, el conjunto de parámetros es  $\Theta = \{\pi_1, \dots, \pi_K, \theta_1, \dots, \theta_K\}$ .

Hasta ahora se ha asumido que hay un modelo de mixtura parametrizado mediante  $\Theta$ . Sin embargo, en la práctica el **valor de los parámetros del conjunto  $\Theta$**  es desconocido. Los valores de los parámetros deben ser estimados a partir de los datos. En este caso, es necesario computar los parámetros de las distribuciones de probabilidad de las distintas componentes y su combinación mediante los coeficientes  $\{\pi_1, \dots, \pi_K\}$ .

Uno de los métodos tradicionales de estimación de parámetros es conocido como el método de estimación de parámetros máximo verosímiles. Se buscan mediante un proceso de optimización los parámetros  $\Theta$  que maximizan la función  $\mathcal{L}(\Theta | \{x_1, \dots, x_n\})$ . El aprendizaje de los parámetros máximo verosímiles tiene la ventaja de que, en diferentes distribuciones, existe una **fórmula cerrada para su cálculo**.



En concreto, se suele buscar el conjunto de parámetros que **maximiza el logaritmo de la verosimilitud**:

$$\hat{\Theta}_{ML} = \underset{\Theta}{\operatorname{argmax}} \log \mathcal{L}(\Theta | \{x_1, \dots, x_n\}) = \underset{\Theta}{\operatorname{argmax}} \sum_{i=1}^n \log p(x_i; \Theta)$$

De este modo, el cómputo entraña menos problemas —el producto de números menores que 1, que rápidamente tiende a 0, se substituye por una suma de valores de mayor rango,  $(-\infty, 0)$ — y se garantiza que los puntos que maximizan tanto la función  $\mathcal{L}$  como la función  $\log \mathcal{L}$  son los mismos.

De entre todos los modelos de mixturas, el más común es el **modelo de mixtura gaussiana**, donde cada componente sigue una distribución de probabilidad normal o gaussiana multivariada:

$$\mathcal{N}(x | \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^v |\Sigma|}} e^{-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu)}$$

Los parámetros de la distribución son  $\mu$  y  $\Sigma$ .  $\mu$  es la media (el vector  $v$ -dimensional que representa al caso medio o centro),  $\Sigma$  es la  $(v \times v)$ -matriz de covarianzas y  $v$  es el número de variables descriptivas del problema. Así, el conjunto de parámetros de la  $k$ -ésima componente de una mixtura gaussiana es  $\theta_k = \{\mu_k, \Sigma_k\}$  y la probabilidad marginal de un ejemplo  $x$  puede expresarse, en este caso, de la siguiente forma:

$$p(x) = \sum_{k=1}^K \pi_k \cdot p(x; \theta_k) = \sum_{k=1}^K \pi_k \cdot \mathcal{N}(x | \mu_k, \Sigma_k)$$

De esta manera, se estiman a partir del conjunto de datos disponible los parámetros máximo verosímiles  $\hat{\Theta}_{ML} = \{\pi_1, \dots, \pi_K, \mu_1, \dots, \mu_K, \Sigma_1, \dots, \Sigma_K\}$ .

Se puede demostrar que existen **fórmulas cerradas** para el cálculo de los parámetros máximo verosímiles,  $\hat{\Theta}_{ML}$ . Por ejemplo, la media de las diferentes componentes se calcula de la siguiente manera:

$$\hat{\mu}_k = \frac{\sum_{i=1}^n \hat{z}_{ik} x_i}{\sum_{i=1}^n \hat{z}_{ik}}$$

En la fórmula anterior,  $\hat{z}$  es una versión probabilista del vector  $z$ , que asigna cada ejemplo  $x_i$  a cada componente  $k$  con cierta probabilidad  $\hat{z}_{ik}$  ( $0 \leq \hat{z}_{ik} \leq 1, \forall k \in \{1, \dots, K\} \wedge \sum_{k=1}^K \hat{z}_{ik} = 1$ ). La versión probabilista  $\hat{z}$  permite trabajar con la incertidumbre de la asignación. Nótese que, al ser este precisamente el objetivo del agrupamiento (descubrir la asignación de casos a clústeres), los vectores  $z$  originales no están disponibles en el momento del entrenamiento.

Tal y como está definida, la **probabilidad**  $\hat{z}_{ik}$  de que la componente  $k$  sea la distribución de probabilidad generadora del caso  $x_i$  puede estimarse a partir del conjunto de datos como la probabilidad condicionada de  $z_i$  dado  $x_i$ :

$$\hat{z}_{ik} = p(z_{ik} = 1 \mid x_i) = \frac{\pi_k \cdot p(x_i; \theta_k)}{\sum_{k'=1}^K \pi_{k'} \cdot p(x_i; \theta_{k'})} = \frac{\pi_k \cdot N(x_i \mid \mu_k, \Sigma_k)}{\sum_{k'=1}^K \pi_{k'} \cdot N(x_i \mid \mu_{k'}, \Sigma_{k'})}$$

Por su parte, la **matriz de covarianza**  $\Sigma_k$  de cada componente se calcula de la siguiente forma:

$$\hat{\Sigma}_k = \frac{\sum_{i=1}^n \hat{z}_{ik} (x_i - \mu_k)(x_i - \mu_k)^T}{\sum_{i=1}^n \hat{z}_{ik}}$$

Y los **coeficientes de las componentes** se obtienen de esta manera:

$$\pi_k = \frac{\sum_{i=1}^n \hat{z}_{ik}}{n}$$



Se habrá observado la **interdependencia** entre los parámetros y valores necesarios para aprender el modelo en sus respectivas fórmulas. El valor de los vectores  $\hat{z}_i$ , desconocido en el momento del aprendizaje, es usado para los estimadores máximo verosímiles  $\hat{\mu}_k$ ,  $\hat{\Sigma}_k$  y  $\pi_k$ . A su vez, un ajuste del modelo,  $\hat{\Theta}_{ML}$ , es necesario para calcular la probabilidad condicionada que da valor a los vectores  $\hat{z}_i$ . Esta doble dependencia sugiere una solución iterativa que no es otra cosa que un caso particular del popular algoritmo esperanza-maximización.

## 2.6.2. Algoritmo esperanza-maximización

El **algoritmo esperanza-maximización** (expectation-maximization en inglés), en adelante EM, es una estrategia muy popular en entornos de aprendizaje donde existe información no observada. En el caso de una mixtura gaussiana, se asume la existencia de una variable aleatoria latente  $z_i$  que determina la componente generadora del caso  $x_i$ .

El EM es un método iterativo que combina la estimación de los valores no observados y el aprendizaje de los parámetros del modelo. En el caso concreto de una mixtura gaussiana, el algoritmo se podría resumir de la siguiente manera:

En la  $t$ -ésima iteración, usando el ajuste del modelo obtenido en la iteración  $t - 1$ , se obtiene una estimación de los vectores  $\hat{z}_i$  para todos los casos  $x_i$  (con  $i \in \{1, \dots, n\}$ ). Con esta nueva estimación de los vectores  $\hat{z}_i$  se aprende un nuevo ajuste del modelo ( $\hat{\mu}_k$ ,  $\hat{\Sigma}_k$  y  $\pi_k$ , con  $k \in \{1, \dots, K\}$ ).

Está demostrado que la verosimilitud del modelo iterativamente reestimado mediante esta técnica crece monótonamente hasta alcanzar un máximo local. En la Tabla 7 se muestra el pseudocódigo de este sencillo algoritmo:

**Tabla 7**

Pseudocódigo del algoritmo EM para aprender una mixtura gaussiana

---

#### Algoritmo EM para mixtura gaussiana

Recibe: conjunto de entrenamiento,  $\{x_1, \dots, x_n\}$ ; número de componentes  $K$ ; umbral de parada  $\epsilon$ .

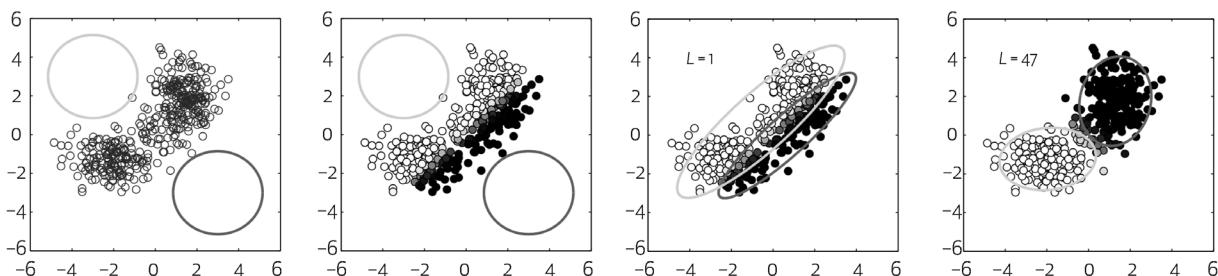
1. Inicializar el modelo con parámetros  $\hat{\mu}_k$ ,  $\hat{\Sigma}_k$  y  $\pi_k$  ( $k \in \{1, \dots, K\}$ ) aleatorios.
2. Repetir:
  - 2.1. Paso E: estimar el valor de los vectores  $\hat{z}_i$  dado el modelo del paso previo,  $\Theta^{t-1}$ .
  - 2.2. Paso M: calcular los estimadores máximo verosímiles  $\hat{\mu}_k$ ,  $\hat{\Sigma}_k$  y  $\pi_k$  dados los nuevos valores de los vectores  $\hat{z}_i$  hasta que la verosimilitud del modelo converja:

$$|\mathcal{L}(\Theta^t | \{x_1, \dots, x_n\}) - \mathcal{L}(\Theta^{t-1} | \{x_1, \dots, x_n\})| < \epsilon$$

---

Devuelve: modelo  $\Theta^t$  y agrupamiento resultante.

---

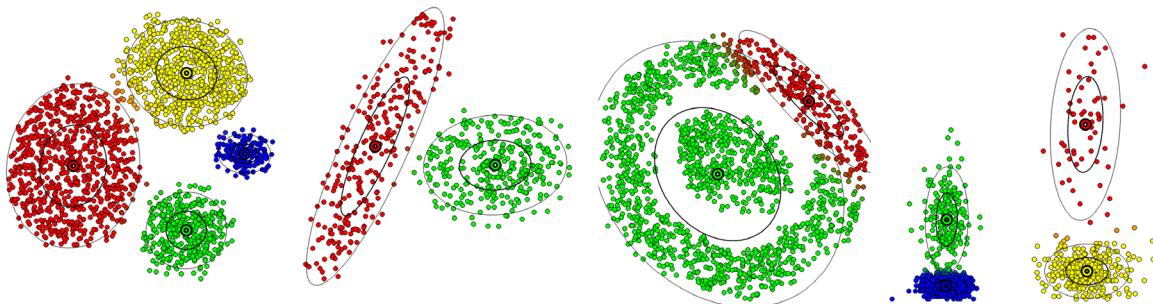


**Figura 23.** Proceso de convergencia del algoritmo EM en un conjunto de datos con dos componentes, desde la inicialización aleatoria (izquierda), al ajuste final tras 47 iteraciones (derecha). Recuperado de *Data Clustering: Algorithms and Applications*, por C. C. Aggarwal y C. K. Reddy (eds.), 2014, Boca Raton: Chapman & Hall/CRC.

Tal y como se muestra en la Figura 23, que presenta un sencillo conjunto de datos de ejemplo bidimensional de dos componentes, el algoritmo se inicializa con dos componentes aleatorias que tienen poca o nula relación con los datos observados. Con la estimación de los valores de los vectores  $\hat{z}_i$ , se obtiene una primera asignación (probabilística) de los ejemplos del conjunto a cada componente (segundo conjunto de datos de la Figura 23). Esto permite al algoritmo llevar a cabo la primera estimación de los parámetros del modelo a partir de los datos observados (tercer conjunto de datos de la Figura 23). Estos dos pasos se iteran hasta converger. El modelo, aunque lejos de ser perfecto, ya se ajusta en cierta medida a los datos.

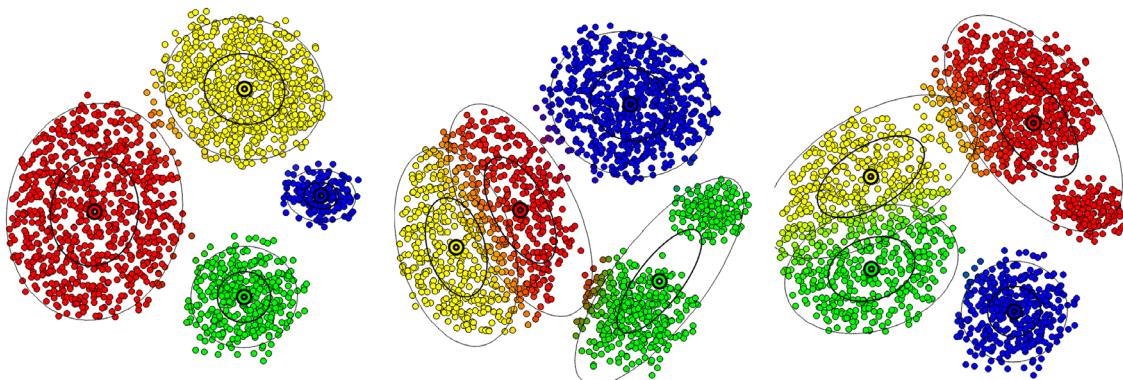
Como se puede ver en la Figura 24, la aplicación de este algoritmo a otros conjuntos de datos de ejemplo usados a lo largo de este documento es satisfactoria en diferentes situaciones. Es incluso **capaz de detectar los clústeres menos densos** del cuarto conjunto de la Figura 24 y, en los otros casos donde los clústeres son cóncavos (cualquier segmento entre dos puntos de un clúster no traspasa los límites del clúster), los detecta.

Solo tiene problemas con el conjunto de datos donde los **clústeres son circunferencias concéntricas**. El algoritmo EM y las mixturas de gaussianas no pueden lidiar con este tipo de conjuntos de datos.



**Figura 24.** Los cuatro conjuntos de datos de la Figura 17 agrupados mediante una mixtura gaussiana.

Como se puede observar, el algoritmo es capaz de detectar clústeres convexos. Los que tienen una forma diferente no se suelen poder modelar mediante una gaussiana.



**Figura 25.** Diferentes agrupamientos sobre el mismo conjunto de datos producto de diferentes inicializaciones del algoritmo EM.

Nótese que el **algoritmo K-means** (véase la sección “2.2.1. K-means”) puede entenderse como un EM que estima una mixtura gaussiana donde la asignación de casos a componentes no es probabilista sino determinista:

$$\hat{z}_{ik} = \begin{cases} 1, & \text{si } \hat{z}_{ik} = \max_{k'} \hat{z}_{ik'} \\ 0, & \text{si } \hat{z}_{ik} \neq \max_{k'} \hat{z}_{ik'} \end{cases}$$

El principal punto a favor del algoritmo EM es su **simplicidad**. Fácil de implementar, tiene la ventaja de que existen garantías formales de que su proceso iterativo converge monótonamente a un máximo de la función de verosimilitud.

Sin embargo, sus **limitaciones** han sido también estudiadas y son bien conocidas. El algoritmo EM depende fuertemente de su **inicialización**, a partir de la cual converge a un máximo local u otro (véase la Figura 25). Así, es fácil obtener agrupamientos que son subóptimos. Para evitar quedar atrapado en máximos locales, existen varias estrategias típicas: ejecutar el algoritmo con diferentes inicializaciones y quedarse con la que maximiza la verosimilitud, o usar un algoritmo clásico de agrupamiento (como K-means) para encontrar un primer agrupamiento con el que inicializar el EM.

Por otro lado, el **número de clústeres** o componentes  $K$  es un parámetro del proceso, es decir, debe ser conocido de antemano. Cuando este no es el caso, lo habitual es probar mediante un proceso de validación cruzada diferentes números de componentes y quedarse con el que produce el mejor modelo de acuerdo con cierto criterio de selección.

Finalmente, otra crítica habitual al algoritmo EM es su **lenta velocidad de convergencia**.

### Enlace de interés

Documentación del algoritmo EM para mixturas de Gaussianas implementado en la librería scikit-learn para el lenguaje de programación Python:

<https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html>

## 2.7. Evaluación de un agrupamiento

Hasta ahora, se han explorado diferentes **técnicas de análisis de agrupamiento**. Estas son las técnicas más usadas en este paradigma del aprendizaje automático no supervisado. Se ha podido observar, por medio de los ejemplos de conjuntos de datos, que las diferentes técnicas tienen distinto rendimiento según las características de los conjuntos de datos de entrenamiento.

En la práctica, suele ser necesario **evaluar el rendimiento** de las técnicas de agrupamiento para poder elegir la mejor parametrización de cada una de las técnicas (como el número de clústeres,  $K$ , en K-means o en la mixtura gaussiana, o los parámetros de densidad para DBSCAN), identificar cuál es la técnica que mejor lidia con los datos de entrenamiento y es capaz de encontrar el mejor agrupamiento, etc.

En unos pocos casos, las **diferencias** de rendimiento entre técnicas y/o parametrizaciones de una técnica son evidentes, por lo que es relativamente fácil identificar cuál es capaz de encontrar el agrupamiento que mejor se ajusta a los datos. Sin embargo, habitualmente las diferencias no son tan evidentes. Más bien al contrario, las diferencias suelen ser más bien sutiles: no existe ninguna estructura natural de los datos o no es observable, o la representación de los ejemplos del problema no es la adecuada, entre otros problemas.

De aquí surge la necesidad de contar con herramientas para la evaluación de las técnicas de agrupamiento y los resultados que devuelven. En esta sección se repasan algunas de las métricas de evaluación más comúnmente usadas en problemas de aprendizaje no supervisado.

Es necesario tener en cuenta que, por definición, en la práctica, la **verdad básica** (*ground truth* en inglés), es decir, la asignación de ejemplos a clústeres, no está disponible ni siquiera en la etapa de entrenamiento, lo que permitiría una evaluación correcta de un agrupamiento. Esta es, de hecho, la principal diferencia paradigmática entre el aprendizaje supervisado y el aprendizaje no supervisado.



### Ejemplo

Se dispone de un conjunto de datos sobre personas, las cuales se describen según características socioeconómicas. Supongamos que se aplica un algoritmo de agrupamiento y se forma una serie de grupos que dividen la población en colectivos según, principalmente, el tipo de trabajo (por ejemplo, empleados públicos, trabajadores en multinacionales, trabajadores en pymes, etc.). Supongamos que el uso de otro algoritmo nos devuelve otra agrupación donde las personas se dividen en colectivos atendiendo a su lugar de residencia (por ejemplo, centro de ciudad, barrio en las afueras de clase trabajadora, urbanización, pueblo, etc.).

¿Cuál es el mejor agrupamiento? ¿Se puede realmente afirmar que uno es mejor que el otro? Siendo un aprendizaje no supervisado, dos algoritmos pueden encontrar diferentes agrupamientos que teóricamente son válidos. El conocimiento experto del usuario debe, en este caso, discriminar y elegir la técnica y el agrupamiento que mejor encajen con el objetivo del análisis.

Ante la necesidad de un procedimiento fiable para valorar el rendimiento de un algoritmo de agrupamiento y dar pie a la comparación de resultados, existe una gran variedad de **medidas de validación** provenientes de diferentes áreas del conocimiento (estadística, teoría de la información, aprendizaje automático, etc.).



A pesar de todo, no hay acuerdo sobre cuál es la mejor y más consistente estrategia de validación. Las diferentes alternativas se pueden agrupar en dos tipos: las técnicas de valuación intrínsecas y extrínsecas.

