

# Aprendizaje Automático

## **Clase 8: Clustering**

Clustering Aglomerativo  
K-Means

Mezclas de Gaussianas y Expectation Maximization  
DBSCAN

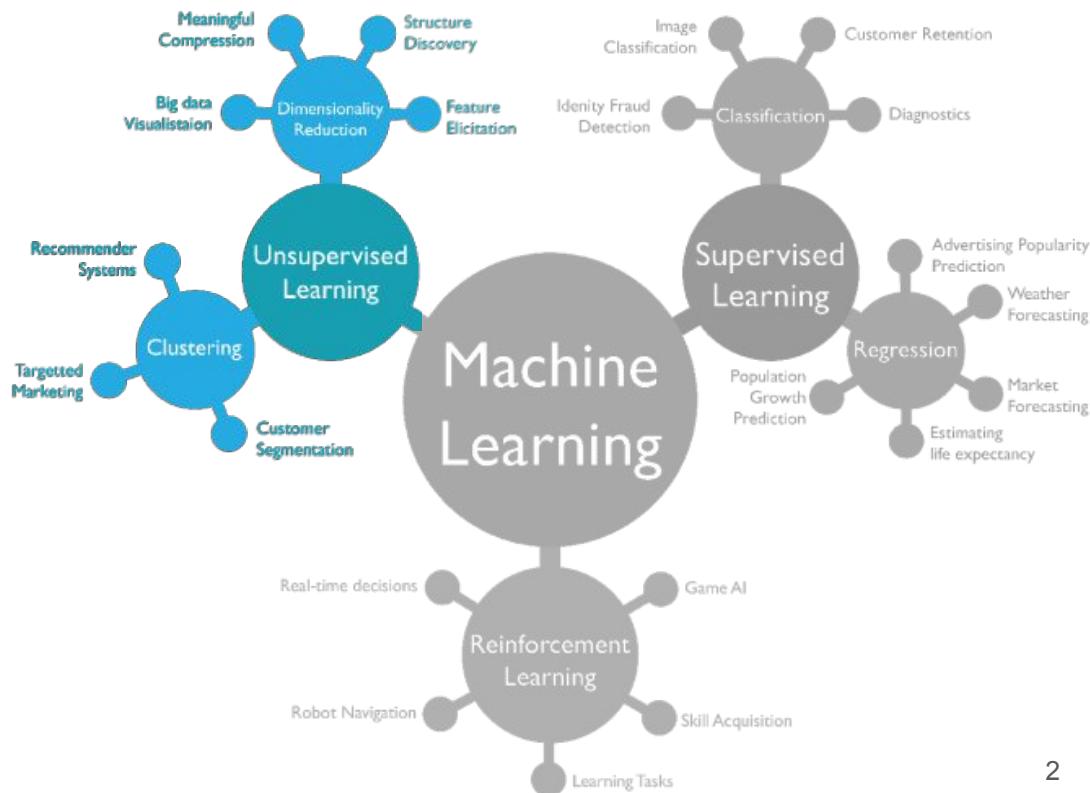
# Aprendizaje no supervisado

## Objetivo del aprendizaje NO supervisado

Busca **descubrir información y estructura** implícita en los datos **sin ninguna guía externa**.

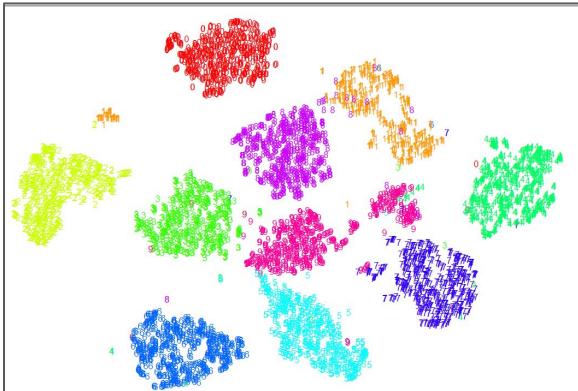
Es decir, descubrir patrones o estructuras ocultas en los datos sin la presencia de etiquetas o respuestas predefinidas.

Sirve para **entender, resumir, relacionar y visualizar los datos** – generalmente de alta dimensionalidad.

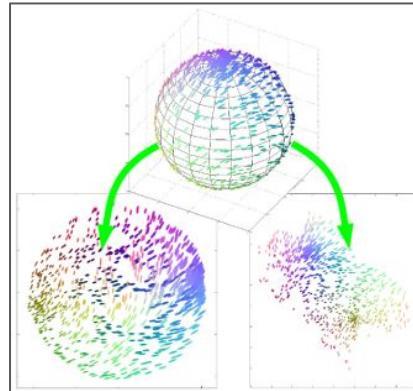


# Aprendizaje no supervisado

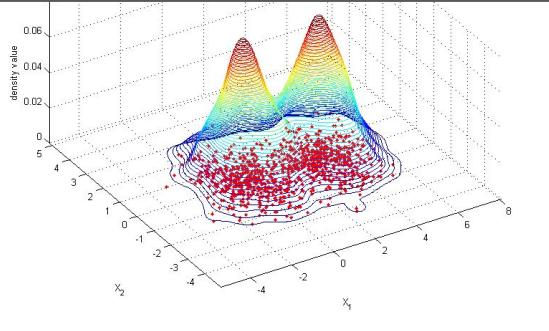
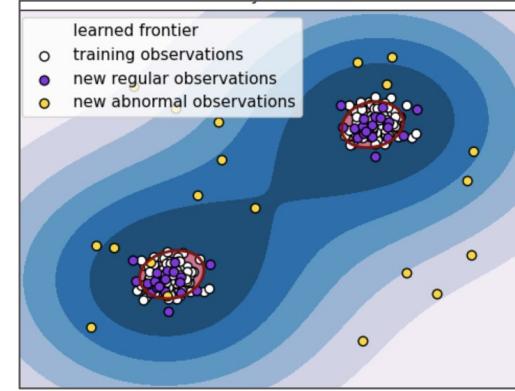
Clustering



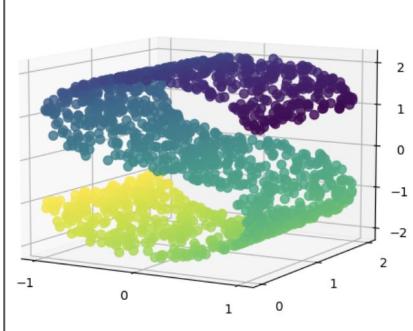
Reducción dimensional



Detección de Anomalías



Estimación de densidades

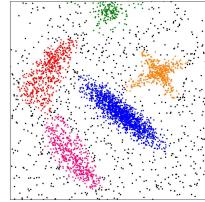


Aprendizaje de variedades



Sistemas de recomendación

# Clustering



# Clustering

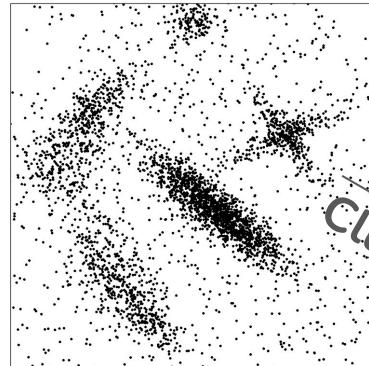
Dado un conjunto de datos no etiquetados  
 $X = \{x^{(1)}, x^{(2)}, \dots, x^{(n)}\}$

Se busca encontrar un conjunto de *clusters* (grupos)  
 $C = \{C_1, C_2, \dots, C_K\}$

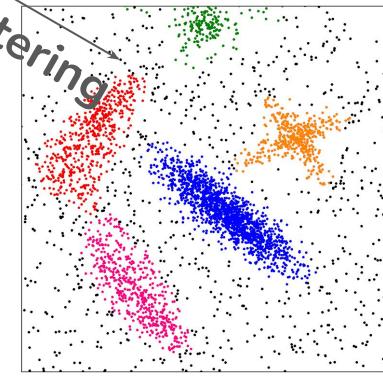
Suponemos que cada clúster contiene elementos similares y se busca minimizar alguna medida de disimilitud o distancia.

**Objetivo:** encontrar grupos de instancias tales que las instancias en un cluster sean:

- similares entre sí,
- diferentes de las instancias en otros clusters.



*Clustering*



Notar que el color asignado a cada cluster es aleatorio, no hay un orden entre los distintos clusters.

**Nota:** Los algoritmos que veremos hoy son heurísticas.

Hallar óptimos globales: “**cuál es la mejor asignación posible de grupos a cada punto**” es un problema NP ~  $O(2^n)$

# Clustering

Dado un conjunto de datos no etiquetados

$$X = \{x^{(1)}, x^{(2)}, \dots, x^{(n)}\}$$

Se busca encontrar un conjunto de *clusters* (grupos)

$$C = \{C_1, C_2, \dots, C_K\}$$

Suponemos que cada clúster contiene elementos similares y se busca minimizar alguna medida de disimilitud o distancia.

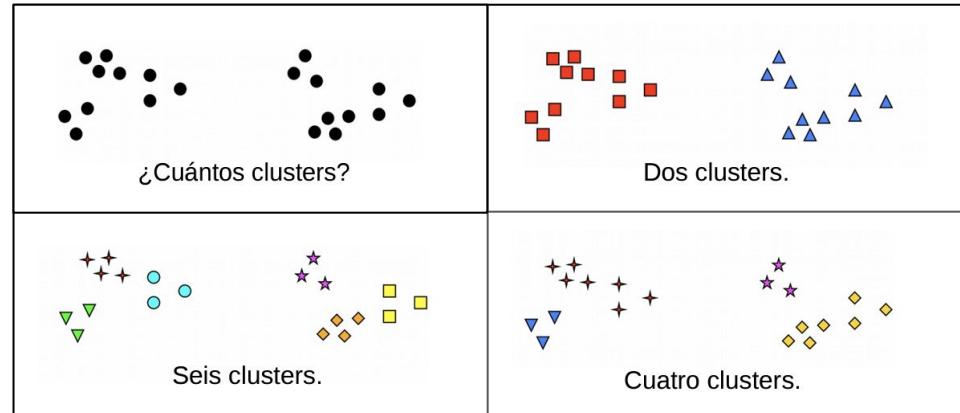
**Objetivo:** encontrar grupos de instancias tales que las instancias en un cluster sean:

- similares entre sí,
- diferentes de las instancias en otros clusters.

ambiguo

**"Cluster" es un concepto **claro**.**

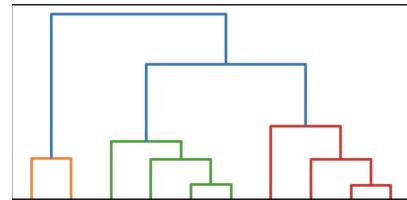
¿Cuántos clusters hay en la imagen? ¿para un K fijo, hay una sola posibilidad?



La definición de *cluster* depende de la naturaleza de los datos y los resultados deseados.

**Nota:** Rara vez podemos visualizar los datos de esta manera tan clara. En general  $p \gg 3$  dimensiones **estamos a oscuras!**. Las técnicas de reducción dimensional (PCA, MDS, T-SNE, etc) son buenas linternas.

