

APRENDIZAJE NO SUPERVISADO

APRENDIZAJE DE MAQUINA I - CEIA - FIUBA

Dr. Ing. Facundo Adrián Lucianna

Dr. Ing. Álvaro Gabriel Pizá

REPASO CLASE ANTERIOR

- Métodos de ensamble
- Modelos que votan
- Bagging y bosques aleatorios
- Boosting
- XGBoost

MÉTODOS DE ENSAMBLE

Si llevamos esta idea al campo del aprendizaje automático, al promediar la salida de un conjunto de predictores es altamente probable que obtengamos mejores métricas que considerando la salida de uno solo de ellos.

Dado que un conjunto de predictores se conoce como **ensamble**, estos métodos comúnmente se llaman **métodos de ensamble**.

Por ejemplo, podríamos entrenar un grupo de árboles de decisión, cada uno en una parte diferente del conjunto de entrenamiento, generar predicciones con cada uno de los árboles individuales y luego predecir la clase que tenga mayor cantidad de votos.

MODELOS QUE VOTAN

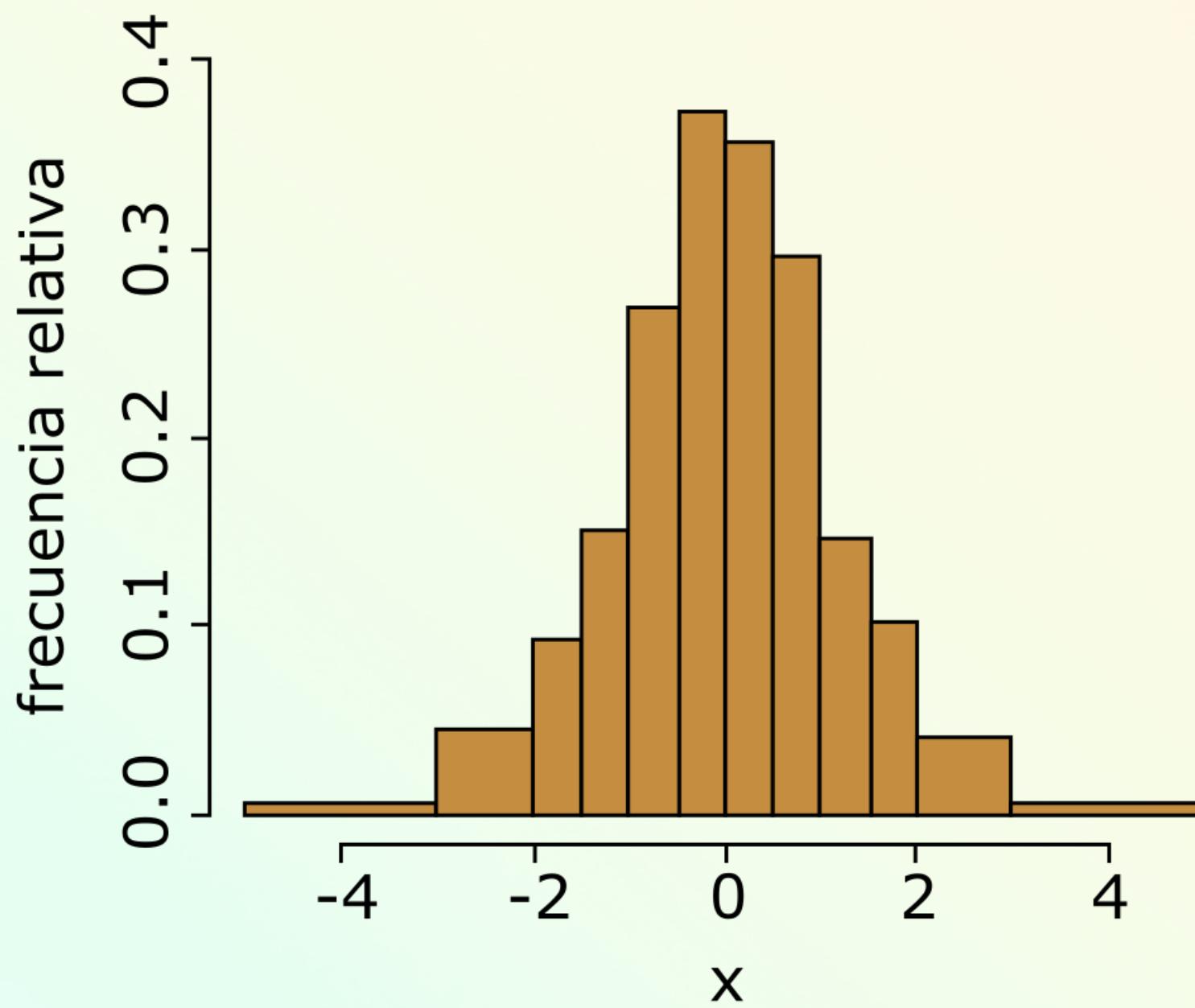
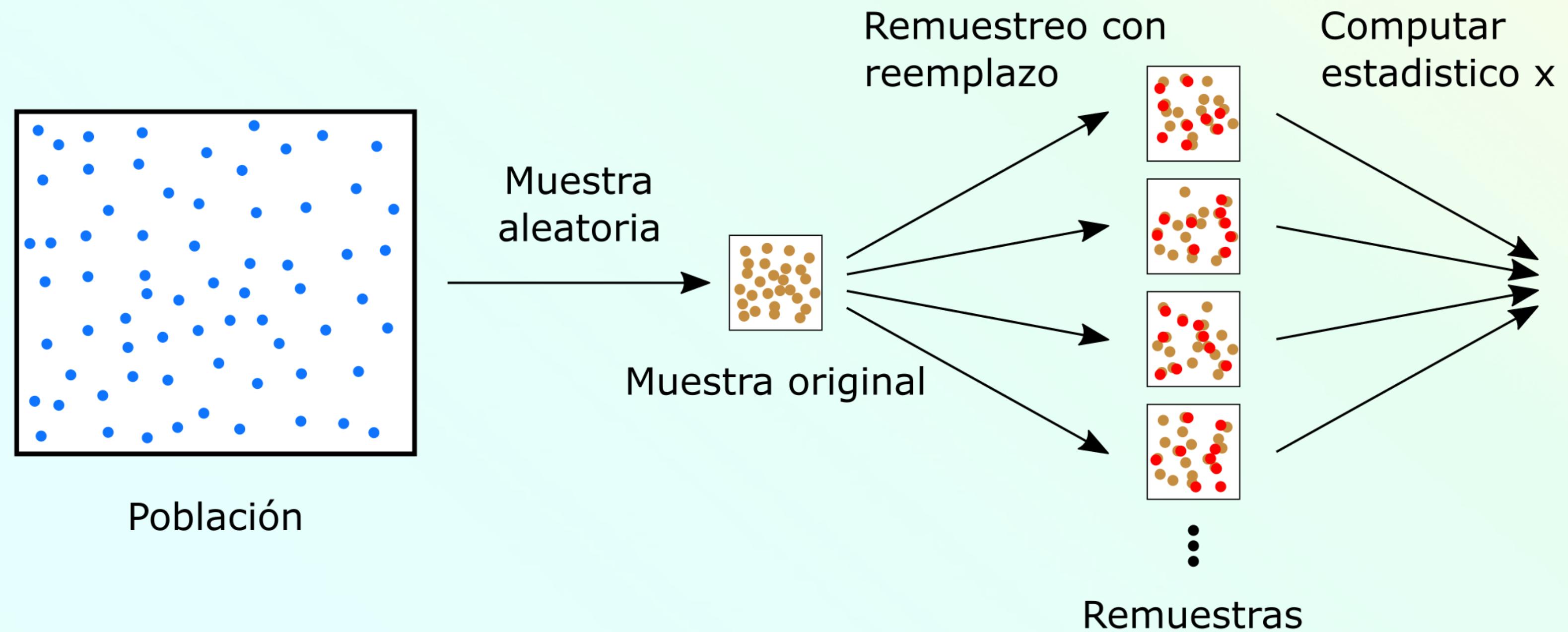
Hard voting classifiers: Por ejemplo, podríamos entrenar un grupo de árboles de decisión, cada uno en una parte diferente del conjunto de entrenamiento, generar predicciones con cada árbol individual y luego predecir la clase que tenga mayor cantidad de votos.

Soft voting classifiers: En este caso, si los modelos entrenados tienen la posibilidad de estimar la probabilidad por clase, lo que se hace es calcular la probabilidad promedio de cada clase sobre la cantidad total de modelos.

Voting regressor: En este caso, dado que los modelos pueden dar una salida continua, se puede obtener la salida promedio (u otra medida de posición central) de los modelos.

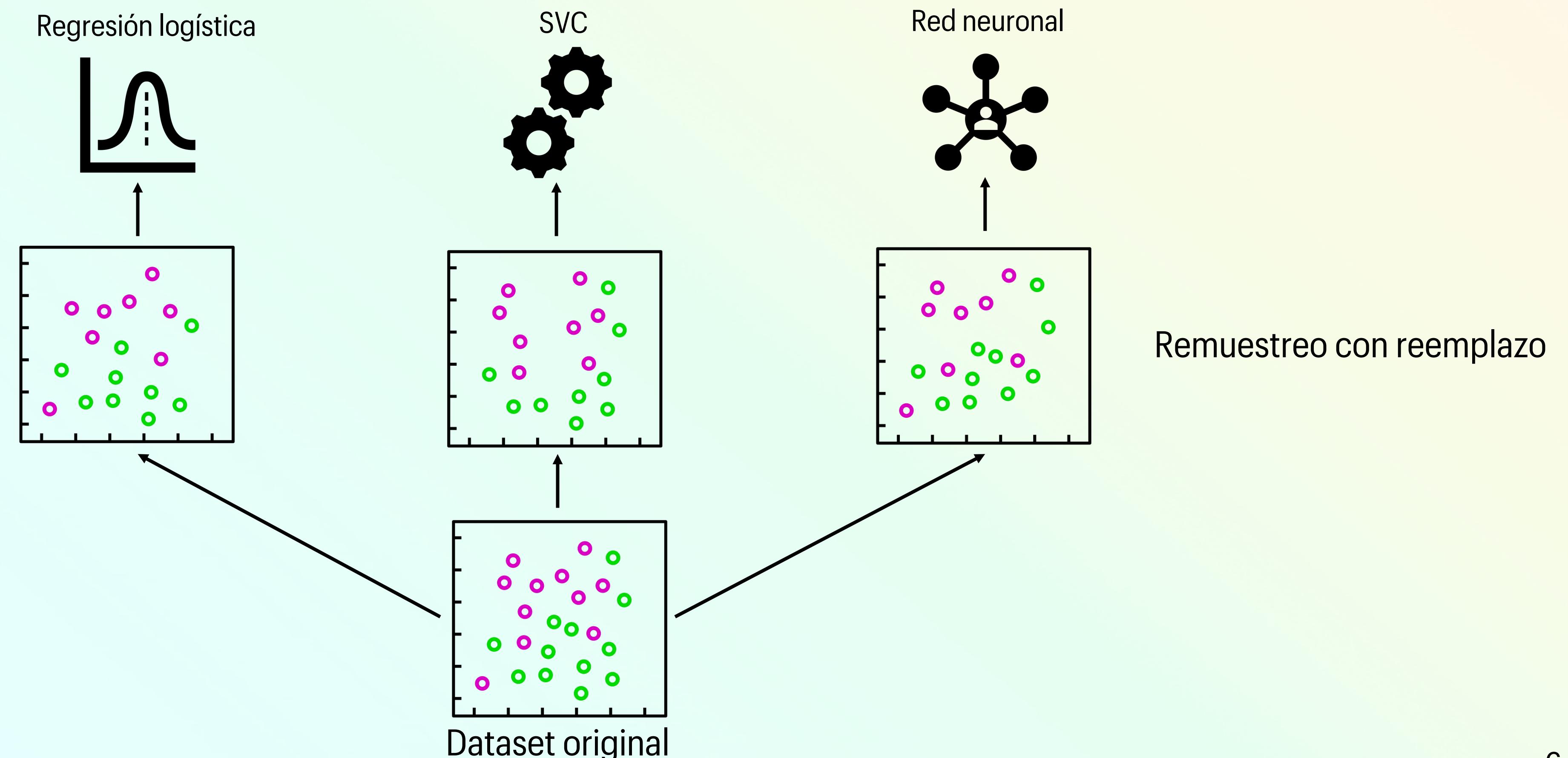
BAGGING

En estadística el muestreo con reemplazo se llama **Bootstrapping**



BAGGING

Para realizar **Bagging (bootstrap aggregating)**, buscamos modelos que tengan mucha varianza y poco sesgo.



BOSQUES ALEATORIOS

Los **bosques aleatorios** son una mejora de los obtenidos mediante **bagging** únicamente.

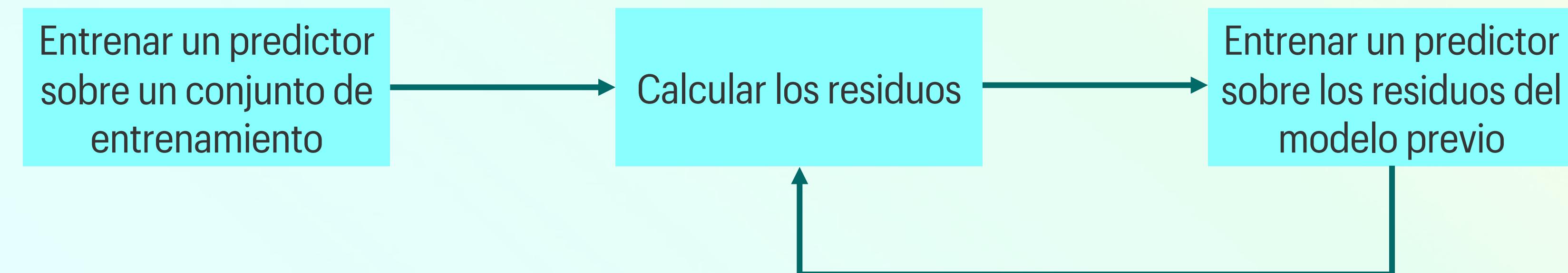
Los bosques aleatorios hacen lo mismo que mediante bagging, pero además se usa una **cantidad aleatoria de atributos**. El valor de cuantos atributos a usar se elige aproximadamente la raíz cuadrada de la cantidad de atributos totales.

Por ejemplo, si tenemos $p=13$ atributos, se usarán $m=4$, y en cada árbol será una combinación al azar diferente.

BOOSTING

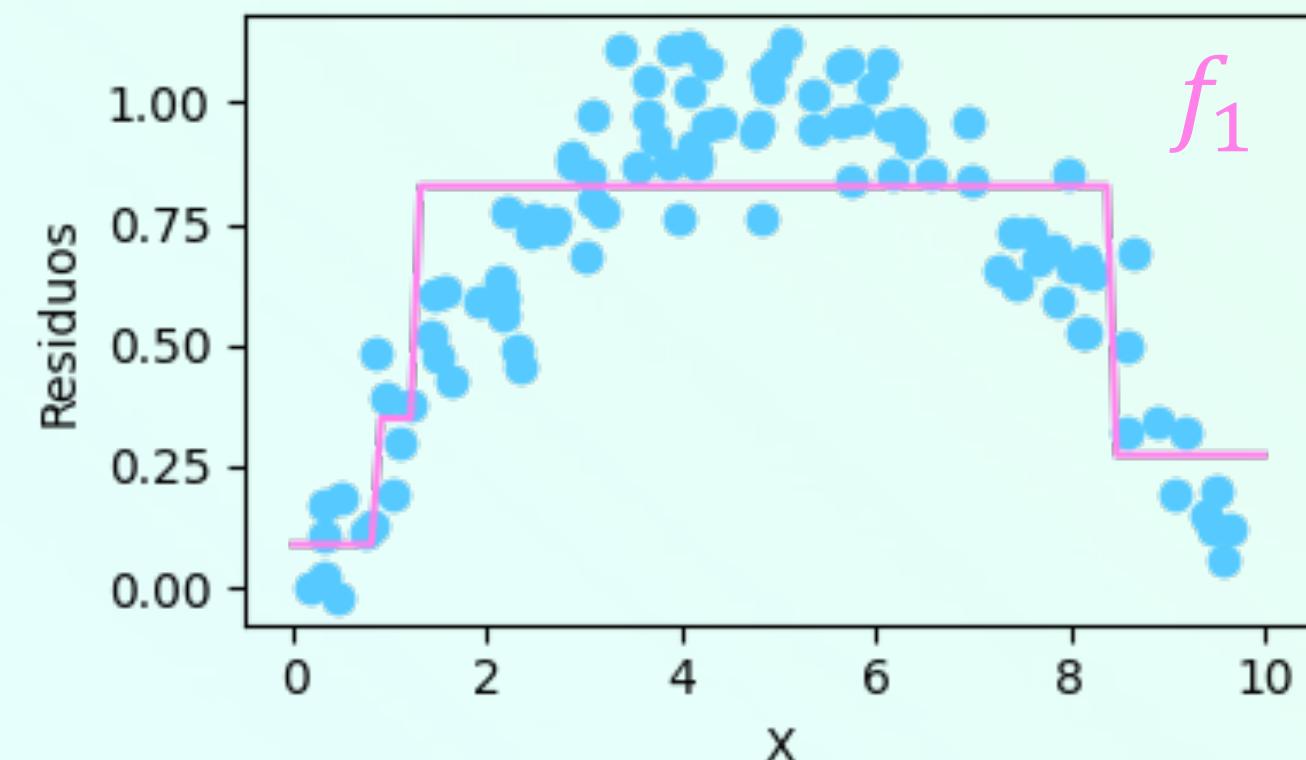
Este tipo de arquitecturas son ensambles que al igual que los vistos anteriormente, combinan varios predictores para hacer una mejor predicción. La principal diferencia es que en las arquitecturas que aplican **Boosting**, el entrenamiento de los predictores se da de forma secuencial.

Cada nuevo predictor que se agrega al ensamble intenta corregir a su predecesor.

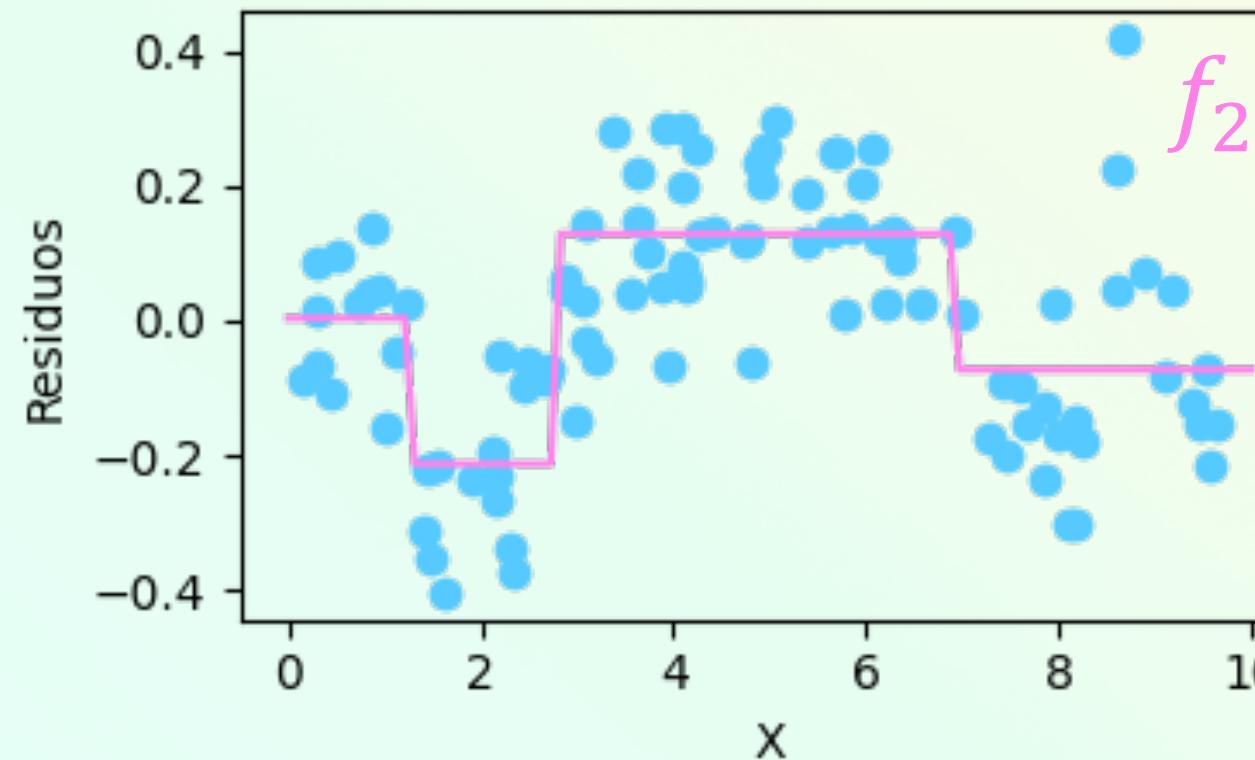


BOOSTING

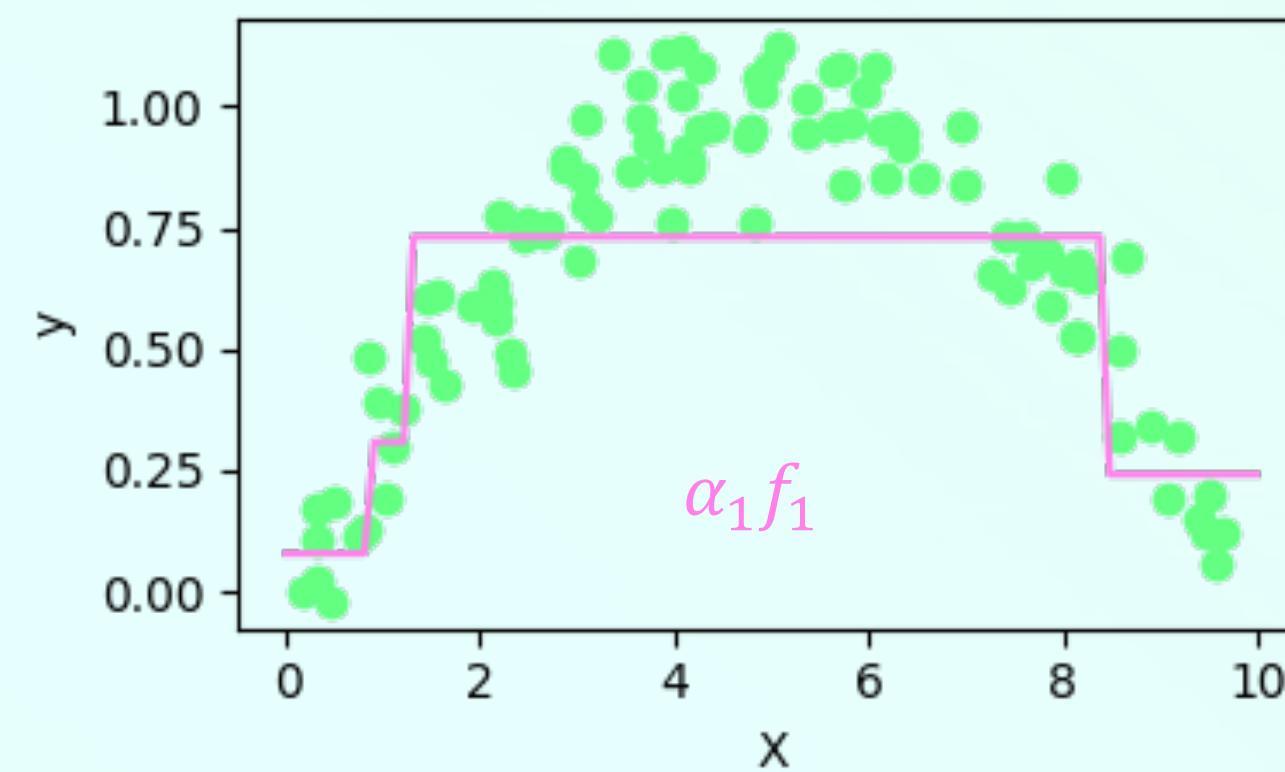
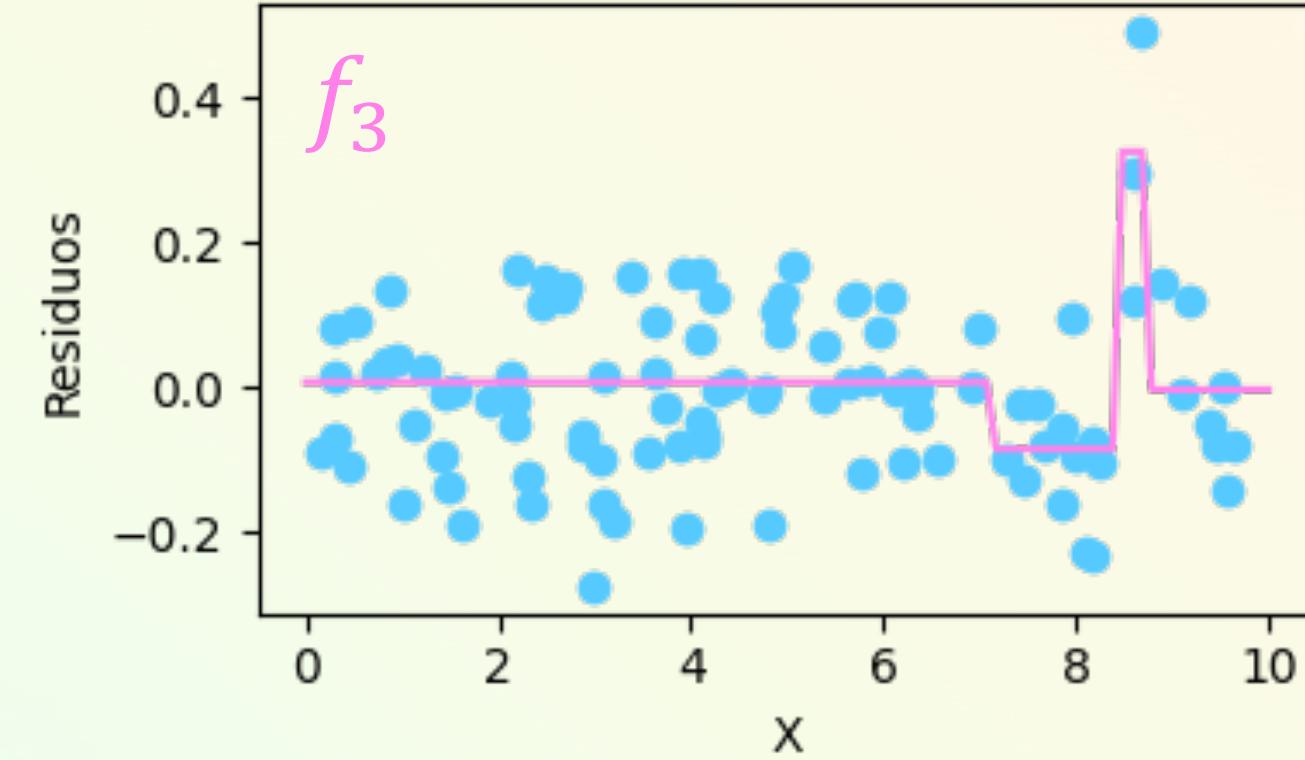
Iteración 1



