

# Análisis de conglomerados (Clustering)

Curso de aprendizaje automático para  
el INE

Víctor Gallego y Roi Naveiro

2020-02-02

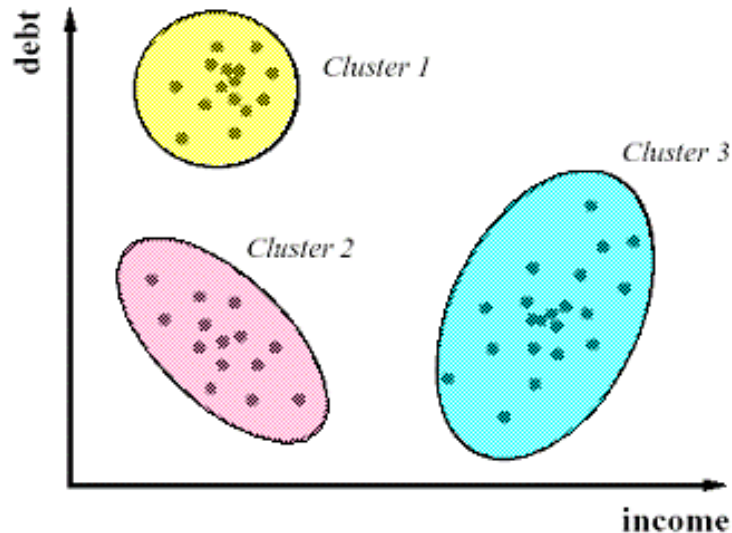
# Introducción

# Repaso

- **Aprendizaje no supervisado:** solo tenemos datos  $\mathcal{X} = \{x_1, \dots, x_N\}$  sin etiquetar.
- Hasta ahora hemos visto PCA y NMF, que tratan de aprender una representación de los datos en un espacio de menor dimensión.
- Hoy nos centraremos en las técnicas de **clustering** (también conocidas como **segmentación de datos**).

# Introducción

- Objetivo: agrupar un conjunto de ejemplos en grupos denominados *clusters*. Ejemplos del mismo clúster más parecidos entre sí que ejemplos de diferentes clústers.
- Otro objetivo: ordenar clusters en jerarquías naturales. En cada nivel de la jerarquía, objetos en el mismo grupo más similares que objetos en grupos distintos.
- Estadístico descriptivo: ¿consisten los datos en un conjunto de subgrupos cada uno con propiedades diferentes?
- Ejemplo típico en banca:



# Matrices de proximidad

La mayoría de algoritmos para clustering requieren la siguiente representación para los datos  $x_1, \dots, x_N$ :

$$\mathbf{D} = \begin{pmatrix} d_{11} & \dots & d_{1N} \\ \dots & \dots & \dots \\ d_{N1} & \dots & d_{NN} \end{pmatrix}$$

- donde  $d_{ii'}$  mide la **disimilaridad** entre  $x_i$  y  $x_{i'}$ .
- $\mathbf{D}$  es una matriz de tamaño  $N \times N$  (lo cual impone un requisito en la complejidad en memoria de los algoritmos de clustering).
- La mayoría de algoritmos asume una matriz simétrica (si no lo es, usar  $(\mathbf{D} + \mathbf{D}^\top)/2$ ).
- **Fundamental:** escoger medida de distancia (disimilaridad).
- Las disimilaridades no son distancias en el sentido estricto, pues no verifican la desigualdad triangular  $d_{ii'} \leq d_{ik} + d_{ki'}$ .
- Esto depende del problema concreto (similar a elección de coste en aprendizaje supervisado).

# Medidas de disimilaridad entre atributos

- Datos:  $x_{ij}$  con  $i = 1, 2, \dots, N$  y  $j = 1, 2, \dots, p$ .  $N$  ejemplos con  $p$  atributos.
- Construir medidas de disimilaridad por pares.
- $d_j(x_{ij}, x_{i'j})$ : disimilaridad entre valores de atributo  $j$ .

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

- No hay por qué pesar todos los atributos igual...

# Medidas de disimilaridad entre atributos

- En función del tipo de atributo.
- Variables cuantitativas:

$$d(x_{ij}, x_{i'j}) = l(|x_{ij} - x_{i'j}|)$$

con  $l$  función monótona creciente. Error cuadrático o absoluto. También es posible usar correlaciones (medida de similaridad):

$$\rho(x_i, x_{i'}) = \frac{\sum_j (x_{ij} - \bar{x}_i)(x_{i'j} - \bar{x}_{i'})}{\sqrt{\sum_j (x_{ij} - \bar{x}_i)^2 \sum_j (x_{i'j} - \bar{x}_{i'})^2}}$$

- Variables categóricas: Si tienen  $M$  categorías: Disimilaridad 0 si categorías iguales, 1 si diferentes. Valores distintos de 1 pueden utilizarse para enfatizar unos errores más que otros.

