

# MÁQUINAS DE VECTORES DE SOPORTE

APRENDIZAJE DE MÁQUINA I - CEIA - FIUBA

Dr. Ing. Facundo Adrián Lucianna

Esp. Lic. María Carina Roldán

# REPASO CLASE ANTERIOR

- KNN
- Métodos de Ajuste de los hiper-parámetros
- Búsqueda de grilla
- Búsqueda aleatoria
- Búsquedas más avanzadas (Framework Optuna)

# CLASIFICACIÓN

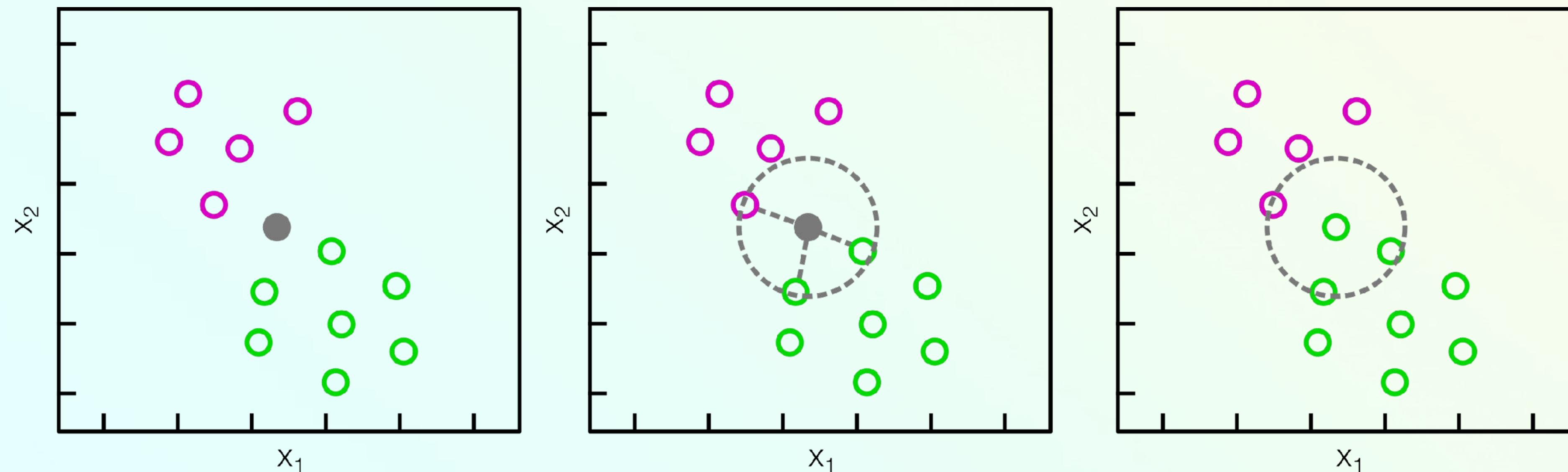
Es más común encontrarnos con problema de clasificación que de regresión:

- Una persona llega a una guardia con un set de síntomas atribuidos a una de tres condiciones médicas.
- Un servicio de banca online debe determinar si una transacción en el sitio es fraudulenta o no, usando como base la dirección IP, historia de transacciones, etc.
- En base a la secuencia de ADN de un número de pacientes con y sin una enfermedad dada, un genetista debe determinar qué mutaciones de ADN generan un efecto nocivo relacionado a la enfermedad.

# KNN

El clasificador de k vecinos más cercanos (KNN o k-NN), es un algoritmo que utiliza la proximidad de sus vecinos para hacer clasificaciones sobre la agrupación de un punto.

La idea se basa de la suposición de que se pueden encontrar puntos similares cerca uno del otro en base a votación de pluralidad (se elige la clase en función de la moda de la clase de sus vecinos).



# MÉTODOS DE AJUSTE DE LOS HIPER-PARÁMETROS

En Inteligencia Artificial se vió que **validación cruzada** nos sirve para buscar ayudarnos a buscar los hiper-parámetros que mejor se nos ajustan a nuestros modelos, permitiendo mantener la generalidad.

Pero por sí solo, no alcanza, necesitamos de alguna forma *movernos* por el espacio de búsqueda.

Una búsqueda consiste en:

- Un modelo (de regresión o clasificación)
- Un espacio de parámetros
- **Un método de búsqueda o de muestreo de candidatos**
- Un esquema de validación cruzada
- Una función de puntaje

# MÉTODOS DE AJUSTE DE LOS HIPER-PARÁMETROS

Dos métodos de búsqueda típicos:

- Búsqueda de grilla: Busca exhaustiva de todas las combinaciones de los parámetros. Es el más completo pero el más ineficiente.
- Búsqueda aleatoria: Busca aleatoriamente tomando datos del espacio de combinaciones de los parámetros, bajo una distribución aleatoria dada. Termina dado una cierta cantidad arbitraria de iteraciones.

# MÉTODOS DE AJUSTE DE LOS HIPER-PARÁMETROS

Estos métodos de búsquedas se pueden atacar con estrategias de Inteligencia Artificial. Hay muchos estudios que aplican algoritmos que buscan encontrar selecciones de hiper-parámetros eficientes, y cortar evaluaciones de combinaciones no tan atractivas.

Un framework que nos ofrece técnicas más avanzadas de búsqueda de hiper-parámetros es **Optuna**. Para buscar hiper-parámetros realiza dos acciones que ayudan a ser más eficiente en su búsqueda:

- Selección de hiper-parámetros que pueden dar buenos resultados.
- Podado de hiper-parámetros que es innecesario buscar por malos resultados.

# **CLASIFICADORES LINEALES**

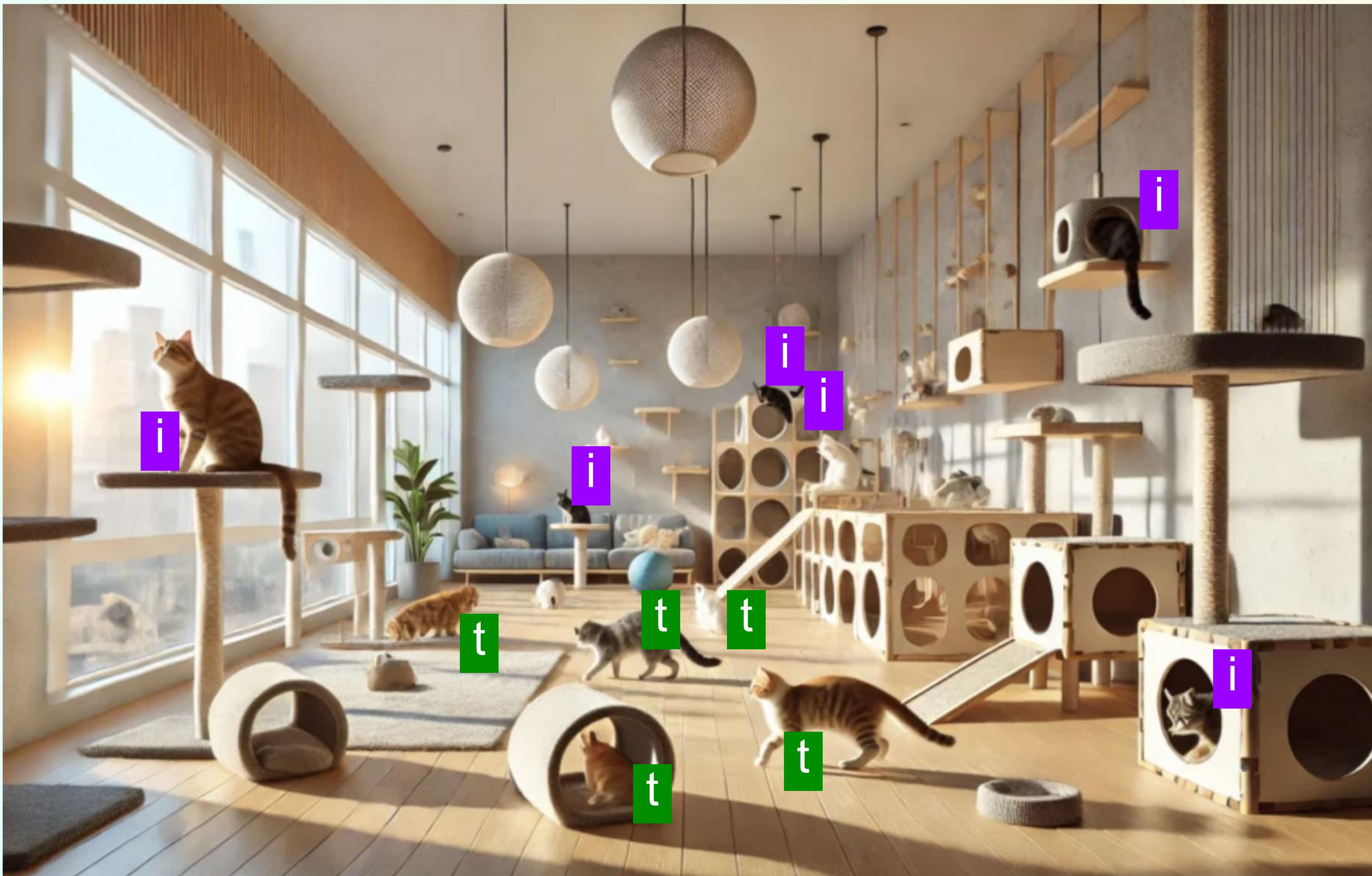
# Separando las clases



Quiero clasificar:

