



# APRENDIZAJE NO SUPERVISADO

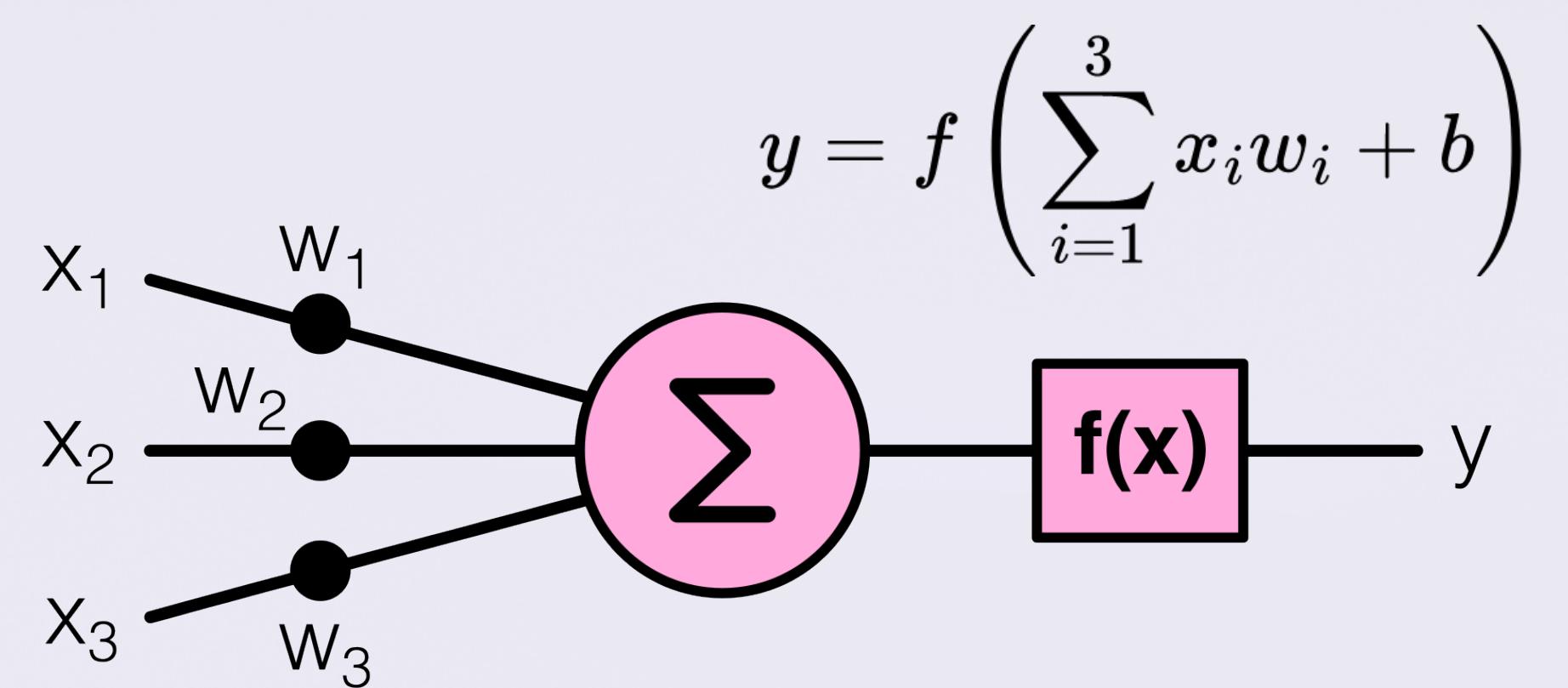
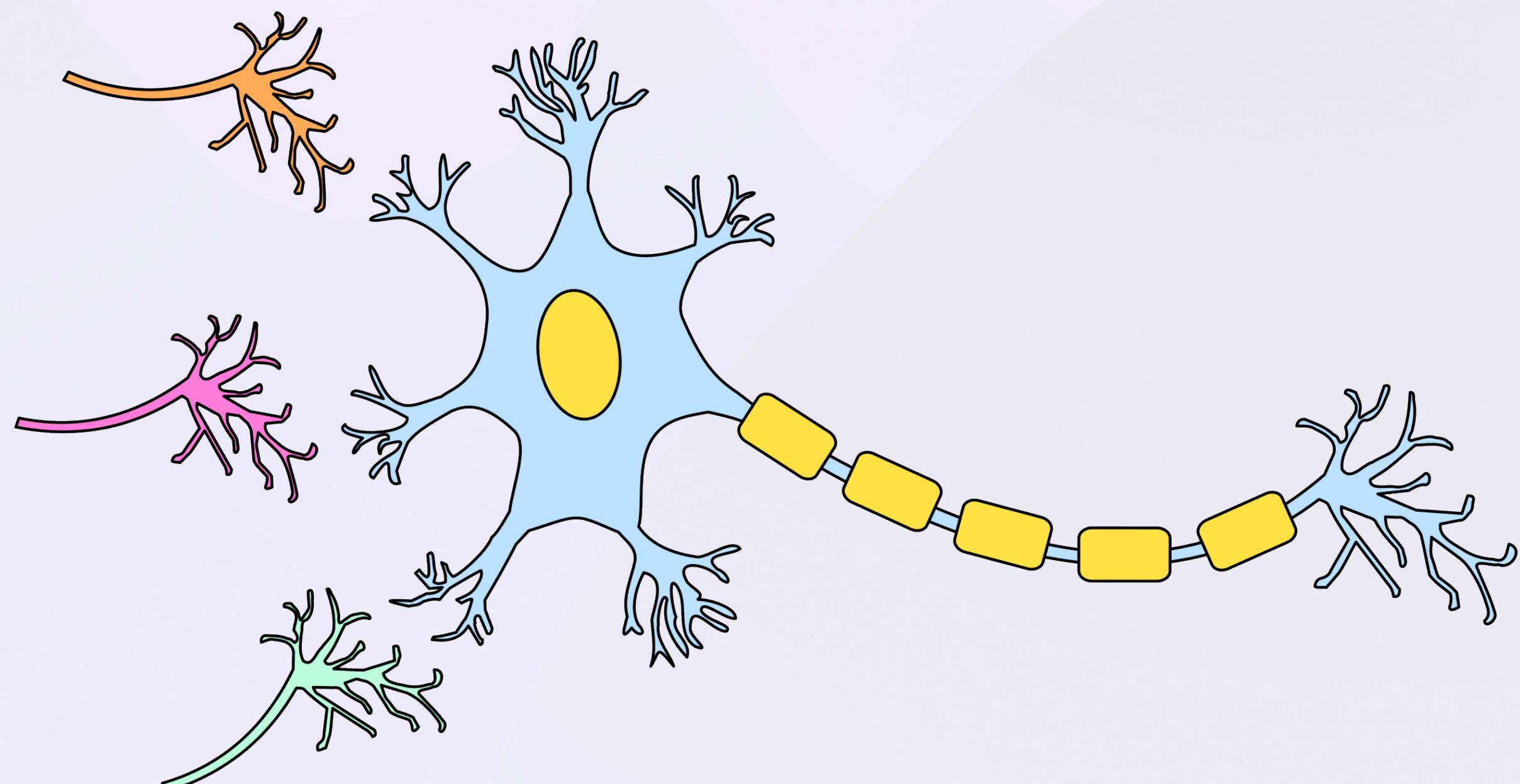
APRENDIZAJE DE MAQUINA I - CEIA - FIUBA

Dr. Ing. Facundo Adrián Lucianna

# REPASO CLASE ANTERIOR

- Perceptron y neuronas sigmoideas
- Funciones de activación
- Redes feed-forward
- Breve introducción a Backpropagation
- Pytorch

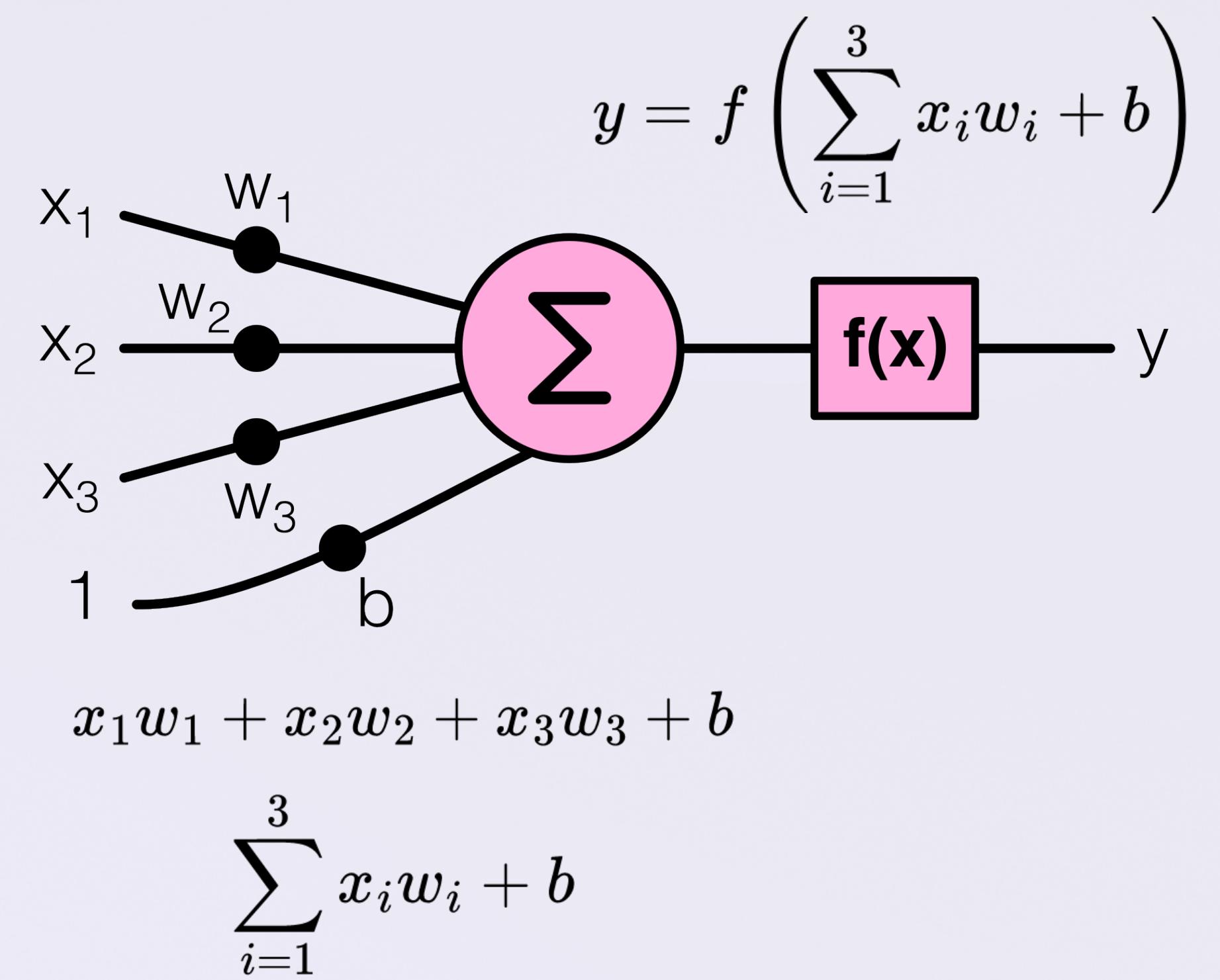
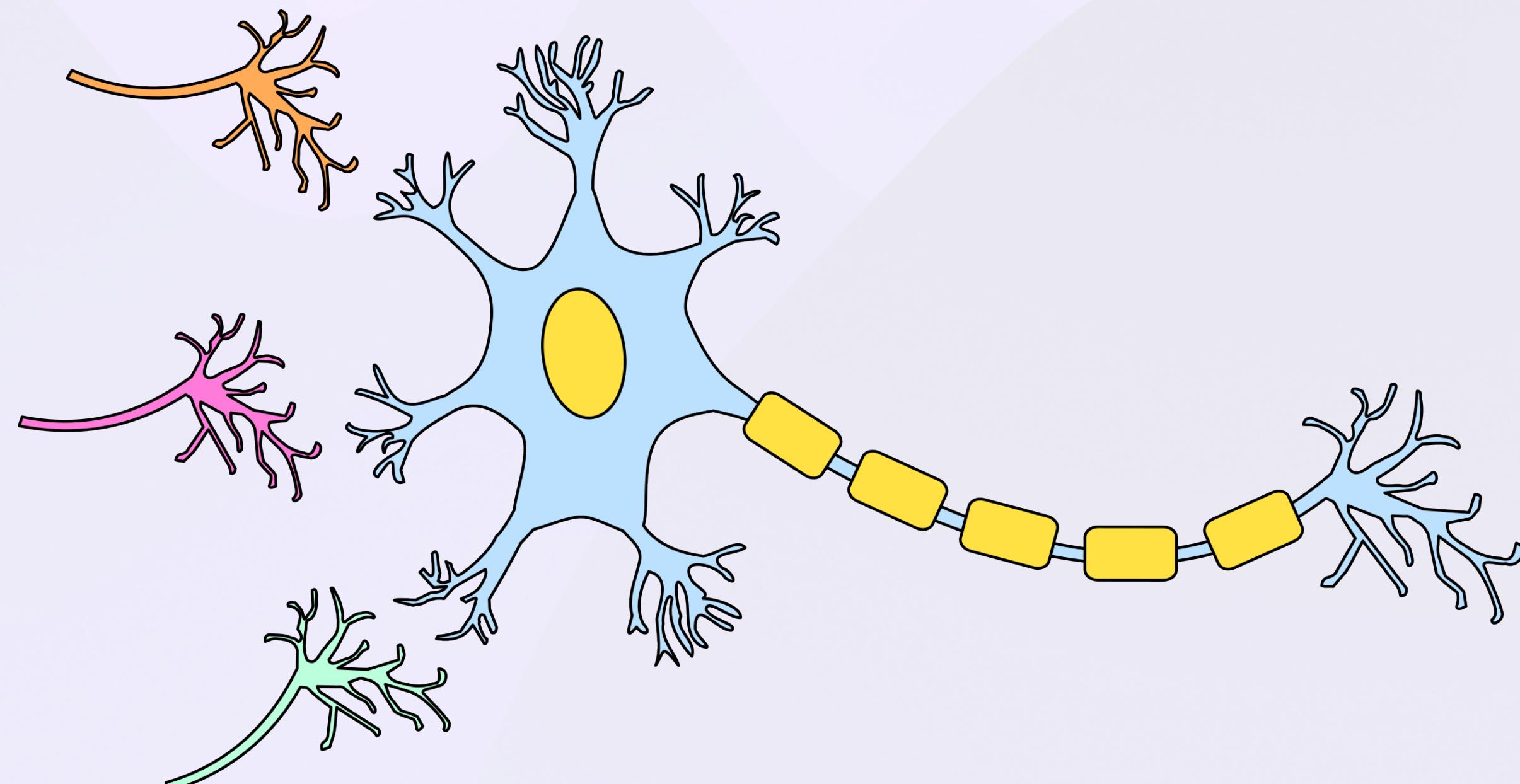
# PERCEPTRÓN



