

Tabla de contenido

1.	Introducción a SVM- SUPPORT VECTOR MACHINES.....	1
2.	Tipos de modelos en SVM según los datos.....	4
	Conjuntos de datos linealmente separables.....	4
	Conjuntos de datos linealmente no separables.....	5
3.	Funcionamiento del algoritmo.....	5
	¿Qué aporta SVM en este contexto?.....	7
4.	SVM: Construcción de la función hipótesis.....	7
	Idea fundamental: el margen.....	8
	Ventaja frente a otros modelos.....	8
	Cómo construye SVM el límite de decisión	9
5.	SVM Hard Margin Classification.....	9
6.	SVM Soft Margin Classification	12
	El papel del hiperparámetro C	13
7.	SVM – Kernels y separación no lineal	14
	Límites de decisión no lineales.....	15
	La solución de SVM: el truco del kernel	16

1. Introducción a SVM- SUPPORT VECTOR MACHINES.

Las **Support Vector Machines (SVM)** son uno de los algoritmos más importantes y utilizados dentro del ámbito del **Machine Learning**. Su popularidad se debe a su **gran potencia y versatilidad**, ya que permiten resolver problemas tanto de **clasificación** como de **regresión**, algo que no todos los algoritmos supervisados ofrecen.

A diferencia de otros algoritmos estudiados, como la **regresión lineal** o la **regresión logística**, las SVM destacan por su **capacidad para trabajar eficazmente con conjuntos de datos pequeños y complejos**. Esta característica es especialmente relevante, ya que en la mayoría de técnicas de Machine Learning ocurre justo lo contrario: suelen funcionar mejor cuando los conjuntos de datos son grandes y relativamente simples. Cuando los datos son **pocos, muy complejos** o presentan **fronteras de decisión difíciles**, muchos algoritmos reducen notablemente su rendimiento. Sin embargo, las SVM mantienen una alta precisión incluso en estas situaciones.

Por este motivo, las Support Vector Machines se consideran una **herramienta fundamental para cualquier profesional del análisis de datos** y del aprendizaje automático.

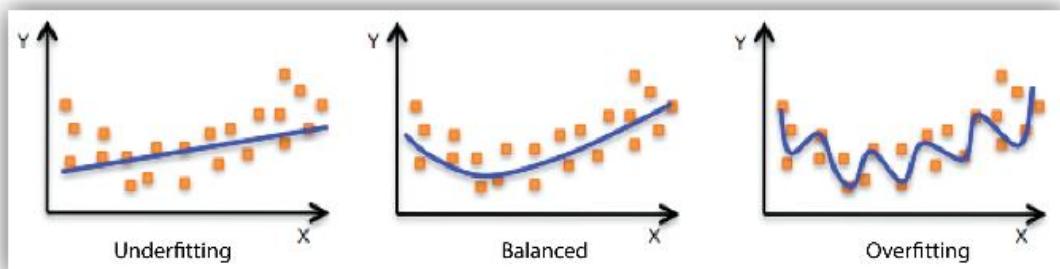
Características principales de SVM

- Es un algoritmo de **aprendizaje supervisado**, por lo que trabaja con **datos etiquetados**.
- Puede utilizarse tanto para:
 - **Clasificación (SVC)**.
 - **Regresión (SVR)**.
- Ofrece un excelente rendimiento en **datasets pequeños o medianos** con estructuras complejas.
- Es capaz de construir:
 - **Modelos lineales**.
 - **Modelos no lineales**.
- Funciona correctamente tanto con datos:
 - **Linealmente separables**.
 - **No linealmente separables**.

Antes de comenzar con este algoritmo vamos a ver los dos problemas más habituales que surgen en este y otros algoritmos de aprendizaje automático.

En *machine learning*, **aprender** no es memorizar datos, sino **detectar patrones** para poder **hacer predicciones correctas con datos nuevos**. Aquí aparecen dos problemas muy habituales:

- **Subajuste → el modelo no aprende lo suficiente.**
- **Sobreajuste → el modelo aprende demasiado (mal).**



Subajuste (Underfitting).

El modelo es **demasiado simple** y **no entiende** los datos. Es como estudiar solo el título del tema antes de un examen.

¿Qué le pasa al modelo?

- No detecta los patrones importantes.
- Hace predicciones malas **incluso con los datos que ya conoce**.

Ejemplo sencillo (vida real)

Quieres predecir la **nota final de un alumno** usando solo:

- El número de días que ha venido a clase.

Ignoras:

- Si estudia en casa.
- Si hace los ejercicios.
- Si entiende la materia.

El modelo no tiene información suficiente → **subajuste**

Ejemplo en machine learning.

- Usar una **línea recta** para datos que claramente hacen una curva.
- Clasificar con una regla muy simple algo que es complejo.

Cómo se detecta.