Por un lado, las **medidas extrínsecas** hacen uso de información privilegiada a la hora de evaluar. Concretamente, se asume que se conoce la verdad básica (*ground truth* en inglés) de los casos de entrenamiento, es decir, cómo se agrupan realmente. Aunque esta información no se usa para aprender el agrupamiento, sí se considera a la hora de evaluar y comparar los resultados de las diferentes técnicas.

Por otro lado, las **medidas intrínsecas** no usan ningún tipo de información privilegiada e intentan dar una estimación de la bondad de la estructura del agrupamiento aprendido. En muchos casos, la información privilegiada requerida por las medidas de evaluación extrínseca no está disponible y, entonces, solo las medidas intrínsecas pueden ser utilizadas.

Sin embargo, si se consigue el *ground truth* del conjunto (o de un subconjunto) de datos de entrenamiento, el abanico de técnicas de validación se amplía y, sobre todo, la toma de decisiones es más informada (por ejemplo, “¿Qué algoritmo, y con qué configuración, produce el resultado que más se parece a la estructura del agrupamiento real?”).



La teoría dice que el objetivo del análisis de agrupamiento consiste en encontrar subgrupos de ejemplos con gran similitud entre los elementos del mismo clúster (intraclúster) y gran diferencia con los de otros clústeres (interclúster). Así, las medidas de evaluación intrínseca suelen basarse en estos dos principios para definir la bondad de un agrupamiento.



### Enlace de interés

Relación de medidas de evaluación de algoritmos de agrupamiento implementadas en la librería scikit-learn para el lenguaje de programación Python:

<https://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation>

## 2.7.1. Medidas de evaluación extrínseca

La **evaluación extrínseca** comparte muchas similitudes con la evaluación en aprendizaje supervisado, ya que en ambos casos se dispone de la verdad básica. Sin embargo, hay una diferencia sutil: mientras que en clasificación supervisada cada ejemplo pertenece a una etiqueta clase concreta, en el análisis de clústeres no hay etiquetas, sino que solo hay grupos de ejemplos. Por eso, en agrupamiento hay que encontrar la mejor correspondencia entre los clústeres reales (según la verdad básica) y los clústeres aprendidos por un algoritmo antes de poder compararlos. Normalmente suele considerarse que un clúster aprendido **representa** un cierto clúster real si el número de ejemplos compartidos por ambos es alto.

A lo largo de esta subsección,  $\{C_k\}_{k=1}^K$  se refiere al agrupamiento conseguido con la aplicación de un algoritmo de agrupamiento automático, y  $\{B_l\}_{l=1}^{K'}$  denota el agrupamiento real (verdad básica) en clústeres del mismo conjunto de datos. De esta manera,  $C_k \cap B_l$  es el subconjunto de ejemplos contenidos tanto en  $C_k$  como en  $B_l$ , y el tamaño de la intersección es  $n_{kl} = |C_k \cap B_l|$ . También se usará  $n_k$  para denotar el tamaño del clúster  $C_k$  y  $n_l$  para el tamaño de  $B_l$ .

Si se considerase que este es un problema de clasificación, se podría calcular el error de la siguiente manera:

$$E = 1 - \frac{1}{n} \max_{\sigma} \sum_{l=1}^{K'} n_{\sigma(l)l}$$

En la fórmula anterior  $\sigma$  es una función determinista que le asigna, a cada clúster original  $B_l$ , un clúster predicho  $C_{\sigma(l)}$ :  $\sigma(l) \in \{1, \dots, K\}$ . Dada la posibilidad de que el número de clústeres  $K$  (predicho) y  $K'$  (real) sea diferente, la función  $\sigma$  puede relacionar dos clústeres reales con el mismo clúster predicho ( $\sigma(l_1) = \sigma(l_2)$  con  $l_1 \neq l_2$ ).

La medida de **precisión** calcula, sobre los elementos de un clúster predicho  $C_k$ , cuántos de ellos pertenecen al mismo clúster original  $B_l$ :

$$P_{kl} = \frac{n_{kl}}{n_k}$$

Por su parte, la **exhaustividad** (*recall* en inglés) mide, sobre el total de elementos de un clúster real  $B_l$ , cuántos de ellos se encuentran también en el clúster predicho  $C_k$ :

$$R_{lk} = \frac{n_{kl}}{n_l}$$

La **pureza**, conocida también como precisión promedio, calcula la precisión media de los clústeres predichos. Sin embargo, como anteriormente se ha comentado, no se conoce *a priori* la correspondencia entre los clústeres predichos y los reales. Por eso se usa la función máximo: se asume que el clúster real que corresponde al clúster predicho  $C_k$  es aquel  $B_l$  que maximiza la precisión  $P_{ki}$ :

$$Pu = \sum_{k=1}^K \frac{n_k}{n} \max_{l \in \{1, \dots, K\}} P_{kl}$$

Nótese que no existe ninguna restricción, de tal manera que dos clústeres predichos diferentes  $C_{k_1}$  y  $C_{k_2}$  podrían corresponderse con el mismo clúster real  $B_l$ .

La **medida F**, por su parte, es el promedio ponderado, sobre todos los clústeres originales  $\{B_l\}_{l=1}^{K'}$ , de la media harmónica de la precisión y la exhaustividad:

$$F1 = \sum_{l=1}^{K'} \frac{n_l}{n} \max_{k \in \{1, \dots, K\}} \left( \frac{2P_{kl}R_{lk}}{P_{kl} + R_{lk}} \right)$$

En este caso, para cada clúster original  $B_l$  se considera que el clúster predicho correspondiente  $C_k$  es aquel que maximiza la media harmónica. Igualmente, se acepta que dos clústeres reales diferentes  $B_{l_1}$  y  $B_{l_2}$  podrían corresponderse con el mismo clúster predicho  $C_k$ .

Las dos últimas medidas que se presentan en este documento provienen del área de la teoría de la información. Por un lado, la **entropía** es una medida del desorden que aumenta según se mezclan los clústeres y disminuye a medida que los clústeres predichos se parecen en mayor medida a los clústeres reales:

$$H = - \sum_{k=1}^K \frac{n_k}{n} \left( \sum_{l=1}^{K'} \frac{n_{kl}}{n_k} \log \frac{n_{kl}}{n_k} \right)$$

Al igual que la pureza, la entropía habla de lo puro o consistente del agrupamiento predicho con respecto al real.

Por otro lado, la **información mutua** mide, como su nombre indica, la cantidad de información contenida en un agrupamiento que explica el otro agrupamiento. Dicho de otra manera, la información mutua mide el grado en que se puede explicar el agrupamiento producido por un algoritmo a medida que vamos conociendo las características del agrupamiento real:

$$I = \sum_{k=1}^K \sum_{l=1}^{K'} \frac{n_{kl}}{n} \log \frac{n \cdot n_{kl}}{n_k \cdot n_l}$$

Estas y otras medidas de evaluación extrínseca dan mucha información (certera) sobre la validez del resultado del agrupamiento. El único punto abierto en este tipo de técnicas es la búsqueda de la mejor correspondencia individual entre clústeres predichos y reales. De resto, aunque cada medida pone su foco en una característica diferente de lo que podría considerarse un buen algoritmo de agrupamiento, todas tienen un criterio fiable para estudiar el rendimiento del algoritmo: la **comparación con la verdad básica**.

Y este es precisamente su punto débil: se basan en la disponibilidad de la verdad básica,  $\{B_l\}_{l=1}^K$ , que en muchas situaciones reales no es posible conseguir. En todas esas situaciones donde la verdad básica no está disponible, solo es posible recurrir a medidas de evaluación intrínseca.

## 2.7.2. Medidas de evaluación intrínseca

A continuación, se presenta una selección de las **medidas de evaluación intrínseca** que calculan y bareman de diferente manera la distancia intraclúster (compactibilidad) y la distancia interclúster (separabilidad) de un agrupamiento.

La raíz del cuadrado de la media de la desviación típica (RMSSTD por sus siglas en inglés) mide la homogeneidad de los clústeres del agrupamiento, sin tener en cuenta la distancia interclúster:

$$\text{RMSSTD} = \sqrt{\frac{\sum_{k=1}^K \sum_{x_i \in C_k} x_i - c_k^2}{v \cdot \sum_{k=1}^K (|C_k| - 1)}}$$

En la fórmula anterior,  $c_k$  es el centro o representante del clúster  $C_k$ , y  $K$  es el número de clústeres resultante.

La medida  **$R^2$**  es el ratio de la distancia intraclúster con respecto a la distancia interclúster de un hipotético grupo que abarca todo el conjunto de datos:

$$R^2 = \frac{\sum_{x_i} x_i - c^2 - \sum_{k=1}^K \sum_{x_i \in C_k} x_i - c_k^2}{\sum_{x_i} x_i - c^2}$$

En la fórmula anterior,  $c$  es el centro o representante de todo el conjunto de entrenamiento. Esta medida se puede entender como un indicador del grado de diferencia entre los distintos clústeres.

Por otra parte, la medida conocida como **ancho de silueta** contempla tanto la distancia interclúster como la intraclúster. Calcula y promedia la medida de silueta para cada punto de todos los clústeres, definida como una diferencia normalizada de la distancia interclúster (mínima distancia a un punto de otro clúster) menos la distancia intraclúster (distancia media al resto de puntos del clúster):

$$S = \frac{1}{K} \sum_{k=1}^K \frac{1}{|C_k|} \sum_{x_i \in C_k} \frac{b_k(x_i) - a_k(x_i)}{\max\{a_k(x_i), b_k(x_i)\}}$$

En la fórmula anterior:

$$a_k(x_i) = \frac{1}{|C_k| - 1} \sum_{x_j \in C_k: x_j \neq x_i} d(x_i, x_j) \quad b_k(x_i) = \min_{h \neq k} \frac{1}{|C_h|} \sum_{x_j \in C_h} d(x_i, x_j)$$

El **índice Calinski-Harabasz** evalúa la bondad de un agrupamiento basado en la suma promedio de distancias interclúster e intraclúster al cuadrado. Concretamente, plantea un cociente entre ambas distancias promedio usando los centros o representantes de los clústeres (y del conjunto de datos) para ello:

$$iCH = \frac{(n - K) \sum_{k=1}^K |C_k| \cdot d(c_k, c)^2}{(K - 1) \sum_{k=1}^K \sum_{x_i \in C_k} d(x_i, c_k)^2}$$

El **índice I**, por su parte, mide la separación interclúster a través de la distancia máxima entre los centros (representantes) de los diferentes clústeres y la compactibilidad intraclúster mediante la distancia entre cada elemento del clúster con su centro o representante.

$$il = \left( \frac{\sum_{x_i} d(x_i, c)}{K \sum_{k=1}^K \sum_{x_i \in C_k} d(x_i, c_k)} \cdot \max_{i,j \in \{1,..,K\}} d(c_i, c_j) \right)^p$$

Finalmente, el **índice de Dunn** es también un cociente donde la distancia interclúster se obtiene como la mínima distancia entre dos elementos de distintos clústeres y la distancia intraclúster como el diámetro máximo entre todos los clústeres. El diámetro de un clúster se define como la mayor distancia entre dos puntos de un mismo clúster:

$$ID = \min_{k \in \{1,..,K\}} \left( \min_{k' \in \{1,..,K\}} \frac{\min_{x_i \in C_k, x_j \in C_{k'}} d(x_i, x_j)}{\max_{k'' \in \{1,..,K\}} \left( \max_{x_i \in C_{k''}, x_j \in C_{k''}} d(x_i, x_j) \right)} \right)$$

Ni la medida RMSSTD ni la R cuadrado son medidas válidas para estimar el mejor valor de  $K$ , ya que siempre muestran la misma respuesta ante el aumento del número de clústeres (RMSSTD se reduce, mientras que R cuadrado se incrementa). A medida que las diferentes fronteras entre clústeres se diluyen y aparecen más puntos en el conjunto de datos que podrían considerarse ruido, todas las métricas tienden a resentirse.

Sin embargo, las medidas de Dunn y de Calinski-Harabasz salen especialmente perjudicadas por su manera de calcular la distancia interclúster y la distancia intraclúster, respectivamente. Cuando el problema se caracteriza por la diferente densidad de los clústeres, el índice I es el que más sufre, debido a la manera de calcular la distancia intraclúster. Si se consideran conjuntos de datos con clústeres de diferente tamaño, la medida de validación que tiende a dar peores resultados es la de Calinski-Harabasz.

El cómputo básico de esta medida, al igual que el algoritmo K-means, tiende a considerar mejores las soluciones que minimizan la distancia media al representante del clúster y, de esta manera, tiende a dividir los clústeres más grandes. De manera similar, todas las métricas que usan un centro o representante del clúster para sus cálculos tienden a tener problemas con los conjuntos de datos cuyos clústeres tienen formas arbitrarias no convexas.



## Capítulo 3

### Análisis de componentes

En el área de análisis de datos, en general, se suele trabajar con un conjunto de ejemplos sobre los que se aplican diferentes técnicas dependiendo del objetivo del estudio. Se suele intentar alterar los datos en la menor medida posible, pero en ciertas ocasiones es conveniente e incluso recomendable trabajar con una transformación de los datos. El **análisis de componentes** transforma los datos a un espacio alternativo con nuevos ejes. La principal hipótesis de este conjunto de técnicas es que los datos, aunque originalmente se hayan descrito a través de  $v$  variables descriptivas, pueden ser expresados por un número menor de variables o componentes.

Probablemente, el principal uso del análisis de componentes es la **reducción de dimensionalidad**, donde el objetivo es expresar los mismos datos, con la menor pérdida de información posible, a través de un menor número de variables. En este caso, la transformación del conjunto de datos y su representación en un espacio alternativo pueden desvelar relaciones ocultas entre las variables.



#### Ejemplo

En un hipotético escenario donde se estudia el rendimiento de la plantilla en su puesto de trabajo, se podría dar la circunstancia de que el número de horas de trabajo real esté directamente relacionado con la antigüedad del material (asiento, monitor, etc.) que usa la plantilla. La relación (a mayor antigüedad, menor comodidad y mayor distracción) puede quedar escondida a simple vista, pero podría ser descubierta con el uso de técnicas de análisis de componentes.

En la práctica, las componentes que explican en menor medida los datos se eliminan para conseguir la reducción de dimensionalidad. Paralelamente, esta práctica hace del análisis de componentes una **técnica efectiva contra el ruido y los valores extraños**. Como se descartan las variables que explican en menor medida los datos, una cantidad considerable de ruido puede ser filtrado por medio de esta práctica. En otras circunstancias, la expresión de los datos en un espacio diferente simplemente permite encontrar representaciones más convenientes de los datos para ciertos tipos de técnicas de análisis.

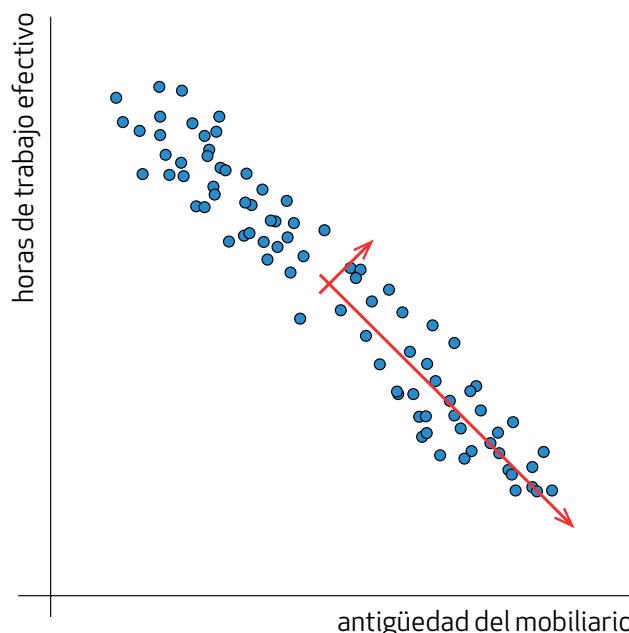
A lo largo de este capítulo, se estudiarán dos técnicas de análisis de componentes: el **análisis de componentes principales** (PCA, por sus siglas en inglés) y el **análisis de componentes independientes** (ICA). Se podría decir que, mientras que PCA permite encontrar representaciones de un conjunto de datos en un espacio reducido, ICA representa los datos a través de subelementos independientes. En otras palabras, mientras que PCA se usa para obtener representaciones comprimidas de los datos, ICA ayuda a separar los datos.

### 3.1. Análisis de componentes principales

Las **componentes principales** son una serie de proyecciones de los datos mutuamente no correlacionadas y ordenadas de acuerdo a la cantidad de varianza de los datos originales que explican. El análisis de componentes principales transforma los datos a un nuevo espacio donde cada componente principal es un eje (variable). Siguiendo el orden establecido, cada eje o componente principal es el que explica mejor la mayor porción de varianza no explicada.

El primer eje es el que explica una mayor cantidad de varianza. El segundo, ortogonal con respecto al primero, es el eje que explica una mayor cantidad de varianza no explicada por el primer eje. El siguiente eje, ortogonal con respecto a los anteriores, es el que explica una mayor cantidad de la varianza no explicada por las componentes anteriores.

En la Figura 26 se muestra una gráfica con los ejes o componentes principales identificados siguiendo el ejemplo hipotético anterior de la relación entre horas trabajadas y antigüedad del mobiliario de oficina.



**Figura 26.** Gráfica de un análisis de componentes principales (ejes en rojo).

Dado un conjunto de datos  $\{x_1, \dots, x_n\}$ , donde cada ejemplo se describe por medio de  $v$  variables descriptivas,  $x_i \in \mathbb{R}^v$  (con  $v = n$ ), el objetivo es obtener otro conjunto de variables ortogonales entre ellas  $q$  (con  $q < v$ ) que permitan explicar los datos originales en un espacio alternativo  $\{z_1, \dots, z_n\}$  con la menor pérdida de información posible.

El primer paso a la hora de analizar las componentes principales es **normalizar** (media y varianza) los datos para que todas las variables originales tengan el mismo rango. Por un lado, las variables se centran para que la media sea en todos los casos 0:

$$x_i \leftarrow x_i - \frac{1}{n} \sum_{i'=1}^n x_{i'}, \quad \forall i \in \{1, \dots, n\}$$

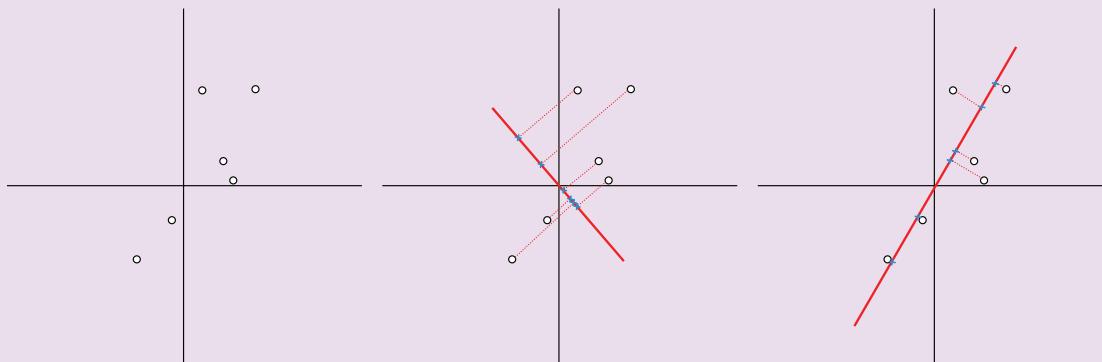
A continuación, se **reescala** cada una de las variables de tal manera que la varianza sea 1:

$$x_{ij} \leftarrow x_{ij} / \sqrt{\frac{1}{n} \sum_{i'=1}^n (x_{i'j})^2}, \quad \forall i \in \{1, \dots, n\} \wedge j \in \{1, \dots, v\}$$

Así se asegura que las variables que originalmente tenían mayor rango no dominen las de menor rango. Sobre los datos normalizados, se computan las componentes ortogonales.