$$x_1 w_1 + x_2 w_2 + x_3 w_3 + b$$

$$\sum_{i=1}^3 x_i w_i + b$$

# PERCEPTRÓN

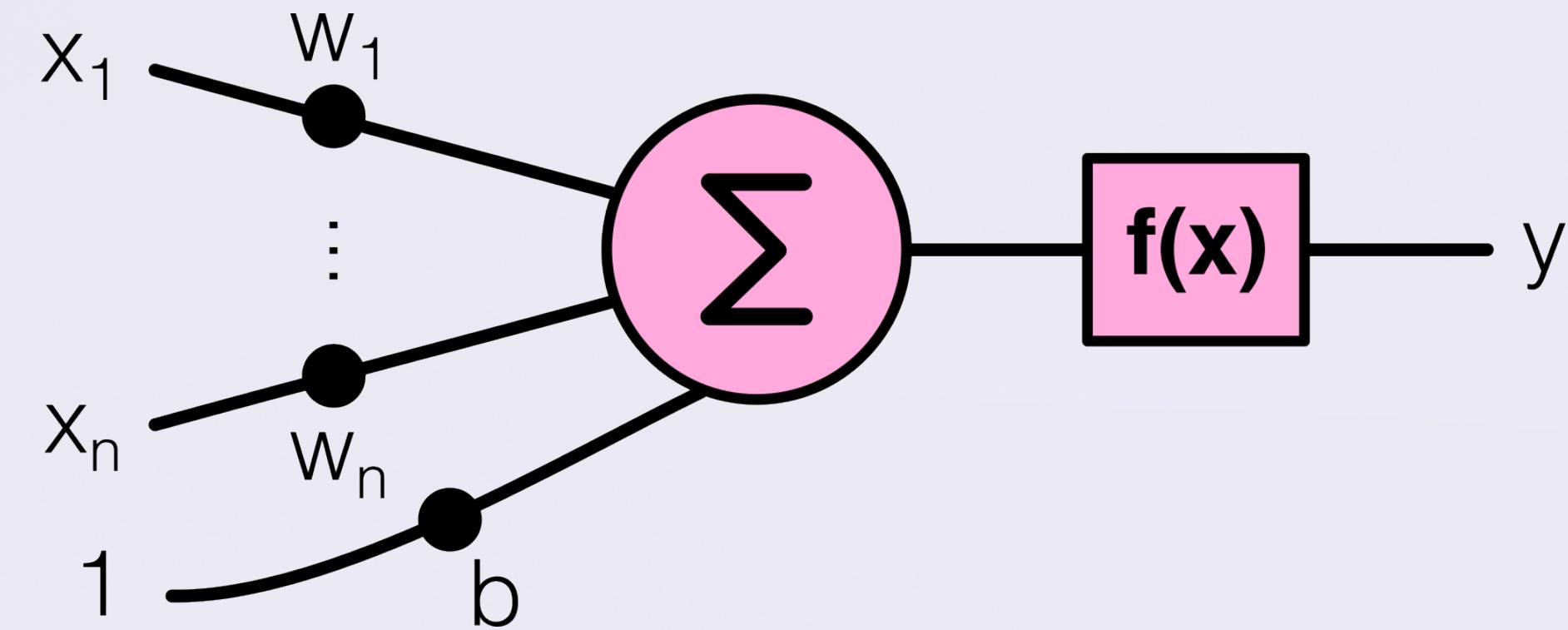


# PERCEPTRÓN

Si tenemos una observación de **n** atributos, la neurona tendrá **n** entradas con **n+1** pesos sinápticos.

Por lo que la parte lineal es:

$$\sum_{i=1}^n x_i w_i + b$$

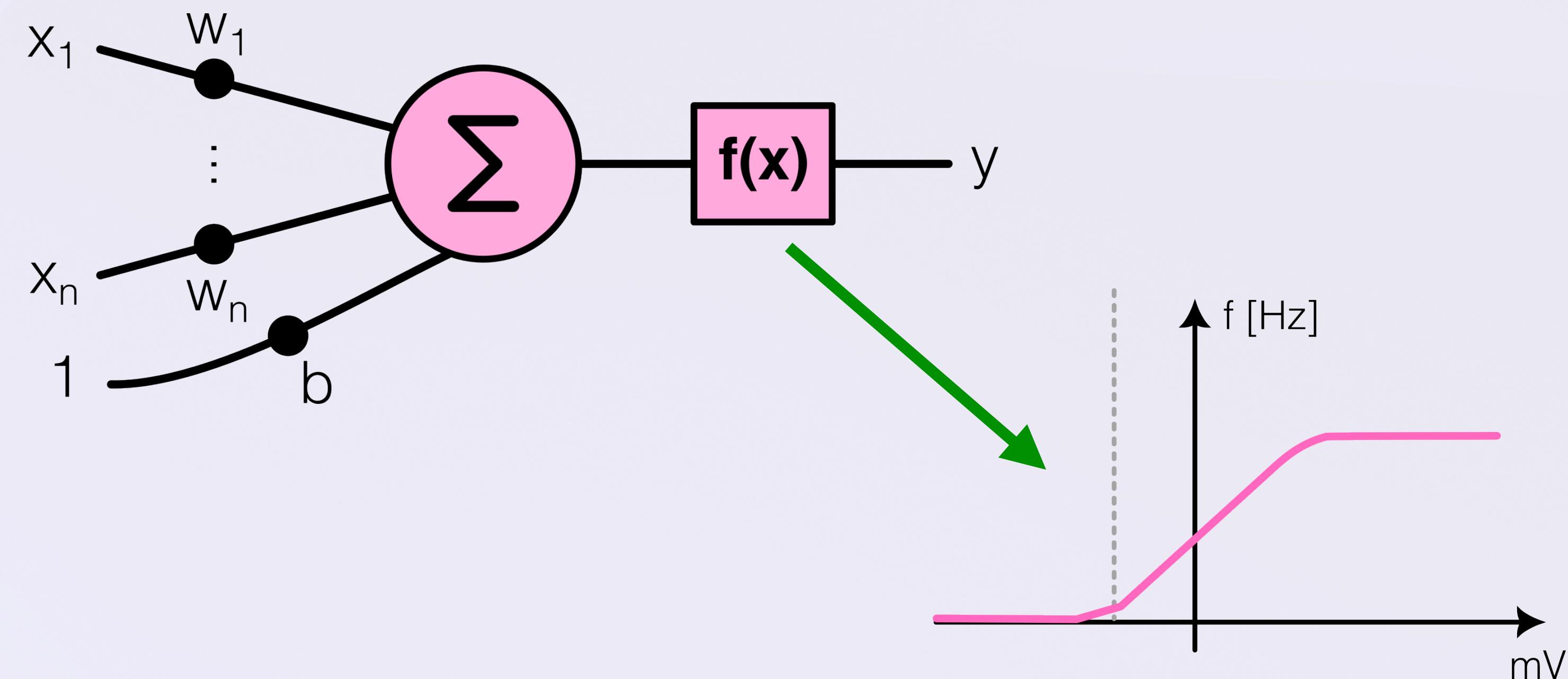


Donde  $w_i$ ,  $b$  son los pesos sinápticos, que representan si la conexión es excitatoria ( $w_i > 0$ ) o inhibitoria ( $w_i < 0$ ), o que no haya conexión sináptica ( $w_i = 0$ )

# PERCEPTRÓN

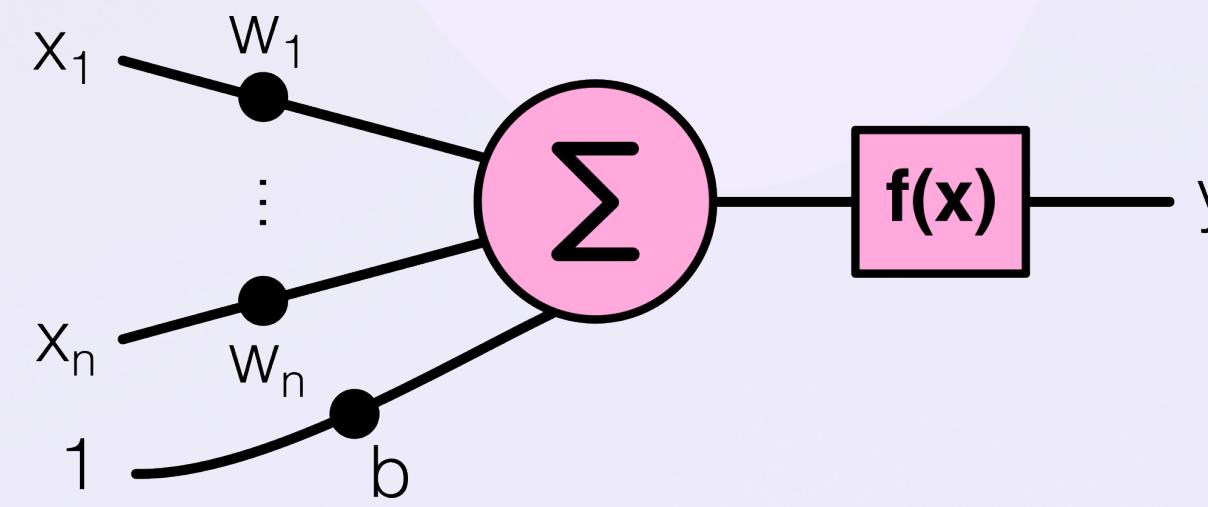
La función de activación es una función no lineal que modela la variación de la tasa de disparo de la neurona.

Y con esto último, tenemos la expresión mínima de una neurona, el cual es una **unidad de cálculo no lineal**.