- Error alto en **entrenamiento**.
- Error alto en **pruebas**.
- Da igual cuántos datos tengas: el modelo sigue fallando.

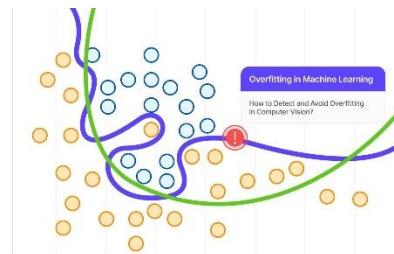
Causas más comunes.

- Modelo demasiado simple.
- Muy pocas variables.
- Variables mal elegidas.
- Demasiada regularización.

Cómo solucionarlo.

- Usar un modelo más complejo.
- Añadir variables útiles.
- Transformar variables.
- Quitar restricciones excesivas.

Sobreajuste (Overfitting)



El modelo **memoriza** en lugar de aprender. Es como aprenderse **las respuestas del examen de memoria**, pero no entender la materia.

¿Qué le pasa al modelo?

- Aprende incluso el **ruido** de los datos.
- Funciona muy bien con los datos de entrenamiento...pero muy **mal con datos nuevos**.

Ejemplo sencillo (vida real)

Memorizas todos los ejercicios resueltos del libro. En el examen:

- Cambian los números.
- Cambian el enunciado.

No sabes resolverlo → **sobreajuste**

Ejemplo en machine learning.

- Un modelo que acierta el **100 %** de los datos de entrenamiento.
- Pero falla mucho cuando se prueba con datos nuevos.

Cómo se detecta.

- Error muy bajo en entrenamiento.
- Error alto en validación o test.
- Gran diferencia entre ambos errores.

Causas más comunes.

- Modelo demasiado complejo.
- Pocos datos.
- Datos con mucho ruido.
- Demasiadas variables irrelevantes.

Cómo solucionarlo.

- Usar más datos.
- Simplificar el modelo.
- Eliminar variables poco útiles.
- Usar regularización.
- Validación cruzada.

Situación	¿Qué hace el modelo?	Resultado
Subajuste	No aprende lo suficiente	Falla siempre
Buen ajuste	Aprende patrones generales	Predice bien
Sobreajuste	Memoriza datos	Falla con datos nuevos

2. Tipos de modelos en SVM según los datos.

El tipo de modelo que construye una SVM depende de la naturaleza del conjunto de datos proporcionado.

Conjuntos de datos linealmente separables.

Son aquellos en los que las clases pueden separarse mediante una línea recta (o un hiperplano en dimensiones superiores). En este caso, la SVM puede trabajar con:

- **Hard Margin Classification.**
 - Separa las clases de forma estricta.
 - No permite errores de clasificación.
 - Solo es adecuada cuando los datos están perfectamente separados y no contienen ruido.
- **Soft Margin Classification.**
 - Permite ciertos errores de clasificación.
 - Es más robusta frente al ruido y a valores atípicos.
 - Es la opción más utilizada en la práctica real.

Conjuntos de datos linealmente no separables.

Son aquellos en los que no es posible separar las clases mediante una línea recta.

En este caso, la SVM utiliza:

- **Kernels**
 - Transforman los datos a un espacio de mayor dimensión.
 - Permiten encontrar una separación lineal en ese nuevo espacio.
 - Hacen posible construir **modelos no lineales**.

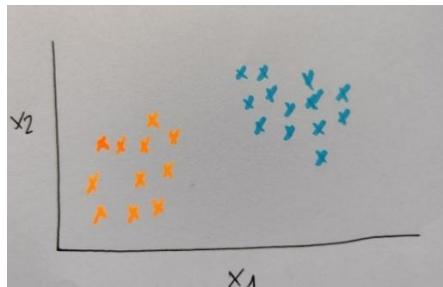
Ejemplos de kernels:

- Lineal.
- Polinómico.
- RBF (Gaussiano).
- Sigmoide.

Las Support Vector Machines buscan siempre el **mejor hiperplano posible**, es decir, el que **maximiza la distancia (margen)** entre las clases, lo que les permite generalizar mejor y ofrecer modelos muy precisos incluso en escenarios complejos.

3. Funcionamiento del algoritmo

Un **conjunto de datos linealmente separable** es aquel en el que las distintas clases pueden separarse mediante una **línea recta** (o un hiperplano, si trabajamos en más dimensiones). Básicamente un conjunto de datos linealmente separables es un conjunto de datos similar al que veis:



Tenemos un **conjunto de datos de dos características de entrada**, x_1 y x_2 . Si lo llevamos a un ejemplo práctico como el **filtro de spam**, podríamos interpretar:

- x_1 : número de signos de interrogación y exclamación en el correo.
- x_2 : número de etiquetas HTML.