El reto consiste en encontrar la **dirección** en la cual se expresan los datos. Esta tarea se puede expresar de la siguiente manera: encontrar un vector unidad  $u$  (esto es,  $\|u\| = 1$ ) tal que, cuando los datos se proyectan en la dirección que representa el vector  $u$ , la varianza de los datos proyectados sea máxima. Obsérvese, por ejemplo, la Figura 27, donde se usa un pequeño conjunto de datos de 6 puntos a modo de ejemplo:



**Figura 27.** Un conjunto de datos sencillo y su proyección en dos ejes o componentes diferentes.

Si trazáramos la dirección (vector) de la primera proyección (centro), podríamos observar que los puntos proyectados en esa dirección conservan en gran medida la varianza original de los datos. En cambio, la dirección trazada en la segunda proyección (derecha) no consigue conservar la varianza de los datos en la misma medida.

Como las componentes son ortogonales entre sí por definición, la primera y más importante tarea es identificar la dirección de máxima varianza. Dado un vector unidad  $u$  cualquiera y un punto  $x$ , la longitud de la proyección de  $x$  en  $u$  se obtiene  $x^t u$ .

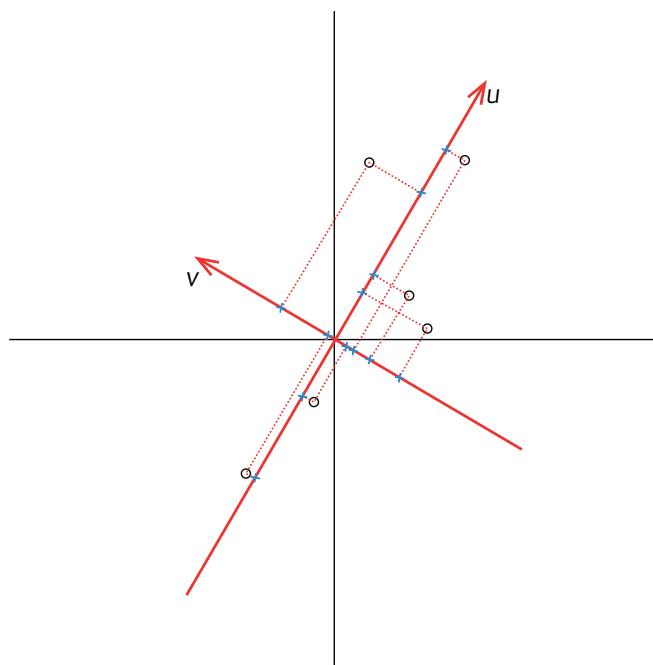
En la Figura 27, el punto  $x$  podría ser cualquiera de los representados mediante un círculo. En la Figura 27 derecha, la proyección  $x^t u$  sería el punto asociado con  $x$  y marcado con una equis azul sobre la línea rojiza, que representa la dirección o vector  $u$ . Dicho esto, el objetivo de máxima varianza implica que se debe buscar el vector  $u$  que maximiza la varianza sobre todo el conjunto de datos,  $\{x_1, \dots, x_n\}$ :

$$\frac{1}{n} \sum_{i=1}^n (x_i^t u)^2 = \frac{1}{n} \sum_{i=1}^n u^t x_i x_i^t u = u^t \left( \frac{1}{n} \sum_{i=1}^n x_i x_i^t \right) u$$

En la fórmula anterior,  $\Sigma = \frac{1}{n} \sum_{i=1}^n x_i x_i^t$  es la matriz de covarianza del conjunto de datos (para que se cumpla esta propiedad, el conjunto de datos debe haber sido normalizado previamente). Dado que  $\|u\| = 1$ , el vector que maximiza el producto anterior es el vector propio principal —es decir, el vector propio asociado con el mayor valor propio— de  $\Sigma$ . De esta manera podríamos obtener una proyección de los datos de entrada en una **única dimensión**, con la consiguiente pérdida de información.

Lo habitual es querer representar los datos en un **nuevo espacio** de  $q < v$  dimensiones. De manera equivalente, los  $q$  vectores ortogonales que maximizan la varianza del conjunto de datos de entrada son los  $q$  vectores propios principales de  $\Sigma$ , es decir, los  $q$  vectores propios con mayor valor propio asociado. Cada vector propio se considera, en este caso, una componente principal.

Las componentes principales están ordenadas por su relevancia (valor propio) decreciente, la cual se mide en términos de la cantidad de variabilidad capturada a través de una componente concreta. Así, el cómputo más importante de esta técnica consiste en la descomposición en vectores propios de la matriz de covarianzas, que es equivalente a la descomposición en valores singulares. La Figura 28 muestra las dos componentes principales del ejemplo anterior.



**Figura 28.** Componentes principales (ortogonales) del conjunto de datos del ejemplo de la Figura 27.

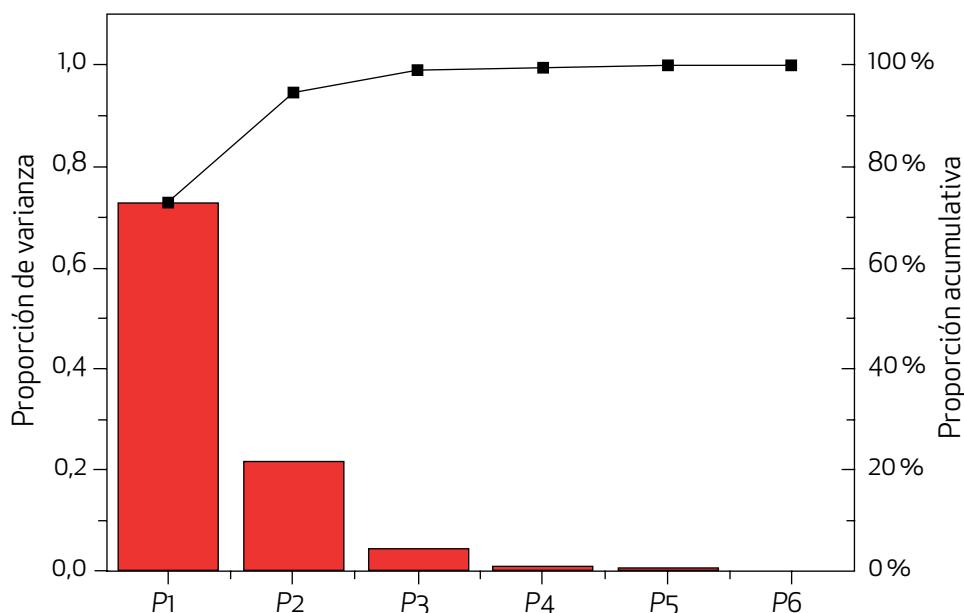
La transformación que entraña cada componente principal sobre los datos de entrada se expresa como una **combinación lineal de las variables originales**. Así, el conjunto de datos en el nuevo espacio se obtiene de la siguiente manera:

$$z_i = \begin{bmatrix} u_1^t x_i \\ u_2^t x_i \\ \vdots \\ u_q^t x_i \end{bmatrix}, \quad \forall i \in \{1, \dots, n\}$$



La tasa de reducción de la dimensionalidad o de compresión obtenida con la aplicación de esta técnica depende del **número de componentes principales  $q$**  seleccionado. Si  $q = v$ , los datos se representan en otro espacio sin ninguna pérdida de información pero sin reducir la dimensionalidad. A medida que se reduce el valor de  $q$ , se consolida la reducción de dimensionalidad pero, a la vez, aumenta la pérdida de información. La redundancia de las variables y las relaciones ocultas en los datos originales determinan el ritmo con el que aumenta la pérdida de información.

En la práctica, la decisión sobre cuántas componentes principales elegir para minimizar la pérdida de información se realiza analizando la **proporción de varianza** que explica cada componente. Esta información se puede mostrar de manera gráfica como se ilustra en la Figura 29.



**Figura 29.** Gráfica de proporciones de varianza por componente principal. Adaptado de "Determinants of Thermostability in Serine Hydroxymethyltransferase Identified by Principal Component Analysis", por F. Leng, L. Y. Wu, C. Lu y X. M. Pan, 2017, *Scientific Reports*, 7, art. núm. 46463.

Puede observarse, además de la proporción de varianza explicada por cada componente, el acumulado de varianza explicada a medida que consideramos un mayor número de componentes.

Así, es habitual fijar un umbral  $s$  en el acumulado (por ejemplo,  $s = 95$ ) y seleccionar las  $q$  componentes (vectores propios de  $\Sigma$ ) principales que expliquen al menos el  $s\%$  de la varianza original de los datos.

El análisis de componentes principales suele usarse como un **paso previo** en problemas de aprendizaje supervisado y también en análisis de agrupamientos. De hecho, técnicas de agrupamiento como K-means pueden aprovechar la proyección de los datos en un nuevo espacio como el que propone el análisis de componentes principales.

En otros casos se usa simplemente como una técnica de **reducción de dimensionalidad**, que, al contrario de lo que devuelven las técnicas de selección de variables (un subconjunto de las variables originales), proyecta los datos en un espacio alternativo donde son necesarias menos variables para explicar los mismos datos.

Cuando se usa para buscar una reducción de dimensionalidad, la idea es obtener un número de componentes  $q$  menor que el número de variables originales,  $q < v$ , que representan los datos originales sin perder (mucha) información.

Ya que las componentes se ordenan según su relevancia (cantidad de varianza explicada), eliminando las últimas componentes principales se consigue reducir el número de variables (componentes) perdiendo la mínima cantidad de información posible.

Entre las principales **ventajas** de esta técnica está el hecho de que no es paramétrica. El único parámetro que debe considerarse es el número de componentes principales  $q$  que se conservan. De hecho, esta elección es posterior a la aplicación del método y queda en manos del usuario y sus intereses. En cualquier caso, existen métodos fiables para su elección. Al no existir parámetros, no es necesario realizar ningún tipo de ajuste.

Otra de las ventajas de este método de aprendizaje no supervisado es el hecho de que en el espacio de optimización no existen máximos locales donde el método pudiese quedar atrapado.

Por el contrario, una de las **críticas** principales es que la representación de los casos en el nuevo espacio puede no ser intuitiva y, en el caso de que las variables originales sean interpretables, esta valiosa característica se pierde con la transformación.

Otra crítica habitual suele ser el hecho de que se limita a momentos muestrales de orden 2 (varianza) y a proyecciones lineales. En este sentido, existen generalizaciones no lineales de este tipo de técnicas, pero quedan fuera del alcance de este documento.



### Enlace de interés

Documentación del algoritmo de análisis de componentes principales implementado en la librería *scikit-learn* para el lenguaje de programación Python:

<https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html>

## 3.2. Análisis de componentes independientes

Al igual que en el caso del análisis de componentes principales, el resultado de la aplicación del **análisis de componentes independientes** es el conjunto de datos transformado y representado en otro espacio. Sin embargo, en este caso el objetivo del método de análisis de componentes independientes no es la reducción de la dimensionalidad, sino la **separación de los datos**.



### Ejemplo

En el problema clásico de la fiesta de cóctel, un grupo de participantes habla simultáneamente. Si pusiésemos varios observadores (por ejemplo, micrófonos) repartidos por la sala donde se celebra la fiesta, tendríamos el problema de que cada observador captaría una mezcla de diferentes subconjuntos de mensajes dependiendo, entre otros, de la distancia entre el observador y el emisor de cada mensaje. La pregunta que se plantea es: ¿se pueden separar las diferentes voces para captar los distintos mensajes?

De manera formal, existe un grupo de  $u$  fuentes independientes que generan un punto,  $s \in \mathbb{R}^u$ . Sin embargo, el punto observado es de alguna manera diferente,  $x$ . Se asume que existe una matriz de mezcla  $A$  tal que:

$$x = As$$

Así, dado un conjunto de muestras observadas  $\{x_1, \dots, x_n\}$ , el objetivo del problema es **recuperar los puntos originales** correspondientes,  $\{s_1, \dots, s_n\}$ . Formulado como un problema de optimización, el objetivo es encontrar la matriz de separación  $W = A^{-1}$  tal que  $s = Wx$ . Específicamente, se busca el vector  $w_j$  que permite recomponer el valor de la  $j$ -ésima fuente original:  $s_j = w_j^t x$  (para todo  $j \in \{1, \dots, u\}$ ). Es decir, dado un instante de la mezcla de conversaciones en la sala del cóctel, si se conoce la matriz  $W$ , es posible identificar (separar) qué decía cada uno de los  $u$  oradores.

Al asumir que las fuentes son independientes, la **distribución de probabilidad conjunta** es igual al producto de las distribuciones marginales de cada fuente:

$$p(s) = \prod_{j=1}^u p_s(s_j)$$

Nótese que la **distribución marginal de las fuentes** está relacionada con la distribución de probabilidad de lo observado dada la matriz  $W$ :

$$p_x(x) = \prod_{j=1}^u p_s(w_j^t x) \cdot |W|$$

En la fórmula anterior, la inclusión del producto por el determinante de la matriz  $W$  es necesaria para que la distribución de probabilidad integre a 1.

En este punto, es necesario elegir una **distribución de probabilidad  $p_s$**  para cada componente original. En vez de elegir una función de densidad directamente, es posible elegir la distribución cumulativa (la densidad es simplemente la derivada de la distribución cumulativa en un punto). La elección más habitual es usar la función sigmoide,  $g(s_j) = 1/(1 + e^{-s_j})$ , como función cumulativa y, en consecuencia, su derivada como función de densidad:  $p_s(s_j) = g'(s_j)$ . Esta función cumple los requisitos de una función de distribución cumulativa: una función monótona que crece de manera suave de 0 a 1.

A estas alturas, es posible calcular el estimador máximo verosímil de la matriz  $W$ . La **verosimilitud** se expresa de la siguiente manera:

$$\mathcal{L}(W; \{x_1, \dots, x_n\}) = \prod_{i=1}^n \left( \prod_{j=1}^u g'(w_j^t x_i) \cdot |W| \right)$$

Y, de manera similar, el **logaritmo de la verosimilitud** es:

$$\log \mathcal{L}(W; \{x_1, \dots, x_n\}) = \sum_i^n \left( \sum_j^u \log g'(w_j^t x_i) + \log |W| \right)$$

Se puede demostrar que la matriz  $W$  que maximiza la verosimilitud se puede obtener mediante un **algoritmo de ascenso de gradiente estocástico**. En este tipo de aproximaciones, dada una matriz  $W$  inicial, se toma un paso en la dirección de máximo ascenso (dada por el gradiente, que es la generalización de la derivada a múltiples dimensiones) usando iterativamente cada uno de los casos de aprendizaje,  $\{x_1, \dots, x_n\}$ . En este caso, la **regla de actualización**, dado un ejemplo observado  $x_i$ , es:

$$W \leftarrow W + \alpha \begin{pmatrix} 1 - 2g(w_1^t x_i) \\ 1 - 2g(w_2^t x_i) \\ \vdots \\ 1 - 2g(w_u^t x_i) \end{pmatrix} x_i^t + (W^t)^{-1}$$

En la regla de actualización,  $u$  es el parámetro de ratio de aprendizaje, que determina el paso al que se actualiza la matriz  $W$  en cada iteración del método. El algoritmo se detiene cuando converge (la matriz  $W$  no cambia sustancialmente entre consecutivas iteraciones) a un máximo local.

Una vez obtenida la estimación máxima verosímil de la matriz  $W$ , las **fuentes originales** pueden ser descubiertas simplemente aplicando la siguiente fórmula sobre cada caso observado  $x_i$ :

$$s_i = W x_i$$

En el cálculo de la verosimilitud se asume que los casos son independientes entre sí (producto de probabilidades no condicionadas). Esta suposición puede no ser muy acertada en muchas aplicaciones reales. En el ejemplo anterior, es obvio que el discurso de cualquier persona (fuente) en un momento concreto  $i$  está relacionado con lo dicho anteriormente (momentos  $i-1, i-2, \dots$ ). Sin embargo, aunque pueda parecer una contradicción, siempre que exista un conjunto de datos suficientemente grande, el aprendizaje está garantizado y el rendimiento del algoritmo no se ve comprometido. En estos casos de evidente correlación entre muestras, es habitual realizar un recorrido aleatorio por los casos del conjunto de datos al implementar el algoritmo de ascenso del gradiente estocástico para acelerar la convergencia.

Es también evidente que el algoritmo **converge a un óptimo local** o a un **punto de silla**. Sin embargo, el comportamiento característico de esta técnica evaluando cada caso de manera individual para producir un paso permite al algoritmo escapar de los óptimos locales con cierta facilidad (dependiendo del ratio de aprendizaje,  $\alpha$ ). Hacer una exploración aleatoria del conjunto de datos en cada iteración ayuda a prevenir este problema.



### Enlace de interés

Documentación del algoritmo de análisis de componentes independientes implementado en la librería scikit-learn para el lenguaje de programación Python:

<https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.FastICA.html>



## Capítulo 4

### Análisis de coagrupamientos

En la sección de análisis de agrupamiento se han presentado múltiples estrategias y técnicas concretas que, dado un conjunto de ejemplos de entrenamiento, buscan **agrupar los casos en subconjuntos** de datos homogéneos de acuerdo con los valores que toman en las diferentes variables descriptivas del problema. Es decir, se busca una división del conjunto inicial por filas (casos o ejemplos).

Sin embargo, en diferentes problemas reales, la tarea de agrupamiento tiene una definición más específica: las variables se tratan de manera similar a los ejemplos, de tal manera que un clúster se define por un subconjunto de ejemplos que comparten un patrón en un subconjunto de variables. Es decir, un coagrupamiento (*biclustering* en inglés) es una **submatriz de la matriz de datos original** restringida a una serie de filas (ejemplos) y columnas (variables).

En la Figura 30, se ilustran las **diferencias entre los dos tipos de agrupamiento** sobre una matriz de entrenamiento de  $n$  ejemplos y  $v$  variables o atributos. El objetivo es descubrir patrones locales que pasan desapercibidos para las técnicas tradicionales de agrupamiento.

Por claridad, los clústeres se han formado con ejemplos (y variables, en el caso del coagrupamiento) consecutivos, aunque esto no es un requisito.

La necesidad de **agrupar tanto los casos como las variables** (tanto filas como columnas de la matriz de datos) se ha descrito en diferentes áreas del conocimiento y ya existen aplicaciones en sistemas de recomendación o minería de textos. Sin embargo, la principal aplicación de estos métodos (y, de hecho, la principal motivación para su desarrollo) es la biomedicina, con usos del coagrupamiento descritos para identificar la interacción entre proteínas o datos de expresión de genes (*microarrays* en inglés).