- gatitos inquietos
- gatitos tranquilos

# Separando las clases

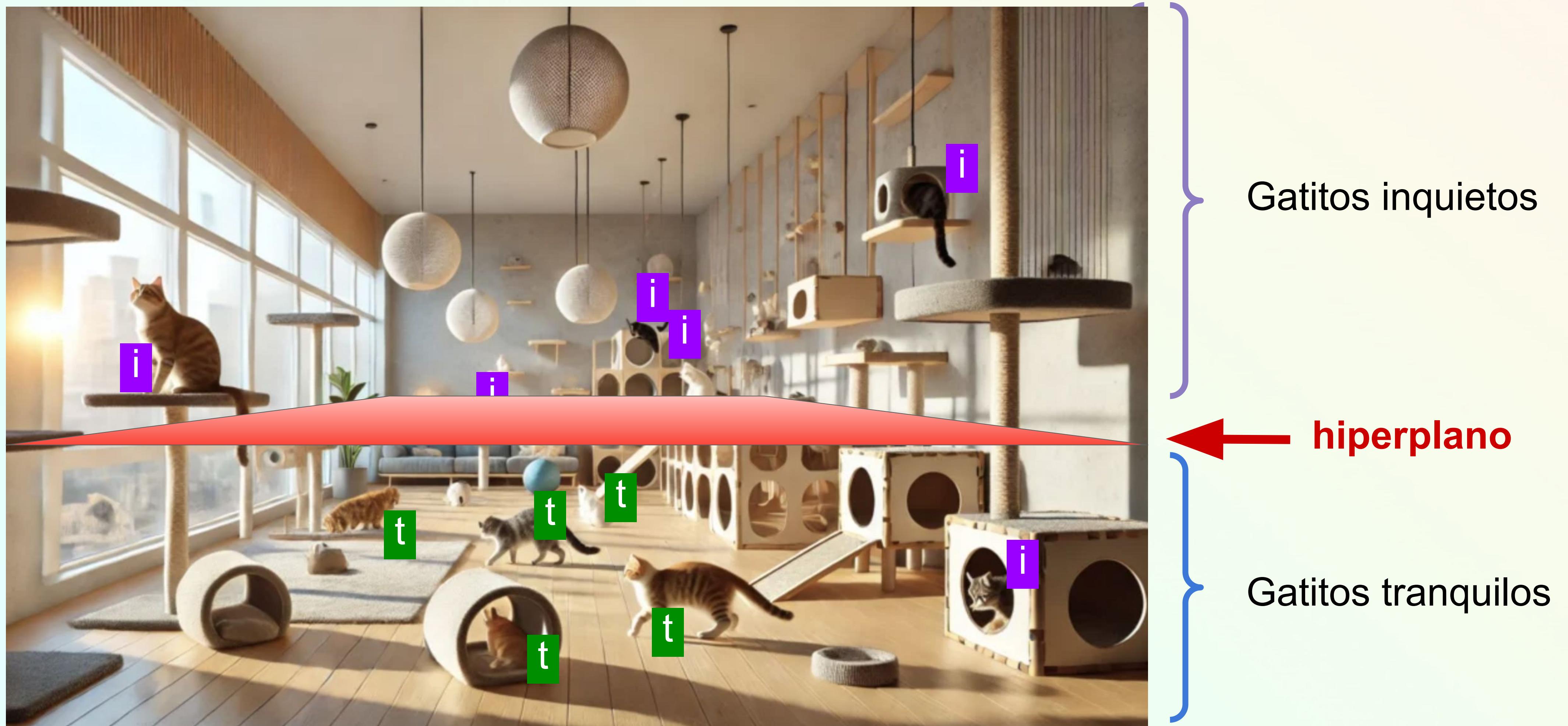


Quiero clasificar:

- gatitos inquietos
- gatitos tranquilos

set de  
entrenamiento

# Separando las clases



# Separando las clases



HIPERPLANO



# **MAXIMAL MARGIN CLASSIFIER**

**( CLASIFICADORES DE MÁXIMO MARGEN  
o *hard-margin SVN*)**

# MAXIMAL MARGIN CLASSIFIER

Empecemos con álgebra, en un espacio p-dimensional, un **hiperplano** es un subespacio de dimensión p-1. Por ejemplo,

- En 2 dimensiones, el hiperplano es la recta.
- En 3 dimensiones, el hiperplano es el plano.

La definición matemática ya la conocemos:

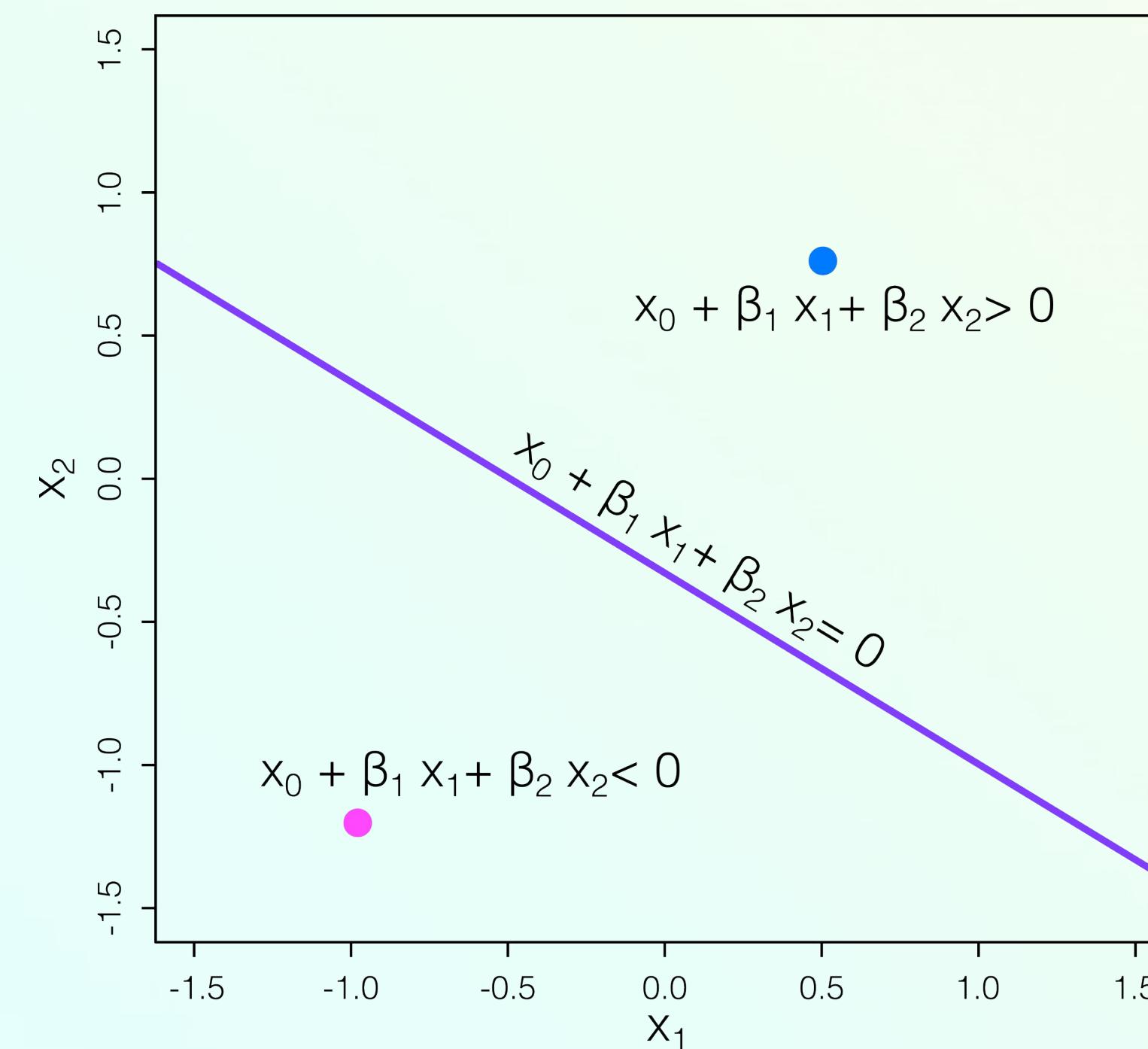
$$\text{2 dimensiones: } \beta_0 + \beta_1 x_1 + \beta_2 x_2 = 0 \quad \beta_0, \beta_1, \beta_2 \in \mathbb{R}$$

$$\text{p-dimensiones: } \beta_0 + \beta_1 x_1 + \dots + \beta_p x_p = 0 \quad \beta_0, \beta_1, \dots, \beta_p \in \mathbb{R}$$

# MAXIMAL MARGIN CLASSIFIER

Dado un punto  $X = (x_1, x_2, \dots, x_p)$ , si verifica la ecuación, decimos que pertenece al hiperplano.