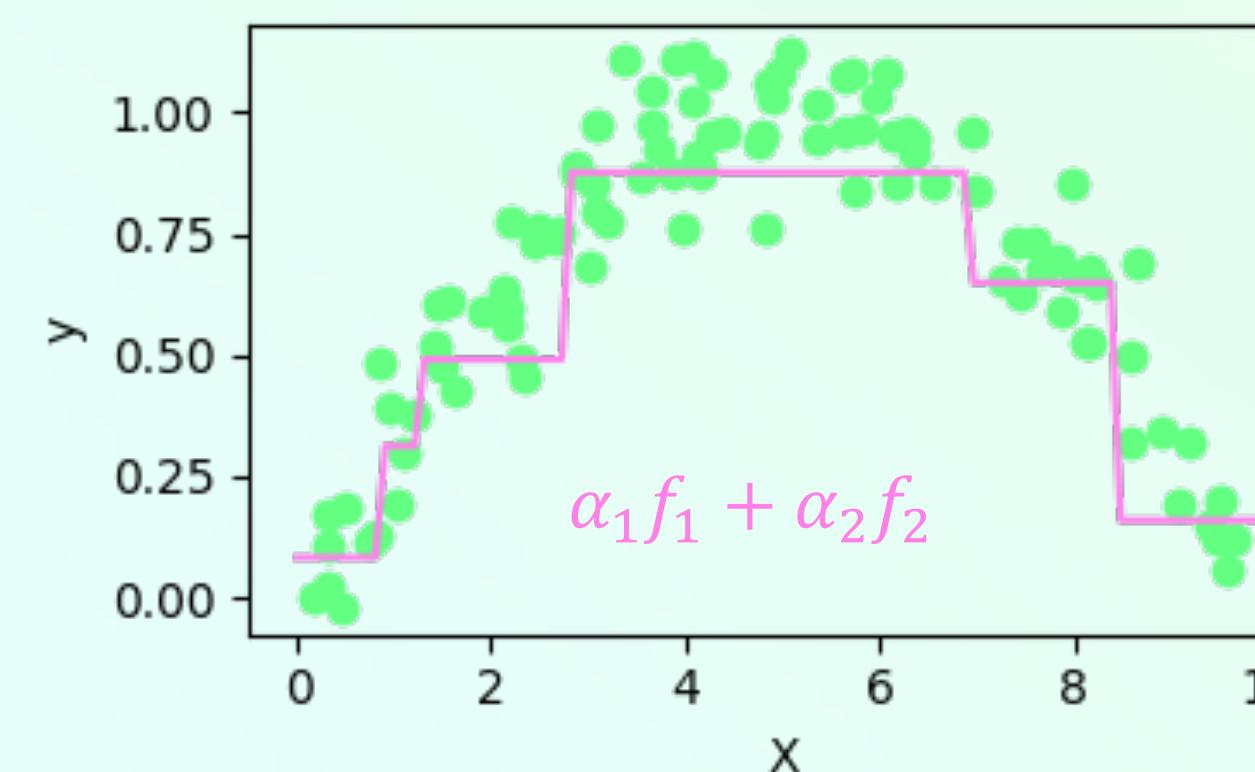
Iteración 2



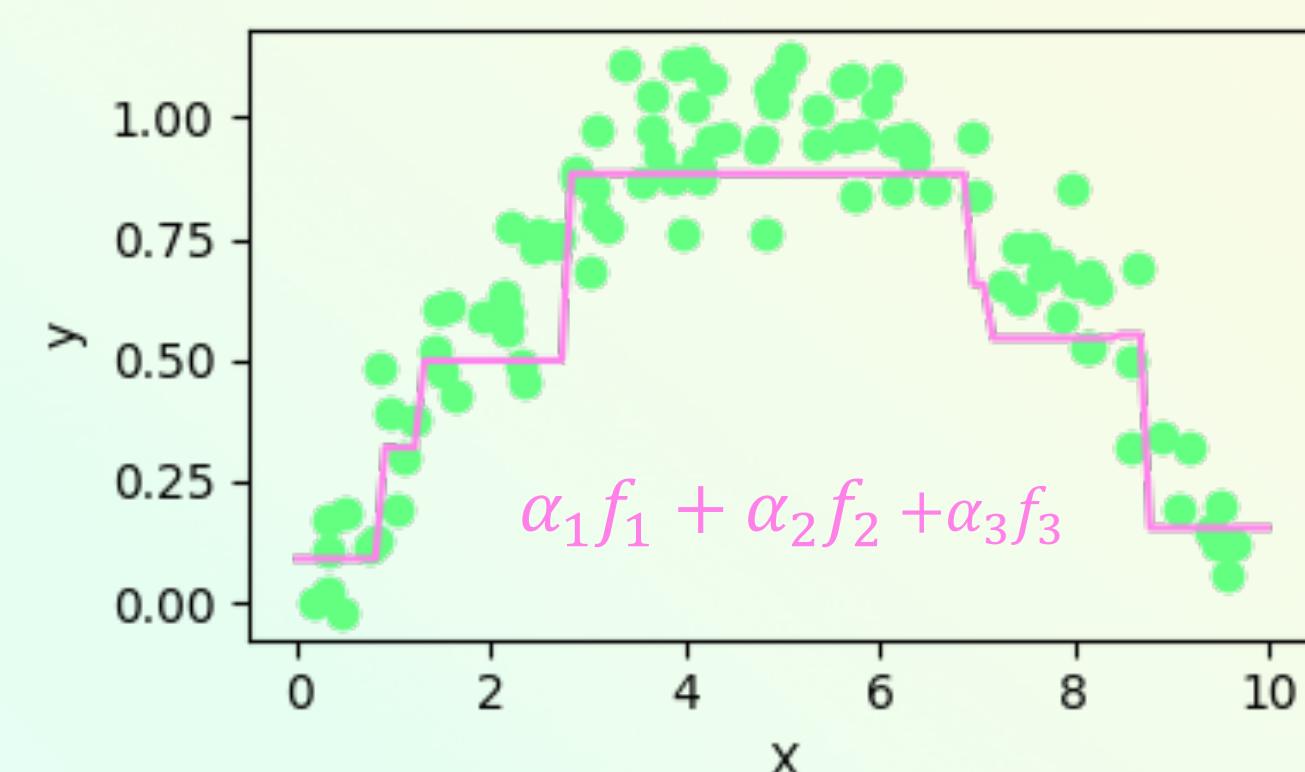
Iteración 3



Salida de ensamble: $F = \alpha_1 f_1$



Salida de ensamble: $F = \alpha_1 f_1 + \alpha_2 f_2$



Salida de ensamble: $F = \alpha_1 f_1 + \alpha_2 f_2 + \alpha_3 f_3$

XGBOOST

El software más utilizado en Boosting es [**XGBoost**](#), una implementación del Boosting de gradiente estocástico desarrollado por Tianqi Chen y Carlos Guestrin en la Universidad de Washington.

En AdaBoost se usa una función exponencial para ponderar cada residuo, con XGBoost se utiliza al gradiente de la función de costo y con ello se pondera a los residuos y los weak learners.

APRENDIZAJE NO SUPERVISADO

APRENDIZAJE NO SUPERVISADO

Como vimos en la clase 1, en aprendizaje no supervisado tenemos un dataset de n observaciones con p atributos, pero no tenemos una variable Y de respuesta.

Esto nos genera un desafío...

APRENDIZAJE NO SUPERVISADO

Como vimos en la clase 1, en aprendizaje no supervisado tenemos un dataset de n observaciones con p atributos, pero no tenemos una variable Y de respuesta.

Esto nos genera un desafío...

No tenemos un **objetivo simple** para el análisis.

Es difícil evaluar los resultados obtenidos, ya que no existe un mecanismo aceptado para realizar una **validación cruzada** o **validar** los resultados en un conjunto de datos independiente.

APRENDIZAJE NO SUPERVISADO

Como vimos en la clase 1, en aprendizaje no supervisado tenemos un dataset de n observaciones con p atributos, pero no tenemos una variable Y de respuesta.

Esto nos genera un desafío...

No tenemos un **objetivo simple** para el análisis.

Es difícil evaluar los resultados obtenidos, ya que no existe un mecanismo aceptado para realizar una **validación cruzada** o **validar** los resultados en un conjunto de datos independiente.

Esto se debe justamente a que no tenemos con que comparar.

APRENDIZAJE NO SUPERVISADO

Entonces, para que usaríamos este tipo de aprendizaje?

Lo que buscamos en aprendizaje no supervisado es descubrir **una estructura sobre la base del conjunto de datos**. A diferencia de los problemas de aprendizaje supervisado, que buscamos predecir un resultado.

En un caso particular, es cuando queremos encontrar en los datos conjuntos que responden de una forma similar, es decir agrupamientos que a priori no son evidentes.

Por ejemplo, un sitio de compras podría intentar identificar grupos de compradores con historiales de navegación y compras similares, así como artículos que sean de particular interés para los compradores dentro de cada grupo.

CLUSTERING

CLUSTERING

Clustering o agrupación se refiere a un conjunto amplio de técnicas para encontrar subgrupos o clusters en un dataset.

Cuando agrupamos las observaciones, buscamos dividirlas en grupos distintos de modo que las observaciones dentro de cada grupo sean **similares entre sí**, mientras que las observaciones en otro grupo sean bastante **diferentes** de las del primero.

Pero, hay que definir que es similar o que es diferente. Esto suele ser una consideración específica de un dominio que debe realizarse basándose en el conocimiento de los datos que se están estudiando.

CLUSTERING

Vamos a ver dos algoritmos de clustering de los más famosos de los múltiples que existen:

- **K-Means**
- **Modelo de Mixtura Gaussiana**

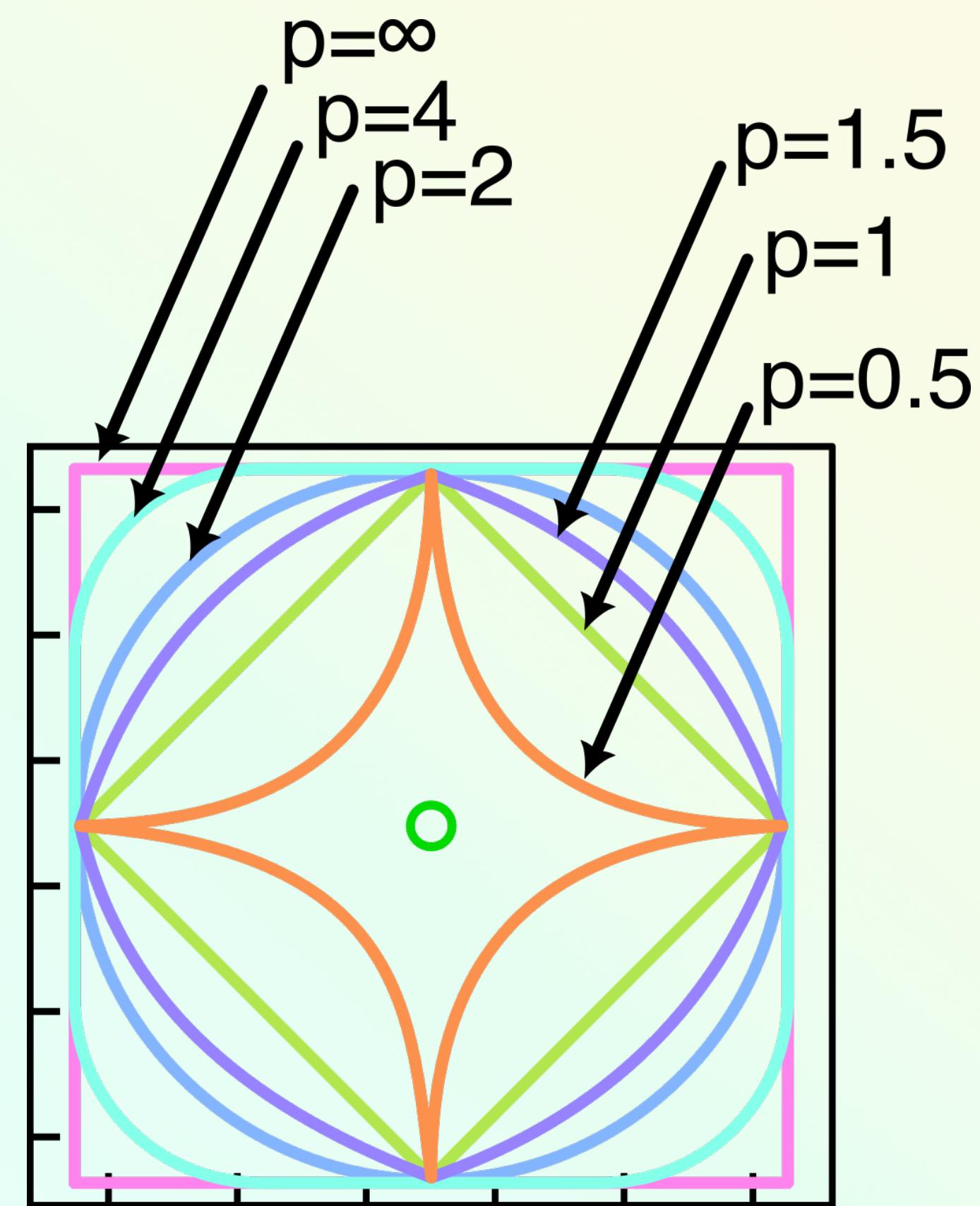
MEDIDAS DE DISTANCIA O SIMILITUD

MEDIDAS DE DISTANCIA O SIMILITUD

En **kNN** vimos las medidas de distancias, en clustering, se utilizan las mismas medidas para indicar que tan diferente o similares son.

- **Distancia de Manhattan**
- **Distancia euclidiana**
- **Distancia de Chebyshev**
- **Distancia de Minkowski**

$$d_p(a, b) = \left(\sum_{i=1}^n (a_i - b_i)^p \right)^{1/p}$$



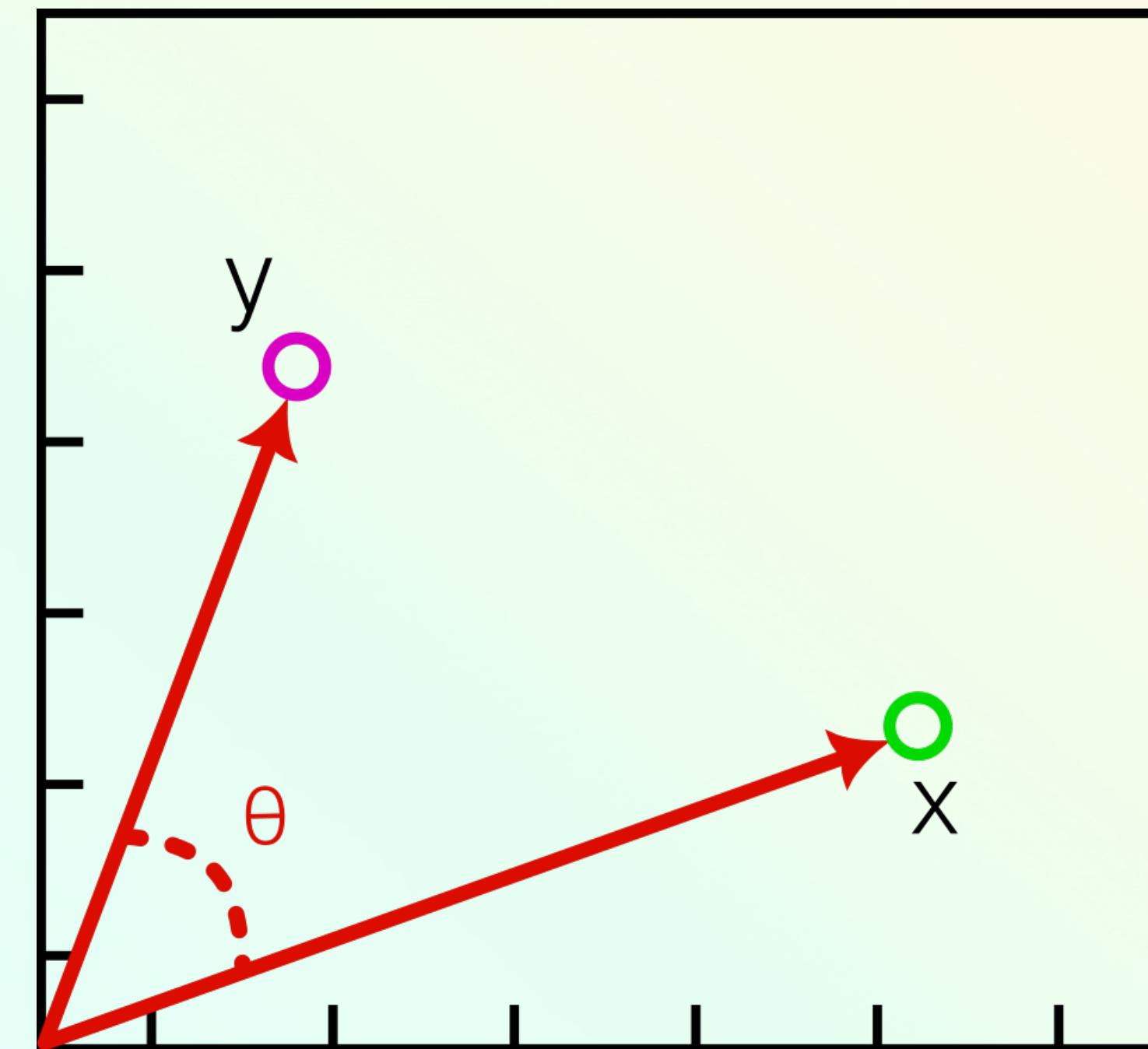
MEDIDAS DE DISTANCIA O SIMILITUD

En **kNN** vimos las medidas de distancias, en clustering, se utilizan las mismas medidas para indicar que tan diferente o similares son.

- **Distancia Coseno**

$$d_c(a, b) = 1 - S_c(a, b)$$

$$d_c(a, b) = \cos(\theta) = \frac{a \cdot b}{\|a\| \|b\|}$$



MEDIDAS DE DISTANCIA O SIMILITUD

En **kNN** vimos las medidas de distancias, en clustering, se utilizan las mismas medidas para indicar que tan diferente o similares son.