# Clustering basado en conectividad

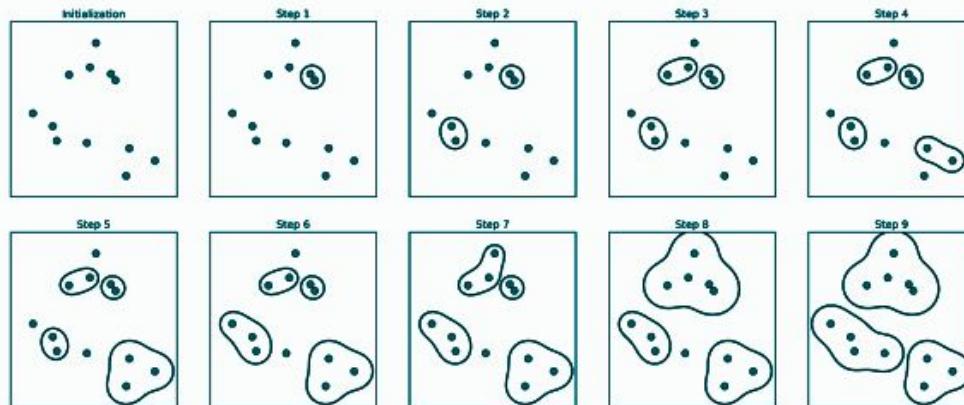


# Clustering Aglomerativo

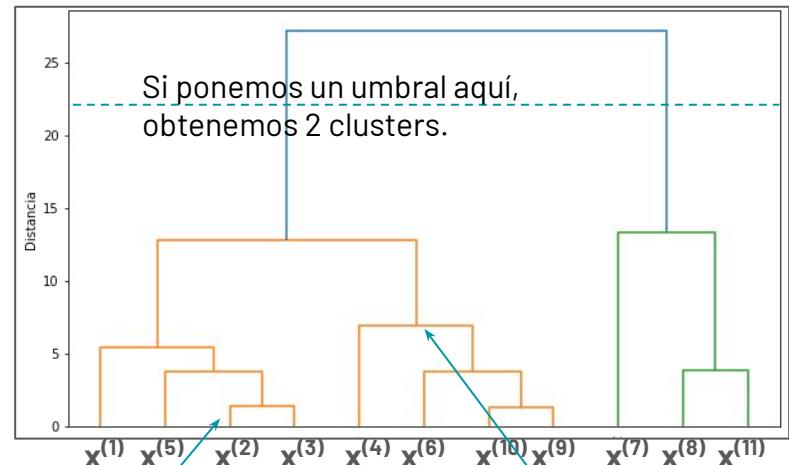
Clustering jerárquico bottom up

## Idea del Algoritmo

1. Cada punto, es un *cluster*.  $C_1 = x^{(1)}$ ;  $C_2 = x^{(2)}$ ; etc.
2. Mergear 2 clusters según algún criterio de cercanía.
3. Repetir paso 2 hasta que haya 1 solo cluster.



Produce un “**Dendrograma**” (árbol jerárquico de clusters)

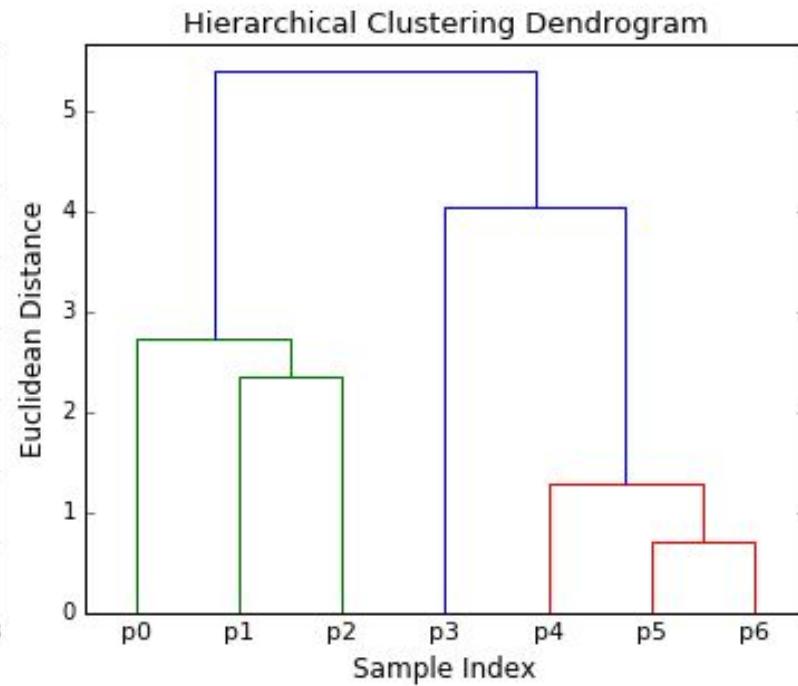
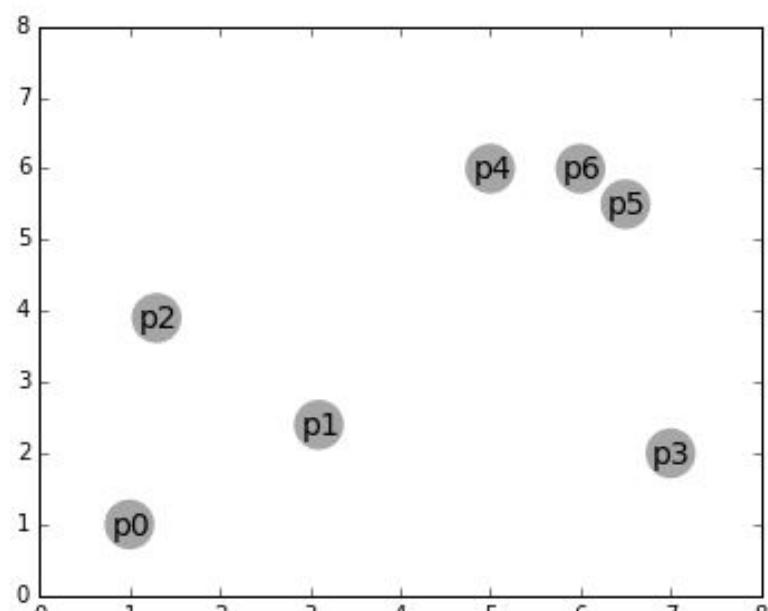


La altura de la unión indica la distancia entre clusters.  
Aquí la distancia entre el cluster  $C=\{x^{(2)}\}$  y  $C'=\{x^{(3)}\}$

Representa la unión de clusters.  
Aquí la del cluster  $C=\{x^{(6)}, x^{(10)}, x^{(9)}\}$  con el cluster  $C'=\{x^{(4)}\}$

# Clustering Aglomerativo

Clustering jerárquico bottom up



# Clustering Aglomerativo

Clustering jerárquico bottom up

**Ejemplo**, supongamos que nuestros datos tienen como atributos **lat** y **long** de distintas **ciudades** argentinas:

## Instancias:

Bariloche; Buenos Aires; Catamarca; Córdoba; La Pampa; La Rioja; Mendoza; Neuquén; Rosario; Santa Fe.

X =

	ciudad	latitud	longitud	poblacion
0	Bariloche	-41.13	-71.31	113000
1	Buenos Aires	-34.61	-58.38	2891000
2	Catamarca	-28.47	-65.78	42460
3	Córdoba	-31.41	-64.18	1391000
4	La Pampa	-36.62	-64.29	318951
5	La Rioja	-29.41	-66.85	17872
6	Mendoza	-32.88	-68.83	1168000
7	Neuquén	-38.95	-68.06	265058
8	Rosario	-32.95	-60.65	1174000
9	Santa Fe	-31.63	-60.70	554700



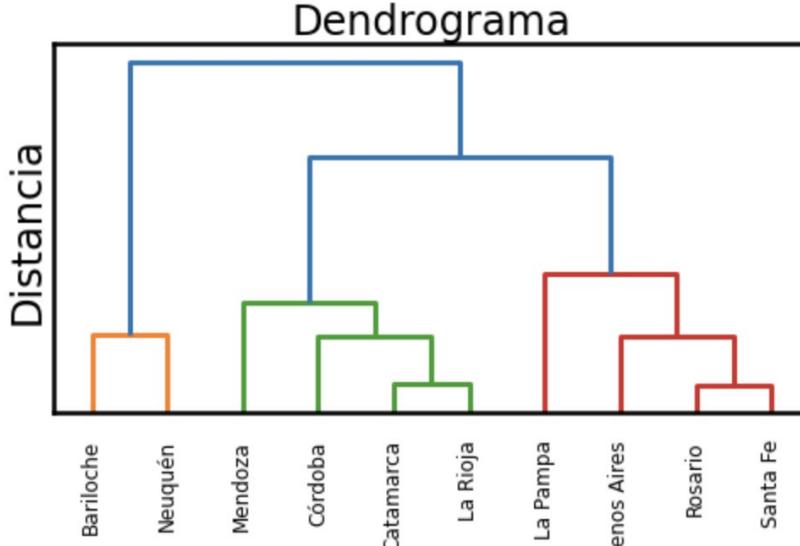
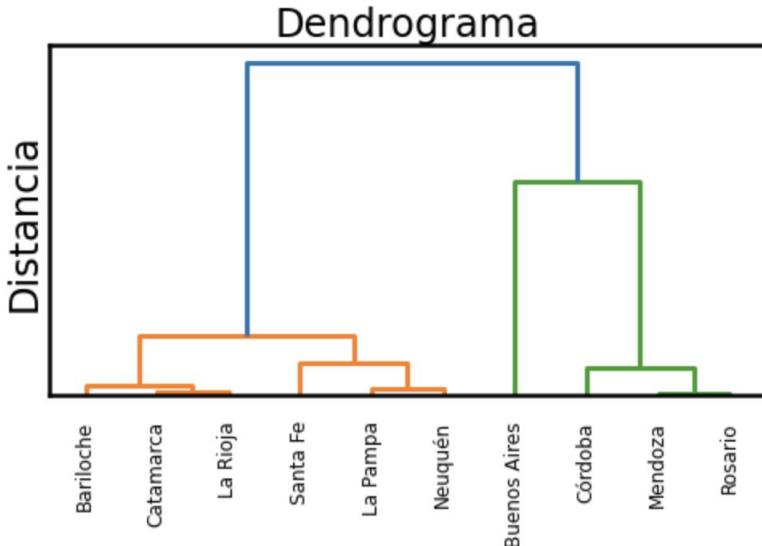
# Clustering Aglomerativo

Clustering jerárquico bottom up

Sean los siguientes dendrogramas, uno a partir de  $X = [\text{población}, \text{latitud}, \text{longitud}]$ , el otro a partir de  $X = [\text{latitud}, \text{longitud}]$ .

Responder:

- ¿Cuál es cuál?
- ¿Qué **información** podés sacar de estos dendrogramas?
- ¿Hay algún **problema evidente** en alguno de los dos casos?



