SUPPORT VECTOR MACHINES

Santiago Alférez

APRENDIZAJE AUTOMÁTICO DE MÁQUINA

MACC – UR – EICT

Contenido

- Hiperplano
- Clasificación usando un hiperplano
- El clasificador de margen máximo
- El clasificador mediante vectores de soporte
- Ampliación de descriptores (polinomial)
- El truco del kernel
- Máquinas de vectores de soporte
- Clasificación con más de dos clases
- Implementación
- Máquinas de vectores de soporte para regresión

Máquinas de vectores de soporte

El problema de clasificación binaria se aproxima directamente:

Al encontrar un plano **que separe** las clases en el espacio de descriptores.

Si no es posible, se puede hacer de dos formas:

- Suavizando lo que quiere decir “ que separe”, y
- Enriqueciendo y aumentando el espacio de descriptores tal que la separación sea posible.

¿Qué es un hiperplano?

- Un hiperplano en *p* dimensiones es un subespacio plano de dimensión *p - 1*.

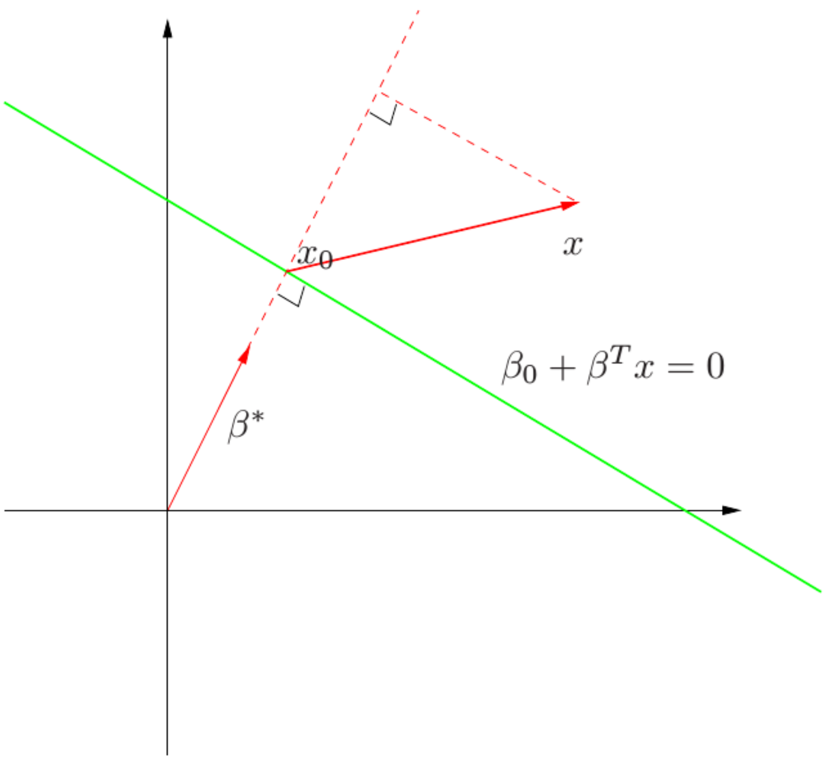
- En general, la ecuación para un hiperplano tiene la forma:

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

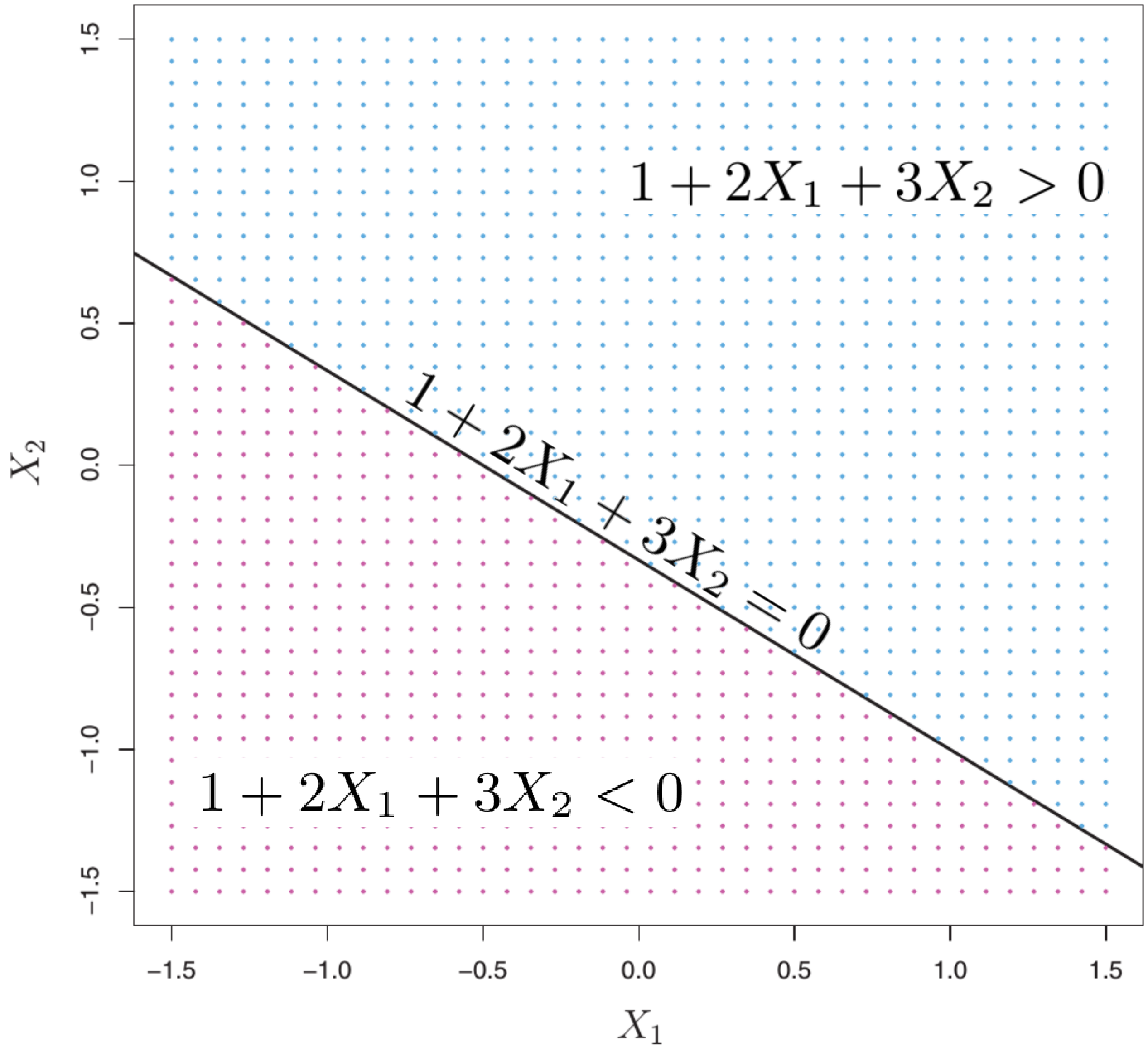
- En $p = 2$ dimensiones, un hiperplano es una línea.

- Si $\beta_0 = 0$, el hiperplano pasa por el origen, de lo contrario no.

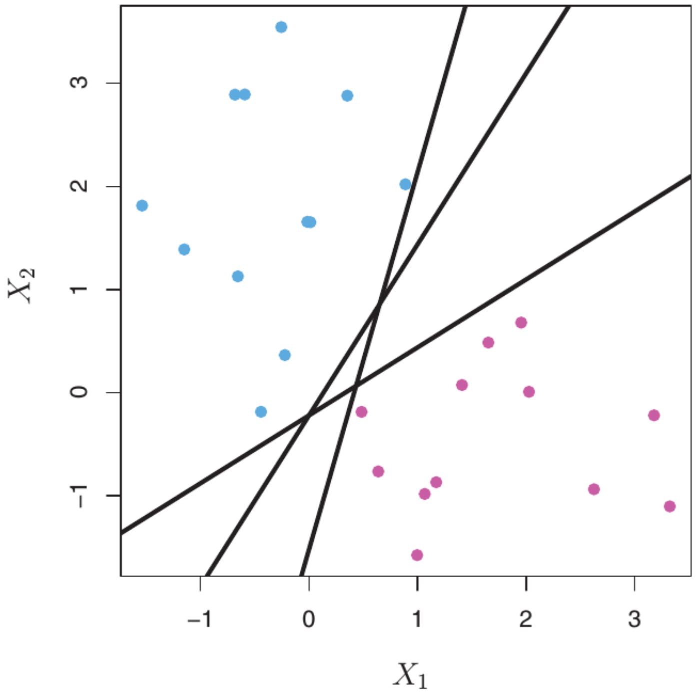
- El vector $\beta = (\beta_1, \beta_2, \cdots, \beta_p)$ se denomina vector normal, apunta en una dirección ortogonal a la superficie del hiperplano.