- **Similitud de Jaccard**

$$J(a, b) = \frac{|a \cap b|}{|a \cup b|}$$

Asimetrico

x	1	0	1	0	0	0	1	1
y	1	0	0	0	1	0	0	1

$\left. \begin{array}{l} J=2/5 \\ d_J=3/5 \end{array} \right\}$

- **Distancia de Hamming**

x	1	0	1	0	0	0	1	1
y	1	0	0	0	1	0	0	1

$\left. \right\} d_H=3$

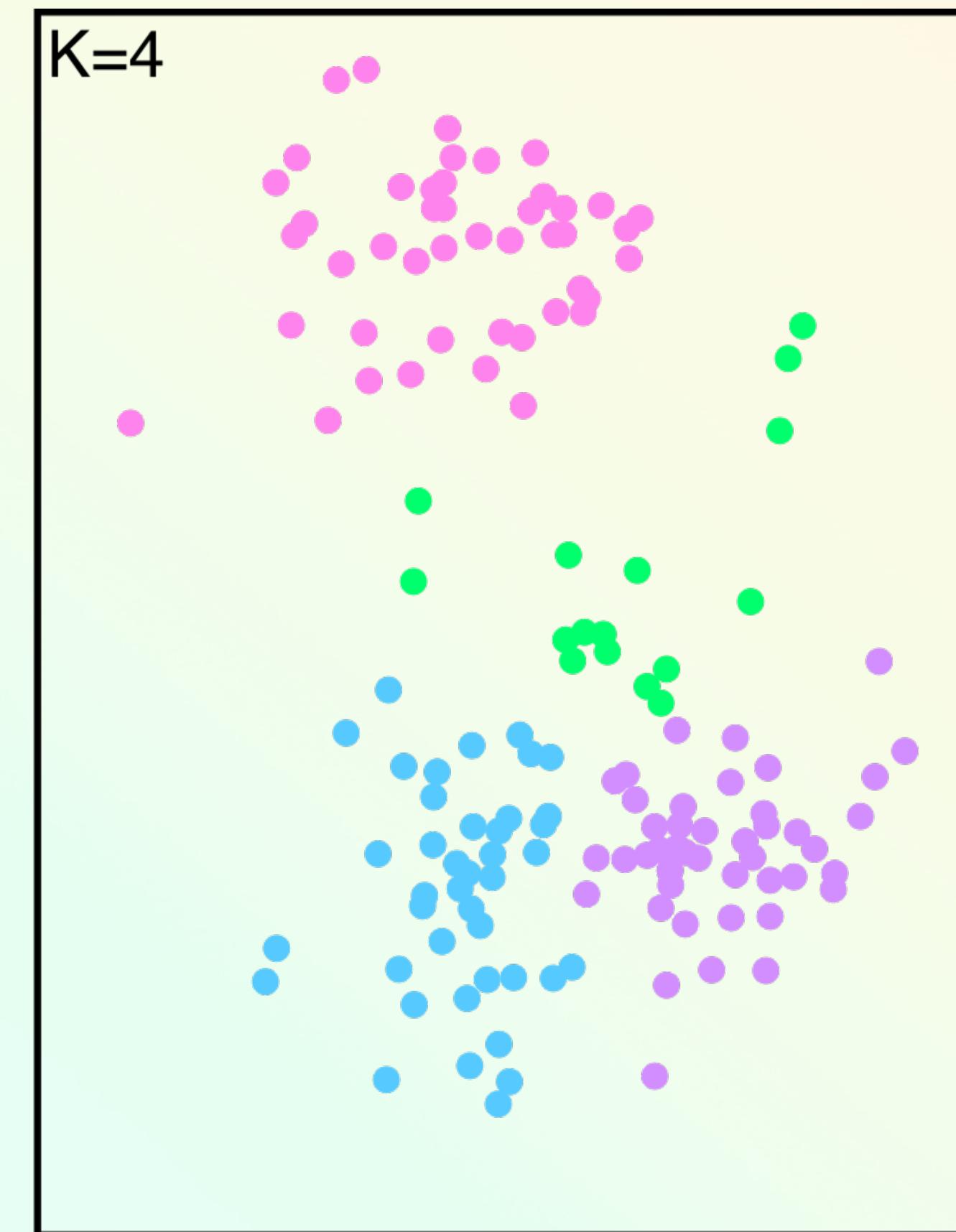
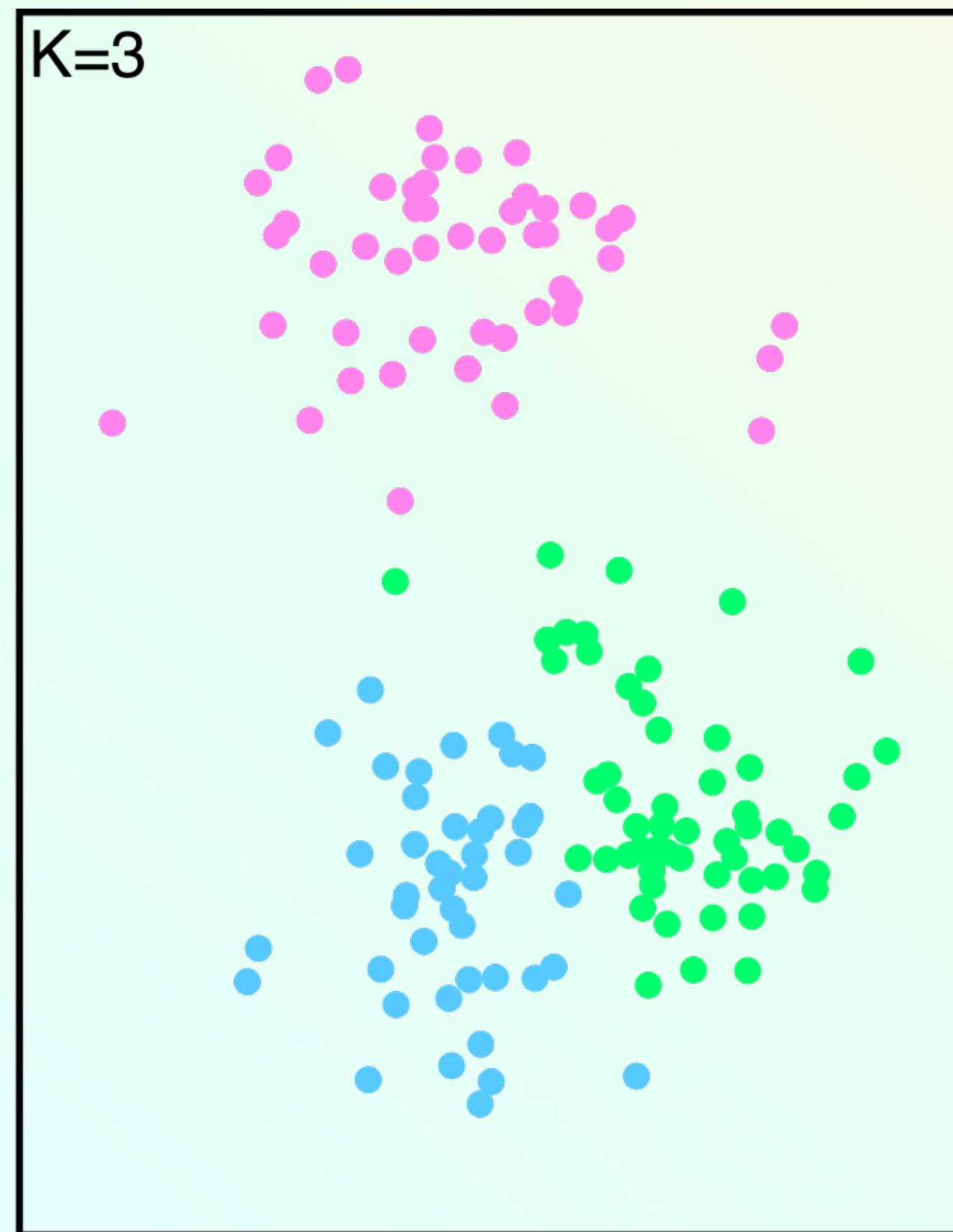
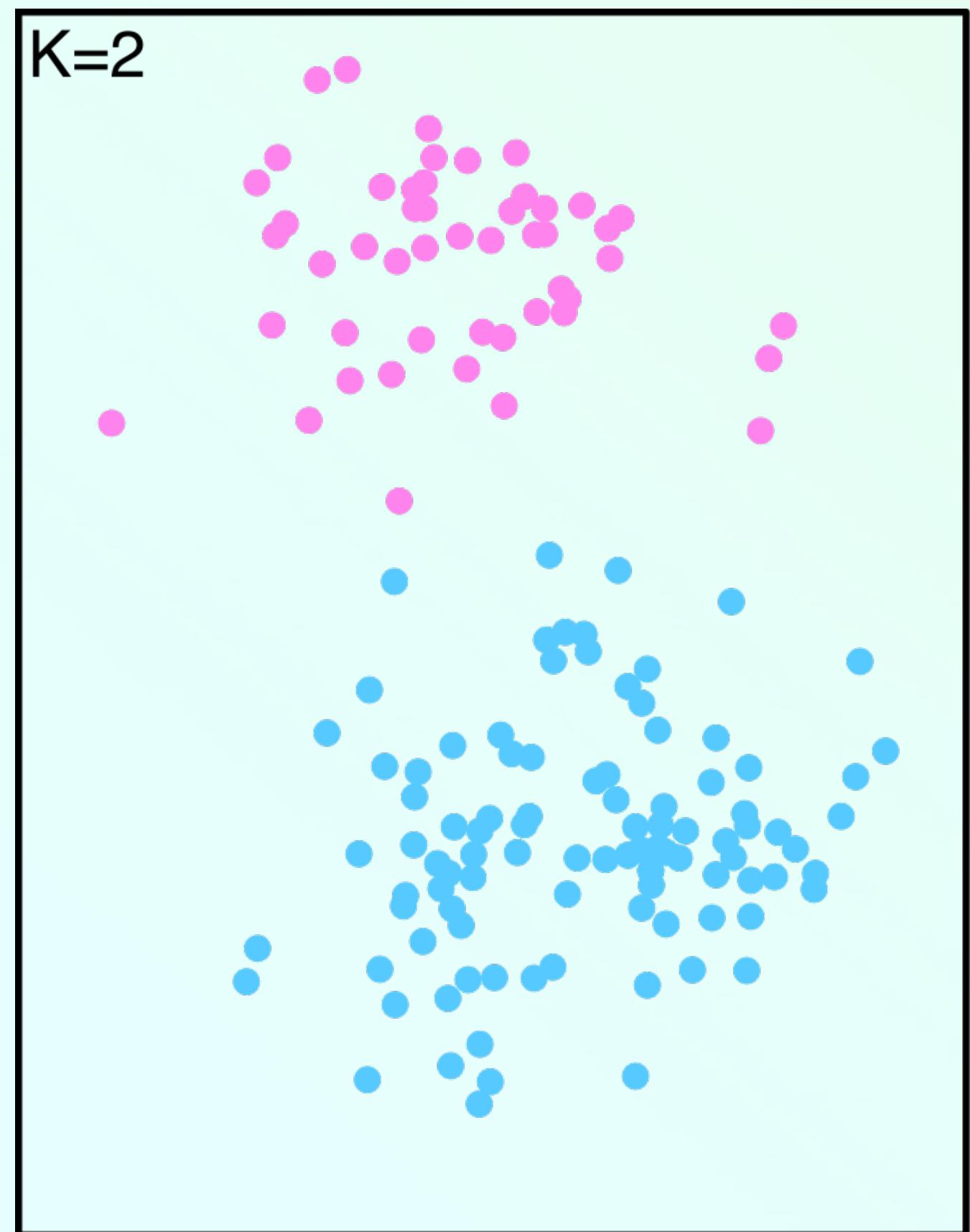
K-MEANS CLUSTERING

K-MEANS

K-means clustering es un enfoque simple para dividir un conjunto de datos en K grupos distintos y no superpuestos.

Para realizar la agrupación K-means, primero debemos especificar el número deseado de agrupaciones K y luego, el algoritmo asignará cada observación a exactamente uno de los K grupos.

K-MEANS



K-MEANS

Pongamos algunas notaciones, sean C_1, C_2, \dots, C_K son conjuntos que contienen los índices de las observaciones del dataset en cada cluster. Y se satisface las siguientes propiedades:

- Cada observación pertenece al menos a uno de los K clusters

$$C_1 \cup C_2 \cup \dots \cup C_K = \{1, 2, \dots, n\}$$

- Los cluster no se superponen, es decir, una observación pertenece a un solo cluster.

$$C_k \cap C_{k'} = \emptyset \quad \forall k \neq k'$$

K-MEANS

La idea detrás del algoritmo es que un buen agrupamiento es uno en donde la variación dentro del cluster es la menor posible.

La variación dentro de un cluster C_k es una medida $W(C_k)$ de la cantidad en la que las observaciones dentro de un cluster difieren entre sí. Lo que buscamos hacer es:

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

K-MEANS

Para poder resolver este problema, debemos definir a $W(C_K)$. Hay muchas métricas posibles, pero la más común es la distancia Euclíadiana cuadrada o Suma de Cuadrados Intracluster:

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^p (x_{ij} - x_{i'j})^2$$

$|C_k|$ es el número de observaciones en el cluster k.

Ahora, resolver esta minimización es un problema difícil de resolver dado a la inmensa cantidad de cálculos.

K-MEANS

Una forma de resolver esto de una forma más eficiente es calcular el centroide del cluster y calcular la distancia con respecto a ese punto.

El centroide de un cluster se calcula de la siguiente forma:

$$\text{Centroide} = \frac{1}{|C_k|} \sum_{i \in C_k} x_i$$

Entonces, ahora podemos calcular la distancia intra-cluster más sencillamente:

$$W(C_k) = \sum_{i \in C_k} \sum_{j=1}^p (x_{ij} - \bar{x}_j)^2$$

K-MEANS

Y ahora buscamos minimizar la suma de las distancias intra-cluster que es lo que se conoce como **inercia**:

$$Inercia = \sum_{k=1}^K W(C_k)$$

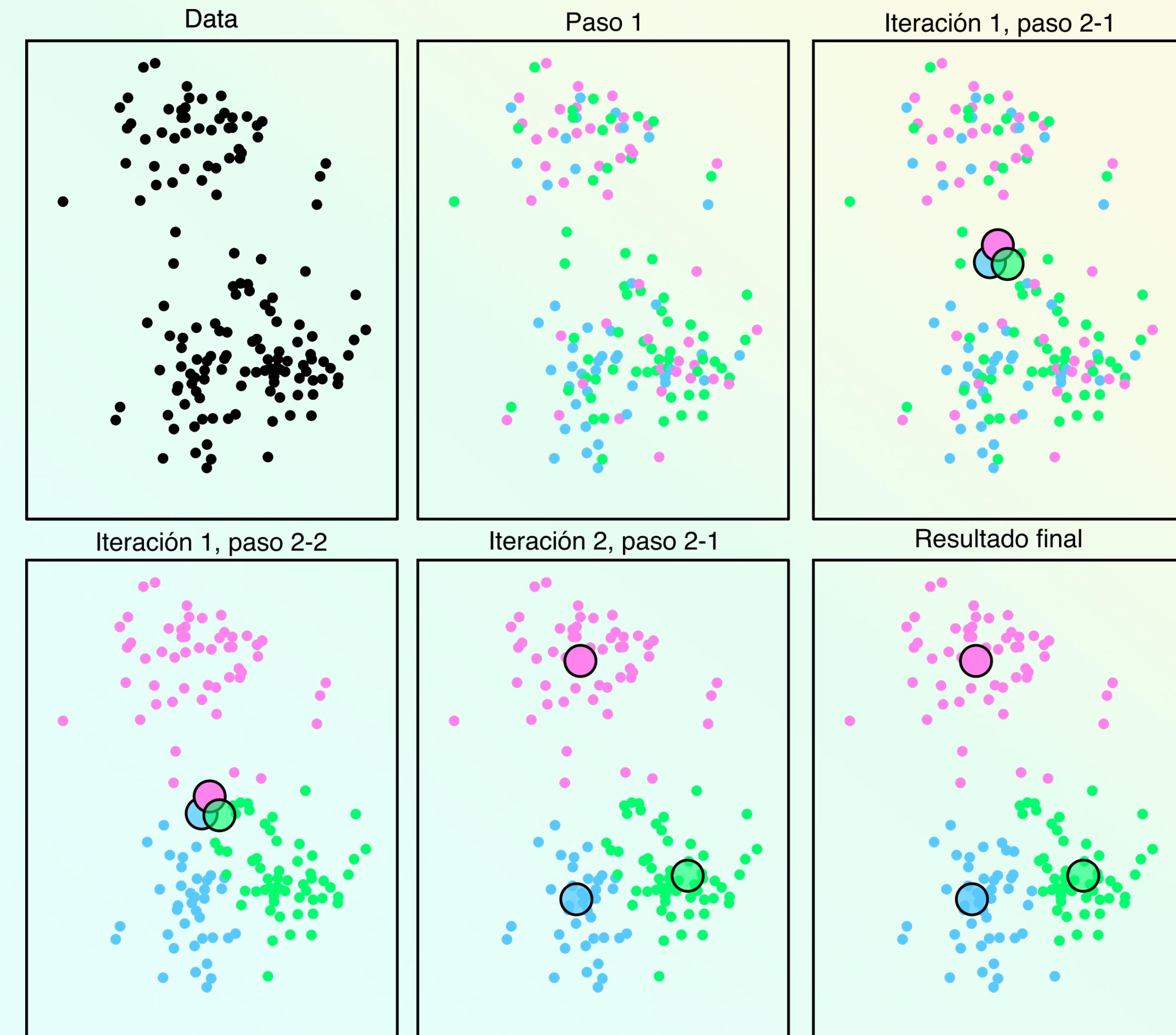
Y ahora sí, el problema de minimización es más eficiente, $\underset{C_1, \dots, C_K}{\text{minimizar}} \left\{ \sum_{k=1}^K W(C_k) \right\}$

El proceso es iterativo, y se va minimizando de a pasos hasta que llegue a un valor estable, siempre en dirección que reduce la inercia.

K-MEANS

1. Asignar aleatoriamente, de 1 a K, a cada observación. A modo de inicialización.
2. Iterar hasta que la asignación de cluster no cambia más,
 1. Para cada uno de los K clusters, se computa el **centroide** del cluster.
 2. Se asignan a cada observación al cluster al que el centroide se encuentre más cercano (mediante la distancia Euclidiana).
 3. Se calcula el valor de inercia.

K-MEANS

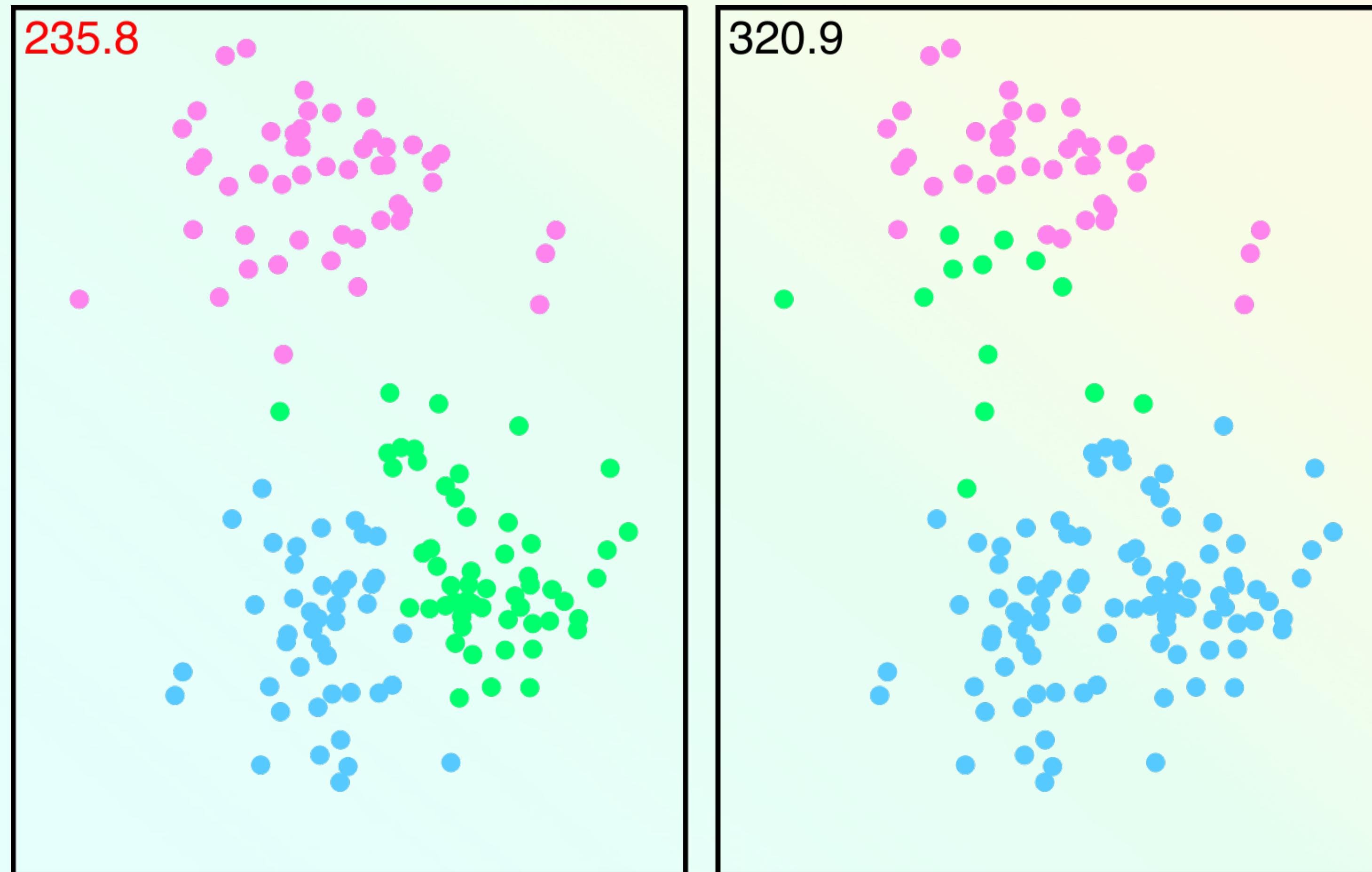


K-MEANS

Este algoritmo garantiza que se va a lograr la minimización de la inercia en cada paso. Además, dado que cada paso de iteración siempre disminuye, el algoritmo se ejecuta hasta que llegue a un mínimo local.

Esto significa que cada vez que se ejecuta este algoritmo, se puede obtener un resultado diferente. Por esta razón es recomendable correrlo múltiples veces y finalmente seleccionar la **mejor** solución (La que tenga menor inercia).

K-MEANS



MÉTODO DEL CODO

MÉTODO DEL CODO

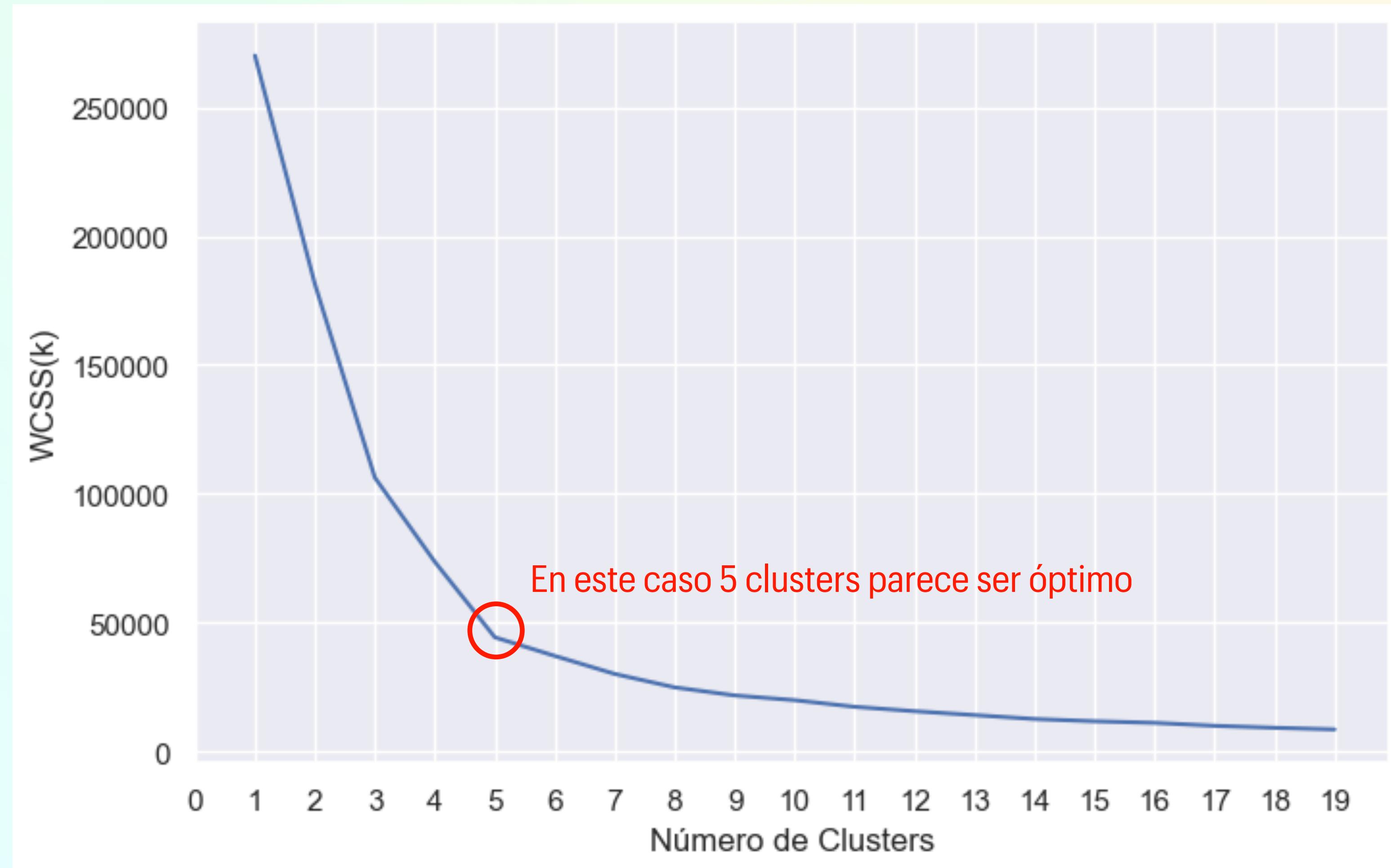
Un problema que nos presenta K-means es determinar el número óptimo de clusters...

El método del codo (Elbow Method) es una técnica utilizada para determinar el número óptimo de clusters en un conjunto de datos. Se basa en ver variación de una métrica de calidad del clustering (generalmente la inercia) a medida que se varía el número de clusters.