Si tomamos el conjunto de datos de entrenamiento, extraemos estas dos características y las representamos gráficamente, podemos distinguir dos tipos de correos:

- **Correos no spam**, representados en color naranja.
- **Correos spam**, representados en color azul.

Este conjunto de datos es **linealmente separable** porque existe al menos una **función hipótesis lineal** (una línea recta) capaz de separar correctamente ambos grupos.

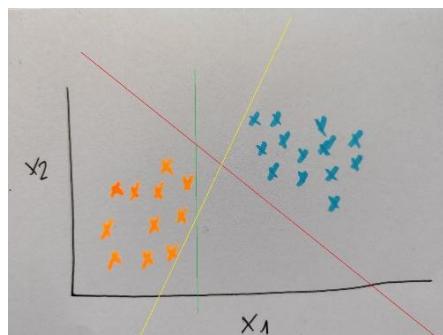
Por ejemplo, podemos trazar una recta que deje:

- A la **derecha**, los correos spam.
- A la **izquierda**, los correos no spam.

De esta forma, el conjunto queda claramente dividido por una línea recta, lo que confirma que los datos son linealmente separables.



Cuando los datos son linealmente separables, es posible construir **múltiples funciones hipótesis lineales** que separen correctamente las clases en el conjunto de entrenamiento. Sin embargo, **no todas estas funciones son igual de buenas**.

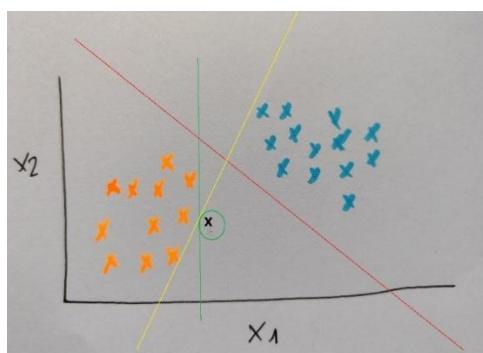


Por ejemplo, si observamos un límite de decisión (representado en color verde), vemos que:

- Clasifica correctamente todos los ejemplos del conjunto de entrenamiento.
- Sin embargo, el límite de decisión está **demasiado cerca** de los correos no spam.

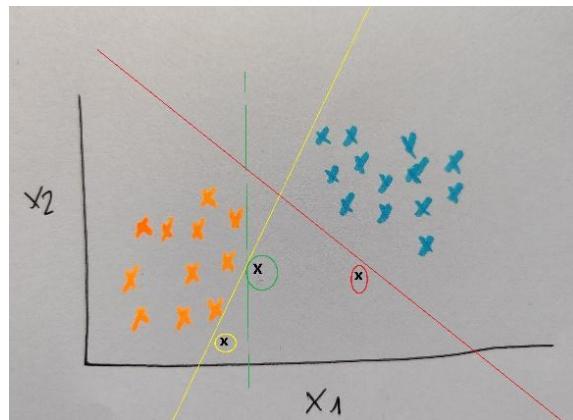
Esto supone un problema importante.

Si llega un **correo nuevo** (representado por una "x" en color negro) que cae ligeramente a la derecha del límite, el modelo lo clasificará como **spam** simplemente por su posición respecto a la línea.



No obstante, si observamos la distribución de los datos, ese nuevo correo está **mucho más cerca de los correos no spam** que de los correos spam. Por tanto, aunque la clasificación sea correcta según la regla del modelo, **no sigue la tendencia real de los datos**, lo que provoca una mala generalización.

Este mismo problema puede ocurrir con otros límites de decisión posibles: aunque separen correctamente los datos de entrenamiento, pueden fallar al clasificar ejemplos nuevos.



¿Qué aporta SVM en este contexto?

Las **Support Vector Machines (SVM)** surgen precisamente para **resolver este problema**. En lugar de buscar cualquier recta que separe las clases, SVM busca el **mejor límite de decisión posible**, es decir, aquel que:

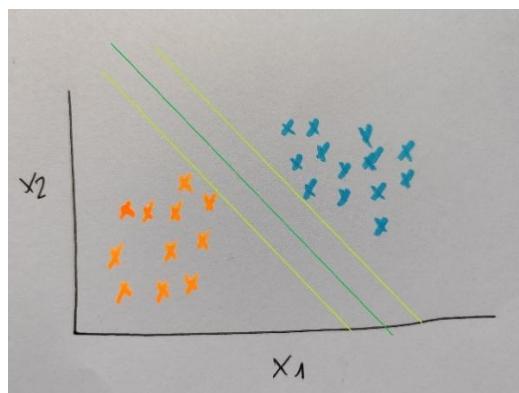
- Maximiza la distancia entre las clases.
- Se sitúa lo más lejos posible de los ejemplos más cercanos de cada clase.
- Generaliza mejor ante datos nuevos.

