Aprendizaje no Supervisado
Clustering

Aprenentatge Automàtic

APA/GEI/FIB/UPC - 2025/2026 1Q

/ Javier Béjar

Introducción

- Hay un fuerte sesgo en la comunidad de aprendizaje automático hacia el aprendizaje supervisado, pero muchos conceptos se aprenden sin supervisión
- El objetivo es encontrar **estructura en los datos**
- Los métodos no supervisados son para fines exploratorios, por lo que no hay una respuesta correcta
- Los resultados deben analizarse, interpretarse y validarse a la luz de lo que se sabe
- El descubrimiento de nuevos conceptos siempre es sin supervisión

- Suponemos que los datos están un espacio N-dimensional con una función de similitud/distancia definida
- La similitud determina cómo se relacionan los ejemplos entre sí
- El agrupamiento tiene como objetivo encontrar patrones significativos en los datos según sus relaciones
- **Sesgo**:
  - Los ejemplos están más relacionados con los ejemplos más cercanos que con los más lejanos
  - Los patrones son grupos compactos que están separados al máximo entre sí

Algoritmos jerárquicos

- Los ejemplos se organizan como un árbol binario
- Basado en la relación entre ejemplos definidos por funciones de similitud/distancia
- No hay división explícita en grupos, tiene que elegirse a posteriori

Algoritmos particionales

- Solo se obtiene una partición del conjunto de datos
- Basado en la optimización de un criterio (supuestos sobre las características del modelo de clúster)

- Nos centraremos en los algoritmos de agrupamiento particional
- Estos algoritmos resuelven el problema de calcular la mejor asignación de N ejemplos en K grupos, este es un problema NP-Hard
- Hay muchos algoritmos de partición que usan diferentes criterios para encontrar los grupos
- Todos estos algoritmos aproximan la solución óptima con un tiempo computacional razonable
- Explicaremos dos algoritmos: K-means y modelos de mezcla gaussiana (GMM)

K-means

- K-means es un algoritmo de agrupamiento particional basado en el cálculo de un conjunto de prototipos $\mathcal{P} = \{\mu_1, \ldots \mu_k\}$ que representan el particionamiento de los datos (clusters)
- Los ejemplos se dividen asignándolos al prototipo más cercano
- Cada ejemplo solo pertenece a un clúster, esto se conoce como particionamiento duro (hard partitioning)
- El modelo que se ajusta corresponde a un conjunto de k grupos de forma hiperesférica, donde las fronteras de decisión corresponden a donde las esferas se encuentran

- Un algoritmo iterativo asigna cada ejemplo a uno de los grupos K (K es un hiperparámetro)
- **Criterio de optimización**: Minimizar la distancia de cada ejemplo al centroide del clúster (error cuadrático)

$$Distorsion = \sum_{k=1}^{K} \sum_{i \in C_k} \| x_i - \mu_k \|^2$$

- La optimización realiza una búsqueda local sobre la función de distorsión
- El algoritmo converge a un mínimo local, que depende de la inicialización (prototipos iniciales)

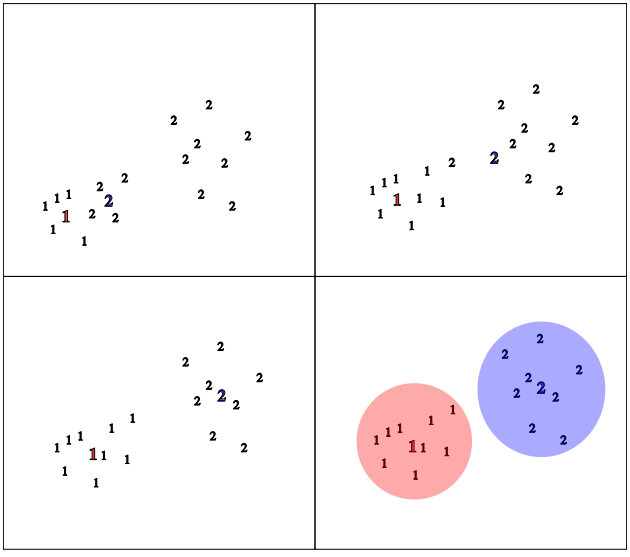
Algorithm: K-means (X: Ejemplos, k:entero)

