

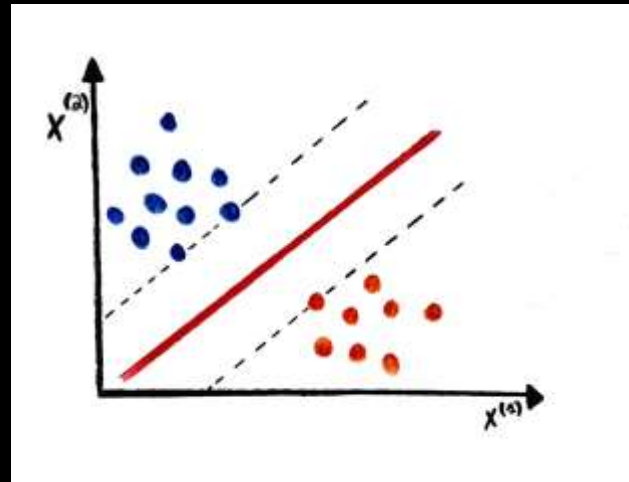
# Machine Learning – Support Vector Machines

# Definición

Algoritmo supervisado de Machine Learning capaz de resolver problemas lineales, no lineales, de clasificación y regresión.

Es uno de los modelos más utilizados debido a su versatilidad y precisión.

Son algoritmos sensibles al escalado, por lo que se suelen estandarizar antes del entrenamiento.



# Linear Support Vector Classifier

# Hiperplano

Es un subespacio plano y afín (no tiene por qué pasar por el origen), de dimensión  $p-1$ , que divide el espacio en dos mitades

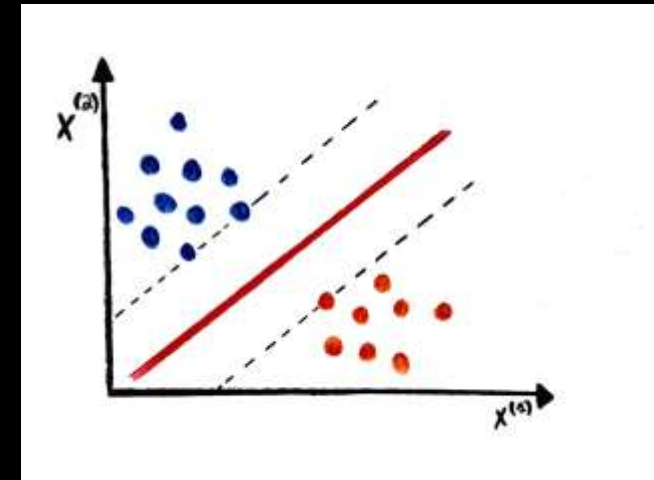
Todos los puntos que caigan en el hiperplano cumplirán:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p = 0$$

O caerán a un lado u otro del hiperplano:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p > 0$$

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p < 0$$



# Hiperplano óptimo

Para un problema de clasificación binaria con buena separación de las clases, existen infinitos hiperplanos que los separan, pero solo un hiperplano óptimo que maximiza la distancia entre las observaciones y el hiperplano.

## Hiperplano óptimo

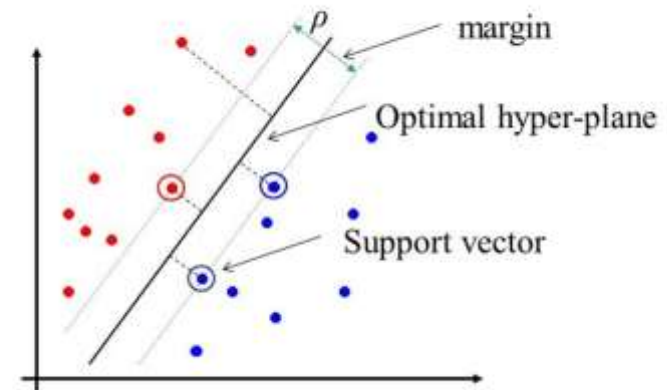
Aquel que tenga la mayor distancia mínima de las observaciones al hiperplano, el mayor **margen**.

## Support Vector

Son los vectores que aguantan (*soportan*) al hiperplano óptimo de separación. Si estos cambian, el hiperplano también lo hará. El movimiento del resto de observaciones no tiene tanto impacto.

