FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE

Para estudiantes de Educación Básica y Media.
UNIVERSIDAD DE CHILE

Clase V-I: Clasificación en *Machine Learning*

FM849 Proyecto de Ciencia de Datos: Inteligencia Artificial (IA) y sus aplicaciones

Máximo Flores Valenzuela (`mflores@dcc.uchile.cl`)

Universidad de Chile • 16 de agosto de 2025

Roadmap de la clase

1. Breve recuerdo de clasificación.
2. Modelos de clasificación: Árbol de decisión, k-Nearest Neighbors (k-NN), Naïve-Bayes, Support Vector Machine (SVM).

Breve recuerdo de clasificación

Clasificación

La clasificación es una técnica de aprendizaje **supervisado**, es decir, requiere que los datos que se usarán para el entrenamiento estén previamente etiquetados.

Sexo	Edad	Salario líquido mensual	Comprador
Hombre	19	$500.000	No
Mujer	25	$1.250.000	Sí
Hombre	49	$10.000.000	No
⋮	⋮	⋮	⋮

Tabla 1: Ejemplo de datos etiquetados mediante la columna «Comprador».

Clasificación

La idea central de la clasificación es, dado un ejemplo desconocido $D = (d_1, ..., d_n)$ (p. ej., la fila de una tabla con atributos), clasificarlo en alguna clase C, es decir, obtener su etiqueta.

- Por notación, $(d_1, ..., d_n)$ son atributos que representan a la observación D con n columnas. Cada coordenada de esa tupla puede ser pensada como un elemento en una lista.

Sexo (d_1)	Edad (d_2)	Salario (d_3)	Comprador (C)
Hombre	25	$1.000.000	???

Tabla 2: Nuevo ejemplo a clasificar, donde la clase de «Comprador» es desconocida.

Clasificación

Entonces, se puede encontrar (¿siempre?) una función lineal $f(x) = ax + b$ que separe estos datos:

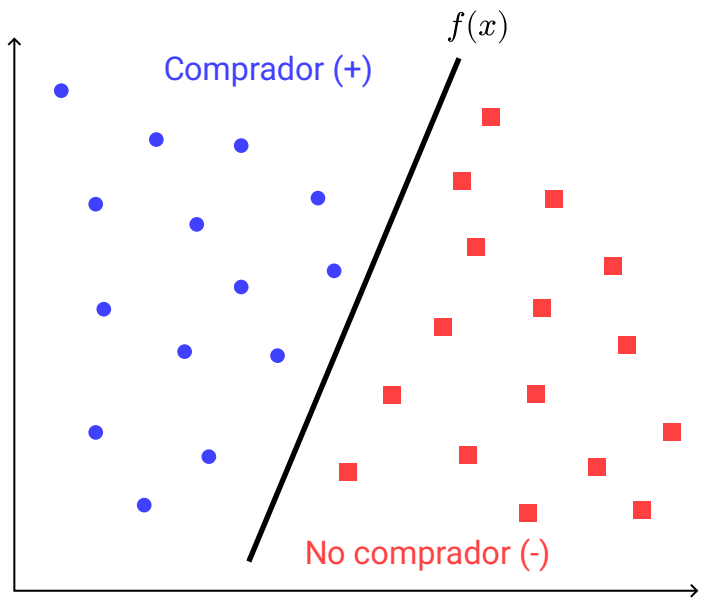

Figura 3: Clasificación binaria para la tabla de la diapositiva anterior.

Observación: En problemas binarios, podemos hablar de clases positivas (+) y negativas (−), pero **no** necesariamente sólo hay dos clases.

Clasificación

Lo cierto es que no siempre los modelos lineales son efectivos para encontrar una separación.