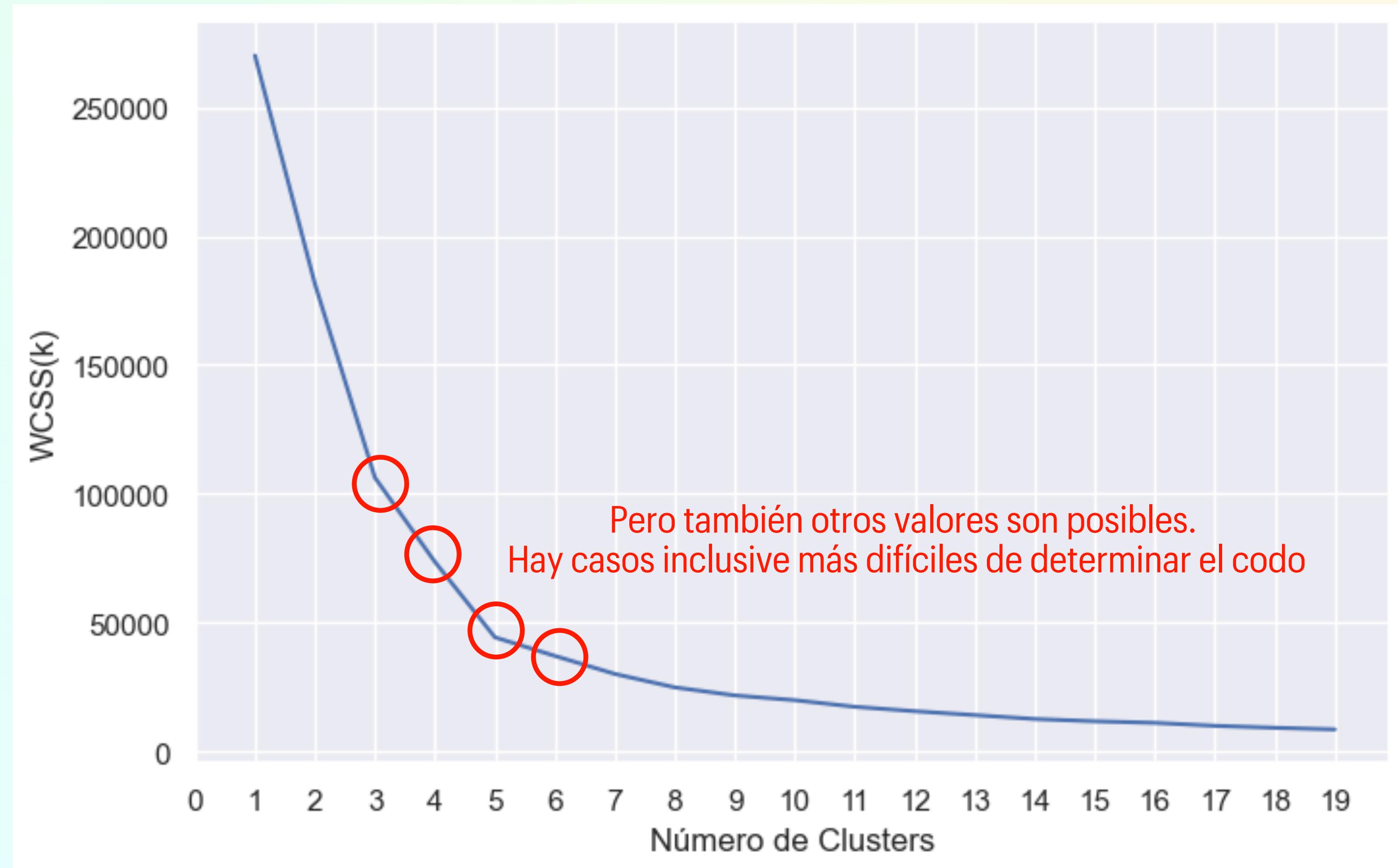
MÉTODO DEL CODO

1. Se ejecuta el algoritmo de clustering (por ejemplo, k-means) en el conjunto de datos para diferentes valores de k (número de clusters).
2. Para cada valor de k , se calcula una métrica de calidad del clustering, generalmente es la suma de cuadrados dentro del cluster (inercia).
3. Luego, se traza un gráfico que muestra los valores de k en el eje x y los valores de la métrica (inercia) en el eje y.
4. Se observa el gráfico y se busca un punto de **codo**. El punto de codo es aquel donde la métrica (inercia) comienza a disminuir a una tasa más lenta. En otras palabras, es el punto donde agregar más clusters ya no mejora significativamente la calidad del clustering.

MÉTODO DEL CODO



MÉTODO DEL CODO



ÍNDICE DE LA SILUETA

ÍNDICE DE LA SILUETA

Otra métrica que podemos ver es el puntaje de la silueta. Es una métrica que combina la cohesión y la separación para evaluar la calidad de los clusters y proporcionar información sobre la "bondad" de la agrupación. Para cada punto en un cluster, se calcula:

$$a_i = \frac{1}{|C_I| - 1} \sum_{j \in C_I, i \neq j} d(i, j)$$

Cohesión de i con sus co-miembros

$$b_i = \min_{J \neq I} \frac{1}{|C_J|} \sum_{j \in C_I} d(i, j)$$

Separación de i con otros clusters

ÍNDICE DE LA SILUETA

Métricas relacionadas al índice de la silueta son:

- **Cohesión**: Es una medida de cuán cerca están los puntos dentro de un mismo cluster.
- **Separación**: Es una medida de cuán lejos están los clusters entre sí.

Estas métricas se podrían calcular como métricas independientes para ver la calidad del modelo.

ÍNDICE DE LA SILUETA

El índice de silueta se calcula para cada punto como:

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)}$$

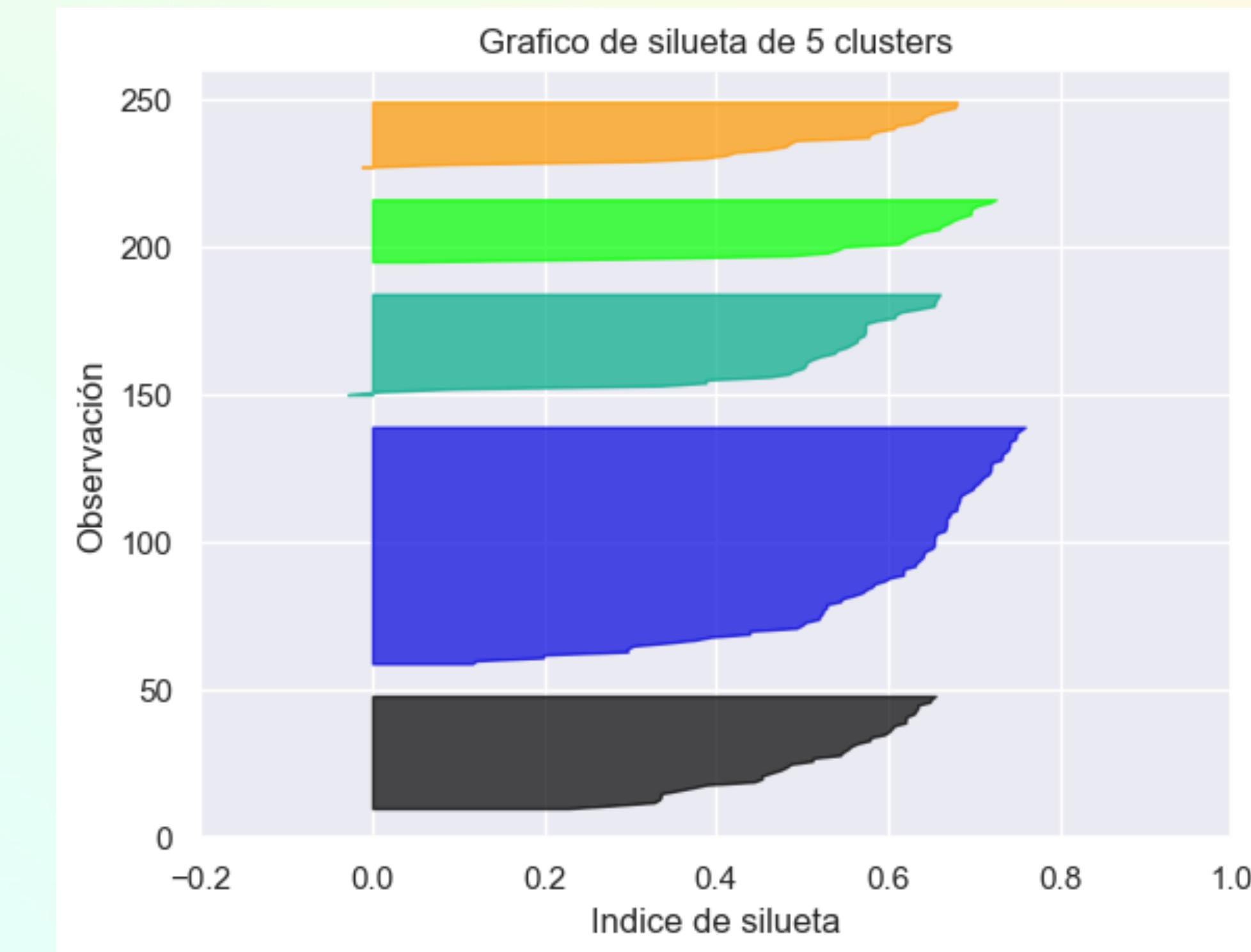
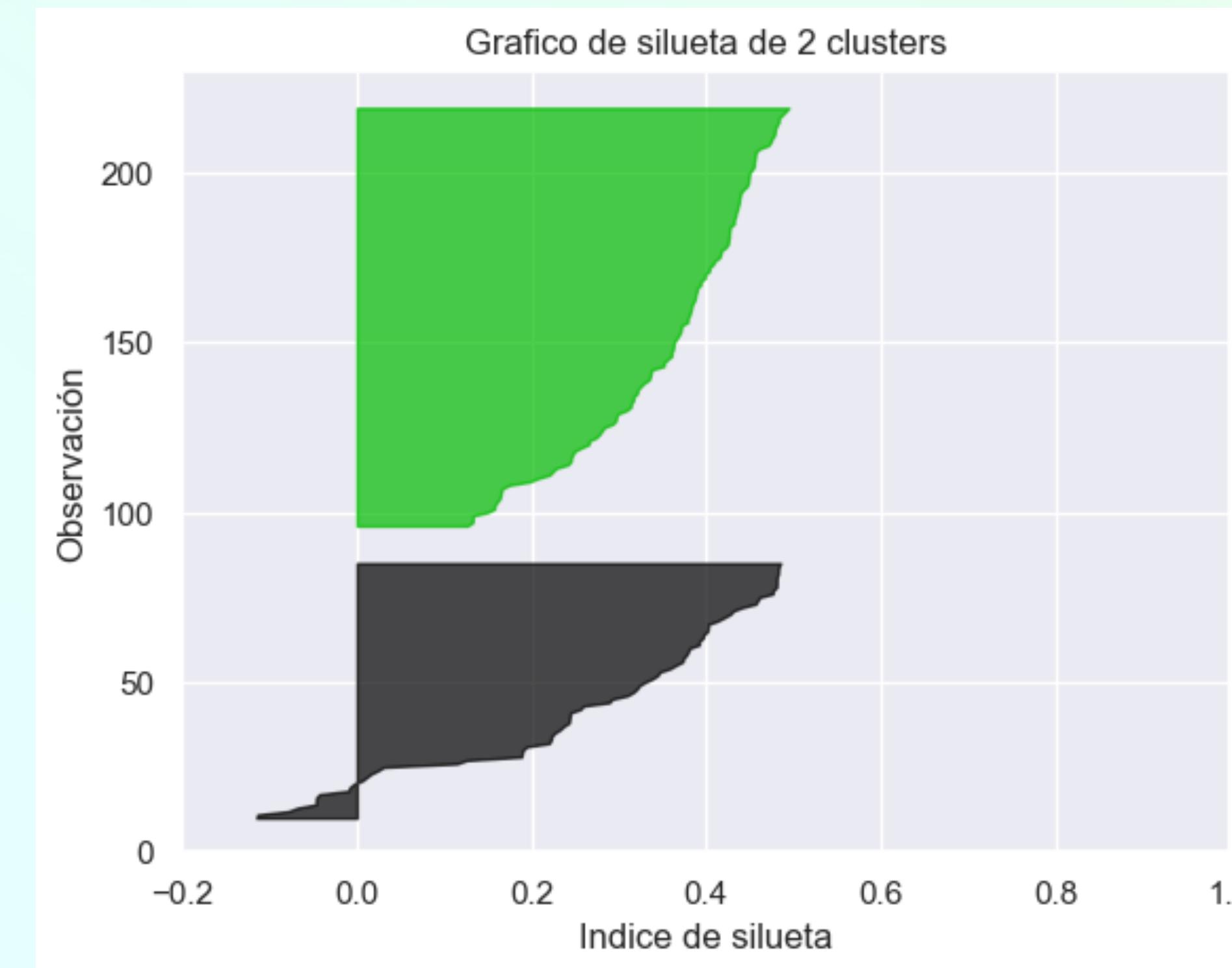
Los valores del índice de silueta varían entre -1 y 1, donde:

- Cercano a 1 indica que una observación es muy similar a su cluster y disímil de los otros
- 0 indica que prácticamente los cluster se superponen y es difícil determinar la pertenencia.
- -1 significa que la observación tendría más sentido que esté asignada a otro cluster.

Dado la dificultad de calcular esto, $O(n^2)$, en realidad se calcula con respecto a los centroides de los clusters.

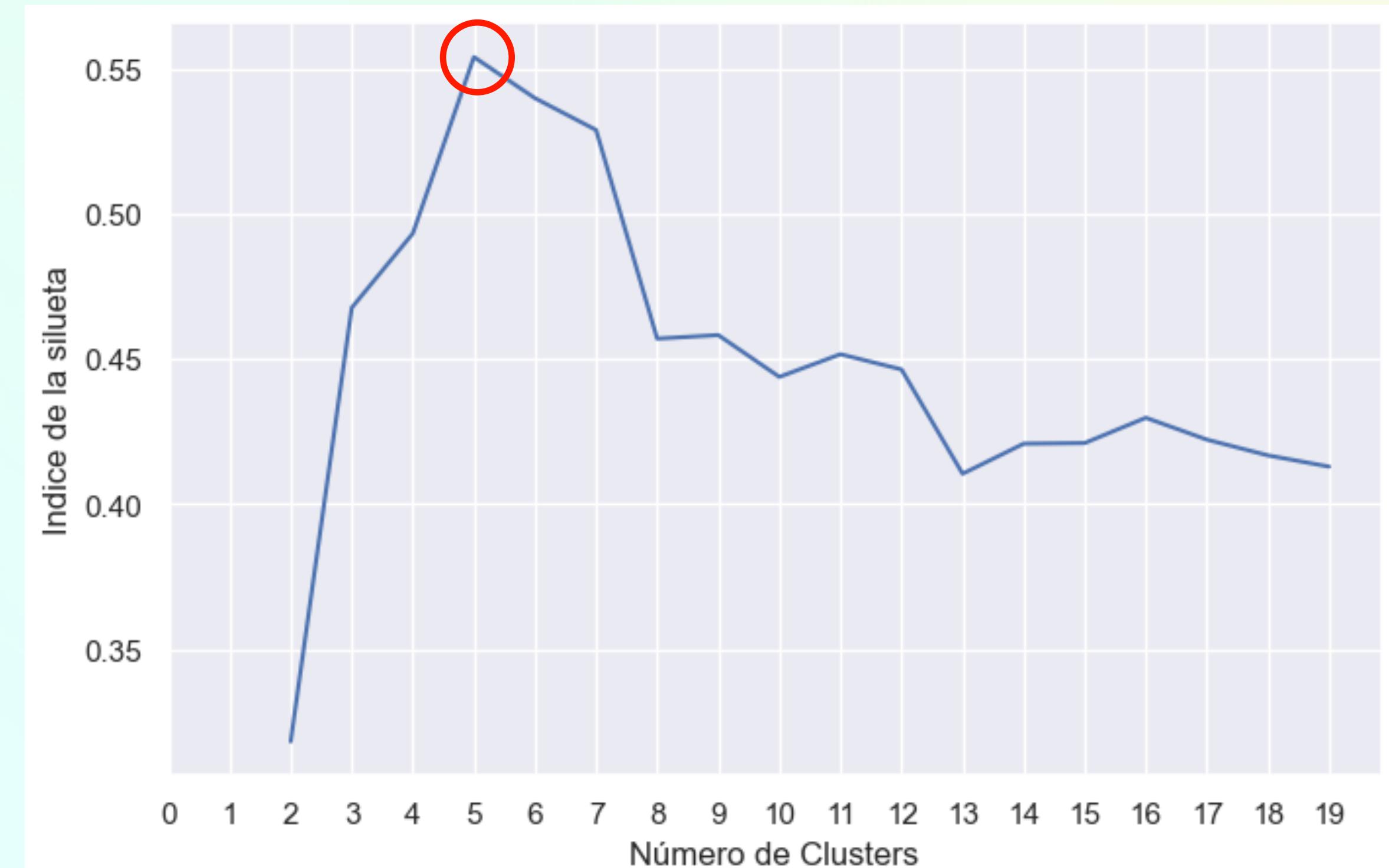
ÍNDICE DE LA SILUETA

Este índice nos permite graficar los cluster en caso de que son de más de dos dimensiones, ver el tamaño de los clusters y que tan seguros son la pertenencia de cada observación en un cluster.



ÍNDICE DE LA SILUETA

Pero la forma que más se usa es para establecer si la cantidad de clusters es apropiada. Para ello se calcula el promedio de todas las observaciones para indicarnos qué tan apropiadamente están los datos agrupados.



FUERZA DE PREDICCIÓN

FUERZA DE PREDICCIÓN

Otro método que creció en popularidad es el llamado **Fuerza de predicción**.

La idea se toma prestado del caso de un clasificador de aprendizaje supervisado, en el cual se minimiza un error de predicción, y además permite obtener estimaciones de observaciones individuales.

Este método se enfoca en el error de predicción en vez de la suma de los cuadrados intra-cluster o la cohesión o separación de los clusters.

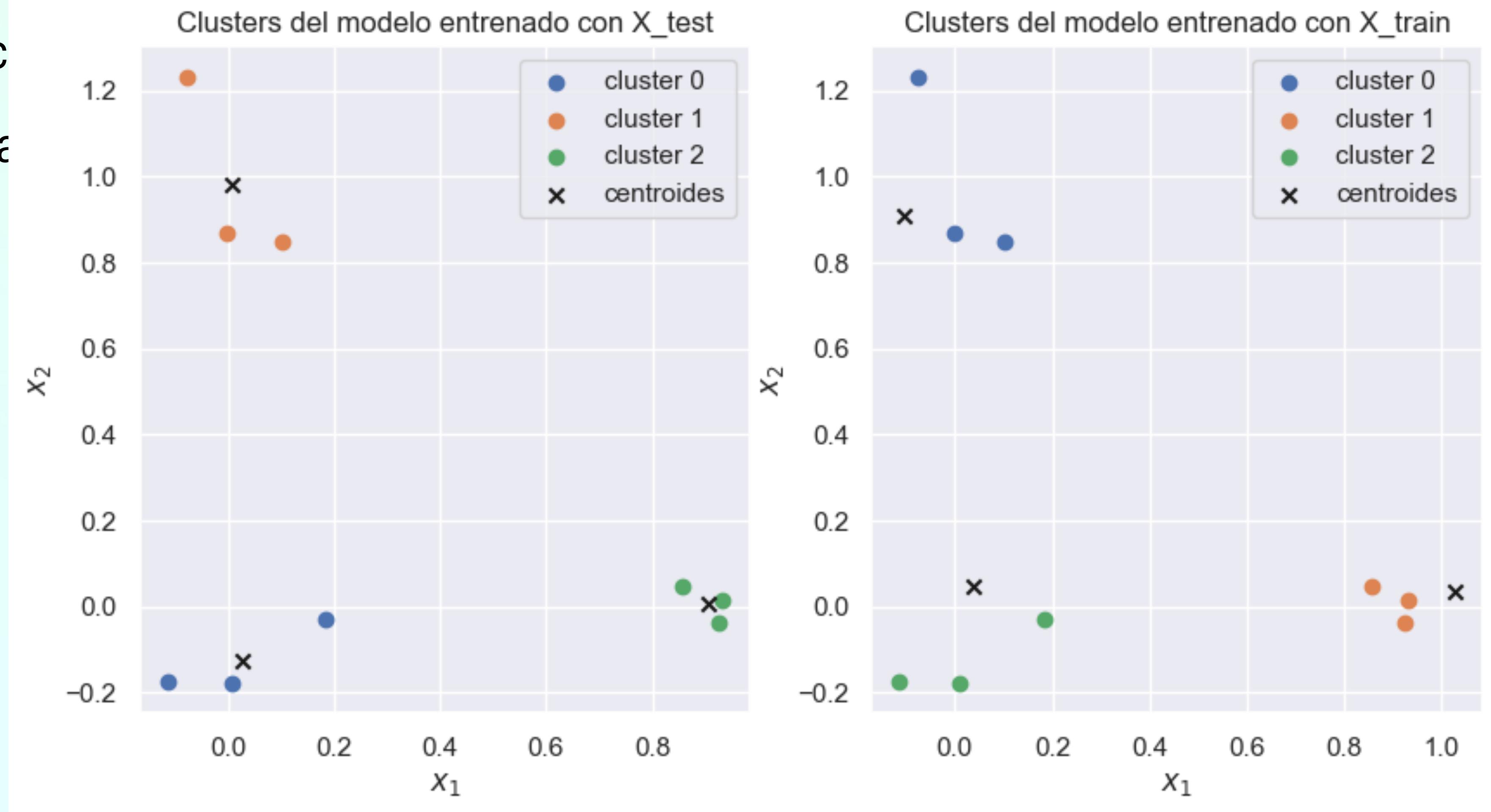
FUERZA DE PREDICCIÓN

¿Cómo hacemos para obtener un error de predicción en un modelo de clustering?

En su forma más simple, el dataset se separa en dos, X_{train} y X_{test} , y para cada uno de ellos se entrena un modelo de clustering y se evalúa a ambos modelos con X_{test} .

FUERZA DE PREDICCIÓN

¿Cómo hac
En su forma
entrena un



FUERZA DE PREDICCIÓN

Entonces para el modelo entrenado con X_{test} , obtenemos la matriz de co-miembros para algún cluster:

	0	1	2	3	4	5	6	7	8
0		1	1						
1	1		1						
2	1	1							
3									
4									
5									
6									
7									
8									

FUERZA DE PREDICCIÓN

Y luego vemos si en el modelo entrenado con X_{train} si estas conexiones se siguen manteniendo:

	0	1	2	3	4	5	6	7	8
0		1	0						
1	1		1						
2	0	1							
3									
4									
5									
6									
7									
8									

FUERZA DE PREDICCIÓN

Finalmente se calcula la proporción de conexiones para el cluster:

$$FP_k = \frac{\text{Conexiones que se mantuvieron}}{|C_I|(|C_I| - 1)}$$

Y el valor para medir la calidad de la cantidad de clusters, se obtiene el mínimo valor de todos los clusters:

$$FP = \min(FP_k)$$

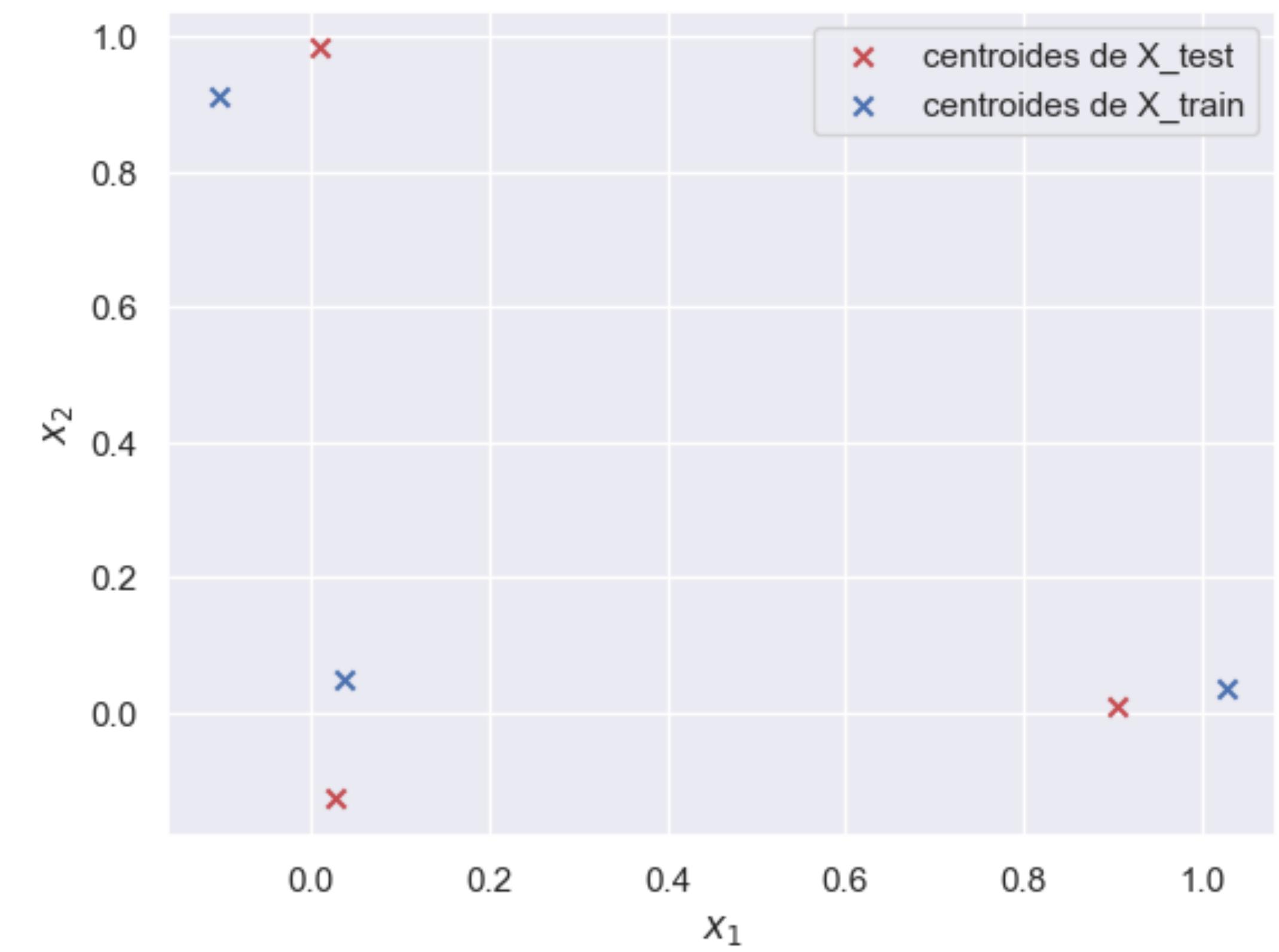
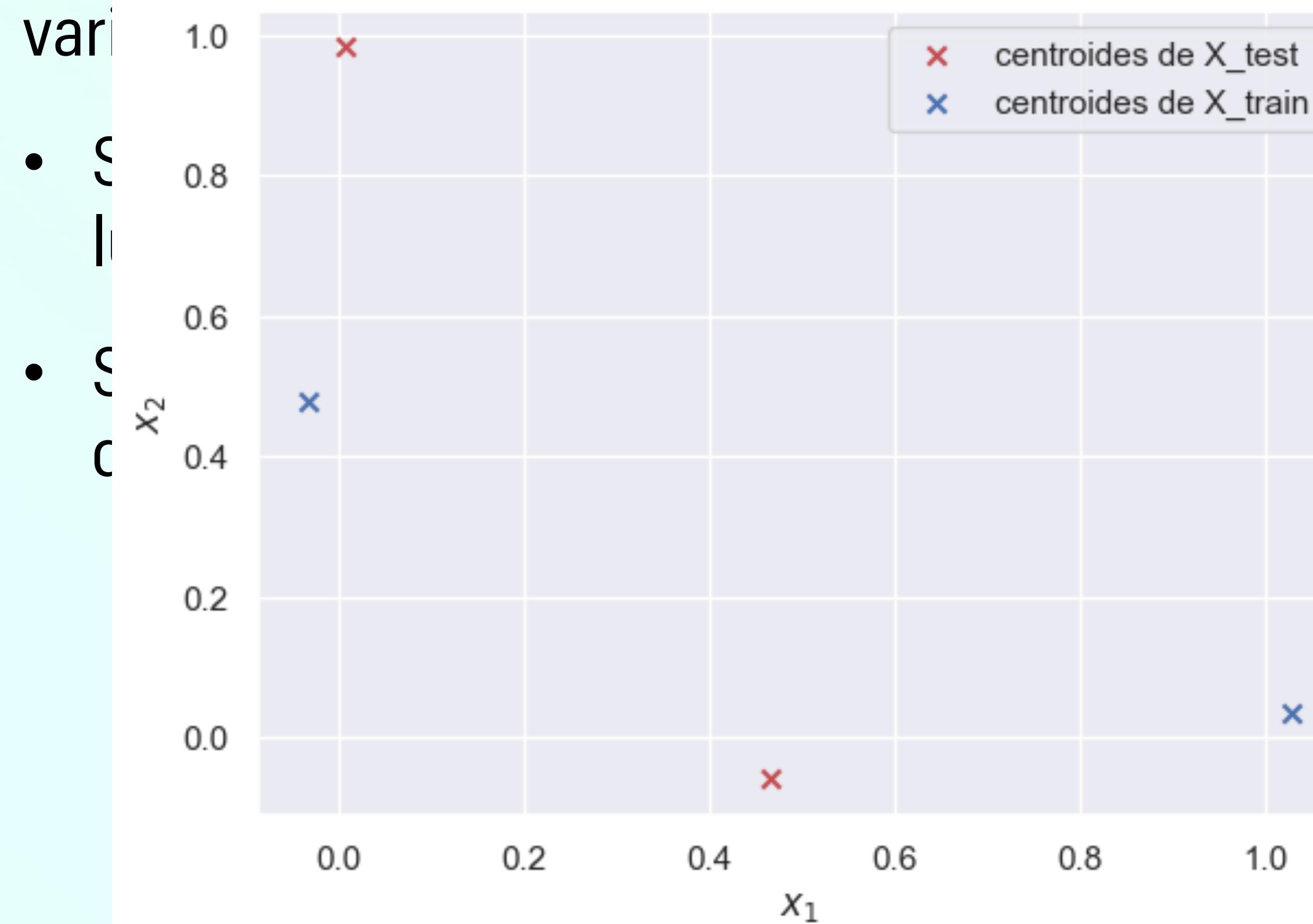
FUERZA DE PREDICCIÓN

¿Por qué este método es tan popular? Esto se debe a que nos permite medir el error de variabilidad,

- Si un apropiado número de clusters es elegido, los centroides de los dos datasets estarán en lugares cercanos con un error de variabilidad bajo.
- Si la cantidad es inapropiada, los centroides van a estar en lugares muy diferentes, y van a depender de cómo se separó el dataset. Es decir, la variabilidad es alta.

FUERZA DE PREDICIÓN

Centroides de cada modelo



FUERZA DE PREDICCIÓN

¿Por qué este método es tan popular?

Además, permite extender el método para usarse con validación cruzada.

VAMOS A PRÁCTICAR UN POCO...

MODELO DE MIXTURA GAUSSIANA

MODELO DE MIXTURA GAUSSIANA

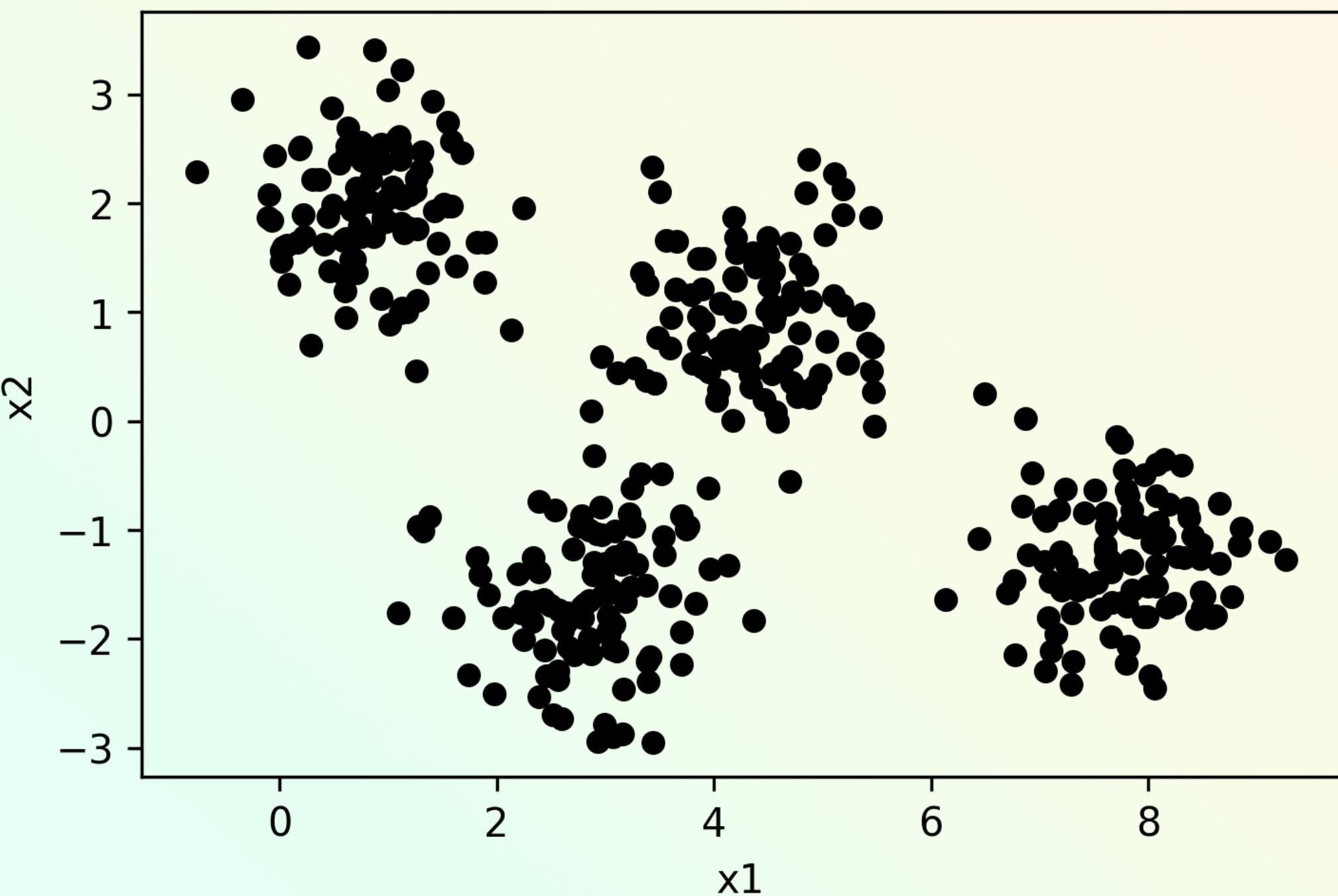
K-means es simple y relativamente fácil de entender, pero su simplicidad plantea desafíos prácticos en su aplicación. **La naturaleza no probabilística** de K-means y el uso de simplemente una distancia desde el centro del grupo para asignar la membresía del grupo conduce a un rendimiento deficiente en muchas situaciones del mundo real.

Veamos entonces a los modelos de mixtura gaussiana (GMM), que pueden verse como una extensión de las ideas detrás de K-means, pero también pueden ser una herramienta poderosa para la estimación más allá de la simple agrupación.

MODELO DE MIXTURA GAUSSIANA

Analicemos las debilidades de k-means:

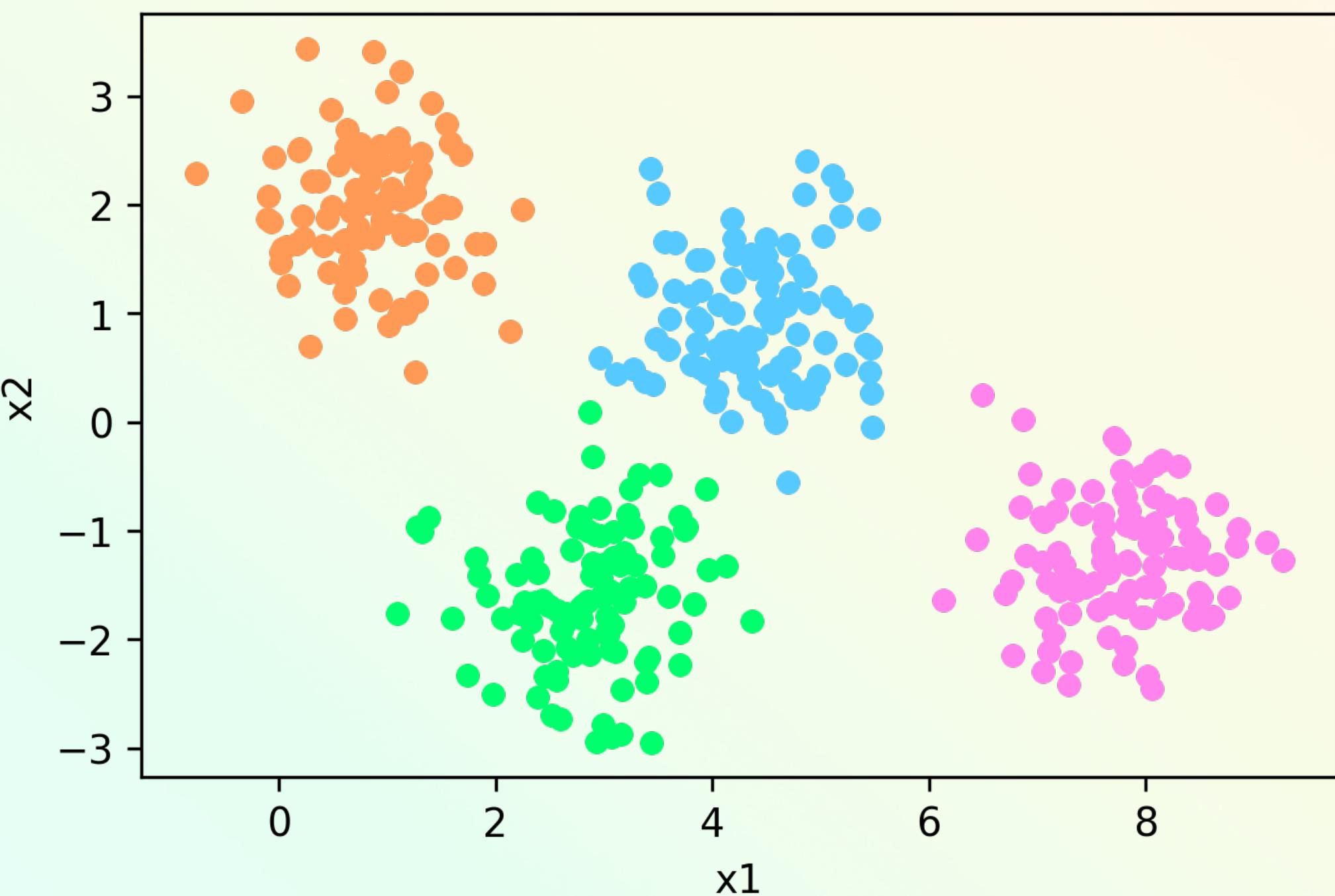
Si tenemos bloques separados de datos, el algoritmo puede agrupas rápidamente esos.



MODELO DE MIXTURA GAUSSIANA

Analicemos las debilidades de k-means:

Si tenemos bloques separados de datos, el algoritmo puede agrupas rápidamente esos.



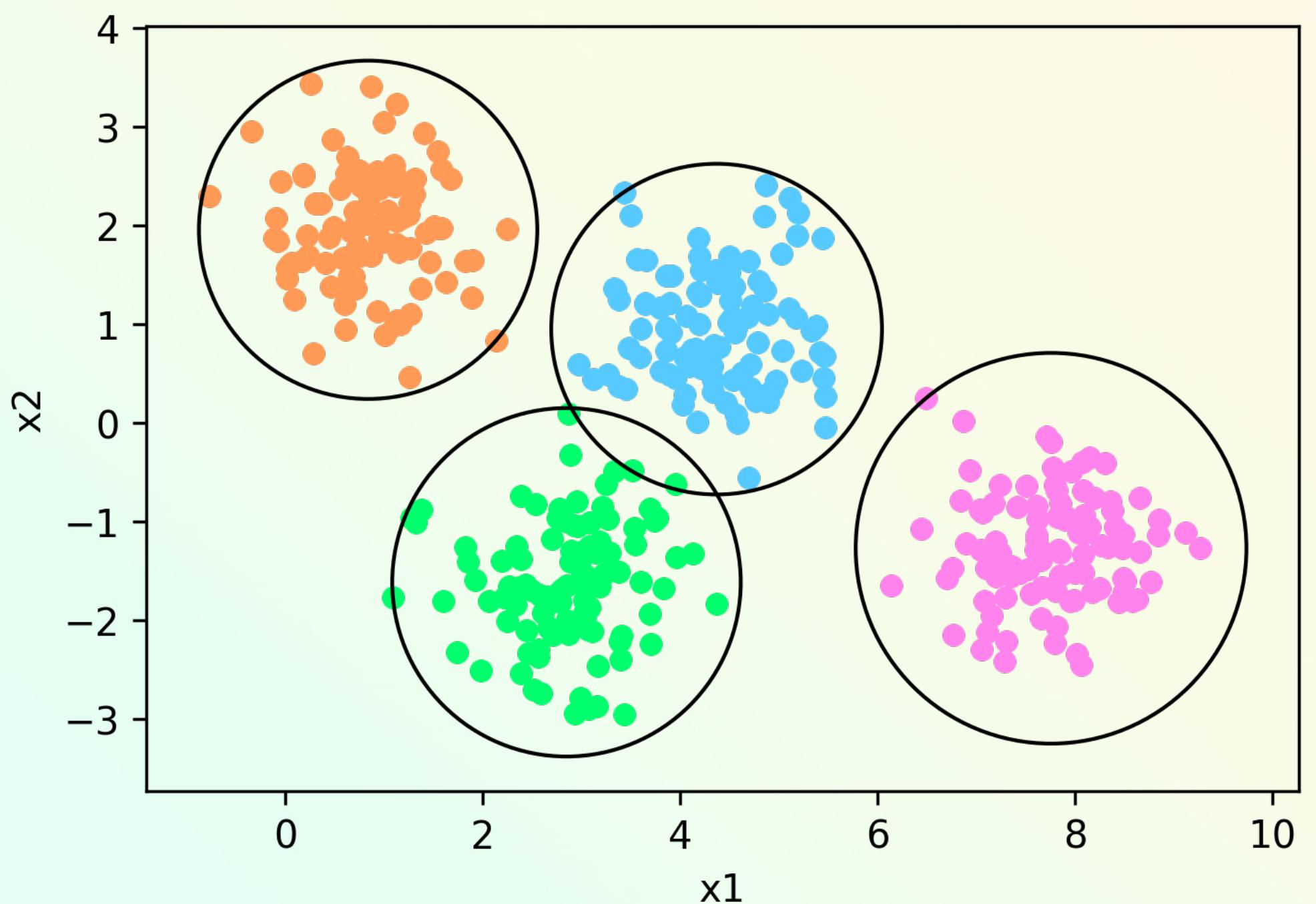
MODELO DE MIXTURA GAUSSIANA

Analicemos las debilidades de k-means:

Si tenemos bloques separados de datos, el algoritmo puede agrupas rápidamente esos.

Hay etiquetados de cluster en observaciones que son más segura que otras, las menos es el espacio entre el cluster azul y el verde.

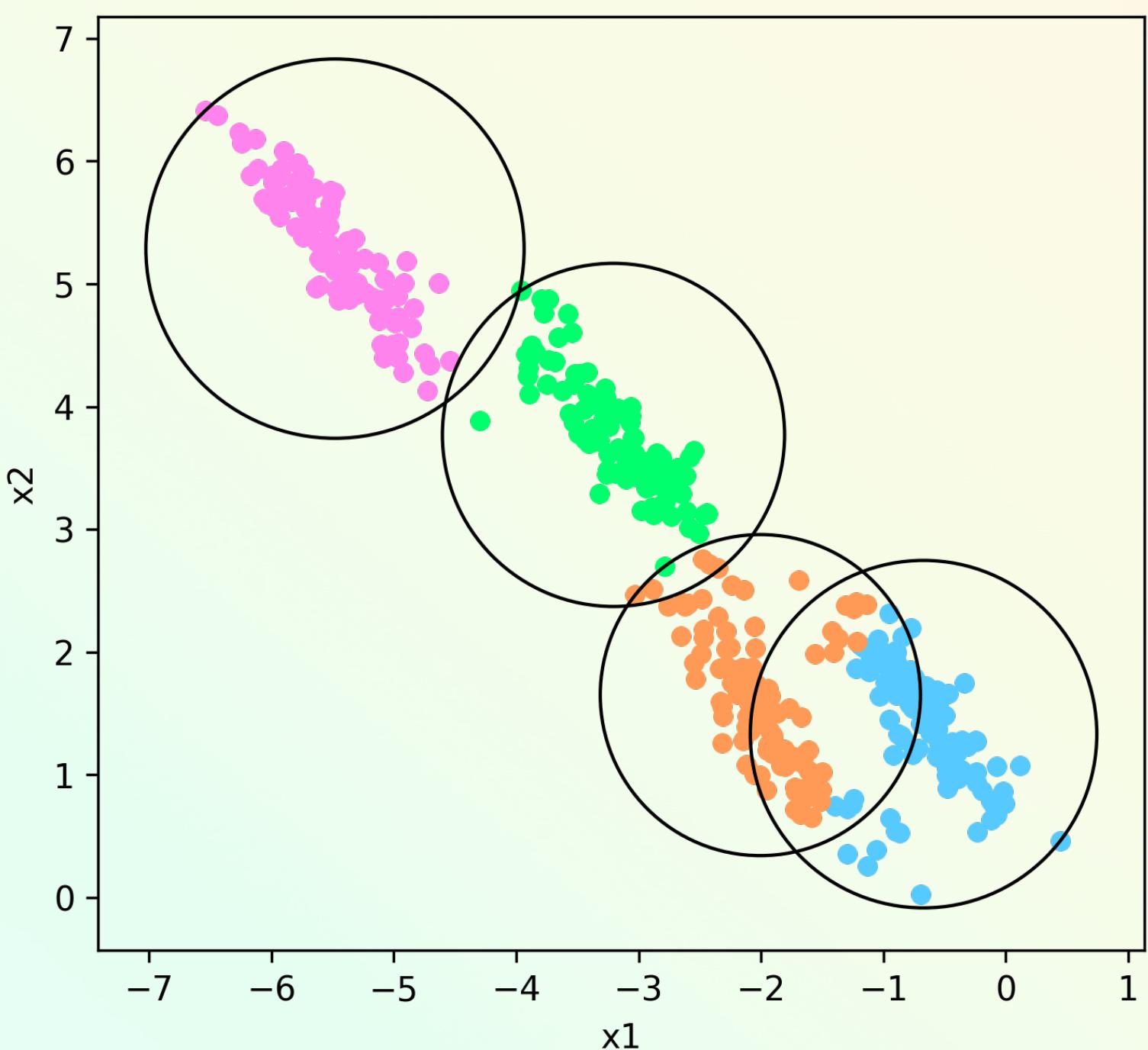
K-means **no tiene una medida de probabilidad o de incertidumbre.**



MODELO DE MIXTURA GAUSSIANA

Analicemos las debilidades de k-means:

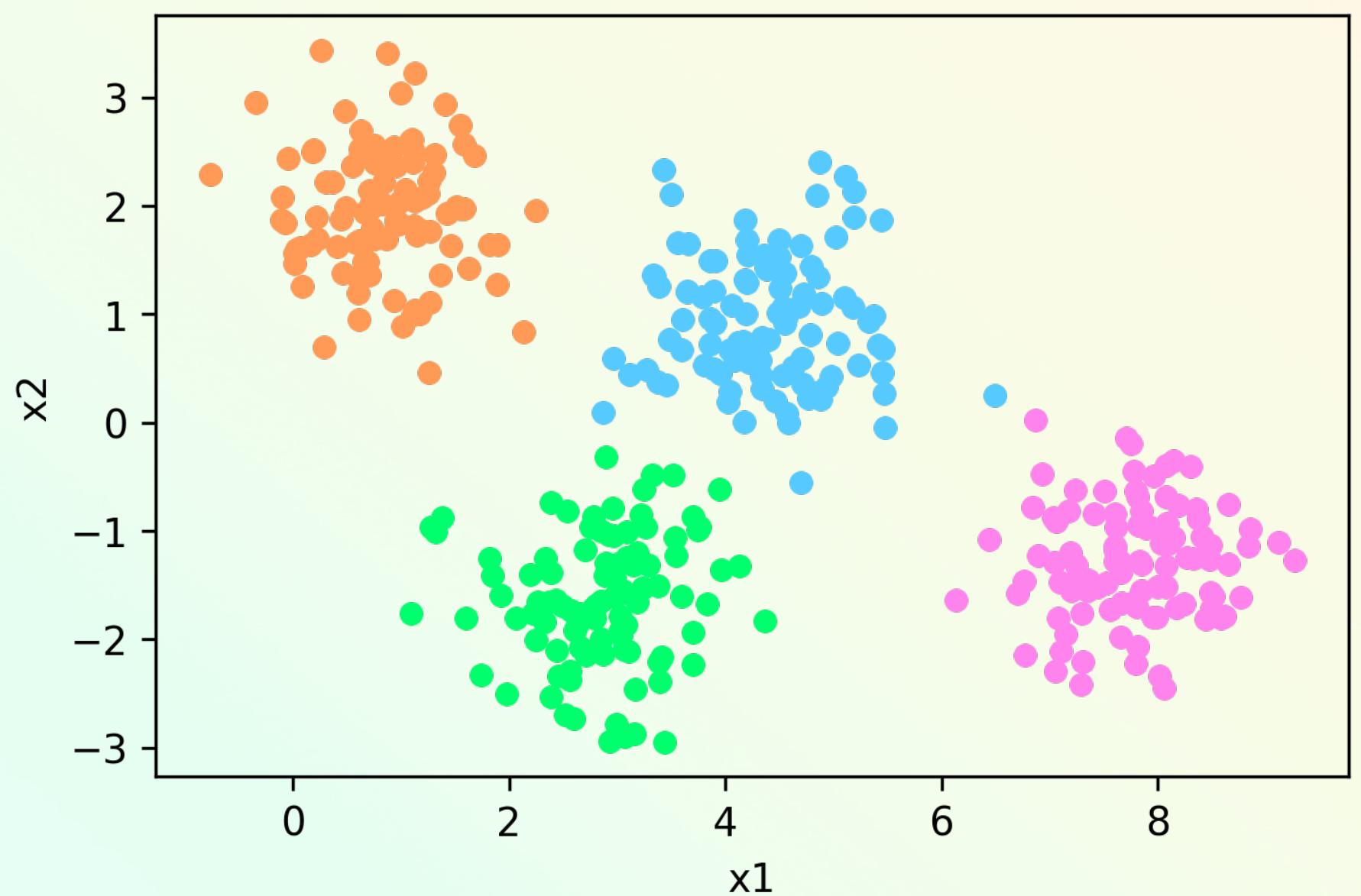
Un punto importante es que los clusters deben ser circulares... **k-means no es suficiente flexible para adaptarse a este tipo de casos.**



MODELO DE MIXTURA GAUSSIANA

Un GMM intenta encontrar una mezcla de distribuciones de probabilidad gaussianas multidimensionales que modelen mejor cualquier conjunto de datos de entrada.