# PERCEPTRÓN

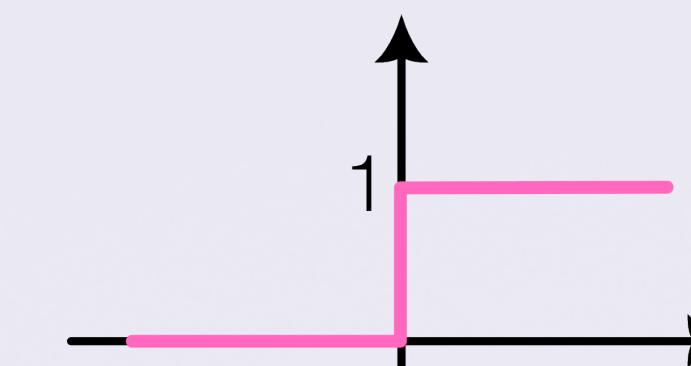
La función de activación que se usa en redes neuronales es una decisión de diseño:



$$y = f \left( \sum_{i=1}^n x_i w_i + b \right)$$

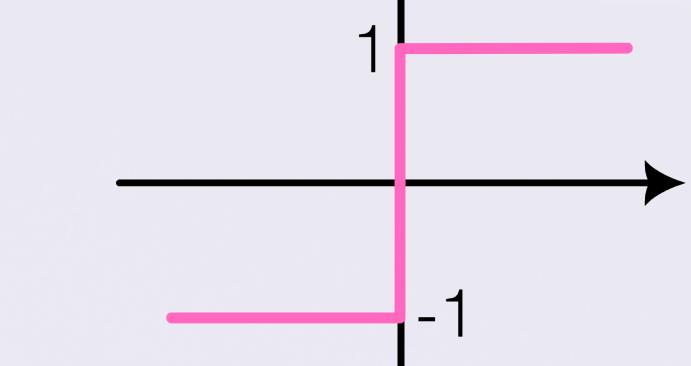
Función escalón

$$f(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases}$$



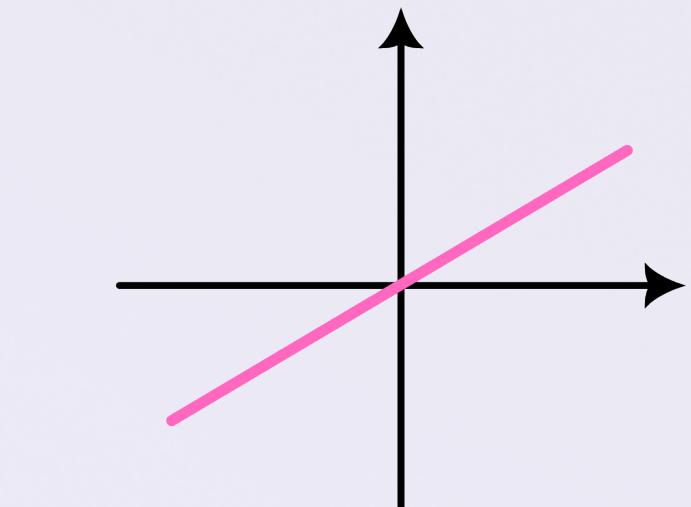
Función signo

$$f(x) = \begin{cases} -1 & x < 0 \\ 1 & x \geq 0 \end{cases}$$



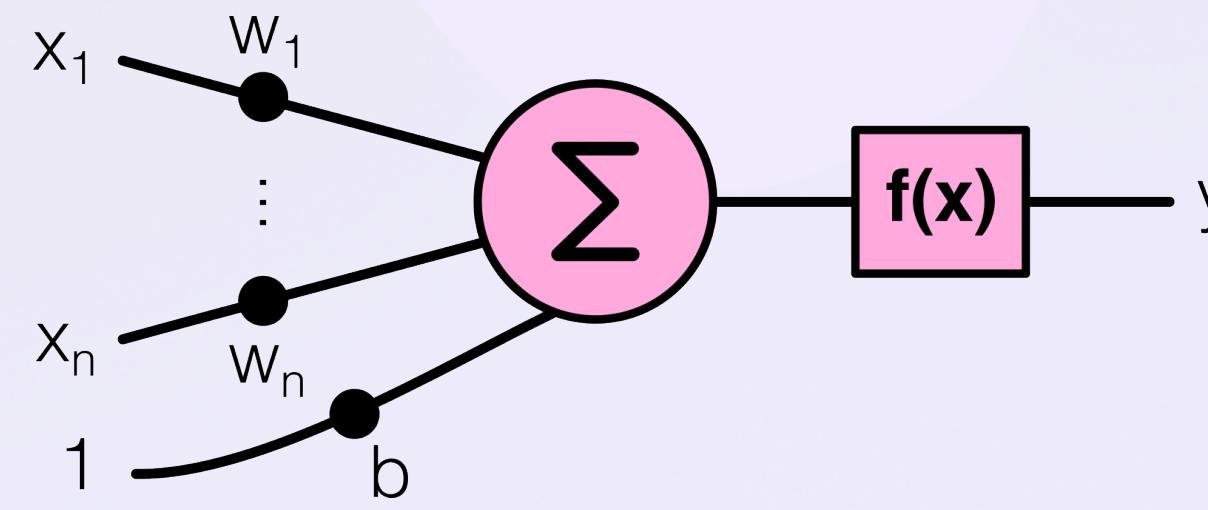
Función lineal

$$f(x) = x$$



# PERCEPTRÓN

La función de activación que se usa en redes neuronales es una decisión de diseño:



$$y = f \left( \sum_{i=1}^n x_i w_i + b \right)$$

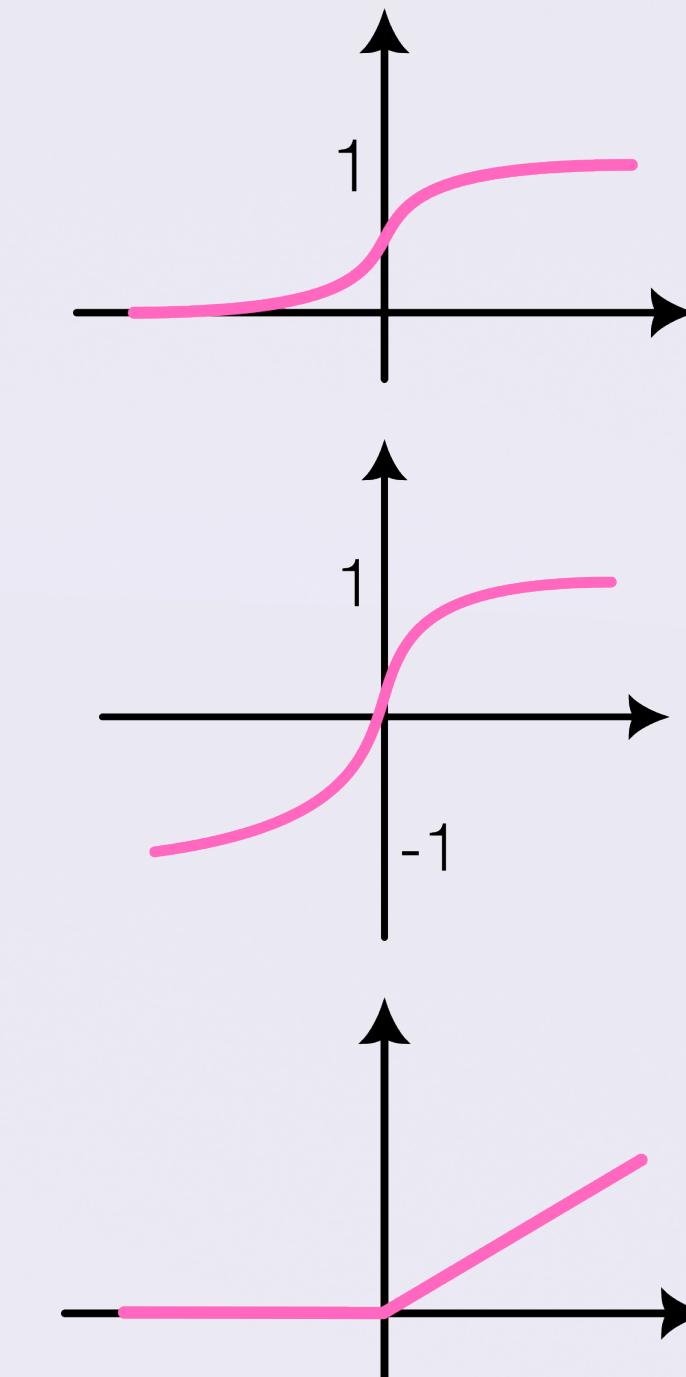
Funciones sigmoideas

$$f(x) = \frac{e^x}{1 + e^x}$$

$$f(x) = \tanh(x)$$

Función ReLu

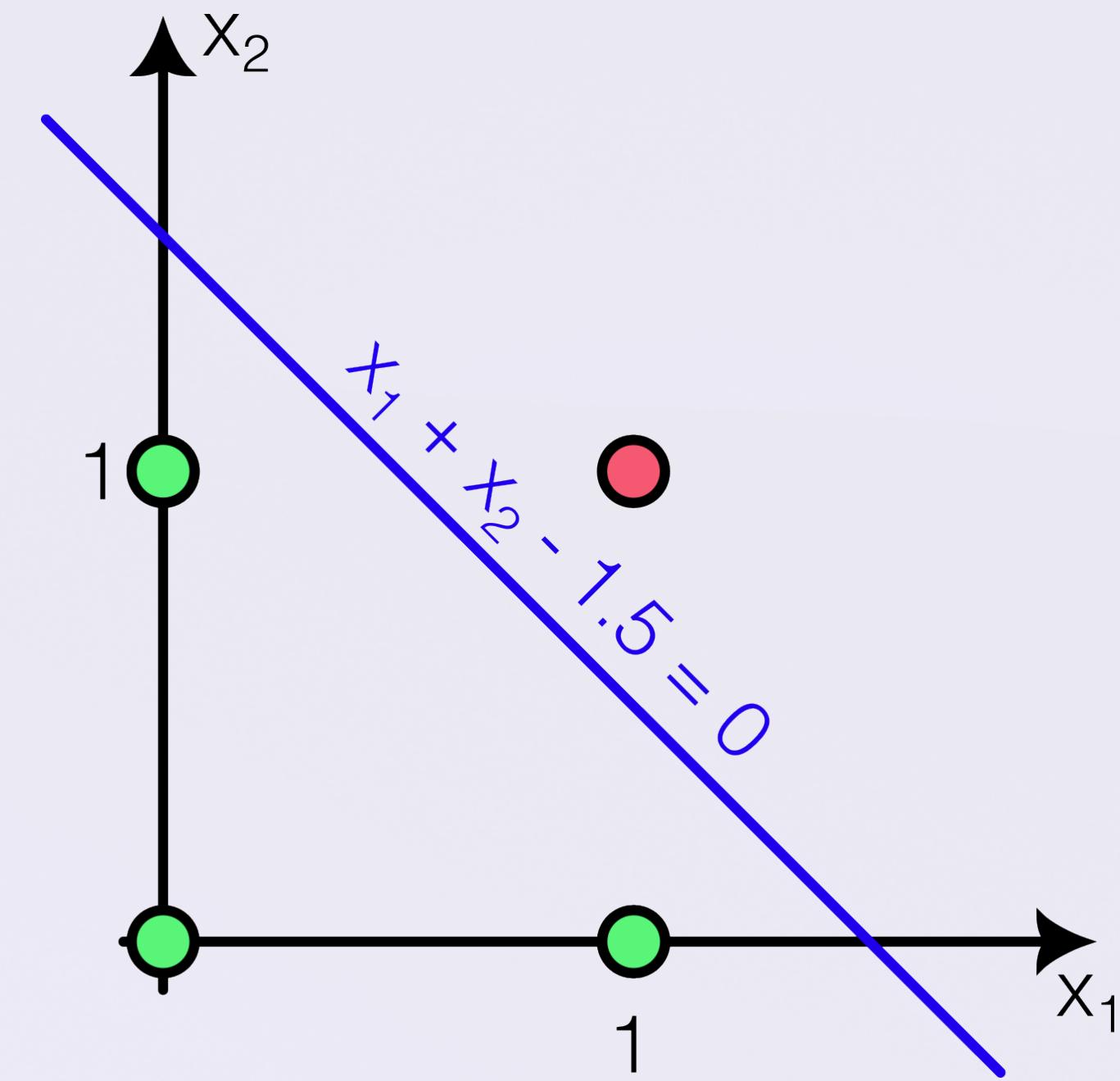
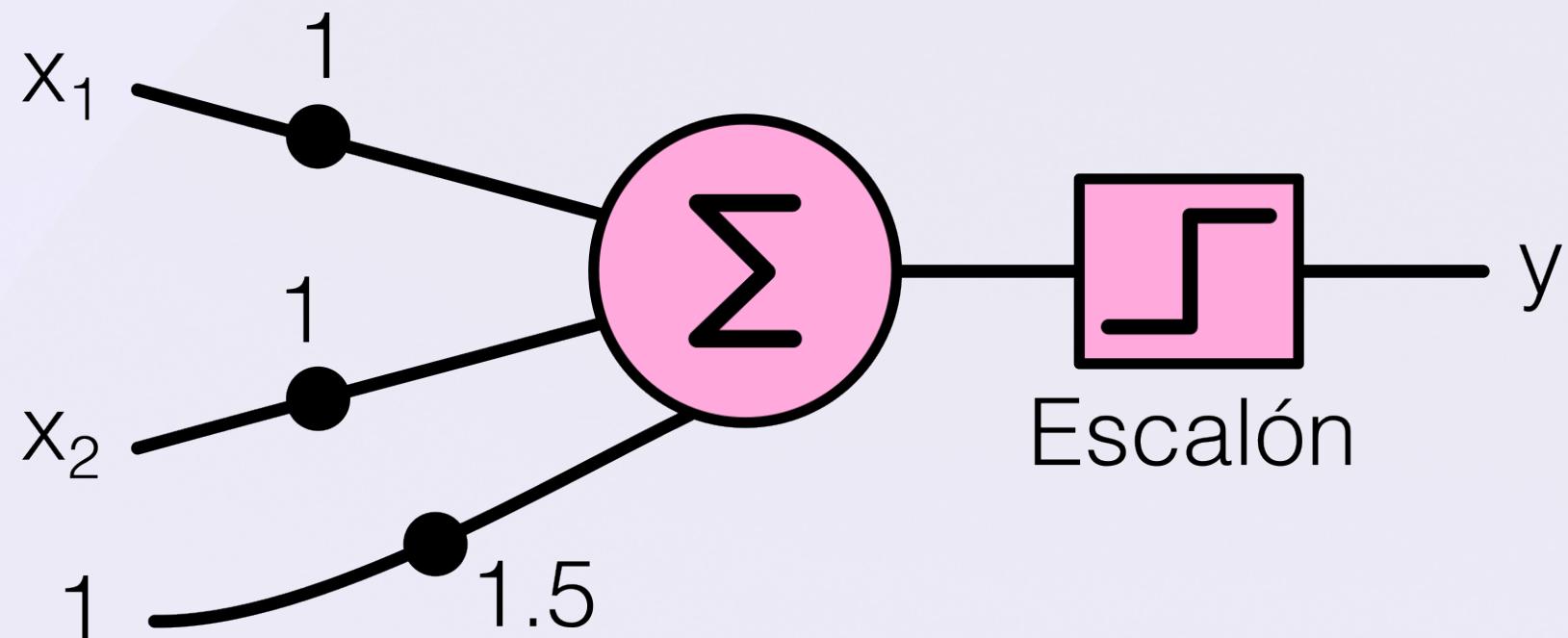
$$f(x) = \begin{cases} 0 & x < 0 \\ x & x \geq 0 \end{cases}$$