# Medidas de disimilaridad entre objetos

- Procedimiento para combinar  $p$  medidas de disimilaridad entre atributos en una medida de disimilaridad entre ejemplos  $D(x_i, x_{i'})$ .
- Combinación convexa

$$D(x_i, x_{i'}) = \sum_{j=1}^p \omega_j \cdot d_j(x_{ij}, x_{i'j}); \quad \sum_{j=1}^p \omega_j = 1$$

- Elección de pesos, depende del problema en cuestión.
- Importante: estudiar cada problema en concreto para construir tanto las medidas de disimilaridad entre atributos como los pesos.
- Aunque no se diga: esta es la parte más importante (y más costosa).



# Algoritmos de clustering

# Tipos de algoritmos de clustering

- **Algoritmos combinatorios:** trabajan con los datos observados sin hacer referencia al modelo probabilístico subyacente.
- **Modelos de mixturas:** asumen que datos son muestras iid de una mixtura de densidades de probabilidad. Cada componente describe un cluster.

# Algoritmos de clustering combinatorios: K-Means

# Algoritmos combinatorios

- Asignan cada observación a un cluster, sin preocuparse de la distribución de probabilidad de subyacente.
- Etiquetamos cada observación con  $i \in \{1, 2, \dots, N\}$
- Postulamos un número fijo de clusters  $K < N$ , etiquetados con  $k \in \{1, \dots, K\}$ .
- Objetivo construir función de asignación  $k = C(i)$  para cada  $i$ .
- Buscar asignación que minimice **dispersión dentro del cluster**

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

# Algoritmos combinatorios

- La dispersión total es

$$\begin{aligned} T &= \frac{1}{2} \sum_{i=1}^N \sum_{i'=1}^N d(x_i, x_{i'}) = \frac{1}{2} \sum_{k=1}^K \sum_{i:C(i)=k} \left[ \sum_{i':C(i')=k} d(x_i, x_{i'}) + \sum_{i':C(i') \neq k} d(x_i, x_{i'}) \right] \\ &= W(C) + B(C) \end{aligned}$$

- $B(C)$  es la **dispersión entre clusters**.
- Minimizar  $W(C)$  es equivalente a maximizar  $B(C)$ .
- Optimización por enumeración complete **inviable**.
- $N = 19$ ,  $K = 4$  número de asignaciones del orden de  $10^{10}$ .

# Algoritmos combinatorios

- Estrategia: *iterative greedy descent*
  1. Asignación inicial.
  2. Iterativamente, cambiar asignación tal que se disminuya  $W(C)$ .
  3. Parar cuando se deje de mejorar.
- Distintos algoritmos de clustering en función del paso 2.
- Asegurada convergencia a óptimo local...

# K-Means

- Todas las variables son cuantitativas.
- Disimilaridad = distancia Euclídea

$$d(x_i, x_{i'}) = \|x_i - x_{i'}\|^2$$

- La distancia Euclídea con pesos puede usarse redefiniendo  $x_{ij}$ .
- En este caso,

$$W(C) = \sum_{k=1}^K N_k \sum_{i:C(i)=k} \|x_i - \mu_k\|^2$$

- Como  $\mu_k = \arg \min_m \sum_{i:C(i)=k} \|x_i - m\|^2$ , vemos que la asignación óptima puede obtenerse resolviendo

$$\min_{C, \{m_k\}_1^K} \sum_{k=1}^K N_k \sum_{i:C(i)=k} \|x_i - m_k\|^2$$

# K-Means

- K-means resuelve utilizando *descenso coordinado*:
  1. Fijar asignación  $C$  y minimizar respecto  $\{m_1, \dots, m_K\}$ , asignando las medias de cada cluster.
  2. Fijar las medias  $\{m_1, \dots, m_K\}$  y minimizar asignando cada observación al cluster de la media más cercana.

$$C(i) = \arg \min_k \|x_i - m_k\|^2$$

1. Iterar 1 y 2 hasta que no cambien las asignaciones.



# Cuantización vectorial

- Técnica de teoría de la señal para aproximar datos de alta dimensión.
- Utilizada para compresión de datos.
- Idea: dividir conjunto grande de puntos en grupos y aproximar cada grupo por su centroide.
- En compresión de imágenes:
  1. Dividir imagen de  $N \times N$  pixels en grupos de  $k \times k$  pixels ( $k < N$ ).
  2. Considerar cada grupo como un vector de dimensión  $k \times k$ .
  3. Aplicar K-means y aproximar cada grupo por su centroide más próximo. (Conjunto de centroides se conoce como codebook).

