

# Resumen Reconocimiento de Patrones

## INDICE

Características geométricas.....	2
<i>Hu, Fourier, LBP, Gabor, Haralick, HoG, SIFT.</i>	
Características de intensidad.....	5
Selección de características.....	5
Búsqueda exhaustiva, <i>Clean, SFS, Branch &amp; Bound, Fisher.</i>	
Reductores de dimensionalidad.....	8
PCA, ICA, PLSR	
Clasificadores.....	9
Bayes, DMIN, LDA, QDA, <i>Mahalanobis</i> , Árboles de decisión, <i>Random Forest</i> , Redes Neuronales, <i>Deep Learning</i> , SVM.	
Evaluaciones de desempeño de clasificación.....	16
<i>Hold Out, Cross Validation, Leave one out, Matriz de confusión.</i>	
Clustering.....	16
<i>Mixture if Gaussians, Hierarchical, K-Means, Mean Shift</i>	

## Normalización

Digamos que tenemos una imagen  $x$  y queremos extraer las características  $x_1$  y  $x_2$  de la imagen.

- El rango de posibles valores de  $x_1$  va de 0 hasta 100.
- El rango de posibles valores de  $x_2$  va de 0 hasta 2.

Problema: Si queremos construir una función para clasificar la imagen según las características de  $x_1$  y  $x_2$  entonces tendríamos un problema, pues  $x_1$  tiende a tomar valores más altos que  $x_2$ , y por ende tendrá más peso a la hora de clasificar la imagen, cuando no necesariamente tiene que ser así.

Solución: Ajustamos las escalas de  $x_1$  y  $x_2$  de manera que el rango de posibles valores de vaya desde 0 hasta 1 (por ejemplo). De esta manera, ambas características tendrán el mismo peso a la hora de clasificar la imagen. Esto se conoce como **normalización** de características.

## Características Geométricas

- **Hu – Moments**: Se refiere a las 7 invariantes de momentos Hu, que son un conjunto de valores numéricos utilizados para describir la forma o la geometría de un objeto en una imagen **binaria**.
- **Descriptores de Fourier**: Conjunto de características para representar la geometría y las propiedades de un objeto en el dominio de la frecuencia. Estos descriptores se basan en la Transformada de Fourier, que es una técnica matemática que descompone una señal o una función en sus componentes de frecuencia.
- **Local Binary Patterns (LBP)**: La idea detrás de LBP es capturar las propiedades locales de la textura en una imagen al **comparar** los valores de intensidad de los píxeles en una **vecindad alrededor de cada píxel central**.



4	6	9
9	6	4
9	6	2

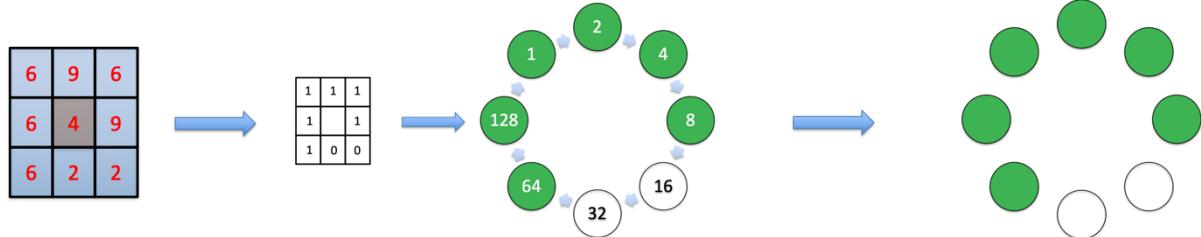
0: <  
1: ≥

0	1	1
1		0
1	1	0

1	2	4
128	+ (circled)	8
64	32	16

= 2+4+32+64+128 = 230

Pesos arbitrarios

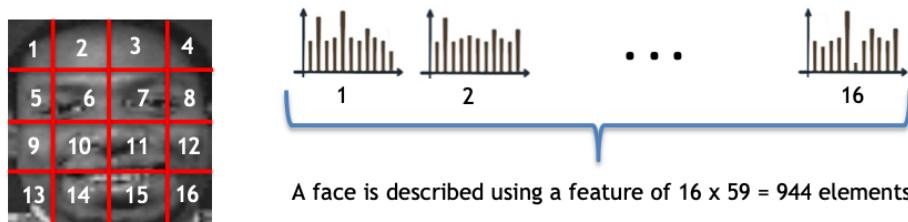
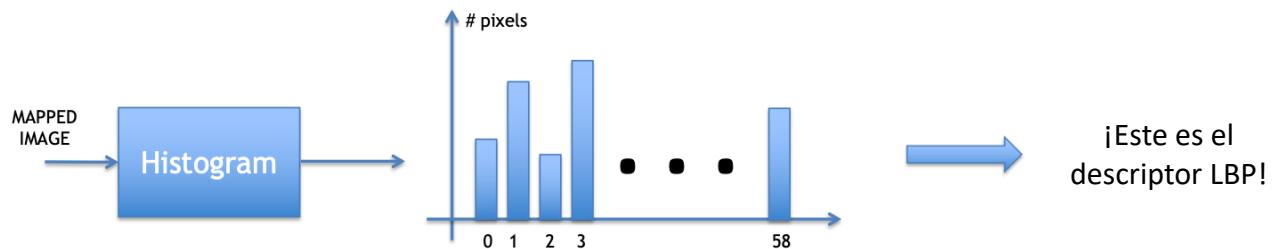
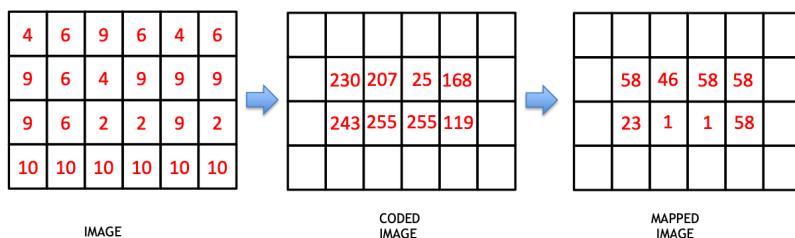


$U = 0$  Hay 0 cambios de color

$U = 2$  ... Hay 2 cambios de color

•  
•  
•

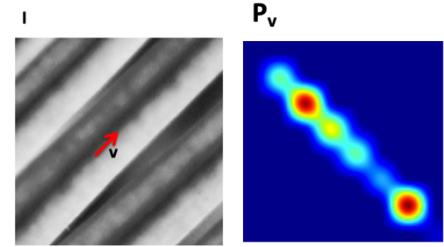
Para cada uno de estos círculos, se les asigna un número del 1 al 58



- **Gabor:** Los filtros de Gabor son el resultado de combinar una función sinusoidal (ondulatoria) con una función gaussiana (gaussiana). La función sinusoidal captura las variaciones de frecuencia en la imagen, mientras que la función gaussiana atenúa las contribuciones en las regiones lejanas del filtro. Esto permite a los filtros de Gabor ser sensibles a diferentes orientaciones y escalas en una imagen, lo que los hace adecuados para detectar características complejas, como **texturas**.

- **Haralick:** La matriz de co-ocurrencia de niveles de gris de Haralick es una técnica para cuantificar las relaciones espaciales de los niveles de gris en una imagen.