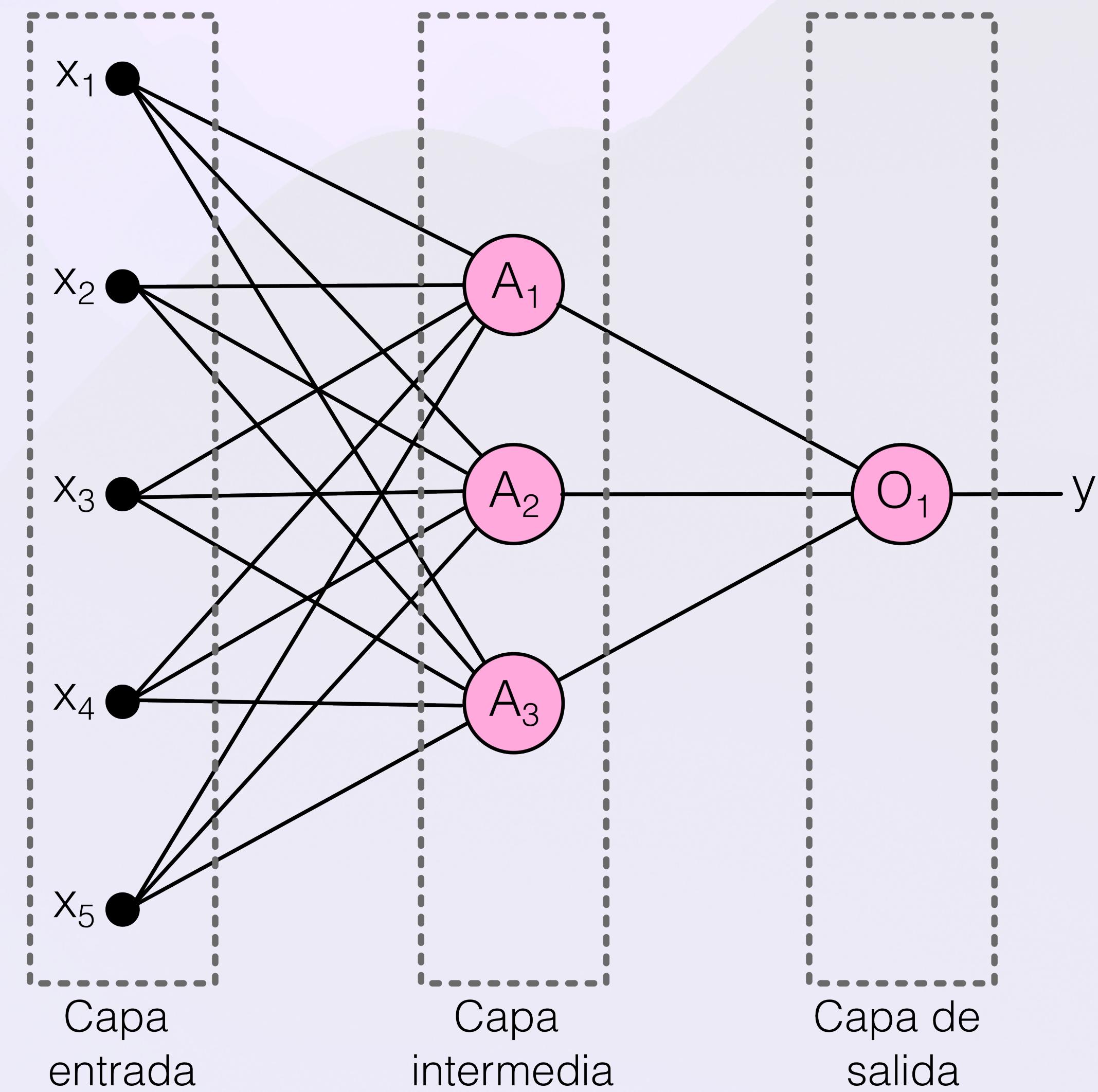
# PERCEPTRÓN

**AND** (2 entradas)

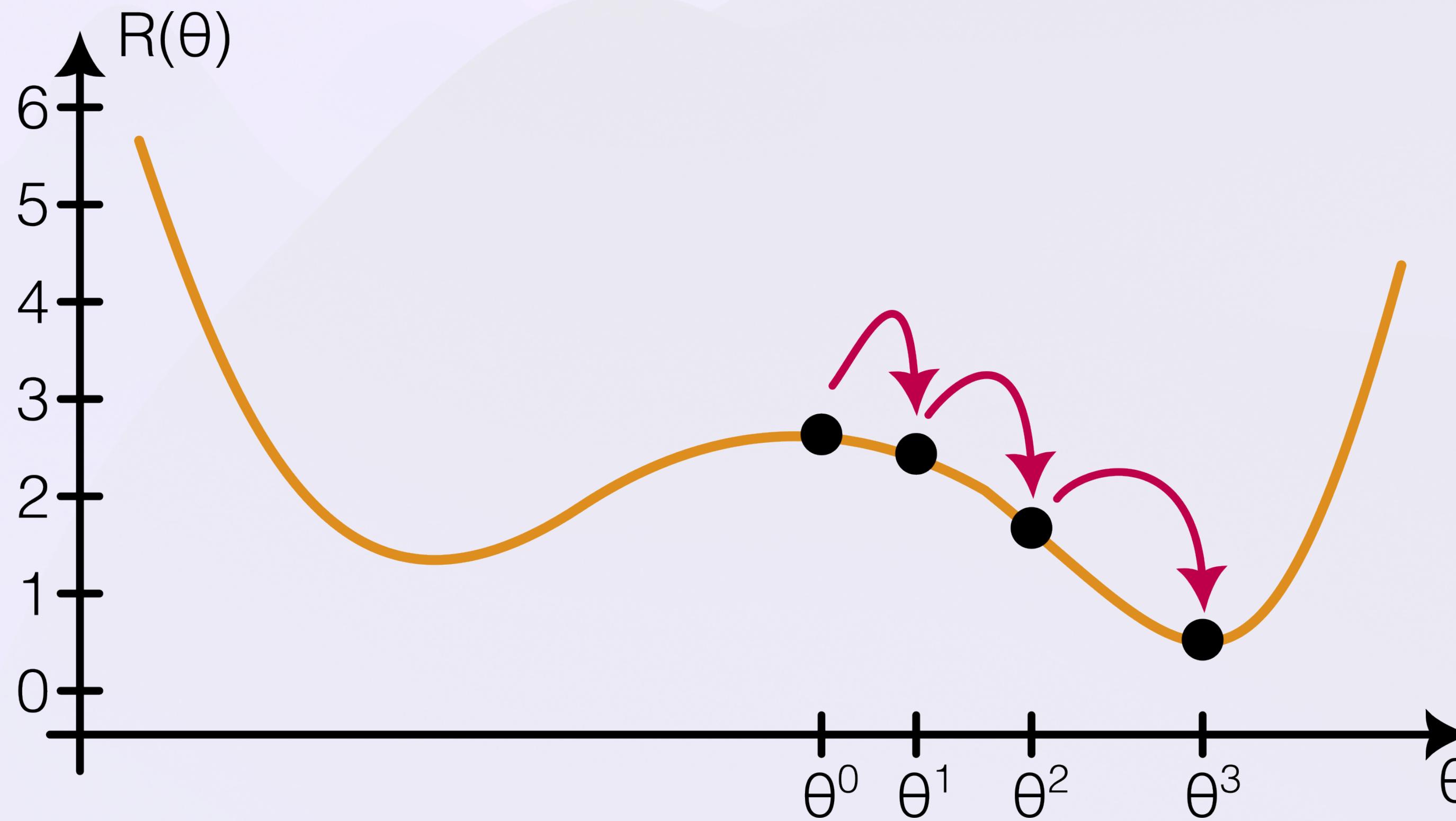
X <sub>1</sub>	X <sub>2</sub>	Y
0	0	0
0	1	0
1	0	0
1	1	1



# REDES FEED-FORWARD



# ENTRENAMIENTO



# PYTORCH

PyTorch es una librería de Deep Learning. Es mantenida por Meta.

Tiene un API para Python, como también para C++. PyTorch nos da toda una infraestructura de:

- Computación de tensores. Es similar a los array de Numpy pero con posibilidad de usarlos en GPU.
- Redes neuronales profundas.

# 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 obtenido, 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 obtenido, 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.*

The background features a series of overlapping, wavy shapes in shades of purple and dark blue, creating a sense of depth and motion.