- La clase pasada hablamos sobre lógica en programación. Vimos el «y» lógico, y el «o» lógico (Tutorial I). Hay un operador de la lógica llamado XOR («o» exclusivo) que tiene la siguiente tabla de verdad:

p	q	p XOR q
V	V	F
V	F	V
F	V	V
F	F	F

Tabla 3: Definición del conectivo lógico XOR.

es decir, **sólo** uno de los dos valores puede ser verdadero.

Clasificación

Una de las limitaciones del *Machine Learning* con funciones lineales es la imposibilidad de clasificar datos que, por ejemplo, siguen la naturaleza del operador `XOR`:

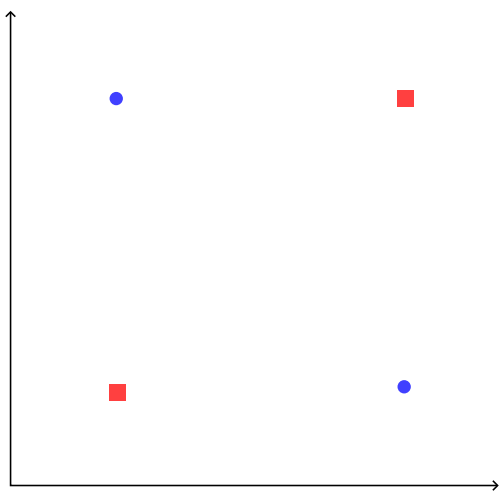

Figura 4: ¿Cómo se clasifican linealmente estos datos que modelan el operador `XOR`?

después de esta limitación, se probaron los modelos no lineales y nace el aprendizaje profundo (*Deep Learning*, quizás quede para otro curso 👀).

Modelos de clasificación

Árbol de decisión (*Decision Tree*)

- Corresponde a una estructura con nodos y aristas que conectan a estos nodos.
- En cada nodo se realizan bifurcaciones (no necesariamente binarias), salvo en los nodos externos, donde ya se tomó la **decisión**.
- Cada arista permite comprender a qué nodo se debe avanzar en cada bifurcación.

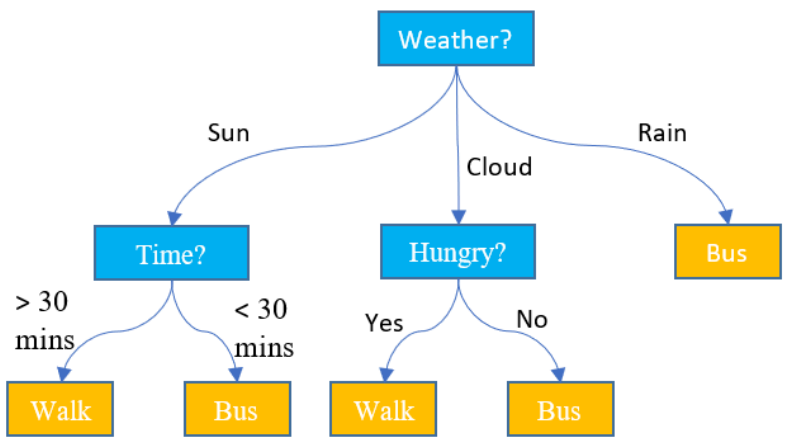

Figura 5: Árbol de decisión.

El ejemplo de tiempos remotos…

Akinator usa un árbol de decisión. Tiene una base de datos con varias entidades (p. ej., personas), y a través de una serie de preguntas puede llegar a la solución.

Figura 6: Akinator como ejemplo de uso de un árbol de decisión.

Funcionamiento intuitivo

Un árbol de decisión en cada bifurcación genera una división paralela a los ejes.