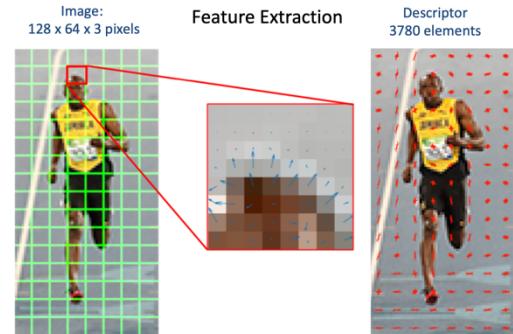
¿Qué pasa si la diagonal de la matriz de co-ocurrencia es baja? Significa que, en la dirección seleccionada los valores de los pixeles cambian significativamente.



Esta información puede ser útil para capturar características **texturales** y patrones en una imagen que no se pueden detectar fácilmente mediante otros métodos.

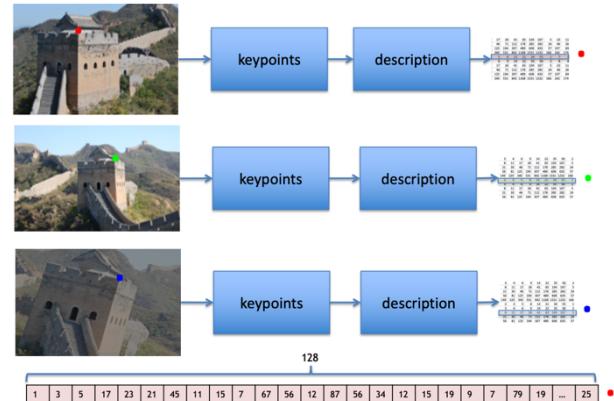
- **HoG (Histogram of Oriented Gradients):** Se utilizan principalmente para describir la forma y la apariencia de objetos en una imagen mediante el análisis de los gradientes locales de intensidad.

Son especialmente útiles en la **detección de objetos en imágenes, como peatones** en imágenes de vigilancia y vehículos en imágenes de tráfico.



- **SIFT (Scale Invariant Feature Transform):** Son utilizadas para detectar y describir características distintivas en una imagen de manera robusta **ante cambios en la escala, la rotación, la iluminación** y otros factores. Sus pasos son:

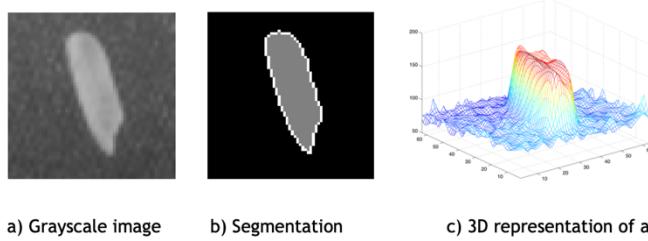
1. Detección de **keypoints**.
2. Asignación de orientaciones: Se asigna una orientación a cada punto clave en función de los gradientes locales de la imagen. Esto hace que las características SIFT sean invariantes a la rotación.
3. Creación de descriptores: Para cada punto clave, se calcula un descriptor que captura información sobre la apariencia y la textura en la vecindad del punto. Estos descriptores son robustos ante cambios en la escala y la iluminación.
4. Coincidencia de puntos clave: Los descriptores de puntos clave en diferentes imágenes se comparan para encontrar coincidencias y establecer correspondencias entre las imágenes.



## Características de Intensidad

Existen 2 categorías de características: **Geométricas** e **Intensidad**.

Las Características **Geométricas** proporcionan información sobre la **ubicación, orientación, forma** y **tamaño**. Las Características de **Intensidad** proporcionan información sobre cómo son los **valores de gris**.



## Selección de características

Razones por las que se debe seleccionar características:

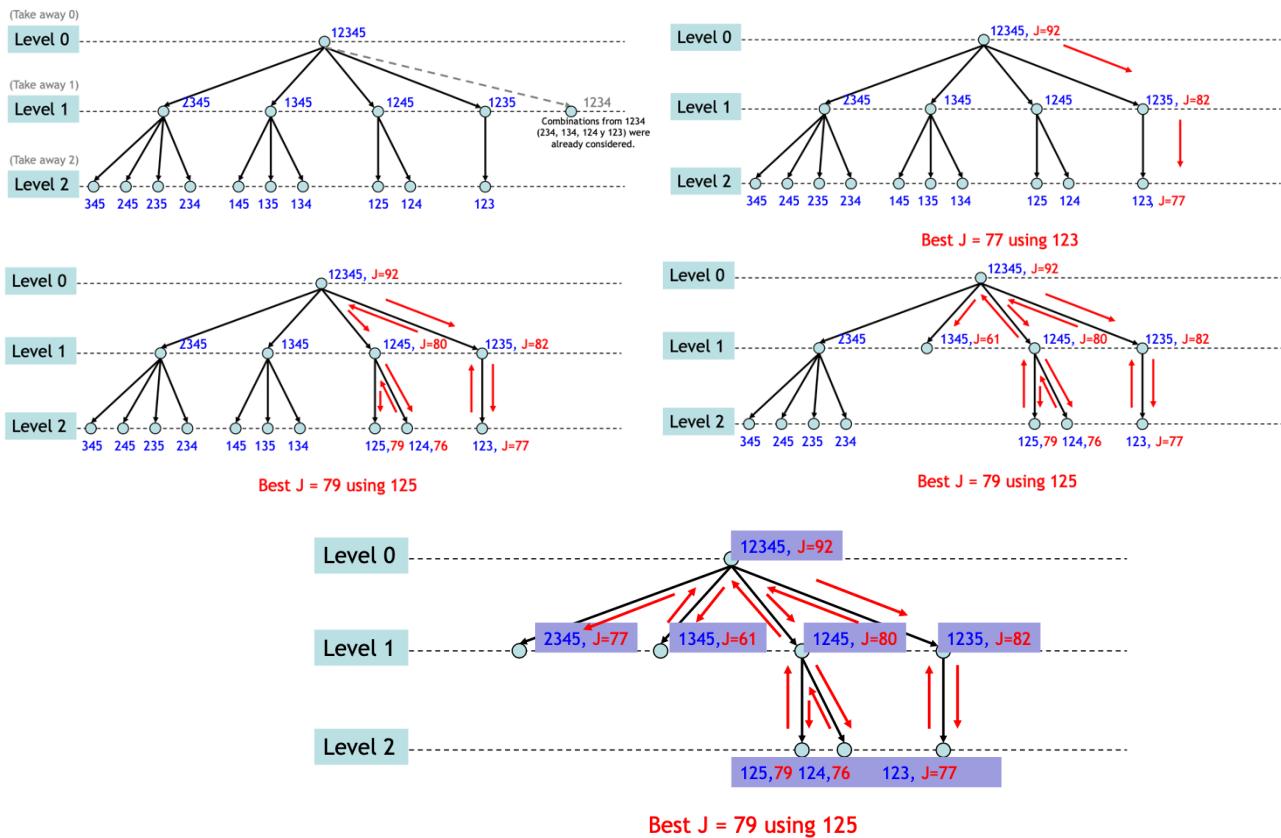
1. Para evitar características no discriminativas (que no sirven).
  2. Para evitar características correlacionadas.
  3. Para simplificar la etapa de prueba.
  4. Para evitar correlaciones falsas.
  5. Para evitar la maldición de la dimensionalidad.
- **Búsqueda exhaustiva:** Dadas  $m$  características extraídas de las imágenes, se extraen bajo búsqueda exhaustiva (probando todas las combinaciones) las mejores  $n < m$  características. **Garantiza la selección de las mejores  $n$  características.**
  - **Clean:** Este paso debe realizarse antes de cualquier algoritmo de selección de características.
    1. Calcular la correlación entre las columnas de  $X$ .
    2. Si existen columnas altamente correlacionadas, seleccionar una al azar y eliminar las demás.
    3. Eliminar características constantes (desviación estándar  $< 10^{-8}$ ).