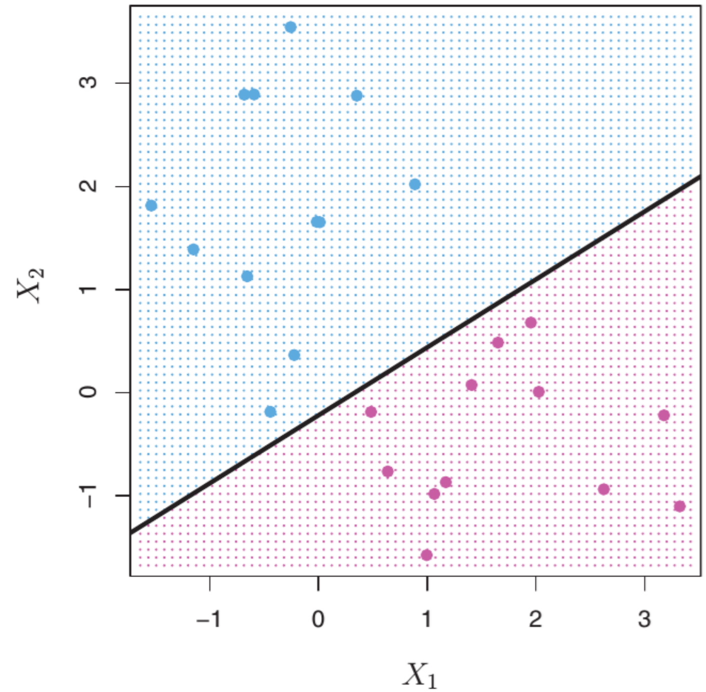


Ejemplo de un hiperplano



Clasificación usando un hiperplano separador





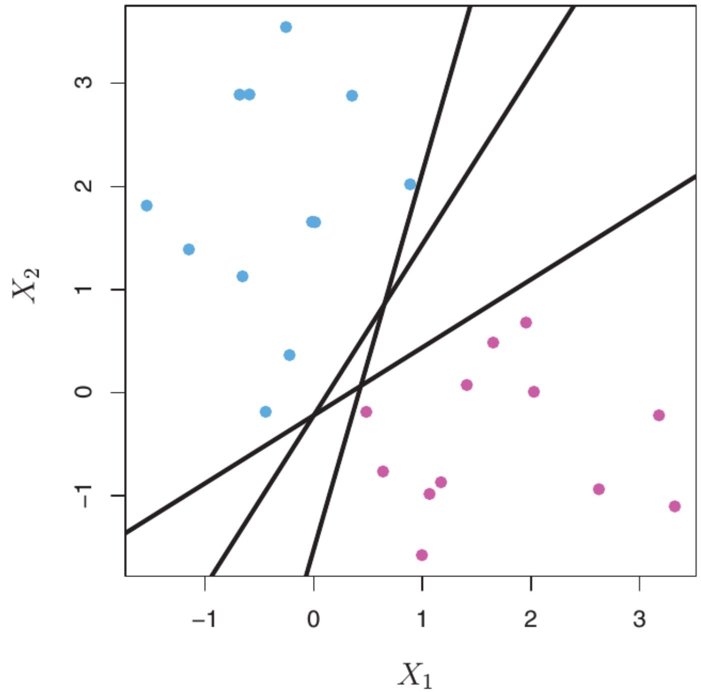
$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip} > 0 \text{ if } y_i = 1$$

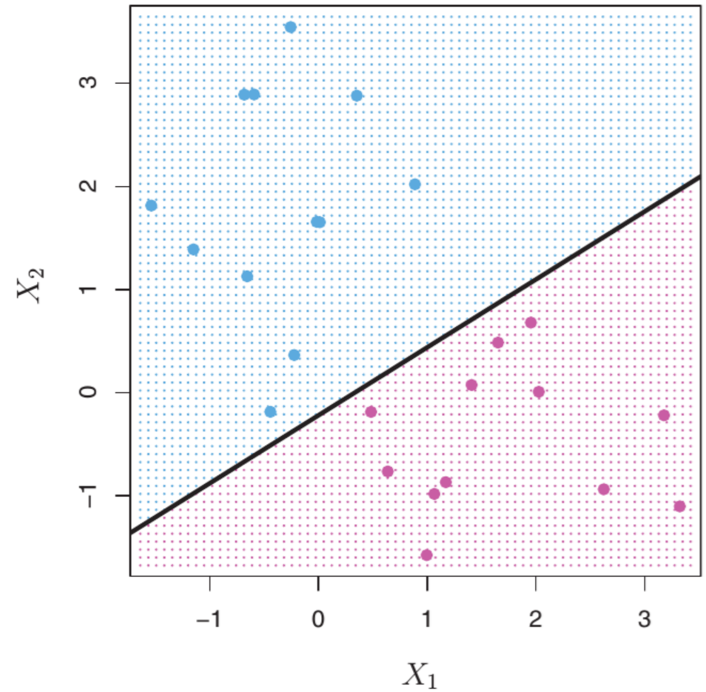
$$\mathbf{x}^T \boldsymbol{\beta} + \beta_0$$

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip} < 0 \text{ if } y_i = -1$$

$$\boxed{y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) > 0}$$

Clasificación usando un hiperplano separador



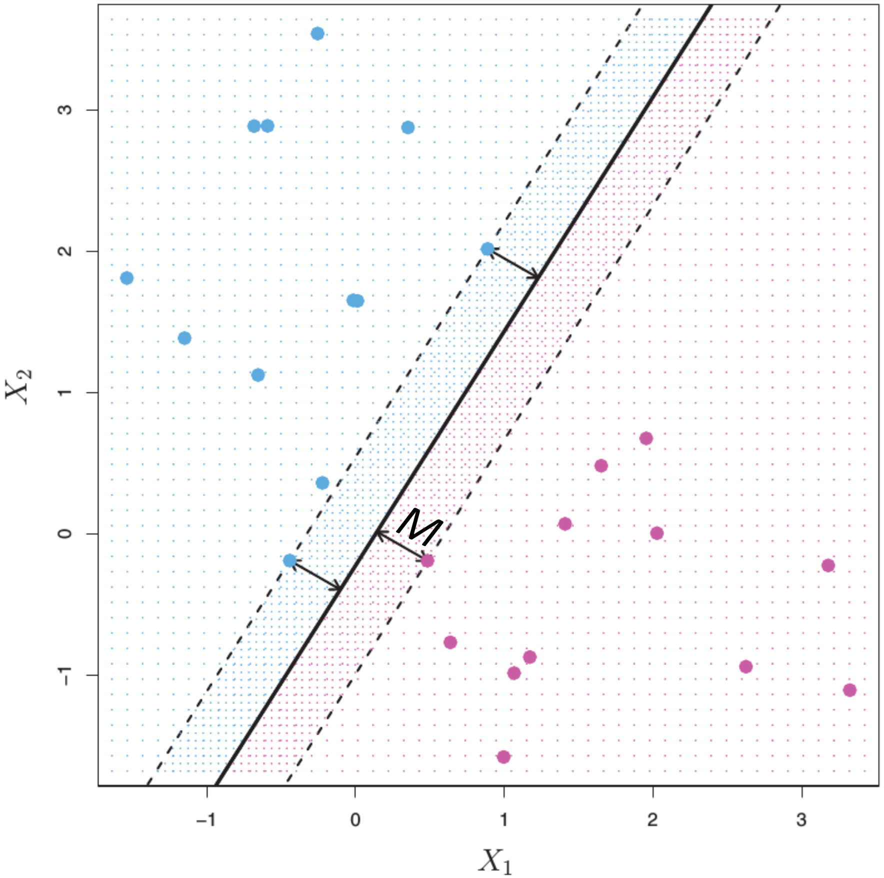


Predicción

$$x^* \Rightarrow f(x^*) = \beta_0 + \beta_1 x_1^* + \beta_2 x_2^* + \ldots + \beta_p x_p^* \Rightarrow \begin{cases} f(x) > 0 \Rightarrow x \in clase\,1 \\ f(x) < 0 \Rightarrow x \in clase\,-1 \end{cases}$$

El clasificador de margen máximo

El **margen** es la “calle” máxima que se puede colocar entre dos observaciones de diferentes clases.



El hiperplano con máximo margen es aquel que soluciona:

$$\boxed{\max_{\beta_0, \beta_1, \ldots, \beta_p} M}$$

$$\text{sujeto a}: \begin{cases} \sum_{j=1}^{n} \beta_j^2 = 1 \\ y_i\left(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{p1}\right) \geq M \end{cases}$$

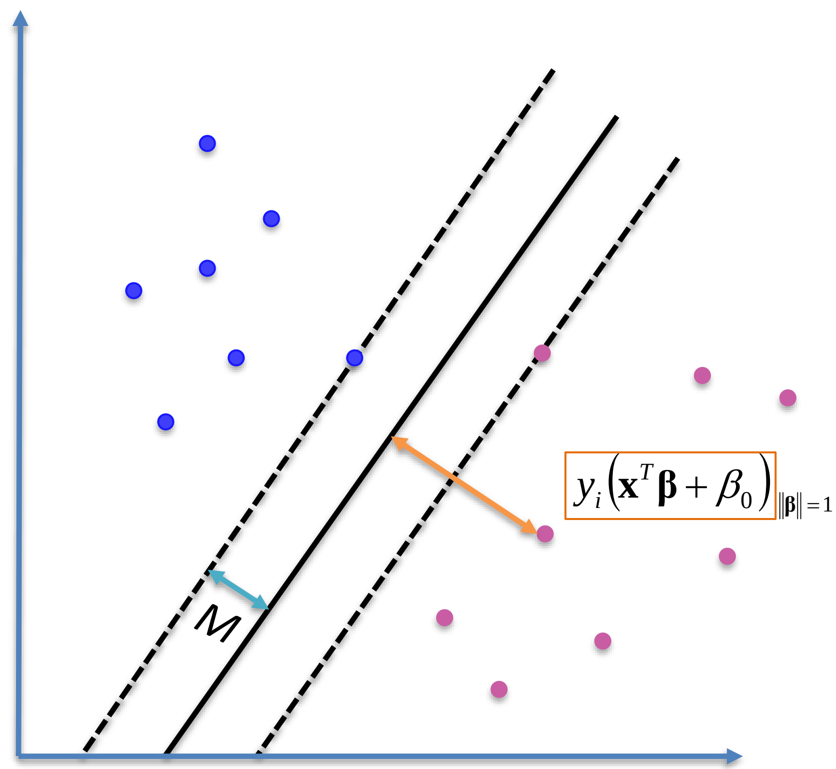
$$\forall\, i = 1, \ldots, N$$

$$\max_{\beta_0, \beta, \|\boldsymbol{\beta}\|=1} M$$

$$\text{sujeto a}: \; y_i\left(\mathbf{x}^T \boldsymbol{\beta} + \beta_0\right) \geq M$$