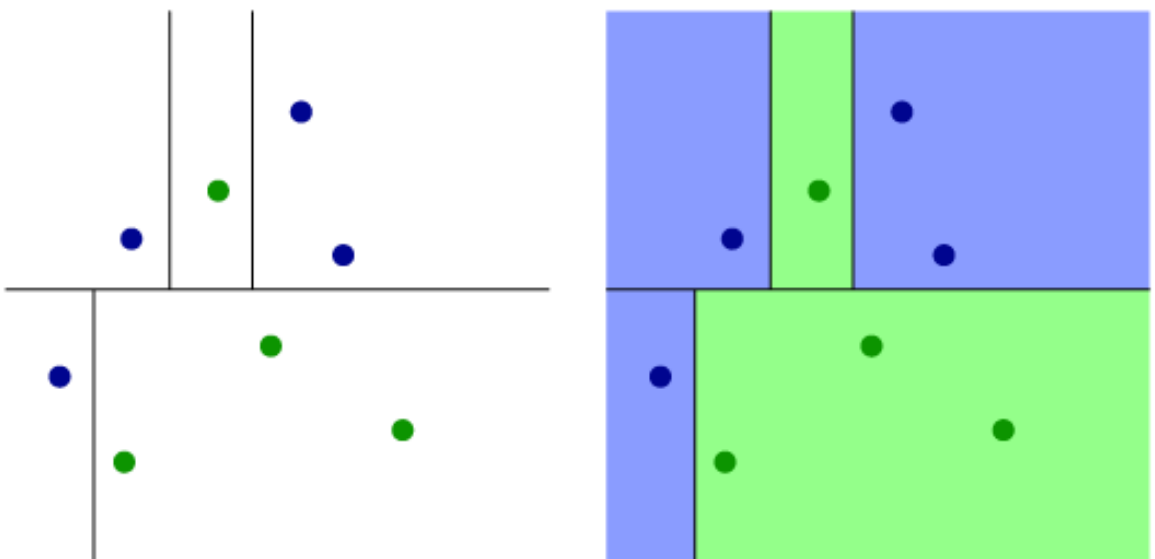

Figura 7: Ejemplo de divisiones paralelas a los ejes en dos dimensiones.

La región resultante en la decisión final es muy pequeña, e idealmente nos dará la clase que estábamos buscando.

Random Forest

- Corresponde al modelo que se obtiene de combinar muchos árboles de decisión.
  - Cada árbol se entrena con distintas muestras aleatorias, con repetición, del *dataset* de entrenamiento y un subconjunto aleatorio de las características.
  - Al final, cada árbol de decisión emite un voto y se realiza una predicción final.
- Tiene mayor capacidad predictiva que un sólo árbol de decisión.

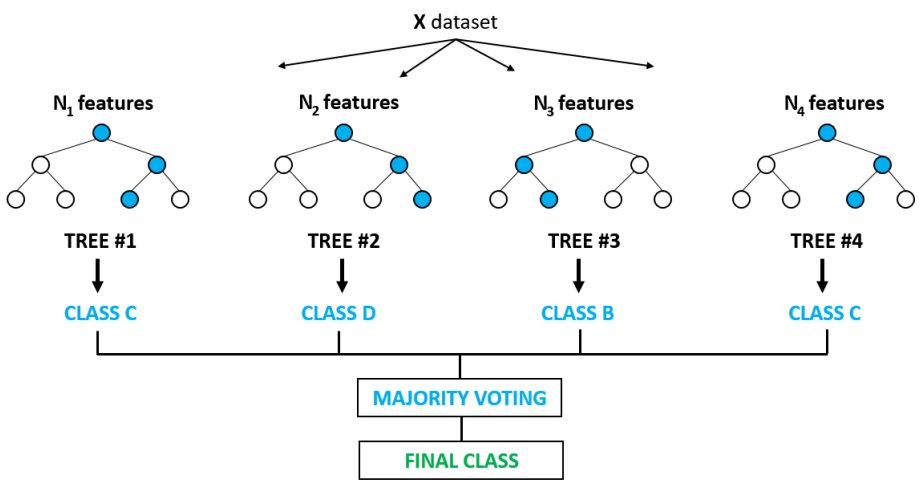

Figura 8: Visualización de un *Random Forest*.