En el caso más simple, los GMM se pueden utilizar para encontrar conglomerados de la misma manera que k-means.

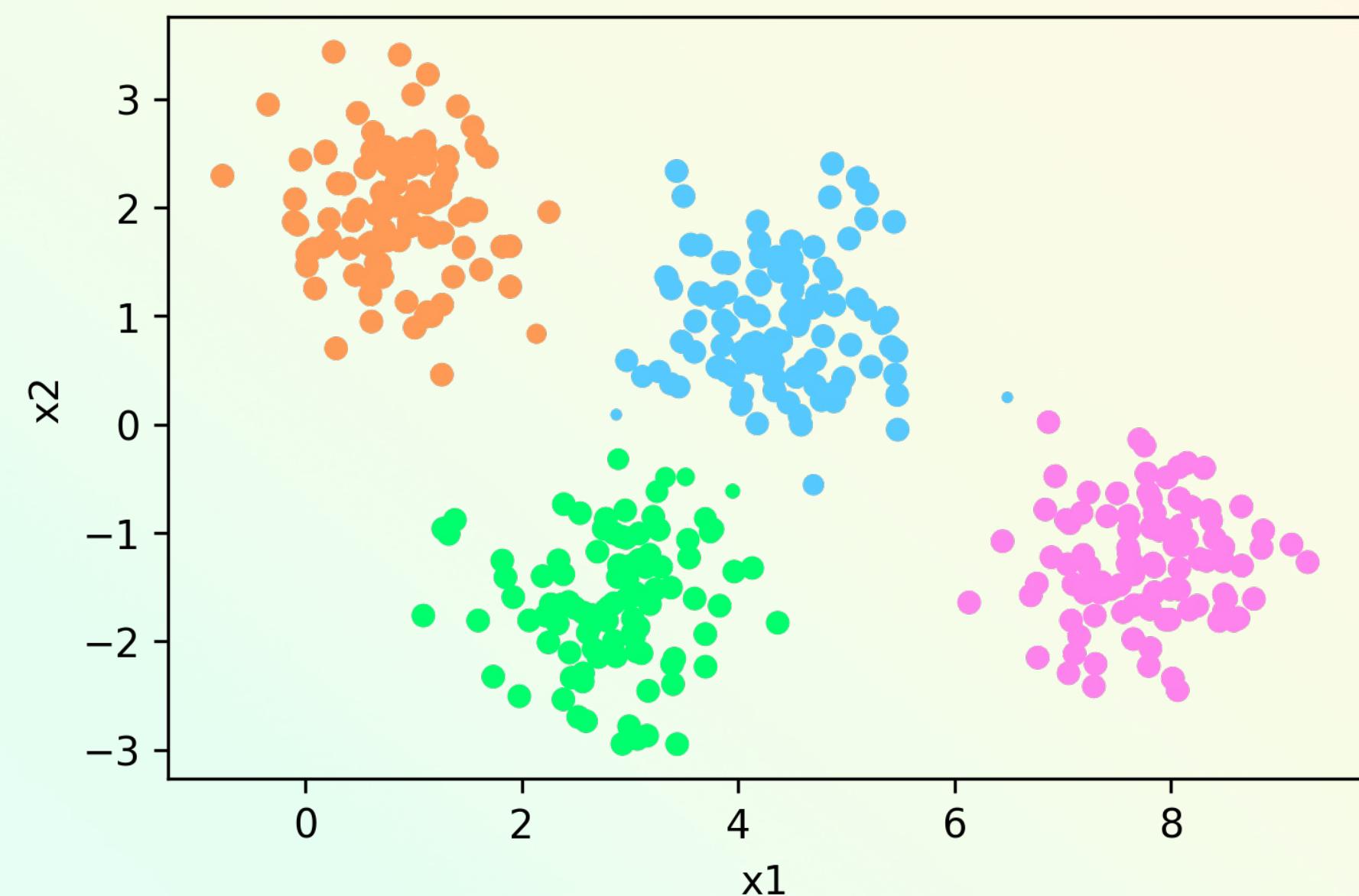


MODELO DE MIXTURA GAUSSIANA

Un GMM intenta encontrar una mezcla de distribuciones de probabilidad gaussianas multidimensionales que modelen mejor cualquier conjunto de datos de entrada.

En el caso más simple, los GMM se pueden utilizar para encontrar conglomerados de la misma manera que k-means.

Pero debido a que GMM contiene un modelo probabilístico, también es posible encontrar asignaciones probabilísticas.

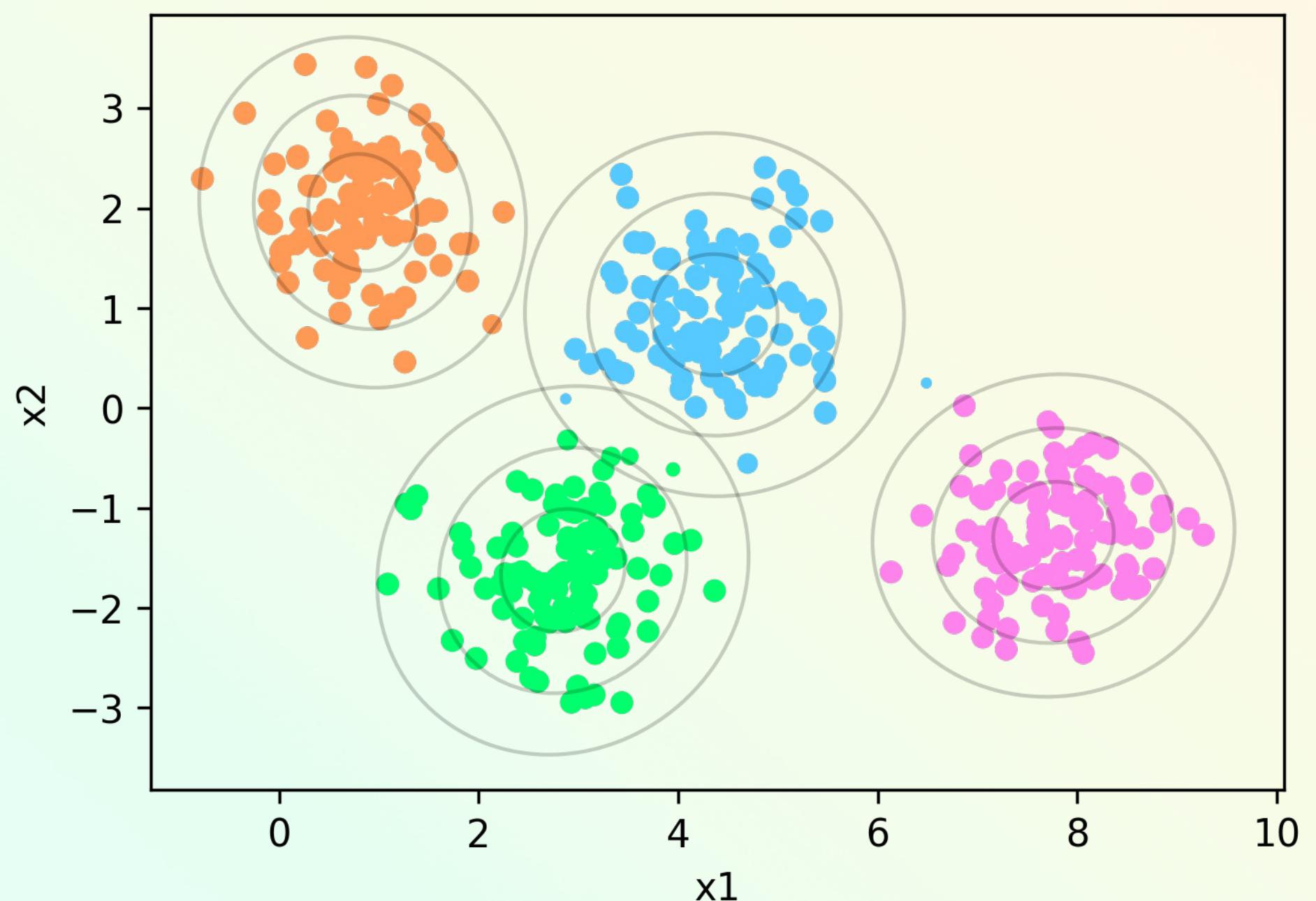


MODELO DE MIXTURA GAUSSIANA

Un GMM intenta encontrar una mezcla de distribuciones de probabilidad gaussianas multidimensionales que modelen mejor cualquier conjunto de datos de entrada.

En el caso más simple, los GMM se pueden utilizar para encontrar conglomerados de la misma manera que k-means.

Pero debido a que GMM contiene un modelo probabilístico, también es posible encontrar asignaciones probabilísticas.

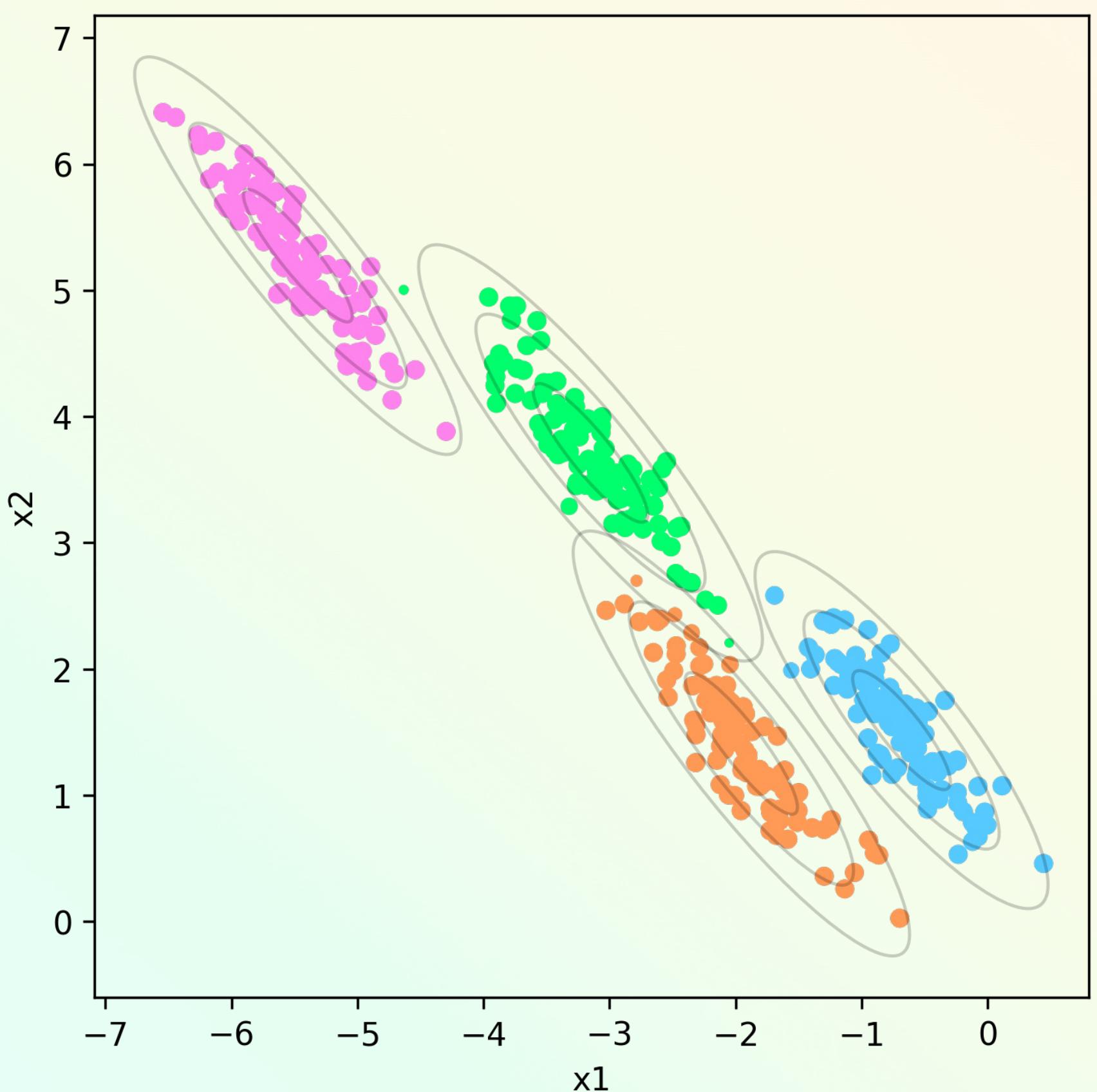


MODELO DE MIXTURA GAUSSIANA

Un GMM intenta encontrar una mezcla de distribuciones de probabilidad gaussianas multidimensionales que modelen mejor cualquier conjunto de datos de entrada.

En el caso más simple, los GMM se pueden utilizar para encontrar conglomerados de la misma manera que k-means.

Pero debido a que GMM contiene un modelo probabilístico, también es posible encontrar asignaciones probabilísticas.



MODELO DE MIXTURA GAUSSIANA

Este modelo tiene la siguiente expresión:

$$f_X = \sum_{j=1}^k \phi_j f_{\mu_j \Sigma_j}$$

Donde $f_{\mu_j \Sigma_j}$ es una distribución normal multivariada con media μ_j y covarianza Σ_j .

Este modelo representa una suma pesada de k distribución normal multivariada cada una con peso ϕ_j .

Los valores de μ_j , Σ_j y ϕ_j son los parámetros por entrenar. Este entrenamiento se obtiene mediante el **algoritmo de máxima expectación (EM)** para optimizar con el **criterio de máxima verosimilitud**.

MODELO DE MIXTURA GAUSSIANA

Para simplificar, tomemos un caso de data unidimensional y dos clusters ($k=2$):

$$f(x | \mu_1 \sigma_1^2) \leftarrow \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-(x-\mu_1)^2 / 2\sigma_1^2} \quad \text{y} \quad f(x | \mu_2 \sigma_2^2) \leftarrow \frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-(x-\mu_2)^2 / 2\sigma_2^2}$$

EM trabaja de la siguiente forma. A principio inicializa al azar los parámetros $\mu_1, \sigma_1^2, \mu_2, \sigma_2^2$ y hace que $\phi_1 = \phi_2 = \frac{1}{k}$.

MODELO DE MIXTURA GAUSSIANA

En cada iteración de EM los siguientes cuatro pasos se ejecutan:

1. Para cada observación $i=1,\dots,N$, se calcula la probabilidad:

$$f(x | \mu_1 \sigma_1^2) \leftarrow \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}} \quad \text{y} \quad f(x | \mu_2 \sigma_2^2) \leftarrow \frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-\frac{(x-\mu_2)^2}{2\sigma_2^2}}$$

2. Usando la regla de Bayes, para cada observación, calcula la probabilidad $b_i^{(j)}$ que la observación pertenece al cluster $j \in \{1,2\}$ (la probabilidad de que la observación sea extraída de la distribución normal j):

$$b_i^{(j)} \leftarrow \frac{f(x_i | \mu_j \sigma_j^2) \phi_j}{f(x_i | \mu_1 \sigma_1^2) \phi_1 + f(x_i | \mu_2 \sigma_2^2) \phi_2}$$

El parámetro ϕ_j refleja la probabilidad de que la distribución gaussiana j con parámetros μ_j y σ_j^2 haya producido la observación. Por eso iniciamos $\phi_j = \frac{1}{2}$ es la probabilidad a priori.

MODELO DE MIXTURA GAUSSIANA

En cada iteración de EM los siguientes cuatro pasos se ejecutan:

3. Computa nuevos valores de μ_j y σ_j^2 :

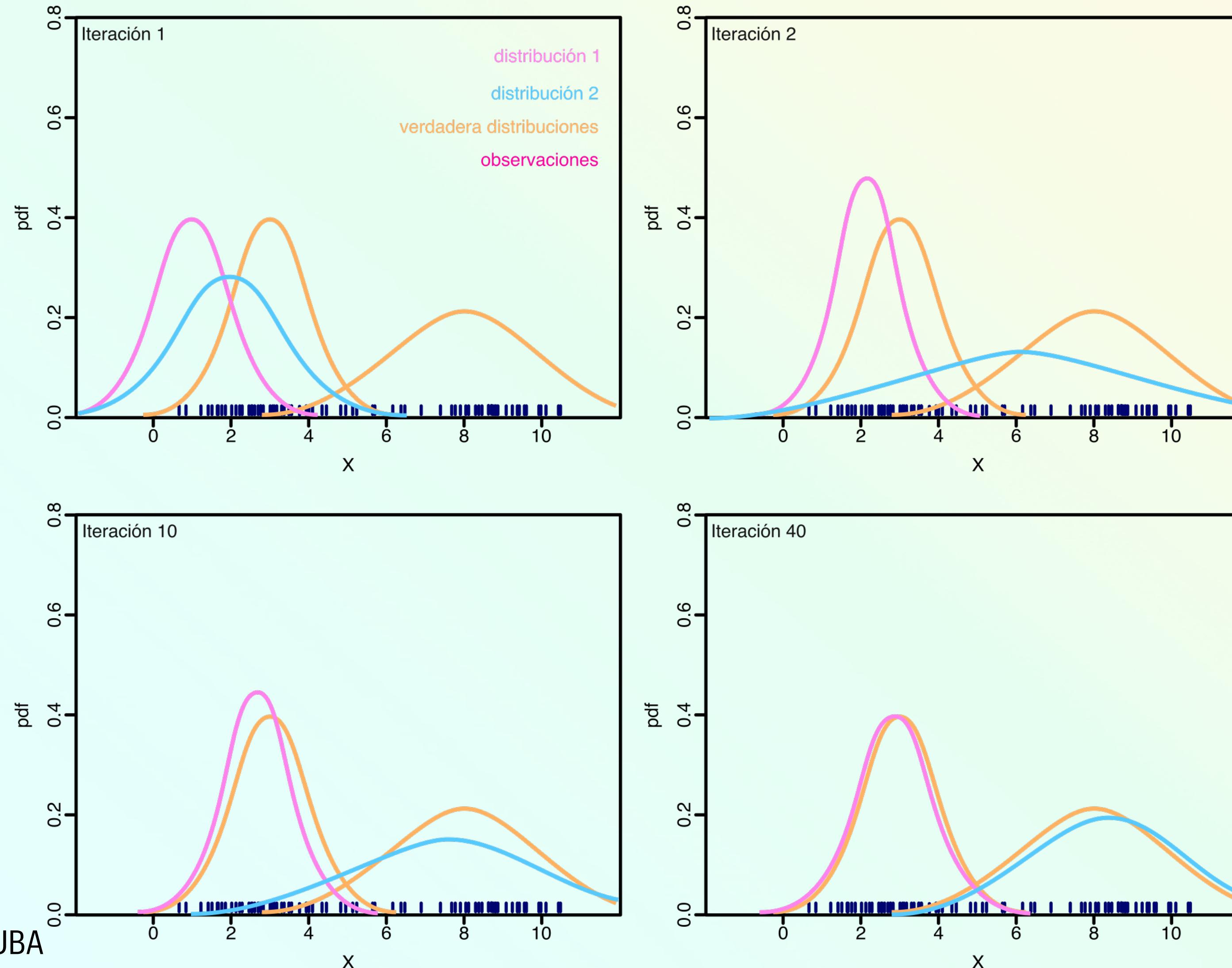
$$\mu_j \leftarrow \frac{\sum_{i=1}^N b_i^{(j)} x_i}{\sum_{i=1}^N b_i^{(j)}} \text{ y } \sigma_j^2 \leftarrow \frac{\sum_{i=1}^N b_i^{(j)} (x_i - \mu_j)^2}{\sum_{i=1}^N b_i^{(j)}}$$

4. Actualiza ϕ_j

$$\phi_j \leftarrow \frac{1}{N} \sum_{i=1}^N b_i^{(j)}$$

Este algoritmo continúa hasta que los valores de μ_j y σ_j^2 no cambian más (se establece un valor umbral).

MODELO DE MIXTURA GAUSSIANA

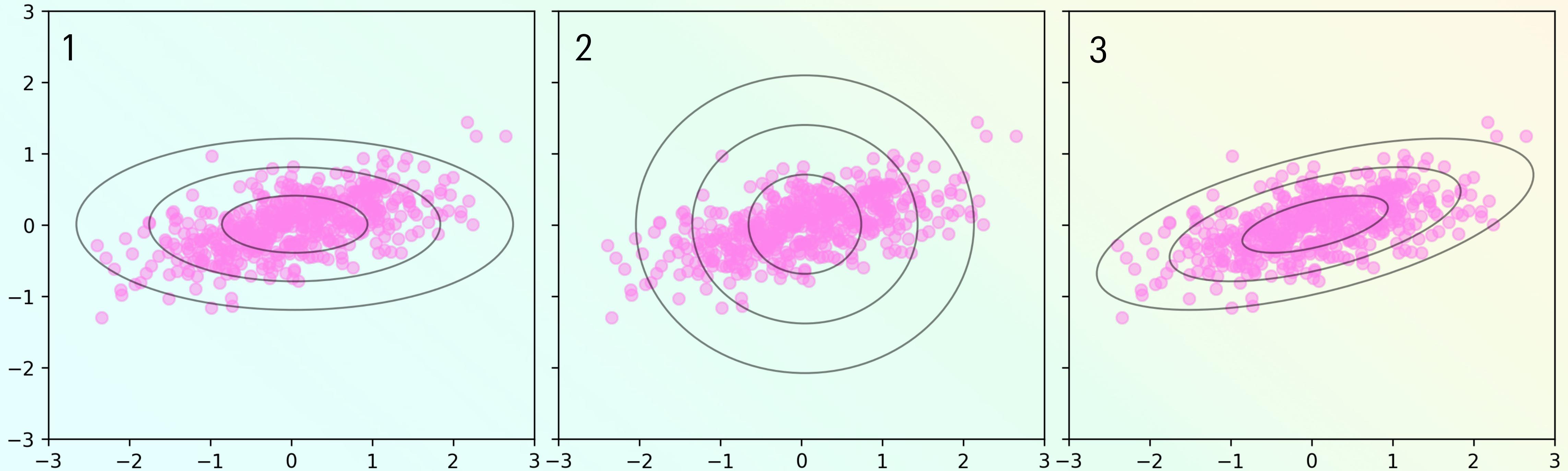


MODELO DE MIXTURA GAUSSIANA

Un detalle importante a tener en el caso multidimensional es que tenemos la matriz de covarianzas y a esta la podemos definir de diferentes formas (que terminaran siendo un hiperparámetro más):

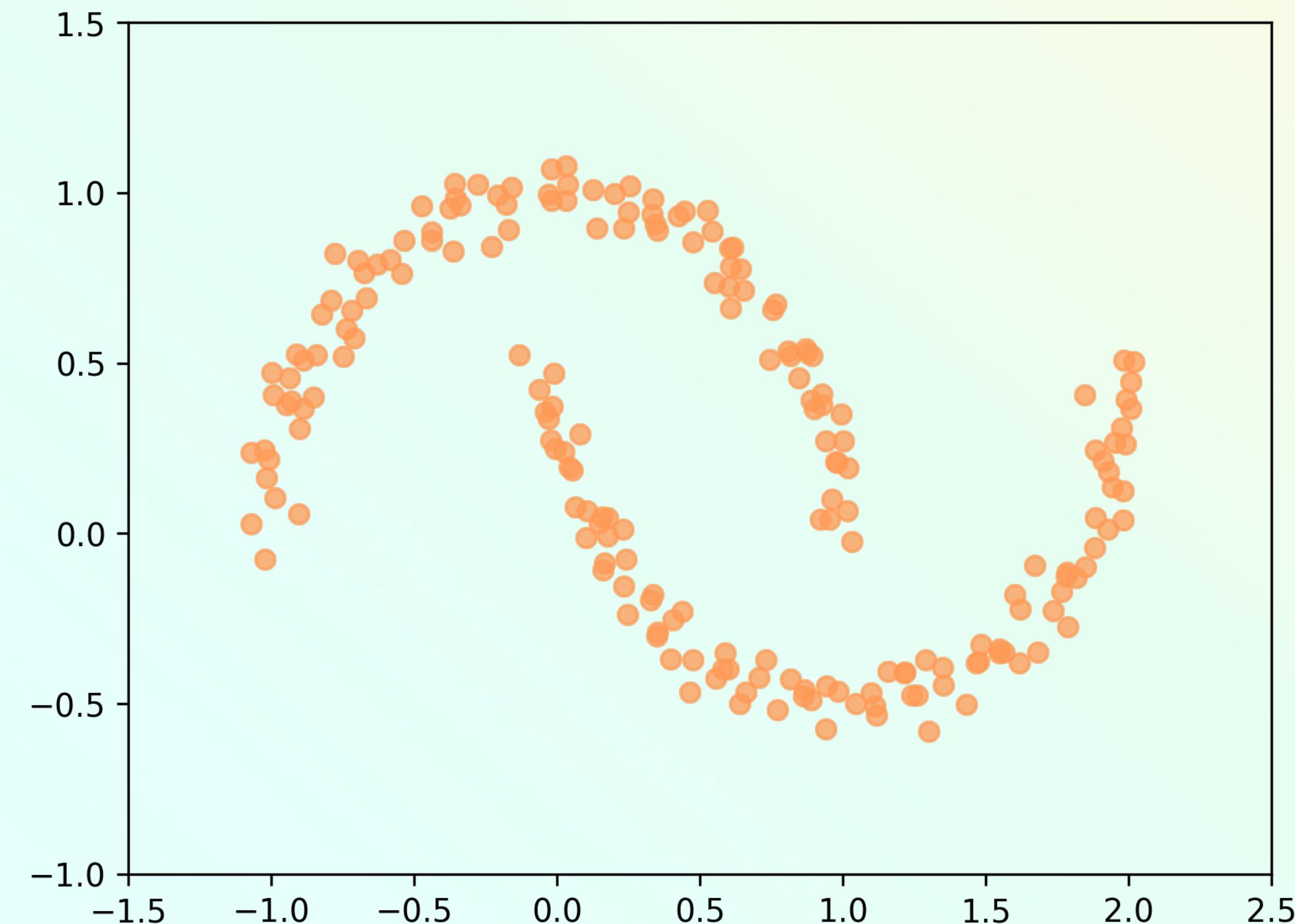
1. El tamaño del cluster va a ser independiente para cada dimensión.
2. El tamaño del cluster es igual en todas dimensiones (establece clusters similares a los de k-means).
3. Hay covarianza entre los atributos haciendo que la distribución se oriente en cualquier sentido. Es el más caro de calcular computacionalmente.