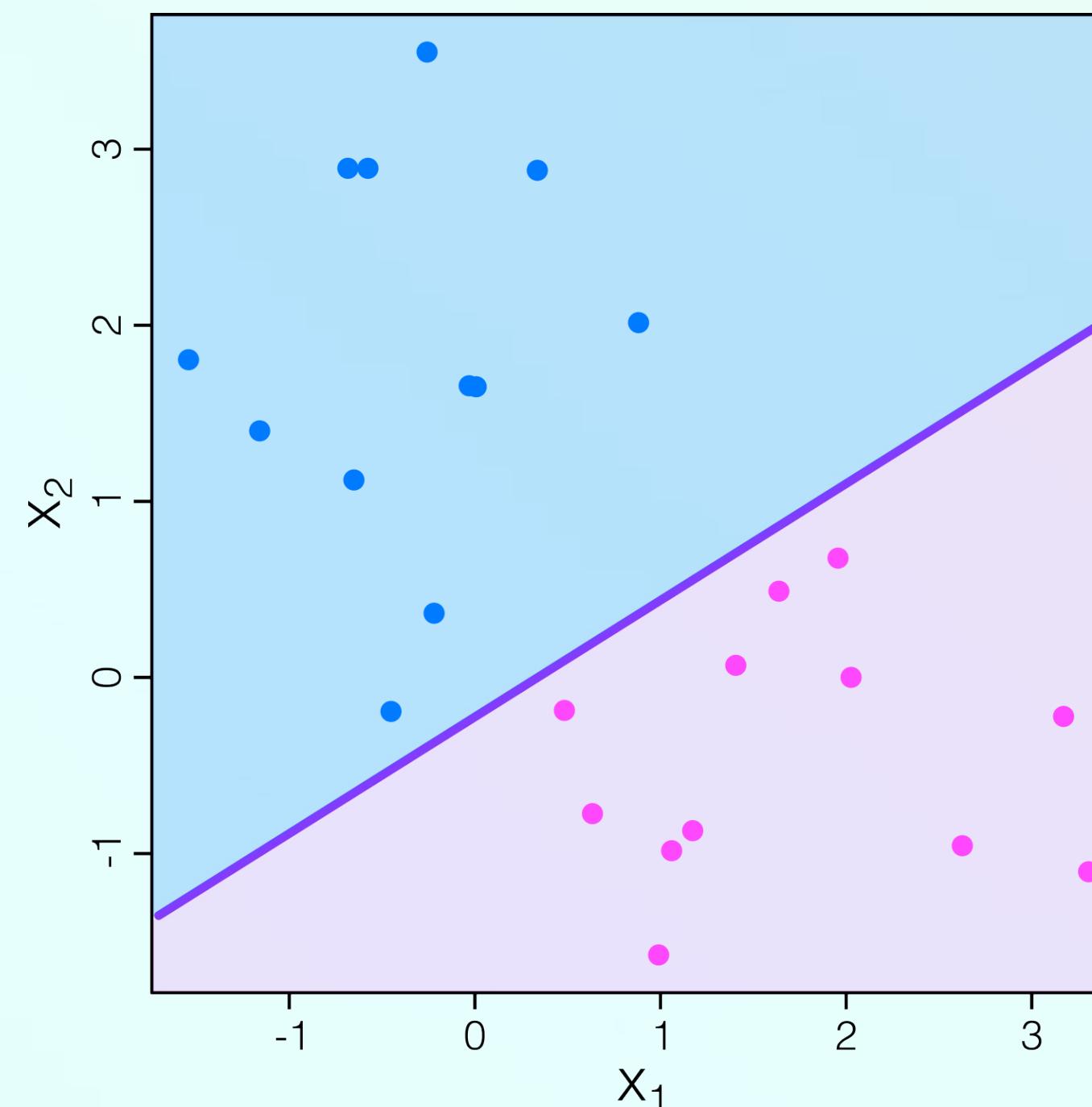
Pero más interesante, si el punto hace que,



# MAXIMAL MARGIN CLASSIFIER

Podemos usar esta idea para clasificar, si tenemos observaciones con  $p$  atributos, y estas caen en dos posibles clases  $\{-1, 1\}$ .

Si de alguna forma podemos construir un hiperplano que separa los datos de entrenamiento perfectamente de acuerdo con su clase, podríamos clasificar como:



$$\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip} > 0 \quad \text{Si } y_i = 1$$

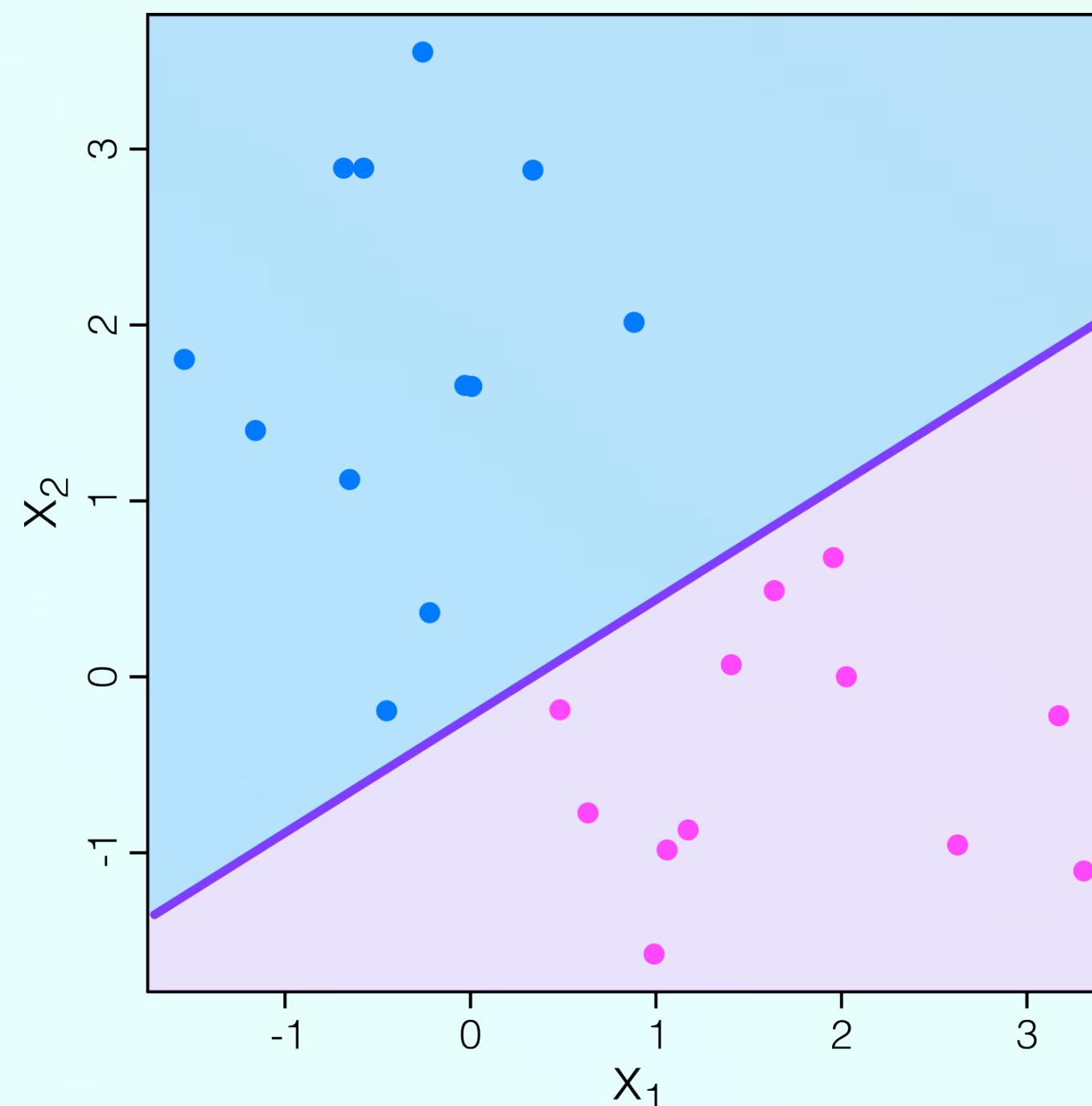
$$\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip} < 0 \quad \text{Si } y_i = -1$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) > 0 \quad i = 1, \dots, n$$

# MAXIMAL MARGIN CLASSIFIER

Podemos usar esta idea para clasificar, si tenemos observaciones con  $p$  atributos, y estas caen en dos posibles clases  $\{-1, 1\}$ .