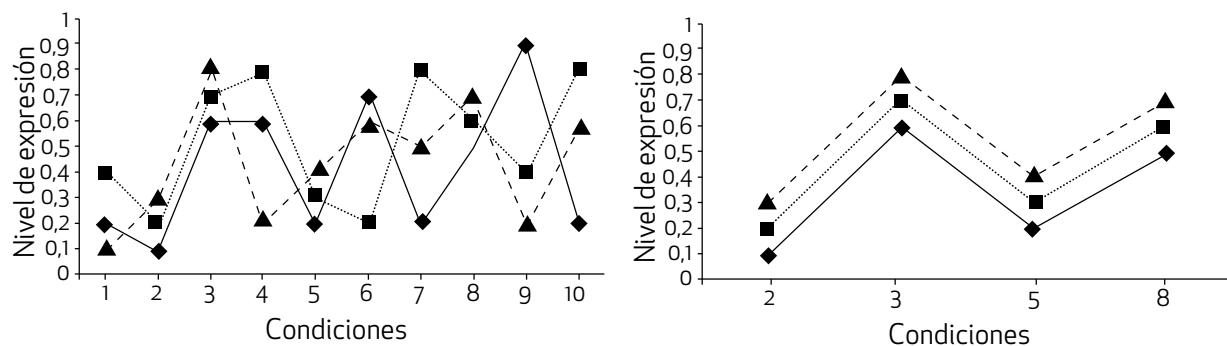
$X_{11}$	$X_{12}$	$X_{13}$	$X_{14}$	$X_{15}$	$X_{16}$	...	$X_{1v}$
$X_{21}$	$X_{22}$	$X_{23}$	$X_{24}$	$X_{25}$	$X_{26}$	...	$X_{2v}$
$X_{31}$	$X_{32}$	$X_{33}$	$X_{34}$	$X_{35}$	$X_{36}$	...	$X_{3v}$
$X_{41}$	$X_{42}$	$X_{43}$	$X_{44}$	$X_{45}$	$X_{46}$	...	$X_{4v}$
$X_{51}$	$X_{52}$	$X_{53}$	$X_{54}$	$X_{55}$	$X_{56}$	...	$X_{5v}$
$X_{61}$	$X_{62}$	$X_{63}$	$X_{64}$	$X_{65}$	$X_{66}$	...	$X_{6v}$
$X_{71}$	$X_{72}$	$X_{73}$	$X_{74}$	$X_{75}$	$X_{76}$	...	$X_{7v}$
$X_{81}$	$X_{82}$	$X_{83}$	$X_{84}$	$X_{85}$	$X_{86}$	...	$X_{8v}$
$X_{91}$	$X_{92}$	$X_{93}$	$X_{94}$	$X_{95}$	$X_{96}$	...	$X_{9v}$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
$X_{n1}$	$X_{n2}$	$X_{n3}$	$X_{n4}$	$X_{n5}$	$X_{n6}$	...	$X_{nv}$

$X_{11}$	$X_{12}$	$X_{13}$	$X_{14}$	$X_{15}$	$X_{16}$	...	$X_{1v}$
$X_{21}$	$X_{22}$	$X_{23}$	$X_{24}$	$X_{25}$	$X_{26}$	...	$X_{2v}$
$X_{31}$	$X_{32}$	$X_{33}$	$X_{34}$	$X_{35}$	$X_{36}$	...	$X_{3v}$
$X_{41}$	$X_{42}$	$X_{43}$	$X_{44}$	$X_{45}$	$X_{46}$	...	$X_{4v}$
$X_{51}$	$X_{52}$	$X_{53}$	$X_{54}$	$X_{55}$	$X_{56}$	...	$X_{5v}$
$X_{61}$	$X_{62}$	$X_{63}$	$X_{64}$	$X_{65}$	$X_{66}$	...	$X_{6v}$
$X_{71}$	$X_{72}$	$X_{73}$	$X_{74}$	$X_{75}$	$X_{76}$	...	$X_{7v}$
$X_{81}$	$X_{82}$	$X_{83}$	$X_{84}$	$X_{85}$	$X_{86}$	...	$X_{8v}$
$X_{91}$	$X_{92}$	$X_{93}$	$X_{94}$	$X_{95}$	$X_{96}$	...	$X_{9v}$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
$X_{n1}$	$X_{n2}$	$X_{n3}$	$X_{n4}$	$X_{n5}$	$X_{n6}$	...	$X_{nv}$

**Figura 30.** Diferencias entre un algoritmo de agrupamiento y otro de coagrupamiento.

Siguiendo el ejemplo de la expresión de una serie de genes ante diferentes condiciones (datos de microarray), en la Figura 31 se ilustra la motivación de las técnicas de coagrupamiento:



**Figura 31.** Comportamiento que motiva la necesidad de algoritmos de coagrupamiento. Adaptado de *Data Clustering: Algorithms and Applications*, por C. C. Aggarwal y C. K. Reddy (eds.), 2014, Boca Raton: Chapman & Hall/CRC.

En la subfigura izquierda, se muestran los niveles de expresión de tres genes en 10 condiciones diferentes. Considerando todas las condiciones, no existe correlación entre los genes. Sin embargo, si se considera solo un subconjunto de condiciones concreto, se observa que los genes están fuertemente correlacionados (subfigura derecha).

Es interesante profundizar un poco más en la **aplicación del estudio de expresión de genes que motivó el desarrollo del coagrupamiento**. Sin entrar en mucho detalle, un gen es la unidad básica de rasgos hereditarios de cualquier organismo vivo y se agrupa con otros genes en las cadenas de ADN. El genotipo es la composición genética de un organismo y el fenotipo son las características observables del organismo determinadas por el genotipo. La expresión de los genes es el nivel más básico en el que el genotipo determina el fenotipo.

Hoy en día, la ingeniería biomédica permite medir el **nivel de expresión de diferentes genes** (unos pocos, miles o incluso todos) en diferentes **condiciones experimentales** (por ejemplo, en diferentes momentos temporales o en diferentes partes del organismo, etc.). Los datos de un *microarray* (o chips de ADN) son una matriz donde cada fila corresponde a un gen y cada columna una condición experimental diferente. Cada celda guarda el nivel de expresión de un gen en una condición experimental específica. De hecho, según el caso de análisis de agrupamiento, se podrían intercambiar las filas y las columnas (transponer la matriz) atendiendo al objetivo de estudio. En algunos casos, es de interés considerar los genes como casos y las condiciones como variables, mientras que en otros estudios es más oportuno considerar los genes como variables y las condiciones como ejemplos.

Así pues, al usar el coagrupamiento, el objetivo no es detectar genes que se expresen igual siempre (en todas las condiciones), ni detectar condiciones en las que los genes se expresen de manera similar, sino identificar genes que en ciertas condiciones muestran unos niveles de expresión similares.



Una técnica de coagrupamiento trata de identificar **clústeres** que tienen las siguientes características:

- Un clúster de ejemplos se define por medio de un solo subconjunto de variables.
- Un clúster de variables se define por medio de un solo subconjunto de ejemplos.
- Los clústeres no son exclusivos ni exhaustivos respecto a los ejemplos y las variables. Un ejemplo puede pertenecer a uno, varios o ningún clúster. Igualmente, una variable puede relacionarse con uno, varios o ningún clúster.

Estas características se ilustran en el agrupamiento de la derecha de la Figura 30. Por un lado, cada uno de los tres clústeres representados se relaciona con un subconjunto diferente de ejemplos y variables. Sin embargo, la cuarta y la quinta variables se relacionan con dos clústeres cada una, al igual que el sexto ejemplo. Por último, la primera variable o el cuarto ejemplo no se contemplan en ningún clúster.

En el caso de datos de expresión de genes, los clústeres están compuestos normalmente por unos pocos genes (ejemplos) y se relacionan con un pequeño subconjunto de condiciones experimentales (variables) en los que muestran patrones de expresión similares. Un gen puede incluirse en más de un clúster o no ser incluido en ninguno. De igual manera, una condición experimental puede relacionarse con uno o varios clústeres o con ninguno.

Formalmente, existe un **conjunto de datos de entrenamiento** (matriz)  $X$  con  $n$  ejemplos y  $v$  variables. Definamos  $I_n$  como el vector de índices de longitud  $n$ ,  $I_n = \{1, \dots, n\}$  y, de igual manera,  $I_v$  como el vector de índices de longitud  $v$ ,  $I_v = \{1, \dots, v\}$ . Así, se puede definir una submatriz  $X'$ , dados sendos subconjuntos de índices  $I'_n \subseteq I_n$  e  $I'_v \subseteq I_v$ , tal que toma por valor  $X'_{ij} = X_{I'_n(i)I'_v(j)}$ , donde  $I'_n(i)$  es la  $i$ -ésima entrada del conjunto  $I'_n$  e  $I'_v(j)$  es la  $j$ -ésima entrada del conjunto  $I'_v$ . Es decir, el valor que toma la primera variable en el primer ejemplo de la submatriz  $X'$  es el valor que toma la matriz original  $X$  en la  $I'_v(1)$ -ésima (primer índice en  $I'_v$ ) variable para el  $I'_n(1)$ -ésimo (primer índice en  $I'_n$ ) ejemplo.

Una submatriz es un clúster si los valores que agrupa siguen un cierto patrón. A lo largo de los años se han considerado y estudiado diferentes patrones:

- Bioclústeres constantes: todos los valores de la submatriz son iguales,  $\forall i \in \{1, \dots, |I'_n|\} \wedge j \in \{1, \dots, |I'_v|\}, x'_{ij} = c$  (con  $c \in \mathbb{R}$ ).
- Bioclústeres con filas constantes: todos los valores de una fila son iguales, pero los valores de diferentes filas son diferentes,  $\forall i \in \{1, \dots, |I'_n|\}, \exists \alpha_i \in \mathbb{R}: \forall j \in \{1, \dots, |I'_v|\}, x'_{ij} = c + \alpha_i$  (con  $c \in \mathbb{R}$ ).
- Bioclústeres con columnas constantes: todos los valores de una columna son iguales, pero los valores de diferentes columnas son diferentes,  $\forall j \in \{1, \dots, |I'_v|\}, \exists \beta_j \in \mathbb{R}: \forall i \in \{1, \dots, |I'_n|\}, x'_{ij} = c + \beta_j$  (con  $c \in \mathbb{R}$ ).
- Bioclústeres basados en patrones (aditivos o multiplicativos): las diferencias (valores aditivos,  $x'_{ij} = c + \alpha_i + \beta_j$ ) o el ratio (valores multiplicativos,  $x'_{ij} = c \cdot \alpha_i \cdot \beta_j$ ) entre los elementos de dos filas o columnas cualesquiera son constantes,  $\forall i, i^* \in I'_n \wedge j, j^* \in I'_v, x'_{ij} - x'_{i^*j^*} = x'_{ij^*} - x'_{i^*j^*}$  (o bien  $x'_{ij}/x'_{ij^*} = x'_{i^*j}/x'_{i^*j^*}$ ).
- Bioclústeres con evoluciones coherentes: más allá del valor exacto, busca bioclústeres con comportamientos coherentes (aumentan o disminuyen a la vez) por filas, columnas o ambas.

En la Figura 32 se ilustra con ejemplos sintéticos sencillos el aspecto que tendría la submatriz  $X'$  de bioclústeres con diferentes patrones:

1	1	1	5	7	9	2	2	2	2	5	4	2	6	4	2.5	7	5
1	1	1	5	7	9	4	4	4	5	8	7	4	12	8	4	10	8.5
1	1	1	5	7	9	6	6	6	3	6	5	5	15	10	5	13	9
1	1	1	5	7	9	8	8	8	6	9	8	8	24	16	8	19	13
1	1	1	5	7	9	10	10	10	4	7	6	3	9	6	3	8	6.5
1	1	1	5	7	9	12	12	12	7	10	9	6	18	12	5.5	14	9

**Figura 32.** Aspecto que podrían tener bioclústeres de diferentes patrones.

De izquierda a derecha, se muestra un ejemplo de bioclúster de patrón constante, de columna constante, de fila constante, de patrón constante aditivo, de patrón constante multiplicativo y de patrón evolutivo coherente.

Desde el pionero trabajo de Cheng y Church (2000), se ha desarrollado una gran variedad de algoritmos de coagrupamiento.

A continuación, se presentarán los más representativos.

## 4.1. Algoritmo de Cheng y Church

Cheng y Church (2000) plantearon el problema del coagrupamiento como un problema de **optimización**. Cada posible biclúster recibe un valor que representa la credibilidad de que esa submatriz sea realmente un biclúster. Se buscan submatrices grandes y uniformes a través de un algoritmo de optimización voraz. Implícitamente, se asume que los niveles de expresión en un clúster son constantes, con la posibilidad de que exista cierto comportamiento aditivo por filas y/o columnas.

Para entender cómo funciona, es necesario empezar por definir el valor medio sobre una **fila** de una submatriz candidata a biclúster como:

$$\bar{x}_{i'v} = \frac{1}{|I'_v|} \sum_{j \in I'_v} x_{ij}$$

Y, de manera similar, el valor medio de una **columna** como:

$$\bar{x}_{l'n} = \frac{1}{|I'_n|} \sum_{i \in I'_n} x_{ij}$$

Finalmente, el valor medio de la **submatriz** se puede calcular de la siguiente manera:

$$\bar{x}_{I'_n I'_v} = \frac{1}{|I'_n| \cdot |I'_v|} \sum_{i \in I'_n} \sum_{j \in I'_v} x_{ij}$$

Dada la suposición anterior, si se substrae el valor medio del biclúster, de la fila y de la columna, el valor residual restante tendería a 0.

El **valor residual** de un punto  $(i,j) \in (I'_n, I'_v)$  es:

$$R_{I'_n I'_v}(i,j) = x_{ij} - \bar{x}_{I'_n I'_v} - \bar{x}_{i'v} - \bar{x}_{l'n}$$

Y el valor residual cuadrático medio de una submatriz:

$$\bar{R}_{I'_n I'_v} = \frac{1}{|I'_n| \cdot |I'_v|} \sum_{(i,j) \in (I'_n, I'_v)} R_{I'_n I'_v}(i,j)^2$$

El problema se plantea como la búsqueda de submatrices  $X'$  cuyo valor residual medio no supere cierto umbral  $\delta$  y que sean máximas (es decir, que no exista otra submatriz mayor con un valor residual medio menor que  $\delta$ ,  $\exists Y' \subset X : \bar{R}_{Y'} < \delta \wedge X' \subset Y'$ ). Este problema es NP-complejo (*NP-hard* en inglés), es decir, irresoluble en un tiempo razonable, si se consideran todas las posibles submatrices.

Así, la solución de Cheng y Church (2000) es un algoritmo voraz que converge a una **matriz localmente máxima con un valor residual medio inferior al umbral**. El algoritmo parte de la matriz completa original ( $I'_n = \{1, \dots, n\}$  y  $I'_v = \{1, \dots, v\}$ ) y consta de dos fases.

En primer lugar, se elimina iterativamente una fila o una columna hasta que  $\bar{R}_{I'_n I'_v} < \delta$ . En cada paso, se elimina la fila o columna que reduzca el valor residual medio en mayor medida:

$$(i^* = \operatorname{argmax}_{i \in I'_n} \frac{1}{|I'_v|} \sum_{j \in I'_v} R_{I'_n I'_v}(i,j) \text{ ó } j^* = \operatorname{argmax}_{j \in I'_v} \frac{1}{|I'_n|} \sum_{i \in I'_n} R_{I'_n I'_v}(i,j)).$$

En la segunda fase, se vuelve a considerar la inclusión de filas y/o columnas eliminadas en el paso anterior. Iterativamente, se selecciona la fila o columna con menor valor ( $i^* = \operatorname{argmin}_{i \in I'_n} \frac{1}{|I'_v|} \sum_{j \in I'_v} R_{I'_n I'_v}(i, j)$ ) o  $j^* = \operatorname{argmin}_{j \in I'_v} \frac{1}{|I'_n|} \sum_{i \in I'_n} R_{I'_n I'_v}(i, j)$ ) y se vuelve a incluir en la submatriz candidata. Este proceso se repite mientras el valor residual medio no supere el umbral  $\delta$ .



Nótese que mediante este procedimiento solo se encuentra **un biiclúster**. Para encontrar más de un biiclúster, los autores recomiendan sustituir en la matriz original  $X$  los valores del biiclúster  $X'$  recientemente descubierto por valores aleatorios. De esta manera, no se encontraría entre ellos ninguna relación que pudiese llevar a la identificación del mismo biiclúster y el algoritmo se derivaría a la búsqueda de otros nuevos.

## 4.2. Algoritmo de firma iterativa

El valor medio de una fila (o de una columna) de un biiclúster debería ser inusualmente alto o bajo en comparación con el valor medio del resto de la fila (o de la columna). De esta intuición nace el **algoritmo de firma iterativa**, que trabaja con  $X^F$  y  $X^C$ , dos versiones de la matriz original  $X$  donde las filas, en el primer caso, y las columnas, en el segundo, se estandarizan (media = 0, varianza = 1).

El algoritmo empieza eligiendo de cierta manera (aleatoriamente o mediante conocimiento experto) un conjunto de filas,  $I'^{(0)}$ . Iterativamente, se eligen una serie de columnas,  $I'^{(t)}_v$ , con respecto a  $I'^{(t-1)}_n$  y, nuevamente, una nueva selección de filas  $I'^{(t)}_n$  a partir de  $I'^{(t)}_v$  (para  $t = \{1, \dots, T\}$ ). El algoritmo para en la iteración  $T$  que satisfaga la siguiente **condición**:

$$\frac{|I'^{(T)}_n \setminus I'^{(T-1)}_n|}{|I'^{(T)}_n \cup I'^{(T-1)}_n|} < \epsilon$$

Es decir, el número de filas presentes en el conjunto actual pero ausentes en el previo (relativo al número de filas en la unión de ambos conjuntos) es inferior a un umbral,  $\epsilon$ .

El paso clave del algoritmo es la **actualización de los conjuntos**  $I'^{(t)}_n$  e  $I'^{(t)}_v$ . En primer lugar, el **subconjunto de columnas** se obtiene de la siguiente manera:

$$I'^{(t)}_v = \left\{ j \in I_v : \left| \bar{x}_{I'^{(t-1)}_n j}^F \right| > t_c \cdot \sigma_c \right\}$$

En palabras, esta expresión quiere decir que se escogen las columnas cuyo valor absoluto medio sobre todas las filas de  $I'^{(t)}_v$  es mayor que un umbral  $t_c$  por la desviación estándar  $\sigma_c$  de los valores medios de todas las columnas de la matriz original ( $\bar{x}_{I'^{(t-1)}_n j}^F, \forall j \in I_v$ ).

De manera similar, el nuevo **subconjunto de filas** se obtiene de la siguiente manera:

$$I'^{(t)}_n = \left\{ i \in I_n : \left| \bar{x}_{i I'^{(t)}_v}^F \right| > t_F \cdot \sigma_F \right\}$$

Es decir, se escogen las filas cuyo valor absoluto medio sobre todas las columnas de  $I'_V^{(t)}$  es mayor que un umbral  $t_F$  multiplicado por la desviación estándar  $\sigma_F$  de los valores medios de todas las columnas de la matriz original ( $\bar{x}_{iI'_V^{(t)}}^F, \forall i \in I_n$ ). Este algoritmo, de manera iterativa, busca subconjuntos de filas (o columnas) que tengan valores medios que destaque con respecto al conjunto total considerando solo un subconjunto de columnas (o filas).

Al igual que el algoritmo anterior, este también devuelve un solo biclúster definido por los conjuntos finales  $I'_n^{(T)}$  y  $I'_V^{(T)}$ . Es bastante fácil inferir que este algoritmo depende fuertemente del conjunto inicial de filas. Si se quieren obtener diferentes biclústeres, se podría ejecutar el algoritmo con diferentes subconjuntos iniciales de filas. De igual manera, el uso de diferentes umbrales ( $\epsilon$ ,  $\sigma_F$  y  $\sigma_C$ ) podría llevar a diferentes resultados.