Gradient Boosting

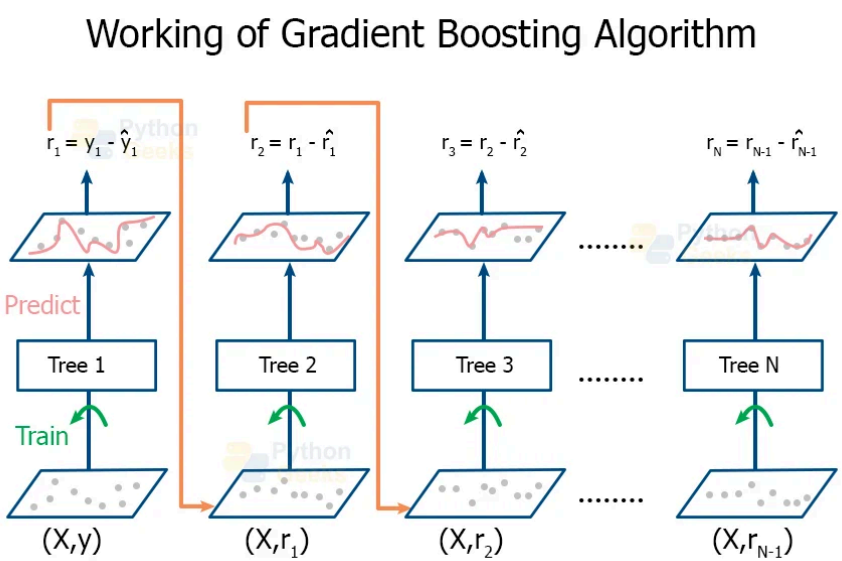

Figura 9: Visualización de *Gradient Boosting*.

- El *boosting* en *Machine Learning* corresponde a una técnica que usa modelos secuenciales, donde el modelo siguiente se enfoca en corregir los errores de los anteriores.
- Para *Gradient Boosting*, los modelos son árboles de decisión.
- La predicción final es una suma ponderada de las predicciones de cada árbol.
- Existen implementaciones eficientes como `XGBoost` y `CatBoost` para atributos categóricos.

Ventajas, desventajas y supuestos

Ventajas

- Su interpretación es intuitiva.
- Su costo computacional crece de manera logarítmica con respecto a la cantidad de datos de entrenamiento.

Desventajas

- Las divisiones pueden aprender de memoria el comportamiento de los datos de entrenamiento (*overfitting**).
- Son inestables en estructura: un pequeño cambio puede generar un árbol totalmente distinto.
- Son sesgados si hay una clase que tiene mucha dominancia.

Supuestos

- La relación entre los datos y la clase puede aproximarse bien con particiones rectangulares.

k-Nearest Neighbors (k-NN)

- Corresponde a una técnica basada en distancias.
- Dado un ejemplo desconocido a clasificar, busca los k ejemplos conocidos más cercanos. Por defecto, cada clase suma 1 voto si un ejemplo suyo está entre los k más cercanos.
- El ejemplo nuevo es clasificado en la clase dominante, es decir, la que tenga más «votos».
- Se pueden usar funciones más sofisticadas para definir los votos en caso de empate.