El clasificador de margen máximo

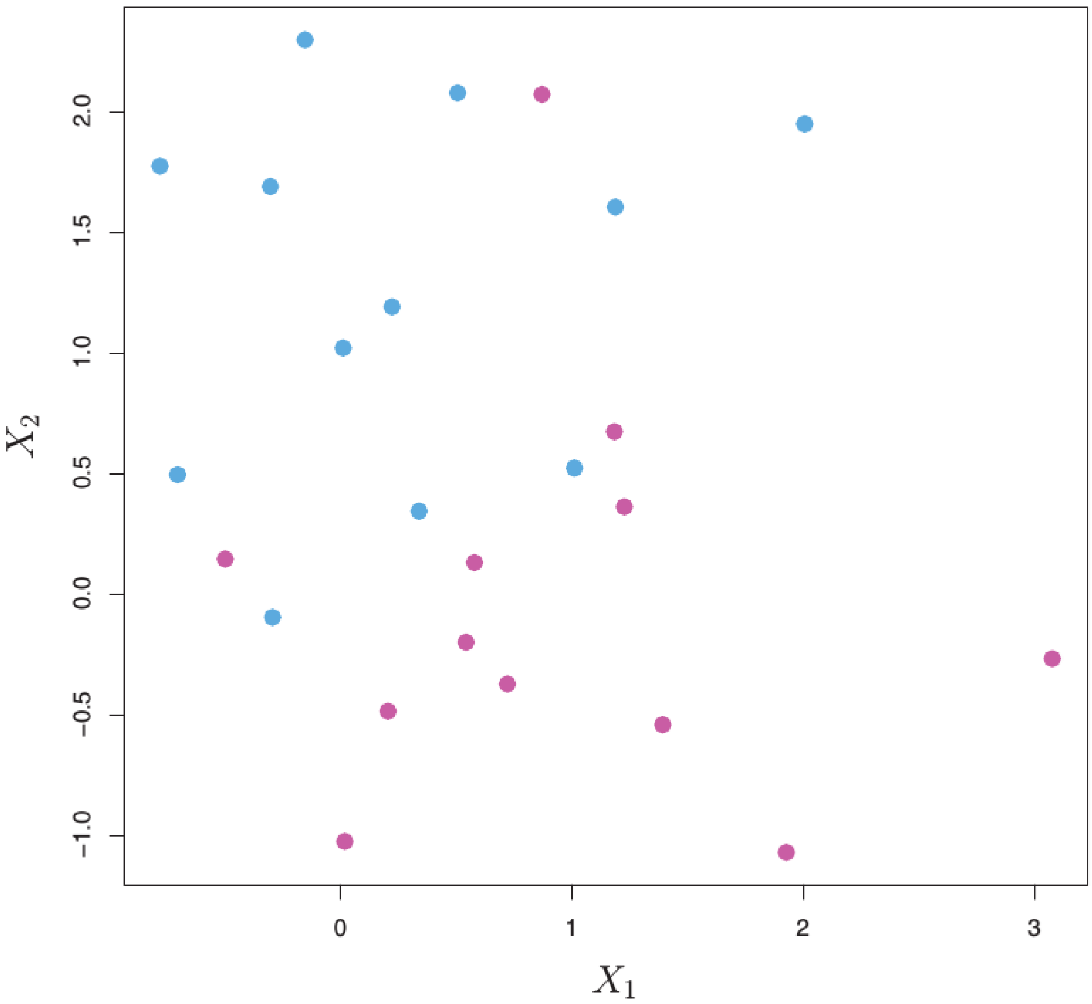
El **margen** es la “calle” máxima que se puede colocar entre dos observaciones de diferentes clases.



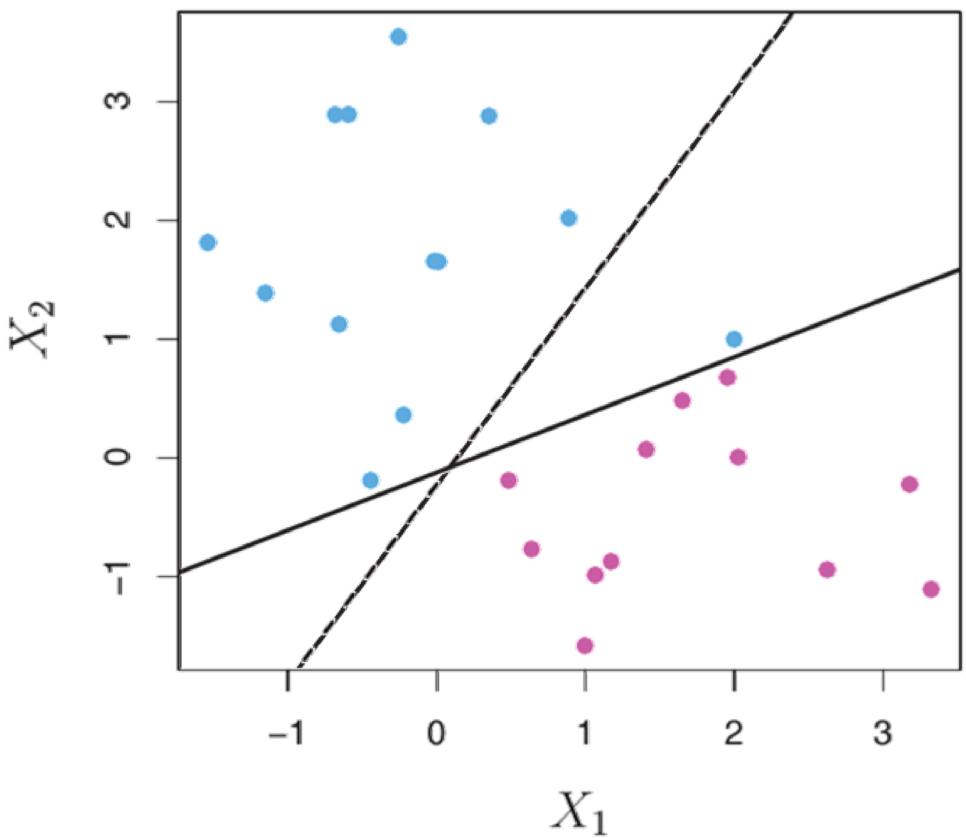
El hiperplano con máximo margen es aquel que soluciona:

$$\boxed{\begin{gathered}\max_{\beta_0,\beta,\|\boldsymbol{\beta}\|=1} M \\ \text{sujeto a}: \ y_i\left(\mathbf{x}^T\boldsymbol{\beta}+\beta_0\right)\geq M\end{gathered}}$$