# Algoritmos de clustering: K-Medoids

# K-Medoids

- K-means es apropiado cuando la disimilaridad es la euclídea cuadrática (datos continuos).
- Además, por estar elevada al cuadrado, pone mucha influencia a distancias grandes, lo que provoca **sensibilidad a outliers**.
- Podemos evitar esto sacrificando eficiencia computacional a cambio de robustez.
- Basta cambiar el paso de minimización en K-means: en lugar de tomar como representante de un cluster las medias de sus observaciones, hacemos que sea alguna observación de ese clúster.

# K-Medoids

- Para una asignación de clúster  $C$ , encontrar la observación que minimice:

$$i_k^* = \arg \min_{i:C(i)=k} \sum_{C(i')=k} D(x_i, x_{i'})$$

- Entonces  $m_k = x_{i_k^*}$ .
- Dado un conjunto de centros de clusters  $\{m_1, \dots, m_K\}$ , calcular

$$C(i) = \arg \min_{1 \leq k \leq K} D(x_i, m_k)$$

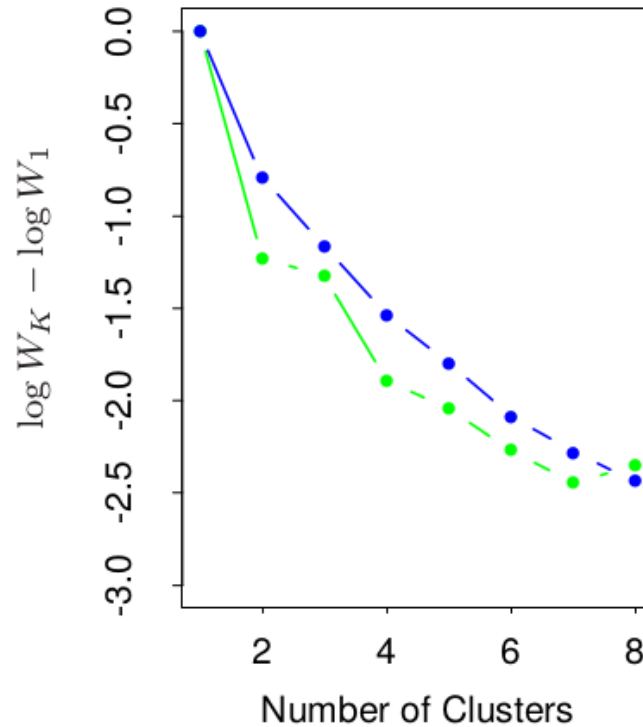
- El paso 1 pasa de tener complejidad lineal a ser  $\mathcal{O}(N_k^2)$

# Cuestiones prácticas

- Para aplicar K-Means o K-Medoids es necesario seleccionar el número de clústers  $K^*$  y una inicialización.
- Para las inicializaciones, la heurística más común es escogerlos **uniformemente aleatorios**, y ejecutar el algoritmo varias veces.
- En cuanto al número de clústers:
  - En aplicaciones de segmentación,  $K$  suele venir especificado (ejemplo, segmentar una cartera de clientes teniendo  $K$  encargados de marketing).
  - En descubrimiento de datos, no es tan fácil.
  - Una heurística es ir probando con  $K$  desde  $1, 2, \dots, K_{max}$ , e ir calculando las disimilaridades intra-clúster  $W_1, W_2, \dots, W_{K_{max}}$ .
  - Habrá un fuerte decrecimiento en sucesivas diferencias  $W_K - W_{K+1}$  en  $K = K^*$ . Esto es,  $\{W_K - W_{K+1} | K < K^*\} \gg \{W_K - W_{K+1} | K \geq K^*\}$ . Por tanto, podemos identificar  $K^*$  buscando un "codo" en la gráfica de  $W_K$  como función de  $K$ .

# Cuestiones prácticas

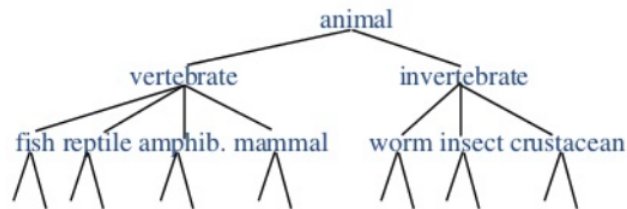
- En este ejemplo el "codo" se ubica en  $K = 2$ .