# Distancias entre Clusters

Para medir distancia entre dos clusters A y B, se definen medidas basadas en distintas definiciones de vinculación ("Linkages")

## Vinculación Completa

**Disimilitud máxima entre clústeres:** dadas las distancias entre todo par de puntos de los clusters A y B, respectivamente.

Quedarse con la **mayor**.

$$\max_{a \in A, b \in B} d(a, b)$$

## Vinculación Simple

**Disimilitud mínima entre clústeres:** dadas las distancias entre todo par de puntos de los clusters A y B, respectivamente.

Quedarse con la **menor**.

$$\min_{a \in A, b \in B} d(a, b)$$

## Vinculación Promedio

**Disimilitud promedio entre clústeres:** dadas las distancias entre todo par de puntos de los clusters A y B, respectivamente.

Computar la **media**.

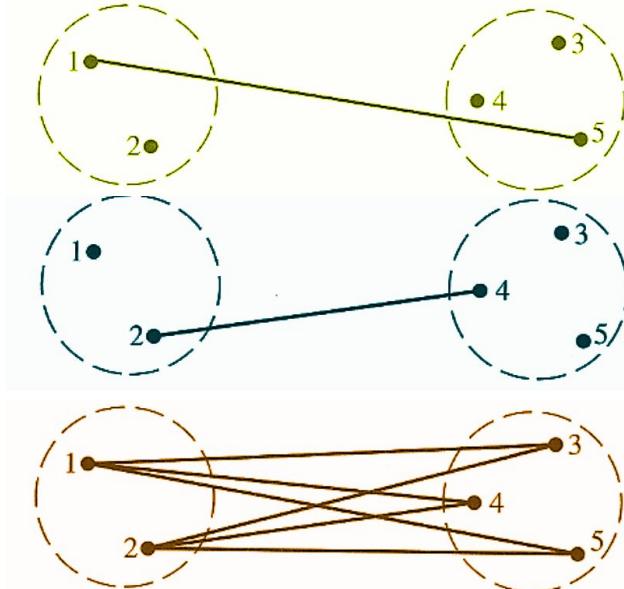
$$\frac{1}{|A| \cdot |B|} \sum_{a \in A} \sum_{b \in B} d(a, b)$$

## Otras

**Disimilitud entre centroides** (lo veremos luego),

**Método Ward** (sklearn),

etc.



A partir de calcular  $\text{Dist}(A, C)$ , y  $\text{Dist}(B, C)$ , calcular  $\text{Dist}(A \cup B, C)$  debería ser  $O(1)$  para poder garantizar cotas de complejidad en el algoritmo

**Ejercicio:** ¿Estas tres definiciones lo cumplen? ¿cómo?

# Clustering Aglomerativo

**Algoritmo** Clustering jerárquico bottom up

1. Cada punto es un cluster:  $C_1 = x^{(1)}$ ; ... ;  $C_n = x^{(n)}$
2. **Calcular una matriz de distancias** entre todo par de clusters.
3. Buscar en la matriz de distancia **el par de clusters más similares**.
4. Fusionar 2 clusters: A y B según su distancia.
  - a. Borrar los elementos de la matriz de distancias correspondientes a los clusters A y B.
  - b. Agregar una nueva fila y columna que contenga la distancia entre  $A \cup B$  y el resto de los clusters.
5. **Repetir** los pasos 3 y 4 hasta conseguir la cantidad de clusters deseados (o hasta que haya sólo un cluster).

## Ejemplo:

Matriz de Distancias entre las instancias originales

	Bariloche	Buenos Aires	Catamarca	Córdoba	La Pampa	La
Buenos Aires	1345 km					
Catamarca	1495 km	978 km				
Córdoba	1255 km	647 km	361 km			
La Pampa	787 km	579 km	917 km	579 km		
La Rioja	1364 km	985 km	148 km	339 km	836 km	
Mendoza	943 km	985 km	570 km	467 km	587 km	42
Neuquén	368 km	987 km	1184 km	909 km	421 km	10
Rosario	1311 km	279 km	699 km	374 km	526 km	70
Santa Fe	1418 km	396 km	602 km	331 km	646 km	63

**Ejercicio.** Calcular la complejidad temporal de cada paso. Suponer que la distancia entre dos clusters en donde sus elementos son de dimensión d es  $O(d)$ .

# Clustering Aglomerativo

**Algoritmo** Clustering jerárquico bottom up

1. Cada punto es un cluster:  $C_1 = x^{(1)}$ ; ... ;  $C_n = x^{(n)}$   $O(n)$
2. **Calcular una matriz de distancias** entre todo par de clusters.  $O(n^2 * d)$
3. Buscar en la matriz de distancia **el par de clusters más similares.**  $O(n^2)$
4. Fusionar 2 clusters: A y B según su distancia.
  - a. Borrar los elementos de la matriz de distancias correspondientes a los clusters A y B.  $O(n)$
  - b. Agregar una nueva fila y columna que contenga la distancia entre  $A \cup B$  y el resto de los clusters.  $O(n)$
5. **Repetir** los pasos 3 y 4 hasta conseguir la cantidad de clusters deseados (o hasta que haya sólo un cluster).  
 $O(n)$  repeticiones

Total Tiempo:  $O(n^2 * d) + O(n^3)$

Total Espacio:  $\Omega(n^2)$

## Ejemplo:

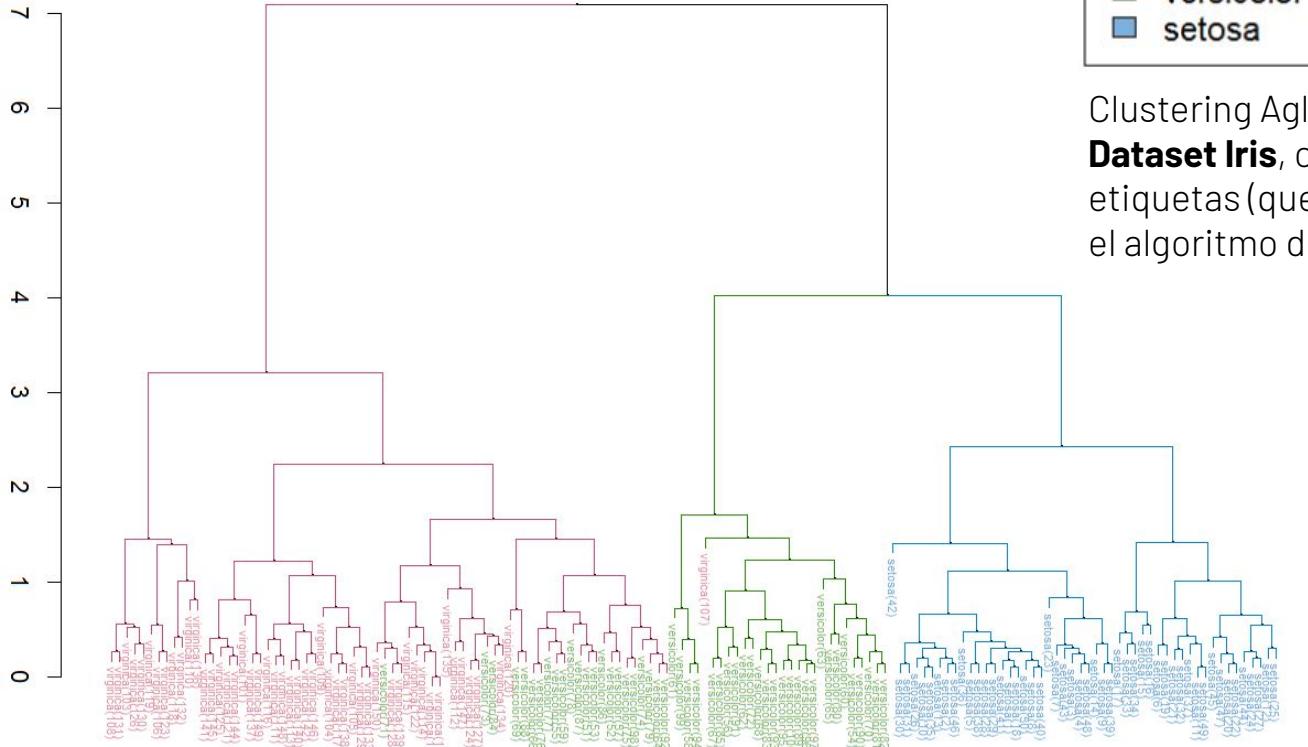
Matriz de Distancias entre las instancias originales

	Bariloche	Buenos Aires	Catamarca	Córdoba	La Pampa	La
Buenos Aires	1345 km					
Catamarca	1495 km	978 km				
Córdoba	1255 km	647 km	361 km			
La Pampa	787 km	579 km	917 km	579 km		
La Rioja	1364 km	985 km	148 km	339 km	836 km	
Mendoza	943 km	985 km	570 km	467 km	587 km	42
Neuquén	368 km	987 km	1184 km	909 km	421 km	10
Rosario	1311 km	279 km	699 km	374 km	526 km	70
Santa Fe	1418 km	396 km	602 km	331 km	646 km	62

**Ejercicio.** Calcular la complejidad temporal de cada paso. Suponer que la distancia entre dos clusters en donde sus elementos son de dimensión d es  $O(d)$ .

[1973, R. Sibson] SLINK (para simple linkage). Tiempo:  $O(n^2)$  espacio:  $O(n)$

# Clustering Aglomerativo



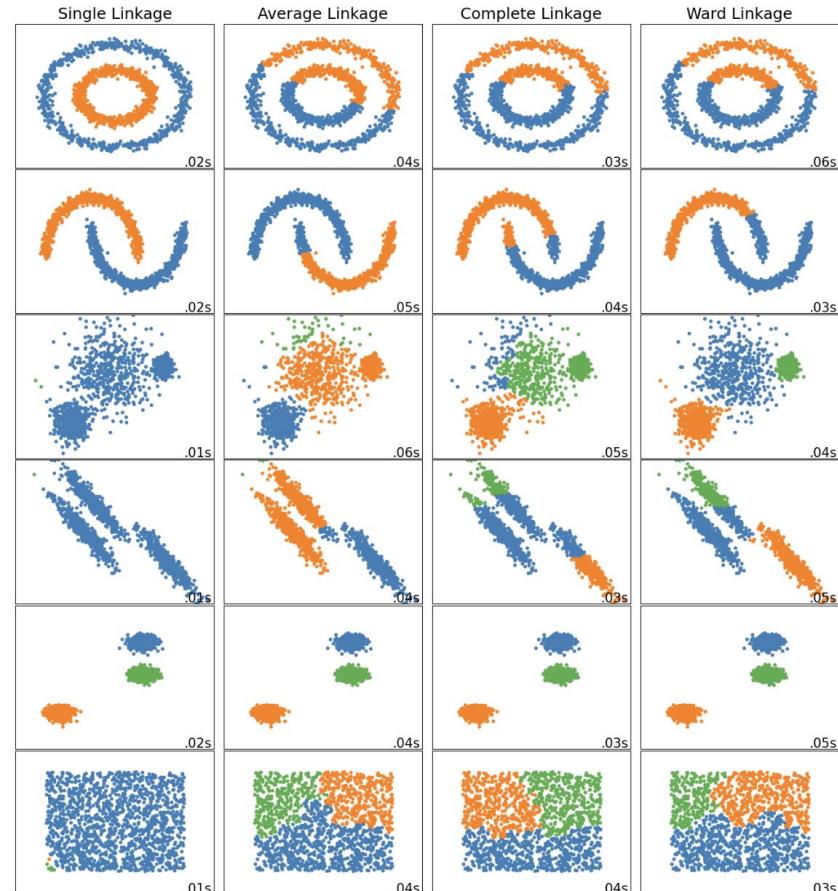
Clustering Aglomerativo aplicado al  
**Dataset Iris**, coloreado según las  
etiquetas (que no son usadas durante  
el algoritmo de clustering).