Datos (linealmente) no separables

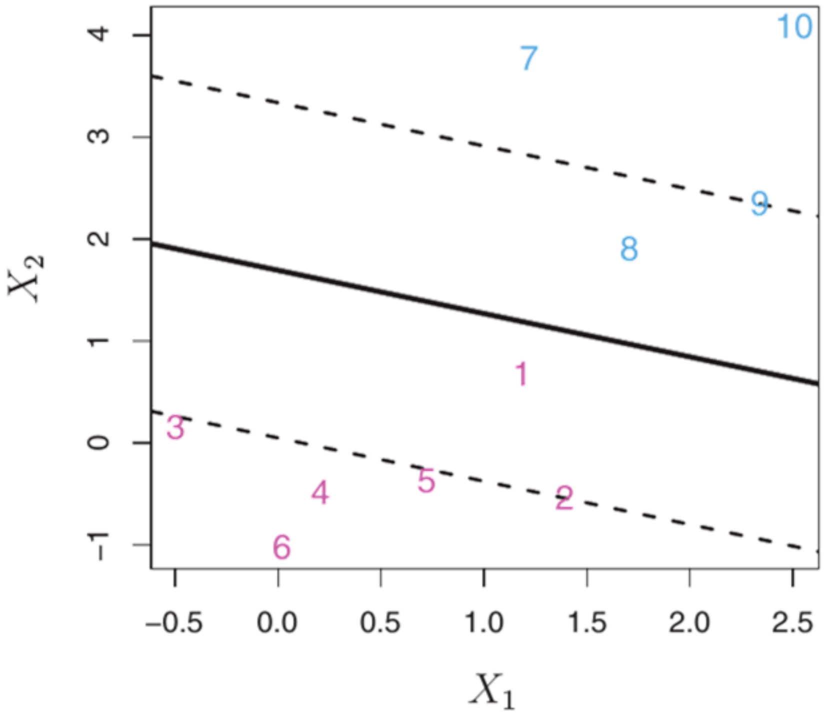


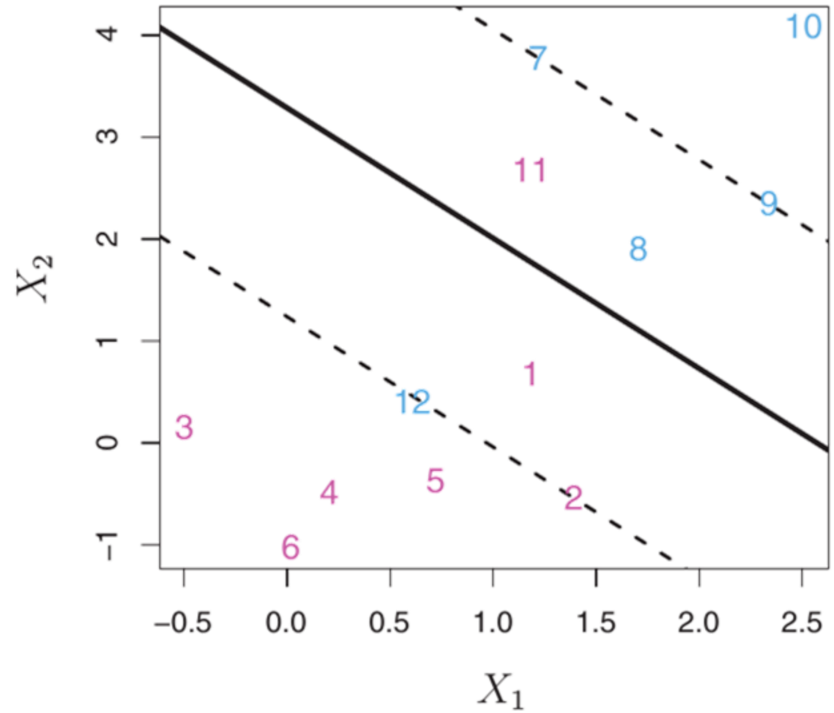
Nuevos datos en el clasificador de margen máximo



- El clasificador de margen máximo es sensible a nuevos datos.
- El margen máximo es muy delgado, es decir, la clasificación es poco confiable.
- Es deseable un clasificador que no separe las dos clases perfectamente con el fin de:
  - ✓ Tener una gran robustez a datos individuales
  - ✓ Mejor clasificación de la *mayoría* (no la totalidad) de los datos del training.

Clasificador mediante vectores de soporte

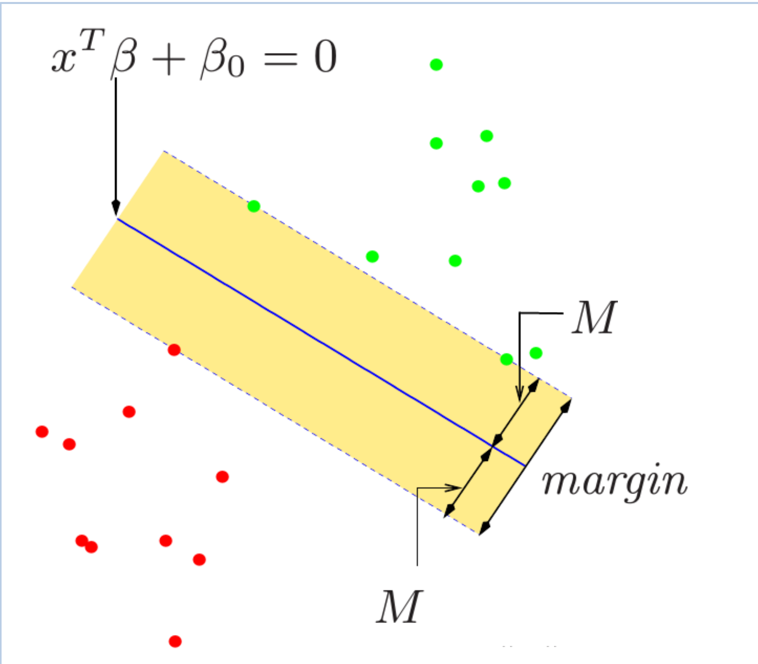


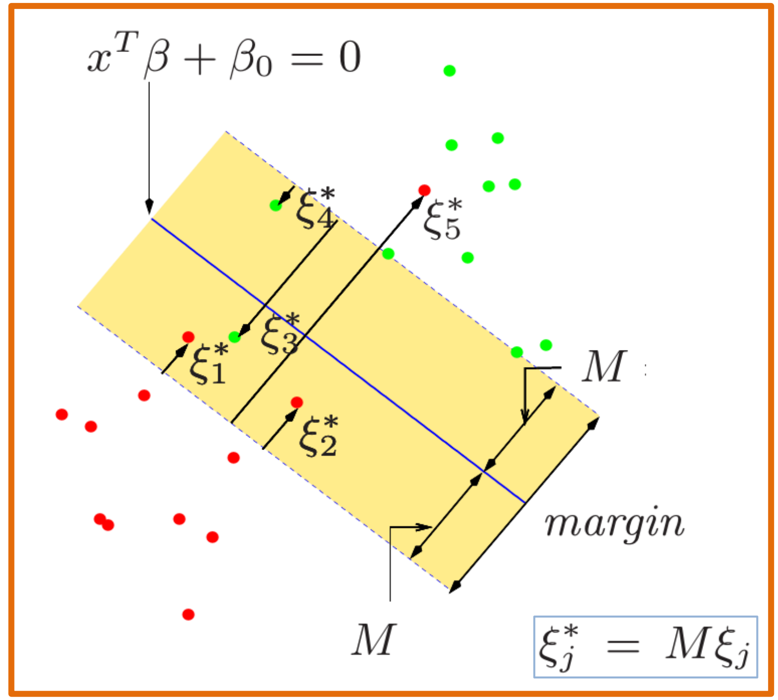


- ✓ **3, 4, 5** y **6** están en el lado correcto del margen.
- ✓ **2** está en el margen y **1** está en el lado incorrecto del margen.
- ✓ **7** y **10** están en el lado correcto del margen.
- ✓ **9** está en el margen.
- ✓ **8** está en el lado incorrecto del margen.
- ✓ No hay observaciones en el lado incorrecto del hiperplano.

- ✓ Hay dos puntos adicionales, **11** y **12**.
- ✓ Estos dos datos en el lado correcto del hiperplano y en el lado incorrecto del margen.

Detalles del clasificador mediante vectores de soporte

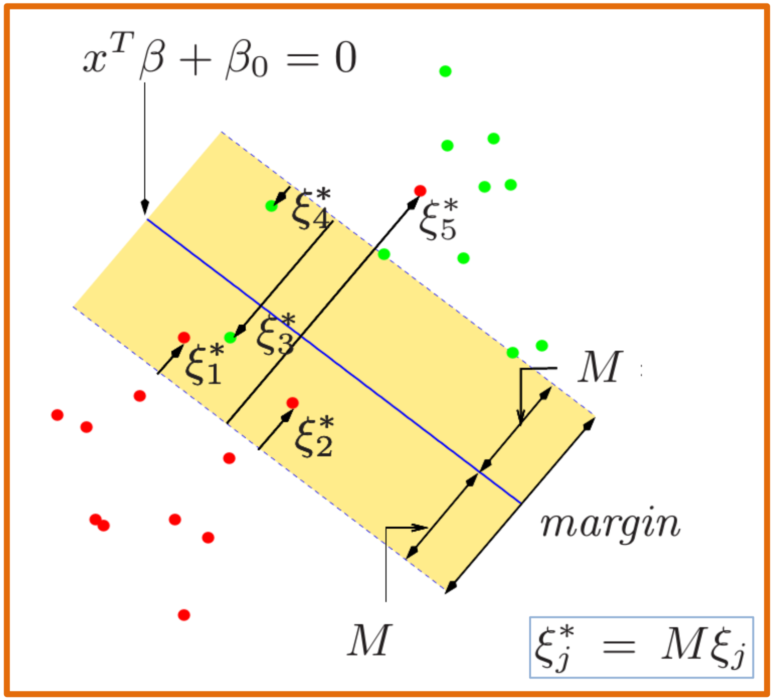




$$\max_{\beta_0,\beta,\xi,\|\boldsymbol{\beta}\|=1} M$$

$$\text{sujeto a}: \begin{cases} y_i\left(\mathbf{x}^T\boldsymbol{\beta} + \beta_0\right) \geq M\left(1-\xi_i\right) \\ \xi_i \geq 0,\ \sum_{i=1}^{n} \xi_i \leq constante = \dfrac{1}{C'} \end{cases}$$