- **SFS (Sequential Forward Selection)**: Los pasos son los siguientes:

1. Inicialización: Comienza con un conjunto vacío de características seleccionadas.
2. Búsqueda secuencial hacia adelante: En cada iteración, se evalúan todas las características no seleccionadas, y se **elige la mejor** (que mejora más el rendimiento del modelo).
3. Criterio de parada: El proceso de selección continúa hasta que se cumple un criterio de parada predefinido. Puede ser un número fijo de características seleccionadas, una mejora de rendimiento mínima alcanzada, o cualquier otro criterio establecido por el usuario.

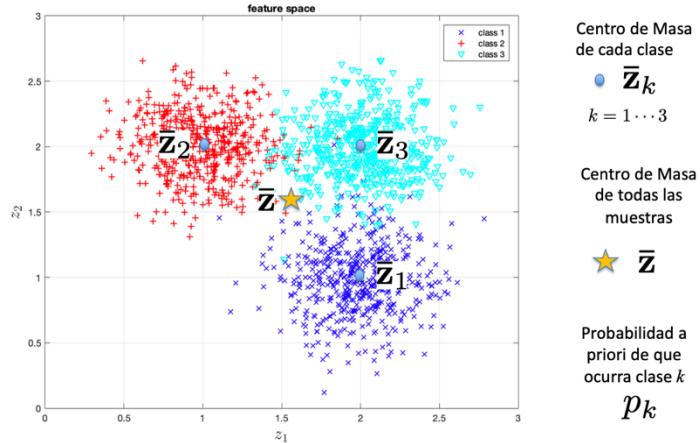
NO garantiza una selección óptima de características, pero es mucho más rápido que la búsqueda exhaustiva.

- **Branch & Bound**: La idea básica detrás del algoritmo Branch & Bound es explorar sistemáticamente el espacio de todas las posibles combinaciones de características, pero de manera eficiente para evitar evaluar todas las combinaciones posibles. El algoritmo se basa en dividir el problema en subproblemas más pequeños y luego "podar" o eliminar ramas de exploración que no pueden llevar a una solución óptima.



- **Fisher:** También conocido como análisis discriminante de Fisher (FDA). El objetivo principal del algoritmo es maximizar la separación entre las clases en los datos mediante la selección de un subconjunto de características que sean más informativas para la tarea de clasificación.

Ejemplo en donde hay 3 clases y se obtienen 2 características  $z_1$  y  $z_2$ .



Se calcula la covarianza **inter-clase**:

$$\mathbf{C}_b = \sum p_k (\bar{Z}_k - \bar{Z})(\bar{Z}_k - \bar{Z})^T$$

Se calcula la covarianza **intra-clase**:

Covarianza de la clase  $k$ :

$$C_k = \frac{1}{N_k - 1} \sum_{j=1}^{N_k} (\bar{Z}_{kj} - \bar{Z}_k)(\bar{Z}_{kj} - \bar{Z}_k)^T$$

$N_k$  = Muestras de la clase  $k$

Se calcula el promedio ponderado de las Covarianzas de cada clase:

$$\mathbf{C}_w = \sum_{k=1}^K p_k C_k$$

Strategy: **Inter-Class HIGH + Intra-Class LOW**

**Criterio de Fisher:**

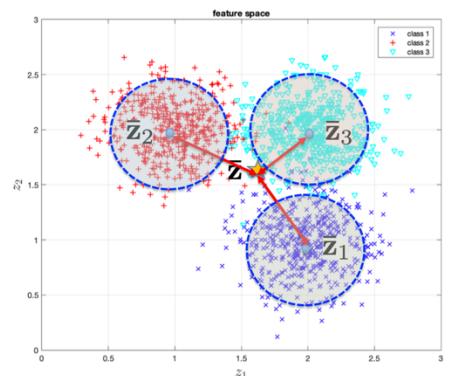
Dice que:

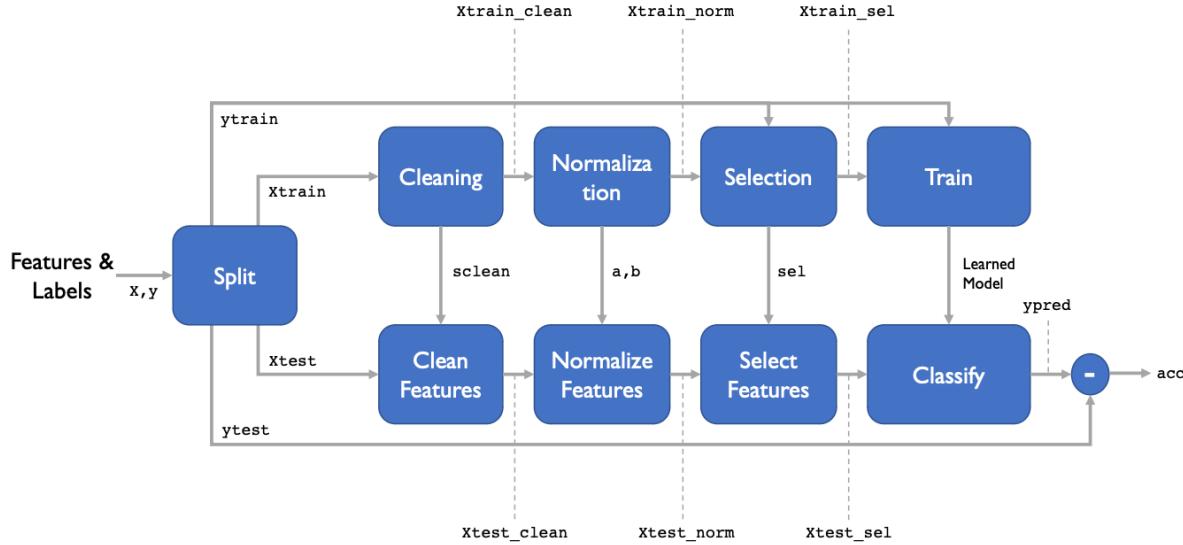
- $\mathbf{C}_b$  debe ser **BAJA**.
- $\mathbf{C}_w$  debe ser **ALTA**.

Por lo tanto, se debe maximizar:

$spur =$  Suma de la diagonal

$$J = spur(\mathbf{C}_w^{-1} \mathbf{C}_b)$$





- **PCA (Principal Component Analysis):** Se utiliza para **reducir la dimensionalidad** de datos, lo que significa que ayuda a simplificar conjuntos de datos complejos al mismo tiempo que conserva la mayor cantidad posible de su variabilidad o información relevante.

El objetivo principal de PCA es transformar un conjunto de variables originales (características) en un nuevo conjunto de variables, llamadas componentes principales, que son **combinaciones lineales de las características originales**. Estas componentes principales están ordenadas de tal manera que la primera componente principal captura la mayor variabilidad de los datos, la segunda captura la segunda mayor variabilidad, y así sucesivamente.

- **ICA (Independent Component Analysis):** A diferencia de otras técnicas como PCA, que buscan componentes que expliquen la mayor variabilidad de los datos, ICA busca componentes que sean **estadísticamente independientes entre sí**.