# Clustering Aglomerativo

La decisión de cómo medir distancia entre clusters afecta el resultado. Algunos ejemplos:

**Vinculación simple:** Un objeto se agregará a un grupo siempre que esté cerca de cualquiera de los otros objetos del grupo, incluso si está relativamente lejos de todos los demás. Por lo tanto, el enlace simple tiene una tendencia a producir grupos largos.

**Vinculación completa:** tiende a producir agrupaciones homogéneas, pero puede ser muy sensible a outliers en los datos.



Fuente imagen:

[https://scikit-learn.org/stable/auto\\_examples/cluster/plot\\_linkage\\_comparison.html#sphx-glr-auto-examples-cluster-plot-linkage-comparison-py](https://scikit-learn.org/stable/auto_examples/cluster/plot_linkage_comparison.html#sphx-glr-auto-examples-cluster-plot-linkage-comparison-py)

# Clustering Aglomerativo

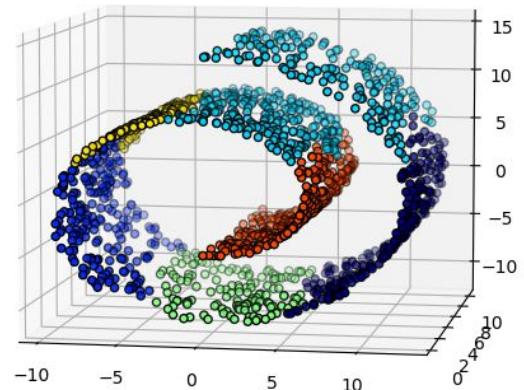
**Forzar la conectividad:** otra idea es crear un grafo de conectividad y **no permitir a los clusters juntar puntos no conectados en el grafo.**

```
from sklearn.cluster import AgglomerativeClustering
from sklearn.neighbors import kneighbors_graph

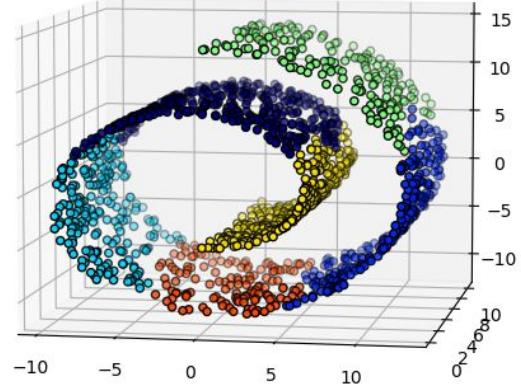
grafo = kneighbors_graph(
    X,
    n_neighbors=10,
    include_self=False
)

clustering = AgglomerativeClustering(
    n_clusters=6,
    metric="euclidean",
    linkage="single",
    connectivity=grafo
).fit(X)
```

Sin conectividad forzada



Con conectividad forzada



# Clustering basado en centroides

# K-Means

Dado un conjunto de datos no etiquetados

$$X = \{x^{(1)}, x^{(2)}, \dots, x^{(n)}\}$$

Se busca encontrar un conjunto de clústers  
(grupos)

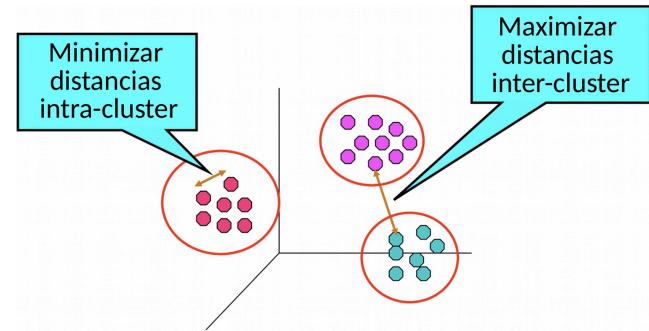
$$C = \{C_1, C_2, \dots, C_K\}$$

Cada clúster contiene elementos similares y se busca minimizar alguna medida de disimilitud o distancia.

En K-means:

- $C_1 \cup C_2 \cup \dots \cup C_K = X$
- $C_i \cap C_j = \emptyset$  (para todo  $i, j: i \neq j$ )

¿Qué pasa si hacemos explícito lo que queremos encontrar?



Queremos **minimizar las distancias intra-cluster** y al mismo tiempo **maximizar las distancias inter-cluster**

Resulta que **minimizar las distancias intra-cluster** automáticamente **maximiza las distancias inter-cluster** (ver Elements sec.)

14.3.5). Podemos quedarnos con una.

# K-Means

Dado un conjunto de datos no etiquetados

$$X = \{x^{(1)}, x^{(2)}, \dots, x^{(n)}\}$$

Se busca encontrar un conjunto de clústers  
(grupos)

$$C = \{C_1, C_2, \dots, C_K\}$$

Cada clúster contiene elementos similares y se busca minimizar alguna medida de disimilitud o distancia.

En K-means:

- $C_1 \cup C_2 \cup \dots \cup C_K = X$
- $C_i \cap C_j = \emptyset$  (para todo  $i, j: i \neq j$ )

Entonces, busquemos un agrupamiento tal que **la variación dentro de cada cluster es lo más pequeña posible:**

$$\min_{C_1, \dots, C_K} \left\{ \sum_{k=1}^K WCV(C_k) \right\}$$

$WCV(C_k)$  “**within-cluster variation**” indica cuánto difieren entre sí las observaciones dentro del cluster  $C_k$ . Para datos continuos,  $WCV$  se define en función de la distancia euclíadiana (para otras distancias ver Elements sec. 14.3):

**Es decir,** la división en  **$K$  clusters** de modo que la **variación total dentro del cluster, sumada sobre todos los  $K$  clusters**, sea lo más pequeña posible.

# K-Means

Dado un conjunto de datos no etiquetados  
 $X = \{x^{(1)}, x^{(2)}, \dots, x^{(n)}\}$

Se busca encontrar un conjunto de clústers  
(grupos)

$$C = \{C_1, C_2, \dots, C_K\}$$

Cada clúster contiene elementos similares y se busca minimizar alguna medida de disimilitud o distancia.

En K-means:

- $C_1 \cup C_2 \cup \dots \cup C_K = X$
- $C_i \cap C_j = \emptyset$  (para todo  $i, j: i \neq j$ )

**Entonces,** busquemos un agrupamiento tal que **la variación dentro de cada cluster es lo más pequeña posible:**

Si la distancia elegida es la distancia euclíadiana, se puede probar que la función a minimizar se puede reescribir como:

$$J_X(C) = \min_{C_1, \dots, C_K} \left\{ \sum_{k=1}^K WCV(C_k) \right\}$$

$$\stackrel{\text{eucl}}{=} \frac{1}{2} \sum_{k=1}^K \sum_{C(i)=k} \sum_{C(i')=k} \|x^{(i)} - x^{(i')}\|_2^2$$

$$= \sum_{k=1}^K \frac{1}{|C_k|} \sum_{C(i)=k} \|x^{(i)} - \tilde{x}_{(k)}\|_2^2$$

En donde  $\tilde{x}_{(k)}$  es el vector promedio de los elementos pertenecientes al cluster. También llamado "**centroide**".  
¿Cómo minimizamos esto? Con una heurística: K-MEANS

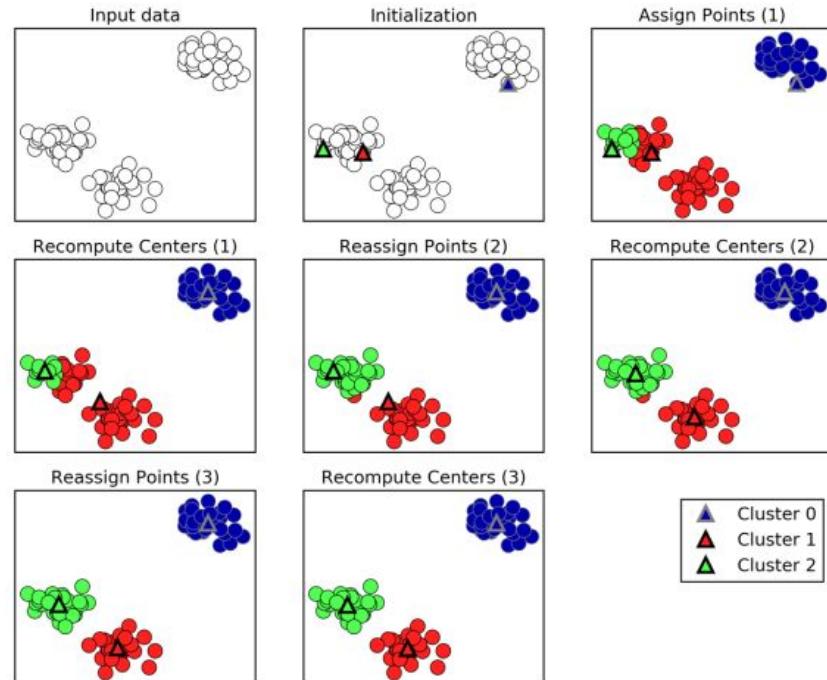
# K-Means

## Algoritmo

Hiperparámetros: K

1. Ubicar K **centroides al azar**.
2. Iterar hasta cumplir con algún criterio de convergencia:
  - a. **Computar distancias** (euclidianas) entre cada punto a cada uno de los centroides.
  - b. A cada punto, **asignar cluster según cercanía a cada centroide**.
  - c. Mover **centroides a la media de cada cluster**.

El **centroide** de un cluster es el vector promedio de las instancias pertenecientes al cluster (vector con promedios de cada atributo).



# K-Means

Se puede probar que el algoritmo de K-medias garantiza disminuir el costo en cada paso.