Detalles del clasificador mediante vectores de soporte



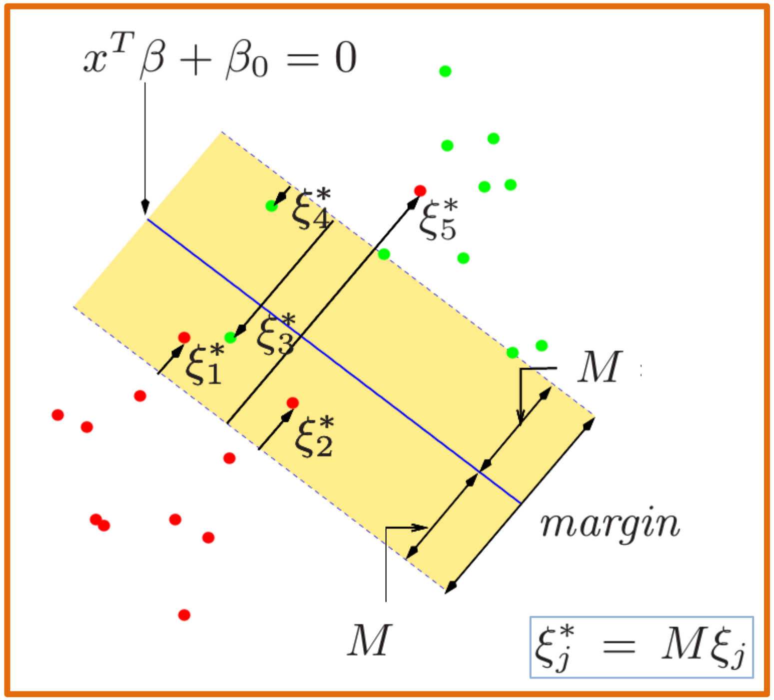
$$\max_{\beta_0,\beta,\xi,\|\boldsymbol{\beta}\|=1} M$$

$$\text{sujeto a}: \begin{cases} y_i\left(\mathbf{x}^T\boldsymbol{\beta}+\beta_0\right) \geq M\left(1-\xi_i\right) \\ \xi_i \geq 0,\ \sum_{i=1}^{n}\xi_i \leq presupuesto = \dfrac{1}{C'} \end{cases}$$

- Si $\xi_i = 0$ entonces el dato *i* esta sobre el lado correcto del margen.
- Si $\xi_i > 0$ entonces el dato *i* esta sobre el lado incorrecto del margen.
- Si $\xi_i > 1$ entonces el dato *i* esta sobre el lado incorrecto del hiperplano.

- Si $C' \to \infty$ (*presupuesto = 0*), no pueden haber violaciones al margen: $\xi_1 = \cdots = \xi_n = 0$
- Para el resto de casos, el número máximo de datos que pueden estar en el lado incorrecto del hiperplano está limitado por el *presupuesto* (máximo *1/C'* datos).
- Tanto como el *presupuesto* se incremente (*C'* disminuya), habrá más tolerancia respecto a las violaciones al margen (el margen se ensanchará).
- Tanto como el *presupuesto* disminuya (*C'* aumente), habrá menos tolerancia y el margen disminuirá.

Detalles del clasificador mediante vectores de soporte



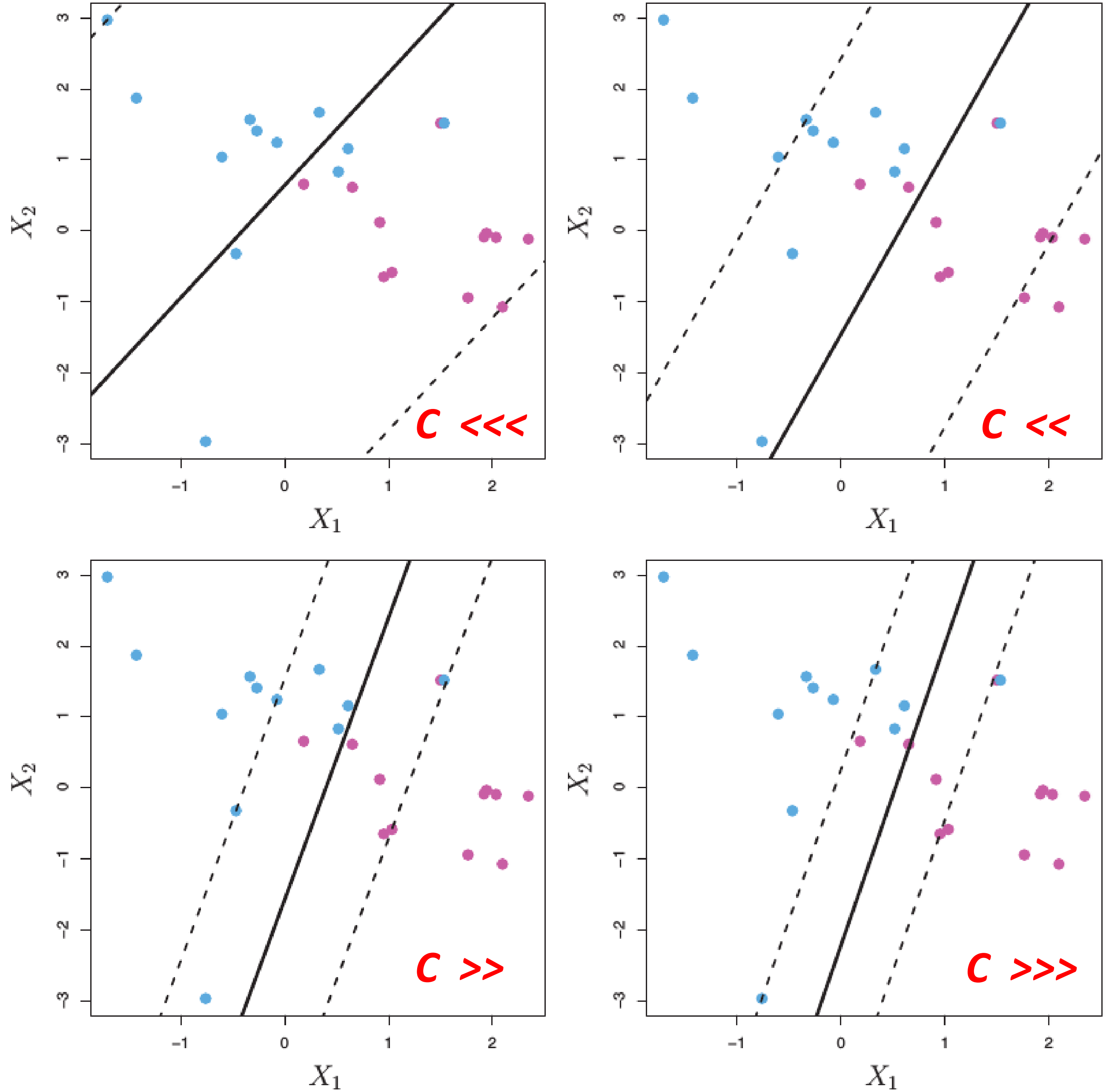
$$\max_{\beta_0,\beta,\xi,\|\boldsymbol{\beta}\|=1} M$$

$$\text{sujeto a}: \begin{cases} y_i\left(\mathbf{x}^T\boldsymbol{\beta} + \beta_0\right) \geq M\left(1-\xi_i\right) \\ \xi_i \geq 0,\ \sum_{i=1}^{n} \xi_i \leq \dfrac{1}{C'} \end{cases}$$

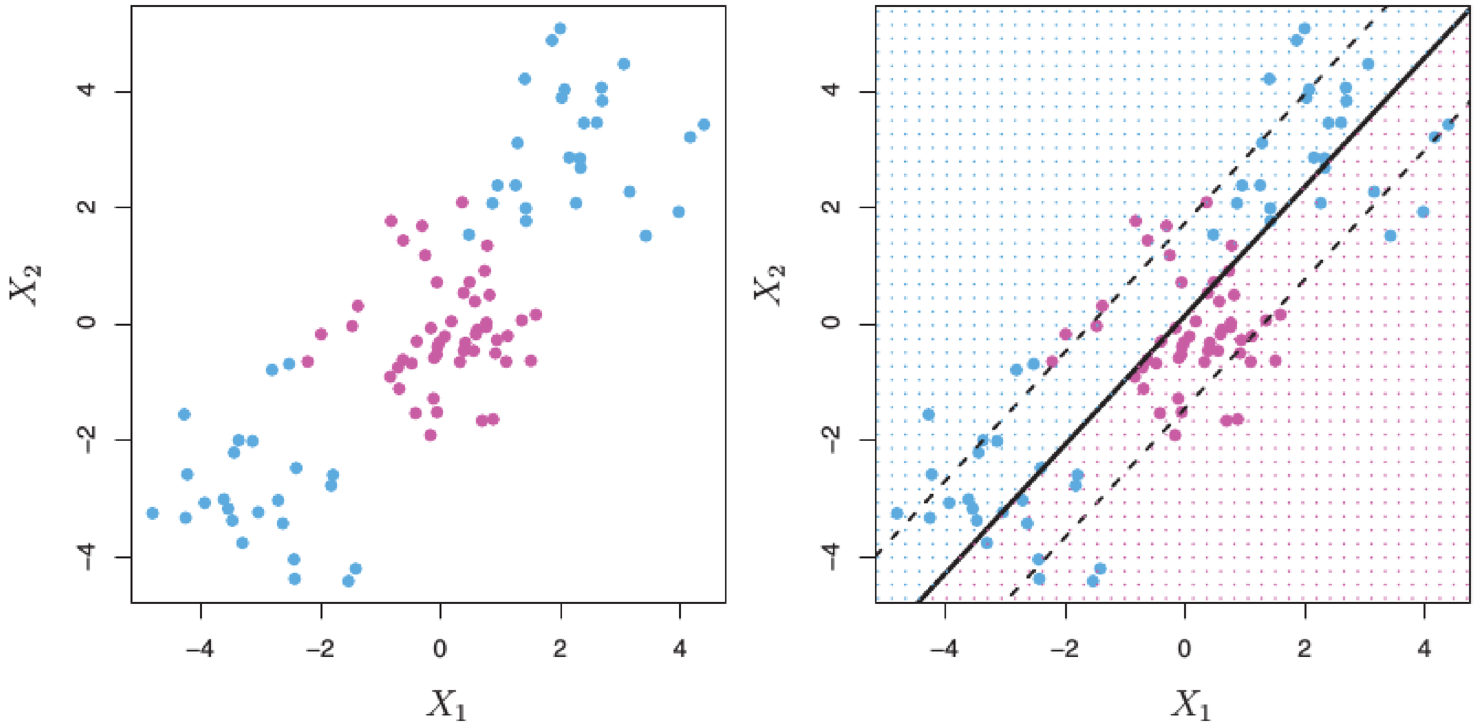
La solución a este problema de **optimización**

- ✓ Sólo los datos que se encuentran sobre el margen o que violan el margen afectan a la determinación del hiperplano. Es decir, **los datos que se encuentran en el lado correcto del margen no afectan al clasificador**.
- ✓ Los datos que se encuentran en el margen o en el lado incorrecto del margen son los **vectores de soporte**.