### 4.3. Algoritmo de coagrupamiento espectral

Kluger, Barsi, Cheng y Gerstein (2003) propusieron el primer desarrollo de un **algoritmo espectral para coagrupamiento**, el cual usa técnicas de álgebra lineal para encontrar biclústeres. Se asume que la matriz  $X$  tiene una estructura de bloques (véase la Figura 33) y biclústeres de patrón multiplicativo por fila, que ciertamente aparecen con regularidad en conjuntos de datos biomédicos. El algoritmo propuesto es capaz de detectar este tipo de estructuras aunque las filas y columnas se presenten desordenadas.

1	1	11	11	3	3
1	1	11	11	3	3
4	4	7	7	12	12
4	4	7	7	12	12

**Figura 33.** Un biclúster de estructura de bloques.



Para entender el funcionamiento del algoritmo, es necesario tener en cuenta que los valores propios de  $XX^T$  y de  $X^TX$  son los mismos. De hecho, dado un vector propio  $e$  de  $X^TX$  con valor propio  $\lambda$ , el vector  $f$  obtenido de multiplicar  $X$  por  $e$  ( $f = Xe$ ) es un vector propio de la matriz  $XX^T$  con el mismo valor propio,  $\lambda$ . Los vectores propios de estas matrices (nótese la estructura de bloques de  $X$ ) siguen también una estructura de bloques (en la Figura 33, tendrían una estructura  $e = (p, p, q, q, o, o)^T$  y  $f = (r, r, s, s)^T$ , donde  $p, q, o, r, s \in \mathbb{R}$ ). Es decir, la estructura de bloques de  $X$  se refleja en los vectores propios de  $XX^T$  y  $X^TX$ .

Para calcular los valores propios, se usa la descomposición en valores singulares,  $X = A\Sigma B^T$ , de tal manera que las columnas de las matriz  $A$  y de la  $B$  son los vectores propios de  $XX^T$  y  $X^TX$  respectivamente. La estructura en bloques de  $X$  se podría confirmar si los vectores propios de  $XX^T$  y  $X^TX$  con el mismo valor propio muestran también la estructura de bloques.

En la práctica no se usa directamente  $X$ . En su lugar, se usa una **versión normalizada** de esta:

$$\check{X} = F^{-\frac{1}{2}} X C^{-\frac{1}{2}}$$

En la fórmula anterior  $F$  es una matriz diagonal de tamaño  $n \times n$  con la suma total por fila ( $F_{ii} = \sum_{j \in \{1, \dots, v\}} x_{ij}$ ) y  $C$  es una matriz diagonal de tamaño  $v \times v$  con la suma total por columna ( $C_{jj} = \sum_{i \in \{1, \dots, n\}} x_{ij}$ ). Sobre esta nueva matriz, se obtiene la descomposición en valores singulares:  $\check{X} = A \Sigma B^T$ . Dado que es una matriz normalizada, el primer par de vectores propios se descarta y se seleccionan los  $p$  mejores vectores propios (columnas) de  $A$  y  $B$  para obtener las matrices reducidas  $A'$  (de tamaño  $n \times p$ ) y  $B'$  (de tamaño  $v \times p$ ). Se proyecta  $\check{X}$  en un espacio de  $p$  dimensiones multiplicándola por  $B'$ , y la matriz resultante  $\check{X}B'$  (de tamaño  $n \times p$ ) se agrupa usando el algoritmo de agrupamiento unidimensional K-means. Así se obtiene la división por filas de la matriz  $X$ : se asume que las filas asignadas por K-means al mismo clúster pertenecen al mismo bloque de  $X$ . El mismo procedimiento se aplica para obtener la división por columnas: se proyecta la matriz traspuesta  $\check{X}^T$  en un espacio de  $p$  dimensiones multiplicándola por  $A'$ , y la matriz resultante  $\check{X}^T A'$  (de tamaño  $v \times p$ ) se agrupa usando el algoritmo de agrupamiento unidimensional K-means. El agrupamiento devuelto por el algoritmo K-means es la partición por columnas de la matriz  $X$ .

Se ha demostrado que este procedimiento es **efectivo en datos biológicos, pero depende de muchos parámetros**. Así, puede ser clave seleccionar el número  $p$  de vectores propios considerados para proyectar  $X$  por filas y por columnas a la hora de buscar la partición de los datos en bloques. Este último paso, basado en el algoritmo original en la técnica de agrupamiento unidimensional K-means, necesita conocer el número de clústeres  $K$  que se buscan. Nótese que  $K$  no tiene por qué ser el mismo número para la partición por filas y por columnas. Por ejemplo, en la Figura 33 hay 2 bloques en las filas y 3 bloques en las columnas. Seleccionar el valor adecuado de  $K$  para ambos usos del algoritmo K-means puede también determinar el resultado del algoritmo.



### Enlace de interés

Documentación del algoritmo de coagrupamiento espectral implementado en la librería scikit-learn para el lenguaje de programación Python:

<https://scikit-learn.org/stable/modules/generated/sklearn.cluster.SpectralCoclustering.html>

## 4.4. Estrategia de agrupamiento de doble sentido

El **agrupamiento acoplado de doble sentido** es una estrategia general que permite usar un algoritmo de agrupamiento estándar para encontrar biclústeres. La idea es aplicar un algoritmo de agrupamiento sobre las filas (matriz en posición normal) y las columnas (matriz traspuesta) de manera iterativa y jerárquica. Cada clúster de filas se obtiene a partir de un clúster previo de columnas y viceversa.

Dado un algoritmo de agrupamiento unidimensional (estándar) y una matriz de datos de entrenamiento, se guarda un conjunto de particiones (clústeres) por filas  $F$  y otro por columnas  $C$ . Inicialmente, se parte de una partición única en cada caso (la que incluye todas las filas y todas las columnas,

respectivamente). En cada iteración, se toma una partición de  $F$  y otra de  $C$ , las cuales representan una submatriz  $X'$ . Se aplica el algoritmo de agrupamiento unidimensional a  $X'$  y a  $(X')^T$ , y los clústeres estables encontrados en cada caso se añaden a  $F$  y  $C$  manteniendo la traza de la jerarquía a través de la cual han surgido los respectivos clústeres y subclústeres. Una correcta implementación implica asegurarse de que un par de clústeres de  $F$  y  $C$  no se considera más de una vez. Finalmente, esta estrategia iterativa para cuando no se detectan más clústeres estables.

Es evidente que el **rendimiento** de esta estrategia depende fuertemente del rendimiento de la estrategia básica de agrupamiento unidimensional y del concepto de estabilidad considerado para detectar clústeres relevantes. Aunque se afirma que cualquier algoritmo de agrupamiento unidimensional puede ser usado como algoritmo básico, la configuración y parametrización de estos hace que muchos algoritmos no sean adecuados. Se han obtenido buenos resultados con el uso del algoritmo de agrupamiento jerárquico SPC. La naturaleza jerárquica de los resultados de esta estrategia suele aportar información relevante sobre el comportamiento del problema estudiado. En concreto, en el caso de la expresión de genes, esta información ha sido usada para entender el comportamiento conjunto de genes y condiciones.



## Capítulo 5

### Aprendizaje semisupervisado

Aunque el aprendizaje automático se divide, con respecto al tipo de supervisión de los datos de entrenamiento, en **dos grandes subgrupos**, la realidad es más flexible y llena de matices. Entre los dos grandes bloques tradicionales, el aprendizaje supervisado y el no supervisado, se encuentra el paradigma del **aprendizaje semisupervisado**.



Tal y como se explicaba en la sección “1.1.1. Aprendizaje semisupervisado”, se trata de un tipo de problema predictivo donde existe la **variable especial o clase**, pero el valor de esta solo se conoce para un subconjunto de ejemplos del conjunto de entrenamiento. En este sentido, el objetivo de este paradigma es aprender un modelo predictivo que, dado un nuevo caso o ejemplo sin valor clase, anticipé o prediga cuál es su valor clase más probable.

Desde este punto de vista, el aprendizaje semisupervisado es más cercano al aprendizaje supervisado. Sin embargo, la falta de supervisión de un subconjunto (habitualmente mayoritario) de ejemplos del conjunto de entrenamiento conlleva el uso de **técnicas de aprendizaje** parecidas a las que se usan en el **aprendizaje no supervisado**. En este sentido, una de las técnicas más recurrentes en el aprendizaje semisupervisado es el algoritmo EM y sus diferentes variantes.

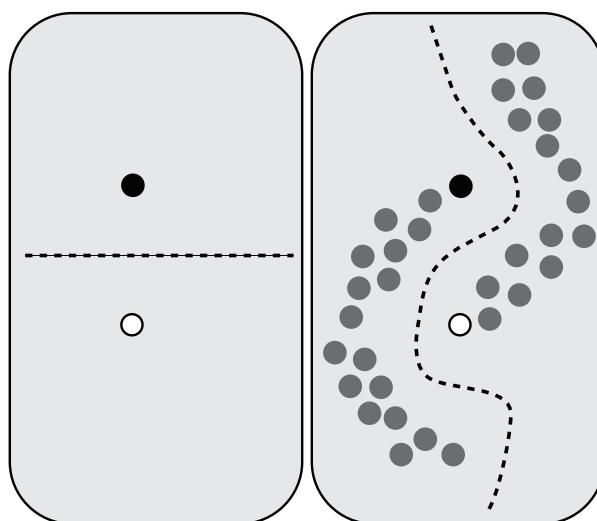


## Ejemplo

Una empresa que gestiona el servicio de correo electrónico de cierto organismo está interesada en aplicar técnicas de aprendizaje automático para mejorar la detección de *spam*. En ese sentido, la empresa ya aplica unas reglas sencillas que permiten filtrar algunos de los correos entrantes, que se colocan en una carpeta separada de correo basura. Sin embargo, no funcionan demasiado bien y filtra como basura correos buenos a la vez que no filtra otros que son evidentemente *spam*. La empresa ha incluido un botón que permite al usuario marcar como *spam* un correo que se ha saltado el filtro y otro botón que le permite marcar como bueno un correo de la carpeta de correo basura.

La empresa quiere aprender de los datos un clasificador de *spam* y, para ello, dispone de todos los correos recibidos en las cuentas del organismo. Para algunos correos, los usuarios han marcado el botón "*Spam*" o "No es *spam*". Los correos así marcados pueden considerarse casos supervisados o etiquetados. Sin embargo, el usuario no está obligado a llevar a cabo esta acción y la mayoría de correos quedan sin etiquetar. Entonces, el clasificador tendrá que ser aprendido a partir de un conjunto de datos de entrada con un subconjunto de ejemplos etiquetados y otro subconjunto (mayoritario) de ejemplos no etiquetados. Esta aplicación de aprendizaje automático es originalmente una de las primeras aplicaciones del mundo real donde se tuvo que recurrir a técnicas de clasificación semisupervisada.

Se podría cuestionar hasta qué punto es necesario **incorporar casos no etiquetados** a un proceso de aprendizaje en el que ya hay disponible un subconjunto de casos etiquetados. La Figura 34 viene a ilustrar la justificación habitual de la mejoría que puede lograrse con la inclusión del subconjunto de casos no etiquetados. Los ejemplos no etiquetados pueden ayudar a definir mejor las fronteras de decisión de un clasificador que, dado un reducido subconjunto de datos etiquetados, podrían ser difíciles de ajustar.



**Figura 34.** Clasificador con solo casos etiquetados y clasificador con casos no etiquetados.

Por Techerin bajo licencia CC BY-SA 3.0. Recuperado de [https://commons.wikimedia.org/wiki/File:Example\\_of\\_unlabeled\\_data\\_in\\_semisupervised\\_learning.png](https://commons.wikimedia.org/wiki/File:Example_of_unlabeled_data_in_semisupervised_learning.png)

A la izquierda, un clasificador evidentemente infraajustado podría ser aprendido solo con casos etiquetados, mientras que, a la derecha, el uso de casos no etiquetados podría mejorar la identificación de la frontera de decisión.

En el presente capítulo se estudiarán **varias aproximaciones** al problema del aprendizaje semisupervisado. Concretamente, se estudiará en detalle cómo abordar el problema utilizando la estrategia EM y algoritmos basados en grafos. También se repasará el concepto de múltiples vistas, cuando los ejemplos de un problema de interés pueden describirse por medio de diferentes conjuntos (complementarios) de variables predictivas-explicativas, y su utilidad en el aprendizaje semisupervisado.

## 5.1. Estrategia EM en aprendizaje semisupervisado

Previamente, en la sección “2.6.2. Algoritmo esperanza-maximización” se estudió la **estrategia EM y su aplicación** para la resolución de problemas de agrupamiento. La característica definitoria de un problema de agrupamiento es el desconocimiento de los grupos o clases que componen el conjunto de datos. En ese caso, se asumía la existencia de  $K$  grupos y, a través del algoritmo EM, se aprendían los parámetros de  $K$  distribuciones de probabilidad gaussianas a la vez que se asignaban probabilísticamente los ejemplos del conjunto de entrenamiento a cada una de las distribuciones o componentes.

La principal diferencia del aprendizaje semisupervisado y, en consecuencia, del uso de la estrategia EM en este contexto es que el número de componentes (clases, en este contexto) es conocido, así como la asignación de parte de los ejemplos a cada una de las clases. La estrategia EM permite aprender un clasificador a la vez que estima la pertenencia de cada uno de los ejemplos no etiquetados a cada una de las clases.

Se trata de un procedimiento iterativo de **dos pasos** (el paso E o esperanza, y el paso M o maximización) que permite obtener los parámetros del modelo de máxima verosimilitud en presencia de datos perdidos. En este caso, los valores perdidos son las etiquetas clase de parte de los ejemplos de entrenamiento.

Se puede demostrar que el algoritmo **converge a un máximo** (local, no necesariamente global) **o a un punto de silla** (casos raros).

Dado un conjunto de datos observados  $X$  y los no observados  $Z$  (valores perdidos), el objetivo es encontrar los parámetros  $\theta$  de un modelo probabilístico  $p$  (dentro de un espacio de parámetros  $\Theta$ ) que mejor expliquen los datos. Para ello, se puede usar el **estimador de máxima verosimilitud**. La verosimilitud se define como:

$$L(\theta; X, Z) = p(X, Z; \theta)$$

Formalmente, los dos **pasos del algoritmo EM** se definen de la siguiente manera:

- **Paso E.** Se define la función  $Q$  de un conjunto de parámetros  $\theta$  como la esperanza condicional (los datos perdidos condicionados en los datos observados y los parámetros actuales del modelo  $\theta^t$ ) de la verosimilitud de  $\theta$ :

$$Q(\theta; \theta^t) = E_{Z|X, \theta^t} [\log L(\theta; X, Z)]$$

- **Paso M.** Se busca el conjunto de parámetros  $\theta^{t+1}$  que aumenta el valor de la función  $Q$  en mayor medida para todo  $\theta \in \Theta$ :

$$Q(\theta^{t+1}; \theta^t) \geq Q(\theta; \theta^t)$$

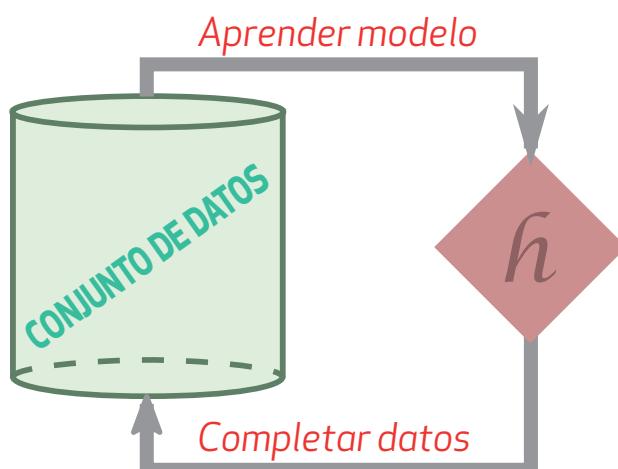
Es decir, se trata de elegir el conjunto de parámetros que maximiza la función  $Q$  dado el conjunto de parámetros previo,  $Q(\cdot; \theta^t)$ :

$$\theta^{t+1} = \operatorname{argmax}_{\theta \in \Theta} Q(\theta; \theta^t)$$



En otras palabras, se podría decir que, en el **paso E**, se estima el valor de los datos perdidos usando el valor esperado dado por el modelo con los parámetros actuales y, en el **paso M**, se estiman unos nuevos parámetros dados los datos completados en el paso E.

Ambos pasos se retroalimentan mutuamente de manera iterativa, tal como se observa en la Figura 35:



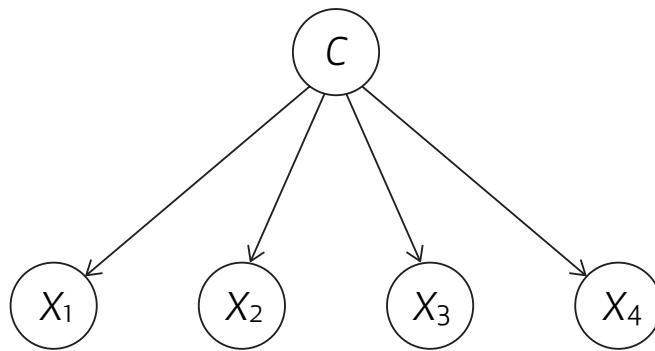
**Figura 35.** Los dos pasos de la estrategia EM en el aprendizaje de un problema semisupervisado.

Consideremos uno de los clasificadores probabilísticos más sencillos para ilustrar el funcionamiento del algoritmo EM, el **clasificador Naive Bayes, en adelante NB**. Este clasificador es una red bayesiana de estructura fija (véase la Figura 36) que asume independencia condicional entre las variables predictivas dada la variable clase,  $X_i \perp X_j | C$  (para todo  $i, j \in \{1, \dots, v\}$ ). La regla de clasificación se obtiene de considerar que la distribución de probabilidad conjunta de variables explicativas y clase puede expresarse de la siguiente manera:

$$p(X_1, \dots, X_v, C) = p(C | X_1, \dots, X_v) \cdot p(X_1, \dots, X_v)$$

Y luego puede reorganizarse para obtener la probabilidad de la clase dada una instancia, así:

$$p(C | X_1, \dots, X_v) = p(X_1, \dots, X_v, C) / p(X_1, \dots, X_v)$$



**Figura 36.** Estructura de un clasificador Naive Bayes con 4 variables explicativas.

Si se aplica la regla de la cadena al numerador del cociente de la parte derecha, este factoriza de la siguiente manera:

$$p(X_1, \dots, X_v, C) = p(X_1 | X_2, \dots, X_v, C) \cdot p(X_2 | X_3, \dots, X_v, C) \cdot \dots \cdot p(X_v | C) \cdot p(C)$$

Pero, dado que el NB asume la independencia condicional de las variables predictivas dada la clase, esta expresión se reduce a:

$$p(X_1, \dots, X_v, C) = p(X_1 | C) \cdot p(X_2 | C) \cdot \dots \cdot p(X_v | C) \cdot p(C) = p(C) \prod_{j=1}^v p(X_j | C)$$

Así pues, dada esta simplificación de la expresión de la probabilidad conjunta, el clasificador, definido como el valor clase  $c$  que maximiza la probabilidad condicional de la clase dadas las variables predictivas, se expresa de la siguiente manera:

$$\hat{h}(x) = \operatorname{argmax}_c p(c | x_1, \dots, x_v) = \operatorname{argmax}_c p(c) \prod_{j=1}^v p(x_j | c) / p(x_1, \dots, x_v)$$

Nótese que el denominador no se ve afectado por la elección de un valor u otro,  $c$ , para la variable clase. Es decir, el valor  $c$  que maximiza  $p(c | x_1, \dots, x_v)$  es independiente de  $p(x_1, \dots, x_v)$ . Por eso, se puede eliminar el denominador y simplificar aún más la **regla de clasificación**, dejándola en la siguiente expresión:

$$\hat{h}(x) = \operatorname{argmax}_c p(c) \prod_{j=1}^v p(x_j | c)$$

La ventaja del clasificador NB es que usa un número reducido de parámetros, cuyos estimadores máximo verosímiles son fácilmente aprendidos a partir de un conjunto de datos de entrenamiento y, aun así, suele mostrar un comportamiento robusto.