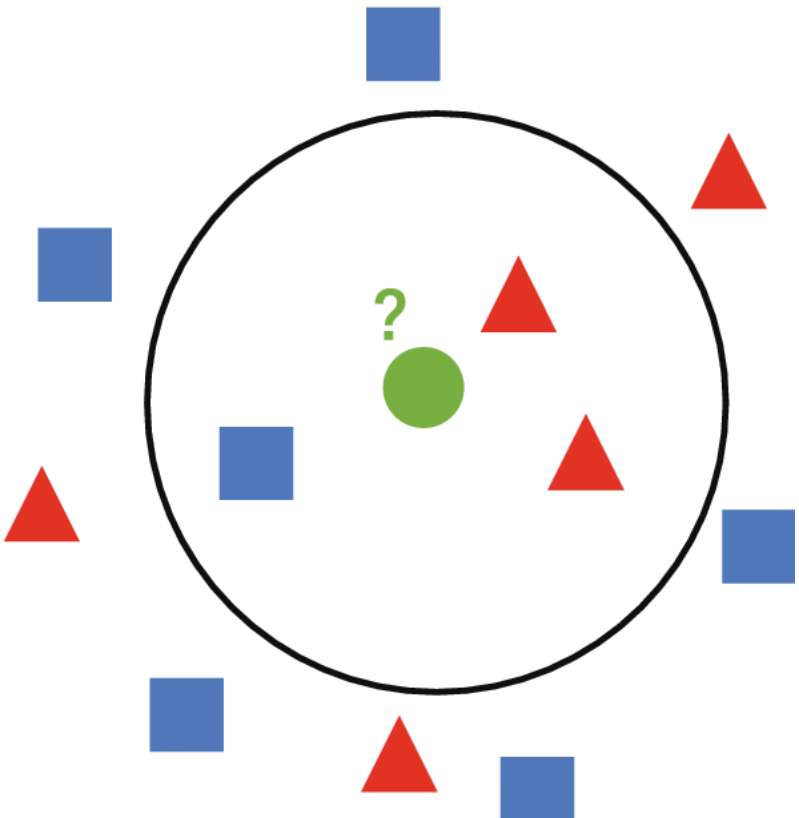

Figura 10: Ilustración de k-NN.

Maldición de la dimensionalidad

La maldición de la alta dimensionalidad es un problema muy común en Ciencia de Datos. Estamos acostumbrados a pensar en la distancia euclideana cada vez que nos mencionan «distancia»:

$$d_{\text{euclid.}}(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2} \quad \text{(en 2 dimensiones)}$$

$$d_{\text{euclid.}}(x, y) = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2} \quad \text{(en } N \text{ dimensiones)}$$

Problema: En altas dimensiones, esta función (y en general, muchas funciones de distancia) pierden la capacidad de distinguir elementos que están muy cerca o muy lejos. Las distancias se empiezan a concentrar alrededor de un valor.

Similitud y distancia coseno

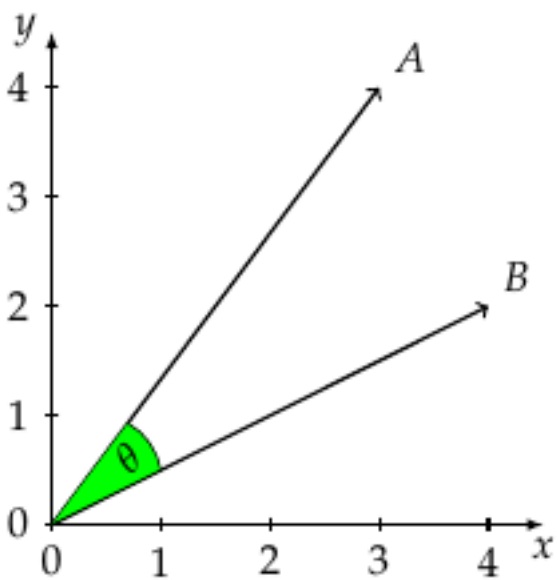

Figura 11: Ángulo θ entre dos vectores.

La solución más común al problema de la maldición de la dimensionalidad es usar distancias basadas en el ángulo.

Primero, se define la **similitud coseno** (basada en el producto punto):

$$A \cdot B = \|A\| \cdot \|B\| \cdot \cos(\theta) \Longrightarrow \cos(\theta) = \frac{A \cdot B}{\|A\| \cdot \|B\|}$$

¡pero esto no es una distancia! Porque mide qué tan similares son dos elementos. La **distancia coseno** se define usando la similitud:

$$d_{\text{coseno}}(A, B) = 1 - \cos(\theta)$$

Dominancia de características

Otro problema de k-NN es que, dado que está basado en distancias, las características de los datos que tengan escalas más distanciadas van a dominar el resultado.