C es un hiperparámetro



¿Qué sucede cuando hay fronteras de decisión no lineales?

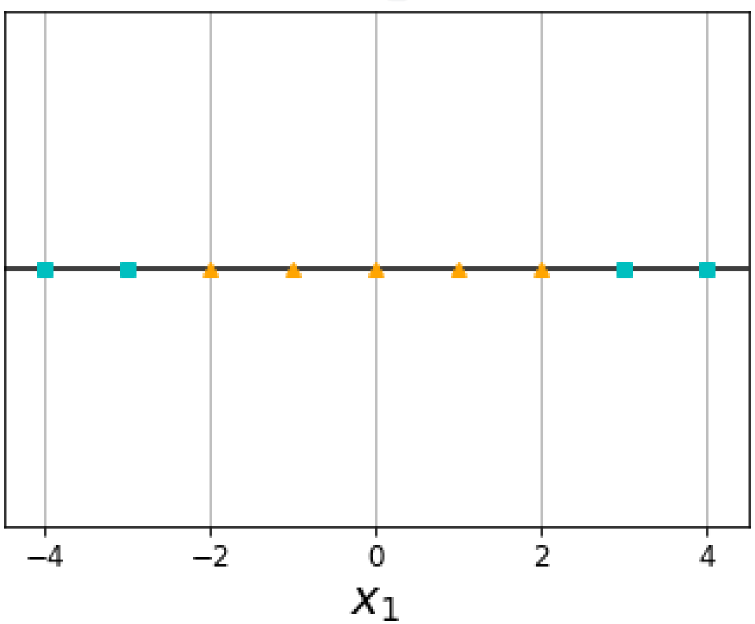


- Algunas veces una frontera decisión lineal simplemente no trabaja, independiente del valor de *C*.
- ¿Qué se puede hacer? -> Máquinas de vectores de soporte

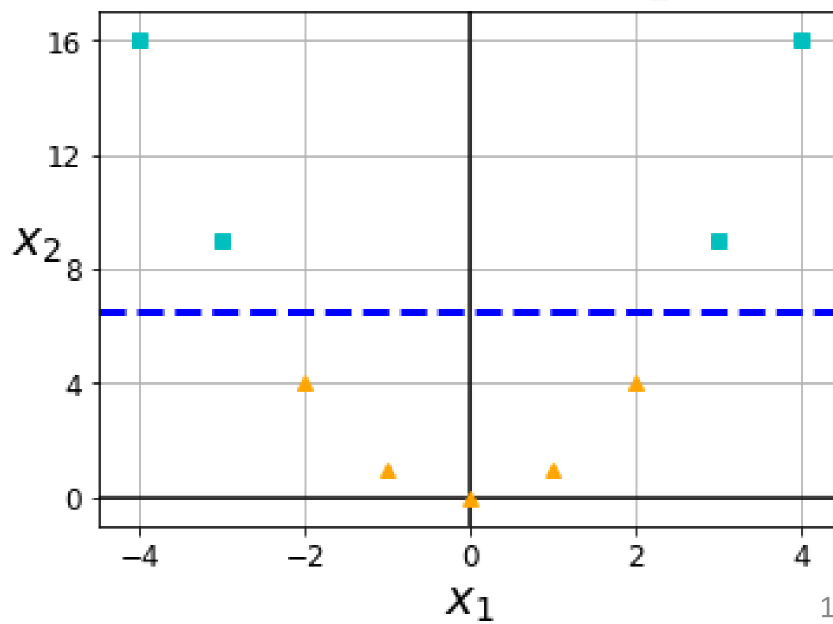
Ampliación de los descriptores (feature expansion)

- Incrementar el espacio de descriptores incluyendo transformaciones, por ejm: $X_1^2, X_1^3, X_1X_2, X_1X_2^2$, …. Se va de un espacio *p-dimensional* a uno *m-dimensional,* donde *m > p.*
- Se ajusta el clasificador mediante vectores de soporte en el espacio ampliado.
- Esto resulta en una frontera de decisión no lineal en el espacio (de descriptores) original.

x_1

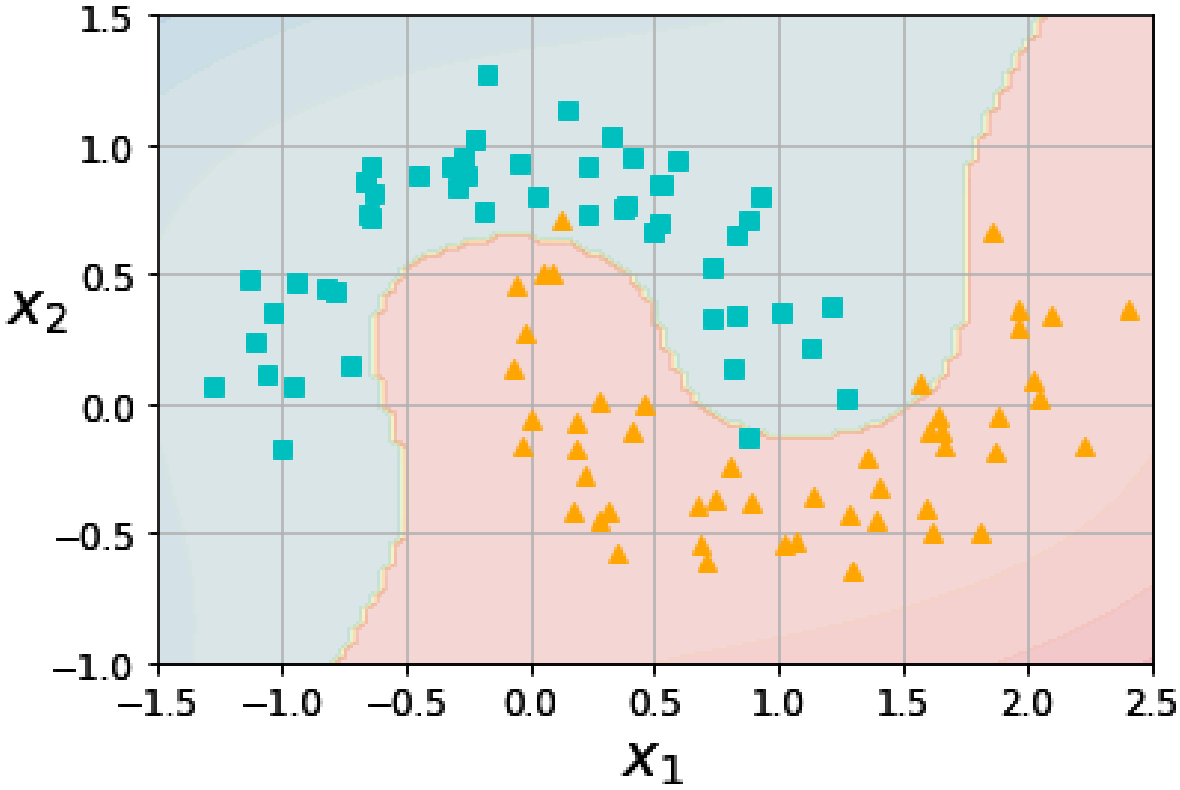


$x_1, x_2 = x_1^2$



Ampliación de los descriptores: polinomio cúbico

$$\beta_0+\beta_1 X_1+\beta_2 X_2+\beta_3 X_1^2+\beta_4 X_2^2+\beta_5 X_1 X_2+\beta_6 X_1^3+\beta_7 X_2^3+\beta_8 X_1 X_2^2+\beta_9 X_1^2 X_2=0$$



- Se ha usado una ampliación cúbica polinomial
- Se incrementa de 2 a 9 variables
- El clasificador mediante vectores de soporte en el espacio ampliado soluciona el problema en el espacio original (de baja dimensión).