Este enfoque permite a SVM construir modelos **más robustos y fiables**, evitando límites de decisión demasiado ajustados que puedan provocar errores en la clasificación de nuevos ejemplos.

4. SVM: Construcción de la función hipótesis.

Las **Support Vector Machines (SVM)** no se limitan únicamente a encontrar una recta (o hiperplano) que separe dos clases. Su objetivo principal va **mucho más allá**: SVM intenta construir un **límite de decisión que esté lo más alejado posible de los ejemplos del conjunto de entrenamiento**, de forma que el modelo generalice mejor cuando aparezcan datos nuevos. Este enfoque recibe el nombre de **large margin classification (clasificación con margen máximo)**.

El algoritmo va a construir un límite de decisión bastante diferente, concretamente el límite de decisión o modelo es el que veis representado en verde:



Idea fundamental: el margen

Si recuperamos el ejemplo anterior del filtro de spam, vimos que era posible construir **varios modelos lineales** que separaban correctamente los correos spam y no spam en el conjunto de entrenamiento. Sin embargo, muchos de esos modelos fallaban cuando llegaban **ejemplos nuevos**, ya que el límite de decisión estaba demasiado cerca de una de las clases.

SVM soluciona este problema construyendo un **límite de decisión diferente**, como el que se representa en color verde, que no solo separa las clases, sino que además:

- Mantiene la **máxima distancia posible** respecto a los ejemplos más cercanos de ambas clases.
- Evita límites de decisión demasiado ajustados a los datos.
- Reduce el riesgo de malas predicciones en datos nuevos.

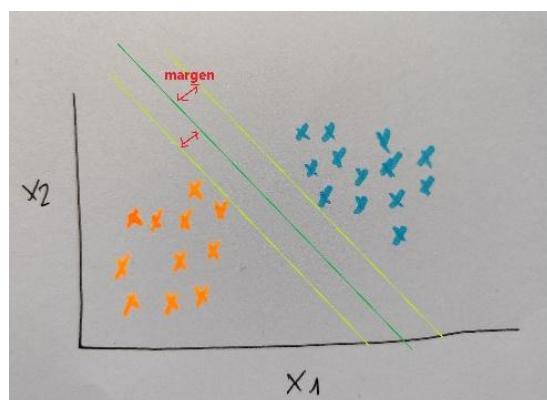
¿Qué es el margen?

El **margen** es la distancia que existe entre el límite de decisión y los ejemplos de entrenamiento más cercanos de cada clase. Estos ejemplos más cercanos reciben el nombre de **vectores soporte** (*support vectors*).

En SVM:

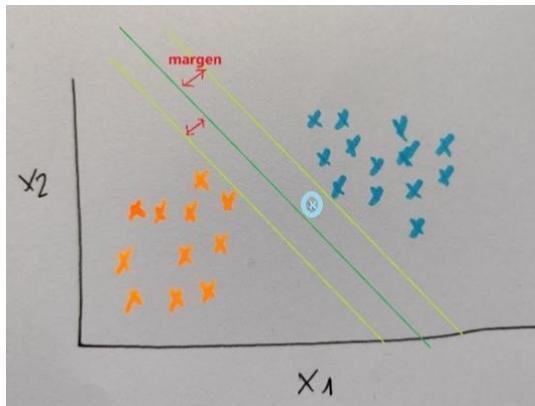
- El margen se mide desde el límite de decisión hasta esos vectores soporte.
- El algoritmo intenta **maximizar ese margen**.
- Cuanto mayor sea el margen, mejor será la capacidad de generalización del modelo.

Por eso se dice que SVM aplica **large margin classification**.



Ventaja frente a otros modelos

Gracias a este enfoque, si llega un **correo nuevo** (un nuevo ejemplo) y cae cerca del límite de decisión:



- El modelo SVM es capaz de clasificarlo correctamente.
- La decisión sigue la **tendencia global del conjunto de datos**.
- Se reduce la probabilidad de clasificar erróneamente ejemplos cercanos a la frontera.

Por ejemplo:

- A la **derecha** del límite → correo **spam**.
- A la **izquierda** del límite → correo **no spam**.

Esta separación es más robusta que la de otros modelos lineales que solo buscan separar las clases sin tener en cuenta la distancia a los datos.

Cómo construye SVM el límite de decisión

La manera en la que SVM construye el límite de decisión depende de **cómo estén distribuidos los datos** y de la **distancia (margen)** que se pueda mantener respecto a los ejemplos de entrenamiento.

