Aprendizaje Automático No Supervisado

Alberto Barbado González

Tema 3 – Diferentes implementaciones de K-Means

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

La Universidad en Internet

La pregunta del día

¿Cómo se comparan y qué diferencias hay en las diversas implementaciones de K-Means y cómo pueden influir en la calidad e interpretación de resultados?

Encuesta previa

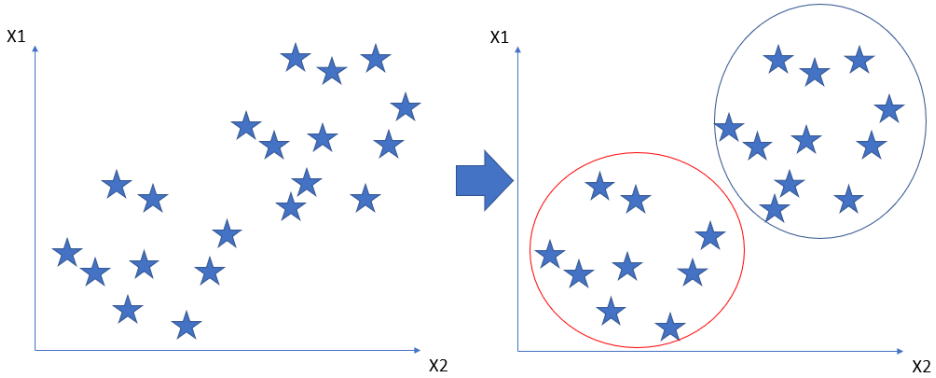
- ¿Recuerdas las limitaciones que tenía el algoritmo K-Means?

- ¿Qué posibles mejoras se pueden aplicar para mitigar algunos de estos problemas?

En el día de hoy

- Recordatorio de K-Means: algoritmo base
- Mejoras sobre K-Means en varios niveles:
  - Inicialización
  - Convergencia
  - Coste computacional
  - Pertenencia a varios clusters
  - Uso de variables categóricas

Clustering: Intuición inicial



Matriz de rasgos (*features*): MR

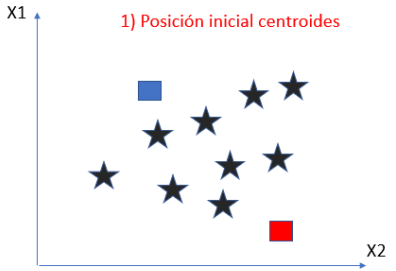
Cliente	Edad	Ingresos (USD)	Frecuencia de Compra (compras/mes)	Total Gastado (USD)
1	25	3000	2	500
2	40	7000	5	3000
3	35	4500	3	1200
4	28	3200	1	200
5	50	10000	7	8000

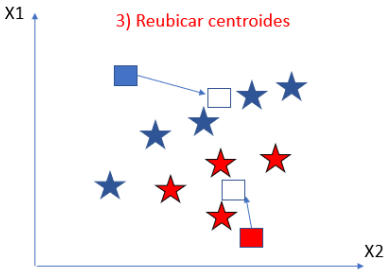
$MR_{n\,x\,p}$ *con* $\boldsymbol{n}$ *el nº de datos y* $\boldsymbol{p}$ *el nº de variables de entrada*

- Con ello, se busca encontrar vectores de datos (***data points***) que sean **similares entre sí.**
- Estos *data points* se **agruparían** dentro de un **mismo conjunto**.
- **Entrada**: Los datos de entrada son **matrices de *features***, similar a otros modelos de ML (e.j., clientes x atributos clientes).
- **Salida**: Cada fila (e.j., cliente) se vinculará con un cluster en función de sus columnas (rasgos/*features*)

Algoritmo K-Means

- **K-Means**: Agrupa datos en **K clústeres asignando** cada dato al **centroide más cercano**. **Recalcula** los centroides hasta que se **estabilizan**, buscando **minimizar la distancia** entre los datos y sus centroides para formar clústeres compactos.





$$I = \sum_{i=1}^{n} \| x_i - \mu_{c(i)} \|^2$$

- n es el número total de puntos.
- x_i es el i-ésimo punto de datos.
- $\mu_{c(i)}$ es el centroide del clúster al que pertenece el punto x_i .
- $\| x_i - \mu_{c(i)} \|^2$ es la distancia euclidiana cuadrada entre el punto x_i y su centroide correspondiente $\mu_{c(i)}$.