## SVM: separable classes

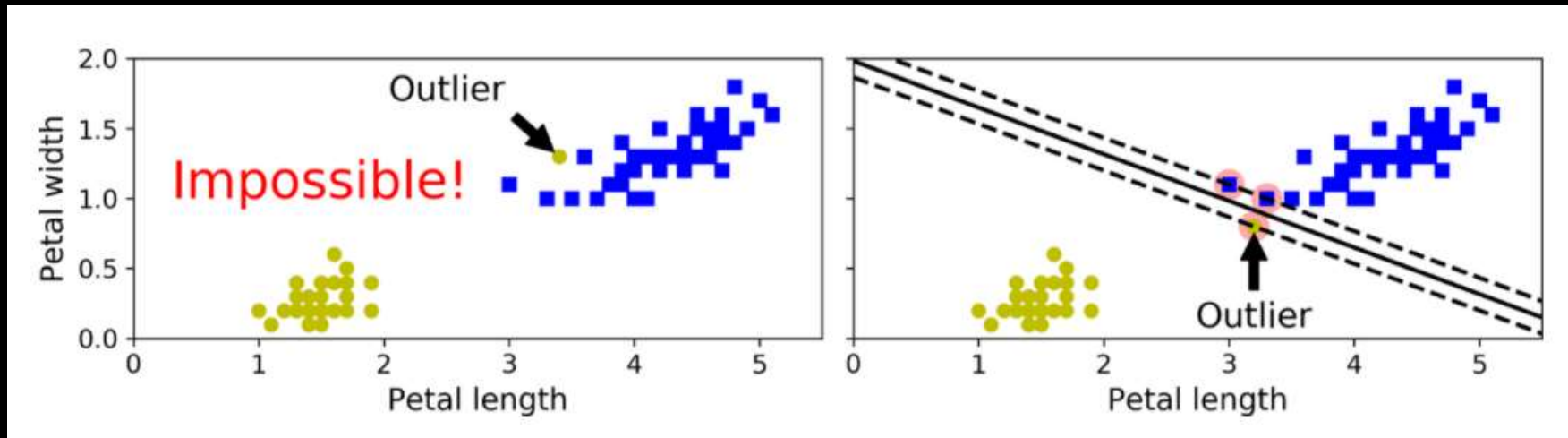
Support vectors uniquely characterize optimal hyper-plane



# Hard margin classifier

Son los clasificadores que utilizan un hiperplano óptimo. El problema es:

- Son sensibles a overfitting
- Funcionan si los datos están linealmente separados
- Muy sensible a outliers

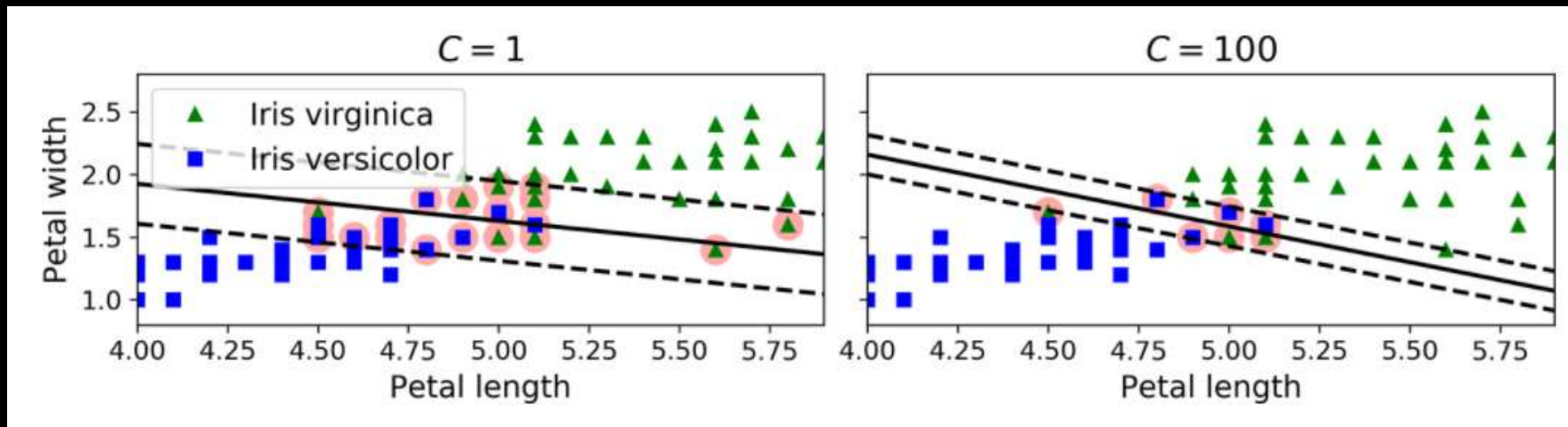


# Soft margin classifier

Hiperplano que no separa perfectamente las clases, permitiendo que algunas observaciones caigan del lado incorrecto.

Ahora los **support vectors** serán tanto los que se encuentren en el margen como los que violen el margen.

Esta flexibilidad se controla mediante  $C$ . Cuanto menor es  $C$ , más es la flexibilidad/tolerante es mi modelo, y por tanto generalizará mejor. Bias vs Variance.