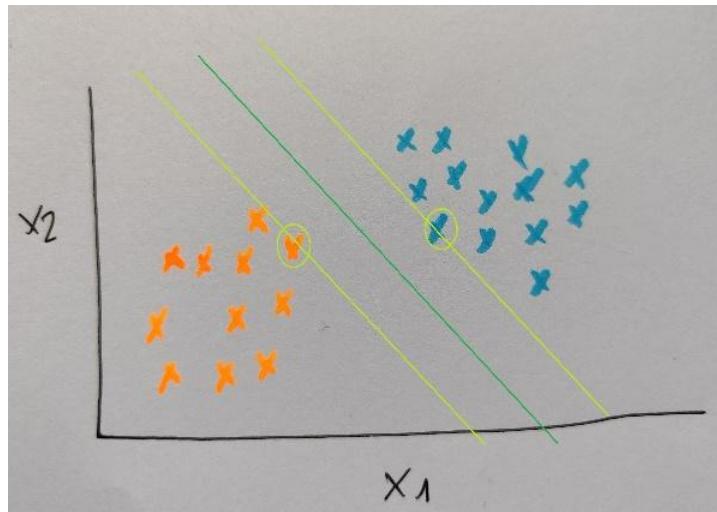
A partir de aquí, se pueden distinguir **dos formas principales** de construir ese límite de decisión:

- **Hard Margin Classification.**
- **Soft Margin Classification.**

Estos dos enfoques determinan cómo SVM gestiona la separación de las clases y la tolerancia a errores, y se estudiarán a continuación.

5. SVM Hard Margin Classification

Una vez que el algoritmo SVM ha encontrado los valores óptimos de los parámetros del modelo (θ_0 y θ_1), se construye el **límite de decisión** que separa las dos clases. Las **líneas paralelas al límite de decisión**, representadas habitualmente en color amarillo, indican el **margen** que el modelo mantiene respecto al conjunto de datos de entrenamiento.



Este margen representa la **distancia mínima** entre el límite de decisión y los ejemplos de entrenamiento más cercanos de cada clase como hemos indicado antes.

¿Qué es Hard Margin Classification?

Cuando hablamos de **Hard Margin Classification**, nos referimos a una configuración de SVM en la que:

- El algoritmo **no permite ningún error de clasificación** en el conjunto de entrenamiento.
- Todos los ejemplos deben quedar **perfectamente separados** por el límite de decisión.
- Ningún ejemplo puede quedar dentro del margen ni al lado incorrecto del límite.

En este enfoque, SVM construye el modelo basándose **únicamente en los ejemplos de entrenamiento más cercanos al límite de decisión**, ya que son los que determinan la posición exacta del margen.

Vectores soporte (Support Vectors)

Los ejemplos que se encuentran **exactamente sobre las líneas del margen** reciben el nombre de **vectores soporte (support vectors)**.

Características clave de los vectores soporte:

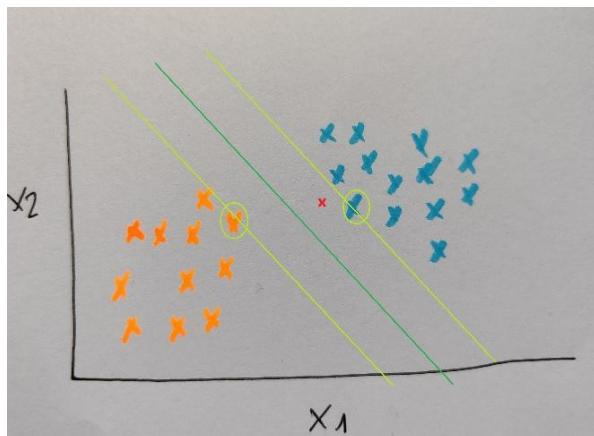
- Son los ejemplos **más cercanos al límite de decisión**.
- Determinan completamente la posición del modelo.
- El resto de ejemplos **no influyen en el modelo** mientras estén más alejados del margen.

Por este motivo, añadir nuevos ejemplos de entrenamiento **frente al margen** (es decir, con una distancia mayor al límite de decisión que los vectores soporte) **no modifica el modelo**.

Influencia de nuevos ejemplos cercanos al margen

La situación cambia cuando se añade un nuevo ejemplo que:

- Se encuentra **más cerca del límite de decisión** que los vectores soporte actuales.
- Tiene un margen menor.



En este caso:

- El conjunto de vectores soporte cambia.
- El algoritmo se ve obligado a **recalcular el límite de decisión**.
- Los márgenes se modifican (representados, por ejemplo, con líneas rojas).
- El modelo resultante es distinto al anterior.

Esto demuestra que el modelo SVM en hard margin depende **exclusivamente** de los ejemplos más cercanos al límite de decisión.