Generar k prototipos iniciales (por ejemplo, k ejemplos aleatorios)

repetir

> Reasignar los ejemplos a su prototipo más cercano
>
> Recalcular prototipos (centroides)

hasta que *sin cambios en las asignaciones o un numero de iteraciones máximo*

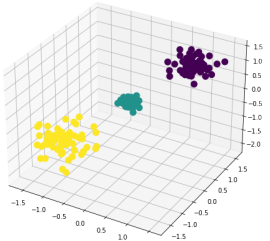


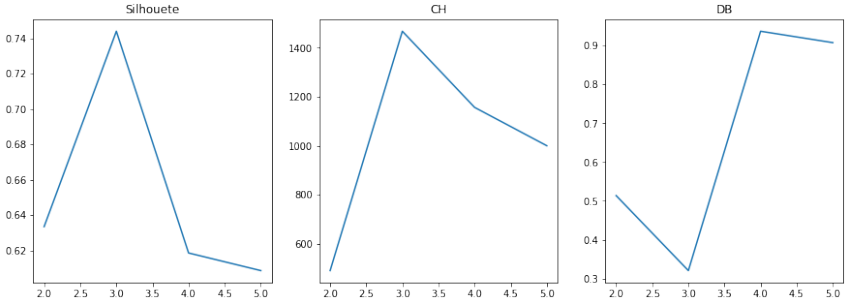
- Ventajas
  - Converge rápido, baja complejidad computacional $(O(kni))$
  - Fácil de implementar
- Inconvenientes
  - Los resultados dependen de la inicialización (ejecutar varias veces, conservar la mejor)
  - Es sensible a valores atípicos y grupos de diferentes tamaños y densidades
  - K debe conocerse a priori o deben calcularse particiones con diferentes valores de K y validarse utilizando medidas de calidad de agrupamiento
  - Usar una partición dura de los datos podría ser demasiado restrictivo para algunos problemas

- K-means++ modifica la estrategia de inicialización buscando maximizar la distancia entre los prototipos iniciales
- **Algoritmo:**
  1. Elegir un centro uniformemente de entre todos los datos
  2. Para cada dato x, calcular $d(x, c)$, la distancia entre x y el centro más cercano ya elegido
  3. Elegir un nuevo de dato al azar como nuevo centro, utilizando una distribución de probabilidad ponderada donde se elige x con probabilidad proporcional a $d(x, c)^2$
  4. Repetir los pasos 2 y 3 hasta que se hayan elegido k centros
  5. Continuar con el algoritmo estándar de K-means

- El número de clústeres es un parámetro que debe decidirse (selección del modelo)
- El criterio de optimización no es útil dado que decrece con el número de clusters
- Hay muchos criterios de calidad de clúster que se pueden usar para seleccionar el parámetro k
- Estos criterios miden un equilibrio entre la separación de los grupos y su compacidad y compensan por el diferente número de grupos
- No existe un criterio perfecto, los más utilizados son el criterio de Calinski-Harabasz, el criterio de Davis-Bouldin y el índice de Silhouette

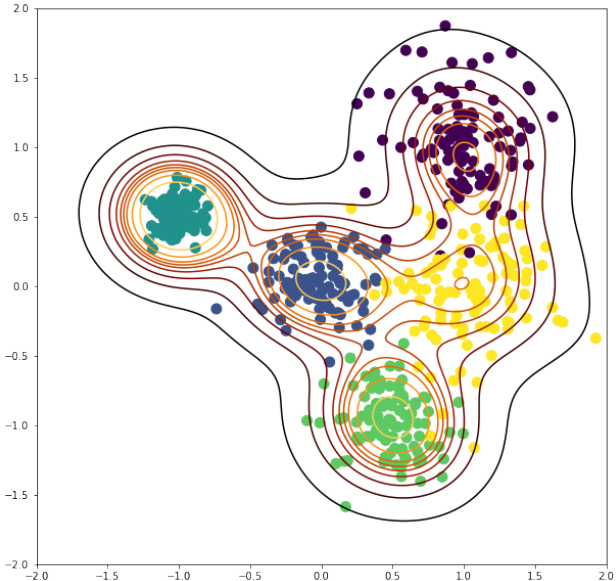
- Índice de Calinski-Harabasz: Relación entre la dispersión dentro del grupo (suma de las distancias de los ejemplos en un grupo) y entre las dispersiones de los grupos (suma de las distancias de los prototipos)
- Índice de Davies-Bouldin: Media de la relación máxima de dispersión de clústeres entre todos los pares de clústeres
- Índice de Silhouette: Máxima dispersión/varianza de grupo, calculada como la media de las proporciones de dispersión dentro del grupo y la suma de las distancias de los ejemplos de un grupo a los ejemplos del grupo más cercano
- Se elige el parámetro que optimiza la medida o el valor que obtiene la mayor variación