MODELO DE MIXTURA GAUSSIANA



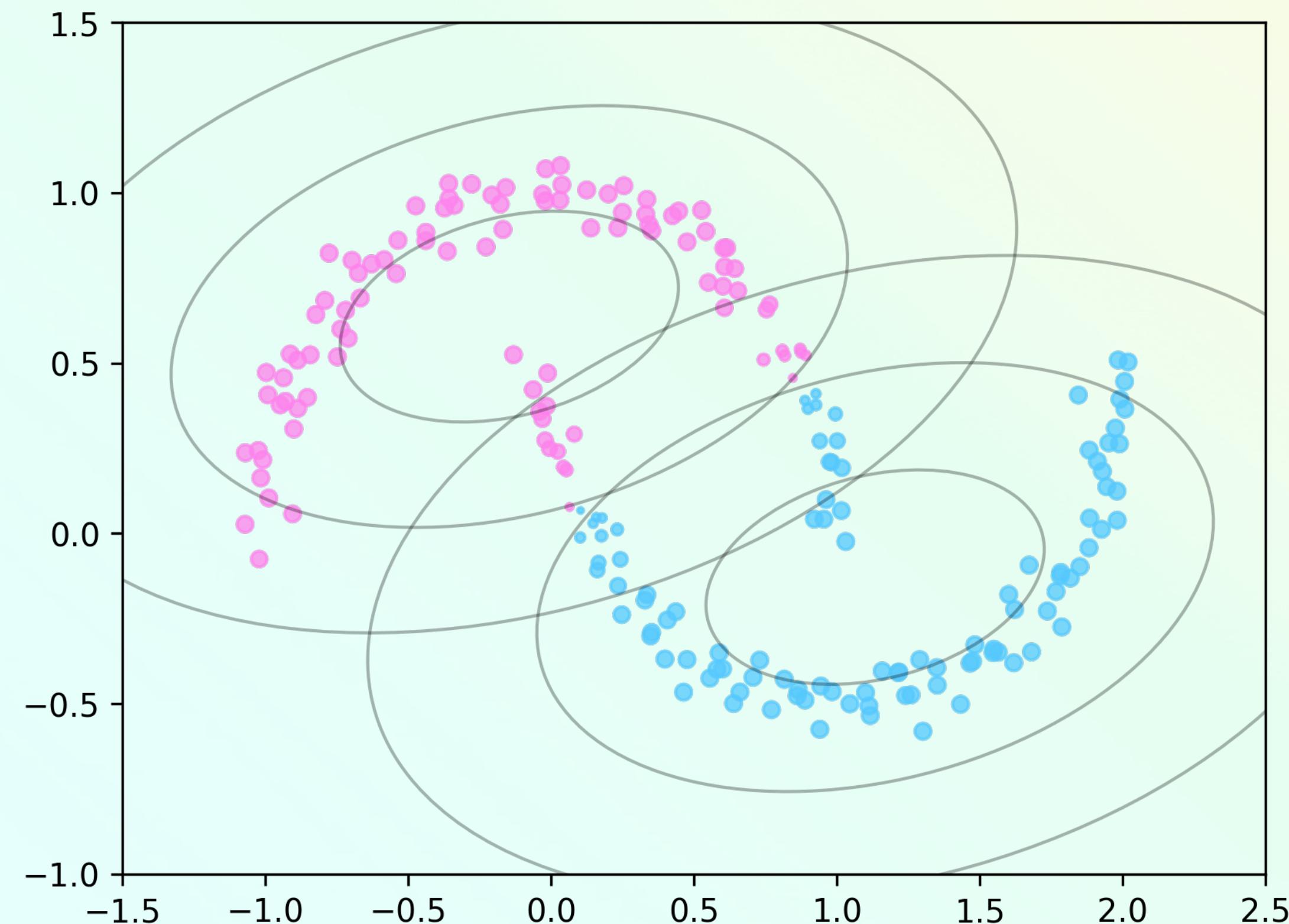
MODELO DE MIXTURA GAUSSIANA

Estos modelos pueden servirnos no solo para armar clusters, sino para **estimar densidades**.



MODELO DE MIXTURA GAUSSIANA

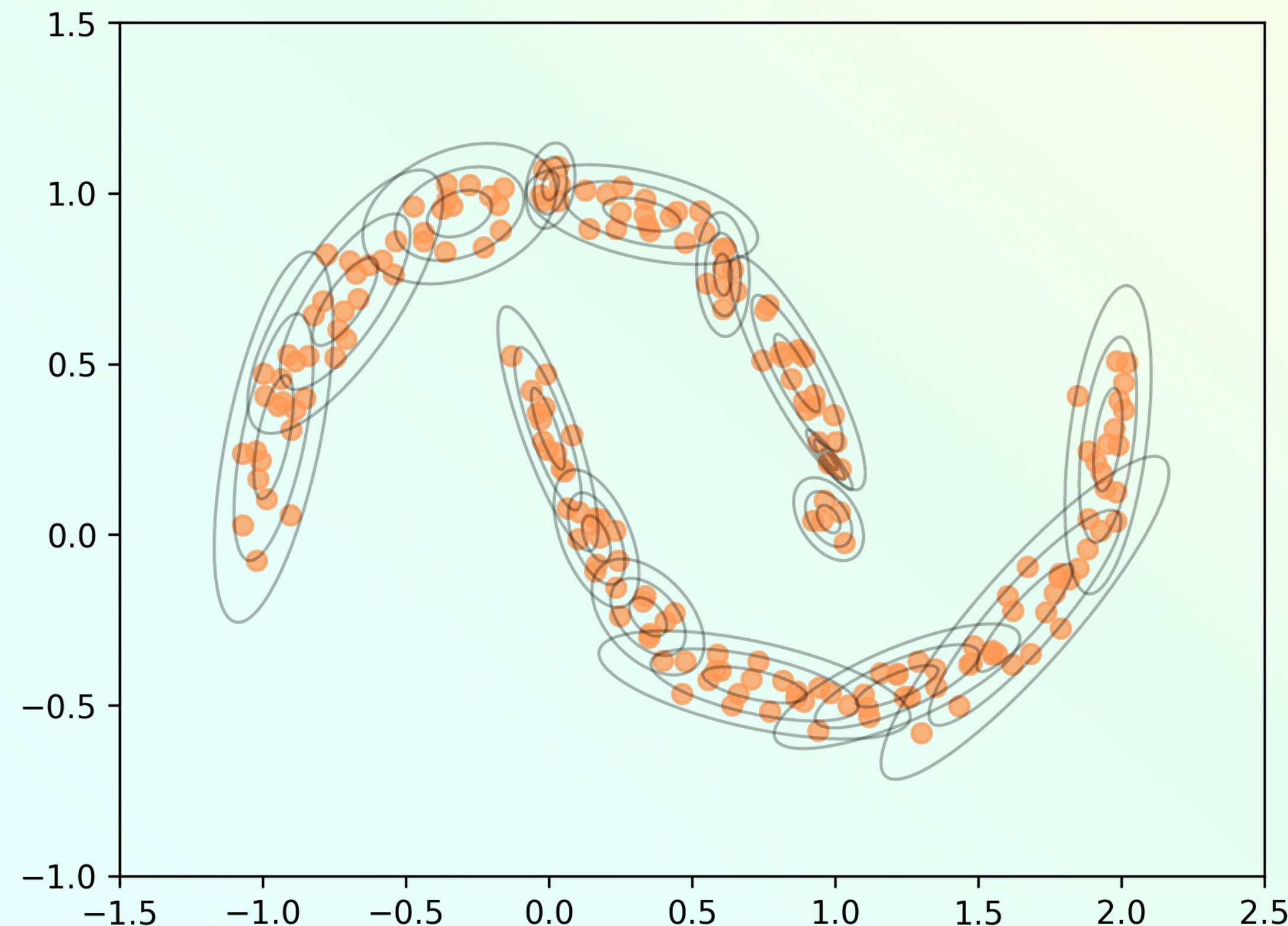
Estos modelos pueden servirnos no solo para armar clusters, sino para **estimar densidades**.



En este caso no genera buenos clusters.

MODELO DE MIXTURA GAUSSIANA

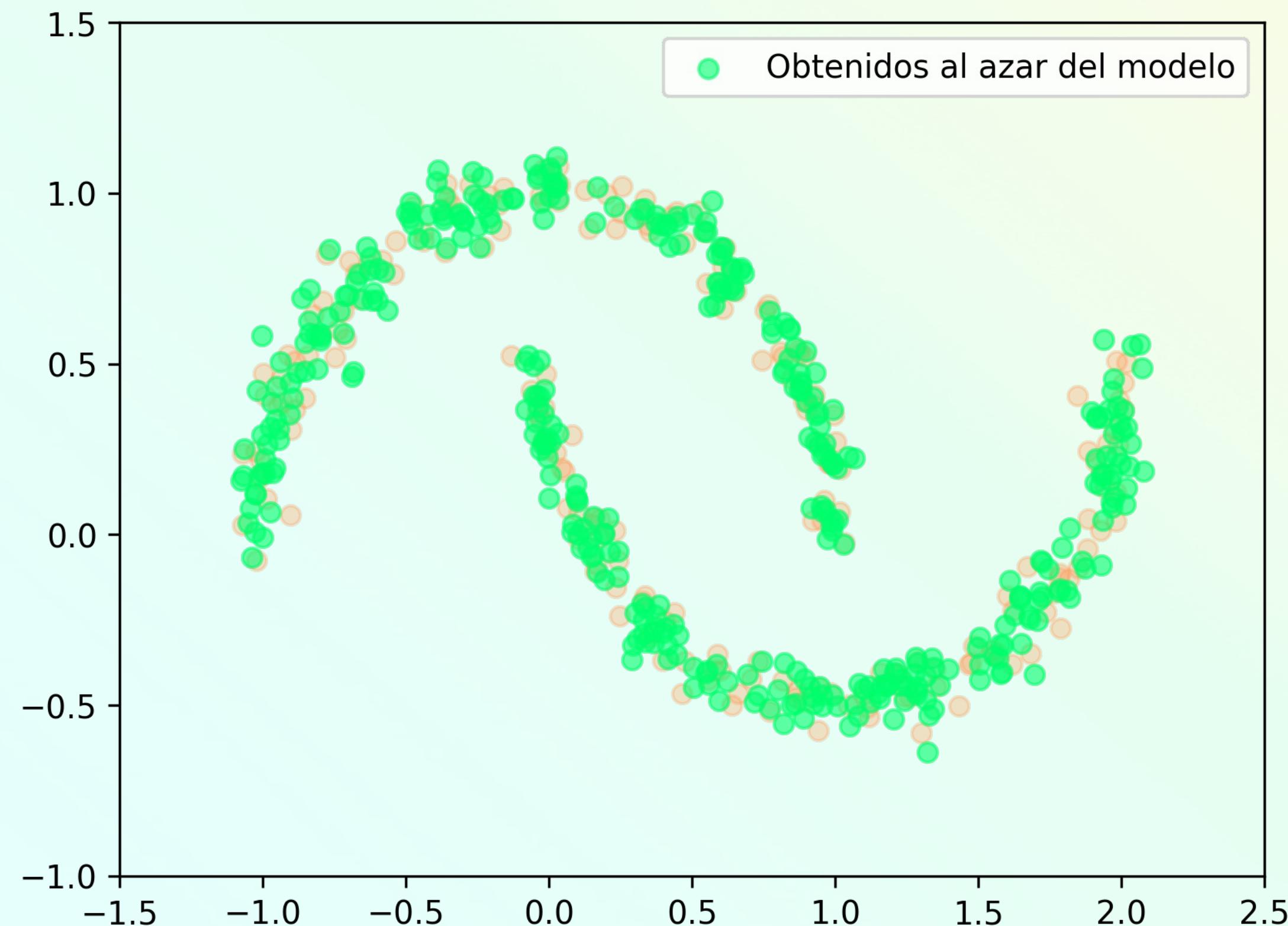
Estos modelos pueden servirnos no solo para armar clusters, sino para **estimar densidades**.



Pero podemos usarlo para modelar la distribución (i.e. con 16 componentes)

MODELO DE MIXTURA GAUSSIANA

Estos modelos pueden servirnos no solo para armar clusters, sino para **estimar densidades**.



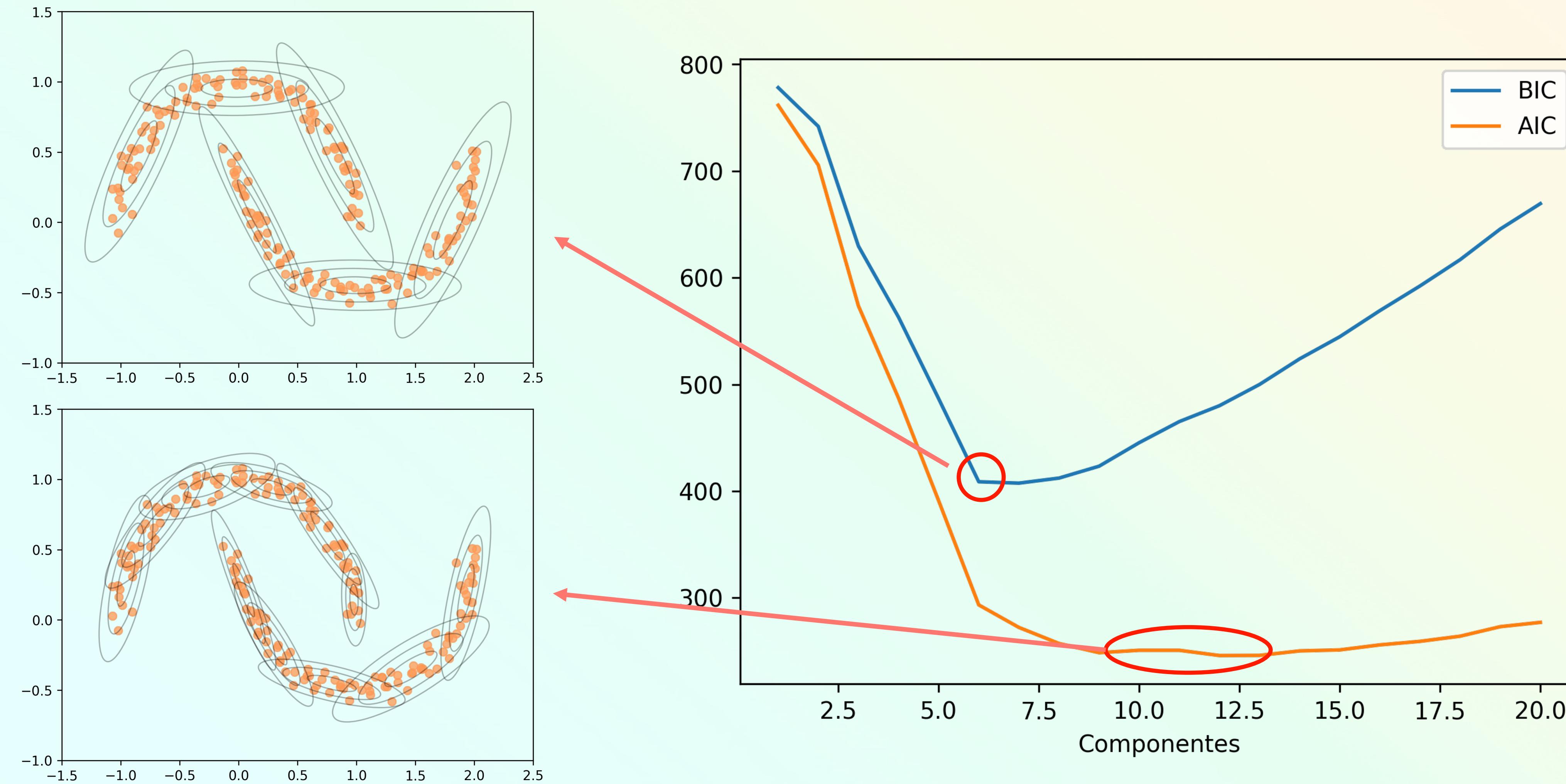
Y una vez obtenido el modelo, podemos generar nuevas muestras al azar de nuestra distribución.

MODELO DE MIXTURA GAUSSIANA

Similar al caso de k-means, establecer la cantidad de clusters es un paso inicial para aplicar este modelo. Podemos resolverlo con las técnicas que vimos, particularmente con **Fuerza de predicción**.

Para el caso de estimación de densidad se puede hacer uso del criterio de información de Aikake (AIC) o el criterio de información Bayesiana (BIC).

MODELO DE MIXTURA GAUSSIANA



VAMOS A PRÁCTICAR UN POCO...

REDUCCIÓN DE DIMENSIONALIDAD

REDUCCIÓN DE DIMENSIONALIDAD

El caso de uso más frecuente para la reducción de dimensionalidad es la visualización de datos: **Podemos ver hasta en un máximo de tres dimensiones.**

Otros beneficios son eliminar características redundantes o altamente correlacionadas, y reducir el ruido en los datos.

Las tres técnicas de reducción de dimensionalidad más utilizadas son:

- Análisis de componentes principales (PCA)
- Uniform Manifold Approximation and Projection (UMAP)
- Autoencoders.

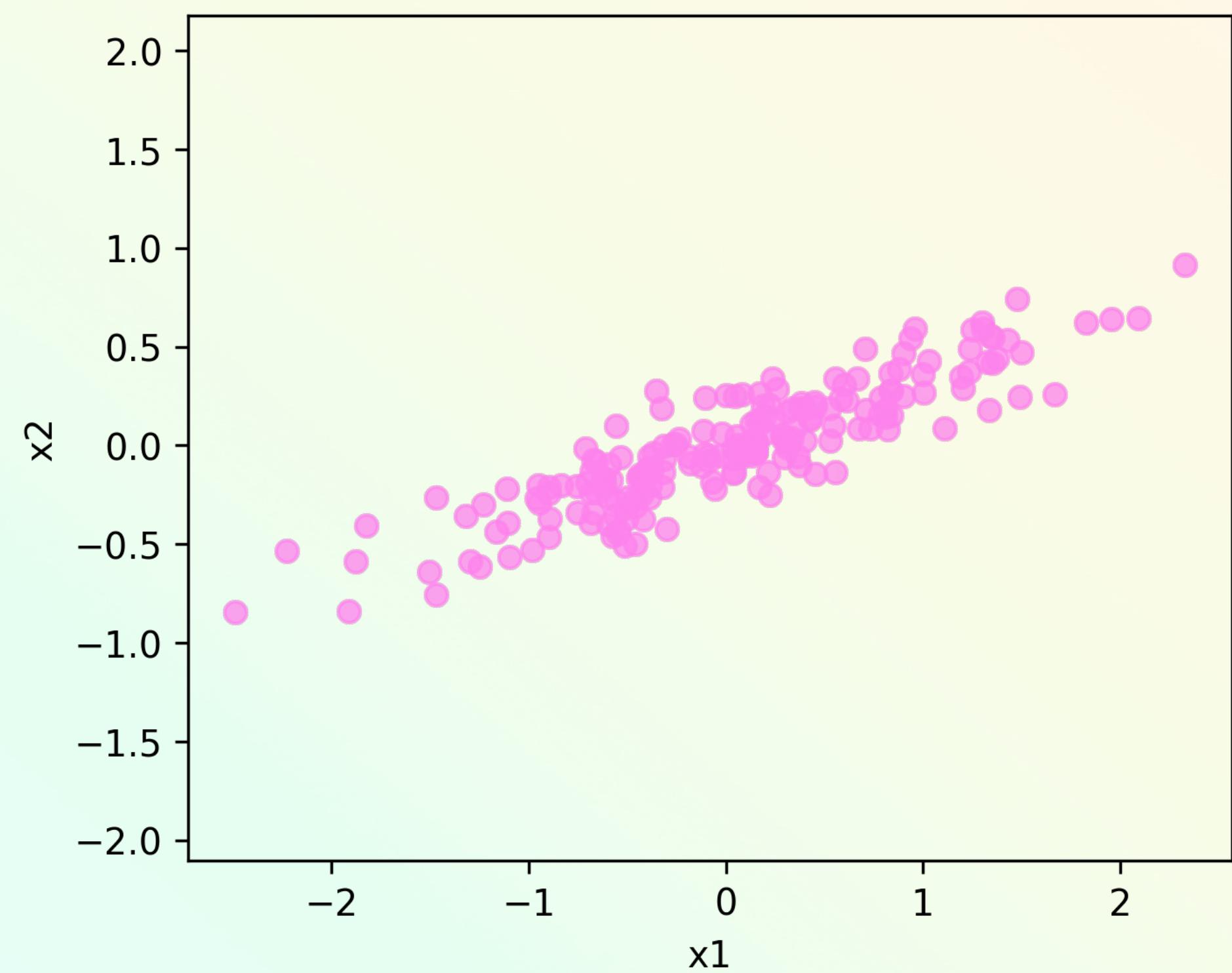
ANÁLISIS DE COMPONENTES PRINCIPALES

ANÁLISIS DE COMPONENTES PRINCIPALES

El comportamiento de PCA es más fácil de visualizar observando un conjunto de datos bidimensional.

Podemos ver que hay una alta correlación lineal entre las dos variables.

A diferencia de regresión, el planteo del problema aquí es ligeramente diferente: en lugar de intentar predecir los valores de x_2 a partir de los valores de x_1 , el problema de aprendizaje no supervisado intenta aprender sobre la relación entre los valores de x_1 e x_2 .