La distancia o márgenes nuevos (líneas rojas):



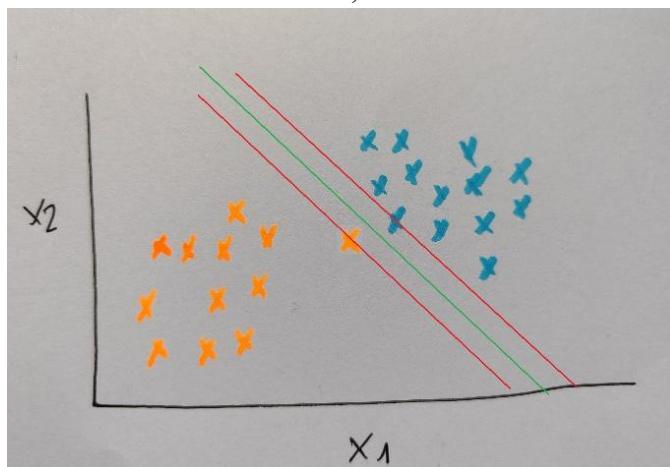
Restricción estricta del Hard Margin

El enfoque de **Hard Margin Classification** impone una condición muy fuerte:

- Todos los ejemplos de entrenamiento de cada clase deben quedar **completamente separados y fuera del margen**.
- No se tolera ningún tipo de ruido ni excepciones en los datos.

Esta rigidez provoca dos **problemas fundamentales**:

- Solo funciona con conjunto de **datos linealmente separables**: Si el conjunto de datos no puede separarse perfectamente mediante una línea recta, el hard margin **no es aplicable**, ya que no existe un límite de decisión que cumpla las restricciones del modelo.
- Es muy **sensible a datos anómalos (outliers)**. Por ejemplo:
 - Si aparece un correo que está **más cerca de los ejemplos de spam**, pero cae al lado contrario del límite de decisión, el modelo lo clasificará como **no spam**, simplemente por su posición respecto a la línea, aunque, siguiendo la distribución de los datos, debería clasificarse como spam.



Esto provoca que:

- El límite de decisión se ajuste incorrectamente.
- El modelo pierda capacidad de generalización.
- Se construya un **mal modelo predictivo**.

El enfoque **Hard Margin Classification**:

- Es conceptualmente sencillo.
- Funciona bien solo en situaciones ideales, sin ruido y con separación perfecta.
- No es adecuado para la mayoría de problemas reales.

Por estas razones, en la práctica se utiliza una variante más flexible denominada **Soft Margin Classification**, que permite ciertos errores y es mucho más robusta frente a datos reales.

6. SVM Soft Margin Classification

La **Soft Margin Classification** es una variante de SVM **menos rígida y más flexible** que la clasificación *Hard Margin Classification*. Su objetivo principal es **evitar los problemas derivados de datos ruidosos o anómalos**, muy habituales en situaciones reales.

En lugar de exigir una separación perfecta entre las clases, este enfoque permite **ciertos errores controlados** con el fin de construir un **modelo más robusto y con mejor capacidad de generalización**.

Objetivo del Soft Margin

El modelo Soft Margin busca mantener un **equilibrio adecuado** entre dos aspectos fundamentales:

- Mantener el **límite de decisión lo más alejado posible** de los ejemplos de entrenamiento (maximizar el margen).
- **Reducir la influencia de ejemplos anómalos (outliers)** que podrían distorsionar el modelo.

Gracias a este equilibrio, el límite de decisión resultante suele representar mejor la **tendencia global de los datos**.

El papel del hiperparámetro C

En la práctica, la flexibilidad del modelo SVM se controla mediante un **hiperparámetro** denominado **C**.

¿Qué es un hiperparámetro?

Los **hiperparámetros** son valores que el usuario configura **antes del entrenamiento** para modificar el comportamiento del modelo.

- No deben confundirse con los **parámetros del modelo** ($\theta_0, \theta_1, \dots$), que son aprendidos automáticamente durante el entrenamiento.
- Los hiperparámetros controlan cómo aprende el modelo y cómo se ajusta a los datos.

En el caso de SVM, **C es el hiperparámetro más importante**.

¿Qué controla el parámetro C?

El parámetro **C** controla el grado de penalización de los errores de clasificación y, por tanto, la **flexibilidad del límite de decisión**.

- **C alto:**
 - Penaliza fuertemente los errores.
 - El modelo intenta clasificar correctamente casi todos los ejemplos.
 - El límite de decisión se ajusta mucho a los datos.
 - Mayor riesgo de sobreajuste.
- **C bajo:**
 - Penaliza menos los errores.
 - Se permite que algunos ejemplos queden mal clasificados.
 - El modelo es más flexible.
 - Mayor robustez frente a outliers.

Soft Margin frente a Hard Margin

No existen dos algoritmos distintos (uno para hard margin y otro para soft margin). En realidad, **el mismo algoritmo SVM puede comportarse como hard margin o soft margin** dependiendo del valor de **C**:

- Valores muy altos de **C** → comportamiento similar a *Hard Margin*.
- Valores más bajos de **C** → comportamiento de *Soft Margin*.

De este modo, el usuario puede **regular el comportamiento del modelo** según la naturaleza de los datos.