- min-max *scaling*:

$$x_{\text{nuevo}} = \frac{x - \min(X)}{\max(X) - \min(X)} \in [0, 1]$$

- Estandarización a una normal $\mathcal{N}(0, 1)$ (campana de Gauss):

$$x_{\text{nuevo}} = \frac{x - \mu_X}{\sigma_X} \in \mathbb{R}$$

- Usando la norma L_2 (cuando se quiere comparar por ángulo y la escala no es tan relevante): $x_{\text{nuevo}} = x/\|x\|_2$.

Ventajas, desventajas y supuestos

Ventajas

- Conceptualmente simple de entender.
- No tiene supuestos tan fuertes.
- Capta relaciones locales en los datos.

Desventajas

- Requiere escalado de los datos.
- Sufre de la maldición de la dimensionalidad.
- Puede ser lento al predecir.
- Es sensible a datos irrelevantes.

Supuestos

- La medida de disimilitud es adecuada para el número de dimensiones.
- La distribución espacial de los puntos permite encontrar distancias descriptivas.

Naïve-Bayes

- Corresponde a una técnica basada en probabilidades.
- Supone que las observaciones entre sí son **condicionalmente independientes** según las clases.
- Para la predicción de la clase (*posterior*), mezcla el conocimiento previo (*prior*), con la verosimilitud (qué tan bien dicha clase explica los datos).

Matemáticamente, se deduce de la siguiente forma (regla de Bayes):

$$p(A \mid B) = \frac{p(A \cap B)}{p(B)} \ \wedge \ p(B \mid A) = \frac{p(A \cap B)}{p(A)} \implies p(A \mid B) \cdot p(B) = p(B \mid A) \cdot p(A)$$

$$\implies p(A \mid B) = \frac{p(B \mid A) \cdot p(A)}{p(B)}$$

Aplicándolo a nuestro caso...

En nuestro análisis, estamos trabajando con una serie de observaciones $X = (x_1, ..., x_N)$ y tenemos clases $C = (c_1, ..., c_k)$ ($k = 2$ en el caso binario). Entonces:

$$\underbrace{p(c_i \mid X)}_{\text{posterior}} = \frac{\overbrace{p(X \mid c_i)}^{\text{verosimilitud}} \cdot \overbrace{p(c_i)}^{\text{prior}}}{\underbrace{p(X)}_{\text{cte. de normalización}}}, \quad \text{para todo } i \in \{1, ..., k\}$$

Bajo el supuesto de **independencia condicional**, se puede escribir la verosimilitud de una forma más fácil de calcular:

$$p(X \mid c_i) = \underbrace{p(x_1, ..., x_N \mid c_i)}_{\text{densidad conjunta}} = \underbrace{p(x_1 \mid c_i) \cdot p(x_2 \mid c_i) \cdot ... \cdot p(x_N \mid c_i)}_{\text{aplicando el supuesto de independencia condicional}} = \prod_{j=1}^{N} p\left(x_j \mid c_i\right)$$

Ya, pero ¿de qué sirve esto?

- Para un computador, calcular multiplicaciones es más fácil que calcular una densidad conjunta, donde generalmente se deben usar métodos numéricos.
- Bajo el supuesto, se puede calcular $p(c_i \mid X)$ para cada $i \in \{1, ..., k\}$. Es importante elegir un buen *prior* (p. ej., con distribución uniforme si no tenemos información previa) y un buen modelo para la verosimilitud. Este «buen modelo» depende del problema a resolver.
- La clase elegida para clasificar es la que maximiza la probabilidad de la *posterior*:

$$c_{\text{elegida}} = \arg\max_c p(c \mid X) \propto \arg\max_c p(X \mid c) \cdot p(c)$$

Se ignora el término $p(X)$ porque es el mismo para todas las clases.