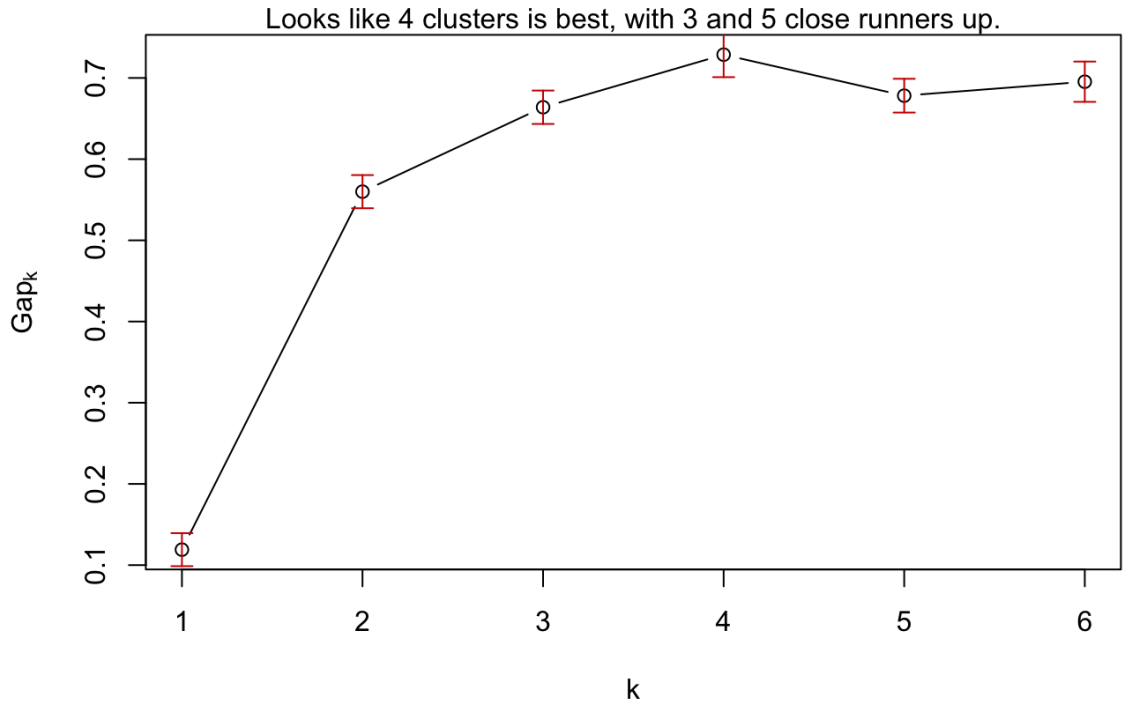
Otras métricas de calidad

- **Estadística de la brecha**: Compara la inercia obtenida al hacer clustering sobre los datos originales con la obtenida al hacer clustering sobre unos datos de referencia aleatorios.

  Para un determinado K
  1. Calcular la **inercia** para los datos reales.
  2. Generar **datos aleatorios** de **referencia**:
     - Genera un conjunto de datos de referencia (de la misma dimensión y rango que los datos originales) de manera aleatoria y uniforme.
     - Aquí no hay clusters reales.
  3. Calcular la **inercia para los datos de referencia**:
     - Para ese valor de K, se aplica K-Means en esos datos de referencia
  4. Calcular la **estadística de la brecha**:
     - Diferencia logarítmica entre la inercia media de los datos de referencia y la inercia de los datos reales
  5. Se **repite el proceso** para distintos valores de K. De esta manera, se selecciona como **K óptimo** el valor con mayor estadística de brecha.

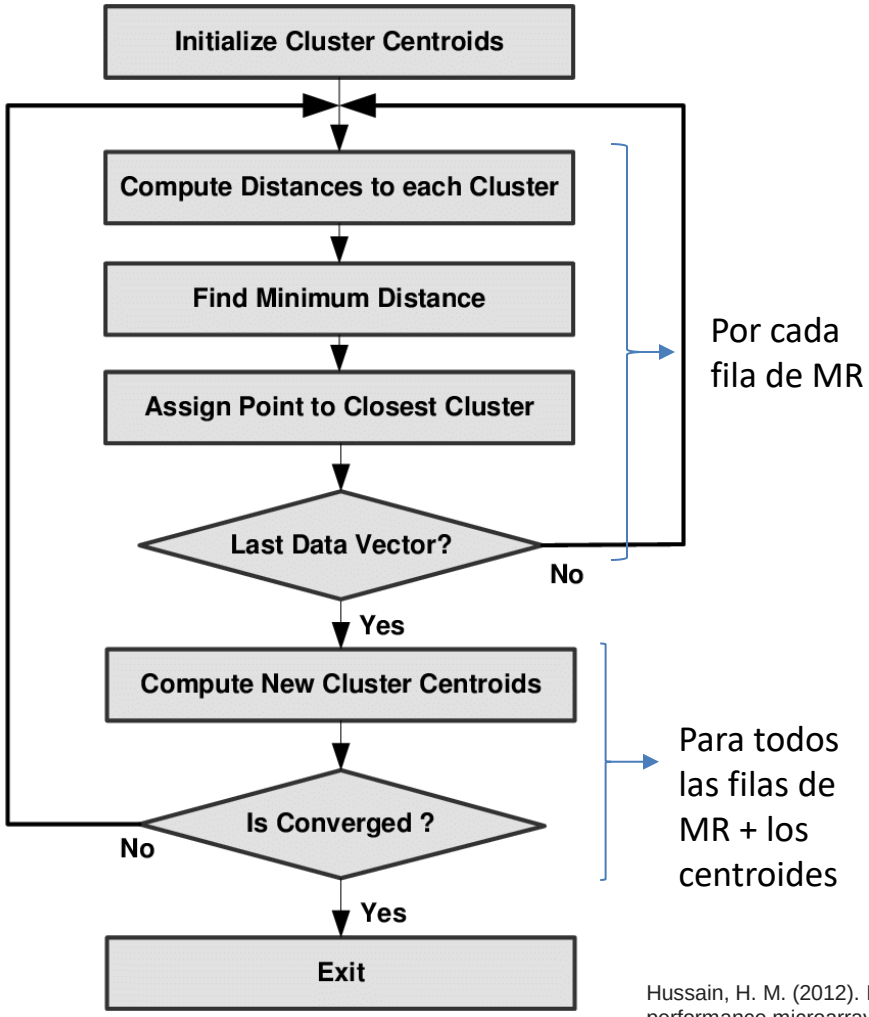
Otras métricas de calidad

- **Estadística de la brecha**: Compara la inercia obtenida al hacer clustering sobre los datos originales con la obtenida al hacer clustering sobre unos datos de referencia aleatorios.



https://joey711.github.io/phyloseq/gap-statistic.html

Lloyd's K-Means (a.k.a. "K-Means Clásico)"



Se eligen los **parámetros del modelo** (e.j., número de clusters, K)

1. **Inicialización:** Se eligen **K puntos aleatorios** como centroides (valores aleatorios, no pertenecientes a MR).
2. **Asignación:** Cada fila de MR se **asigna al centroide k más cercano** (menor distancia). Se tienen así K clusters.
3. **Reubicación:** Se **calcula el nuevo centroide** en función de los datos de cada cluster.
4. **Repetición:** Se analiza **convergencia**. Por ejemplo:
   - Si "cambian mucho" los centroides -> No converge -> Se vuelve al **punto 2**.
   - Si "no cambian mucho" los centroides, se termina.

Hussain, H. M. (2012). Dynamically and partially reconfigurable hardware architectures for high performance microarray bioinformatics data analysis.