# Algoritmos de clustering: Clustering jerárquico

# Clustering jerárquico

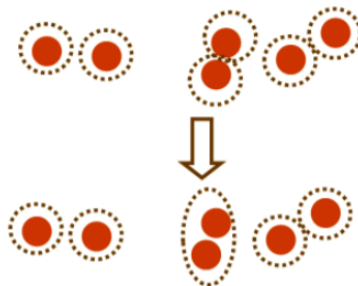
- **Problema:** ¿cómo elegimos el número de clústers  $K$  de antemano?
- Los algoritmos de clustering jerárquico evitan este problema: en lugar de tener que especificarlo:
- construyen toda una jerarquía sobre las observaciones.
- Ejemplos típicos de Biología y Genética:





# Tipos de clustering jerárquico

- Clustering **divisivo** (de arriba a abajo)
  - Empieza con todos los puntos en un único clúster (la raíz), luego:
  - Parte la raíz en un conjunto de clústers hijos, y sigue dividiendo recursivamente
  - Para cuando hay un único clúster por observación.
- Clustering **aglomerativo** (de abajo a arriba)
  - Cada observación es un clúster.
  - Se van fusionando clústers más similares en cada iteración.
  - Más estudiado que los divisivos.



# Clustering aglomerativo

El algoritmo es el siguiente:

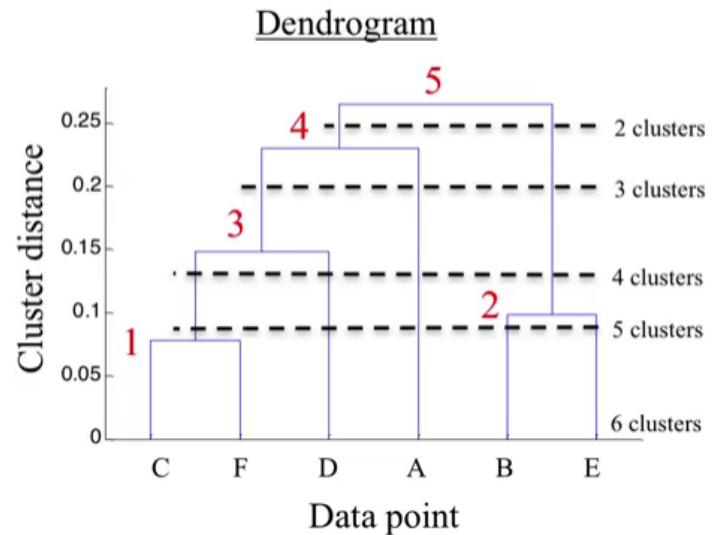
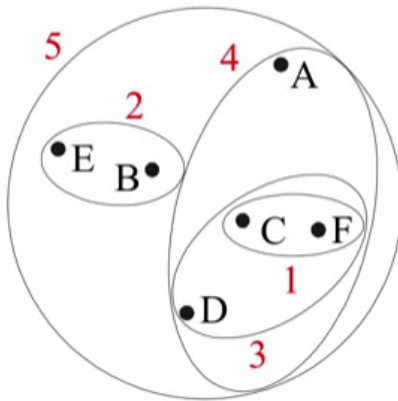
1. Empezar con  $n$  observaciones y una medida de todas las disimilaridades de los  $\binom{n}{2}$  posibles emparejamientos. Tratar cada observación como un único clúster.
2. Para  $i = n, n - 1, \dots, 2$ :
  - Examinar todas las disimilaridades inter-cluster e identificar el par de clústers más similar, fusionándolos. La disimilaridad entre estos dos clústers indica la altura del dendrograma a la que irá anotada la fusión.
  - Calcular las nuevas disimilaridades inter-clusters entre los  $i - 1$  grupos restantes.
- **Importante:** tenemos que generalizar nuestra noción de distancia entre observaciones a una que mida distancia entre clústers de observaciones.

# Dendrogramas

- Los algoritmos aglomerativos pueden ser representados como un árbol binario:
  - Los nodos representan los diversos clústers.
  - El nodo raíz es todo el dataset.
  - Los  $N$  nodos terminales son cada una de las  $N$  observaciones.
  - Dado un nodo, sus dos hijos representan qué clústers han sido fusionados.
- La mayoría de algoritmos aglomerativos poseen la **propiedad de monotonía**: la disimilaridad entre clústers que se fusionan es monótona creciente a medida que aumentamos en la jerarquía.
- Representación gráfica mediante un **dendrograma**:
  - Los nodos terminales (observaciones) los ponemos a altura 0.
  - La altura del resto de nodos es proporcional al valor de la disimilaridad entre sus dos hijos.

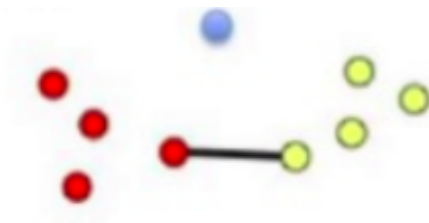
# Clustering aglomerativo

- Ejemplo de un posible resultado:

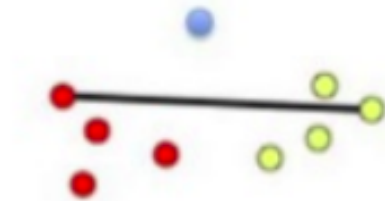


# Funciones de link

- ¿Cómo definimos una noción de distancia entre clusters? Hay varias opciones
- **Single link:**  $d_{SL}(G, H) = \min_{i \in G, j \in H} d_{ij}$ .