Si de alguna forma podemos construir un hiperplano que separa los datos de entrenamiento perfectamente de acuerdo con su clase, podríamos clasificar como:



Para ordenarnos, vamos a definir una función  $f(X)$  que nos servirá para clasificar:

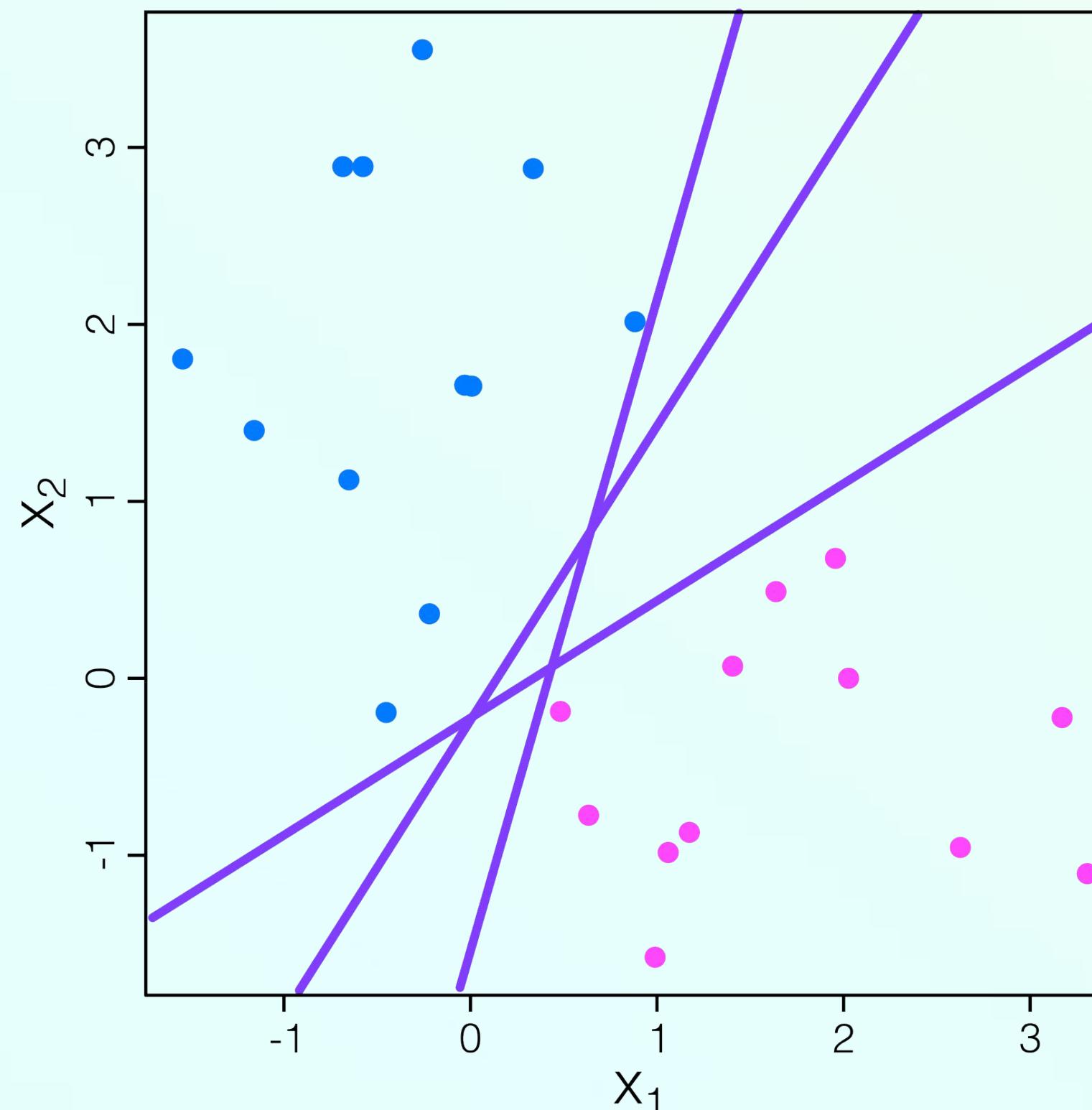
$$f(X_i) = \beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip} > 0 \text{ entonces } \tilde{y}_i = 1$$

$$f(X_i) = \beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip} < 0 \text{ entonces } \tilde{y}_i = -1$$

Que, además, nos permite ver la confidencia del clasificador. Es decir, cuanto más **grande** es el valor absoluto de  $f(X)$ , más **segura o certera** es la clasificación.

# MAXIMAL MARGIN CLASSIFIER

En general si nuestra data es linealmente separable, puede existir un número infinito de hiperplanos que van a funcionar.



Por lo que necesitamos algún criterio de selección.

El caso que aquí estamos en busca del hiperplano que más lejos se encuentra de los datos de entrenamiento.

Es decir:

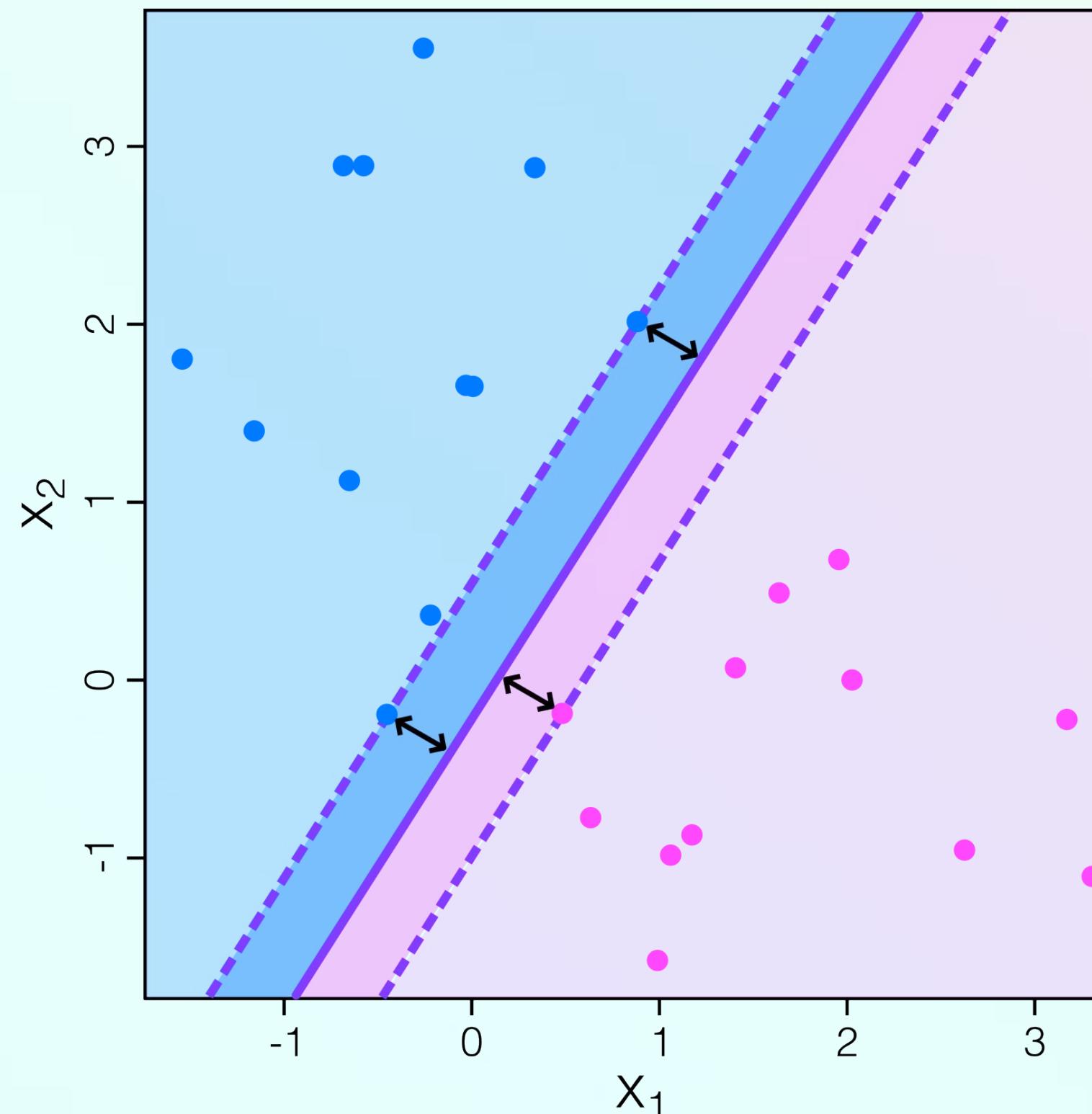
1. Tomamos un hiperplano.
2. Calculamos las distancias de cada punto de entrenamiento.
3. Obtenemos la distancia más chica de todas las distancias.

Esa distancia la llamamos **margen**.

Cada hiperplano tendrá su margen.

# MAXIMAL MARGIN CLASSIFIER

En general si nuestra data es linealmente separable, puede existir un número infinito de hiperplanos que van a funcionar.