Es particularmente útil en situaciones en las que se asume que las señales observadas son una mezcla de señales fuente originales y las mezclas se han combinado de una manera lineal pero desconocida. Por ejemplo, en el campo de la neurociencia, ICA se utiliza para separar las señales de actividad cerebral registradas en un conjunto de electrodos, lo que permite identificar patrones de actividad cerebral independientes.

- **PLSR (Partial Least Squares Regression):** En lugar de tratar de encontrar una única combinación lineal de variables independientes que explique la variabilidad en la variable dependiente, como lo hace la regresión lineal ordinaria, PLSR busca encontrar componentes ortogonales que expliquen conjuntamente la variabilidad tanto en las variables independientes como en la variable dependiente.

## Clasificadores

- **Clasificador de Bayes:**

$\omega_1$  = Clase Mandarina.

$\omega_2$  = Clase Naranja.

$x$  = Característica (Área).

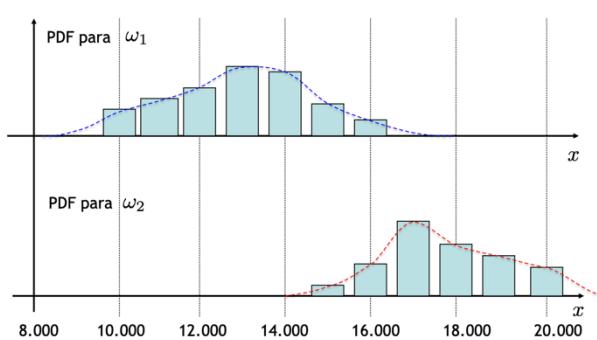
$p(\omega_1)$  = Probabilidad a priori de clase  $\omega_1$

$p(\omega_2)$  = Probabilidad a priori de clase  $\omega_2$

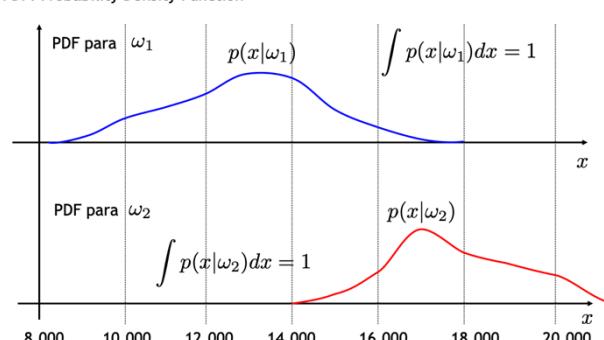
$p(x | \omega_1)$  = Distribución de  $x$  para Mandarinas.

$p(x | \omega_2)$  = Distribución de  $x$  para Naranjas.

PDF: Probability Density Function



PDF: Probability Density Function



Dado el valor  $x$  (área), calculamos:

- $p_1$ , la probabilidad de que sea  $\omega_1$  (mandarina).
- $p_2$ , la probabilidad de que sea  $\omega_2$  (naranja).

Clasificamos  $x$  como  $\omega_1$  si  $p_1 > p_2$ , en caso contrario clasificamos  $x$  como  $\omega_2$ .

Dado que sabemos el valor de  $x$

ALGORITMO:

$$p(\omega_1|x) > p(\omega_2|x) \rightarrow \omega_1 \text{ (mandarina)}$$

else  $\rightarrow \omega_2 \text{ (naranja)}$

$$p_1 = p(\omega_1|x)$$

$$p_2 = p(\omega_2|x)$$

Usando el teorema de Bayes:

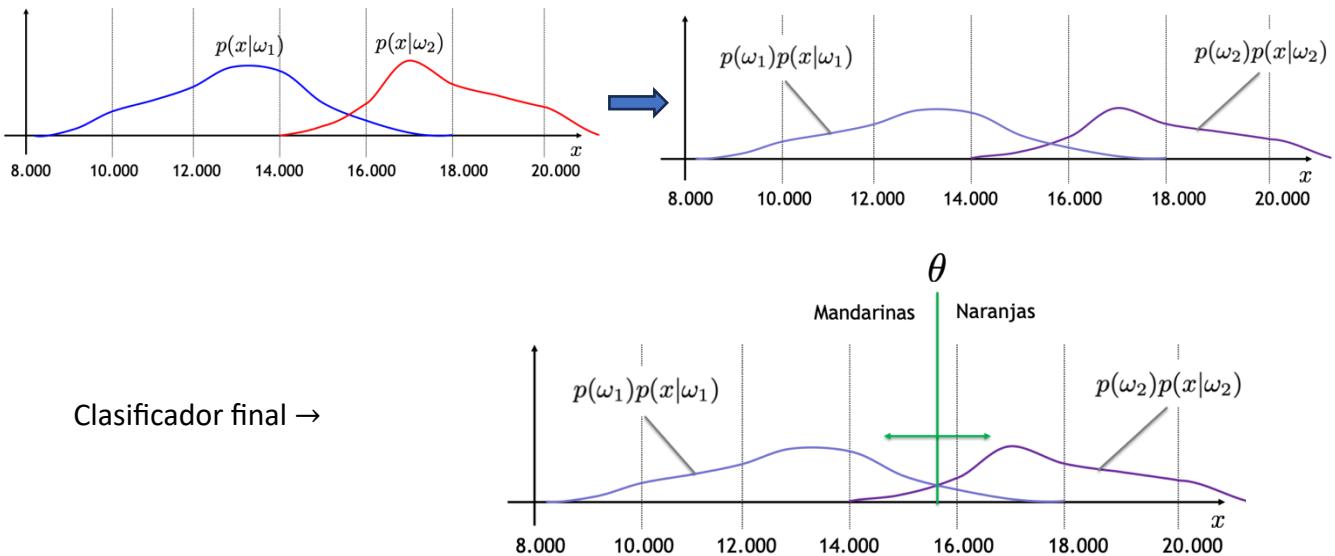
$$p(\omega_1 | x) = \frac{p(\omega_1)p(x | \omega_1)}{p(x)} \quad p(\omega_2 | x) = \frac{p(\omega_2)p(x | \omega_2)}{p(x)}$$

Por lo tanto, tenemos que:

ALGORITMO:

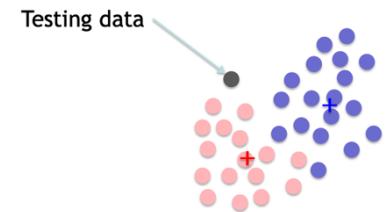
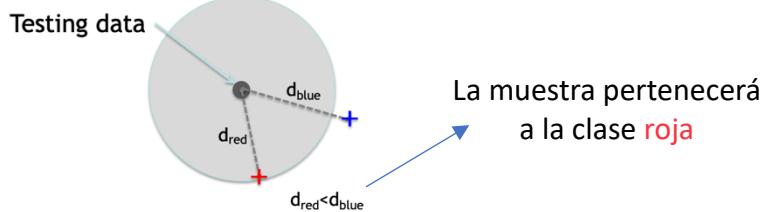
$$p(\omega_1)p(x|\omega_1) > p(\omega_2)p(x|\omega_2) \rightarrow \omega_1 \text{ (mandarina)}$$

else  $\rightarrow \omega_2 \text{ (naranja)}$



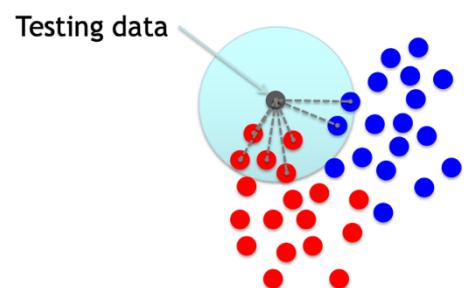
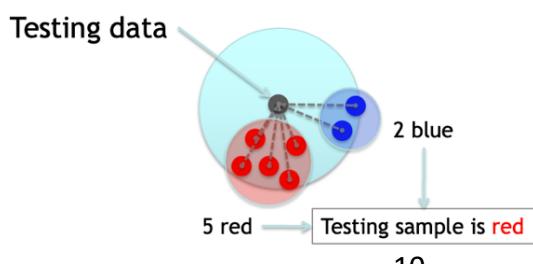
- **Distancia Mínima (DMIN):**