Influencia de los datos anómalos

Gracias al parámetro **C**, el modelo puede:

- **Reducir el peso de ejemplos anómalos.**
- Evitar que un único outlier determine el límite de decisión.
- Incorporar más vectores soporte, no solo el ejemplo más extremo.
- Construir el límite de decisión en una zona que represente mejor a la mayoría de los datos.

Esto permite que el límite de decisión se sitúe **donde se concentra la mayor parte de los ejemplos**, en lugar de adaptarse a casos excepcionales.

¿Por qué no se puede elegir C “a ojo”?

El valor óptimo de **C**:

- Depende del dataset.
- Depende del nivel de ruido.
- Depende del tipo de kernel utilizado.
- No se puede conocer antes de entrenar.

Por ello, es necesario usar **técnicas sistemáticas de validación**, que permitan evaluar el rendimiento del modelo con distintos valores de **C**.

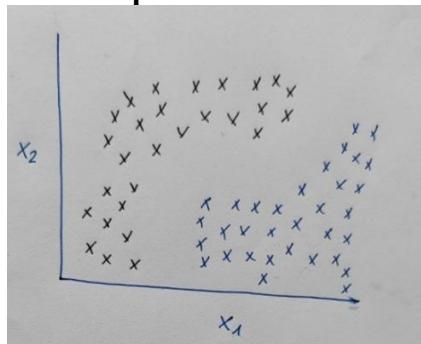
La **Soft Margin Classification** es el enfoque utilizado en la mayoría de aplicaciones reales de SVM porque:

- Es más flexible.
- Tolera ruido y datos imperfectos.
- Generaliza mejor que el hard margin.
- Permite ajustar el comportamiento del modelo mediante el parámetro **C**.

Por estas razones, el uso de **Soft Margin SVM** es la opción recomendada en problemas reales de clasificación.

7. SVM – Kernels y separación no lineal

Los **clasificadores lineales** (como la regresión logística o SVM lineal) son eficientes y funcionan muy bien en muchos problemas. Sin embargo, existen numerosos conjuntos de datos en los que **no es posible separar las clases mediante una línea recta**, es decir, conjuntos de datos **no linealmente separables**.



En estos casos, es necesario construir **límites de decisión no lineales** que se adapten mejor a la distribución real de los datos.

Límites de decisión no lineales

Cuando los datos no pueden separarse linealmente, el objetivo es transformar el problema original en otro donde sí sea posible una separación lineal. Una idea intuitiva consiste en:

- Crear **nuevas características** a partir de las originales.
- Aumentar la dimensión del espacio de características.
- Permitir que un clasificador lineal actúe como si fuera no lineal en el espacio original.

Regresión polinómica como primera aproximación.

Una de las técnicas más sencillas para generar límites de decisión no lineales es la **regresión polinómica**.

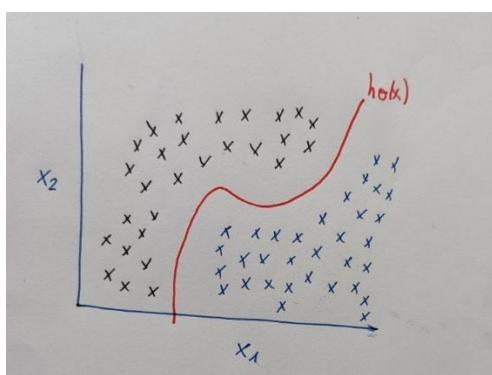
¿En qué consiste?

La regresión polinómica se basa en **crear nuevas características polinómicas** a partir de las características originales.

Ejemplo:

- Características originales: x_1, x_2
- Características polinómicas:
 - x_1^2
 - x_2^2
 - $x_1 \cdot x_2$
 - x_1^3 , etc.

Con estas nuevas características, un modelo lineal puede generar **fronteras de decisión curvas** en el espacio original.



Problemas de la regresión polinómica explícita.

Aunque esta técnica permite crear límites de decisión no lineales, presenta **importantes inconvenientes**.

- **Polinomios de bajo grado:**
 - Generan pocas características nuevas.
 - Los límites de decisión son demasiado simples.

- No permiten capturar patrones complejos.
- El modelo sigue teniendo un alto error.
- **Polinomios de alto grado:**
 - Generan un **gran número de características**.
 - Aumentan drásticamente la dimensionalidad.
 - Requieren **muchas capacidades de cómputo**.
 - El entrenamiento se vuelve lento.
 - Incrementan el riesgo de sobreajuste (*overfitting*).

En la práctica, crear explícitamente todas las características polinómicas **no es eficiente ni escalable**.