Producto interno (escalar) y vectores de soporte

Producto interno: $\langle \boldsymbol{x}_i | \boldsymbol{x}_{i'} \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j}$

- El clasificador mediante vectores de soporte puede representarse como:

$$f(x) = \beta_0 + \sum_{i=1}^{n} \underbrace{\alpha_i}_{\tilde{\alpha}_i y_i} \langle \boldsymbol{x} | \boldsymbol{x}_i \rangle$$

- Para estimar los *n* parámetros $\alpha_1, \alpha_2, \ldots, \alpha_n$ y β_0, se requieren $\binom{n}{2}$ productos internos $\langle \boldsymbol{x}_i | \boldsymbol{x}_{i'} \rangle$ entre todos los pares de datos de entrenamiento.
- Resulta que la mayoría de α_i pueden ser cero (los que no están relacionados a los vectores de soporte):

$$f(x) = \beta_0 + \sum_{i \epsilon S}^{n} \alpha_i \langle \boldsymbol{x} | \boldsymbol{x}_i \rangle$$

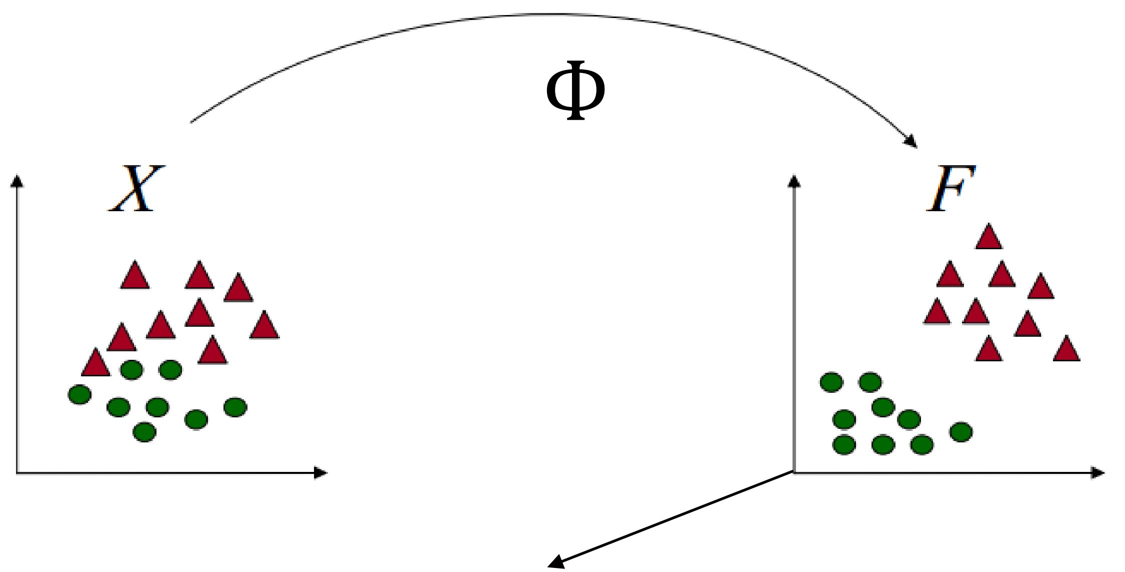
Donde S es el conjunto (de vectores) de soporte de los índices i tal que $\alpha_i > 0$.

El truco del kernel (kernel trick)

- Se mapean los datos a un espacio de descriptores ampliado vía una función de mapeo $\Phi(x)$.
- El mapeo puede evaluarse a través de una función kernel (teorema de Mercer):

$$K(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = \langle \Phi(\boldsymbol{x}_i) | \Phi(\boldsymbol{x}_{i'}) \rangle$$

- Se construye una función lineal en el nuevo espacio ampliado de descriptores.



Máquinas de vectores de soporte (SVM)

- Si se pueden calcular los productos internos entre los datos, se puede ajustar un clasificador SVM.

- Algunas funciones kernel especiales pueden hacer esto directamente, por ejm:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = (\gamma\langle \boldsymbol{x}_i, \boldsymbol{x}_{i'}\rangle + r)^d$$

  calcula el productor interno necesario para un polinomio de dimensión *d*, que normalmente necesitaría calcular un número $\binom{p+d}{d}$ de funciones base.

- La solución ahora tiene la forma:

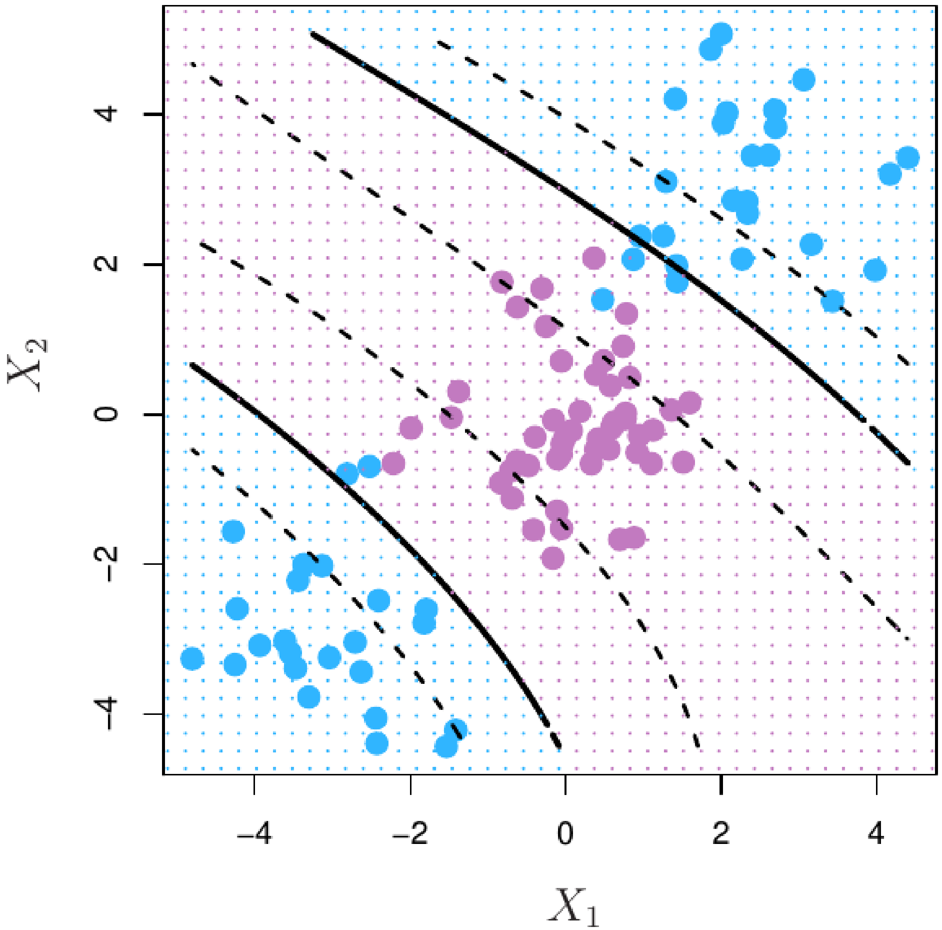
$$f(x) = \beta_0 + \sum_{i \epsilon S}^{n} \widehat{\alpha}_i K(\boldsymbol{x}_i, \boldsymbol{x}_{i'})$$

Máquinas de vectores de soporte (SVM)

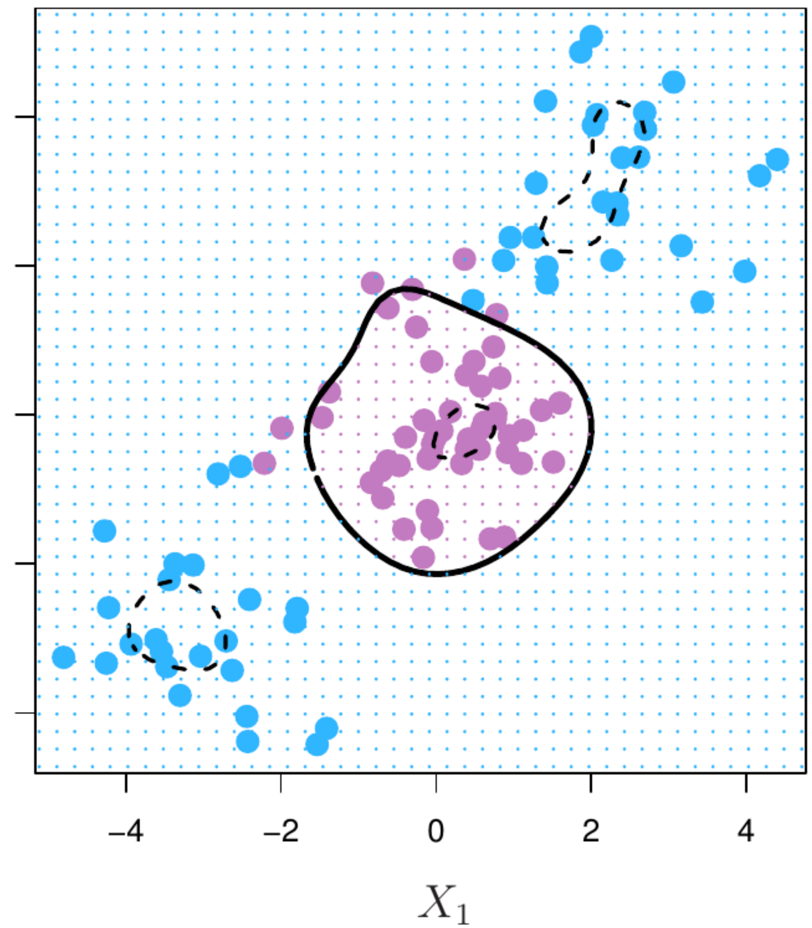
Algunos tipos usuales de funciones kernel son:

- Lineal: $K(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = \langle \boldsymbol{x}_i, \boldsymbol{x}_{i'} \rangle$
- Polinomial: $K(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = (\gamma \langle \boldsymbol{x}_i, \boldsymbol{x}_{i'} \rangle + r)^d$
- Gaussiano (RBF):
$$K(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = \exp(-\gamma \|\boldsymbol{x}_i - \boldsymbol{x}_{i'}\|^2)$$
- Sigmoide:
$$K(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = \tanh(\gamma \langle \boldsymbol{x}_i, \boldsymbol{x}_{i'} \rangle + r)$$