### Enlace de interés

Documentación y explicación de las diferentes variantes del clasificador Naive Bayes implementadas en la librería scikit-learn para el lenguaje de programación Python:

[https://scikit-learn.org/stable/modules/naive\\_bayes.html](https://scikit-learn.org/stable/modules/naive_bayes.html)

Para la explicación del algoritmo EM, estamos interesados en un clasificador probabilístico, como el NB, que pueda devolver una clase o la probabilidad de cada uno de los posibles valores de la variable clase. El algoritmo EM se inicializa con unos parámetros al azar o con el conjunto de datos completado de alguna manera concreta. Así, el primer paso (paso E) podría ser asignar cada caso no etiquetado a una clase al azar. Sin embargo, este algoritmo es capaz de trabajar con asignaciones probabilísticas. Es decir, se podría decir que el caso  $x$  pertenece a la clase  $c$  con cierta probabilidad,  $w_x^c$ . Así, una opción válida sería asignar todos los casos no etiquetados a todas las clases con la misma probabilidad ( $1/n$ úmero de clases). En el caso de las instancias del subconjunto etiquetado, la asignación es no probabilística,  $w_{x_i}^c = \mathbb{I}[c_i = c]$ , porque existe evidencia certera acerca del valor clase de esos ejemplos. Dado el conjunto de datos completado probabilísticamente, se aprenden los parámetros del modelo (paso M).

En el caso del NB, los **estimadores máximo verosímiles** son:

$$p(c) = \frac{n_c}{n}; p(x_j | c) = \frac{n_{x_j c}}{n_c}$$

En la fórmula anterior,  $n$  es el número total de instancias,  $n_c = \sum_{i=1}^n w_{x_i}^c$ ,  $n_{x_j c} = \sum_{i=1}^n w_{x_i}^c \cdot \mathbb{I}[x_{ij} = x_j]$  (siendo  $\mathbb{I}[\text{condición}]$  una función que devuelve 1 si la condición se cumple y 0 en cualquier otro caso). Calculando estos estimadores para todos los valores de  $c$  y de  $x_i$  se obtienen los parámetros del modelo predictivo NB.

En el paso E, el clasificador recientemente aprendido se usa para predecir la probabilidad de que cada instancia no etiquetada  $x$  pertenezca a cada etiqueta clase  $c$ ,  $w_x^c = p_{NB}(c | x)$ . En ningún caso la asignación no probabilística de los ejemplos originalmente etiquetados debe ser modificada. De esta manera, se vuelve a obtener un nuevo conjunto de datos completado que será utilizado para aprender un nuevo modelo predictivo NB en el siguiente paso M.

Los pasos E y M **se iteran hasta alcanzar la convergencia**. Se suele aceptar que se alcanza este supuesto cuando la diferencia entre el valor de los parámetros de los dos modelos NB aprendidos en iteraciones consecutivas es menor que cierto umbral  $\epsilon$ .

El principal **inconveniente** de la estrategia EM es el hecho de no tener garantizada la convergencia a un óptimo global: el algoritmo converge a un óptimo local. La solución práctica más habitual consiste en ejecutar múltiples veces el algoritmo con diferentes inicializaciones. En este sentido, diferentes inicializaciones pueden ser diferentes asignaciones de los casos no etiquetados a cada una de las clases o diferentes parámetros iniciales.

## 5.2. Aprendizaje semisupervisado basado en grafos

El aprendizaje semisupervisado se ha abordado con muchas técnicas diferentes. Aparte de las técnicas basadas en el algoritmo EM, los métodos más habituales suelen ser los basados en **propagación de etiquetas**. La idea subyacente es que ejemplos parecidos deben tener etiquetas similares.

Las **técnicas basadas en grafos** construyen una red entre puntos (ejemplos del conjunto de datos de entrenamiento) de acuerdo a cierta medida de similitud y, a través de dicho grafo, se propagan las etiquetas de los casos etiquetados a los no etiquetados aplicando criterios de cercanía y/o similitud.

En aprendizaje semisupervisado, el conjunto de datos está dividido en dos subconjuntos: el de casos etiquetados (de tamaño  $l$ ) y el de no etiquetados (de tamaño  $u$ ), con  $n = l + u$ . Definamos  $k$  como el número de valores posibles (etiquetas) de la variable clase  $C$ . Se puede definir también la matriz  $W$  (de tamaño  $n \times k$ ) como aquella que guarda, en cada una de sus filas, la distribución de probabilidad de las clases para cada uno de los casos del conjunto de entrenamiento. La matriz  $W$  se puede subdividir,  $W = \begin{bmatrix} W^L \\ W^U \end{bmatrix}$ , en el subconjunto de filas relativas a los casos etiquetados y las relativas a los casos no etiquetados. Dada una matriz de transiciones  $T$  (de tamaño  $n \times n$ ), que codifica las similitudes entre ejemplos, las etiquetas se propagan como el producto de ambas matrices,  $TW$ . La matriz  $T$  también puede dividirse,  $T = \begin{bmatrix} T^{LL} & T^{LU} \\ T^{UL} & T^{UU} \end{bmatrix}$ .

Así, el cómputo de las **probabilidades de pertenencia** a cada una de las clases puede calcularse como:

$$W = TW \rightarrow \begin{bmatrix} W^L \\ W^U \end{bmatrix} = \begin{bmatrix} T^{LL} & T^{LU} \\ T^{UL} & T^{UU} \end{bmatrix} \begin{bmatrix} W^L \\ W^U \end{bmatrix}$$

Concretamente, las etiquetas (probabilísticas) de los ejemplos no etiquetados se obtienen iterativamente de la siguiente manera:

$$W^U = T^{UL}W^L + T^{UU}W^U$$

Y, reformulando esta expresión,  $W^U$  se podría calcular mediante una fórmula cerrada:

$$W^U = (I - T^{UU})^{-1}T^{UL}W^L$$

En la fórmula anterior,  $I$  es la matriz identidad, la cual se da siempre que la matriz  $(I - T^{UU})$  sea invertible.

La inicialización de la submatriz  $W^U$  suele hacerse tomando valores aleatorios no negativos, donde cada fila es forzada posteriormente a sumar 1 para garantizar que sea una distribución de probabilidad sobre los diferentes valores clase. El valor de  $W^L$  es determinista  $W_{ij}^L = \mathbb{I}[c_i = j]$ .

La clave de este algoritmo recae por lo tanto en la **construcción de la matriz de transición  $T$** . El primer paso consiste en calcular las distancias entre todos los pares de ejemplos en el conjunto de entrenamiento:

$$D_{ij} = e^{-\frac{x_i - x_j^2}{2\sigma^2}}$$

Esa matriz de distancias se normaliza por fila:

$$P = \text{diag}(D\mathbf{1})^{-1}D$$

En la fórmula anterior,  $\mathbf{1}$  es un vector de valores 1 de tamaño  $n \times 1$ . Nótese que la inversión de una matriz diagonal es la misma matriz con los valores de su diagonal invertidos. En pocas palabras,  $P$  se obtiene de dividir los elementos de cada fila de  $D$  por la suma total de los elementos de la fila,  $P_{ij} = D_{ij} / \sum_{j=1}^n D_{ij}$ . La matriz de transiciones podría definirse como la matriz traspuesta de  $P$ ,  $T = P^T$ , de tal manera que una **etiqueta probabilística  $W_{ic}$**  se podría obtener de la siguiente manera:

$$W_{ic} = P_{1i}W_{1c} + P_{2i}W_{2c} + \dots + P_{ni}W_{nc} = T_{i1}W_{1c} + T_{i2}W_{2c} + \dots + T_{in}W_{nc}$$

Es habitual aplicar un **último paso** para evitar la normalización de los valores de  $W_{ic}$  en cada iteración. Se puede observar con facilidad que el resultado del siguiente sumatorio no es 1:

$$\sum_{c=1}^k W_{ic} = P_{1i} \sum_{c=1}^k W_{1c} + P_{2i} \sum_{c=1}^k W_{2c} + \dots + P_{ni} \sum_{c=1}^k W_{nc} = P_{1i} + P_{2i} + \dots + P_{ni}$$

Sin embargo, es necesario que  $\sum_{c=1}^k W_{ic} = 1$ , ya que cada fila de la matriz  $W$  debe ser una distribución de probabilidad sobre los diferentes valores clase. Para asegurarse de que ese sumatorio devuelva valor 1, se usa la **matriz de transición  $T$  normalizada**:

$$T_{ij} \leftarrow T_{ij} / \sum_{j'=1}^n T_{ij'}$$

Una alternativa al cálculo de  $T$  sería considerar directamente la matriz  $P$  sin trasponer,  $T = P$ . De esta manera, la etiqueta probabilística  $W_{ic}$  se obtendría como:

$$W_{ic} = P_{1i} W_{1c} + P_{2i} W_{2c} + \dots + P_{ni} W_{nc} = T_{1i} W_{1c} + T_{2i} W_{2c} + \dots + P_{ni} W_{nc}$$

En este caso, como la matriz  $P$  está normalizada por filas, el sumatorio es igual a 1:

$$\sum_{c=1}^k W_{ic} = P_{1i} \sum_{c=1}^k W_{1c} + P_{2i} \sum_{c=1}^k W_{2c} + \dots + P_{ni} \sum_{c=1}^k W_{nc} = P_{1i} + P_{2i} + \dots + P_{ni} = 1$$

Y, por lo tanto, la matriz de transición no necesita normalización para que su aplicación produzca una distribución de probabilidad sobre el conjunto de valores clase.

Una vez calculadas las matrices  $T$  y  $W$ , el algoritmo itera la actualización de  $W$  (restringiendo la modificación de  $W$  sólo a la submatriz  $W^U$ ) hasta que los valores no cambian entre iteraciones consecutivas.

El **pseudocódigo** del algoritmo está disponible en la Tabla 8:

#### Tabla 8

*Pseudocódigo del algoritmo de propagación de etiquetas para aprendizaje semisupervisado*

---

##### Algoritmo de propagación de etiquetas

Recibe: conjunto de entrenamiento etiquetado,  $\{(x_1, c_1), \dots, (x_l, c_l)\}$ , y no etiquetado,  $\{(x_u, ?), \dots, (x_w, ?)\}$ ; parámetro de dispersión,  $\sigma$ .

- 
1. Crear la matriz de distancia  $D$ :  $D_{ij} = \exp(-\|x_i - x_j\|^2 / 2\sigma^2)$ ,  $\forall i, j \in \{1, \dots, n\}$ .
  2. Obtener  $P$ , la matriz de distancias normaliza por filas:  $P = \text{diag}(D\mathbf{1})^{-1}D$ .
  3. Obtener  $T$ , la matriz de transición:  $T = P^T$  y  $T_{ij} \leftarrow T_{ij} / \sum_{j'}^n T_{ij'}$ ; o  $T = P$ .
  4. Inicializar  $W$ :  $W^L$  con valores según  $\{c_1, \dots, c_l\}$  y  $W^U$  con valores aleatorios.
  5. Repetir:

- 5.1.  $W \leftarrow TW$  (los valores de la submatriz  $W^L$  no se actualizan).

Hasta convergencia: la diferencia entre la matriz  $W$  de dos iteraciones consecutivas es inferior a cierto umbral  $\epsilon$ .

---

Devuelve: matriz  $W$ .

Cuando, como en esta implementación, se usa la función gaussiana como distancia, el parámetro  $\sigma$  debe ser ajustado. Este parámetro controla la influencia de las instancias cercanas en el cálculo de la etiqueta de un ejemplo no etiquetado.

Cuando el valor de  $\sigma$  tiende a 0, los ejemplos que realmente cuentan en el cálculo de la etiqueta probabilística son los más cercanos. Al contrario, a medida que el valor de  $\sigma$  crece, ejemplos más lejanos adquieren mayor influencia hasta el extremo de  $\sigma = \infty$ , cuando todos los ejemplos tienen la misma influencia.

Dada la relevancia de este parámetro, se han propuesto diferentes técnicas heurísticas para calcular el valor de  $\sigma$ . Una de las más usadas consiste en calcular el valor  $T_{ij}$  mínimo que conecta dos componentes con diferente etiqueta clase. Ese valor mínimo dividido por tres es el valor que se asigna a  $\sigma$ . Otra opción consistiría en aprender de los datos el mejor valor de  $\sigma$ .

Por otro lado, este cómputo podría considerar un valor diferente de  $\sigma$  para cada variable predictiva  $X_v$ . El aprendizaje del mejor conjunto de valores  $\sigma_v$  podría conducir al descubrimiento de variables irrelevantes en el conjunto de datos.

La matriz final  $W$  incluye el etiquetado determinista de los ejemplos etiquetados y un etiquetado probabilístico de los ejemplos no etiquetados. Cuando el objetivo es devolver la etiqueta que el algoritmo asigna a los ejemplos no etiquetados, la solución más directa consiste en elegir la **clase c con mayor probabilidad**:

$$\hat{h}(x_i^U) = \operatorname{argmax}_{c \in \{1, \dots, k\}} W_{ic}^U$$

Sin embargo, al realizar la asignación de esta manera no se conserva ninguna clase de control sobre las proporciones finales de clases.

Esta solución es válida si existen clases bien separadas o el número de ejemplos etiquetados es grande. En otras situaciones suele ser aconsejable usar algún tipo de restricción sobre las proporciones finales de clases.

Así, considerando que conocemos las proporciones reales,  $\{p_1, \dots, p_k\}$  (con  $\sum_{c=1}^k p_c = 1$ ), se han propuesto diferentes **restricciones**:

- **La normalización de la masa de las clases.** Se escalan las columnas de la submatriz  $W^U$  de tal manera que las sumas de las columnas  $\{W_{1c}^U, \dots, W_{kc}^U\}$  (donde  $W_{ic}^U = \sum_{i=1}^U W_{ic}^U$ ) cumplan las proporciones de etiquetas clase,  $\{p_1, \dots, p_k\}$ . Sin embargo, este método tampoco garantiza estrictamente las proporciones de etiquetas,  $\{p_1, \dots, p_k\}$ .
- **La puja por etiquetas.** Se quieren repartir  $\{p_1, \dots, p_k\} \cdot u$  etiquetas de cada clase. Se considera que  $W_{ic}^U$  es la puja que hace el ejemplo  $x_i^U$  por la etiqueta  $c$ . Haciendo un recorrido por las pujas de más alta a más baja, dada una puja  $W_{ic}^U$ , se asigna la clase  $c$  al ejemplo  $x_i^U$  solo si quedan etiquetas  $c$  por repartir. Una vez asignada una clase a un ejemplo, el resto de pujas de ese ejemplo se descartan. Así se garantiza que las proporciones,  $\{p_1, \dots, p_k\}$ , se cumplen de manera estricta.

## 5.3. Coentrenamiento

El aprendizaje semisupervisado se ha abordado tradicionalmente en el mismo entorno que el aprendizaje supervisado: con un conjunto de datos descrito por una serie de variables descriptivas y una variable clase, donde el objetivo es aprender un clasificador. Sin embargo, una tarea intermedia de gran relevancia que las diferentes técnicas abordan de una manera más o menos explícita es el **etiquetado de los ejemplos no etiquetados** del conjunto de entrenamiento.



El **coentrenamiento** (*co-training* en inglés) se sale del enfoque estándar. Dado un conjunto de ejemplos, el coentrenamiento asume que existen varios conjuntos de variables explicativas que permiten describir el espacio de ejemplos de diferente manera. La variable clase es única y común a todas las representaciones y, como en el enfoque estándar, solo se conoce su valor para un subconjunto de ejemplos. La idea consiste en aprovechar las diferentes representaciones de los ejemplos para completar iterativamente el etiquetado de los ejemplos de entrenamiento y aprender mejores clasificadores.



### Ejemplo

Una investigadora está interesada en identificar textos que incluyan mensajes de odio en una red social. Para ello, crea un conjunto de entrenamiento con miles de textos públicos. A la hora de aprender a clasificar cuáles contienen (y cuáles no) mensajes de odio, revisa personalmente un subconjunto de ellos y los etiqueta, aunque la gran mayoría quedan sin etiquetar por la gran cantidad de tiempo que implica leer cada texto para identificar el odio. Así, tiene un conjunto de ejemplos parcialmente etiquetado y, por lo tanto, debe acudir a técnicas de aprendizaje semisupervisado.

Cuando se propone extraer las características del texto para construir la matriz  $X$  descriptiva, tiene varias ideas. Por un lado, decide extraer las palabras más frecuentes y contar el número de apariciones en cada texto. Por otro lado, decide extraer información social de los textos en la red social (por ejemplo, número de "Me gusta", número de comentarios, velocidad de difusión, etc.). La investigadora dispone, así, de dos maneras de representar cada texto del conjunto de entrenamiento, un entorno propicio para el uso de técnicas de coentrenamiento.

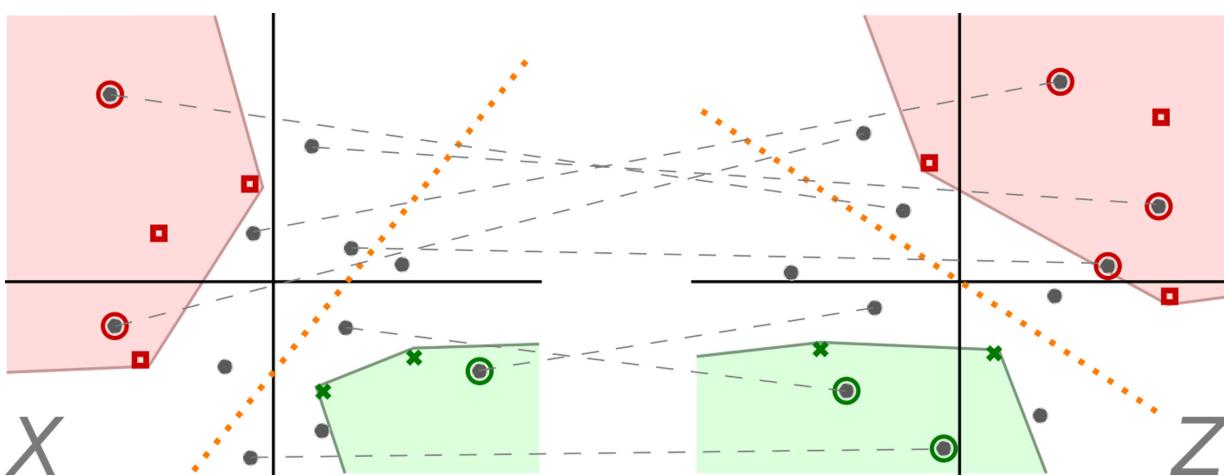
Formalmente, se asume la existencia de diferentes conjuntos de variables explicativas (como  $X = \{X_1, X_2, \dots, X_{v_1}\}$ ,  $Y = \{Y_1, Y_2, \dots, Y_{v_2}\}$  y  $Z = \{Z_1, Z_2, \dots, Z_{v_3}\}$ ) que permiten describir el conjunto de ejemplos de entrenamiento  $D$ . Además, existe una variable clase  $C$  común a todas las descripciones que aporta información sobre la pertenencia de cada caso a una clase o etiqueta. Sin embargo, esta información de etiquetado, en un entorno semisupervisado, no está garantizada. Así, el conjunto de datos para aprendizaje en coentrenamiento se compone, como en el enfoque semisupervisado estándar, por un total de  **$n = l + u$  casos etiquetados y no etiquetados**:

$$D = \{(e_1, c_1), (e_2, c_2), \dots, (e_l, c_l), (e_{l+1}, ?), \dots, (e_{l+u}, ?)\}$$

En la fórmula anterior, cada ejemplo se compone de las diferentes descripciones  $e = (x, y, z) = (x_1, \dots, x_{v_1}, y_1, \dots, y_{v_2}, z_1, \dots, z_{v_3})$  o vistas. Idealmente, las diferentes vistas de los ejemplos serían **independientes** dada la clase.