# 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 mas famosos de los múltiples que existen:

- **K-Means**
- **Hierarchical clustering**



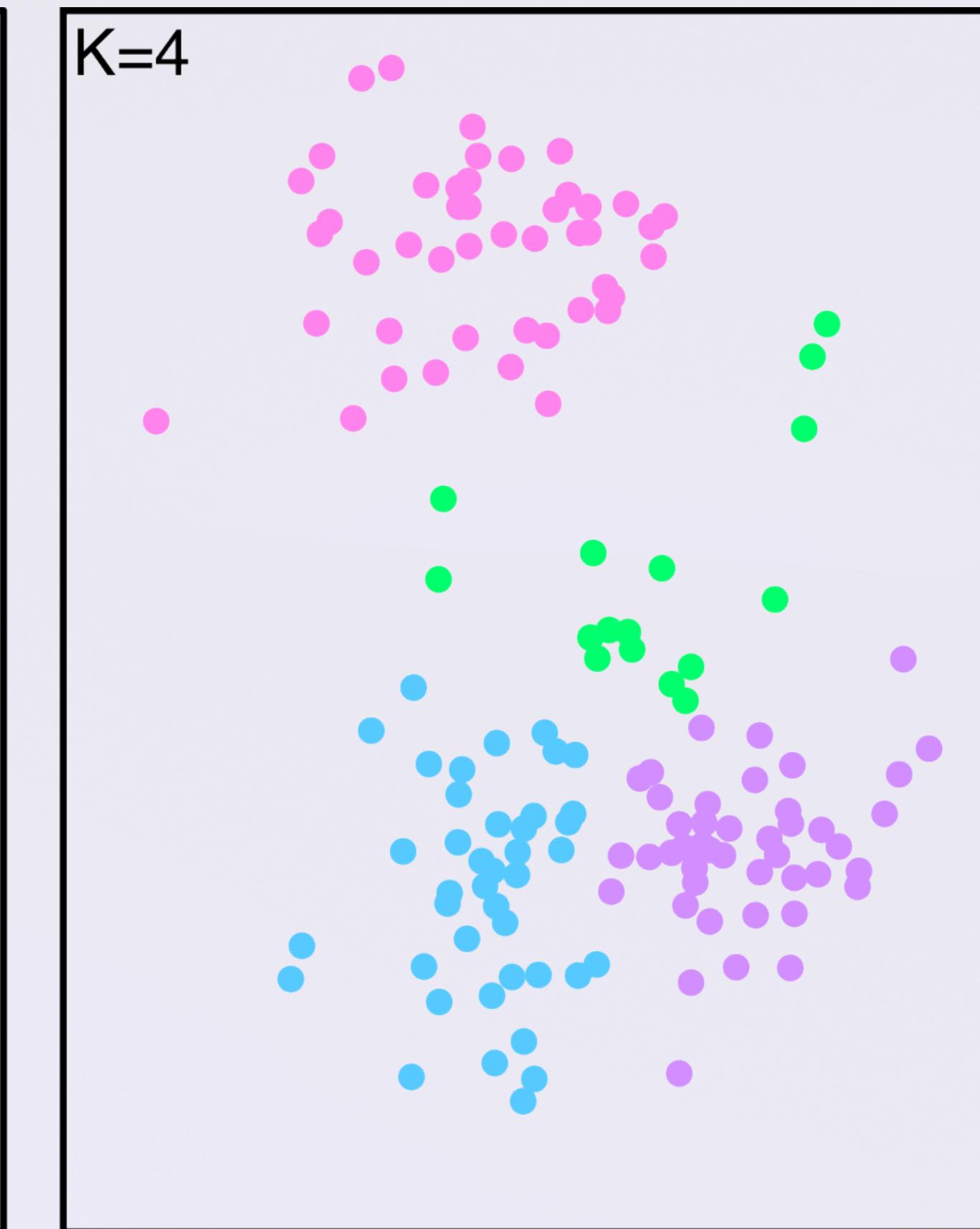
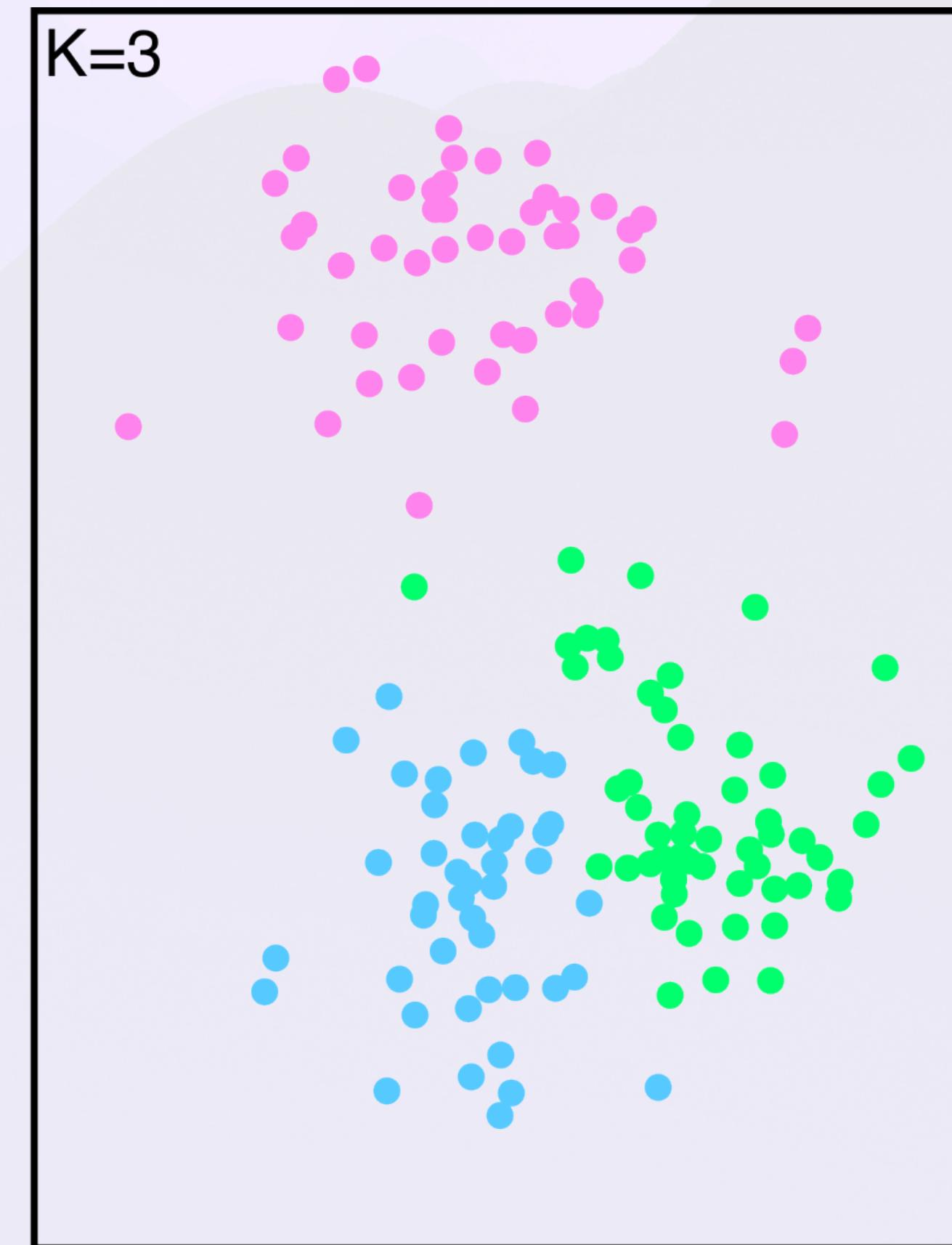
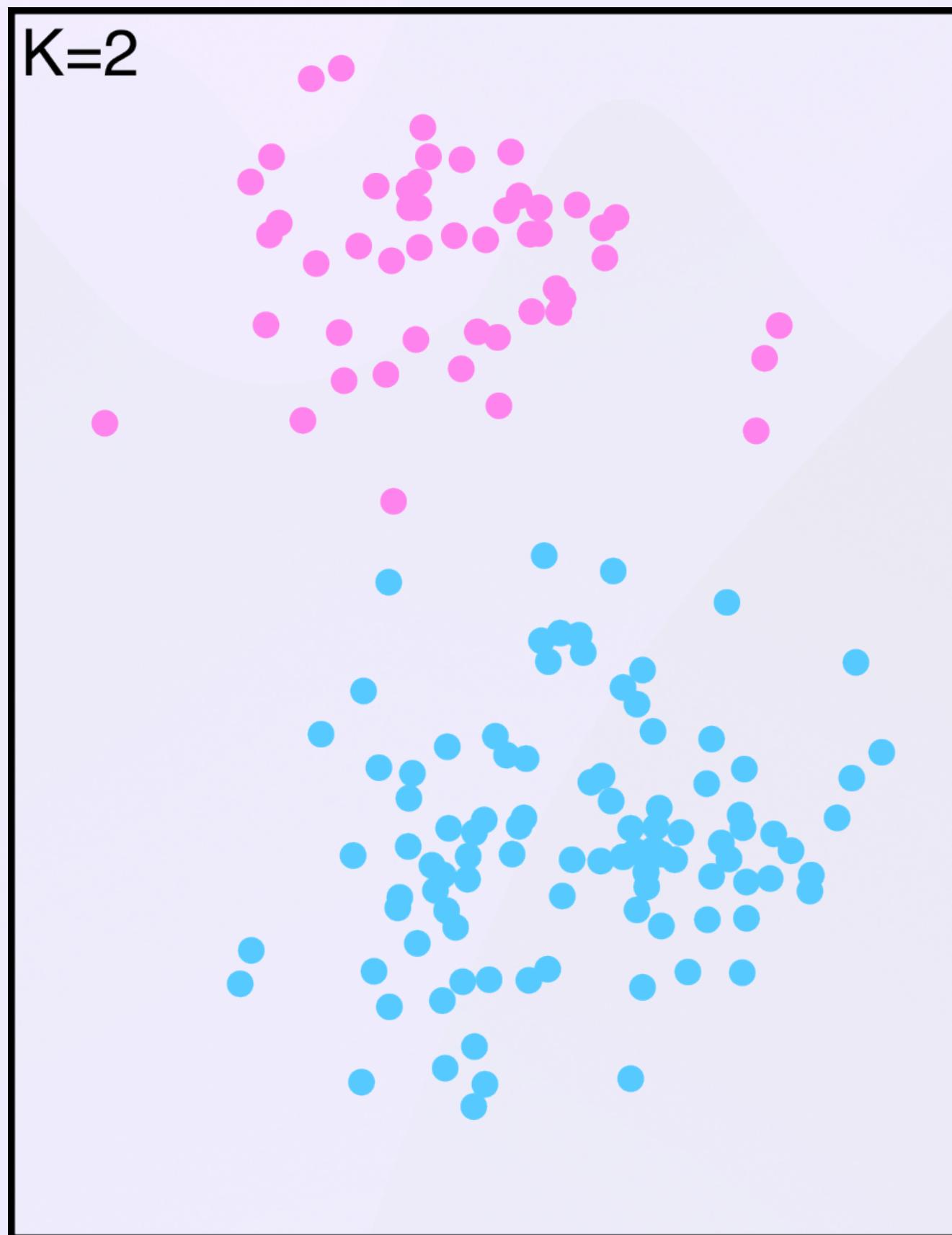
# K-MEANS CLUSTERING

# 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 indices 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

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

- 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 al que las observaciones dentro de un cluster difieren entre si. 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.

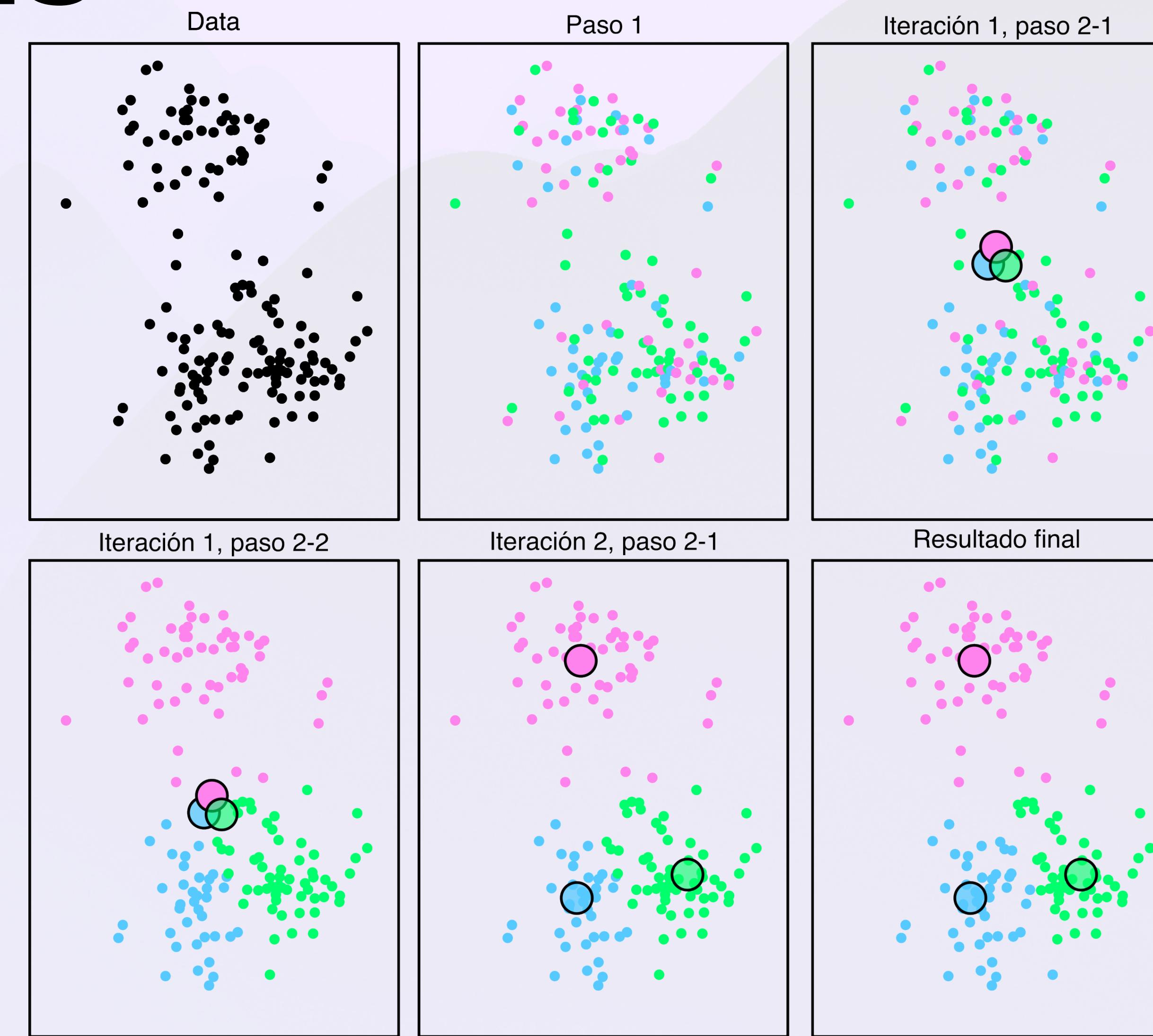
Ahora, resolver este minimización es un problema muy difícil de resolver con precisión, ya que hay casi  $K^n$  formas de dividir n observaciones en K grupos.

# K-MEANS

Una forma de resolver esto de una forma más eficiente es usando el centroide del cluster, con el siguiente algoritmo:

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. El centroide del cluster k es un vector de p elementos con las medias de cada atributo para las observaciones de ese cluster k.
  2. Se asignan a cada observación al cluster al que el centroide se encuentre más cercano (mediante la distancia Euclídea).