1. Calcular los centros de masa de cada clase.
2. Calcular la distancia de la muestra de prueba a cada centro de masa.
3. La muestra corresponderá a la clase que tenga el centro de masa más cercano

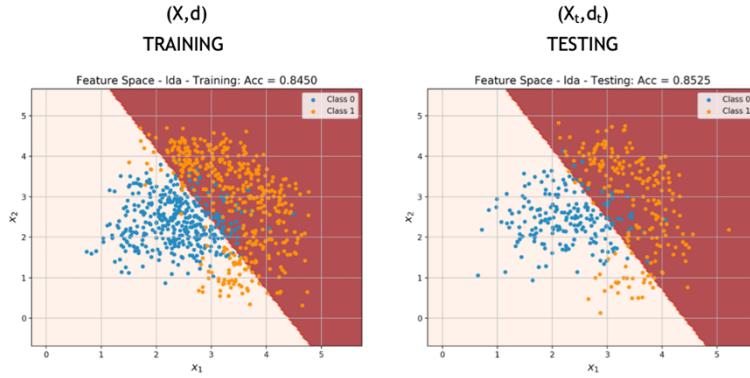


- **KNN ( $k$  nearest neighbors):**

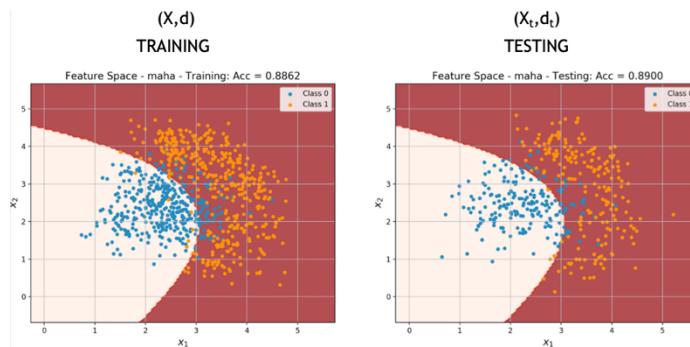
1. Se calcula la distancia a todas las muestras.
2. Se ordenan de menor a mayor.
3. Se toman las  $k$  distancias menores (más cercanas).
4. Se toma la clase que tenga la mayoría.
5. Ideal que  $k$  sea impar para desempates.



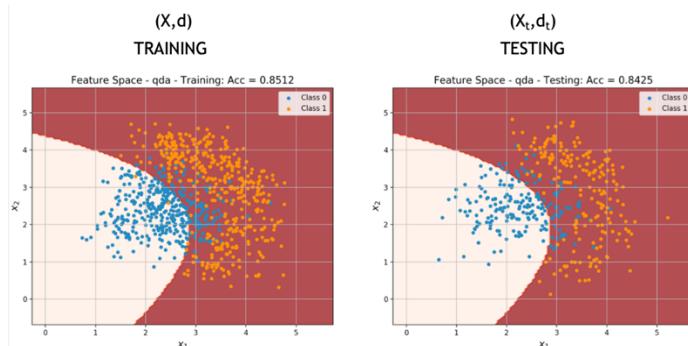
- **LDA (Análisis de Discriminante Lineal):** Similar al clasificador de Bayes, solo que se asume que las matrices de covarianzas son todas iguales. A diferencia del clasificador de Bayes, que se basa en probabilidades condicionales, LDA busca **maximizar la separación entre clases mientras minimiza la varianza dentro de cada clase**. Es especialmente útil cuando se trata de clasificar datos con múltiples clases.



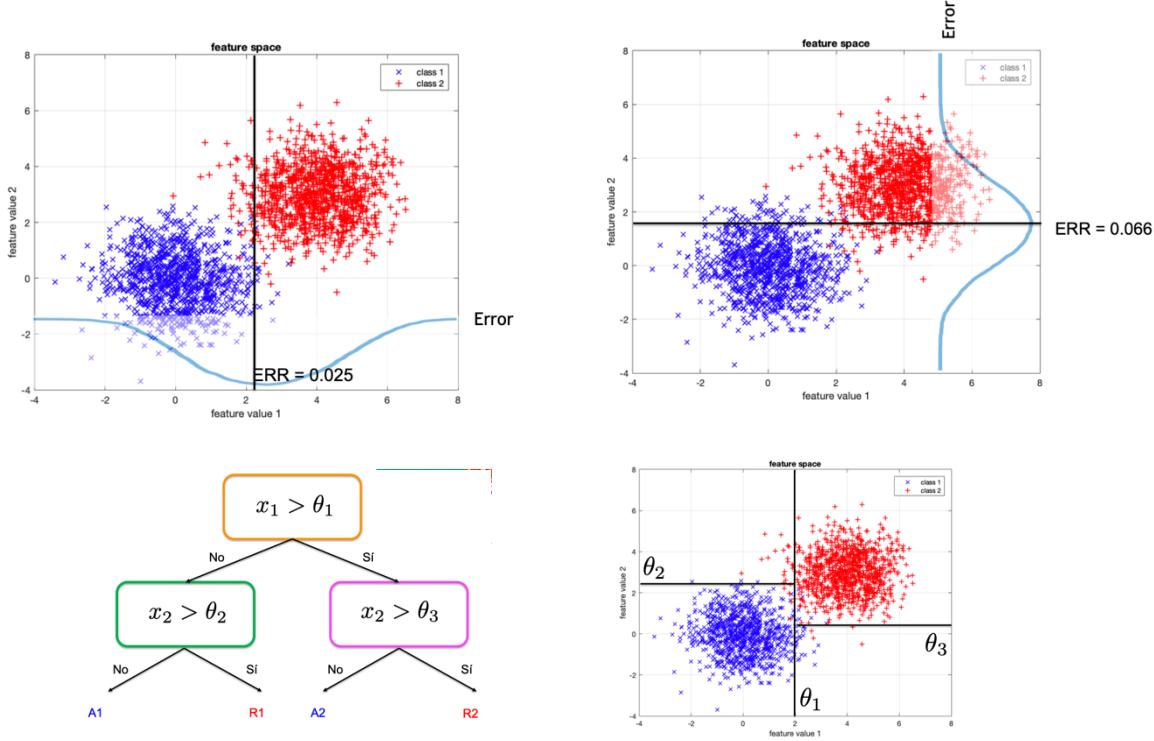
- **Mahalanobis:** A diferencia de otros métodos de clasificación, como el clasificador de Bayes o el clasificador de LDA, el clasificador de Mahalanobis considera las relaciones de covarianza entre las características y ajusta la métrica de distancia utilizada para medir la similitud entre los objetos.



- **QDA:** Es similar al LDA, pero en lugar de asumir una matriz de covarianza compartida entre todas las clases, QDA permite que cada clase tenga su propia matriz de covarianza. Esto hace que el clasificador de QDA sea más flexible en ciertos escenarios donde las clases pueden tener formas y tamaños diferentes en el espacio de características.