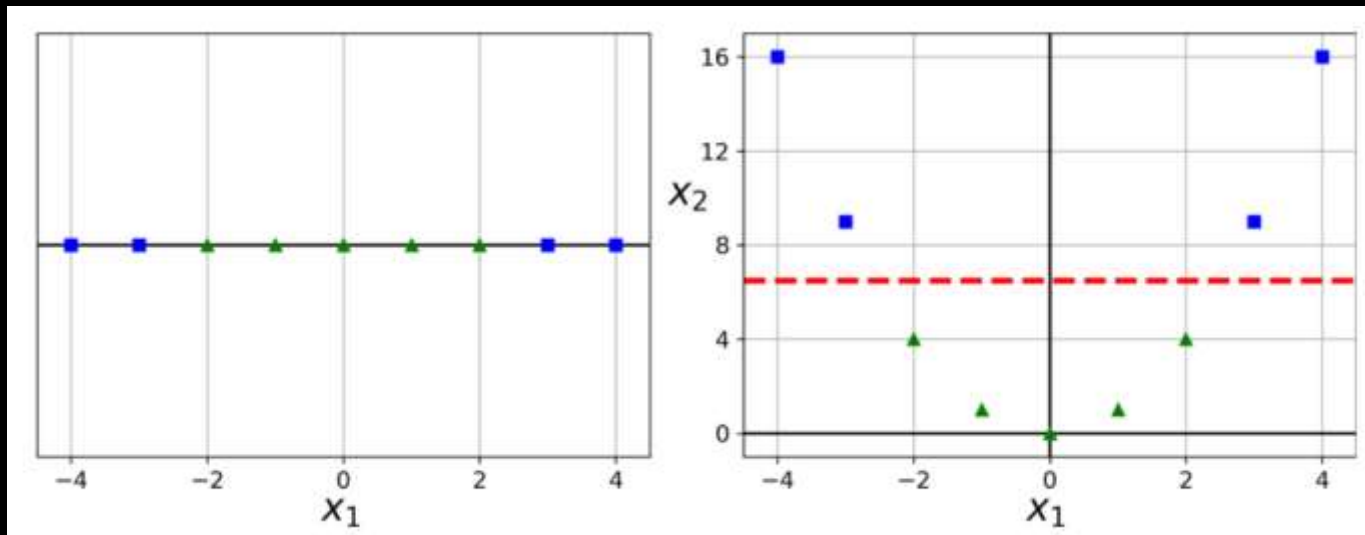
# Nonlinear Support Vector Classifier



# Nonlinear SVC

Los SVC lineales funcionan muy bien, pero para problemas lineales, que en la vida real son problemas poco frecuentes.

Una posible solución es introducir nuevas features polinómicas (feature expansion). El problema es que las regresiones polinómicas no se adaptan bien a datasets complejos, y le añaden demasiada **compejidad al modelo, provocando que sea muy lento y podamos caer en overfitting.**

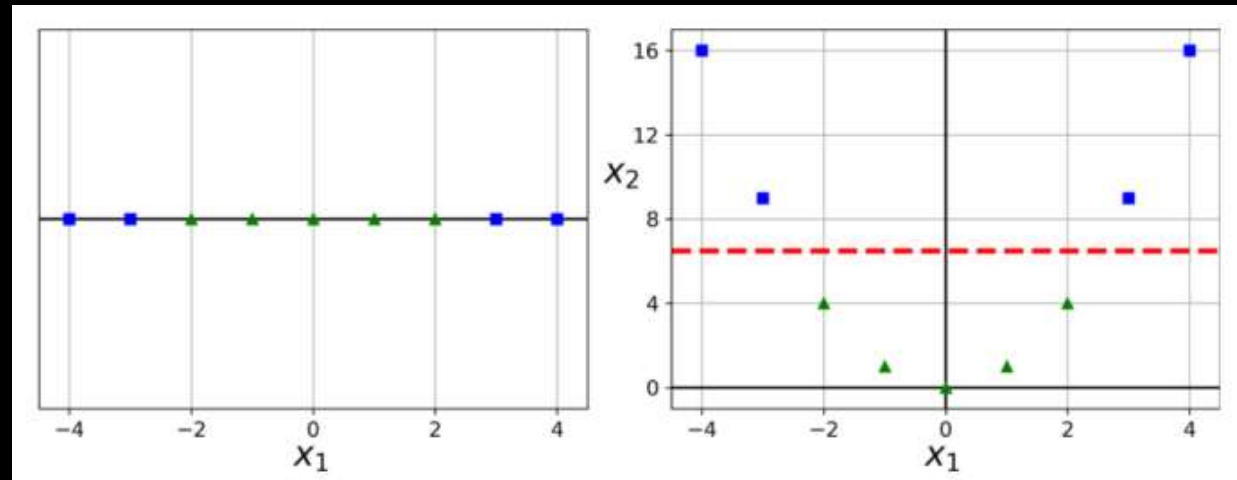


# kernels

Para el caso del kernel polinómico, podemos obtener el mismo resultado que añadiendo features polinómicas, pero realmente sin que las estemos añadiendo. Esta simulación es lo que se conoce como **kernel trick**