- **Complete link:**  $d_{CL}(G, H) = \max_{i \in G, j \in H} d_{ij}$ .

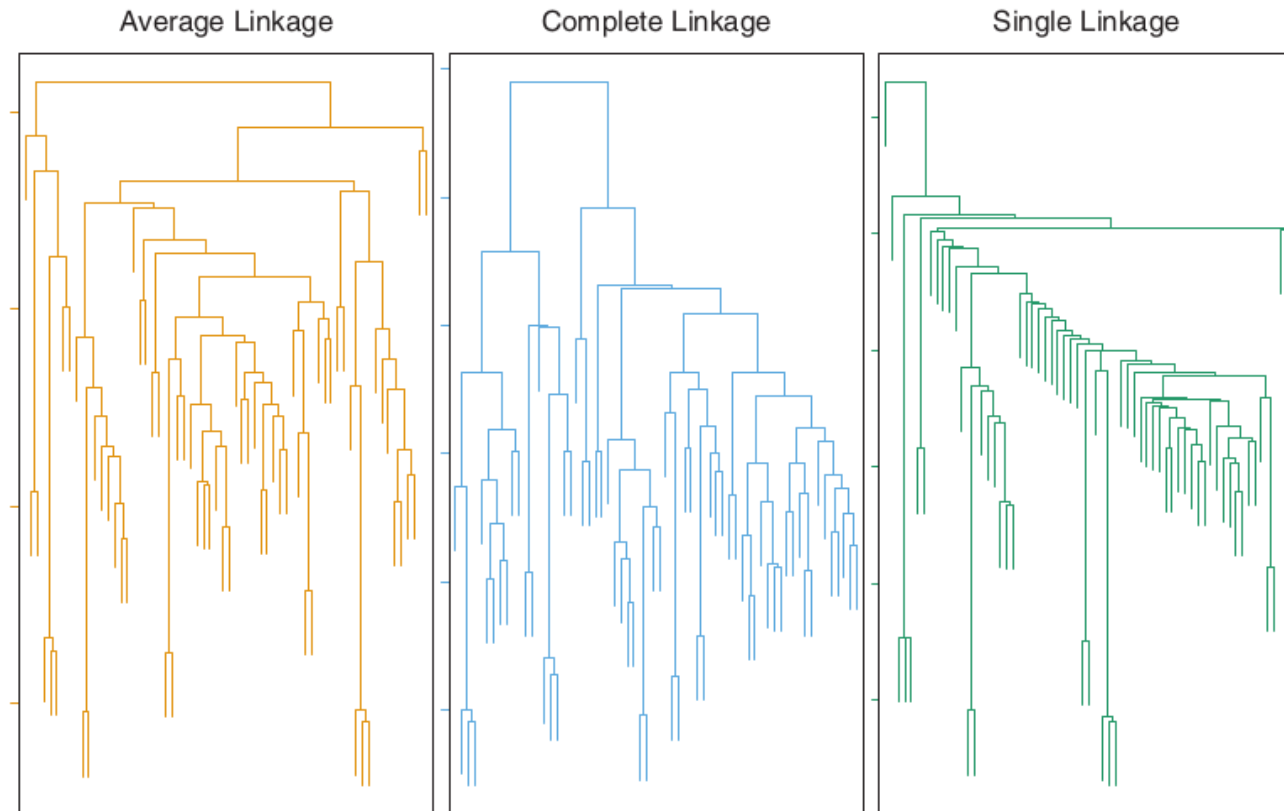


- **Promedio** (group average):  $d_{GA}(G, H) = \frac{1}{N_G N_H} \sum_{i \in G} \sum_{j \in H} d_{ij}$ .
- También hay otras, como las de la familia Ward que minimizan la varianza intracluster total.

# Funciones de link

- ¿Cómo escogemos la función de link?
- Solo hay algunas heurísticas:
  - Single link: puede generar clústers alargados (chaining: a cada clúste solo se le añade una observación cada vez).
  - Complete link: tiende a producir clústers mucho más compactos, aunque puede violar la propiedad de cercanía.
  - Link Promedio: supone un compromiso entre los dos, aunque es sensible a la escala de las distancias  $d_{ij}$ .