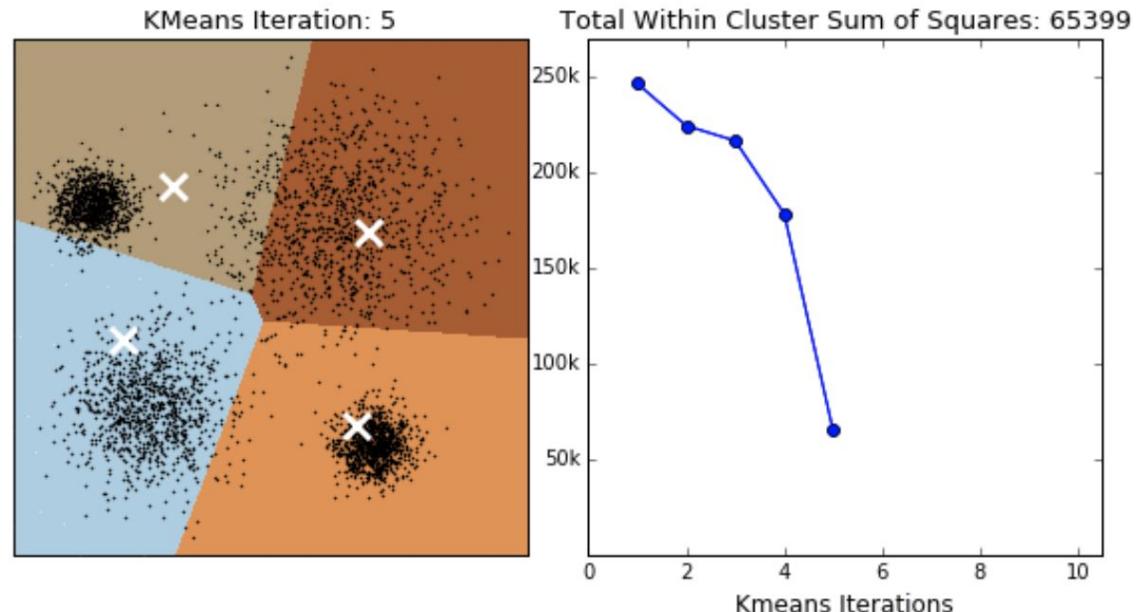
**No garantiza mínimos globales.**

$$\min_{C_1, \dots, C_K} \left\{ \sum_{k=1}^K \text{WCV}(C_k) \right\}$$

## Complejidad

(n = #instancias, d = #atributos, iter = #iteraciones)

- **Tiempo:**  $O(\text{iter} * n * K * d)$
- **Espacio:**  $O((n+K) * d)$



<https://dashee87.github.io/images/kmeans.gif>

# K-Means

## Ventajas:

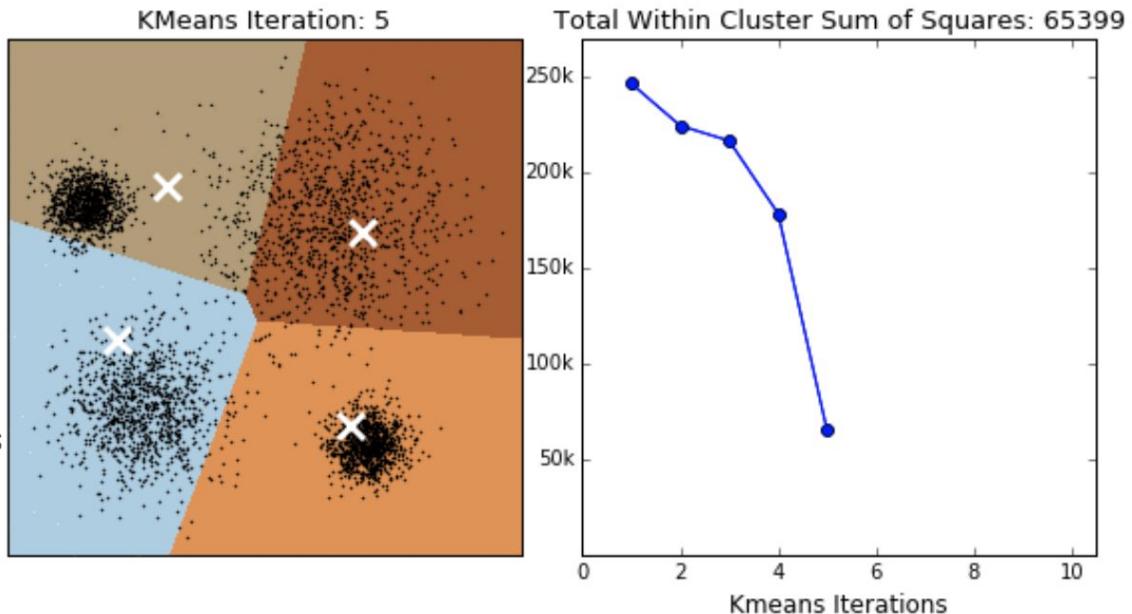
- Simple, eficiente y general.

## Desventajas:

- Hay que especificar el K.
- Sensible a ruido y outliers.
- Muy sensible a la elección de los centroides iniciales.
- Solo pueden encontrarse clusters globulares

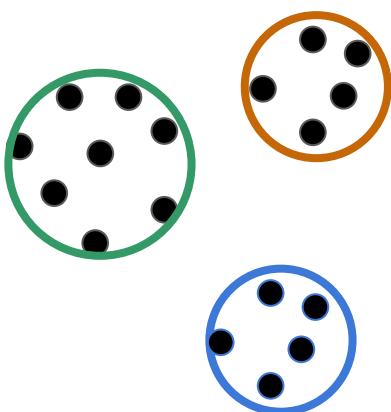
Sobre elección del K recomiendo que vean [método del codo](#) y el [método de silhouette](#):

- [Youtube.com/watch?v=AtxQ0rvd0IA](https://www.youtube.com/watch?v=AtxQ0rvd0IA)
- [https://scikit-learn.org/stable/auto\\_examples/cluster/plot\\_kmeans\\_silhouette\\_analysis.html](https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html)

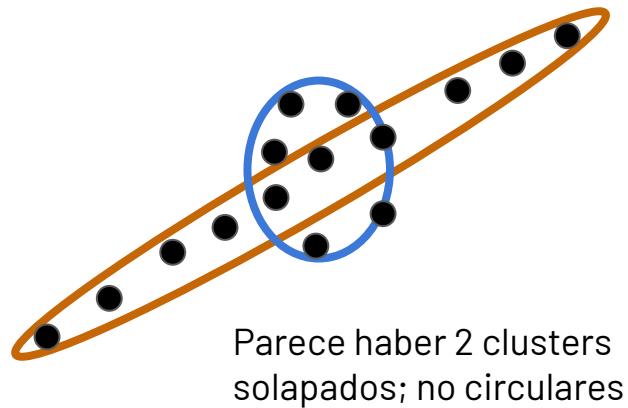


# Clustering basado en Distribuciones

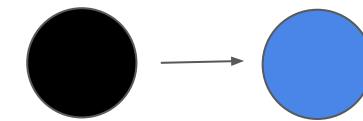
# Soft Clustering



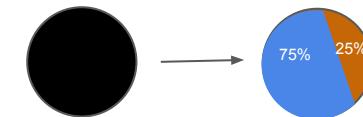
A veces, los datasets  
son más complicados



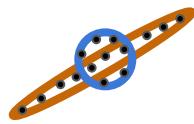
En el método anterior, los  
puntos son de un color u  
otro.



Aquí tendría más sentido  
hablar de **proporciones o  
incertidumbre**.



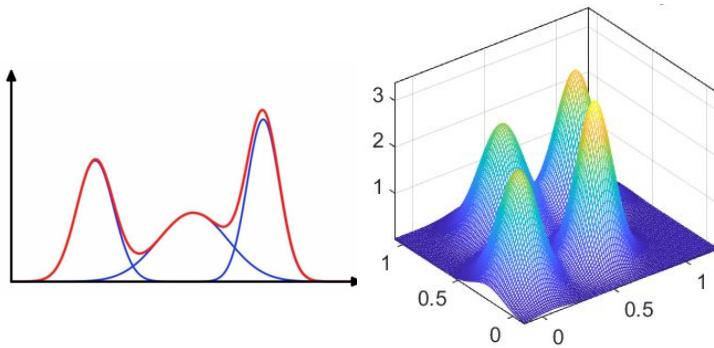
Una técnica que permite modelar estas características: **Mezclas de Gaussianas (GMMs)**



# Modelo de Mezclas Gaussianas

## Mezclas de Gaussianas (GMMs)

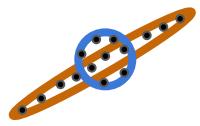
Una distribución en donde la PDF se construye como una combinación lineal de las PDFs de distintas Gaussianas.



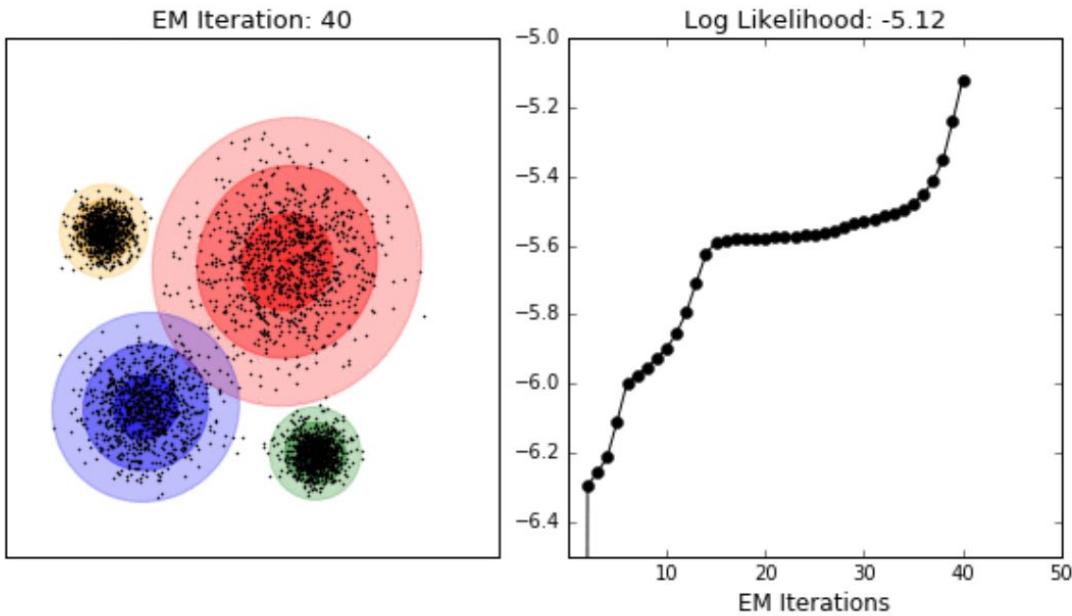
## Ventajas de ajustar datos utilizando GMMs

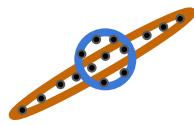
Al ajustar una mezcla de Gaussianas a los datos, resulta en un modelo probabilístico que permite:

- + Muestreo de nuevos datos (ej: imputación de valores faltantes)
- + Comparación entre datasets (ej: train vs test)  $p(x) < \epsilon \rightarrow$  anomalía
- + Detección de anomalías:
- + **Clustering**
- + ETC



# Modelo de Mezclas Gaussianas

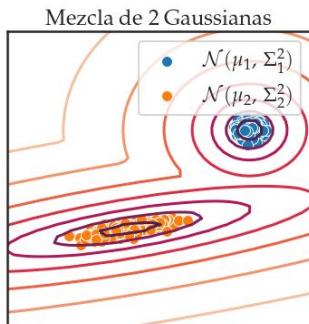




# Modelo de Mezclas Gaussianas

## Proceso generativo de datos con GMMs

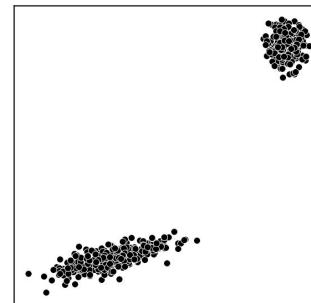
- Supongamos que tenemos **K** clusters.
- Supongamos también que conocemos la densidad de probabilidad de cada cluster.