La solución de SVM: el truco del kernel

Para resolver estos problemas, SVM introduce una solución elegante y eficiente conocida como el **kernel trick** (*truco del kernel*). La idea fundamental es que en lugar de transformar explícitamente los datos a un espacio de mayor dimensión, SVM:

- Calcula directamente el producto escalar entre los datos en ese espacio de alta dimensión,
- Sin necesidad de calcular ni almacenar las nuevas características.

Esto permite trabajar con **modelos no lineales** manteniendo una complejidad computacional razonable.

¿Qué es un kernel?

Un **kernel** es una función matemática que mide la similitud entre dos ejemplos, como si estos hubieran sido transformados a un espacio de características de mayor dimensión.

Kernel polinómico

El **kernel polinómico** permite a SVM generar límites de decisión polinómicos sin crear explícitamente las características.

Ventajas

- Permite modelar relaciones no lineales.
- Evita la explosión de características.
- Mucho más eficiente que la regresión polinómica explícita.

Inconvenientes

- Elección del grado d puede ser complicada.
- Puede sobreajustar si el grado es alto.

Ejemplo:

Tienes estos puntos en un plano (x, y):

Clase A (●):

- A1 = (1, 0)
- A2 = (0, 1)

Clase B (○):

- B1 = (2, 2)
- B2 = (3, 1)

1. Representa mentalmente (o en un eje) los puntos.
2. ¿Crees que una recta los separa fácilmente en el plano original?
3. Ahora usamos una transformación polinómica sencilla:

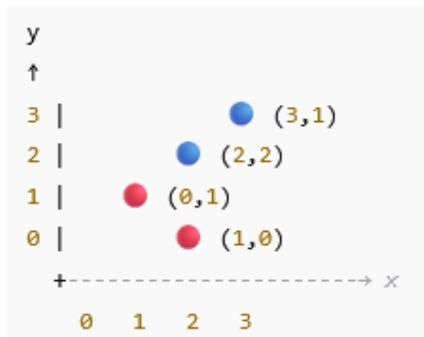
$$\phi(x, y) = (x^2, y^2)$$

En el nuevo espacio (x^2, y^2), intenta proponer una separación simple usando una regla del tipo:

$$x^2 + y^2 > T$$

Encuentra un valor T que separe A y B.

Como vemos los punto de A están cerca del origen (valores pequeños) y los de B están más lejos (valores grandes):



En el plano original puede ser difícil justificar una recta “ limpia”, porque depende de cómo caiga la recta y hay muchos casos donde los datos reales se mezclan más.

Aquí lo importante es lo que pasa al “potenciar”.

$$\text{Transformación } \phi(x, y) = (x^2, y^2)$$

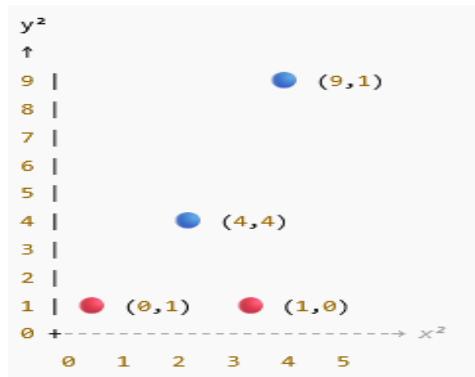
Calculamos:

Clase A (●):

- A1 = (1, 0) $\rightarrow (1^2, 0^2) = (1, 0)$
- A2 = (0, 1) $\rightarrow (0^2, 1^2) = (0, 1)$

Clase B (○):

- B1 = (2, 2) $\rightarrow (4, 4)$
- B2 = (3, 1) $\rightarrow (9, 1)$



4) Buscamos un umbral T con $x^2 + y^2 > T$

Calculamos $x^2 + y^2$ en el nuevo espacio (equivale a “distancia al origen”, pero sin raíz):

Clase A:

- A1: $(1,0) \rightarrow 1+0 = 1$
- A2: $(0,1) \rightarrow 0+1 = 1$

Clase B:

- B1: $(4,4) \rightarrow 4+4 = 8$
- B2: $(9,1) \rightarrow 9+1 = 10$

Ahora elige un T entre 1 y 8, por ejemplo:

Si $T = 4$

- A: 1 no es mayor que 4 \rightarrow se queda en “Clase A”
- B: 8 y 10 sí son mayores que 4 \rightarrow se van a “Clase B”

Regla final que separa perfecto:

$$x^2 + y^2 > 4 \Rightarrow \text{Clase B, si no, Clase A}$$

Nosotros hemos calculado x^2 e y^2 a mano para verlo claro. SVM con kernel polinómico consigue el mismo efecto sin crear esas columnas: calcula directamente la similitud como si existieran.

El kernel polinómico hace que SVM vea estos puntos como si estuvieran en otro espacio donde una recta ya sirve, aunque nosotros no lo veamos aquí.

Ejercicio: Busca ejemplos de Kernel polinómicos dónde se vea que crea estos límites sin problemas. Puedes ayudarte de la IA.