# Funciones de link



# Correlación cofenética

- El *cophenetic correlation coefficient* se define como la correlación entre las disimilaridades entre observaciones  $d_{ii'}$  y las disimilaridades cofenéticas  $C_{ii'}$ .
- $C_{ii'}$  se define como la disimilaridad entre los grupos cuando las observaciones  $i$  e  $i'$  se fusionan en el dendrograma.
- Esto es,  $C_{ii'}$  puede verse como la altura a la que  $i$  e  $i'$  se fusionaron.
- También podemos calcular la correlación cofenética entre dos dendrogramas (generados mediante diferentes funciones de link), para obtener una medida de **cómo de distintos son dos dendrogramas**.



# Resumen de clustering jerárquico

- Para un dataset consistente en  $n$  puntos:
  - Requiere  $\mathcal{O}(n^2)$  memoria (por la matriz de distancias).
  - Requiere  $\mathcal{O}(n^3)$  de tiempo en la mayoría de casos (aglomerativo).
- Ventajas:
  - Muy útiles para visualización (dendrograma).
  - Cuando los datos tienen estructura jerárquica, más útil que otros tipos de clustering.
- Inconvenientes:
  - Elevado coste computacional (ver arriba).
  - Sensibles a tipo de algoritmo/distancia escogidas.
- En R:
  - hclust de la librería estándar, bastante sólida.
  - dendextend para trabajar con dendrogramas (ver práctica).

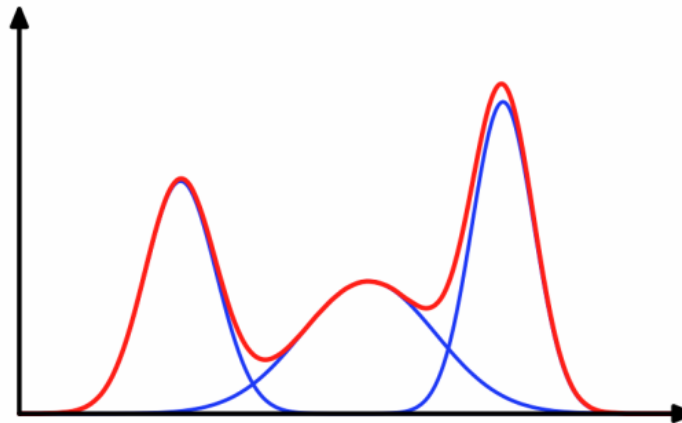
# Clustering Probabilístico

# Mixturas de Gaussianas (MoG)

- Consisten en una combinación convexa de  $K$  distribuciones normales,

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

- donde  $\sum_{k=1}^K \pi_k = 1$  y  $\pi_k \geq 0$ , consiguiendo modelizar distribuciones *multimodales*.



# Modelos probabilísticos

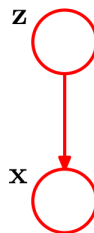
- Podemos refinar nuestras inferencias si distinguimos dos tipos de variables (aleatorias):
  1. Variables **observadas**:  $x$ , los datos del problema.
  2. Variables **latentes**:  $z$ , no son observables, queremos inferirlas a partir de las observaciones.
- Distinguir entre los dos tipos de variables anteriores es la base de los **modelos probabilísticos**.
- Estos modelos definen una distribución conjunta  $p(x, z)$ .
- Estamos interesados en hacer inferencias de la forma  $p(z|x)$  (la distribución a posteriori), para ello recurrimos al Teorema de Bayes:

$$p(z|x) = \frac{p(x, z)}{p(x)} = \frac{p(x, z)}{\int p(z, x) dz}$$

- $p(x)$  suele ser intratable de calcular: surgen numerosas técnicas de aproximación Bayesiana (Markov Chain Monte Carlo, Expectation-Maximization, Variational Inference, ...)

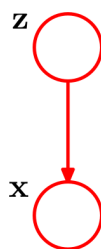
# Mixturas de Gaussianas (MoG)

- En el caso de la mixtura de Gaussianas, ¿qué representan las variables latentes  $z$ ?
- Intuición con clustering:  $z$  será una **variable categórica (factor)** que indica a que componente (cluster) pertenece la observación  $x$ .
- Podemos simular de una mixtura de Gaussianas mediante el siguiente proceso:
  1.  $z \sim \text{Categorical}\{1, \dots, K\}$  (seleccionamos cluster)
  2. Sea  $z = k$ ,  $x \sim \mathcal{N}(\mu_k, \Sigma_k)$  (simulamos de ese cluster)
- Gráficamente, lo representamos mediante



donde los nodos son variables aleatorias, y los arcos indican dependencia probabilista.