Ventajas, desventajas y supuestos

Ventajas

- Es rápido de entrenar y predecir.
- Funciona bien en altas dimensiones.
- Es robusto a datos irrelevantes.

Desventajas

- Tiene un supuesto fuerte que en la práctica casi no se cumple.
- Requiere calibración de las probabilidades en casos extremos (como cuando alguna vale 0).

Supuestos

- Independencia condicional sobre las observaciones dada una clase cualquiera.

Support Vector Machine (SVM)

- Corresponde a un método que busca separar las clases con un hiperplano (en 2D, una recta).
- El hiperplano elegido es el que maximiza el margen que se genera con los vectores de soporte.
- Intuitivamente, cada vez que evaluemos un nuevo punto en la ecuación del hiperplano $w^\top \cdot x + b = 0$, clasificamos según el signo de $w^\top \cdot x + b$.

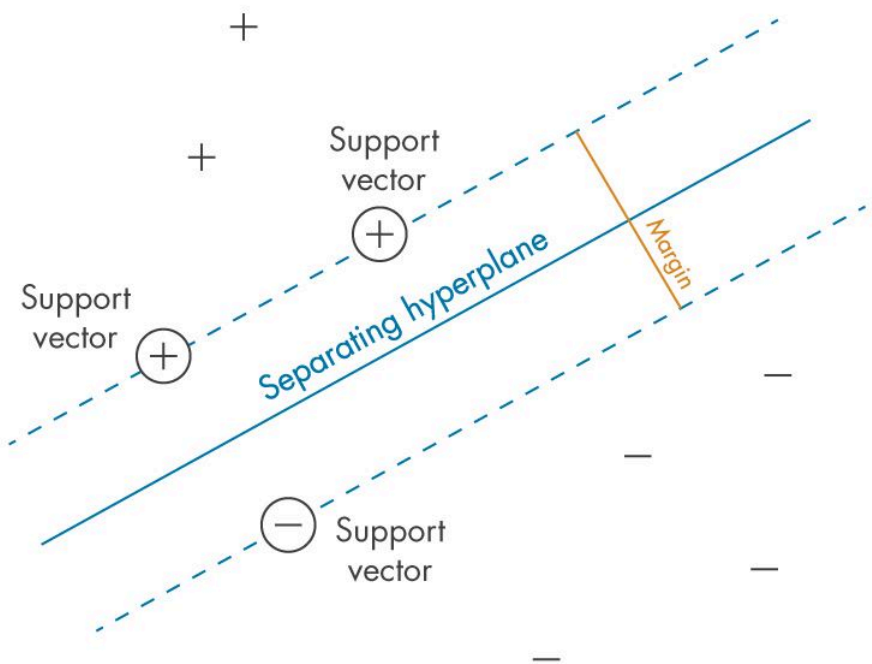

Figura 12: Descripción gráfica de una SVM.