Lloyd's K-Means

Algorithm 1 Lloyd

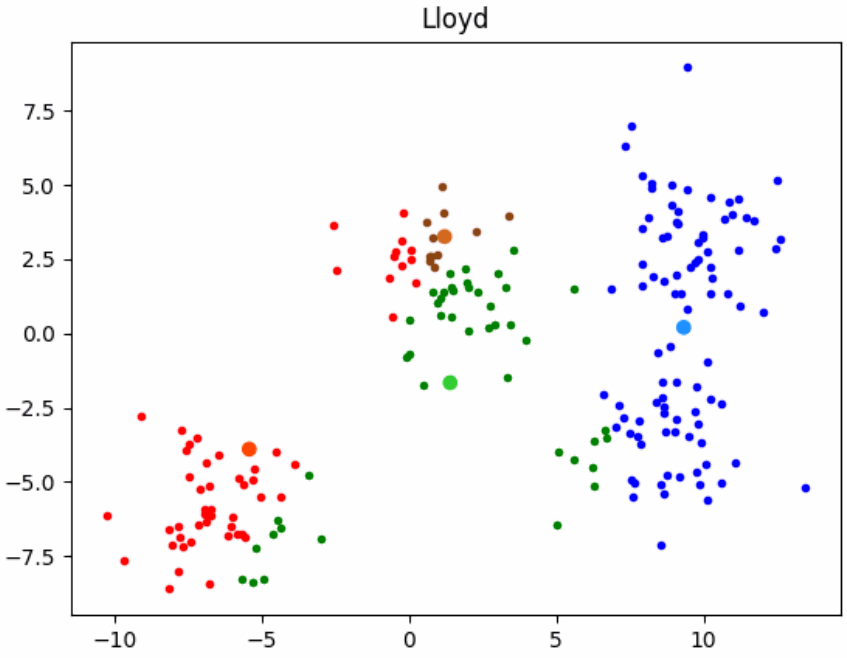
1: choose k as the number of centroids
2: Assign all points to its closest centroid
3: calculate the centroids as mean of their assigned points
4: **repeat**
5: reassign each datapoint to its closest centroid
6: recalculate centroids as mean over their assigned datapoints
7: **until** convergence

Ventajas:

- Uno de los algoritmos de clustering de mejor **coste computacional**

Desventajas:

- Problemas en la **asignación** de ciertos puntos a los clusters por actualizarlos una vez por iteración
- Alta influencia de la **selección inicial** de clusters



https://towardsdatascience.com/three-versions-of-k-means-cf939b65f4ea

MacQueen's K-Means

Algorithm 1 MacQueen

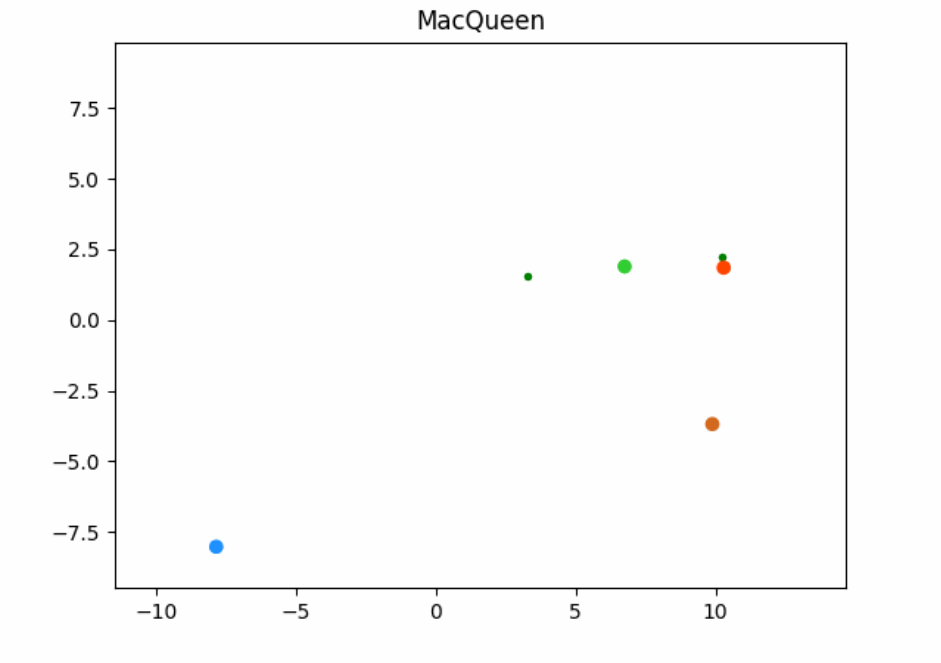
```
choose k as the number of clusters
randomly choose k datapoints as centroids
repeat
    for each datapoint do
        assign point to closest centroid
        recalculate centroid as mean over all points assigned
    end for
until convergence
```

Ventajas:

- **Mejora la asignación** de los puntos a los cluster
- Para clustering de datos que llegan en **flujo continuo**, es mejor computacionalmente.
- **Convergencia más rápida** al ajustarse los centroides por cada punto.

Desventajas:

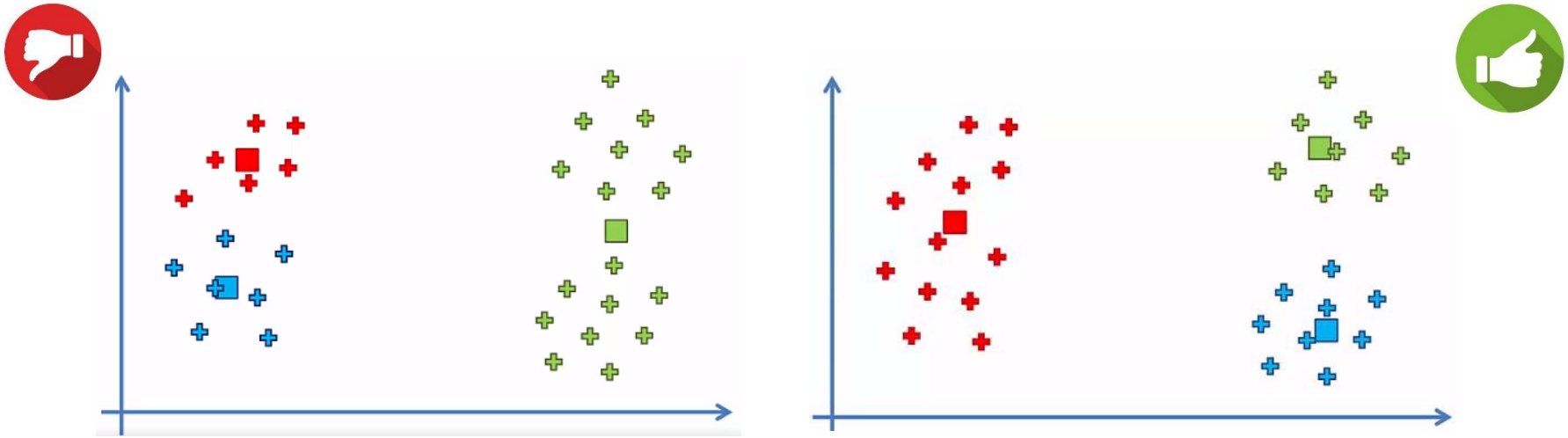
- En algunas situaciones **puede suponer un mayor coste computacional** (e.j., clustering de un gran volumen de datos)
- Alta influencia de la **selección inicial** de clusters