# Mixturas de Gaussianas (MoG)



- Nos indica que la probabilidad conjunta  $p(x, z)$  se ha factorizado de la siguiente manera al definir el modelo

$$p(x, z) = p(z)p(x|z)$$

- ¿Quiénes son los factores?
  - $p(z)$ : lo definimos tal que  $p(z_k = 1) = \pi_k$ , donde  $z$  es una variable categórica representada mediante OHE.
  - $p(x|z_k = 1) = \mathcal{N}(x|\mu_k, \Sigma_k)$
- De esta forma, tenemos que la marginal es (**coincide con lo original**)

# Mixturas de Gaussianas (MoG)

- Si hemos llegado a lo mismo que al principio, **¿para qué hemos hecho todo esto?**
- Al introducir las variables latentes  $z$ , podemos responder de forma natural a inferencias del tipo: ¿a qué cluster asignamos  $x$ ? mediante el posterior  $p(z|x)$ .
- En el ámbito de MoGs, al posterior también se le denomina **responsabilidad**, y su expresión es

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

- En lugar de obtener una asignación **rígida** a un cluster, tenemos una noción de incertidumbre via  $p(z|x)$ .
- Además, al introducir las variables latentes  $z$ , se pueden desarrollar algoritmos de inferencia más sofisticados (como EM).

# Máxima Verosimilitud (ML)

- Todavía no hemos visto cómo obtener valores buenos para los parámetros de las dos distribuciones involucradas:

$$\pi_k, \mu_k, \Sigma_k$$

- El algoritmo más básico (no necesita latentes  $z$ ) es el de **máxima verosimilitud**.
- Supongamos que tenemos  $x_1, \dots, x_N$  observaciones (independientes), escribimos la log-verosimilitud del modelo

$$\log p(x|\pi, \mu, \Sigma) = \sum_{n=1}^N \log \left\{ \sum_{k=1}^K \pi_k \mathcal{N}(x_n | \mu_k, \Sigma_k) \right\}$$

- Hemos obtenido un problema de optimización

$$\pi^*, \mu^*, \Sigma^* = \arg \max_{\pi, \mu, \Sigma} \log p(x|\pi, \mu, \Sigma)$$



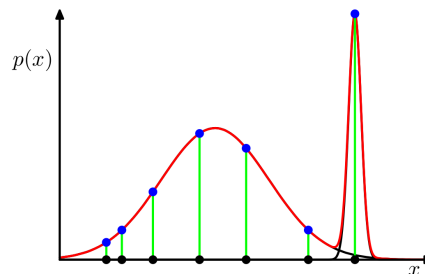
# Cuestiones técnicas sobre ML

- Desafortunadamente, no hay una solución explícita para el óptimo, pero podemos utilizar **descenso por el gradiente** o similares.
- El problema de optimización presenta **singularidades**. Si "tenemos suerte", puede ocurrir que durante la optimización  $x_n = \mu_j$ , teniendo que

$$\mathcal{N}(x_n | x_n, \sigma_j^2 I) = \frac{1}{\sqrt{2\pi}} \frac{1}{\sigma_j} \rightarrow \infty$$

cuando  $\sigma_j \rightarrow 0$ .

- Es decir, ajustando una componente a un único punto  $x_n$ , podemos maximizar la función objetivo tanto como queramos. **El overfitting es real.**



# Algoritmo EM (1)

- Técnica general para encontrar soluciones de máxima verosimilitud en modelos probabilísticos con variables latentes.
- Sea un modelo probabilístico en el que denotamos por  $X$  el vector de variables observadas y  $Z$  el vector de variables latentes (discretas).
- La distribución conjunta,  $p(X, Z|\theta)$  viene gobernada por el vector de parámetros  $\theta$ . Queremos maximizar

$$p(X|\theta) = \sum_Z p(X, Z|\theta)$$

- Asumimos que la optimización directa de  $p(X|\theta)$  es difícil, pero la de  $p(X, Z|\theta)$  es fácil.

# Algoritmo EM (2)

- Para cualquier distribución  $q(Z)$  sobre las variables latentes se verifica

$$\log[p(X|\theta)] = \mathcal{L}(q, \theta) + KL(q||p)$$

- Donde

$$\mathcal{L}(q, \theta) = \sum_Z q(Z) \log \left[ \frac{p(X, Z|\theta)}{q(z)} \right]$$
$$KL(q||p) = - \sum_Z q(Z) \log \left[ \frac{p(Z|X, \theta)}{q(z)} \right]$$

- **Ejercicio:** demostrarlo.
- La divergencia de Kullback-Leibler verifica  $KL(q||p) \geq 0$  y  $KL(q||p) = 0$  si y solo si  $q(Z) = p(Z|X, \theta)$ .
- Por tanto,  $\mathcal{L}(q, \theta) \leq \log[p(X|\theta)]$ .