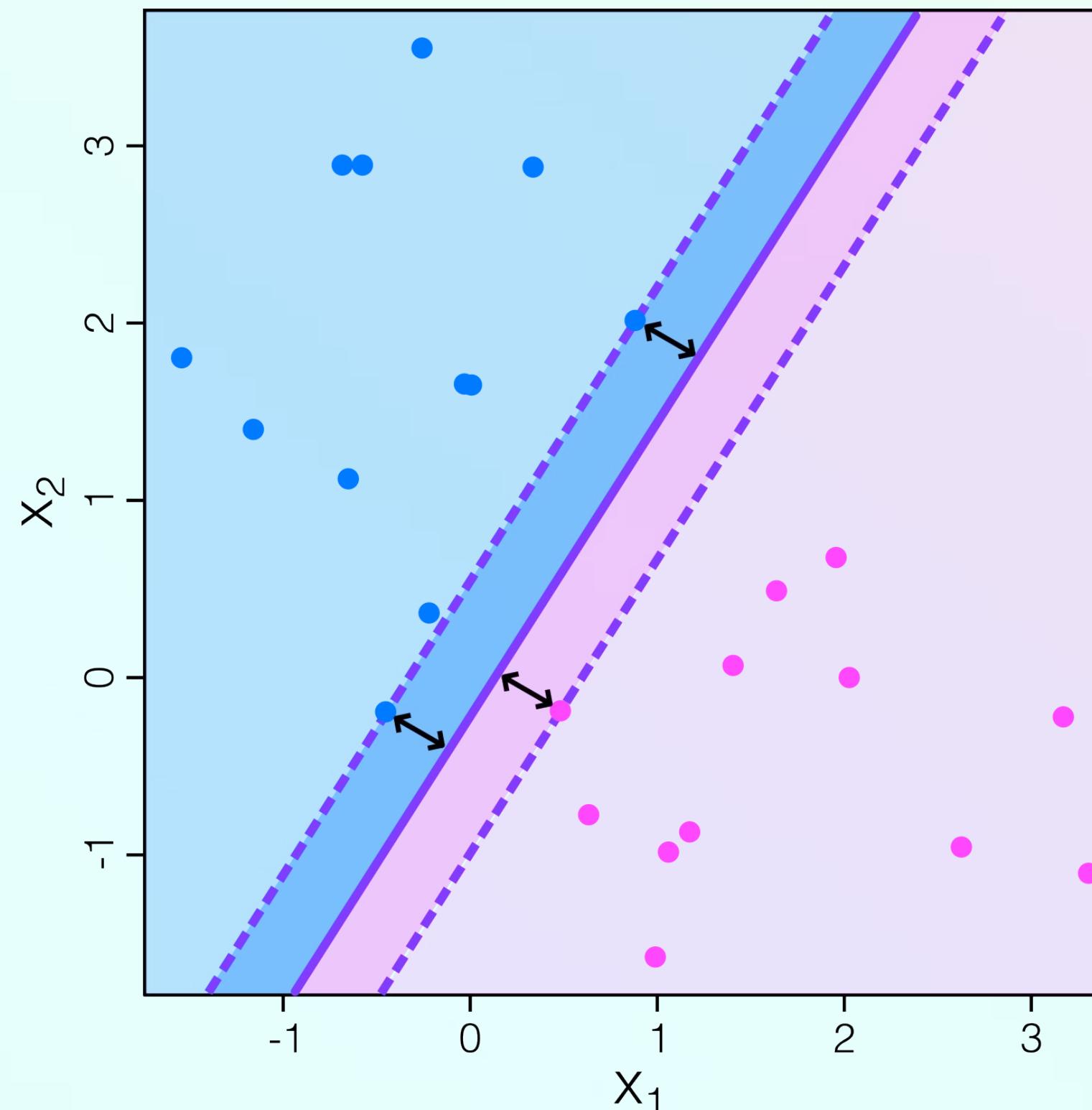


El objetivo es buscar el hiperplano con mayor margen y, el algoritmo que hace esto es el **Maximal Margin Classifier**.

Podemos pensar que el clasificador busca, entre las rectas que pueden pasar entre las clases, la de máximo **grosor**.

# MAXIMAL MARGIN CLASSIFIER

En este ejemplo, vemos que hay tres observaciones que están equidistante desde el hiperplano de máximo margen.



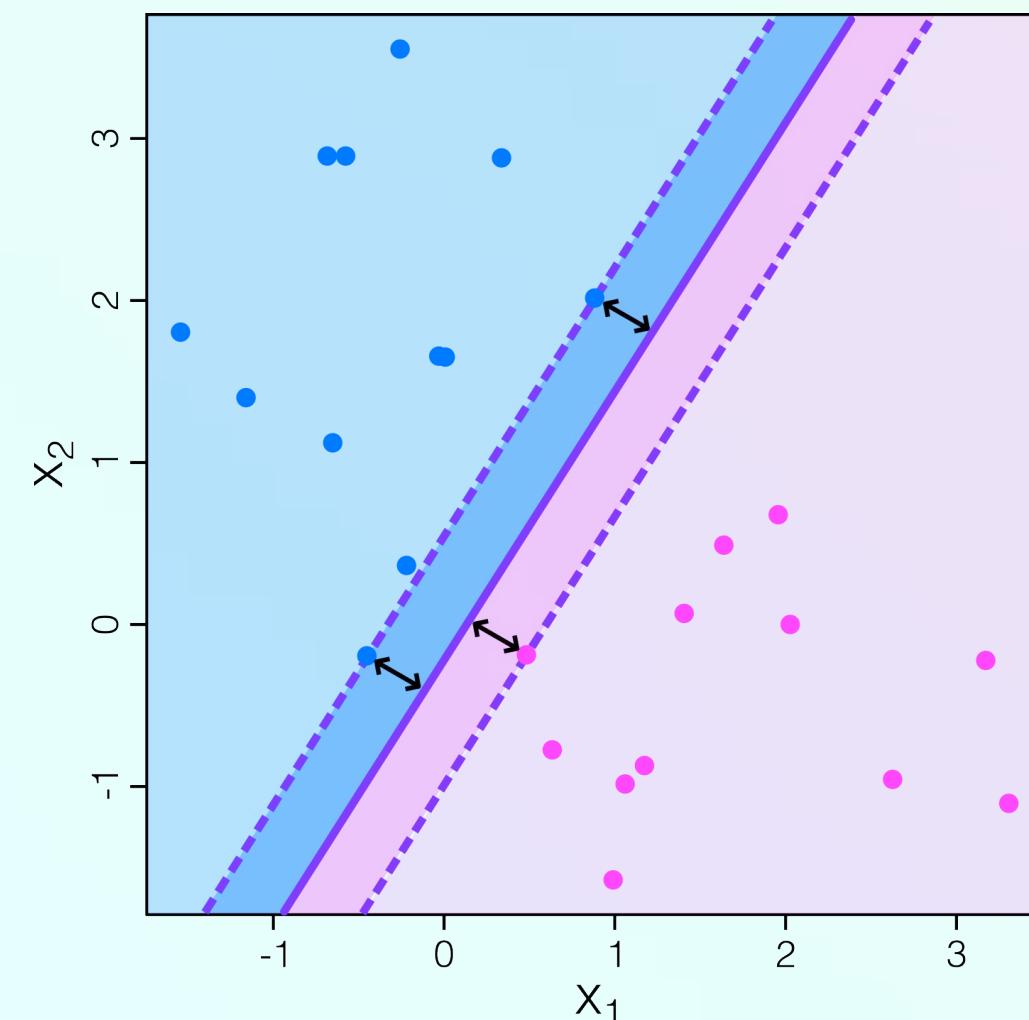
Estas tres observaciones son conocidas como **vectores de soportes**, dado que soportan el hiperplano.

Esto es porque si uno de estos puntos se mueve, aunque sea un poco, el hiperplano también se va a mover.

El hiperplano depende solo de esos tres puntos y no del resto de las observaciones.

# MAXIMAL MARGIN CLASSIFIER

Para construir este hiperplano, necesitamos resolver el problema de optimización:



$$\begin{aligned} & \text{maximizar } M \\ & \beta_0, \beta_1, \dots, \beta_p \end{aligned}$$

$$\text{sujeito a } \sum_{j=1}^p \beta_j^2 = 1$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) > M \quad \forall i = 1, \dots, n$$

Para cada observación de entrenamiento

# MAXIMAL MARGIN CLASSIFIER

Para construir este hiperplano, necesitamos resolver el problema de optimización:

maximizar  $M$   
 $\beta_0, \beta_1, \dots, \beta_p$

$$\begin{aligned} & \exists \beta_0, \dots, \beta_p \quad \beta_0 + \beta_1 x_1 + \dots + \beta_p x_p = 0 \\ & \Rightarrow \kappa(\beta_0 + \beta_1 x_1 + \dots + \beta_p x_p) = 0 \quad \kappa \in \mathbb{R} \end{aligned}$$

Hay infinitos valores que verifican esto.

# MAXIMAL MARGIN CLASSIFIER

Para construir este hiperplano, necesitamos resolver el problema de optimización:

$$\underset{\beta_0, \beta_1, \dots, \beta_p}{\text{maximizar}} M$$

$$\text{sujeto a } \sum_{j=1}^r \beta_j^2 = 1$$

$$\begin{aligned} \exists \beta_0, \dots, \beta_p \quad & \beta_0 + \beta_1 x_1 + \dots + \beta_p x_p = 0 \\ \Rightarrow \quad & \kappa(\beta_0 + \beta_1 x_1 + \dots + \beta_p x_p) = 0 \quad \kappa \in \mathbb{R} \end{aligned}$$

Con esta restricción (usamos un versor, vector normal del hiperplano con norma = 1) aseguramos el valor de M y además que:

$$y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip})$$

para una observación i, es la medida de la distancia perpendicular al hiperplano

$$y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) \geq M \quad \forall i = 1, \dots, n$$

# MAXIMAL MARGIN CLASSIFIER

Para construir este hiperplano, necesitamos resolver el problema de optimización:

$$\underset{\beta_0, \beta_1, \dots, \beta_p}{\text{maximizar}} M$$

$$\text{sujeto a } \sum_{j=1}^r \beta_j^2 = 1$$

$$\exists \beta_0, \dots, \beta_p \quad \beta_0 + \beta_1 x_1 + \dots + \beta_p x_p = 0$$

$$\Rightarrow \kappa(\beta_0 + \beta_1 x_1 + \dots + \beta_p x_p) = 0 \quad \kappa \in \mathbb{R}$$

Con esta restricción (usamos un versor, vector normal del hiperplano con norma = 1) aseguramos el valor de M y además que:

$$y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip})$$

para una observación i, es la medida de la distancia perpendicular al hiperplano

$$y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) \geq M \quad \forall i = 1, \dots, n$$

Con estas dos restricciones logramos que cada observación esté del lado correcto y al menos a la distancia de M.