- **Árboles de decisión:** Separamos los datos según la línea que tenga menos error. Repetimos este proceso una cantidad arbitraria de veces.

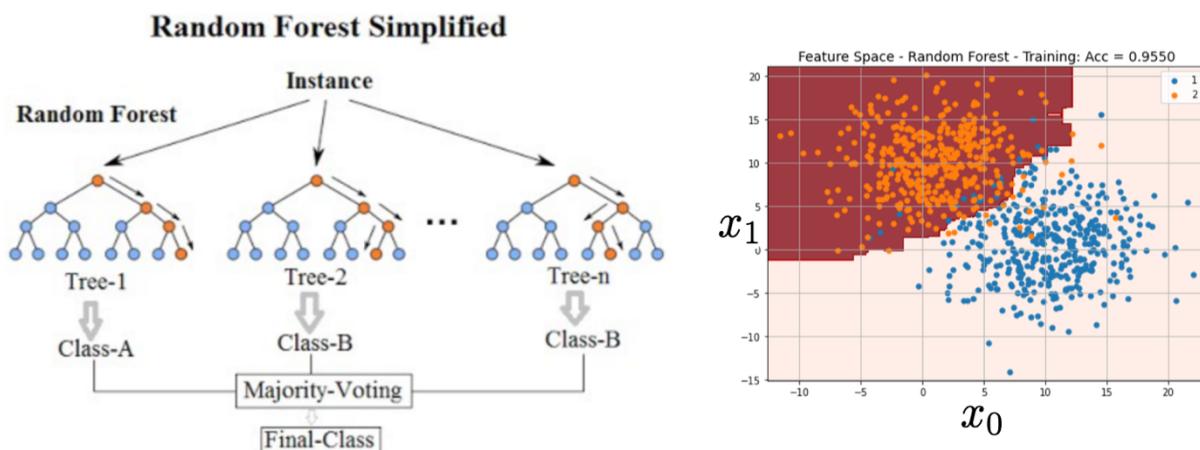


- **Random Forest:**

Training: Iteramos desde  $i = 1$  hasta  $n$ , y en cada iteración escogemos de manera aleatoria un subconjunto de *training*. Entrenamos un árbol de decisión  $A_i$ .

Testing: Iteramos desde  $i = 1$  hasta  $n$ , y en cada iteración clasificamos la muestra de *testing* usando  $A_i$ .

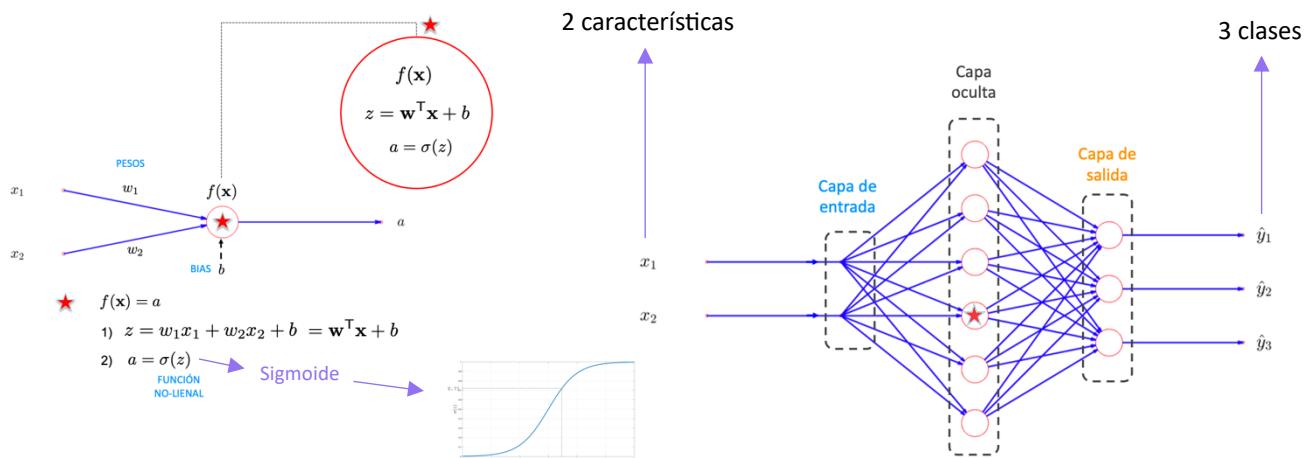
Clasificar la muestra según la mayoría de los  $n$  votos.



El algoritmo de *Random Forest* presenta varias ventajas significativas sobre el clásico árbol de decisión, algunas de estas son:

1. Reducción de sobreajuste (overfitting): Los árboles de decisión individuales tienden a ser propensos al sobreajuste, lo que significa que pueden ajustarse demasiado a los datos de entrenamiento y no generalizar bien a nuevos datos. En contraste, *Random Forest* combina múltiples árboles de decisión, lo que reduce en gran medida el riesgo de sobreajuste. La agregación de múltiples modelos ayuda a equilibrar los errores individuales y a mejorar la generalización.
2. Mayor precisión: Debido a que *Random Forest* combina las predicciones de varios árboles, suele proporcionar predicciones más precisas y estables que un solo árbol de decisión.

- **Redes Neuronales:**



- Cada flecha azul que entra a un nodo tiene asociado un “peso”.
- Cada nodo (círculo rojo) tiene asociado una función no-lineal y un valor de “bias”.
- Se conectan todos los nodos de una capa con todos los nodos de la siguiente capa.
- Cuando se diseña una red neuronal, se debe decidir la **función no lineal**, el número de **capas ocultas**, y el número de **nodos de las capas ocultas**.
- Cuando se entrena una red neuronal se debe estimar todos los pesos y los “bias” de tal forma que la salida real sea lo más parecida a la salida ideal.
- Si tenemos 3 clases, y tomamos un elemento de la “**Clase 3**” por ejemplo, la salida ideal sería:

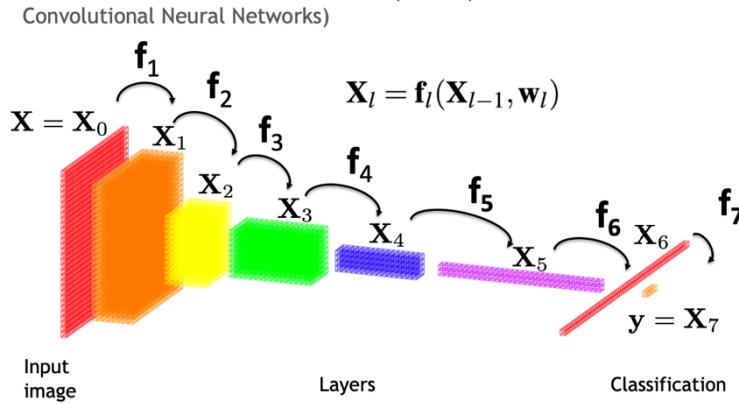
$$\mathbf{y} = (\mathbf{0}, \mathbf{0}, \mathbf{1})$$

Sin embargo, es posible que la salida real sea:

$$\hat{\mathbf{y}} = (\mathbf{0}, \mathbf{1}, \mathbf{0}, \mathbf{2}, \mathbf{0}, \mathbf{7})$$

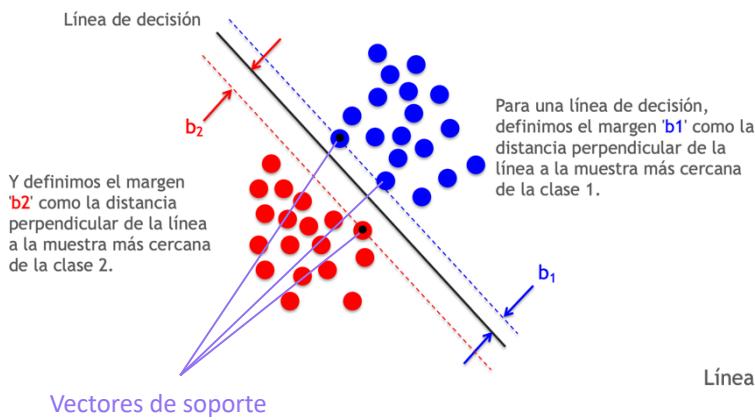
y como el máximo de la salida es el tercer elemento, entonces la clasificación será “**Clase 3**”.