# Algoritmo EM (3)

- El algoritmo EM es un algoritmo de optimización iterativo con dos etapas utilizado para encontrar  $\theta$  que maximiza la verosimilitud.
- Supongamos que el valor actual de  $\theta$  es  $\theta^{old}$
- Paso E: Maximizar  $\mathcal{L}(q, \theta^{old})$  con respecto a  $q(Z)$ . (Demostrar que el máximo se alcanza cuando  $q(Z) = p(Z|X, \theta^{old})$ ).
- Paso M: Mantener fija  $q(Z)$  y maximizar  $\mathcal{L}(q, \theta)$  con respecto a  $\theta$ , dando lugar a  $\theta^{new}$ .
- El paso M hace que  $\mathcal{L}(q, \theta)$  crezca y por tanto también la log verosimilitud.
- Como  $q$  está determinada usando los parámetros viejos, ya no es idéntica a  $p(Z|X, \theta^{new})$  y por tanto la KL será distinta de 0.
- El aumento en log verosimilitud es mayor que el aumento en su cota inferior.

# Algoritmo EM (4)

- Después del paso E tenemos que

$$\mathcal{L}(q, \theta) = \sum_Z p(Z|X, \theta^{old}) \log p(X, Z|\theta) - \sum_Z p(Z|X, \theta^{old}) \log p(X, Z|\theta^{old})$$

- En el paso M maximizamos el valor esperado de  $\log p(X, Z|\theta)$ .
- $\theta$  aparece únicamente en el logaritmo. Si la distribución conjunta es de la familia exponencial, la optimización es muy sencilla.

# MoG mediante EM (1)

- En MoG queremos maximizar

$$\log p(X|\pi, \mu, \Sigma) = \sum_{n=1}^N \log \left\{ \sum_{k=1}^K \pi_k \mathcal{N}(x_n | \mu_k, \Sigma_k) \right\}$$

- Difícil pues la suma ocurre dentro del logaritmo.
- En cambio, la **verosimilitud completa**, más fácil pues cambia orden de logaritmo y suma

$$\log p(X, Z|\pi, \mu, \Sigma) = \sum_{n=1}^N \sum_{k=1}^K z_{nk} \{ \log \pi_k + \log \mathcal{N}(x_n | \mu_k, \Sigma_k) \}$$

# MoG mediante EM (2)

- Usamos EM.
- En el paso  $E$  debemos evaluar  $p(Z|X, \mu^{old}, \Sigma^{old}, \pi^{old})$

$$p(z_k = 1|x_n) = \frac{\pi_k^{old} \mathcal{N}(x_n|\mu_k^{old}, \Sigma_k^{old})}{\sum_{k=1}^K \pi_k^{old} \mathcal{N}(x_n|\mu_k^{old}, \Sigma_k^{old})} = \gamma(z_{nk})$$

- En el paso  $M$ , calculamos  $\mu^{new}, \Sigma^{new}, \pi^{new}$ , maximizando

$$\mathbb{E}[\log p(X, Z|\mu, \Sigma, \pi)] = \sum_{n=1}^N \sum_{k=1}^K \gamma(z_{nk}) \{ \log \pi_k + \log \mathcal{N}(x_n|\mu_k, \Sigma_k) \}$$

# MoG mediante EM (3)

- Para  $\mu_k$  (definiendo  $N_k = \sum_{n=1}^N \gamma(z_{nk})$ )

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) x_n$$

- Para  $\Sigma_k$

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) (x_n - \mu_k)(x_n - \mu_k)^\top$$

- Para  $\pi_k$

$$\pi_k = \frac{N_k}{N}$$



# Relación entre K-medias y MoG

- Considérese una MoG con  $\Sigma_k = \epsilon I$ .

$$p(x|\mu_k, \Sigma_k) = \frac{1}{(2\pi\epsilon)^{1/2}} \exp\left\{-\frac{1}{2\epsilon} \|x - \mu_k\|^2\right\}$$

- Usemos EM en este caso. La distribución a posteriori de  $z$  será

$$\gamma(z_{nk}) = \frac{\pi_k \exp\left\{-\frac{1}{2\epsilon} \|x - \mu_k\|^2\right\}}{\sum_j \pi_j \exp\left\{-\frac{1}{2\epsilon} \|x - \mu_j\|^2\right\}}$$

- En el límite  $\epsilon \rightarrow 0$ ,  $\gamma(z_{nk}) = 0$  para todo  $k$  excepto aquel para el cual  $\|x - \mu_k\|^2$  sea mínimo.
- Cada punto asignado al cluster con centroide más cercano!!

# Relación entre K-medias y MoG

- En este límite, el valor esperado de la log verosimilitud completa se puede escribir como

$$-\frac{1}{2} \sum_{n=1}^N \sum_{k=1}^K I(C(n) = k) \|x_n - \mu_k\|^2 + \text{cte}$$

- Maximizarlo (respecto a  $\mu_k$ ) equivale a escoger

$$\mu_k = \arg \min_m \sum_{n:C(n)=k} \|x_n - m\|^2$$