El objetivo es aprender un clasificador de cada representación del conjunto de ejemplos para ampliar iterativamente el subconjunto de datos etiquetados con el fin último de aprender mejores clasificadores. Este enfoque se basa en la idea de que los clasificadores aprendidos de diferentes representaciones de los datos pueden aprender diferentes fronteras de decisión que permitan identificar con seguridad la etiqueta de ciertos casos no etiquetados.

En la Figura 37 se ilustra la intuición de este enfoque usando **dos espacios de representación** de los ejemplos diferentes, X y Z:



**Figura 37.** Representación de la utilidad de dos espacios de casos (X y Z).

Cada representación permite identificar áreas donde los ejemplos tienen, con alta probabilidad, una u otra etiqueta (las áreas coloreadas). Los ejemplos no etiquetados que caen en una de esas áreas en un espacio pueden no caer en esas áreas de alta probabilidad en el otro espacio. El etiquetado de esos ejemplos originalmente no etiquetados que se encuentran en áreas de alta probabilidad (iterativamente, primero en un espacio y después en el otro) permitiría ir afianzando las fronteras de decisión de los clasificadores aprendidos en ambos espacios gracias a la disposición de un mayor número de casos etiquetados.

Los ejemplos no etiquetados (puntos grises) que un espacio de representación caen en un área donde no hay incertidumbre sobre su etiqueta (áreas coloreadas) se sitúan, en el otro espacio, en áreas no asignadas. El uso combinado de ambos espacios permitiría identificar las clases de los ejemplos y la frontera de decisión real (línea discontinua naranja).

El **algoritmo propuesto por Blum y Mitchell** (1998) es un procedimiento iterativo que aprende de un clasificador por cada vista de los ejemplos disponible. Siguiendo con la formalización anterior, donde a modo de ejemplo se usaban tres vistas ( $X, Y$  y  $Z$ ), se **aprenderían** los clasificadores  $\hat{h}_X$ ,  $\hat{h}_Y$  y  $\hat{h}_Z$  usando solo los ejemplos etiquetados. Cada uno de los clasificadores se usa para **predecir** la probabilidad de pertenecer a cada una de las clases de un subconjunto de ejemplos no etiquetados.

Específicamente, los  $p$  ejemplos con mayor probabilidad de ser positivos según cada uno de los clasificadores se **etiquetan** como positivos y se transfieren al subconjunto de etiquetados. Lo mismo ocurre con la clase negativa: los  $n$  ejemplos con mayor probabilidad de ser negativos según cada uno de los clasificadores se etiquetan como negativos y se transfieren al subconjunto de etiquetados.

Estos pasos (aprendizaje, predicción y etiquetado) se repiten iterativamente hasta alcanzar la convergencia (o un número determinado de pasos), mientras que el subconjunto de ejemplos no etiquetados usado para la predicción se realimenta para conservar su tamaño. Por simplicidad, en estas explicaciones se contempla un problema de clasificación binario, aunque la extensión a un problema multi-clase es directa.

### Tabla 9

Pseudocódigo del algoritmo de coentrenamiento para aprendizaje semisupervisado

---

#### Algoritmo de coentrenamiento

Recibe: conjunto de entrenamiento etiquetado,  $D^L = \{(e_1, c_1), \dots, (e_l, c_l)\}$ , y no etiquetado,  $D^U = \{(e_u, ?), \dots, (e_u, ?)\}$ ; número de positivos a etiquetar,  $p$ ; número de negativos a etiquetar,  $n$ ; número máximo de iteraciones,  $t$ .

1. Obtener un subconjunto aleatorio de ejemplos no etiquetados,  $\check{D}^U \subset D^U$ .
2. Repetir:
  - 2.1. Aprender un clasificador  $\hat{h}_f$  con cada vista  $f$  de los datos en  $D^L$ .
  - 2.2. Con cada clasificador  $\hat{h}_f$ , etiquetar como positivos los  $p$  ejemplos con mayor probabilidad e incluirlos en  $D^L$  (y eliminarlos de  $\check{D}^U$  y  $D^U$ ).
  - 2.3. Con cada clasificador  $\hat{h}_f$ , etiquetar como negativos los  $n$  ejemplos con menor probabilidad e incluirlos en  $D^L$  (y eliminarlos de  $\check{D}^U$  y  $D^U$ ).
  - 2.4. Pasar  $(n + p) \times \text{Num_Vistas}$  nuevos ejemplos desde  $D^U$  y  $\check{D}^U$ .

Hasta convergencia o número máximo de iteraciones  $t$ .

Devuelve:  $D^L$  y  $D^U$ , junto con los clasificadores aprendidos en la iteración final.

---



La suposición de que las **vistas son condicionalmente independientes** dada la clase es muy fuerte. Este es precisamente uno de los puntos débiles de esta aproximación. El coentrenamiento necesita que los clasificadores no estén correlacionados para que un mismo ejemplo pueda obtener diferente probabilidad de ser positivo según el clasificador considerado. Este hecho permite que este procedimiento iterativo mejore progresivamente con el uso de un conjunto etiquetado cada vez mayor y más variado. Sin embargo, no es fácil encontrar subconjuntos de variables condicionalmente independientes. Esta condición suele suavizarse y se trata de conseguir clasificadores que difieran en sus predicciones en un tanto por ciento. Incluso en estas condiciones donde la condición de independencia se quiebra, está demostrado que el coentrenamiento es útil y ventajoso.



## Capítulo 6

### Otras técnicas no supervisadas



El concepto de aprendizaje no supervisado suele, con cierta frecuencia, considerarse un sinónimo de agrupamiento.

La realidad impone que la mayoría de las veces que se pretende analizar un conjunto de datos no etiquetado se suele acudir a técnicas de agrupamiento, aunque, como se ha visto previamente con el análisis de componentes, existen **otros tipos de técnicas para analizar datos no supervisados**.

En este último capítulo se estudiarán problemas tradicionales del análisis de datos exploratorio con la característica principal de que el conjunto de datos no se representa mediante una matriz X.

Para cada problema, se estudiarán las técnicas más populares diseñadas para su análisis.

Aunque existe una amplia variedad de tipos de datos y sus respectivas técnicas de análisis, en este documento nos centraremos en dos de los problemas más comunes: el estudio de grafos y el algoritmo PageRank, por un lado, y el análisis de reglas de asociación y el algoritmo Apriori, por el otro.

## 6.1. Análisis de grafos

De manera natural, mucha información de diferentes dominios reales se podría codificar mediante un **grafo**. Cada nodo del grafo representaría una entidad de la aplicación que se considere. Por su parte, cada arista del grafo implicaría una relación de cierto tipo entre las dos entidades que conecta. La expresividad de un grafo a la hora de establecer relaciones entre entidades es difícilmente representable en una matriz cuadrada de datos como las que se usan en los entornos clásicos de aprendizaje automático. Es por ello que se ha desarrollado todo un subcampo de análisis de datos representados mediante grafos.



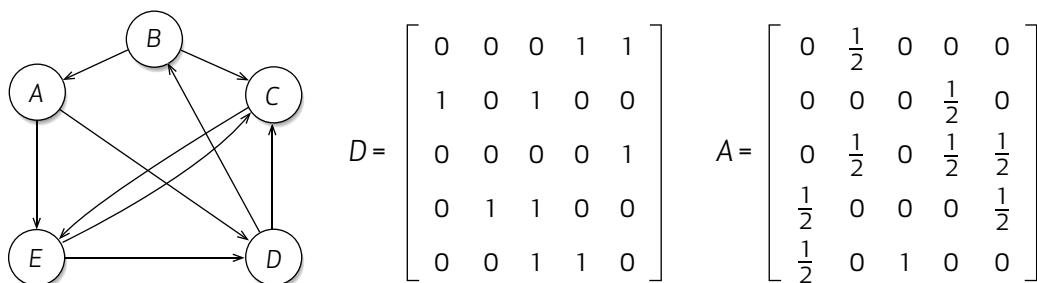
### Ejemplo

En el mundo digital, son muchas las aplicaciones que pueden representarse mediante un grafo. El ejemplo más básico en este entorno se da en internet. Si consideramos cada sitio web como una entidad o nodo del grafo, y los hipervínculos entre webs como enlaces o aristas entre nodos, se puede representar internet como un enorme grafo. En este caso, la red o grafo resultante sería dirigido, ya que los enlaces tienen una dirección: una página web vincula en su contenido otra página web a conciencia, pero la administradora de una página web no puede controlar qué otras webs apuntan a la suya.

Un grafo puede ser representado por medio de una **matriz de adyacencias**  $D$ , con tantas filas y columnas como entidades ( $n$ ) existen en el problema de estudio, y donde  $D_{ij} = 1$  si existe una relación desde el nodo  $i$  al nodo  $j$ . Si la matriz es simétrica,  $D_{ij} = D_{ji}$ , se considera que las relaciones son no dirigidas. Otra representación equivalente a la matriz de adyacencias es la matriz de transiciones  $A$ , que, con el mismo tamaño ( $n \times n$ ), codifica en cada celda  $A_{ij}$  la probabilidad, cuando se está en el nodo  $i$ , de moverse al nodo  $j$  si se siguiese aleatoriamente cualquiera de las aristas que conectan el nodo  $i$  con otros nodos.

Es decir, si existen tres aristas que conectan el nodo  $i$  con otros tantos nodos ( $j, k, l$ ), la probabilidad de transitar a uno de esos tres nodos desde  $i$  es  $1$  entre el número de aristas salientes  $A_{ij} = A_{ik} = A_{il} = \frac{1}{\#Aristas} = \frac{1}{3}$ , y la probabilidad de transitar a cualquier otro nodo desde el nodo  $i$  es  $0$ .

En la Figura 38 se puede ver la matriz de adyacencias y la de transiciones de un grafo de ejemplo.



**Figura 38.** Grafo, su matriz de adyacencia y su matriz de transición.



Con datos representados por medio de grafos, las tareas que se pueden afrontar se asemejan, salvando la distancia del tipo de dato disponible, a las tareas del aprendizaje automático clásico (agrupamiento, clasificación, etc.). Sin embargo, aparecen también **tareas genuinas** de los datos de tipo red o grafo, como encontrar subgrafos relevantes o subgrafos repetidos, o descubrir nuevas relaciones entre nodos del grafo.

Existen muchos **tipos de grafos**: los que tienen diferentes tipos de entidades (nodos), los que tienen distintos tipos de relaciones (aristas), los que tienen relaciones dirigidas (tienen validez en un único sentido), los que pueden tener relaciones con diferentes pesos, etc. Aunque este tipo de información añade riqueza al grafo y permite llevar a cabo un análisis más informado, en este documento estudiaremos un popular algoritmo que usa grafos sencillos con un único tipo de entidades y relaciones dirigidas: el algoritmo PageRank.

### 6.1.1. Algoritmo PageRank

El PageRank calcula para cada página web la probabilidad de que un internauta acabe en dicha web tras navegar aleatoriamente por internet desde un punto de inicio elegido al azar. La probabilidad asignada a cada web puede ser considerada como una medida de la relevancia de la página. Este fue el objetivo inicial para el cual fue propuesto el algoritmo *PageRank*: establecer una medida de relevancia que pudiese ser usada como criterio para ordenar las páginas webs según su importancia. Sin embargo, el algoritmo subyacente se puede generalizar de tal manera que puede ser aplicado en cualquier grafo.

En general, este método permite explorar el conjunto de nodos de un grafo para establecer una medida (y una escala) de la **importancia de cada uno de los nodos en relación al resto**. El concepto de importancia se entiende como una noción de centralidad en las relaciones del grafo: la probabilidad de visitar un nodo siguiendo un camino aleatorio a través del nodo. Un nodo importante, o bien conectado, tiene más probabilidad de ser visitado en un camino aleatorio que un nodo poco conectado o irrelevante.

Formalmente, el algoritmo PageRank devuelve una **distribución de probabilidad sobre los nodos de un grafo**. El valor que recibe cada nodo  $i$  es la probabilidad de que un camino aleatorio que comienza en cualquier nodo acabe en ese nodo  $i$ . Se puede representar mediante un vector (de probabilidades) de tamaño  $n$  (número de entidades o nodos en el grafo). Por definición, ese **vector  $P$**  es el que, multiplicado por la matriz de transiciones  $A$ , devuelve el mismo vector  $P$ :

$$AP = P$$

Resolver esta igualdad o el sistema de ecuaciones equivalente puede ser complejo si el número de nodos  $n$  es muy grande. Por eso, existen procedimientos que **aproximan el valor** del vector de PageRank.

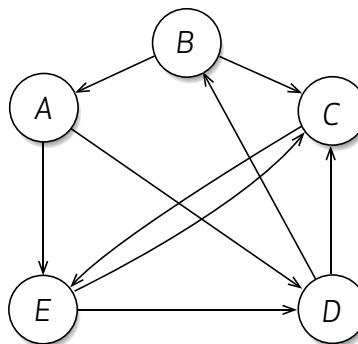
Uno de ellos se basa en una propiedad interesante de la matriz de transiciones: el cuadrado de la matriz de transiciones ( $A^2 = AA$ ) devuelve la matriz de transiciones en dos pasos. El valor  $A_{ij}^2$  representa la probabilidad de llegar al nodo  $j$  en dos pasos y empezando en el nodo  $i$  (es decir, realizar el

camino  $i \rightarrow k \rightarrow j$ , donde  $k$  es un tercer nodo intermedio cualquiera). Esta propiedad se cumple para cualquier exponente:  $A^m$  es la matriz de transiciones en  $m$  pasos ( $A_{ij}^m$  es la probabilidad de llegar al nodo  $j$  tras  $m$  pasos empezando en el nodo  $i$ ). Asumiendo que  $A^*$  es la matriz de transiciones en infinitos pasos, el **vector de PageRank  $P$**  se puede calcular de la siguiente manera:

$$A^*F = P$$

En la fórmula anterior,  $F$  es un vector de inicialización que señala dónde comienza el camino. Como, por definición, PageRank considera caminos aleatorios con origen también aleatorio, el vector  $F$  se construye como el vector de probabilidad uniforme  $F = \left[ \frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n} \right]^T$ . De esta manera, el resultado de usar cualquier valor  $m$  finito para el producto  $A^m F$  podría ser considerado una aproximación al vector PageRank real. En la medida en que  $m$  tiende a infinito ( $m \rightarrow \infty$ ), la aproximación será más ajustada.

Otro procedimiento que permite realizar una estimación aproximada del vector de probabilidades de PageRank consiste en realizar  $r$  caminos aleatorios de longitud  $m$  y guardar un contador de las veces que cada nodo es visitado. Al finalizar el proceso, los conteos se normalizan para que sumen 1 y obtener el vector de probabilidades  $P$ . Esta manera de simular los caminos de manera **manual**, similar a un muestreo, tiende a acercarse al vector real  $P$  a medida que el número de pasos  $m$  y, sobre todo, el número de caminos  $r$  tienden ambos a infinito ( $m \rightarrow \infty, r \rightarrow \infty$ ).



**Figura 39.** Grafo donde el nodo  $B$  no tiene ninguna arista entrante y la única que tiene el nodo  $A$  proviene de  $B$ .

Los nodos  $A$  y  $B$  quedarían aislados y obtendrían un PageRank de 0 si no se consideraran los saltos aleatorios.

Sin embargo, este algoritmo tiene un problema que consiste en que ciertos nodos a los que no se puede acceder desde ningún otro nodo (ver Figura 39) quedan sin **asignación de probabilidad** ( $P_i = 0$ ). Este hecho, además de no ser realista, influye en cálculos posteriores, como puede ser la escala o la ordenación de los nodos según su importancia. Por eso, se suele considerar una **variación en la concepción de los caminos aleatorios**: un camino aleatorio puede hacer un salto aleatorio (dejar el nodo actual y continuar desde cualquier otro) en cualquier momento.

Desde el punto de vista de la persona que navega en la web, esta consideración tendría en cuenta que una persona puede moverse a otra web sin seguir los enlaces de la web actual, simplemente,

introduciendo su dirección. Teniendo esta nueva consideración en cuenta, el algoritmo se redefine sustituyendo la matriz de transiciones  $A$  por una **nueva matriz  $M$**  compuesta por la propia matriz  $A$  y nuevas transiciones de cualquier nodo a cualquier otro con cierta probabilidad  $p$ :

$$M = (1-p)A + pB$$

En la fórmula anterior,  $B$  es una matriz cuadrada ( $n \times n$ ) que toma en todas sus posiciones el valor  $1/n$ . Es decir, se combina la matriz original de transiciones  $A$  con otra matriz alternativa  $B$  que considera la posibilidad de transitar de cualquier nodo a cualquier otro con la misma probabilidad (salto aleatorio).

La **elección del parámetro  $p$** , o la probabilidad de salto aleatorio, es clave en esta concepción. Aunque puede tomar cualquier valor entre 0 y 1, se suelen considerar valores pequeños que permitan realizar caminos relativamente largos por el grafo antes de que un salto aleatorio sea altamente esperable.

Dada la matriz  $M$ , se redefinen las diferentes técnicas para el cálculo del vector de probabilidades de PageRank  $P$ . El método algebraico exacto es:

$$MP = P$$

La versión aproximada basada en la potencia de la matriz de transiciones aplica el cálculo de la potencia a la matriz  $M$ :

$$M^*F = P$$

En la segunda versión aproximada, basada en la simulación de  $r$  caminos de longitud  $m$ , se considera que todas las transiciones son posibles en cada paso con la probabilidad correspondiente asignada en la fila  $M_i$  de la nueva matriz de transiciones.

De esta manera, se obtiene un valor de probabilidad para cada nodo a pesar de que no sea accesible desde el resto. Por ejemplo, los nodos  $A$  y  $B$  del grafo ilustrado en la Figura 39 recibirían las probabilidades 0,043 y 0,03, lo que permitiría, entre otras cosas, establecer que  $A$  es más importante que  $B$ . Sin considerar los saltos aleatorios, tanto el nodo  $A$  como el nodo  $B$  obtendrían un valor de PageRank de 0, por lo que no se podría establecer ninguna relación entre ellos.

## 6.2. Análisis de reglas de asociación

El ejemplo más habitual a la hora de ilustrar la relevancia de las **reglas de asociación** es el de la cesta de la compra. Cuando un supermercado quiere estudiar los patrones de consumo de sus clientes, simplemente dispone, en el peor de los casos, de un conjunto de compras, donde cada compra no es más que el conjunto de productos que compró cierto cliente.