# K-MEANS

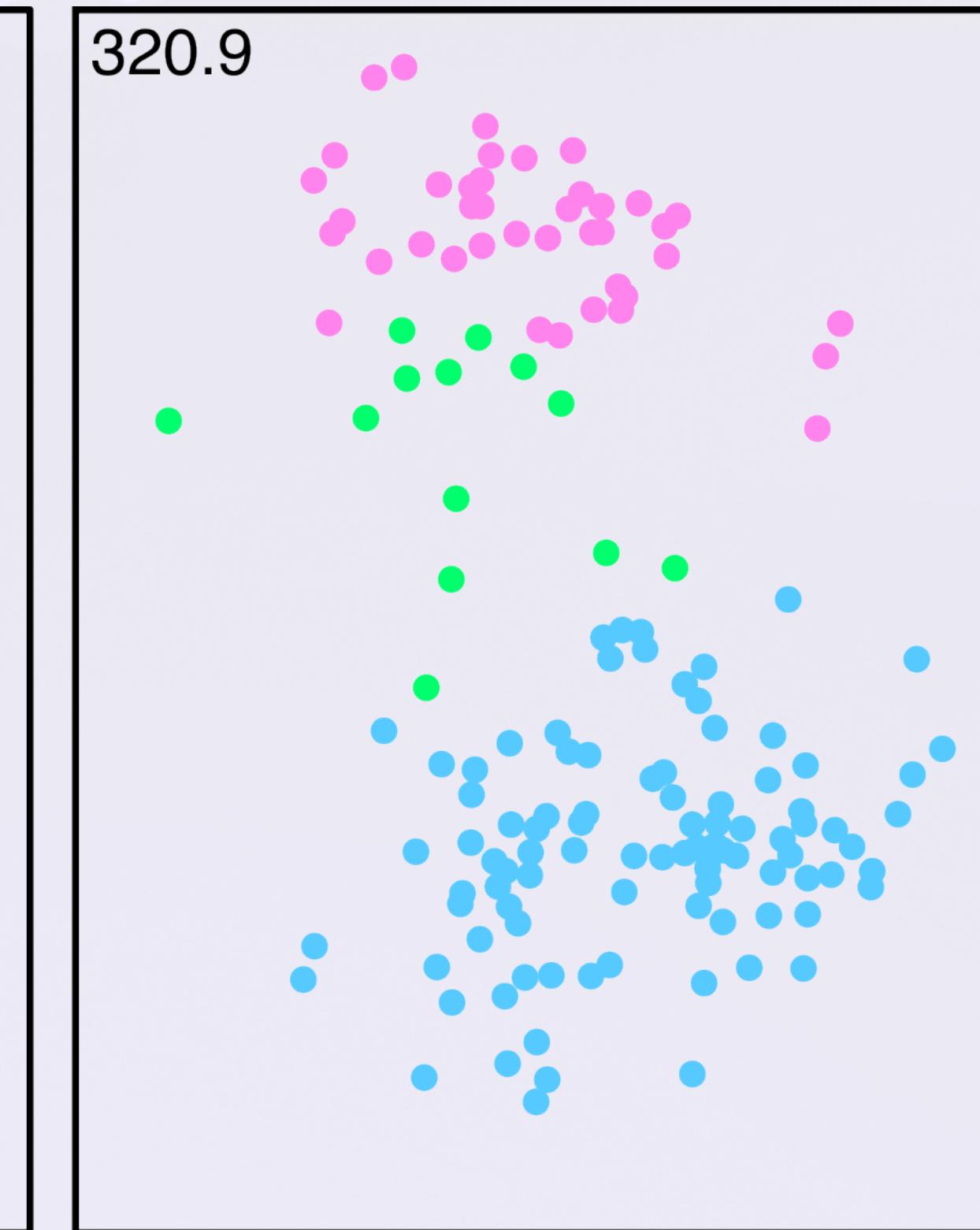
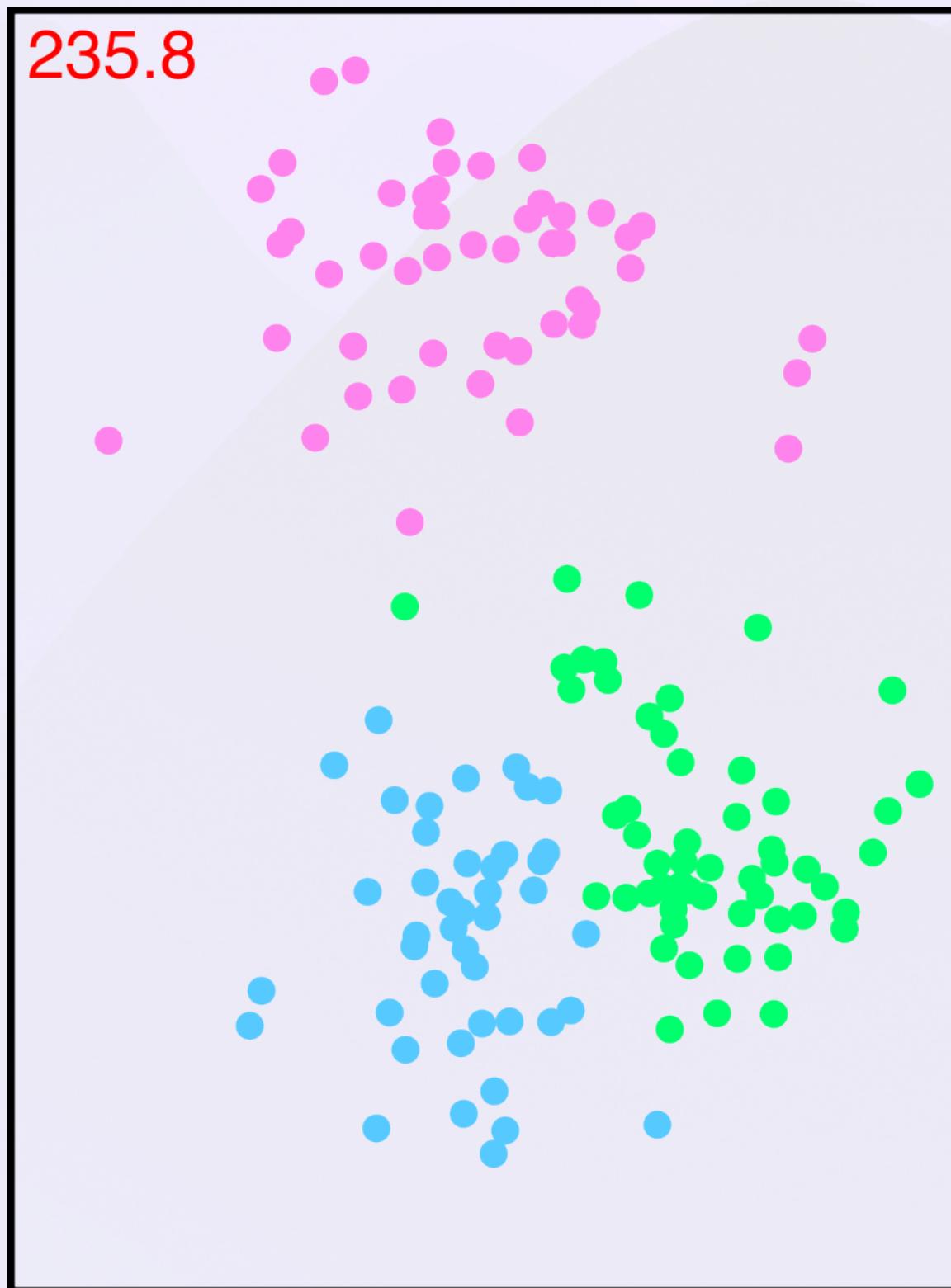


# K-MEANS

Este algoritmo garantiza que se va a lograr la minimización de  $W(C_K)$  en cada paso. Ademas, dado que cada paso de iteración siempre disminuye, el algoritmo carrera 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 múltiple veces y finalmente seleccionar la **mejor** solución (La que tenga menor  $\sum_{k=1}^K W(C_K)$  ).

# K-MEANS



The background features a minimalist design with abstract, wavy shapes in shades of purple and dark blue. These shapes are layered and overlap, creating a sense of depth and movement across the entire frame.

**VAMOS A PRÁCTICAR UN POCO...**

# HIERARCHICAL CLUSTERING

# HIERARCHICAL CLUSTERING

Una de las principales desventajas de K-means es que se requiere per-especificar el número de clusters.

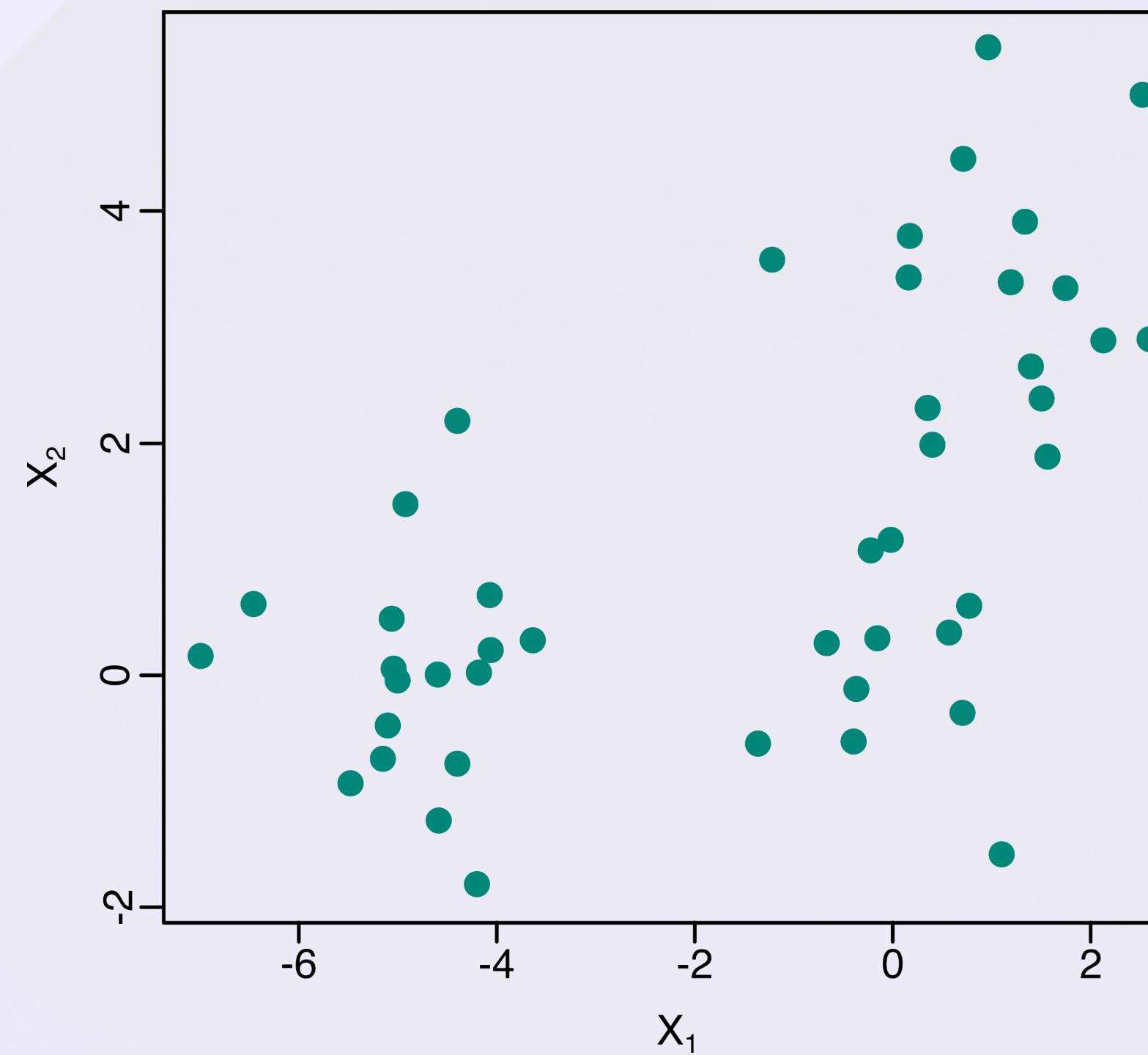
En cambio **hierarchical clustering** es una alternativa que no requiere especificar una cantidad específica de clusters.

Una ventaja adicional es que da como resultado una representación basada en árbol de las observaciones, llamada **dendrograma**.

La principal desventaja es que este algoritmo es  $O(n^3)$  salvo casos muy particulares

# DENDROGRAMA

Supongamos que tenemos el siguiente caso, donde los datos provienen de 3 grupos, pero supongamos que no tenemos la información de las clases.