- **Deep Learning:** Se enfoca en el uso de redes neuronales artificiales profundas para abordar tareas complejas de procesamiento de datos y toma de decisiones. El término “profundo” se refiere a la arquitectura de múltiples capas en estas redes neuronales, lo que les permite modelar relaciones y patrones altamente abstractos y no lineales en los datos.



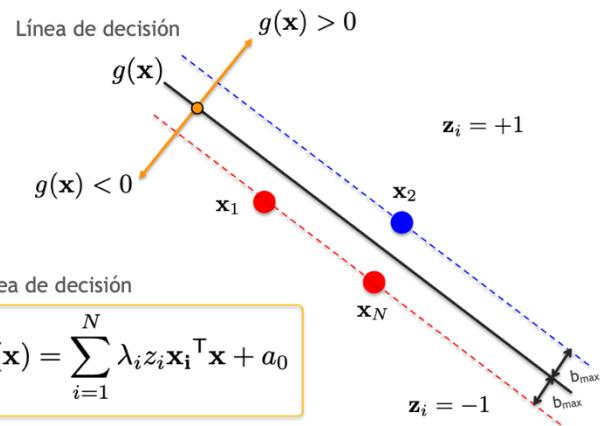
El término “épocas” se refiere a la cantidad de veces que un algoritmo de entrenamiento procesa el conjunto completo de datos de entrenamiento durante el proceso de ajuste de los pesos de la red neuronal. Cada época representa un ciclo completo a través de todos los datos de entrenamiento, lo que permite que la red neuronal ajuste gradualmente sus pesos para mejorar su capacidad de hacer predicciones precisas en la tarea específica.

- **SVM (Máquinas vectoriales de soporte):**



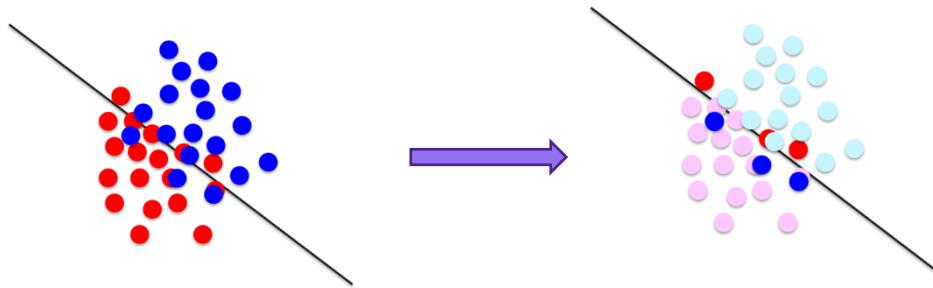
Ideas clave:

1.  $b_1 = b_2 = b$
2.  $b$  debe maximizarse
3. La solución es  $g(\mathbf{x})$



### ¿Qué pasa cuando nuestras muestras están entre mezclada?

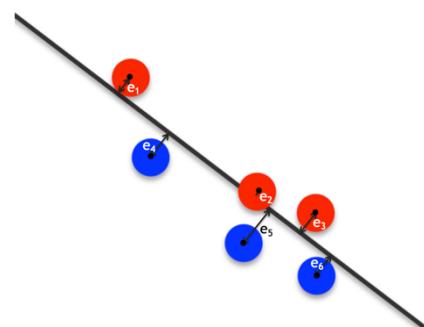
- Consideramos solo las muestras clasificadas **erróneamente**.



Ahora nuestros vectores de soporte serán cada  $e_i = \text{errores}$ .

Debemos encontrar la línea de decisión para que:

$$e = \sum e_i \rightarrow \min$$

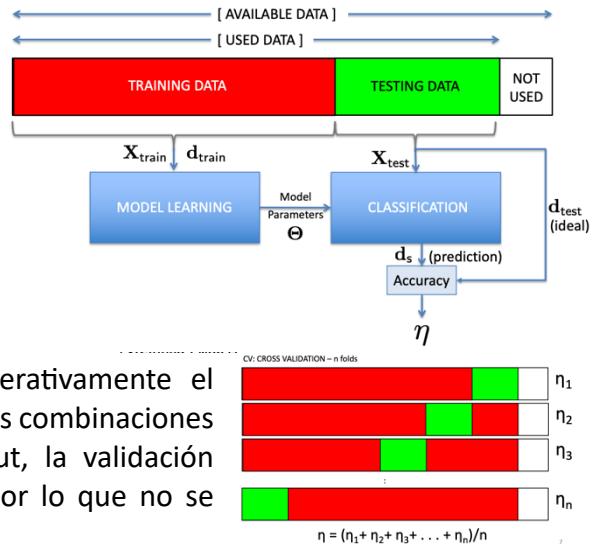


### ¿Qué pasa si quiero una línea de decisión NO lineal?



## Evaluación de desempeño de clasificación

- **Hold out:** implica dividir el conjunto de datos en dos partes principales: un conjunto de *training* y un conjunto de *testing*. Esta técnica es una forma simple y efectiva de evaluar el rendimiento de un modelo en datos no vistos y evitar el sobreajuste.



- **Cross Validation:** La idea básica detrás de la validación cruzada es dividir el conjunto de datos en múltiples subconjuntos (*folds*) y luego realizar iterativamente el proceso de entrenar y evaluar el modelo en diferentes combinaciones de pliegues. A diferencia del método de Hold Out, la validación cruzada requiere de mucho tiempo de cómputo, por lo que no se recomienda para set de datos muy grandes.
- **Leave 1 out:** Lo mismo que el *Cross Validation* pero sólo se deja 1 muestra de *testing* en cada *fold*. El *accuracy* por lo tanto será de 0% o 100% en cada *fold*.