Silhouete
0.74
0.72
0.70
0.68
0.66
0.64
0.62
2.0
2.5
3.0
3.5
4.0
4.5
5.0
CH
1400
1200
1000
800
600
2.0
2.5
3.0
3.5
4.0
4.5
5.0
DB
0.9
0.8
0.7
0.6
0.5
0.4
0.3
2.0
2.5
3.0
3.5
4.0
4.5
5.0

Modelos de mezcla gaussiana

- Podemos usar un enfoque generativo para agrupar datos
- Asumimos que los datos son una mezcla de K distribuciones de probabilidad
- El objetivo es separar las distribuciones, asignando grupos de datos a la distribución que los genera
- Obteniendo los grupos tenemos una estimación de los parámetros de las distribuciones
- Esto nos da un modelo generativo que podemos muestrear como con los modelos generativos supervisados


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

- Asumimos que los datos se generan de una mezcla de distribuciones gaussianas
- El objetivo es buscar en el espacio de parámetros para obtener la mezcla que mejor explique los datos (estimación de parámetros)
- El modelo de la distribución de los datos es:

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

con K el número de clusters y $\sum_{k=1}^{K} \pi_k = 1$, π_k representa la contribución de la distribución k a la mezcla

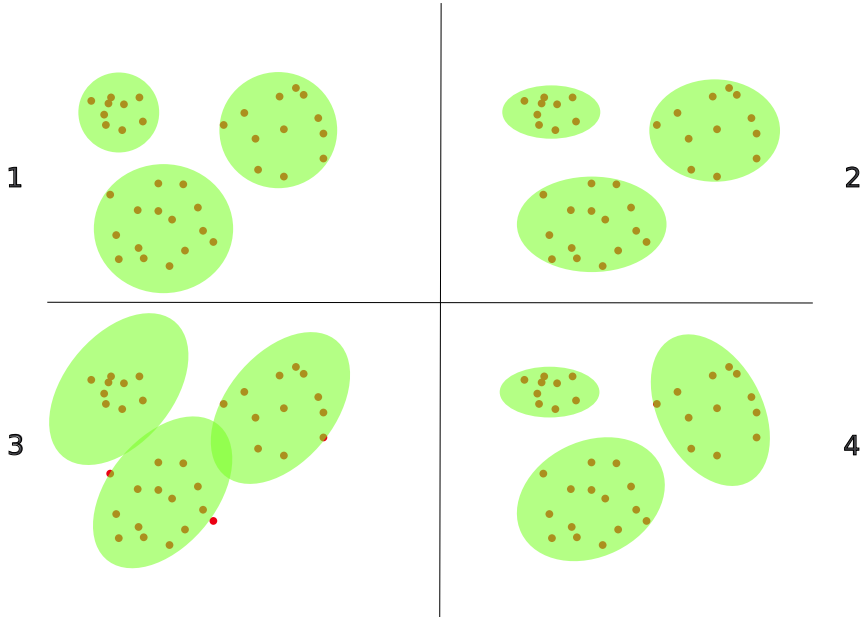
- Cada ejemplo tiene una probabilidad de pertenecer a un clúster, particiones blandas (soft)

- Como en los modelos supervisados, podemos estimar los parámetros de la mezcla de distribuciones utilizando máxima verosimilitud
- Para el caso gaussiano usamos el modelo:

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

  Siendo μ_k los vectores de medias y Σ_k las matrices de covarianza de cada distribución gaussiana
- Calcular el logaritmo de verosimilitud de esta distribución e igualar la derivada a cero nos dará los estimadores para μ_k y parámetros Σ_k
- La estimación dependerá de nuestra suposición sobre las gausianas