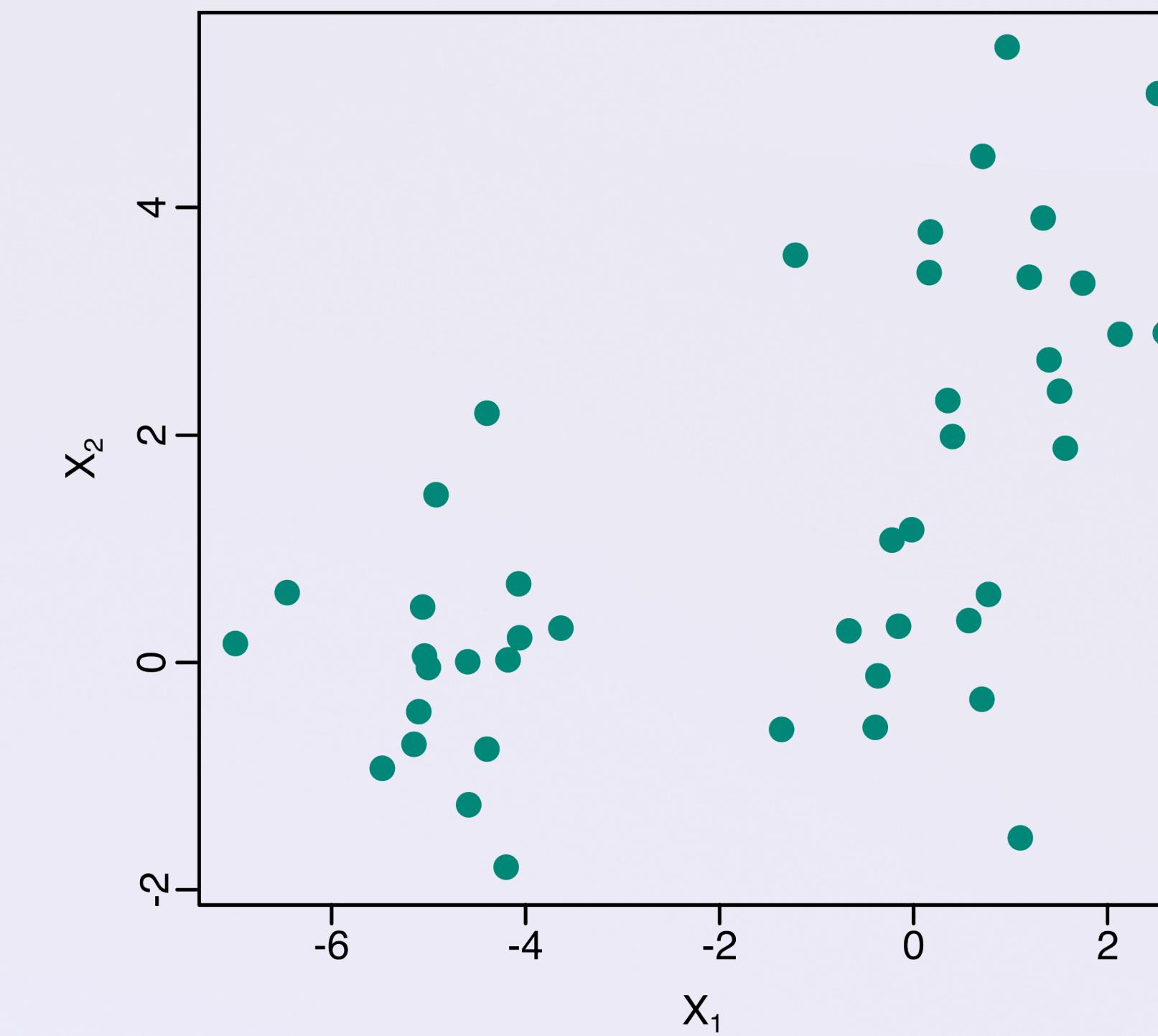
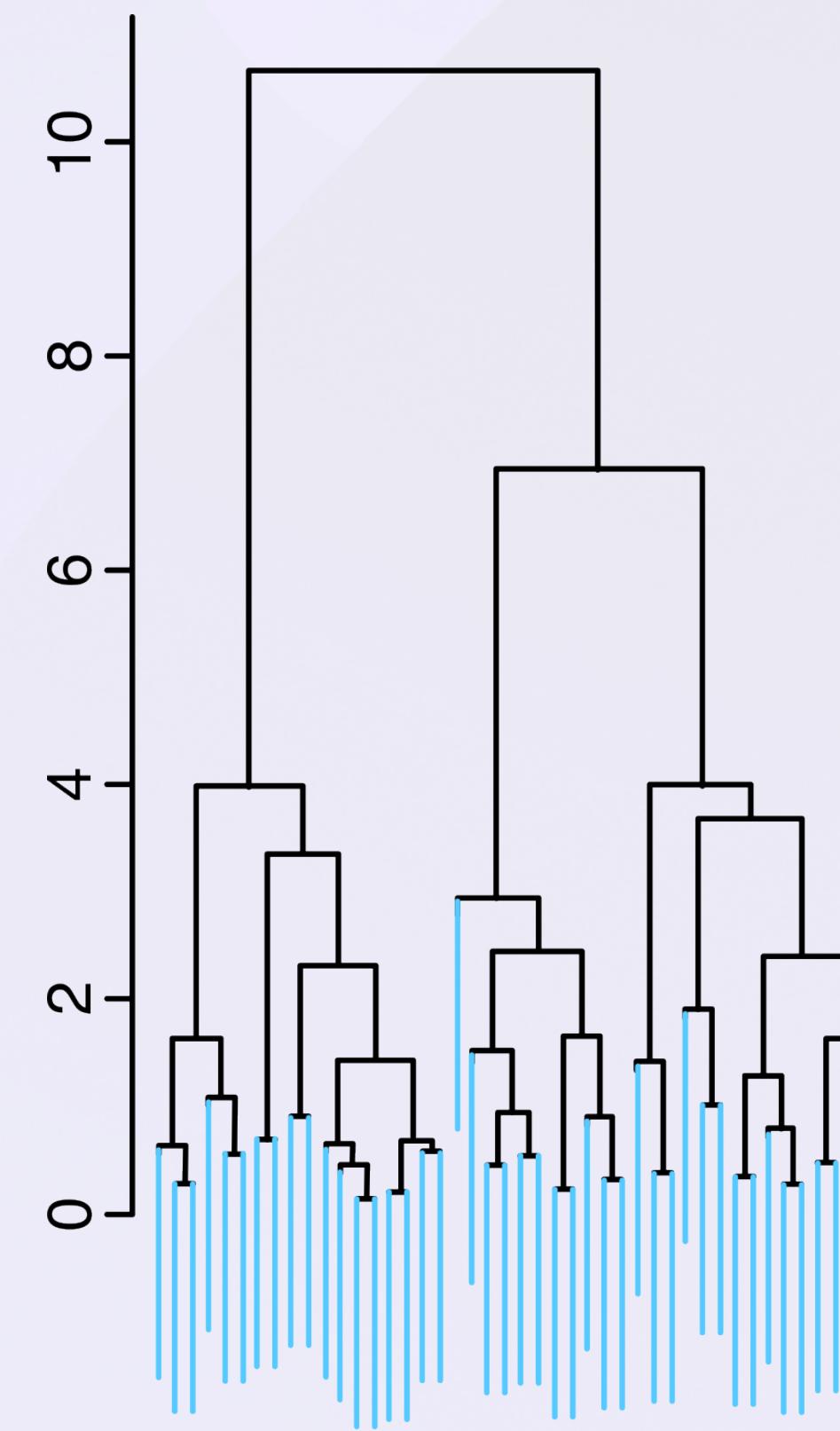
# HIERARCHICAL CLUSTERING

Una de las principales desventajas de K-means es que se requiere especificar el número de clusters previo al cálculo. En cambio hierarchical clustering es un alternativa que no requiere especificar **una cantidad específica de clusters**.

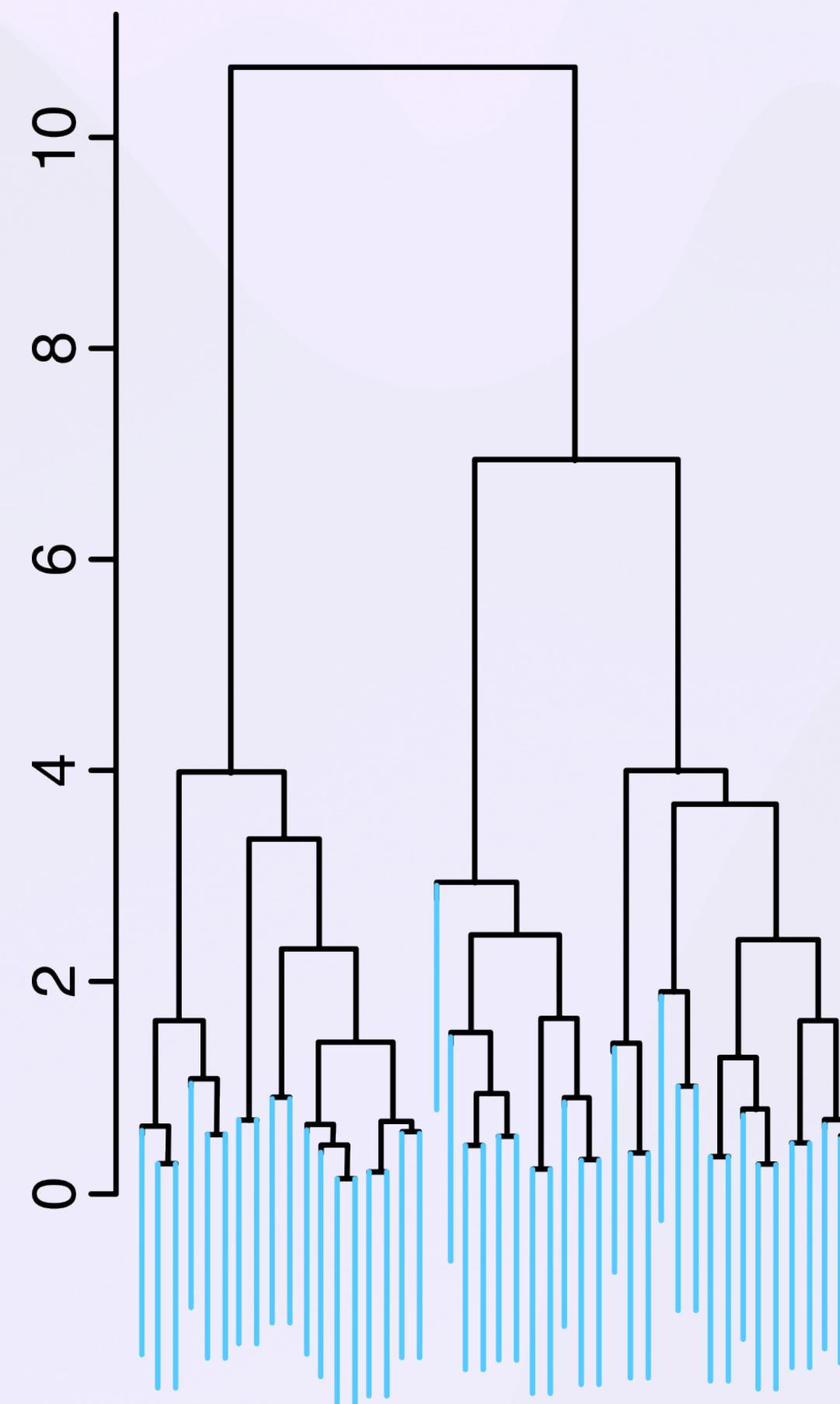
Una ventaja adicional es que da como resultado una representación basada en árbol de las observaciones, llamada **dendrograma**.

# HIERARCHICAL CLUSTERING

Un dendrograma es el siguiente, donde cada hoja es una de las observaciones del dataset original



# HIERARCHICAL CLUSTERING

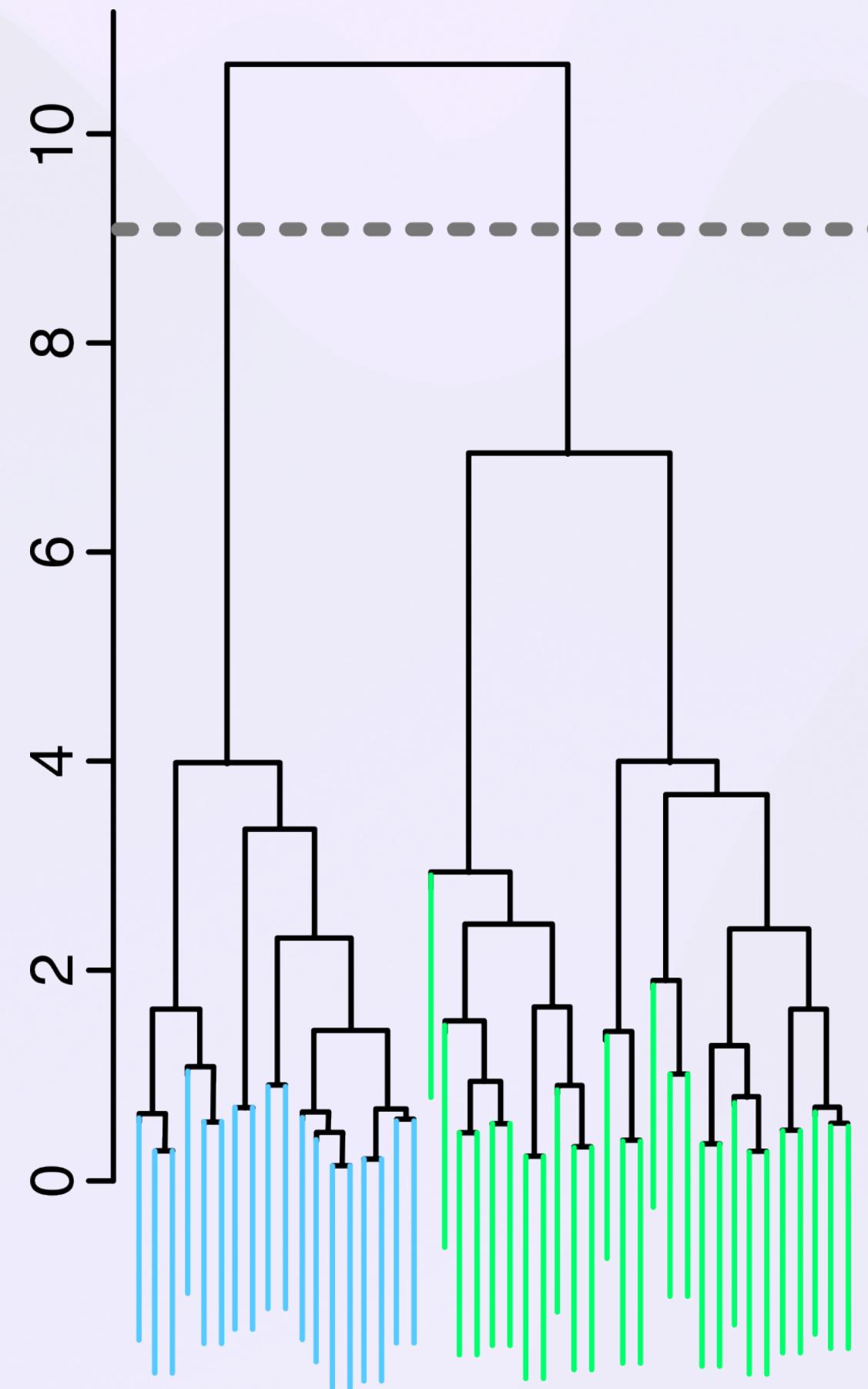


A medida que subimos el árbol, algunas hojas se van fusionando en ramas. Estas fusiones corresponden a observaciones que son similares entre si.