# MAXIMAL MARGIN CLASSIFIER

Para construir este hiperplano, necesitamos resolver el problema de optimización:

$$\underset{\beta_0, \beta_1, \dots, \beta_p}{\text{maximizar}} M$$

$$\text{sujeto a } \sum_{j=1}^r \beta_j^2 = 1$$

$$\exists \beta_0, \dots, \beta_p \quad \beta_0 + \beta_1 x_1 + \dots + \beta_p x_p = 0$$

$$\Rightarrow \kappa(\beta_0 + \beta_1 x_1 + \dots + \beta_p x_p) = 0 \quad \kappa \in \mathbb{R}$$

Con esta restricción (usamos un versor, vector normal del hiperplano con norma = 1) aseguramos el valor de M y además que:

$$y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip})$$

para una observación i, es la medida de la distancia perpendicular al hiperplano

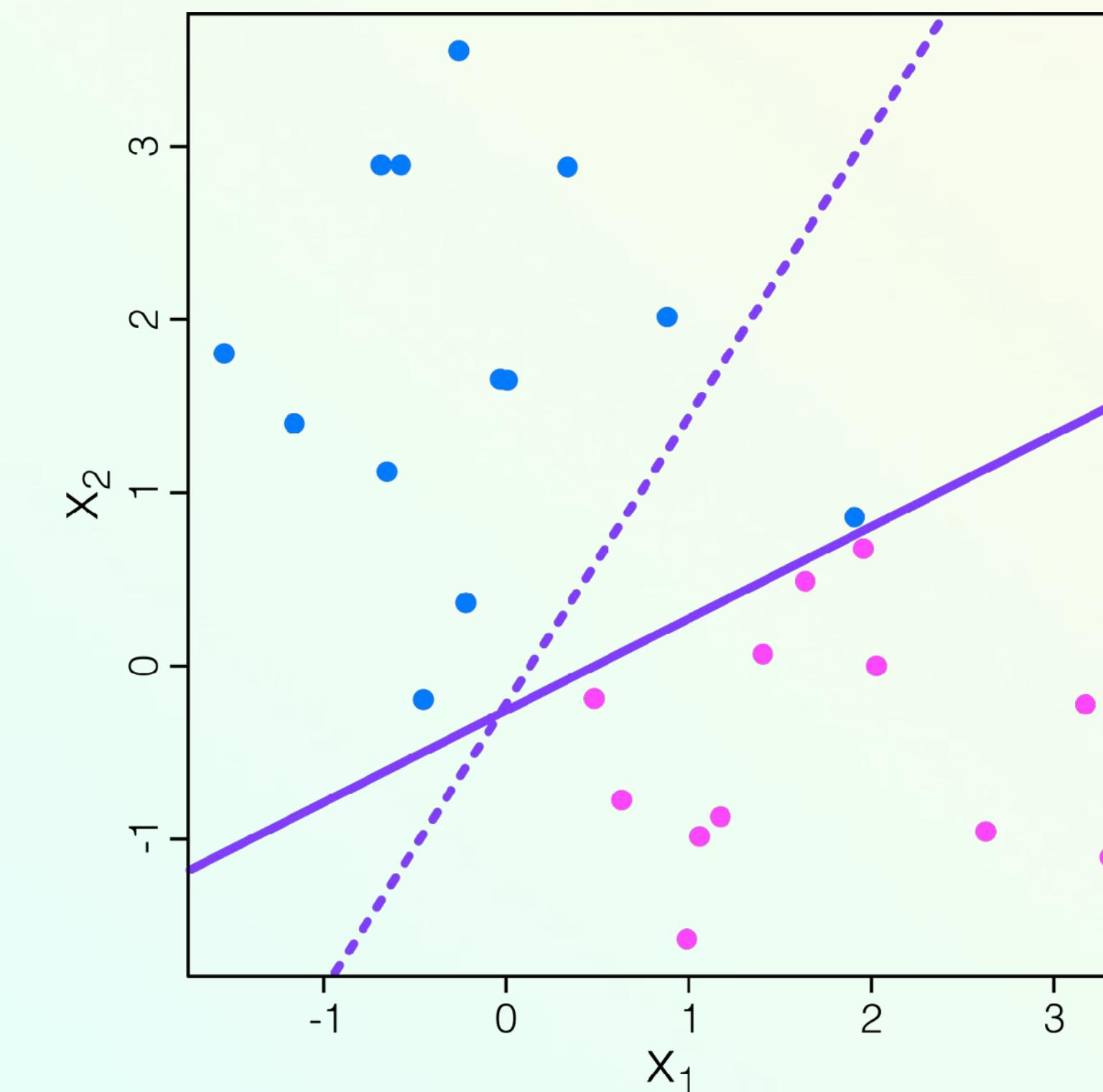
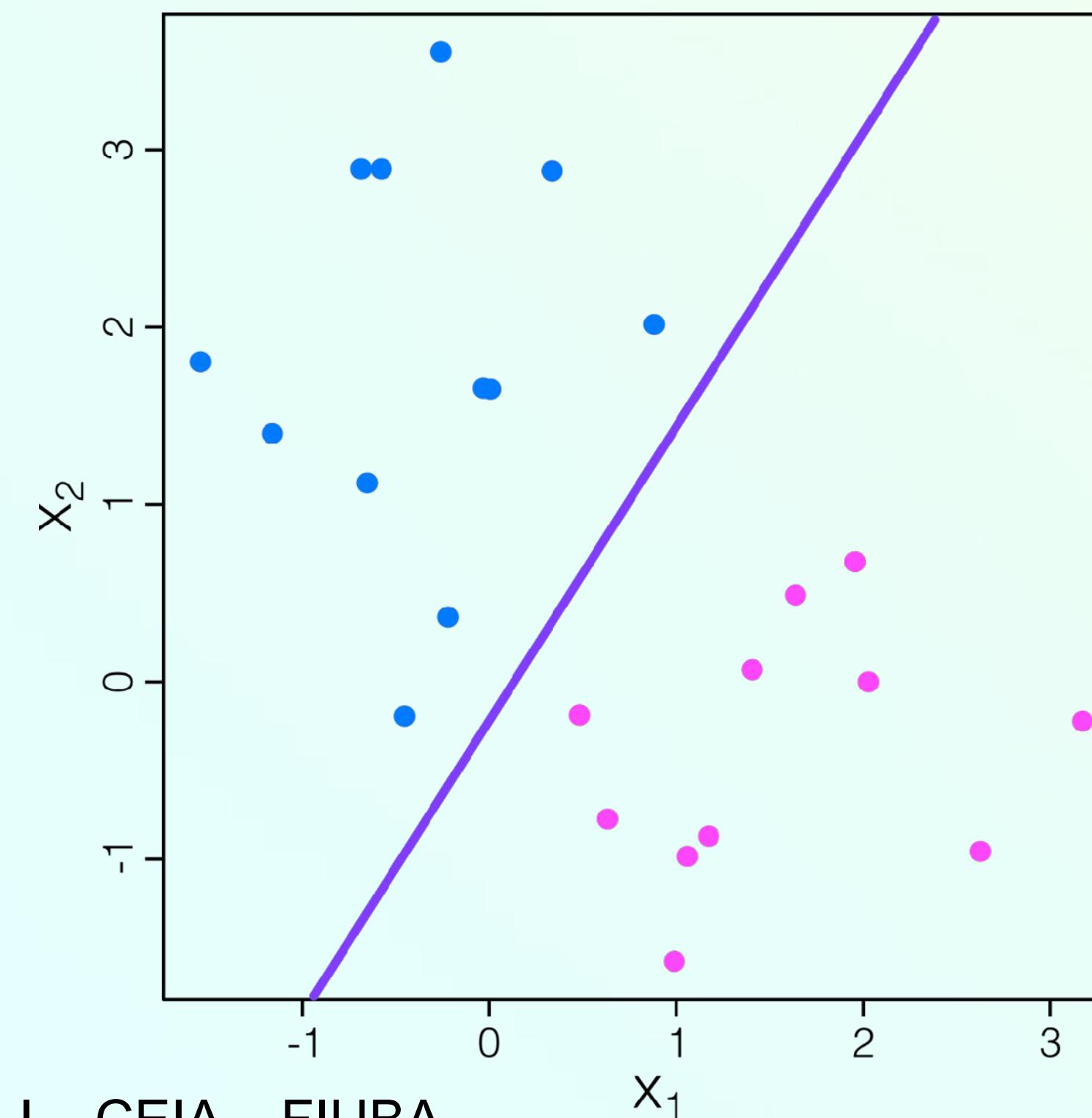
$$y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) \geq M \quad \forall i = 1, \dots, n$$

Con estas dos restricciones logramos que cada observación esté del lado correcto y al menos a la distancia de M.

Es un problema de optimización convexo (criterio cuadrático, desigualdades de restricción lineales) y se resuelve mediante multiplicadores de Lagrange con métodos numéricos tradicionales. Ver Hastie et al. Página 132.