https://towardsdatascience.com/three-versions-of-k-means-cf939b65f4ea

Problemas de convergencia

- Con K-Means puedo llegar a un escenario donde asignar al centroide más cercano **no sea la mejor solución** (porque los clusters son más heterogéneos).
- ¿Puedo pensar en otro criterio para hacer las asignaciones?



https://www.linkedin.com/pulse/everything-k-means-navya-rao/

Hartigan-Wong K-Means

Algorithm 1 Hartigan-Wong

```
choose k as the number of centroids
randomly assign all points to a centroid
calculate the centroids as mean of their assigned points
repeat
    for each datapoint d do
        for each centroid c do
            assign datapoint d to centroid c
            compute the sum of squared distances from each point to its
            centroid
        end for
        assign d to the centroid which resulted in the smallest sum
        recalculate centroids as mean over all points assigned
    end for
until convergence
```

Inicialmente, asigno de manera **aleatoria** los puntos a los centroides (distinto de las implementaciones previas)

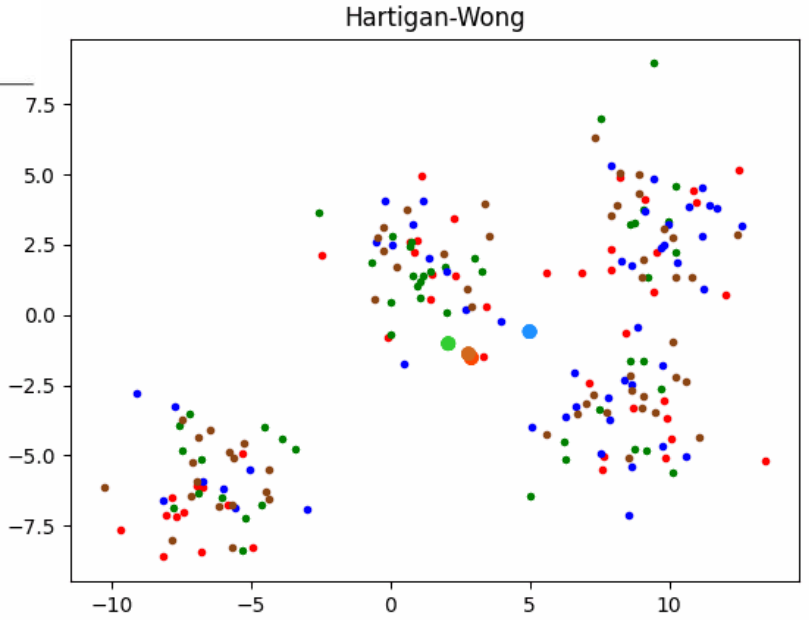
Función objetivo. Minimizar inercia/WCSS

Enfoque por iteración:

- Asocio el punto d a cada centroide -> calculo la **inercia** (con todos los puntos) de esa asociación -> Asocio de manera definitiva el punto al centroide que de menor inercia
- Así, no asocio necesariamente al centroide más cercano, sino al que **mejor inercia da**

Ventajas/Desventajas:

- Similares a MacQueen pero mitigando el problema de la inicialización de clusters.



https://towardsdatascience.com/three-versions-of-k-means-cf939b65f4ea

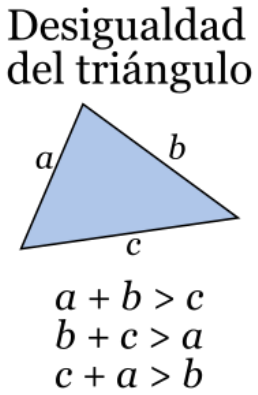
The k-means clustering technique: General considerations and implementation in Mathematica

Elkan's K-Means

- **Objetivo:** Reducir el coste computacional

Sigue los mismos pasos que K-Means, pero incluyendo **ciertas reglas** para reducir el coste computacional.

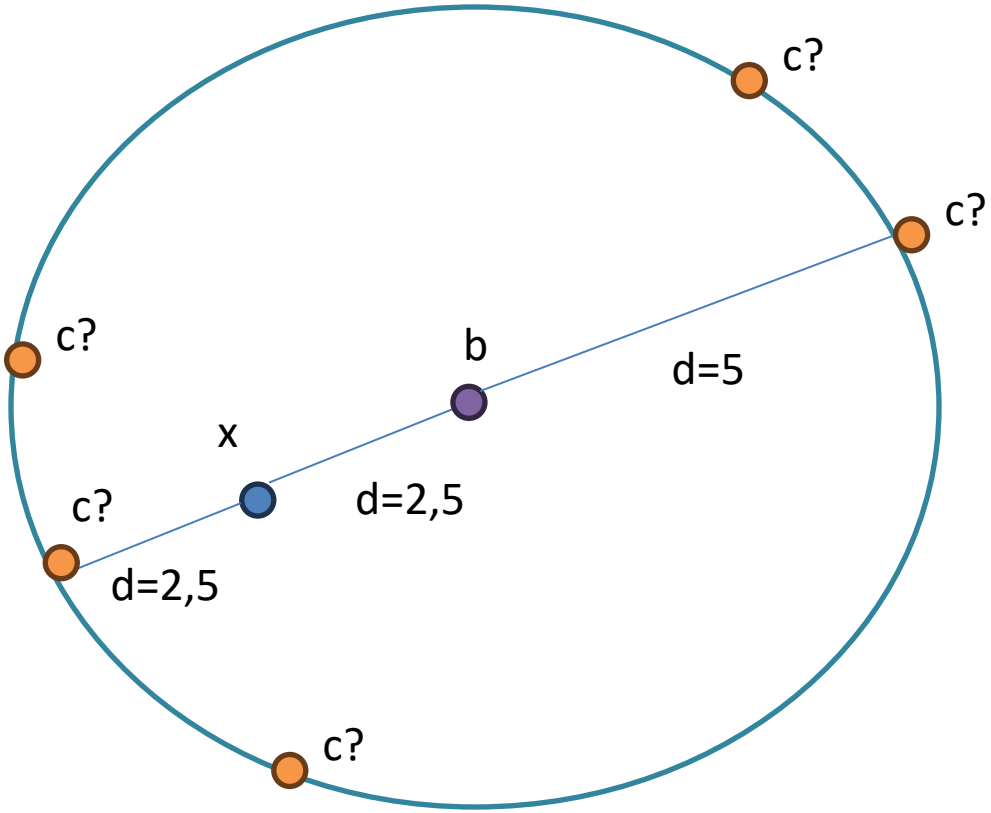
1. **Reducción de cálculos de distancia**
2. **Desigualdad triangular**
3. **Actualización eficiente**