En este tipo de entorno, la manera más común de extraer información útil es mediante el uso de reglas de asociación. Siguiendo con el ejemplo del supermercado, una regla de asociación explicaría que cada vez que alguien compra leche, esa misma persona suele llevarse también un paquete de galletas. Es decir, **relaciona un escenario** (por ejemplo, un cliente compra leche) **con otro** (por ejemplo, un cliente compra galletas). Las reglas de asociación se expresan de la siguiente manera:

$$\text{leche} \Rightarrow \text{galletas}$$

Así pues, *leche* es el antecedente y *galletas* el consecuente. Aunque para este ejemplo solo se haya usado un ítem o producto tanto en el antecedente como en el consecuente, se puede observar más de un ítem en ambos casos (mediante el operador lógico de conjunción  $\wedge$ ):

$$\text{leche} \wedge \text{cereales} \Rightarrow \text{galletas}$$



Hay que tener en cuenta que las reglas de asociación **no son necesariamente deterministas**. Es decir, cada vez que se da el antecedente, no siempre tiene que darse el consecuente. Por eso, es subjetivo considerar que una regla de asociación es realmente válida o no.

Sin embargo, existen múltiples métricas para medir la relevancia de una regla de asociación. Dado un conjunto de **transacciones**  $S = \{T_1, T_2, \dots, T_n\}$  (siguiendo el ejemplo, un conjunto de compras) sobre un conjunto finito de ítems  $I = \{i_1, i_2, \dots, i_m\}$  (los productos a la venta), las métricas más comunes son:

- **Soporte** (*support* en inglés) de un subconjunto de ítems  $X \subseteq I$ , que mide la proporción de transacciones del conjunto  $S$  en las que aparece el subconjunto de ítems  $X$ :

$$\text{soporte}(X; S) = \frac{|\{T \in S : X \subseteq T\}|}{|S|}$$

Se puede entender como la probabilidad conjunta (marginal) de  $X$ . El soporte de una regla de asociación  $X \Rightarrow Y$  se expresa como el soporte de la unión de antecedente y consecuente:  $\text{soporte}(X \Rightarrow Y; S) = \text{soporte}(X \cup Y; S)$ .

- **Confianza** (*confidence* en inglés) de una regla de asociación  $X \Rightarrow Y$ , que mide con cuánta frecuencia se cumple la regla de asociación:

$$\text{confianza}(X \Rightarrow Y; S) = \frac{\text{soporte}(X \cup Y; S)}{\text{soporte}(X; S)}$$

La confianza tiende a 1 a medida que se observa con mayor frecuencia  $Y$  cada vez que aparece  $X$  en las transacciones del conjunto  $S$ . Se puede entender también como la probabilidad condicionada de  $Y$  dado  $X$ .

- **Mejora** (*lift* en inglés) de una regla de asociación  $X \Rightarrow Y$ , o ratio del soporte como indicador de la creencia de que la regla pueda ser producto del azar:

$$\text{mejora}(X \Rightarrow Y; S) = \frac{\text{soporte}(X \cup Y; S)}{\text{soporte}(X; S) \times \text{soporte}(Y; S)}$$

En la fórmula anterior, un valor de mejora igual a 1 indica que ambos subconjuntos de ítems se relacionan al azar, un valor mayor que 1 indica cierto grado de coocurrencia y un valor menor que 1 indica complementariedad (cuando se observa un subconjunto, no se observa el otro).

El uso de estas y otras métricas permite identificar y discriminar reglas de asociación que pueden ser de interés. Tradicionalmente, las reglas de asociación que se consideran interesantes suelen mostrar un nivel de soporte y confianza mínimos. Fijar cuál es el nivel mínimo aceptable en ambos casos suele ser tarea de los expertos en el área de aplicación. Las reglas que superan los umbrales fijados de soporte y confianza mínimos se conocen como **reglas fuertes**.

El aprendizaje de reglas de asociación permite identificar de manera automática **patrones relevantes** que cumplen con la definición propuesta apoyándose en el uso de las métricas descritas. El descubrimiento de este tipo de regularidades en las transacciones de una empresa le permitiría llevar a cabo ofertas sobre ciertos productos de manera planificada e informada, colocar los productos de manera estratégica o realizar recomendaciones a partir del comportamiento extraído de transacciones previas.

En general, la tarea del descubrimiento de reglas de asociación puede verse como un proceso de dos etapas. En la primera, el objetivo es **identificar** todos los conjuntos de ítems que aparecen con cierta frecuencia en un conjunto de transacciones. Una vez identificados todos los conjuntos frecuentes de ítems, se **construyen a partir de ellos reglas de asociación fuertes**. El algoritmo más popular para el descubrimiento de reglas de asociación es el algoritmo Apriori.

## 6.2.1. Algoritmo Apriori

El **algoritmo Apriori** es un método de búsqueda en anchura que permite identificar reglas de asociación fuertes en un conjunto de transacciones basándose en la idea de que un conjunto solo puede ser frecuente (aparecer un cierto número de veces) si todos sus subconjuntos también lo son. Esta simple idea permite reducir el espacio de búsqueda en gran medida y abordar el problema de manera iterativa-aglomerativa.

El método se basa en la definición de conjuntos de ítems (*itemsets* en inglés) de tamaño  $k$ : un  $k$ -ítem es un conjunto que incluye exactamente  $k$  ítems. Un  $k$ -ítem es frecuente si aparece en al menos  $\epsilon$  transacciones del conjunto de referencia  $S$ .

Dado un conjunto de transacciones,  $S$ , y el umbral de soporte,  $\epsilon$ , se obtienen los 1-ítems que superan el umbral de soporte,  $\epsilon$ . Estos son los ítems individuales que aparecen en al menos  $\epsilon$  transacciones. A continuación, se buscan iterativamente los  $k$ -ítems frecuentes para valores crecientes de  $k > 1$ . Tomando todos los  $(k - 1)$ -ítems frecuentes, se construyen todos los  $k$ -ítems posibles de tal manera que, para cada  $k$ -ítem candidato  $T$ , todos los subconjuntos (ítems) de tamaño  $k - 1$  son  $(k - 1)$ -ítems frecuentes. En la Tabla 10 se muestra el pseudocódigo de este algoritmo, que se detiene cuando, para un valor de  $k$  concreto, no se obtiene ningún  $k$ -ítem frecuente.

**Tabla 10**  
Pseudocódigo del algoritmo Apriori

### Algoritmo Apriori

Recibe: conjunto de transacciones,  $S = \{T_1, T_2, \dots, T_n\}$ ; umbral de soporte,  $\epsilon$ .

1.  $L_1 \leftarrow$  Todos los 1-ítems (dado  $\epsilon$ ).
2. Para  $k = 2, 3, \dots$

- 2.1. Crear un conjunto de  $k$ -ítems candidatos a partir de los  $(k - 1)$ -ítems fuertes obtenidos en el paso anterior:

$$C_k \leftarrow \{T = T^{k-1} \cup \{i\} : (T^{k-1} \in L_{k-1}) \wedge (i \notin T^{k-1}) \wedge (\forall j \in T, (T = \{j\}) \in L_{k-1})\}$$

- 2.2. Contar las apariciones de los  $k$ -ítems candidatos de  $C_k$  en el conjunto de transacciones  $S$ .

- 2.3. Filtrar los  $k$ -ítems de  $C_k$  que son realmente fuertes o frecuentes:

$$L_k = \{T \in C_k : \text{soporte}(T) > \epsilon\}$$

Parar si  $L_k = \emptyset$ .

Devuelve:  $k$ -ítems frecuentes, para todo  $k$ .

El algoritmo Apriori devuelve un conjunto de ítems frecuentes. Posteriormente, algún tipo de tratamiento debe aplicarse para obtener reglas de asociación a partir de esos conjuntos frecuentes de ítems.

Apriori es ciertamente el método de descubrimiento de reglas de asociación más popular en la literatura científica relacionada, aunque adolece de **varios problemas**. El listado de todos los posibles candidatos en cada paso puede ser un procedimiento exhaustivo de gran complejidad computacional. En ese sentido, es habitual buscar los ítems candidatos solamente como una combinación de los ítems frecuentes de la iteración anterior ( $k - 1$ ). Otra crítica habitual es la necesidad del algoritmo de explorar múltiples veces el conjunto de transacciones de referencia  $S$  para contar el número de apariciones de los diferentes ítems considerados a lo largo de todo el proceso. Diferentes mejoras propuestas consideran subconjuntos de transacciones en la búsqueda de patrones o mantener solo las transacciones en las que es posible descubrir algún patrón (no se encontrarán ítems de tamaño  $k$  en una transacción que no tenía ninguna de tamaño  $k - 1$ ).

# Glosario



## Agrupamiento

Conjunto de grupos o clústeres de ejemplos.

## Agrupación, grupo o clúster

Grupo o conjunto de ejemplos que comparten características comunes.

## Aprendizaje

Dada una familia de modelos y un conjunto de datos del problema a estudiar, consiste en configurar los parámetros y la estructura característicos de la familia de modelos de acuerdo a la información extraída del conjunto de datos con el objetivo de construir un modelo específico que se adapte a las características del problema estudiado.

## Búsqueda exhaustiva

Dado un problema en el que existen diferentes soluciones posibles, la búsqueda exhaustiva evalúa todas y cada una de ellas y elige la solución óptima.

## Búsqueda heurística

Dado un problema en el que existen diferentes soluciones posibles (habitualmente, un gran número), la búsqueda heurística diseña una estrategia que recorre el espacio de soluciones de manera más o menos inteligente y evalúa las soluciones que encuentra en su camino. Devuelve la mejor solución encontrada. Estas técnicas no garantizan el descubrimiento de la mejor solución global, y a pesar de ello son altamente valoradas por su bajo coste computacional.

## Caso

Véase *ejemplo*.

## Centro

En análisis de agrupamiento, el centro es un ejemplo del problema estudiado que es considerado como el representante de un clúster por su situación en el punto medio de este de acuerdo con cierta distancia de disimilitud o similitud. El centro de un clúster se calcula y no tiene por qué formar parte del conjunto de entrenamiento.

## Centroide

De manera equivalente al concepto de *centro*, el centroide es un ejemplo del problema estudiado que es considerado como el representante de un clúster por su situación cercana al punto medio de este de acuerdo con cierta distancia de disimilitud o similitud, con la condición añadida de que el ejemplo debe ser necesariamente un elemento del conjunto de entrenamiento.

## Convexo

Se dice de un clúster en el que la línea que se traza entre dos elementos cualesquiera del clúster pasa siempre por áreas que pertenecen al clúster.

## Covarianza

Es la generalización del concepto de varianza a múltiples dimensiones. Mide la variabilidad conjunta de dos dimensiones de un vector aleatorio (o de dos variables). Aumenta (tiende a 1) cuando los valores de ambas variables o dimensiones crecen o decrecen de manera conjunta y disminuye (tiende a -1) cuando los valores de una variable o dimensión crecen a la vez que los de la otra decrecen.

## Dendrograma

Se trata de un árbol binario usado para representar el resultado de un algoritmo de agrupamiento jerárquico donde cada nodo representa un clúster. Los dos hijos de cada nodo se reparten los ejemplos que se incluyen en el nodo padre. El árbol se representa gráficamente de tal manera que la distancia entre dos nodos y su padre es proporcional a la disimilitud interclúster de los dos nodos hijos.

## Disimilitud

La disimilitud entre dos ejemplos o grupos de ejemplos determina cuánto se diferencian entre ellos. Una medida de disimilitud normalizada entre 0 y 1 devuelve valor 0 cuando se comparan dos ejemplos (o grupos de ejemplos) iguales y 1 cuando la diferencia entre ambos es máxima. Aunque la *distancia* mide también la diferencia entre (grupos de) ejemplos, la similitud es un concepto más amplio. Es el concepto contrario a *similitud*.

## Dispersión

Se dice de un grupo que es homogéneo si los elementos que lo componen son similares entre ellos. Es el concepto contrario a la *homogeneidad*.

## Distancia

Se define formalmente como una función  $d$  de dos elementos  $(a, b)$  en los números reales  $\mathbb{R}$  que es no negativa ( $d(a, b) \geq 0$ ), simétrica ( $d(a, b) = d(b, a)$ ), inexistente cuando se compara consigo misma ( $d(a, a) = 0$ ) y cumple la desigualdad triangular ( $d(a, c) \geq d(a, b) + d(b, c)$ ). Es un caso específico de medida de *disimilitud*, ya que mide la diferencia entre elementos.

## Ejemplo

Cada uno de los posibles casos del problema estudiado, que se codifica para el análisis mediante un vector de características que asigna un valor a cada una de las variables aleatorias que se usan en el estudio para describir el problema. Es sinónimo de *instancia*.

## Entrenamiento

Véase *aprendizaje*.

## Etiqueta, categoría o valor clase

Valor posible de la variable especial clase en un problema de clasificación supervisada. Representa cada una de las posibles categorías a las cuales se pueden asignar los casos del problema.

## Homogeneidad

Se dice de un grupo que es homogéneo si los elementos del mismo son similares entre ellos. Es el concepto contrario a la *dispersión*.

## Instancia

Véase *ejemplo*.

## Interclúster

Se dice de la propiedad compartida entre distintos grupos y los elementos de los mismos. Por ejemplo, la dispersión interclúster entre dos clústeres mide la disimilitud entre los clústeres (y sus respectivos subconjuntos de ejemplos).

## Intraclúster

Se dice de la propiedad compartida entre los elementos del mismo grupo o clúster. Por ejemplo, la dispersión intraclúster mide la disimilitud media entre los elementos de un mismo clúster.

## Similitud

La similitud entre dos ejemplos o grupos de ejemplos determina cuánto se parecen entre ellos. Una medida de similitud normalizada entre 0 y 1 devuelve valor 1 cuando se comparan dos ejemplos (o grupos de ejemplos) iguales y 0 cuando la diferencia entre ambos sea máxima. Es el concepto contrario a *disimilitud*.

## Supervisión

La información de supervisión, que proporciona un experto, trata sobre la distribución de los ejemplos en clases y guía el aprendizaje supervisado. En la práctica, la información de supervisión son las etiquetas o valores respuesta que un experto asigna a los ejemplos del conjunto de datos de entrenamiento.

## Supervisado

Se dice de un conjunto o ejemplo que tiene información de supervisión, es decir, que tiene un valor (etiqueta o respuesta) asignado para la variable clase o respuesta.

## Validación cruzada

Procedimiento de validación de un método de aprendizaje usado para la selección de modelos o la simple evaluación de este cuando el conjunto de datos es relativamente pequeño. Consiste en dividir el conjunto de datos en  $k$  subconjuntos. Cada subconjunto se utiliza como datos de validación de un modelo aprendido con el resto de subconjuntos. Así, se obtienen  $k$  modelos y sus respectivas medidas de rendimiento obtenidas a partir de datos que no han sido usados para aprender los modelos. Se puede obtener una medida de evaluación global tomando la media.

## Valor atípico

Se trata de un caso o elemento del conjunto de entrenamiento que se diferencia considerablemente del resto de ejemplos del conjunto de datos. Existen dos consideraciones habituales para los valores atípicos: los casos extremadamente diferentes del resto que son un producto de algún error de medición (inválidos) y los casos peculiares muy poco comunes en el problema de estudio que no son producto de un error de medición sino de un evento extraño poco habitual. En inglés se llama *outlier*.

## Valor respuesta

Valor que toma la variable respuesta o dependiente. Se usa principalmente en problemas de regresión.

## Variable aleatoria

Concepto matemático que hace referencia a una variable que toma valor como producto de un fenómeno aleatorio, es decir, que con cierta probabilidad toma un valor u otro. Las variables aleatorias pueden ser continuas o discretas.

## Variable clase

Variable respuesta o dependiente de un problema de clasificación supervisada que toma el valor de la categoría a la que pertenece cada ejemplo. Es una variable aleatoria discreta y cada uno de los posibles valores que toma se conoce como *etiqueta*.

## Variable dependiente

Véase *variable respuesta*.

## Variable descriptora

Variable aleatoria que describe una característica de los casos del problema de estudio. También se llama *variable independiente* o *variable predictiva*.

## Variable independiente

Véase *variable descriptora*.

## Variable predictiva

Véase *variable descriptora*.

## Variable respuesta

Variable aleatoria especial de un problema que toma el valor de respuesta de cada ejemplo. También se llama *variable dependiente* o *variable objetivo*. En la etapa de aprendizaje, el valor de esta variable es la información de supervisión que guía el entrenamiento del modelo. En la etapa de predicción, el valor de esta variable no se conoce y es el que debe ser predicho. Este nombre se usa principalmente en regresión. En clasificación, véase *variable clase*.

## Varianza

Medida de la variabilidad de una *variable aleatoria* que calcula cuánto se alejan los valores que toma la variable del valor medio. Se calcula como la esperanza de la distancia al cuadrado entre la variable aleatoria y su media.

## Enlaces de interés



### Kaggle

Portal de competiciones abiertas (remuneradas o no) de análisis de datos. Además, funciona como una comunidad colaborativa de aprendizaje donde existen problemas de análisis resueltos mediante diferentes técnicas.

<https://www.kaggle.com>

### UC Irvine Machine Learning Repository

Repositorio de la Universidad de California en Irvine que guarda un gran número de conjuntos de datos abiertos de diferentes ámbitos y dominios. Muchos de los conjuntos presentes en este repositorio forman parte de diferentes benchmarks en la investigación de aprendizaje automático.

<http://archive.ics.uci.edu/ml/>

### Asignatura de Aprendizaje Automático en Carnegie Mellon University

Información y videolecturas de la asignatura de aprendizaje automático de la prestigiosa Carnegie Mellon University. Muchos de los temas y algoritmos tratados en esta asignatura son explicados por los mayores expertos en el área de aprendizaje automático.

<http://www.cs.cmu.edu/~ninanf/courses/601sp15/>

### Asignatura de Aprendizaje Automático en Stanford University

Información y videolecturas de la asignatura de aprendizaje automático de la prestigiosa Universidad de Stanford. Muchos de los temas y algoritmos tratados en esta asignatura son explicados por los mayores expertos en el área de aprendizaje automático (véanse, principalmente, las clases 12 a 15).

<https://see.stanford.edu/Course/CS229/>

## Bibliografía



- Arthur, D.; y Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding. En *Proceedings of the 18th annual ACM-SIAM Symposium on Discrete Algorithms* (pp. 1027-1035). Philadelphia: Society for Industrial and Applied Mathematics.
- Blum, A.; y Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. En *Proceedings of the 11th annual Conference on Computational Learning Theory* (pp. 92-100). Madison: Association for Computing Machinery.
- Cheng, Y.; y Church, G. M. (2000). Biclustering of expression data. En *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology* (pp. 93-103). San Diego: AAAI Press.
- Comaniciu, D.; y Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5), 603-619.
- Ester, M.; Kriegel, H. P.; Sander, J.; y Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. En *International Conference on Knowledge Discovery and Data Mining* (pp. 226-231). Portland: AAAI Press.
- Frey, B. J.; y Dueck D. (2007). Clustering by Passing Messages Between Data Points. *Science*, 5814(315), 972-976.
- Hastie, T.; Tibshirani, R.; y Friedman, J. (2008). Unsupervised learning. En *The elements of statistical learning*, 2.a ed. (pp. 485-585). Nueva York: Springer.
- Kaufman, L.; y Rousseeuw, P. (1987). Clustering by means of medoids. En Dodge, Y. (ed.), *Statistical Data Analysis Based on the L1 Norm and Related Methods* (pp. 405-416). Amsterdam: North-Holland.
- Kluger, Y.; Barsi, R.; Cheng, JT.; y Gerstein, M. (2003). Spectral biclustering of microarray data: coclustering genes and conditions. *Genome Research*, 13(4), 703-16.
- Macnaughton-Smith, P.; Williams, W. T.; Dale, M. B.; y Mockett, L. G. (1964). Dissimilarity analysis: a new technique of hierarchical sub-division. *Nature*, 202, 1034-1035.

**Autor**

Jerónimo Hernández-González

**Profesor**

Félix José Fuentes Hurtado