Podemos muestrear datos de la siguiente manera:

- 1) Elegir un cluster al azar  $z \in \{1, 2, \dots, K\}$  con probabilidad  $\pi_1, \dots, \pi_K$
- 2) Muestrear un punto de la distribución normal correspondiente al cluster  $z$  con media  $\mu_z$  y covarianza  $\Sigma_z$ .

Entender el proceso generativo generalmente ayuda a pensar la ingeniería reversa del método.



## El problema

¿Qué sucede si no conocemos estas distribuciones (ni sus pesos)?

¿Podemos estimar los parámetros del modelo a partir de datos?

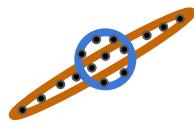
## Parámetros del modelo

- $\mu_1 \dots \mu_K$
- $\Sigma_1 \dots \Sigma_K$
- $\pi_1, \dots, \pi_K$

## Clustering

Luego, sólo bastaría con medir la pertenencia de cada punto a cada cluster

**Ejercicio:** ¿Cómo ajustarían GMMs a los datos? 29



# Modelo de Mezclas Gaussianas

Algoritmo "Expectation Maximization" (EM)

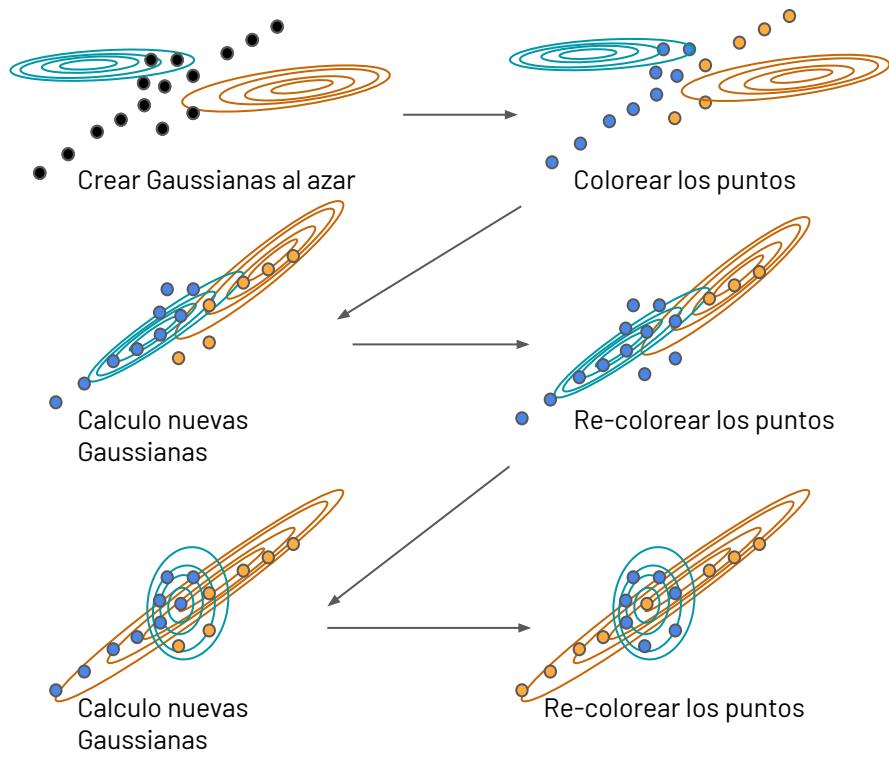
¿Cómo ajustar una GMM a datos?

*"An elegant and powerful method for finding maximum likelihood solutions for models with latent variables is called the expectation-maximization algorithm, or EM algorithm (Dempster et al., 1977; McLachlan and Krishnan, 1997)."*

– Pág 435, Bishop

Algoritmo (sin formalidades por ahora)

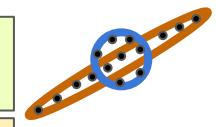
- Empezar con una cantidad fija de **Gaussianas al azar**.
- Mientras no se cumpla un **criterio de detención**:
  - **Paso E:** Colorear puntos de acuerdo a las Gaussianas.
  - **Paso M:** Ajustar Gaussianas de acuerdo a los puntos coloreados.



# Modelo de Mezclas Gaussianas

Algoritmo "Expectation Maximization" (EM)

Paso E: Colorear puntos de acuerdo a las Gaussianas.



Paso M: Ajustar Gaussianas de acuerdo a los puntos coloreados.

Paso M

¿Cómo construir una gaussiana a partir de puntos en el espacio?  
(supongamos, para los puntos **azules**)

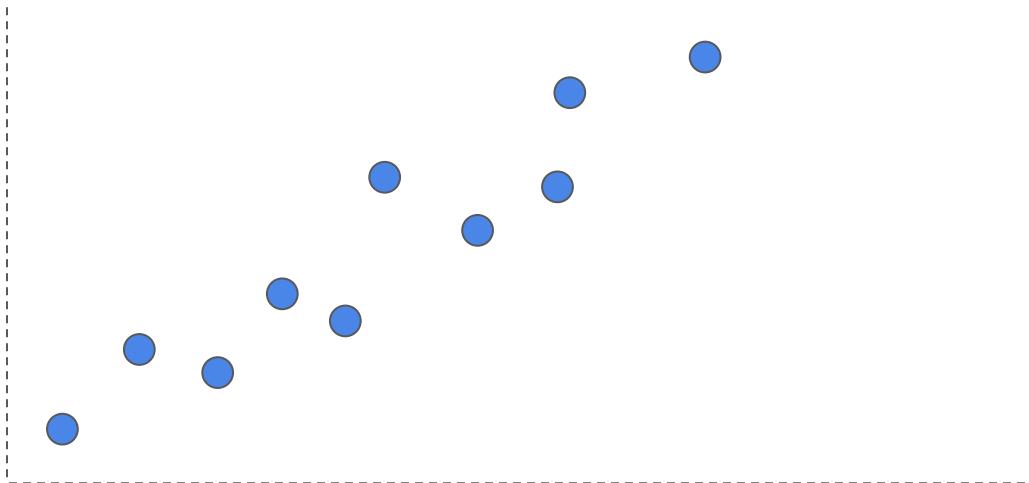
Puntos

$x_1$

$x_2$

....

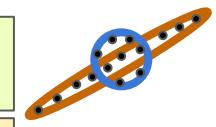
$x_n$



# Modelo de Mezclas Gaussianas

Algoritmo "Expectation Maximization" (EM)

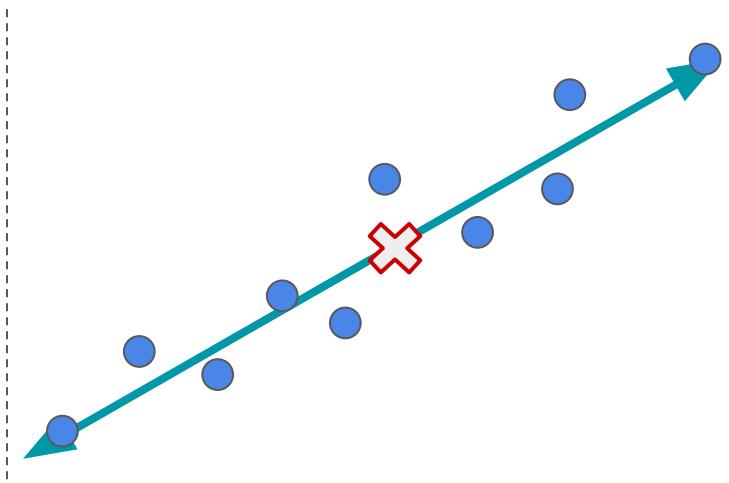
Paso E: Colorear puntos de acuerdo a las Gaussianas.



Paso M: Ajustar Gaussianas de acuerdo a los puntos coloreados.

## Paso M

¿Cómo construir una gaussiana a partir de puntos en el espacio?  
(supongamos, para los puntos azules)



Utilizamos **estimadores de máxima verosimilitud**

$$\theta_{ML} = \operatorname{argmax}_{(\theta)} p(X | \theta)$$

## Puntos

$$x_1$$

$$x_2$$

....

$$x_n$$

## Media

$$\mu_{ML} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n$$

## Covarianza

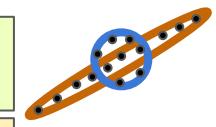
$$\Sigma_{ML} = \frac{1}{N} \sum_{n=1}^N (\mathbf{x}_n - \mu_{ML})(\mathbf{x}_n - \mu_{ML})^T$$

– Demo: Bishop: Pág 93

# Modelo de Mezclas Gaussianas

Algoritmo "Expectation Maximization" (EM)

Paso E: Colorear puntos de acuerdo a las Gaussianas.

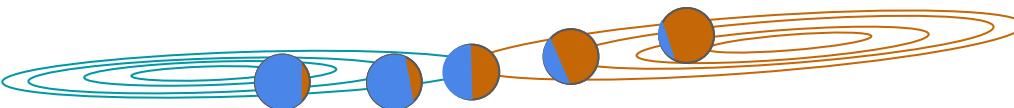


Paso M: Ajustar Gaussianas de acuerdo a los puntos coloreados.

Paso E

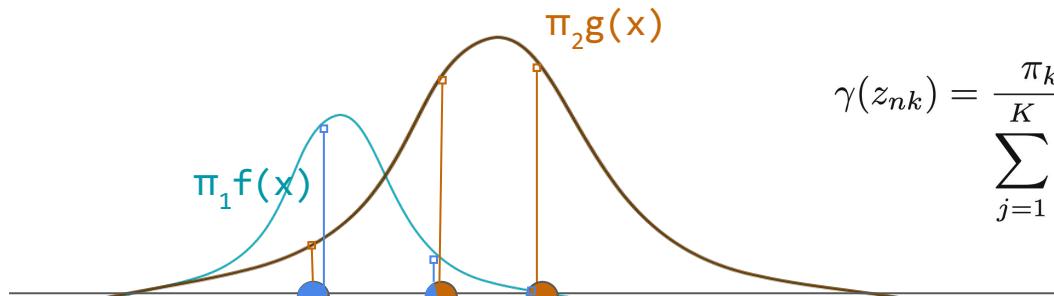
Colorear puntos:

Evaluar responsabilidades.



Para este paso suponemos conocidas las Gaussianas (que estamos queriendo estimar).