https://es.wikipedia.org/wiki/Desigualdad_triangular

Elkan's K-Means

- **Inutición inicial**



Ejemplo
- d(x,b) = 2,5
- d(b,c) = 2x2,5 = 5

- Ejemplo de centroides: b y c
- Calculamos las distancias entre todos los centroides (no son muchos cálculos)
- Para un punto x, calculamos d(x,b).
- No sabemos d(x,c), pero, ¿es necesario calcularla?
  - Depende de la distancia entre b y c.
  - Según sea esta distancia, Podemos ver si compensa o no

- Vemos que como **2 * d(b,c) = d(x,b)**, c está, en el mejor de los casos, igual de cerca que b (incluso puede estar más lejos).
- No es necesario calcular d(x,c)
- Si 2 * d(b,c) > d(x,d), todavía está más claro que no es necesario calcularlo
- En cambio, si 2 * d(b,c) < d(x,c), ahí podría estar c más cerca y para asegurar sí que habría que calcular d(x,c)

Elkan's K-Means

First, pick initial centers. Set the lower bound $l(x, c) = 0$ for each point x and center c. Assign each x to its closest initial center $c(x) = \text{argmin}_c\, d(x, c)$, using Lemma 1 to avoid redundant distance calculations. Each time $d(x, c)$ is computed, set $l(x, c) = d(x, c)$. Assign upper bounds $u(x) = \min_c d(x, c)$.

Next, repeat until convergence:

1. For all centers c and c', compute $d(c, c')$. For all centers c, compute $s(c) = \frac{1}{2} \min_{c' \neq c} d(c, c')$.
2. Identify all points x such that $u(x) \leq s(c(x))$.
3. For all remaining points x and centers c such that
   (i) $c \neq c(x)$ and
   (ii) $u(x) > l(x, c)$ and
   (iii) $u(x) > \frac{1}{2} d(c(x), c)$:

Lemma 1: Let x be a point and let b and c be centers. If $d(b, c) \geq 2d(x, b)$ then $d(x, c) \geq d(x, b)$.

Proof: We know that $d(b, c) \leq d(b, x) + d(x, c)$. So $d(b, c) - d(x, b) \leq d(x, c)$. Consider the left-hand side: $d(b, c) - d(x, b) \geq 2d(x, b) - d(x, b) = d(x, b)$. So $d(x, b) \leq d(x, c)$. ■

- Se calculan primero las distancias entre centroides
- Despues, se va punto a punto viendo las distancias a cada centroide
  - Antes se comprueba si d(b,c) >= 2d(x,d)
  - En caso de que lo sea, como d(x,c) sería >= d(x,b), no es necesario calcularlo al ser un centroide mas lejano
  - Con ello, se asigna el pto al cluster del centroide mas cercano

Elkan, C. (2003). Using the triangle inequality to accelerate k-means. In *Proceedings of the 20th international conference on Machine Learning (ICML-03)* (pp. 147-153).

Elkan's K-Means

Next, repeat until convergence:

1. For all centers c and c', compute $d(c, c')$. For all centers c, compute $s(c) = \frac{1}{2} \min_{c' \neq c} d(c, c')$.
2. Identify all points x such that $u(x) \leq s(c(x))$.
3. For all remaining points x and centers c such that
   (i) $c \neq c(x)$ and
   (ii) $u(x) > l(x, c)$ and
   (iii) $u(x) > \frac{1}{2} d(c(x), c)$:

   3a. If $r(x)$ then compute $d(x, c(x))$ and assign $r(x) =$ false. Otherwise, $d(x, c(x)) = u(x)$.

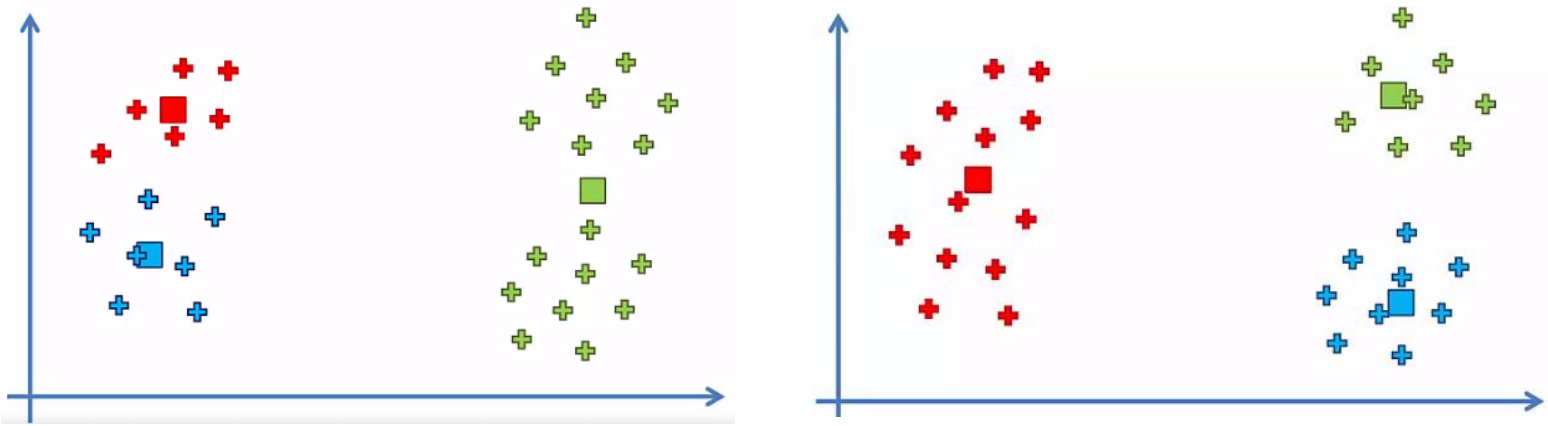
   3b. If $d(x, c(x)) > l(x, c)$ or $d(x, c(x)) > \frac{1}{2} d(c(x), c)$ then
   Compute $d(x, c)$
   If $d(x, c) < d(x, c(x))$ then assign $c(x) = c$.
4. For each center c, let $m(c)$ be the mean of the points assigned to c.
5. For each point x and center c, assign
   $l(x, c) = \max\{l(x, c) - d(c, m(c)), 0\}$.
6. For each point x, assign
   $u(x) = u(x) + d(m(c(x)), c(x))$
   $r(x) =$ true.
7. Replace each center c by $m(c)$.