1. Atributos independientes, las matrices de covarianza son diagonales y diferentes, pero todos los atributos de un componente comparten la misma varianza ($O(k)$ parámetros, hiperesferas)
2. Atributos independientes, pero con varianzas diferentes, las matrices de covarianza son diagonales, ($O(kd)$ parámetros, elipsoides paralelos al eje de coordenadas)
3. Atributos dependientes, misma matriz de covarianza para todos los componentes ($O(d^2)$ parámetros, hiperelipsoides idénticos no paralelos al eje de coordenadas)
4. Atributos dependientes, matriz de covarianza completa para todos los componentes ($O(kd^2)$ parámetros, hiperelipsoides no paralelos al eje de coordenadas)


1
2
3
4

- No existe una expresión exacta para el cálculo de los parámetros, por lo que se debe utilizar un algoritmo iterativo, este se conoce como Expectativa-Maximización (EM)
- El objetivo es estimar los parámetros de la distribución que describe cada grupo (por ejemplo, μ y Σ)
- EM es un algoritmo general que se utiliza para la tarea de estimación de parámetros por Máxima Verosimilitud (no es específico de GMM)
- El algoritmo maximiza la probabilidad de la distribución con respecto a los datos

- Realiza iterativamente dos pasos:
  - **Expectativa:** Calculamos una función que asigna a todos los ejemplos un grado de pertenencia a las K distribuciones de probabilidad, calculamos la verosimilitud de cada ejemplo respecto a los parámetros actuales
  - **Maximización:** Reestimamos los parámetros de la distribuciones para maximizar las pertenencias, cambiamos los parámetros para maximizar la verosimilitud

- Para el caso de A atributos independientes se cumple que:

$$p(x|\mu_k, \Sigma_k) = \prod_{j=1}^{A} \mathcal{N}(x|\mu_{kj}, \sigma_{kj})$$

- El modelo a ajustar es

$$p(x|\mu, \sigma) = \sum_{k=1}^{K} \pi_k \prod_{j=1}^{A} \mathcal{N}(x|\mu_{kj}, \sigma_{kj})$$

  tendremos una matriz de covarianza diagonal

- Podemos derivar los estimadores de los parámetros utilizando Máxima Verosimilitud a partir de la log-verosimilitud de $P(x|\mu, \sigma)$ ($\hat{\mu}$, $\hat{\sigma}$ y $\hat{\pi}$)

- El paso de expectativa calcula los pesos para cada ejemplo y componente (responsabilidad)

$$\hat{\gamma}_{ik} = \pi_k \mathcal{N}(x_i|\mu_k, \sigma_k)$$

- Esto representa la probabilidad de que un ejemplo x_i sea generado por el componente $\mathcal{C}_k$

- El paso de maximización vuelve a calcular $\hat{\mu}$, $\hat{\sigma}$ y $\hat{\pi}$ para cada componente proporcionalmente a los pesos calculados en el paso de expectativa (sus responsabilidades)

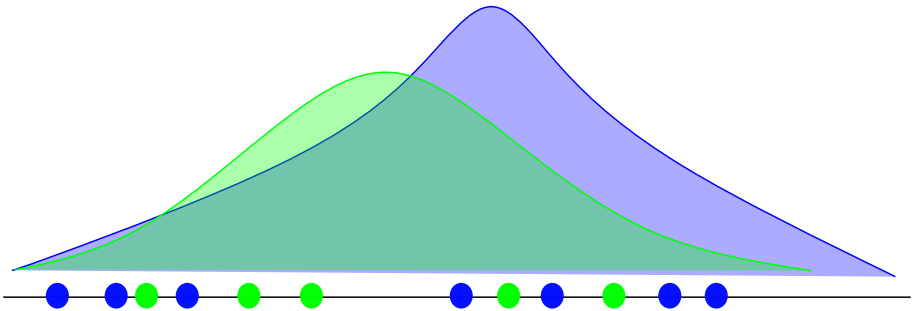
$$\hat{\mu}_k = \frac{\sum_{i=1}^{N} \hat{\gamma}_{ik} x_i}{\sum_{i=1}^{N} \hat{\gamma}_{ik}}$$

$$\hat{\sigma}_k = \frac{\sum_{i=1}^{N} \hat{\gamma}_{ik} (x_k - \hat{\mu}_i)^2}{\sum_{i=1}^{N} \hat{\gamma}_{ik}}$$