## Kernels

Linear:	$K(\mathbf{a}, \mathbf{b}) = \mathbf{a}^T \mathbf{b}$
Polynomial:	$K(\mathbf{a}, \mathbf{b}) = (\gamma \mathbf{a}^T \mathbf{b} + r)^d$
Gaussian RBF:	$K(\mathbf{a}, \mathbf{b}) = \exp(-\gamma \ \mathbf{a} - \mathbf{b}\ ^2)$
Sigmoid:	$K(\mathbf{a}, \mathbf{b}) = \tanh(\gamma \mathbf{a}^T \mathbf{b} + r)$



¿Cuándo usar un kernel u otro? Probar siempre primero con el lineal (LinearSVC más rápido que SVC(linear)). Si el set de datos no es muy grande, prueba con Gaussian RBF y el Polynomial que suelen dar buenos resultados.

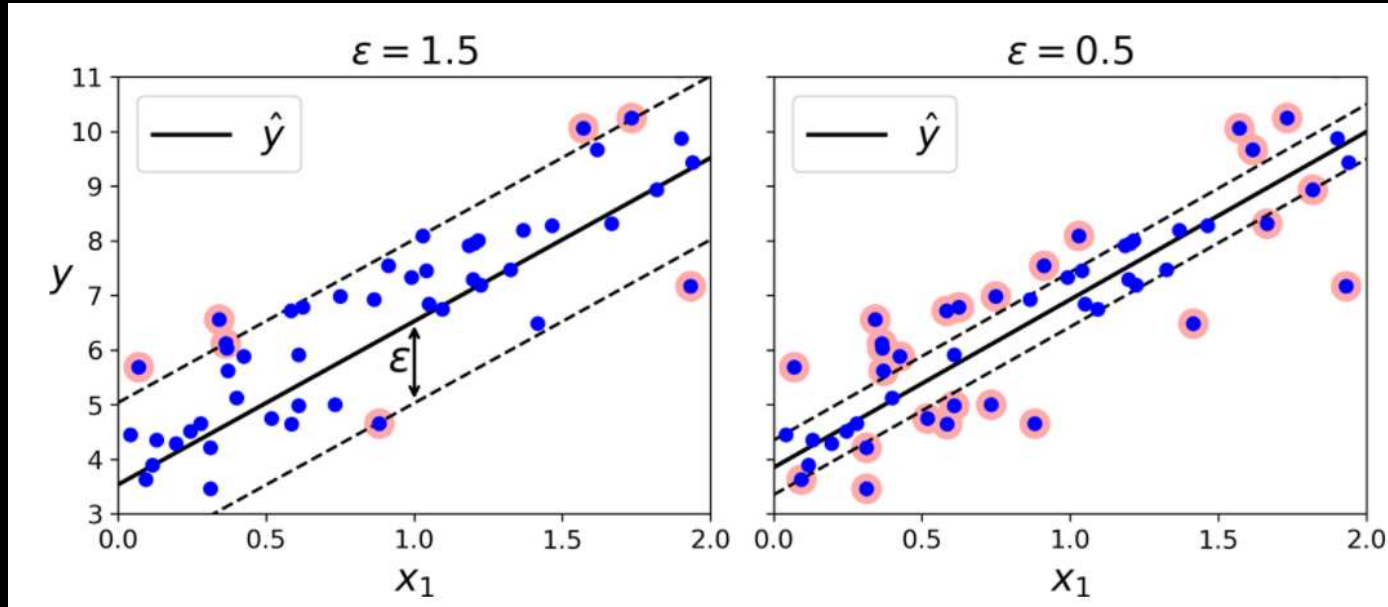
# Support Vector Regression

# Support Vector Regression

Podemos utilizar SVM para algoritmos de regresión. A diferencia de clasificación, el objetivo es el inverso, SVM Regression intenta ajustarse al máximo para que abarque la mayor cantidad de muestras.

La anchura de los márgenes la controlaremos con el hiperparámetro  $\varepsilon$ , que es la tolerancia.

También podemos usar kernels para problemas no lineales.



# Ventajas

1. Efectivo con muchas features
2. Eficaz computacionalmente
3. Varios kernels para varios problemas
4. Clasificación/Regresión
5. Robustos frente overfitting
6. Robustos frente outliers

# Desventajas

1. Mal performance si  $n^{\circ}$  features  $>$   $n^{\circ}$  samples
2. Hay que realizar mucha combinatoria de kernels e hiperparámetros para conseguir el modelo óptimo
3. Es caja negra