- En base a las reglas comentadas previamente, en cada iteración no se recalcula la distancia de cada punto a cada cluster
- Se eligen solo determinados subconjuntos de puntos
- Esto se puede lograr gracias a los upper/lower bounds, que permiten hacer comparaciones sin tener que recalcular siempre las distancias

Elkan, C. (2003). Using the triangle inequality to accelerate k-means. In *Proceedings of the 20th international conference on Machine Learning (ICML-03)* (pp. 147-153).

K-Means ++

- **Objetivo:** Mejorar la selección inicial de centroides para evitar caer en soluciones subóptimas.



- **Random inicialization trap:** Según dónde se definan los puntos de inicio se pueden obtener agrupaciones totalmente distintas

https://www.linkedin.com/pulse/everything-k-means-navya-rao/

K-Means ++

Pasos del algoritmo

1. **Seleccionar el primer centroide*:** Elegir un punto al azar del conjunto de datos como el primer centroide, denotado como $C1$
2. **Calcular distancias:** Para cada punto p que aún no ha sido seleccionado como centroide, calcular la distancia $D(p)$ entre p y el centroide más cercano de los que ya han sido seleccionados.
3. **Seleccionar siguiente centroide:** Seleccionar el siguiente centroide con una probabilidad proporcional al cuadrado de la distancia $D(p)^2$. Es decir, los puntos que están más lejos de los centroides actuales tienen más probabilidad de ser seleccionados.

Formalmente, la probabilidad de seleccionar un punto p como nuevo centroide es:

$$P(p) = \frac{D(p)^2}{\sum_{p' \in P} D(p')^2}$$

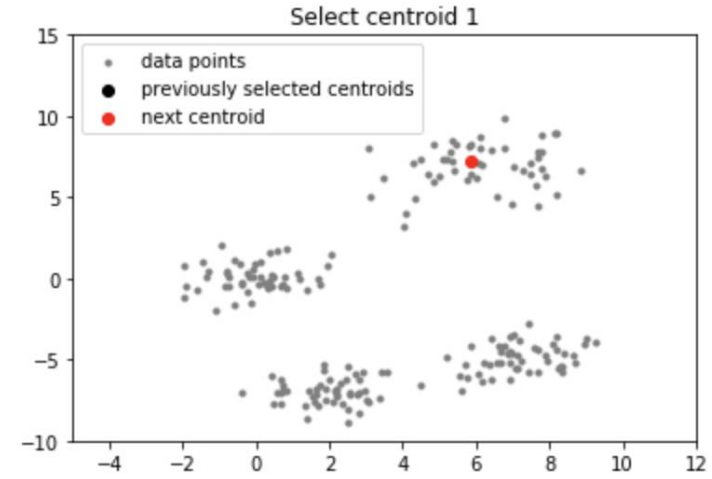
Donde:

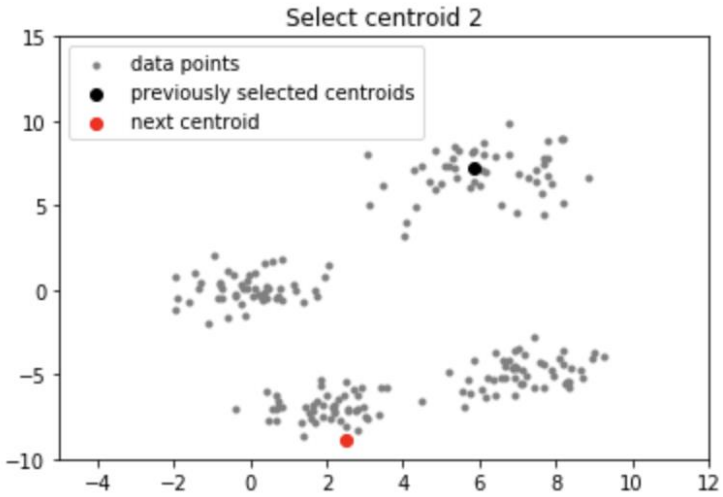
- $D(p)$ es la distancia entre p y el centroide más cercano.
- $P(p)$ es la probabilidad de que p sea seleccionado como un nuevo centroide.

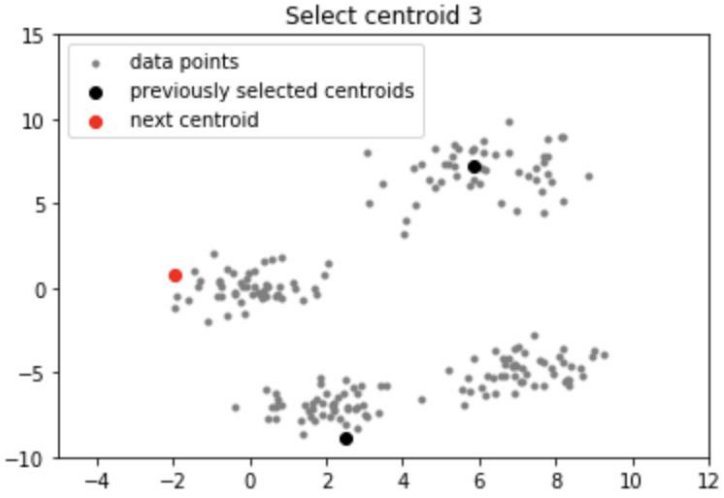
4. **Repetir 2-3** hasta seleccionar los k centroides.
5. **Ejecutar K-means** sobre estos centroides iniciales.