Máquinas de vectores de soporte (SVM)



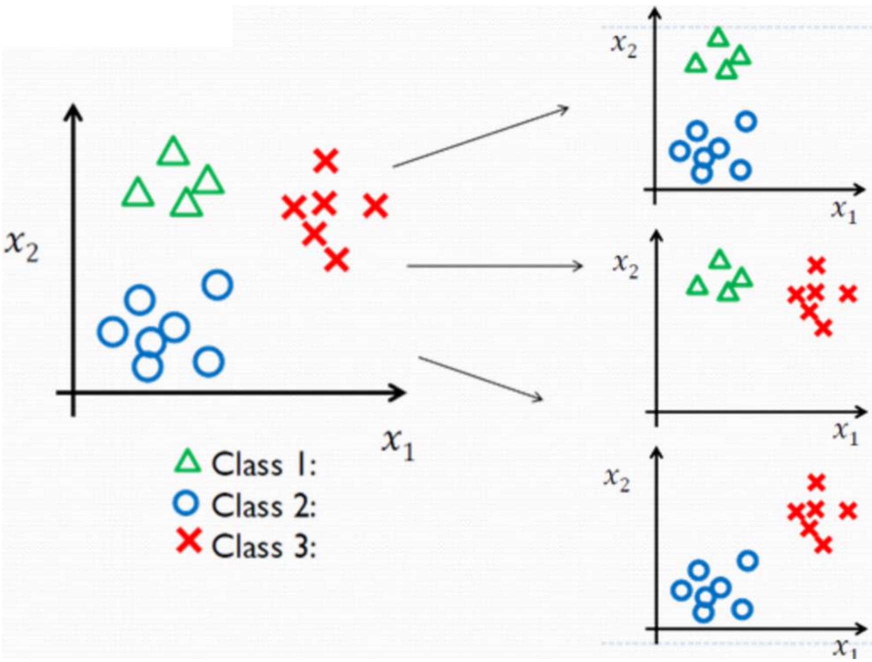
Kernel polinomial de grado 3



Kernel Gaussiano

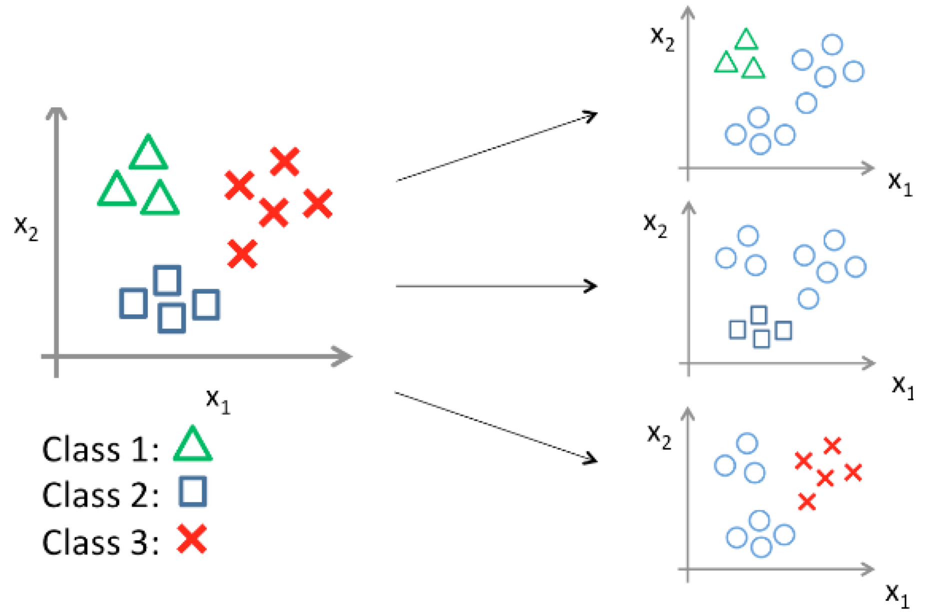
SVM con más de dos clases

One vs one



- Se escoge para cada observación la clase que gane la mayoría de clasificaciones
- $\binom{K}{2}$ clasificadores

One vs All (one vs rest)



- Se escoge para cada observación la clase con la mayor probabilidad.
- Un clasificador por clase.

Implementación con scikit-learn

Class	Time complexity	Out-of-core support	Scaling required	Kernel trick
`LinearSVC`	$O(m \times n)$	No	Yes	No
`SGDClassifier`	$O(m \times n)$	Yes	Yes	No
`SVC`	$O(m^2 \times n)$ to $O(m^3 \times n)$	No	Yes	Yes

Implementación con scikit-learn

```
from sklearn.svm import LinearSVC
```

Librería: **liblinear**

Muy rápida, pero solo kernel lineal

```
iris = datasets.load_iris()
X = iris["data"]
y = (iris["target"])
svm_clf = Pipeline((
        ("scaler", StandardScaler()),
        ("linear_svc", LinearSVC(C=1)),
    ))

svm_clf.fit(X, y)
```

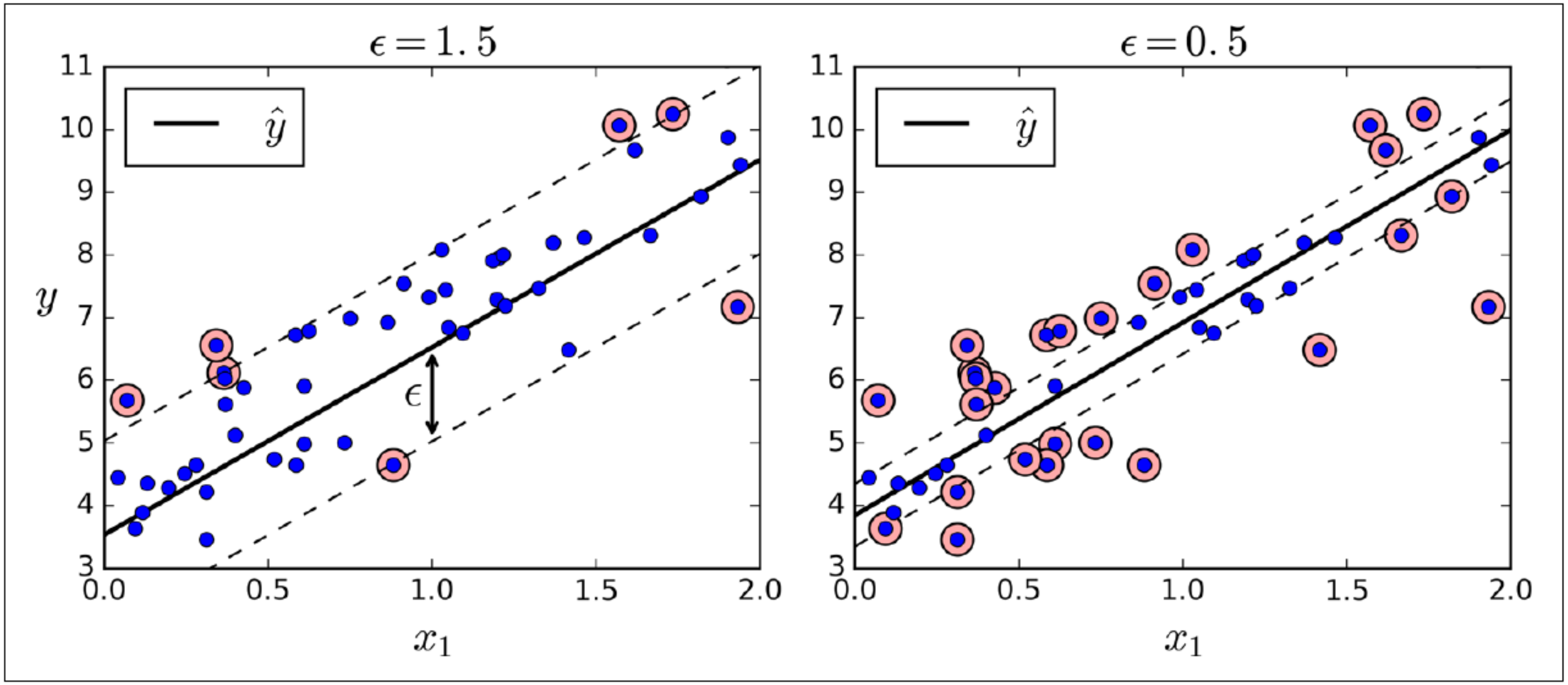
Implementación con scikit-learn

Librería: **libsvm**
No tan rápida
Diferentes kernels

```
from sklearn.svm import SVC

iris = datasets.load_iris()
X = iris["data"]
y = (iris["target"])

svm_clf = Pipeline((
        ("scaler", StandardScaler()),
        ("RBF_svc", SVC(C=10, kernel='rbf', gamma=0.5)),
    ))

svm_clf.fit(X, y)
```

Support vector regression (SVR)

- El método de SVM puede ampliarse para resolver problemas de regresión. Este método se llama Regresión mediante vectores de soporte (SVR).



Implementación con scikit-learn

Librería: **libsvm**
No tan rápida
Diferentes kernels

```python
from sklearn.svm import SVR
X = [[0, 0], [2, 2]]
y = [0.5, 2.5]
clf = SVR()
clf.fit(X, y)


# clf.predict([[1, 1]])
clf
```

```
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='auto',
  kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
```