Seguimos subiendo, y las ramas se van fusionando con otras ramas u hojas.

Cuando mas bajo en el árbol ocurra una fusión, más similares son el grupo de observaciones afectados. En cambio, si es más arriba, la similitud es mucho menor.

# HIERARCHICAL CLUSTERING

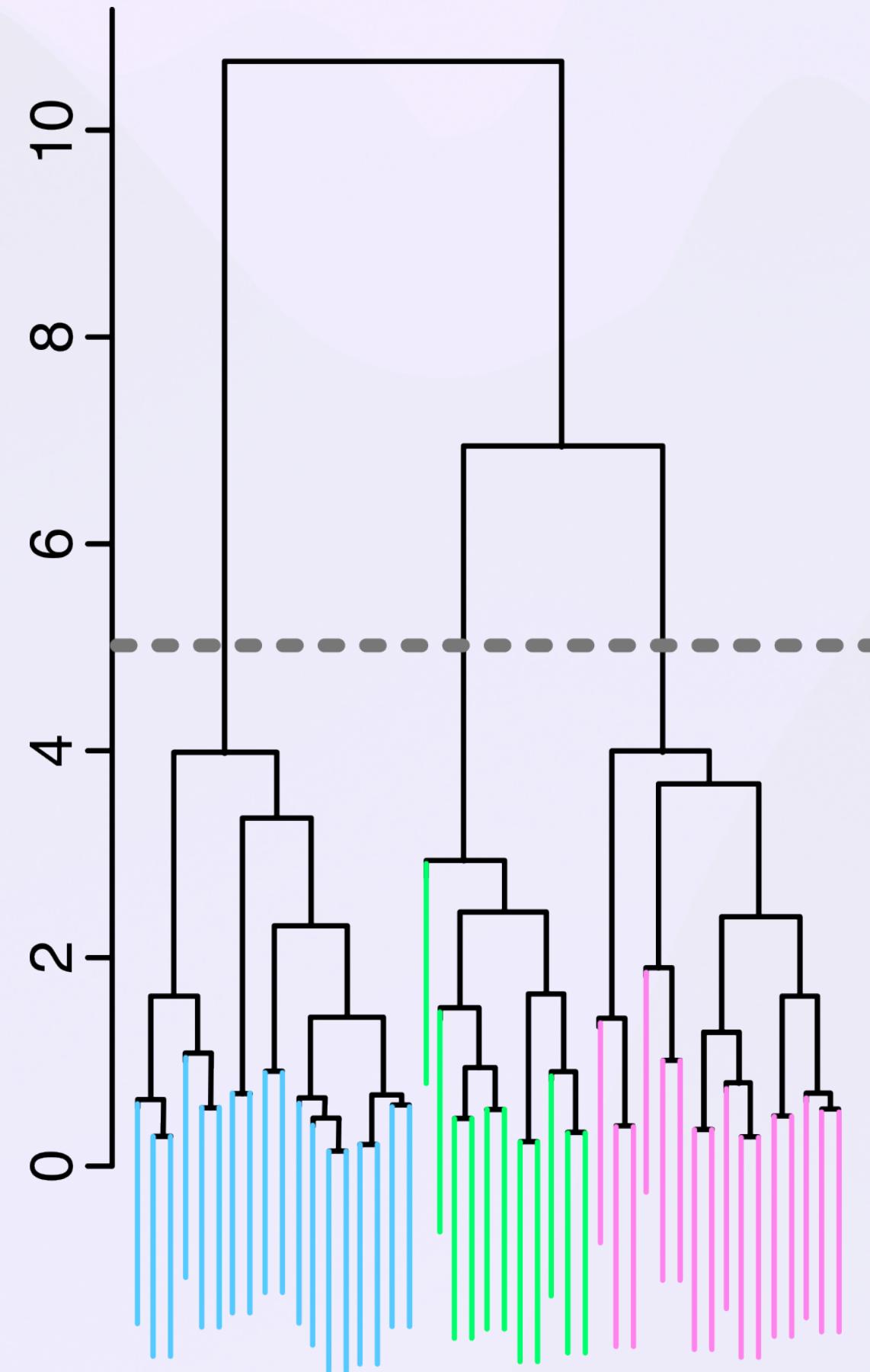


Entonces eligiendo la altura del árbol, se puede elegir la cantidad de clusters.

Podemos ir desde **1 a n** clusters, en donde **n** es el numero de observaciones.

Aunque este formato es atractivo, en dataset reales, muchas veces es difícil definir la altura de corte.

# HIERARCHICAL CLUSTERING



Entonces eligiendo la altura del árbol, se puede elegir la cantidad de clusters.

Podemos ir desde **1 a n** clusters, en donde **n** es el numero de observaciones.

Aunque este formato es atractivo, en dataset reales, muchas veces es difícil definir la altura de corte.

# HIERARCHICAL CLUSTERING

El término jerárquico (hierarchical) se refiere al hecho de que los grupos obtenidos al cortar el dendrograma a una altura determinada están necesariamente anidados dentro de los grupos obtenidos al cortar el dendrograma a una altura mayor.

Sin embargo, en un conjunto de datos arbitrario, esta suposición de estructura jerárquica podría resultar poco realista.

# HIERARCHICAL CLUSTERING

Para obtener el dendrograma se necesita una medida de disimilitud entre pares de observaciones. Similar a K-means, la distancia euclidiana es una medida común.

El algoritmo que construye este árbol usando un **método greedy**, arranca desde las hojas, cada una de las  $n$  observaciones se trata como un cluster propio.

Los dos clusters que son mas similares con cada uno se van fusionando, de tal forma que hay  $n-1$  clusters. Luego los dos clusters que son más similares, se fusionan. Y el algoritmo continúa así hasta que todas las observaciones pertenezcan a una sola clase.

# HIERARCHICAL CLUSTERING

Es decir, el algoritmo es:

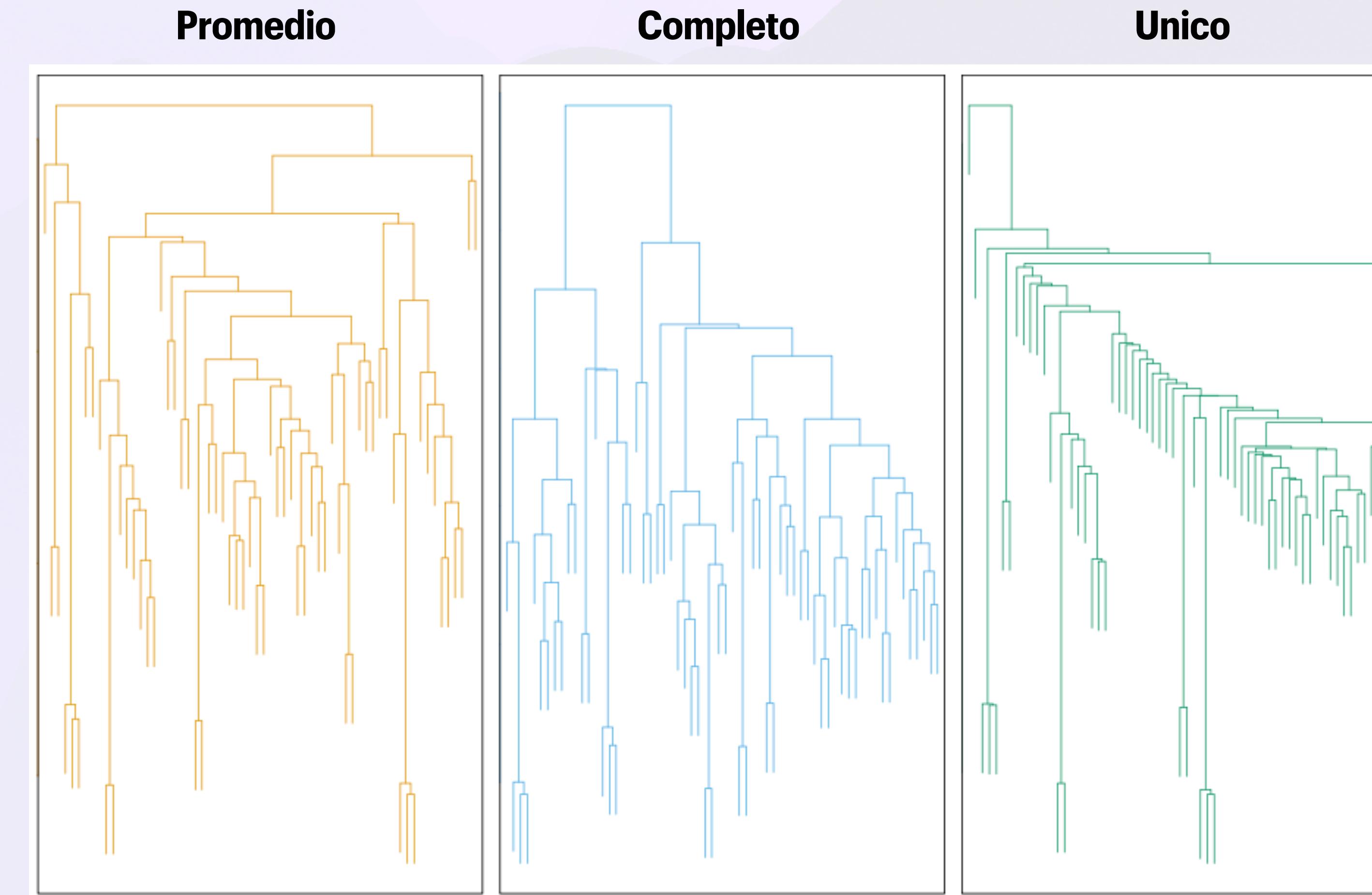
1. Arrancamos con  $n$  observaciones y una medida (tal como Euclídea) de todos los pares de similitudes. Se trata a cada observación como propia.
2. Para cada  $i = n, n - 1, \dots, 2$ 
  1. Examine todas las diferencias entre grupos por pares entre los  $i$  grupos e identifique el par de grupos que son menos diferentes. Fusiona estos dos grupos. La diferencia entre estos dos grupos indica la altura en el dendrograma a la que se debe colocar la fusión.
  2. Calcule las nuevas diferencias entre grupos por pares entre los  $i - 1$  grupos restantes.

# HIERARCHICAL CLUSTERING

Una cuestión que nos queda por ver es con quien se calcula la disimilitud entre dos clusters:

- **Completo**: Se computa todos los pares de disimilitudes entre las observaciones del cluster A y las observaciones del cluster B y se registra la **mayor disimilitud**.
- **Único**: Se computa todos los pares de disimilitudes entre las observaciones del cluster A y las observaciones del cluster B y se registra la **menor disimilitud**.
- **Promedio**: Se computa todos los pares de disimilitudes entre las observaciones del cluster A y las observaciones del cluster B y se registra la **disimilitud promedio**.
- **Centroide**: Se calculan los centroides de los clusters A y B, y luego se mide la disimilitud de los centroides.

# HIERARCHICAL CLUSTERING



The background features a minimalist design with abstract, wavy shapes in shades of purple and dark blue. These shapes are layered and overlap, creating a sense of depth and movement across the entire frame.

**VAMOS A PRÁCTICAR UN POCO...**