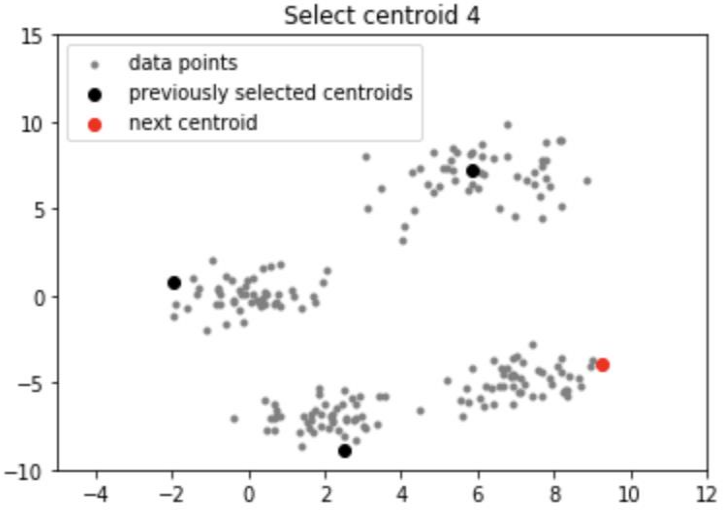
Centroide*: En realidad, **medoide**

K-Means ++









https://www.geeksforgeeks.org/ml-k-means-algorithm/

Soft K-Means / Fuzzy K-Means

- **Objetivo:** Permitir que los datos pertenezcan parcialmente a **múltiples grupos**, en lugar de asignar cada punto estrictamente a un único grupo.

1. **Inicialización:**
   - Seleccionar k centroides iniciales $C_1, C_2, \dots, C_k$
   - Inicializar los valores de **membresía/pertenencia u_{ij}** para cada punto p_i y cada grupo C_i aleatoriamente, asegurando que se cumple lo siguiente (al ser probabilidades):

$$\sum_{j=1}^{k} u_{ij} = 1$$

2. **Agrupación de centroides:**
   - Para cada grupo C_i se calcula el nuevo centroide como el promedio ponderado de todos los puntos, usando la función de pertenencia/membresía u_{ij}:

$$C_j = \frac{\sum_{i=1}^{n} u_{ij}^m \times p_i}{\sum_{i=1}^{n} u_{ij}^m}$$

   - m es el **parámetro de difusividad** (fuzziness). Normalmente es m>1. Si m=1, el resultado es el del algoritmo K-Means base.

Soft K-Means / Fuzzy K-Means

- **Objetivo:** Permitir que los datos pertenezcan parcialmente a **múltiples grupos**, en lugar de asignar cada punto estrictamente a un único grupo.

3. Actualizar los valores de pertenencia:

- Para cada punto p_i y cada grupo C_j se actualiza el valor de u_{ij}en función de la distancia entre el punto y el centroide. Los valores de pertenencia/membresía se calculan como

$$u_{ij} = \frac{1}{\sum_{k=1}^{K} \left(\frac{d(p_i, C_j)}{d(p_i, C_k)} \right)^{\frac{2}{m-1}}}$$

- Donde $d(p_i, C_j)$ es la distancia entre el punto p_iy el centroide C_j

4. Repetir:

- Se repiten los pasos 2-3 hasta alcanzar criterio de convergencia

Soft K-Means / Fuzzy K-Means - Ejemplo

Supongamos:

- Tenemos 3 puntos: $P_1 = (1, 1)$, $P_2 = (2, 2)$, y $P_3 = (5, 5)$.
- Queremos agrupar estos puntos en 2 clusters (es decir, $k = 2$).
- Usaremos un valor de $m = 2$, que es el parámetro de difusividad.
- Los centros iniciales de los clusters son $C_1 = (1, 1)$ y $C_2 = (5, 5)$.

Soft K-Means / Fuzzy K-Means - Ejemplo

Paso 1: Inicialización

- Asignamos **valores de pertenencia iniciales** para cada punto en relación a los dos clusters, de forma aleatoria. Por simplicidad, usamos valores aproximados de u_{ij}:

Punto	$u(P_1, C_1)$	$u(P_1, C_2)$	$u(P_2, C_1)$	$u(P_2, C_2)$	$u(P_3, C_1)$	$u(P_3, C_2)$
$P_1 = (1, 1)$	0.9	0.1	-	-	-	-
$P_2 = (2, 2)$	-	-	0.7	0.3	-	-
$P_3 = (5, 5)$	-	-	-	-	0.2	0.8

La suma de $u(P_i, C_1)$ y $u(P_i, C_2)$ para cada punto debe ser igual a 1:

$$u(P_1, C_1) + u(P_1, C_2) = 0.9 + 0.1 = 1$$

$$u(P_2, C_1) + u(P_2, C_2) = 0.7 + 0.3 = 1$$

$$u(P_3, C_1) + u(P_3, C_2) = 0.2 + 0.8 = 1$$

Soft K-Means / Fuzzy K-Means - Ejemplo

Paso 2: Actualización de los centroides

Ahora, calculamos los nuevos centroides C_1 y C_2 en función de los valores de pertenencia. El centroide C_j se actualiza utilizando la fórmula:

$$C_j = \frac{\sum_{i=1}^{n} u_{ij}^m \cdot p_i}{\sum_{i=1}^{n} u_{ij}^m}$$

Donde u_{ij}^m es el valor de pertenencia elevado a la potencia m, que en este caso es $m = 2$.