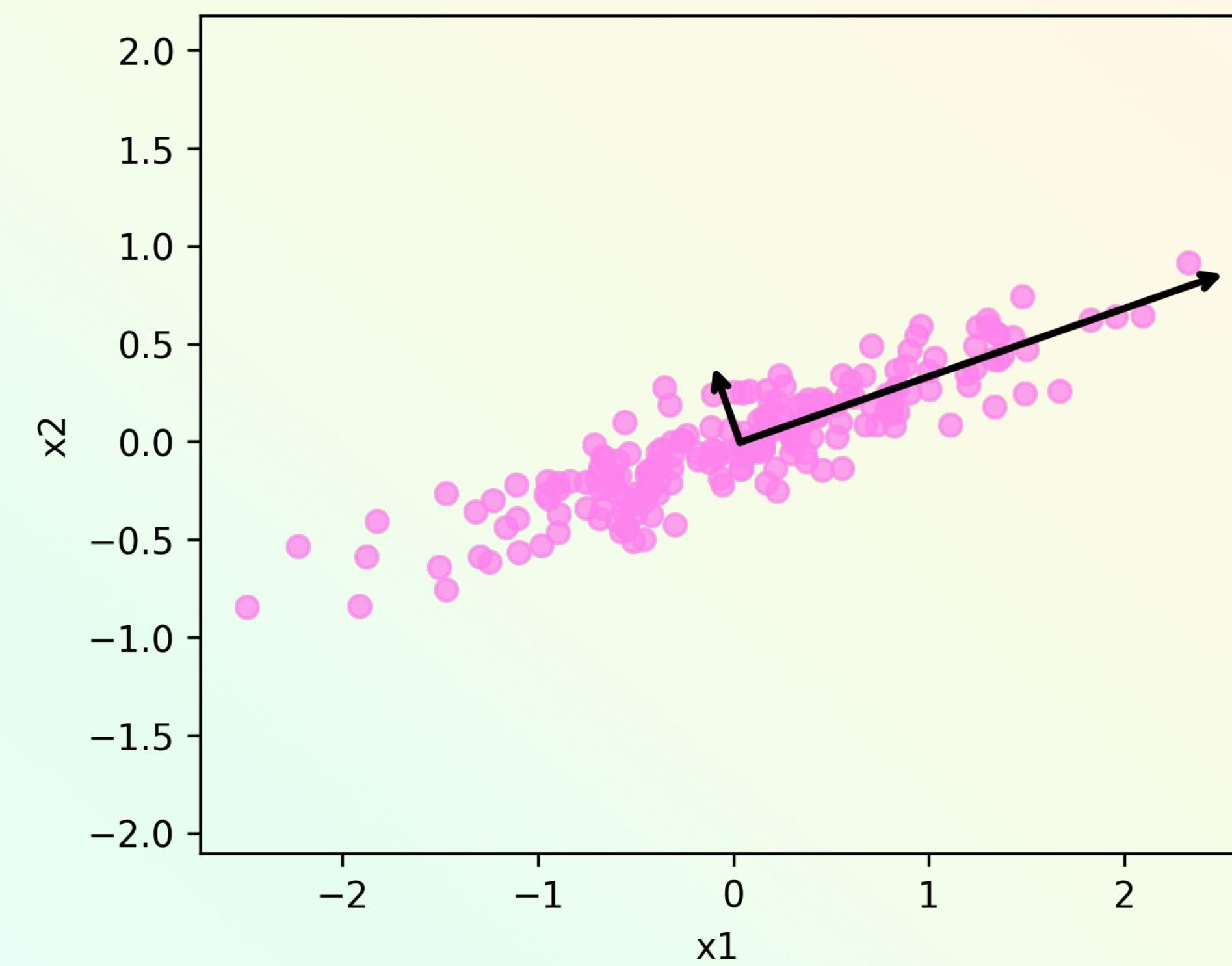


ANÁLISIS DE COMPONENTES PRINCIPALES

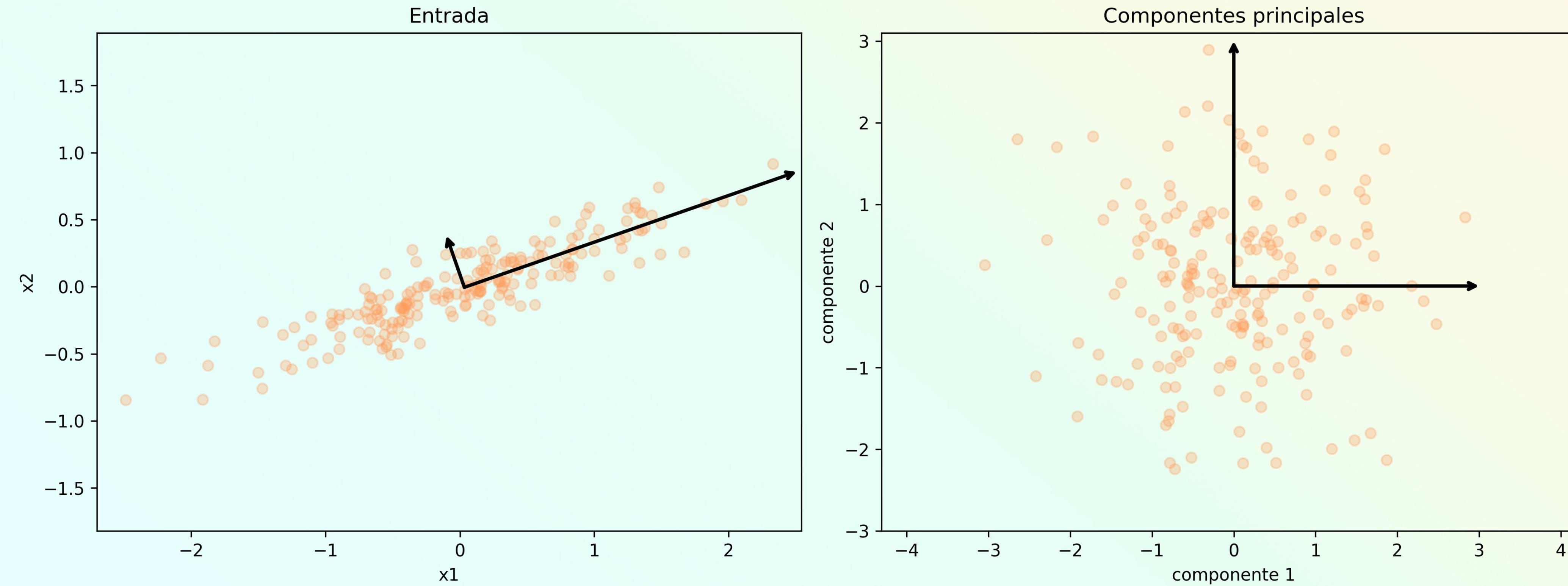
En PCA, esta relación se cuantifica encontrando una lista de los ejes principales en los datos y utilizando esos ejes para describir el conjunto de datos.

Estos vectores representan **los ejes principales de los datos**, y la longitud del vector es una indicación de cuán **importante** es ese eje para describir la distribución de los datos; más precisamente, es una medida de la **varianza de los datos cuando se proyectan**. sobre ese eje.

La proyección de cada punto de datos sobre los ejes principales son los **componentes principales** de los datos.



ANÁLISIS DE COMPONENTES PRINCIPALES

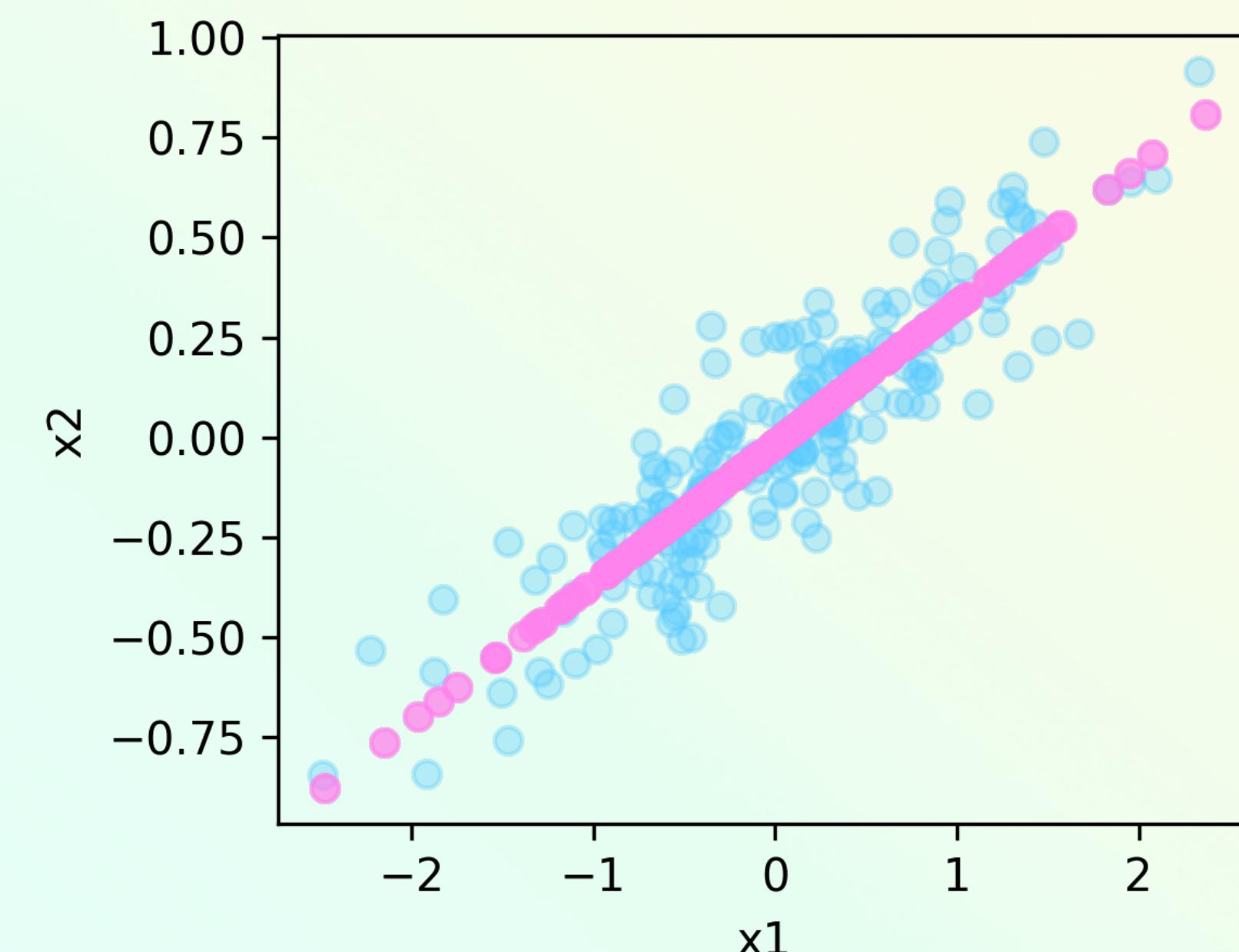
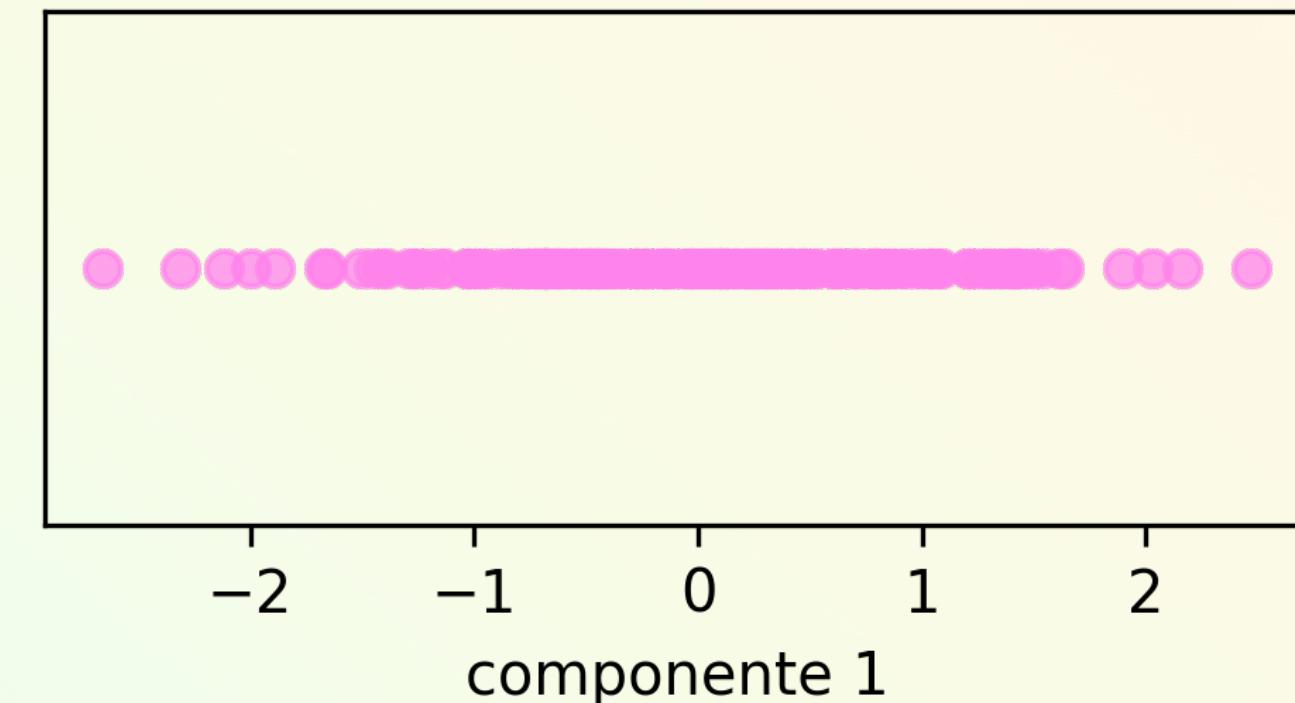


ANÁLISIS DE COMPONENTES PRINCIPALES

El uso de PCA para la reducción de dimensionalidad implica seleccionar los primeros n componentes principales más grande, lo que da como resultado una proyección de los datos de menor dimensión que preserva la mayor varianza de los datos.

La fracción de varianza que se recorta es aproximadamente una medida de cuánta **información se descarta en esta reducción de dimensionalidad**.

¡Estamos **comprimiendo con perdida!**



ANÁLISIS DE COMPONENTES PRINCIPALES

El primer componente de una observación de p atributos $X = (x_1, x_2, \dots, x_p)$ está dado por la combinación lineal de los atributos:

$$z_1 = \phi_{11}x_1 + \phi_{21}x_2 + \cdots + \phi_{p1}x_p$$

Esta combinación lineal está normalizada ($\sum_{j=1}^p \phi_{j1}^2 = 1$). Se restringe a los vectores de carga como $\phi_1 = (\phi_{11}, \phi_{21}, \dots, \phi_{p1})$ para que su suma de cuadrados sea igual a uno, ya que, de lo contrario, establecer estos elementos en un valor absoluto arbitrariamente grande podría dar como resultado una varianza arbitrariamente grande.

ANÁLISIS DE COMPONENTES PRINCIPALES

Dado que estamos interesados en la varianza, se asume que cada atributo está centrado para tener media cero. Una vez hecho esto, tomamos la combinación lineal de las observaciones:

$$z_{i1} = \phi_{11}x_{i1} + \phi_{21}x_{i2} + \cdots + \phi_{p1}x_{ip} = \phi_1^T X_i \quad (\|\phi_1\| = 1).$$

El vector de carga de la primera componente resuelve el siguiente problema de optimización:

$$\underset{\phi_1^T}{\text{maximizar}} \left\{ \frac{1}{n} \sum_{i=1}^n (\phi_1^T X_i)^2 \right\} \quad \text{sujeto a} \quad \|\phi_1\| = 1$$

Dado que $\frac{1}{n} \sum_{i=1}^n x_{ij} = 0$, el promedio de z_{11}, \dots, z_{n1} es cero también. Por lo tanto, el objetivo que estamos maximizando es la varianza muestral de las n observaciones de z_{i1} .

ANÁLISIS DE COMPONENTES PRINCIPALES

$$V_1 = \frac{1}{n} \sum_{i=1}^n (\phi_1^T X_i)^2 = \frac{1}{n} \sum_{i=1}^n \phi_1^T X_i X_i^T \phi_1$$

ANÁLISIS DE COMPONENTES PRINCIPALES

$$V_1 = \frac{1}{n} \sum_{i=1}^n (\phi_1^T X_i)^2 = \frac{1}{n} \sum_{i=1}^n \phi_1^T X_i X_i^T \phi_1$$

ϕ_1 no depende de la sumatoria

$$V_1 = \phi_1^T \left(\frac{1}{n} \sum_{i=1}^n X_i X_i^T \right) \phi_1 = \phi_1^T \left(\frac{1}{n} \sum_{i=1}^n X_i X_i^T \right) \phi_1$$

ANÁLISIS DE COMPONENTES PRINCIPALES

$$V_1 = \frac{1}{n} \sum_{i=1}^n (\phi_1^T X_i)^2 = \frac{1}{n} \sum_{i=1}^n \phi_1^T X_i X_i^T \phi_1$$

ϕ_1 no depende de la sumatoria

$$V_1 = \phi_1^T \left(\frac{1}{n} \sum_{i=1}^n X_i X_i^T \right) \phi_1 = \phi_1^T \left(\frac{1}{n} \sum_{i=1}^n X_i X_i^T \right) \phi_1$$

Es la matriz de covarianza

ANÁLISIS DE COMPONENTES PRINCIPALES

$$V_1 = \frac{1}{n} \sum_{i=1}^n (\phi_1^T X_i)^2 = \frac{1}{n} \sum_{i=1}^n \phi_1^T X_i X_i^T \phi_1$$

ϕ_1 no depende de la sumatoria

$$V_1 = \phi_1^T S \phi_1$$

Entonces la optimización es:

$$\underset{\phi_1^T}{\text{maximizar}} \{ \phi_1^T S \phi_1 \} \quad \text{sujeto a} \quad \|\phi_1\| = 1$$

Esto no es más que un problema de optimización que se resuelve con multiplicadores de Lagrange.

ANÁLISIS DE COMPONENTES PRINCIPALES

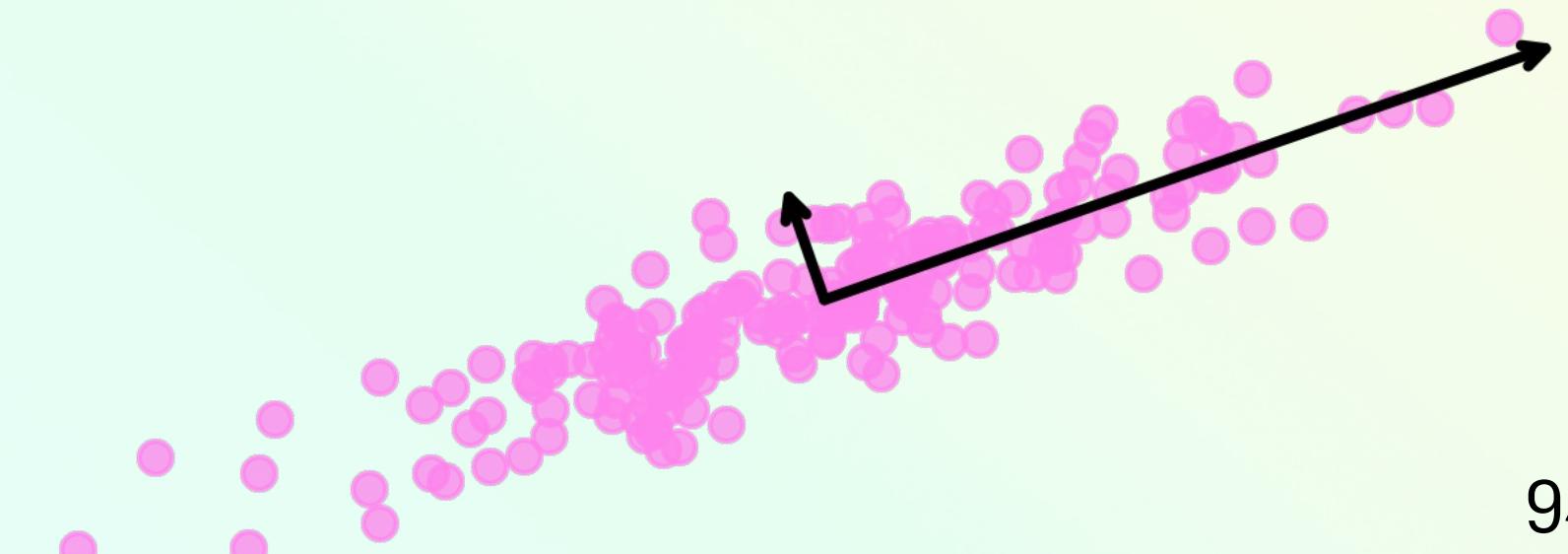
Encontrando el Lagrangiano, derivando e igualando a cero, nos queda el sistema:

$$\begin{aligned} S\phi_1 &= \lambda_1\phi_1 \\ \phi_1^T\phi_1 &= 1 \end{aligned}$$

Esto no es más que la descomposición de valores y vectores propios. ϕ_1 es un vector propio de la covarianza S de los datos, y el multiplicador de Lagrange es el valor propio:

$$V_1 = \phi_1^T S \phi_1 = \phi_1^T \lambda_1 \phi_1 = \lambda_1 \phi_1^T \phi_1 = \lambda_1$$

Es decir, es la varianza de los datos proyectados en un sub-espacio de una dimensión.



ANÁLISIS DE COMPONENTES PRINCIPALES

Encontrando el Lagrangiano, derivando e igualando a cero, nos queda el sistema:

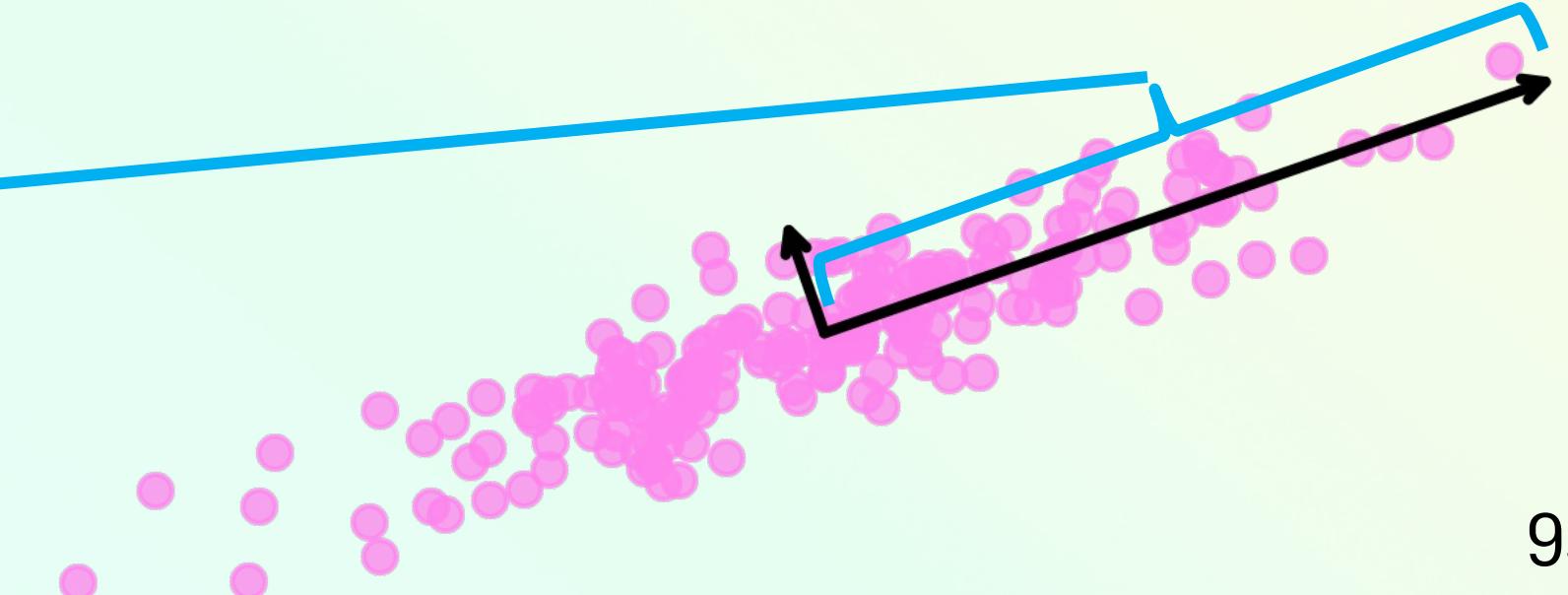
$$\begin{aligned} S\phi_1 &= \lambda_1\phi_1 \\ \phi_1^T\phi_1 &= 1 \end{aligned}$$

Esto no es más que la descomposición de valores y vectores propios. ϕ_1 es un vector propio de la covarianza S de los datos, y el multiplicador de Lagrange es el valor propio:

$$V_1 = \phi_1^T S \phi_1 = \phi_1^T \lambda_1 \phi_1 = \lambda_1 \phi_1^T \phi_1 = \lambda_1$$

Es decir, es la varianza de los datos proyectados en un sub-espacio de una dimensión.

ϕ_1 es la dirección
 λ_1 es el largo del vector relacionado a la varianza



ANÁLISIS DE COMPONENTES PRINCIPALES

De forma similar podemos extender a M componentes principales, es decir buscar M vectores propios y valores propios.

Para determinar el orden de los componentes, se ordenan los valores de λ_m de mayor a menor. Y la varianza explicada por los primeros M componentes es:

$$V_M = \sum_{m=1}^M \lambda_m$$

La varianza perdida se calcula como: $J_M = V_D - V_M$, donde V_D es la varianza de los datos.

O la varianza relativa perdida: $1 - V_M/V_D$

VAMOS A PRÁCTICAR UN POCO...