Responsabilidad  $\gamma$  (surge de aplicar Teo. Bayes a  $P(\theta | x_n)$ )



$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

$$\mathcal{N}(\underline{x} ; \underline{\mu}, \Sigma) = \frac{1}{(2\pi)^{d/2}} |\Sigma|^{-1/2} \exp \left\{ -\frac{1}{2} (\underline{x} - \underline{\mu})^T \Sigma^{-1} (\underline{x} - \underline{\mu}) \right\}$$

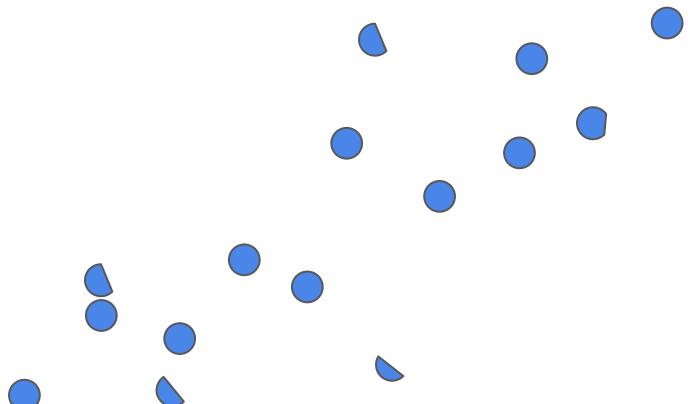
# Modelo de Mezclas Gaussianas

Algoritmo "Expectation Maximization" (EM)

## Paso M (revisitado)

¿Cómo construir una gaussiana a partir de puntos en el espacio?

**¿Qué hacer si esos puntos fueron parcialmente asignados?**



**Estimadores de máxima verosimilitud**

$$\theta_{ML} = \operatorname{argmax}_{(\theta)} p(X | \theta)$$

**Puntos**

$x_1$

$x_2$

...

$x_n$

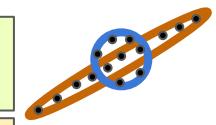
**Media**

$$\mu_{ML} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n$$

**Covarianza**

$$\Sigma_{ML} = \frac{1}{N} \sum_{n=1}^N (\mathbf{x}_n - \mu_{ML})(\mathbf{x}_n - \mu_{ML})^T$$

**Paso E:** Colorear puntos de acuerdo a las Gaussianas.

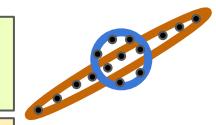


**Paso M:** Ajustar Gaussianas de acuerdo a los puntos coloreados.

# Modelo de Mezclas Gaussianas

Algoritmo "Expectation Maximization" (EM)

Paso E: Colorear puntos de acuerdo a las Gaussianas.

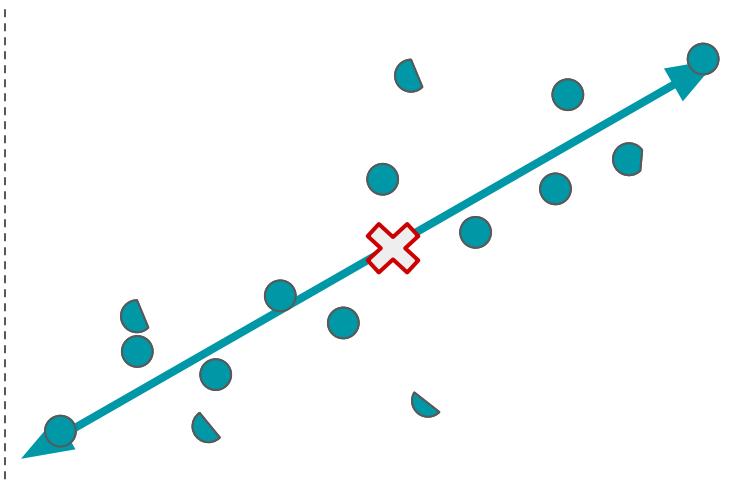


Paso M: Ajustar Gaussianas de acuerdo a los puntos coloreados.

Paso M (revisitado)

¿Cómo construir una gaussiana a partir de puntos en el espacio?

¿Qué hacer si esos puntos fueron parcialmente asignados?



Estimadores de máxima verosimilitud

$$\theta_{\text{ML}} = \underset{(\theta)}{\operatorname{argmax}} p(X | \theta)$$

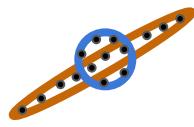
Puntos	Respons.
$x_1$	$\gamma(z_{1k})$
$x_2$	$\gamma(z_{2k})$
...	...
$x_n$	$\gamma(z_{nk})$

Media PESADA

$$\mu_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n \quad N_k = \sum_{n=1}^N \gamma(z_{nk})$$

Covarianza PESADA

$$\Sigma_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) (\mathbf{x}_n - \mu_k^{\text{new}}) (\mathbf{x}_n - \mu_k^{\text{new}})^T$$



# Modelo de Mezclas Gaussianas

Algoritmo “Expectation Maximization” (EM)

El objetivo es **maximizar la función de verosimilitud** con respecto a los parámetros (medias, covarianzas de los componentes y los coeficientes de mezcla).

1. **Inicialice** las medias, las covarianzas y los coeficientes de mezcla, y evalúe el valor inicial de log-verosimilitud.

2. **Paso E.** Evaluar las responsabilidades utilizando los valores de los parámetros actuales.

3. **Paso M.** Volver a estimar los parámetros utilizando las responsabilidades actuales

4. Utilizar la log-verosimilitud o el cambio en los parámetros para determinar **convergencia**.

Si no se cumple el criterio seleccionado, vuelve al paso 2

$$\ln p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

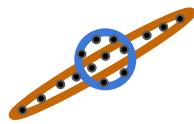
$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

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

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) (\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}) (\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})^T$$

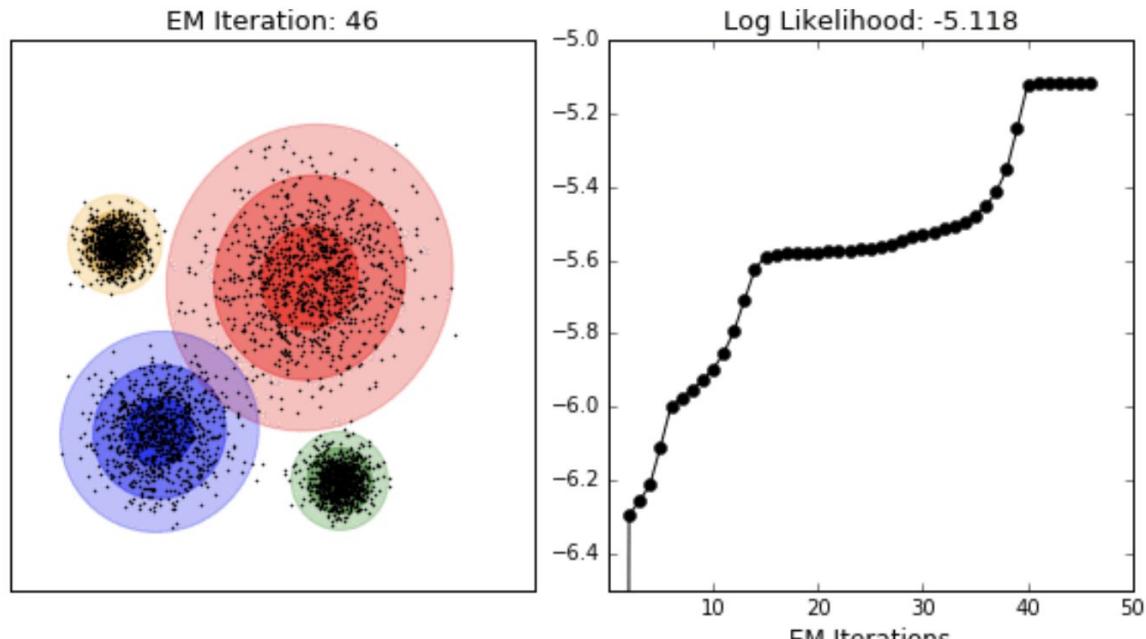
$$\pi_k^{\text{new}} = \frac{N_k}{N}$$



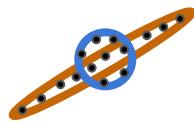
# Modelo de Mezclas Gaussianas

Convergencia del método

- Cada iteración **mejora** la verosimilitud de los datos:  
Convergencia garantizada
- El método encuentra un **mínimo local**, por lo tanto habrá que correr varias veces y quedarse con el que obtenga mayor log-likelihood.
- También es recomendable **inicializar** los valores de las medias utilizando K-means (por cuestiones computacionales que veremos a continuación)



[https://dashee87.github.io/images/em\\_only.gif](https://dashee87.github.io/images/em_only.gif) (Autor: David Sheehan)



# Modelo de Mezclas Gaussianas

Consideraciones prácticas

## Costo computacional

$n$  = #instancias;  $K$  = número de clusters;  $d$  = dimensión

## KMeans

Temporal:  $O(\text{iter}.n.K.d)$  Espacial:  $O((n + k) . d)$

## GMM-EM

Temporal:  $O(\text{iter}.n.K.d^3)$  Espacial:  $O(n.K.d^2)$

$O(d^3)$  se produce al **invertir la matriz de covarianza**, cuyo tamaño es  $d \times d$ :

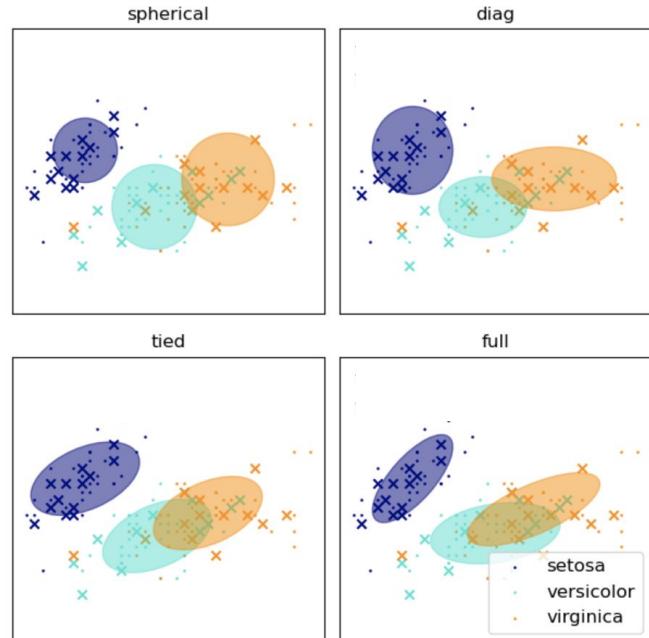
$$\ln p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = -\frac{ND}{2} \ln(2\pi) - \frac{N}{2} \ln |\boldsymbol{\Sigma}| - \frac{1}{2} \sum_{n=1}^N (\mathbf{x}_n - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}_n - \boldsymbol{\mu}).$$

Comparado con K-means, este algoritmo se vuelve **prohibitivo** para dimensiones altas.