Para el centroide C_1:

$$C_1 = \frac{(0.9^2 \cdot P_1) + (0.7^2 \cdot P_2) + (0.2^2 \cdot P_3)}{(0.9^2 + 0.7^2 + 0.2^2)}$$

Calculamos los valores:

$$0.9^2 = 0.81, \quad 0.7^2 = 0.49, \quad 0.2^2 = 0.04$$

Ahora, sumamos:

$$C_1 = \frac{(0.81 \cdot (1,1)) + (0.49 \cdot (2,2)) + (0.04 \cdot (5,5))}{0.81 + 0.49 + 0.04}$$

$$C_1 = \frac{(0.81, 0.81) + (0.98, 0.98) + (0.20, 0.20)}{1.34}$$

$$C_1 = \frac{(1.99, 1.99)}{1.34} \approx (1.49, 1.49)$$

Soft K-Means / Fuzzy K-Means - Ejemplo

Paso 2: Actualización de los centroides

Ahora, calculamos los nuevos centroides C_1 y C_2 en función de los valores de pertenencia. El centroide C_j se actualiza utilizando la fórmula:

$$C_j = \frac{\sum_{i=1}^{n} u_{ij}^m \cdot p_i}{\sum_{i=1}^{n} u_{ij}^m}$$

Donde u_{ij}^m es el valor de pertenencia elevado a la potencia m, que en este caso es $m = 2$.

Para el centroide C_2:

$$C_2 = \frac{(0.1^2 \cdot P_1) + (0.3^2 \cdot P_2) + (0.8^2 \cdot P_3)}{(0.1^2 + 0.3^2 + 0.8^2)}$$

Calculamos los valores:

$$0.1^2 = 0.01, \quad 0.3^2 = 0.09, \quad 0.8^2 = 0.64$$

Ahora, sumamos:

$$C_2 = \frac{(0.01 \cdot (1,1)) + (0.09 \cdot (2,2)) + (0.64 \cdot (5,5))}{0.01 + 0.09 + 0.64}$$

$$C_2 = \frac{(0.01, 0.01) + (0.18, 0.18) + (3.2, 3.2)}{0.74}$$

$$C_2 = \frac{(3.39, 3.39)}{0.74} \approx (4.58, 4.58)$$

Soft K-Means / Fuzzy K-Means - Ejemplo

Paso 3: Actualización de los valores de pertenencia

Después de actualizar los centroides, recalculamos los valores de pertenencia u_{ij} para cada punto utilizando la fórmula:

$$u_{ij} = \frac{1}{\sum_{k=1}^{K} \left(\frac{d(p_i, C_j)}{d(p_i, C_k)} \right)^{\frac{2}{m-1}}}$$

Donde $d(p_i, C_j)$ es la distancia entre el punto p_i y el centroide C_j, y $m = 2$.

Para el punto $P_1 = (1, 1)$:

1. Calculamos la distancia $d(P_1, C_1)$ y $d(P_1, C_2)$:

$$d(P_1, C_1) = d((1,1), (1.49, 1.49)) = \sqrt{(1.49-1)^2 + (1.49-1)^2} \approx 0.69$$

$$d(P_1, C_2) = d((1,1), (4.58, 4.58)) = \sqrt{(4.58-1)^2 + (4.58-1)^2} \approx 5.06$$

2. Actualizamos los valores de pertenencia:

$$u(P_1, C_1) = \frac{1}{1 + \left(\frac{0.69}{5.06}\right)^2} \approx 0.98$$

Para j=k

$$u(P_1, C_2) = \frac{1}{1 + \left(\frac{5.06}{0.69}\right)^2} \approx 0.02$$

Soft K-Means / Fuzzy K-Means - Ejemplo

Repetimos para P_2 y P_3, obteniendo:

Punto	$u(P_1, C_1)$	$u(P_1, C_2)$	$u(P_2, C_1)$	$u(P_2, C_2)$	$u(P_3, C_1)$	$u(P_3, C_2)$
$P_1 = (1, 1)$	0.98	0.02	-	-	-	-
$P_2 = (2, 2)$	-	-	0.962	0.038	-	-
$P_3 = (5, 5)$	-	-	-	-	0.014	0.986

Paso 4: Repetir

Repetimos los pasos 2 y 3 hasta que los cambios en los valores de pertenencia sean muy pequeños, indicando convergencia.

Otros: K-Prototype

- **Objetivo:** Poder trabajar tanto con datos numéricos como **categóricos.**

  - **Distancia Mixta**: Se utiliza una métrica que combina la **distancia euclidiana** para los datos numéricos y la **coincidencia de categorías** para los datos categóricos:
    - Para los datos numéricos, se usa la distancia euclidiana estándar.
    - Para los datos categóricos, se usa una función de disimilitud binaria (0 si las categorías coinciden, 1 si no coinciden).

La fórmula para la distancia mixta entre un punto x y un prototipo p es:

$$d(x,p) = \sum_{i \in N} (x_i - p_i)^2 + \gamma \sum_{j \in C} \delta(x_j, p_j)$$

Donde:

- N es el conjunto de variables numéricas.
- C es el conjunto de variables categóricas.
- $\delta(x_j, p_j)$ es 0 si $x_j = p_j$ (las categorías coinciden) y 1 si no coinciden.
- γ es un parámetro que controla la importancia de las variables categóricas frente a las numéricas.

Ji, J., Bai, T., Zhou, C., Ma, C., & Wang, Z. (2013). An improved k-prototypes clustering algorithm for mixed numeric and categorical data. *Neurocomputing*, *120*, 590-596.

En resumen

- Existen **diferentes aproximaciones** sobre K-Means que buscan mejorar el algoritmo base en distintos aspectos.
- Algunos aspectos que mejoran:
  - **Inicialización** de los clusters (K-Means ++)
  - **Asignación** de clusters para evitar caer en soluciones subóptias (MacQueen, Hartigan-Wong)
  - **Coste computacional** (Elkan)
  - Abordar casos donde los **puntos puedan pertenecer a varios clusters** (Fuzzy K-Means)
  - Permitir usar **variables categóricas** (K-Prototype)

En la próxima semana

¿Cuál es la metodología para descubrir estructuras y categorías latentes dentro de un conjunto de datos, sin tener información previa de categorías o etiquetas?

Bibliografía para la asignatura

UML (general)

- **An Introduction to Statistical Learning: with Applications in Python (James et al.)**
- **Information Theory, Inference, and Learning Algorithms (MacKay)**

Detección anomalías:

- **Outlier Analysis (Charu C. Aggarwal)**

RL:

- **Reinforcement Learning: An Introduction (Surton, Barto)**

www.unir.net