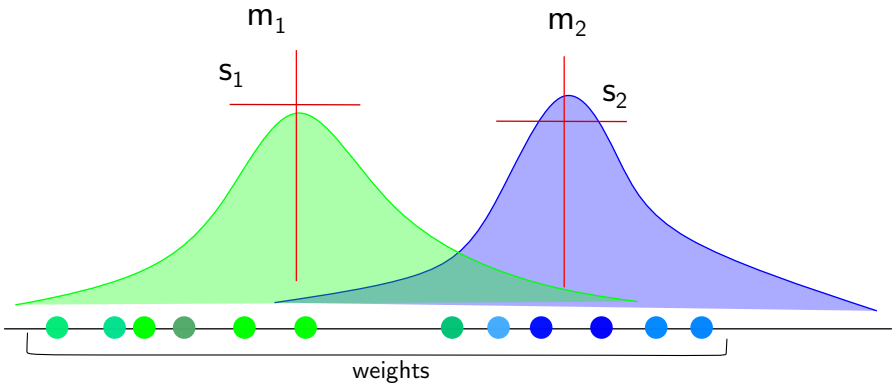
$$\hat{\pi}_k = \frac{1}{N} \sum_{k=1}^{N} \hat{\gamma}_{ik}$$

- Se generan K distribuciones iniciales $\mathcal{N}(\mu_k, \sigma_k)$, definiendo sus parámetros μ_k y σ_k, K-means puede usarse como estimación inicial
- Repetir hasta la convergencia (sin mejora de log-verosimilitud):
  1. **Expectativa:** Calcular la pertenencia de cada ejemplo a cada componente
     - Cada instancia tendrá un peso ($\hat{\gamma}_{ik}$, responsabilidad) calculado a partir de los componentes de la iteración anterior (probabilidad de que el ejemplo sea de un componente)
  2. **Maximización:** Recalcular los estimadores de los parámetros usando los pesos del paso anterior para obtener los nuevos $\hat{\mu}$, $\hat{\sigma}$ y $\hat{\pi}$ para cada distribución
- Al final, cada ejemplo tiene una probabilidad de pertenencia a cada componente que corresponde a la responsabilidad (asignación blanda)

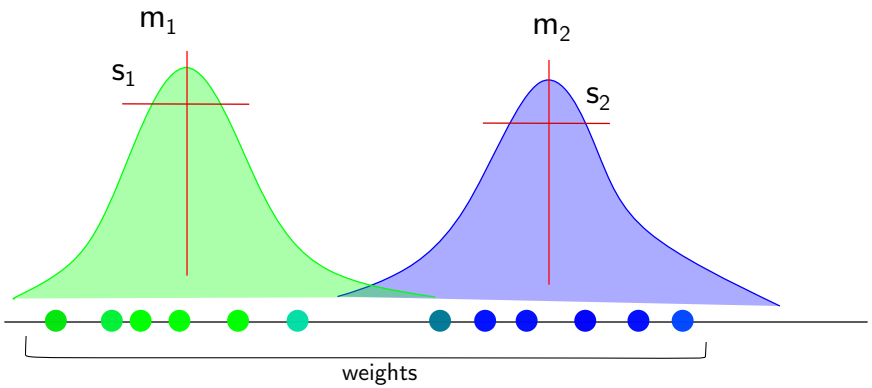
Asignación inicial



Expectativa + Maximización = nuevos parámetros



Expectativa + Maximización = nuevos parámetros



- K-means es un caso particular de este algoritmo (partición dura)
- Suponemos que todos los atributos son independientes, por lo que solo hay medias μ y las varianzas σ todas son iguales (y no nos importa su valor)
- Las responsabilidades de cada ejemplo tienen un valor de uno para un componente (la partición más probable) y cero para el resto
- Los pasos de K-means se corresponden con los pasos de EM:

  Expectativa: Asignamos los ejemplos a su clase más cercana (responsabilidad 1 para esa clase, 0 para las demas)

  Maximización: Reestimamos las medias acorde con la reasignación

Este notebook muestra la aplicación de K-means y GMM a un conjunto de datos sencillo