Por ello, se utilizan simplificaciones para la matriz de covarianza:

API en Scikit-learn:

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



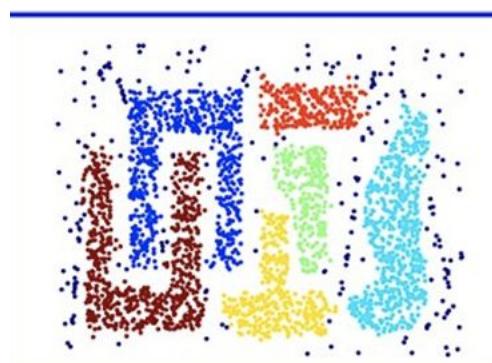
# Modelo de Mezclas Gaussianas

## Conclusiones

- Las **GMMs** son un modelo generativo general que puede utilizarse para clustering.
- **EM** proporciona una solución iterativa a la estimación de máxima verosimilitud con variables latentes.
- El modelo y el algoritmo asociado puede ser utilizado para diversos problemas más allá de Clustering.
- El método converge a un **mínimo local**.
- Escala bien con la **cantidad de datos**, pero no así con su **dimensión**.

# Clustering basado en Densidades

Volvemos 20:15



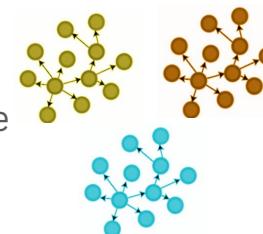
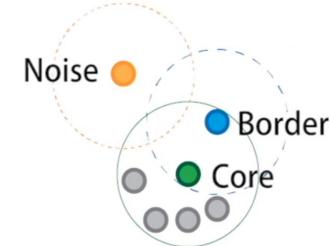
# DBSCAN

Density-based spatial clustering of applications with noise

## Algoritmo

HiperParámetros:  $m$  (*min\_samples*),  $\varepsilon$  (*epsilon*).

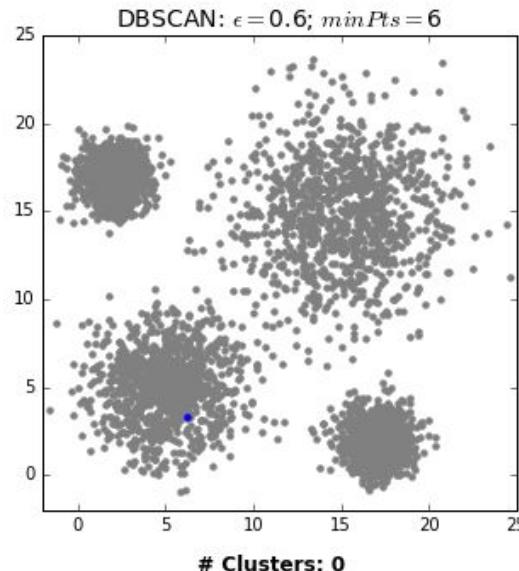
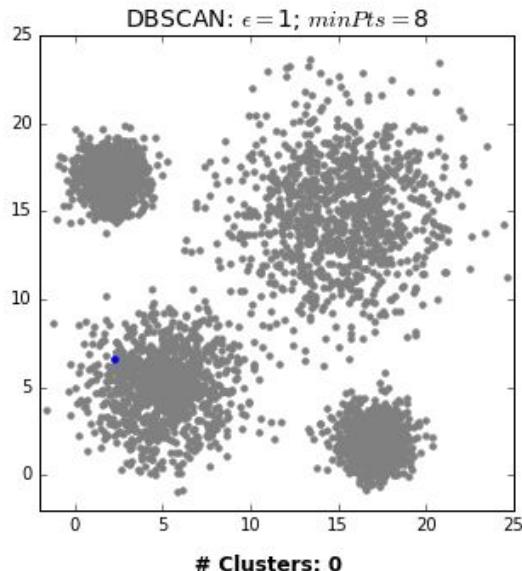
1. Etiquetar a todo punto:
  - a. como **Core** si tiene al menos  $m$  vecinos a menos de  $\varepsilon$  distancia.
  - b. **Non-Core** si no.
2. Conectamos a **cada core point con sus vecinos** formando un grafo.
3. A partir de cada punto **Core** escanear a qué puntos puede llegar a través de saltos en este grafo.
4. Clasificamos a los **Non-Core** como:
  - a. **Border**: si quedó conectado en el grafo (es decir, si tenía al menos un core en su vecindario y no es core).
  - b. **Noise**: al resto.
5. Cada componente conexa se corresponde a un cluster.
6. No asignamos ningún cluster a los puntos **Noise**.



Veamos juntos este StatQuest.  
<https://youtu.be/RDZUdRSD0ok?t=187>  
Luego vean este gif también:  
[https://dashee87.github.io/images/DBSCAN\\_tutorial.gif](https://dashee87.github.io/images/DBSCAN_tutorial.gif)

# DBSCAN

Density-based spatial clustering of applications with noise



[https://dashee87.github.io/images/DBSCAN\\_search.gif](https://dashee87.github.io/images/DBSCAN_search.gif)

# DBSCAN

Density-based spatial clustering of applications with noise

**Complejidad temporal:**  $O(n^2d)$  en el peor caso.

$O(d * n \log n)$  si se usan k-d trees.

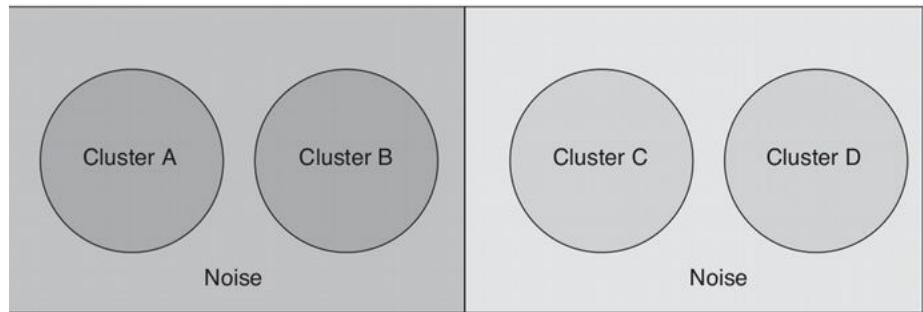
**Complejidad espacial:**  $O(n)$

## Ventajas:

- No es necesario especificar K.
- Puede encontrar clusters de formas arbitrarias.
- Robusto al ruido.

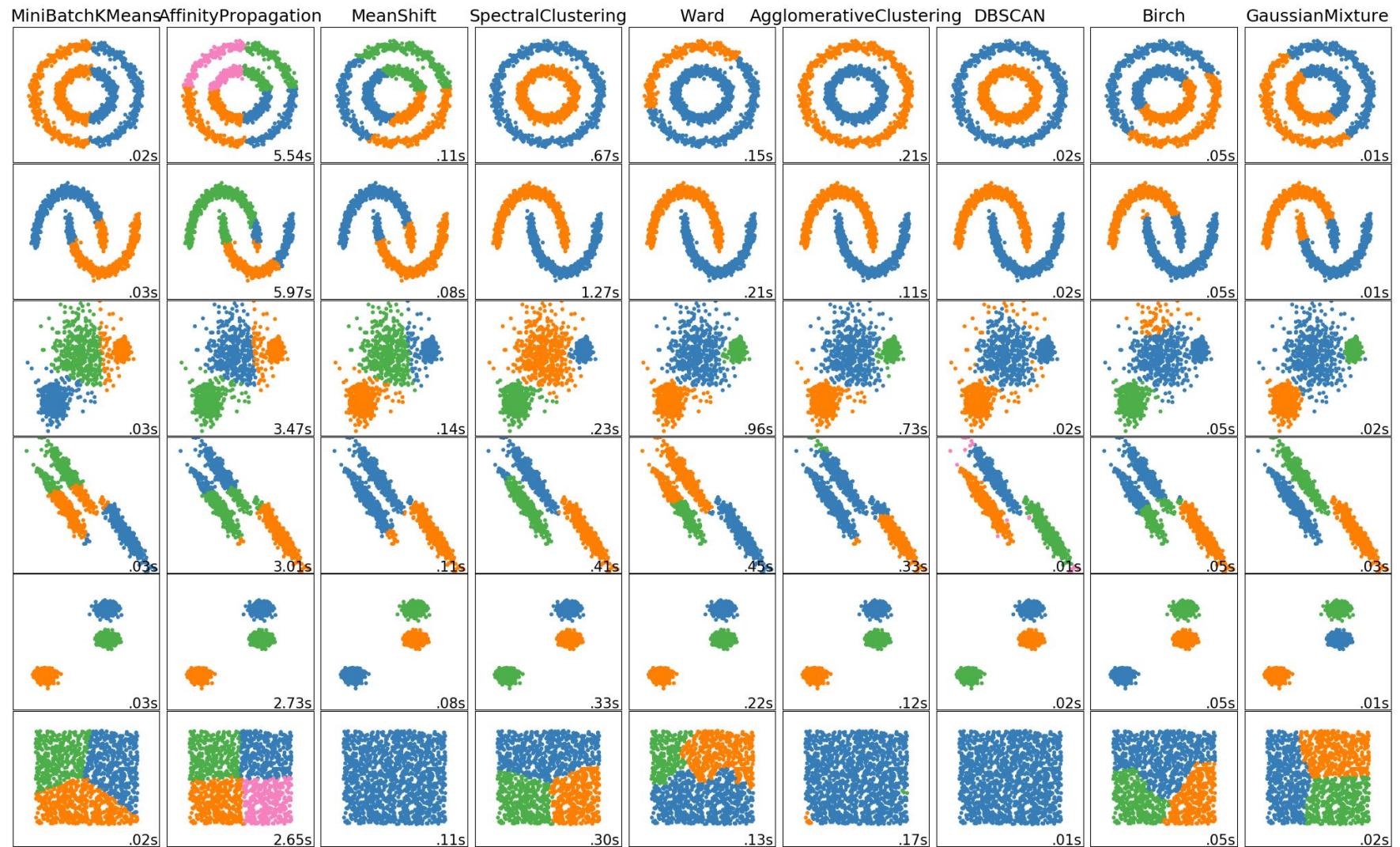
## Desventajas:

- Elegir  $m$  y  $\epsilon$  requiere conocimiento de los datos (difícil en casos de alta dimensionalidad).
- Funciona mal con datos con densidad variable.



Densidad = tonos de gris.

Ruido alrededor de A y B: misma cantidad que C y D. DBSCAN colapsa A, B y ruido que los rodea, o bien ignora C y D como si fueran ruido.



# Tarea

## Obligatorio:

- **Hastie, Tibshirani & Friedman**, "The Elements of Statistical Learning", 2nd ed, Springer, 2009.
  - Sección 14.3.4, 14.3.5, 14.3.6, 14.3.7, 14.3.12 (hasta página 528)
- **Bishop**, "Pattern Recognition and Machine Learning".
  - Sección 2.3.9 Mixtures of Gaussians (esto es sobre la distribución en sí misma)
  - Sección 9.2 Mixtures of Gaussians (ya en el contexto de clustering)
- Notebook: Tendrán que implementar K-Means, EM para GMMS y Agglomerative Clustering.

## Opcional:

- Sección **12.4** del **ISLR** "Clustering Methods"
- **Tan, Steinbach & Kumar**, "Introduction to Data Mining", Capítulo 8: "Cluster Analysis"
- **Bishop**, "Pattern Recognition and Machine Learning". Sección 9.3 "An Alternative View of EM"