- **Matriz de confusión:**

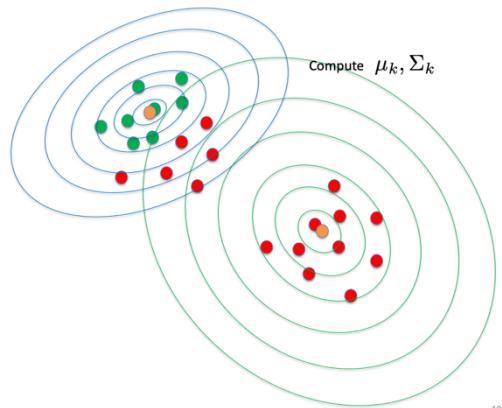
		predicted		actual	P = TP+FN (positive instances)	N = FP+TN (negative instances)	D = TP+FP (detections)
		$\omega_1$	$\omega_0$				
actual	$\omega_1$	TP	FN				
	$\omega_0$	FP	TN				

	$\omega_1$	$\omega_2$	$\omega_3$	...	$\omega_K$
$\omega_1$	1230	90	78		9
$\omega_2$	66	890	120		5
$\omega_3$	59	95	1527		90
:					
$\omega_K$	40	67	129		912

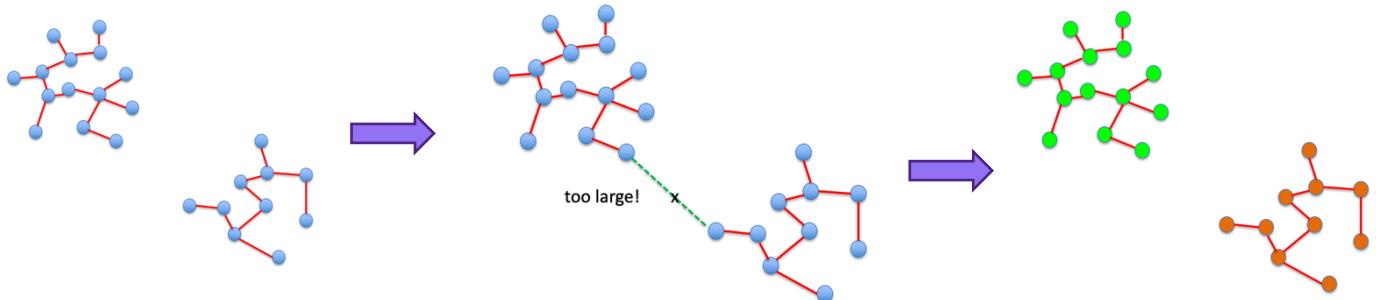
Example: there are 78 samples of class 1 that have been classified as class 3.

## Clustering

- **Mixture of Gaussians:** La idea principal detrás de este algoritmo es que los datos se generan a partir de varias distribuciones Gaussianas diferentes, y el objetivo es encontrar los parámetros (media y desviación estándar) de estas distribuciones para describir adecuadamente los grupos presentes en los datos.



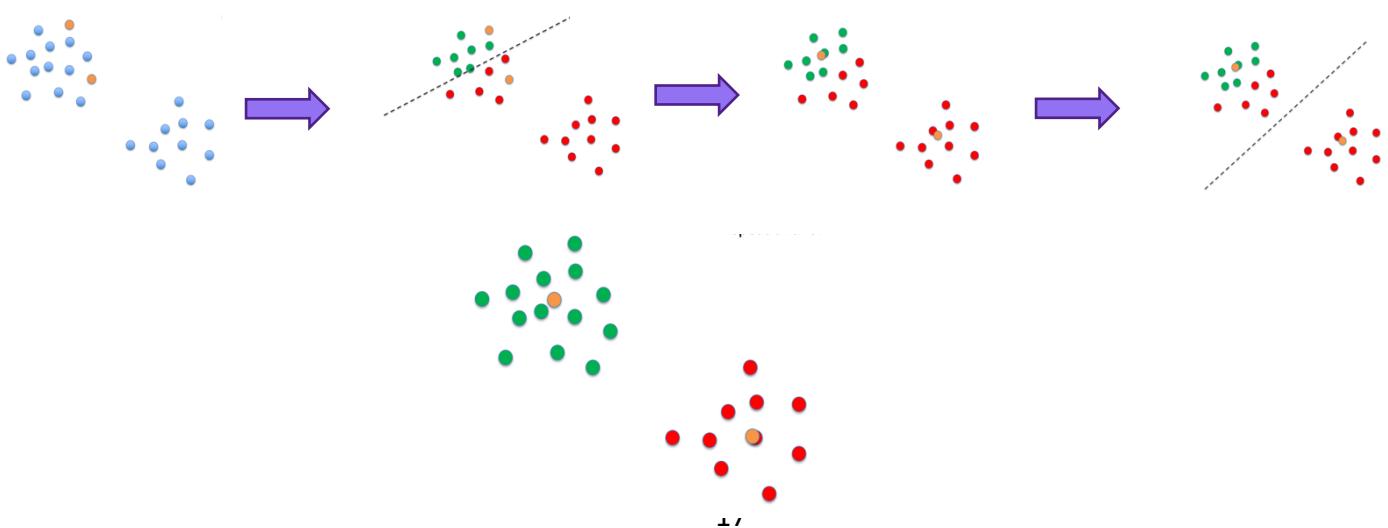
- **Hierarchical:**



A diferencia de otros métodos de *clustering* que generan un único conjunto de clústeres, el *clustering* jerárquico crea una estructura de árbol que representa la relación de agrupamiento entre los puntos de datos.

- **K-means:** Su objetivo principal es partitionar un conjunto de datos en un número predeterminado de clústeres, donde cada punto de datos pertenece a uno de los clústeres en función de su similitud. Pasos:

1. Selecciona aleatoriamente  $K$  muestras (las llamaremos **centroïdes**), donde  $K$  es el número deseado de clústeres. Cada centroïde representa el “centro” de un clúster potencial.
2. Asignamos cada muestra al punto al clúster cuyo centroïde está **más cerca**.
3. Calculamos los **centros de masa** de cada grupos de clases estos serán nuestros nuevos **centroïdes**.
4. Los pasos 2 y 3 se **repiten iterativamente hasta que la asignación de puntos a clústeres ya no cambie significativamente** o hasta que se alcance un número máximo de iteraciones predefinido.

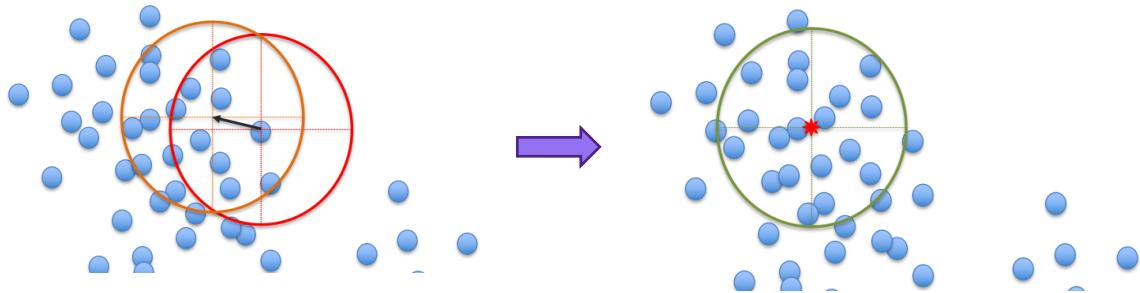


- **Mean Shift:** A diferencia de *K-means*, *Mean Shift* es capaz de descubrir automáticamente el número de clústeres y puede manejar clústeres de formas y tamaños irregulares.

La idea central detrás de *Mean Shift* es encontrar los máximos locales de una función de densidad de probabilidad estimada a partir de los datos. En otras palabras, el algoritmo busca los puntos en el espacio de características donde la densidad de datos es más alta, lo que se traduce en la ubicación de los centroides de los clústeres.

Los pasos son:

1. Seleccionar una muestra.
2. Calcular una vecindad arbitraria ( $x$  puntos más cercanos).
3. Computar el centro de masa de toda la vecindad (centroide).
4. Nos desplazamos hacia ese centroide (**mean shift**).
5. Repetimos los pasos 2, 3 y 4 hasta que el nuevo centro de masa NO cambie.



6. Para cada punto se hace este cálculo.
7. Agrupamos los puntos de muestra que tienen el mismo modo de desplazamiento de media.