El problema del ruido

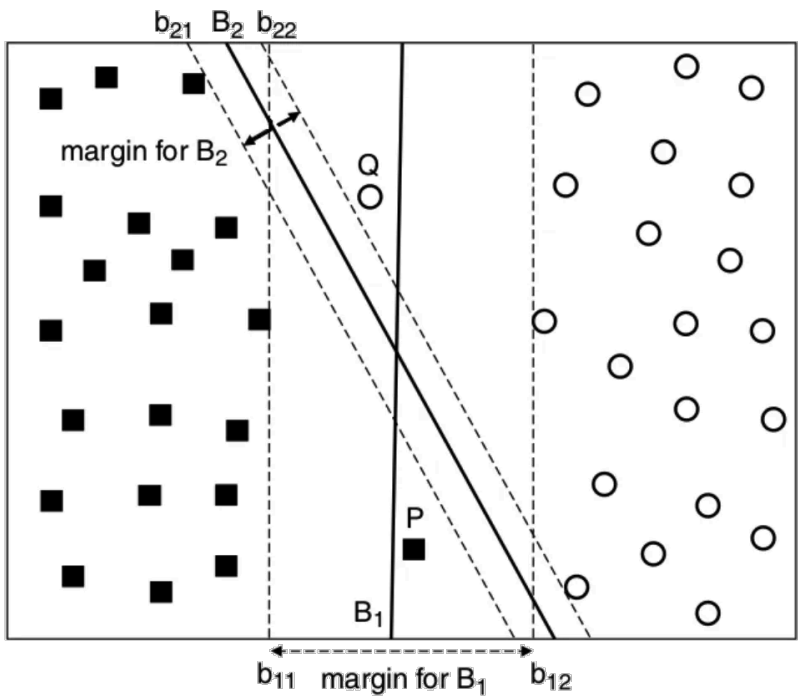

Figura 13: Hiperplano «óptimo» con ruido.

Cuando hay ruido, es decir, clasificaciones que son *outliers* con respecto a su clase, el hiperplano que maximiza el margen no es necesariamente el mejor.

- Puede generar problemas de sobreajuste (*overfitting*).
- Se debe introducir cierta «holgura» al problema de optimización. Esta holgura está controlada por un parámetro C.

Truco del *kernel*

- En ocasiones, no es posible generar separaciones por hiperplanos en el espacio original de los datos.
- De esta forma, se genera una transformación de los datos a un nuevo espacio, donde la separación sí se puede realizar.
  - El espacio transformado puede ser un reescalado del espacio original, o involucrar un aumento de las dimensiones (cuidado con la maldición de la dimensionalidad).
- Controlado por el hiperparámetro `kernel`: `linear`, `poly`, `rbf` (gaussiano), etc.

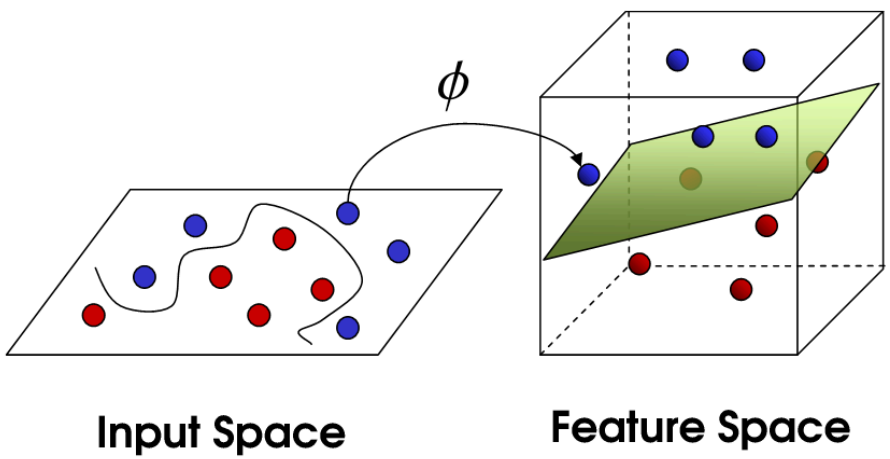

Figura 14: Aplicación del truco del *kernel*.

Ventajas, desventajas y supuestos

Ventajas

- Si se encuentran márgenes grandes hay un buen poder de generalización.
- Robusto al *overfitting* cuando se usan buenos ajustes de la regularización.
- Soporta clases desbalanceadas usando `class_weight`.

Desventajas

- Requiere una base matemática sólida para resolver el problema de optimización [SKLearn, SVM Mathematical Formulation].
- Es mandatorio el escalado de variables.
- En SVM no lineales el entrenamiento es costoso.

Supuestos

- El ruido debe ser moderado para poder generar una separación razonable entre clases.

¿Preguntas?