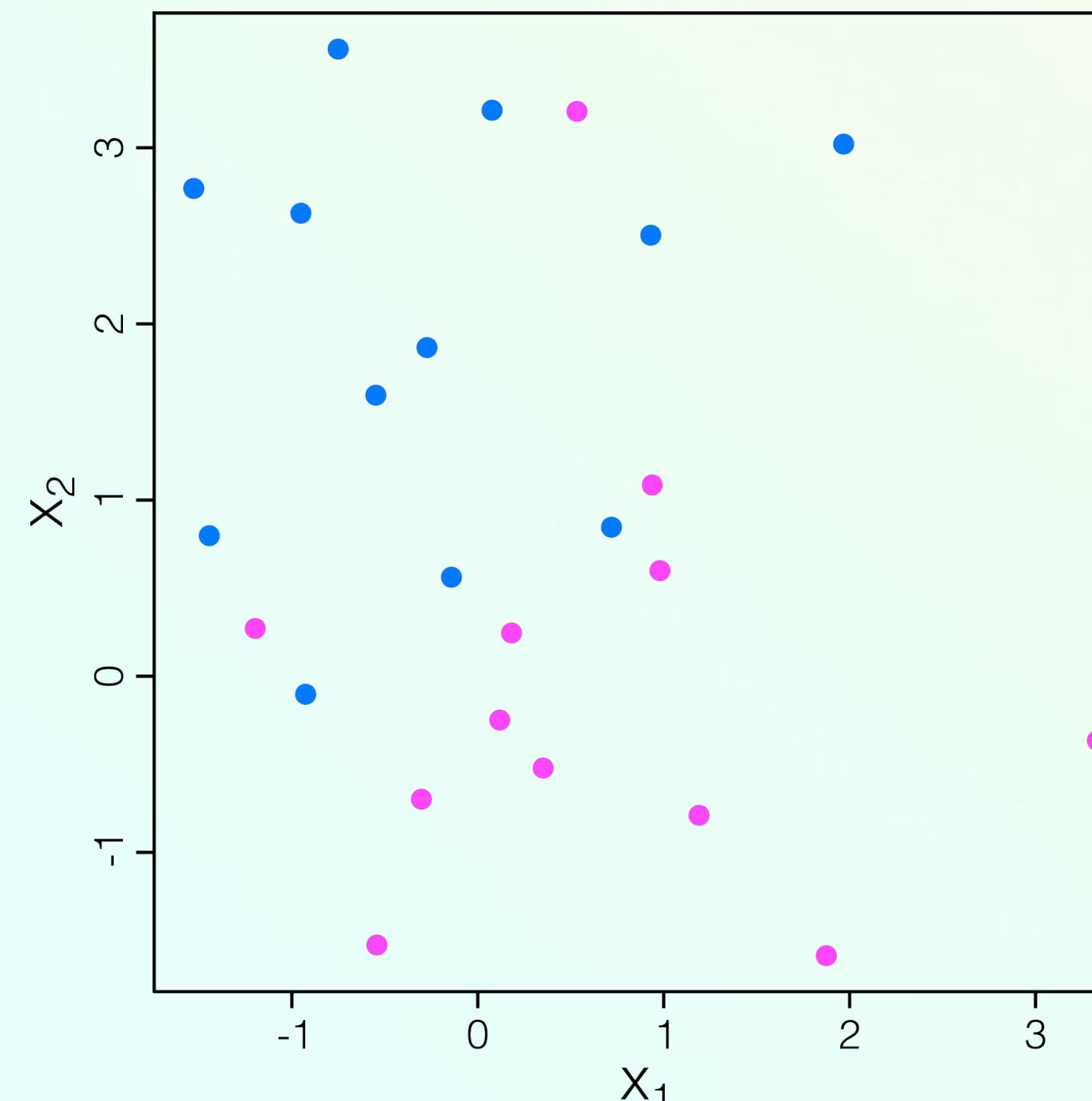
# LIMITACIONES DEL MAX-MARGIN CLASSIFIER

El modelo como está planteado es excesivamente sensible a set de entrenamiento, es decir tiene sobreajuste (error de varianza).



# LIMITACIONES DEL MAX-MARGIN CLASSIFIER

No solo eso. Este modelo que vimos solo funciona si es posible separar. Si el problema **no es separable**, no hay hiperplano que maximiza el margen. Y, por consiguiente, el problema de optimización tal cual lo planteamos no tiene solución..... **Necesitamos mejorar el modelo.**



# **CLASIFICADOR DE VECTOR DE SOPORTES (o *soft-margin SVN*)**

# CLASIFICADOR DE VECTOR DE SOPORTES

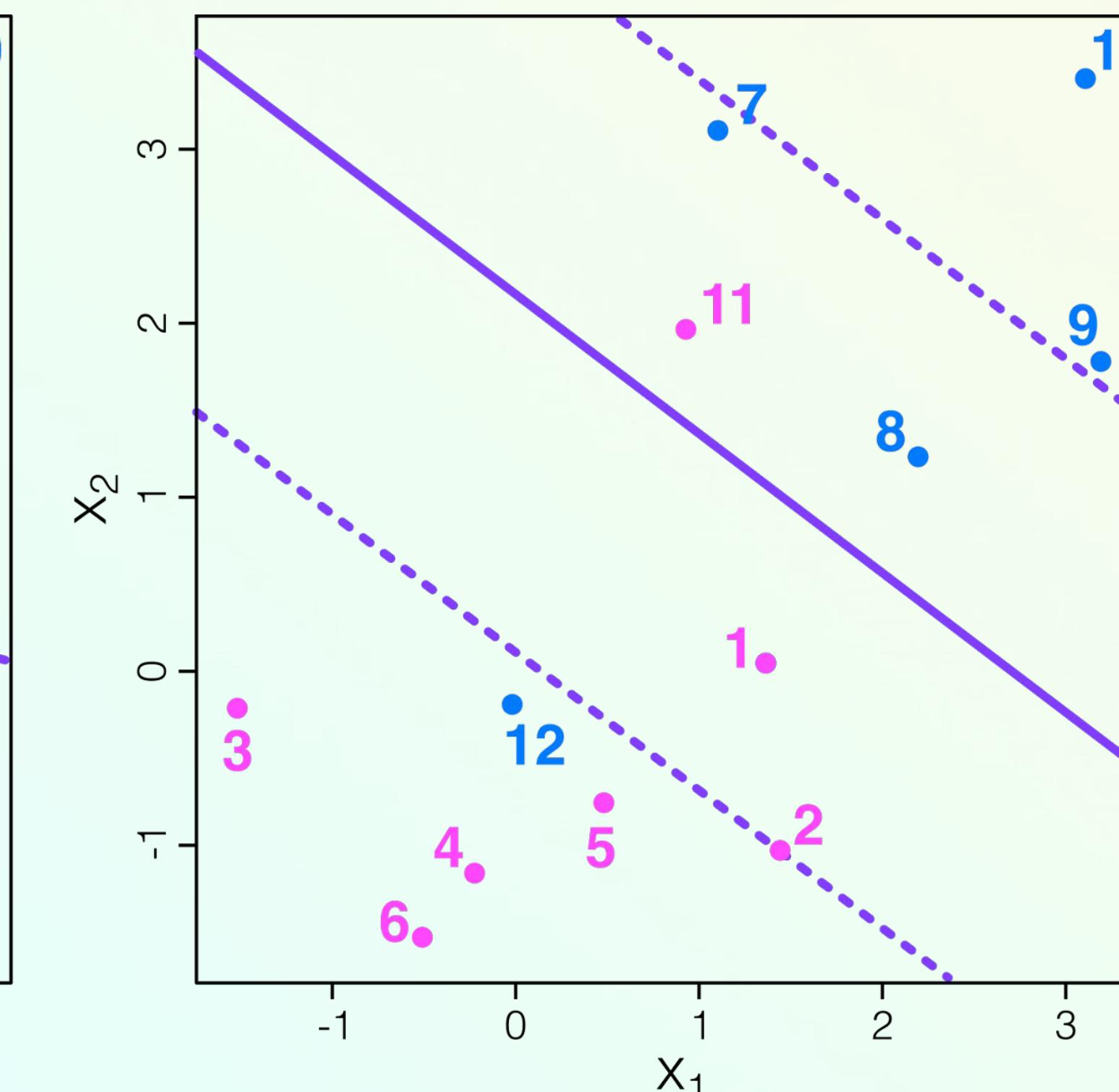
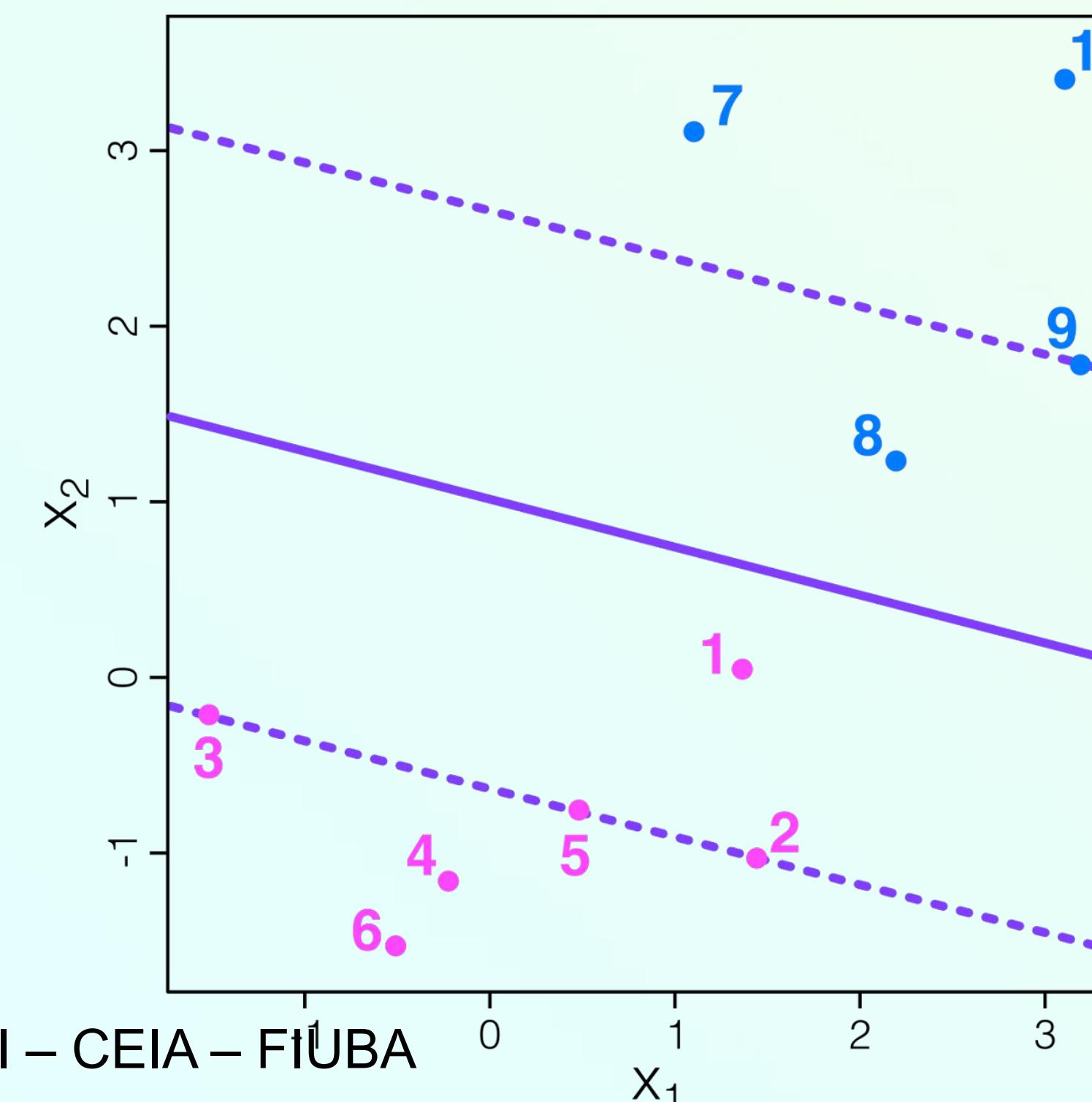
Por lo que vimos, si queremos seguir usando un hiperplano, debemos relajar las exigencias:

- Mayor robustez a observaciones individuales.
- Mejor clasificación de la **mayoría** (no todas) de las observaciones de entrenamiento.

Es decir, podría valer la pena clasificar **erróneamente algunas observaciones** de entrenamiento para poder clasificar mejor las observaciones restantes.

# CLASIFICADOR DE VECTOR DE SOPORTES

El modelo clasificador de vectores de soporte es el modelo apropiado para este caso. Con este modelo una observación puede estar no solo en el lado equivocado del **margen**, sino también en el **lado equivocado del hiperplano**.



# CLASIFICADOR DE VECTOR DE SOPORTES

Para construir este caso, se modifica el problema de optimización de recién, agregando **regularización**:

$$\begin{array}{ll} \text{maximizar } M & \\ \beta_0, \beta_1, \dots, \beta_p & \text{su}jeto \text{ a } \sum_{j=1}^r \beta_j^2 = 1 \end{array}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) \geq M(1 - \epsilon_i) \quad \forall i = 1, \dots, n$$

$$\epsilon_i \geq 0 \quad \sum_{i=1}^n \epsilon_i \leq \frac{1}{C} \quad C > 0$$

# CLASIFICADOR DE VECTOR DE SOPORTES

Para construir este caso, se modifica el problema de optimización de recién, agregando **regularización**:

$$\underset{\beta_0, \beta_1, \dots, \beta_p}{\text{maximizar}} M$$

$$\text{sujeito a } \sum_{j=1}^r \beta_j^2 = 1$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) \geq M(1 - \epsilon_i) \quad \forall i = 1, \dots, n$$

$$\epsilon_i \geq 0 \quad \sum_{i=1}^n \epsilon_i \leq \frac{1}{C} \quad C > 0$$



$\epsilon_i$  son variables que permiten que las observaciones individuales estén en el lado equivocado del margen o del hiperplano.

# CLASIFICADOR DE VECTOR DE SOPORTES

Para construir este caso, se modifica el problema de optimización de recién, agregando **regularización**:

$$\begin{array}{l} \text{maximizar } M \\ \beta_0, \beta_1, \dots, \beta_p \end{array}$$

$$\text{sujeato a } \sum_{j=1}^r \beta_j^2 = 1$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) \geq M(1 - \epsilon_i) \quad \forall i = 1, \dots, n$$

$$\epsilon_i \geq 0 \quad \sum_{i=1}^n \epsilon_i \leq \frac{1}{C} \quad C > 0$$

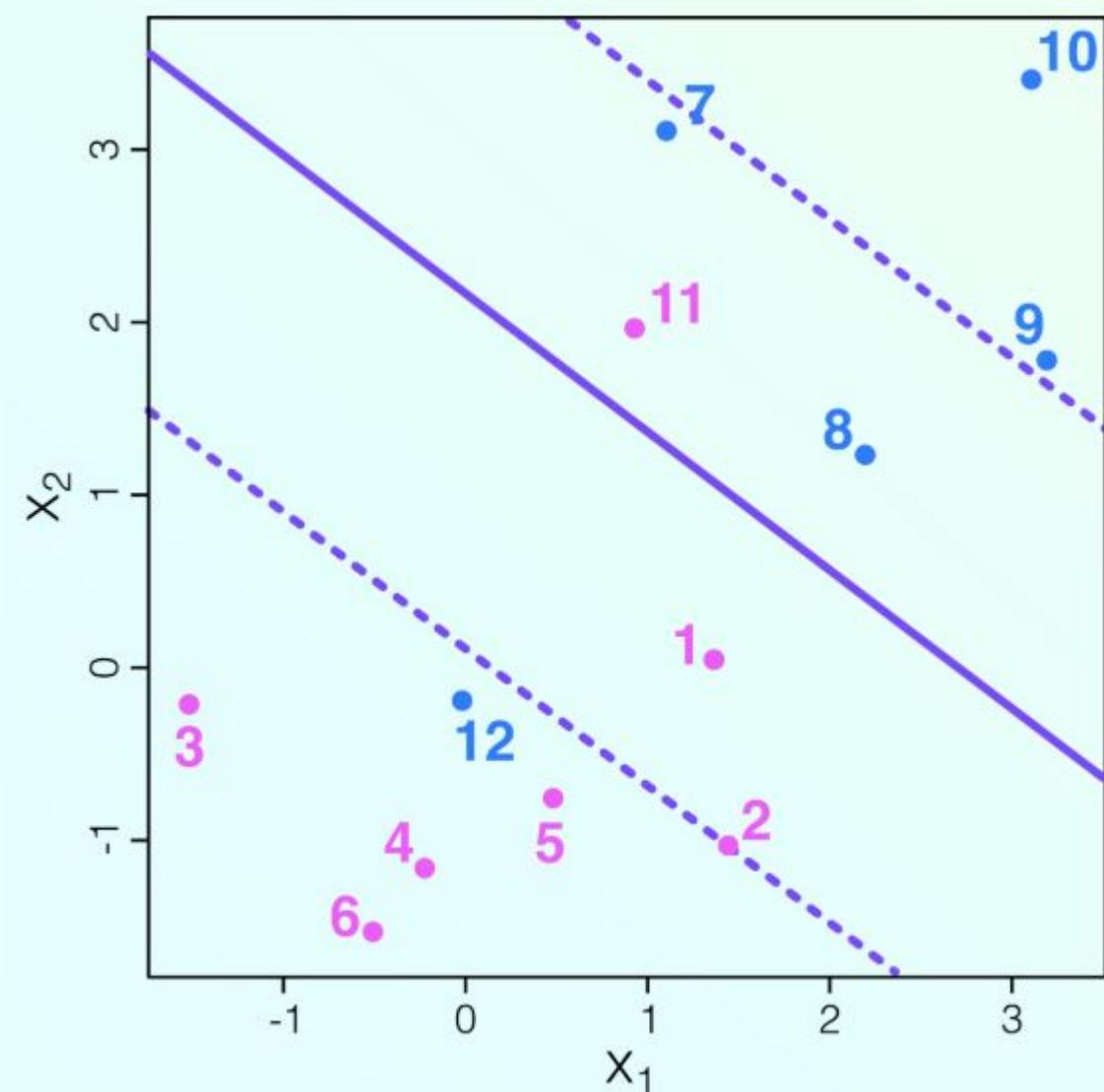
↓  
 $\epsilon_i$  son variables que permiten que las observaciones individuales estén en el lado equivocado del margen o del hiperplano.

→ C es un hiper-parámetro que pone un límite a los errores.

# CLASIFICADOR DE VECTOR DE SOPORTES

Los  $\epsilon_i$  permiten que las observaciones individuales estén en el lado equivocado del margen o del hiperplano.

$$y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) \geq M(1 - \epsilon_i) \quad \forall i = 1, \dots, n \quad \epsilon_i \geq 0 \quad \sum_{i=1}^n \epsilon_i \leq \frac{1}{C} \quad C > 0$$



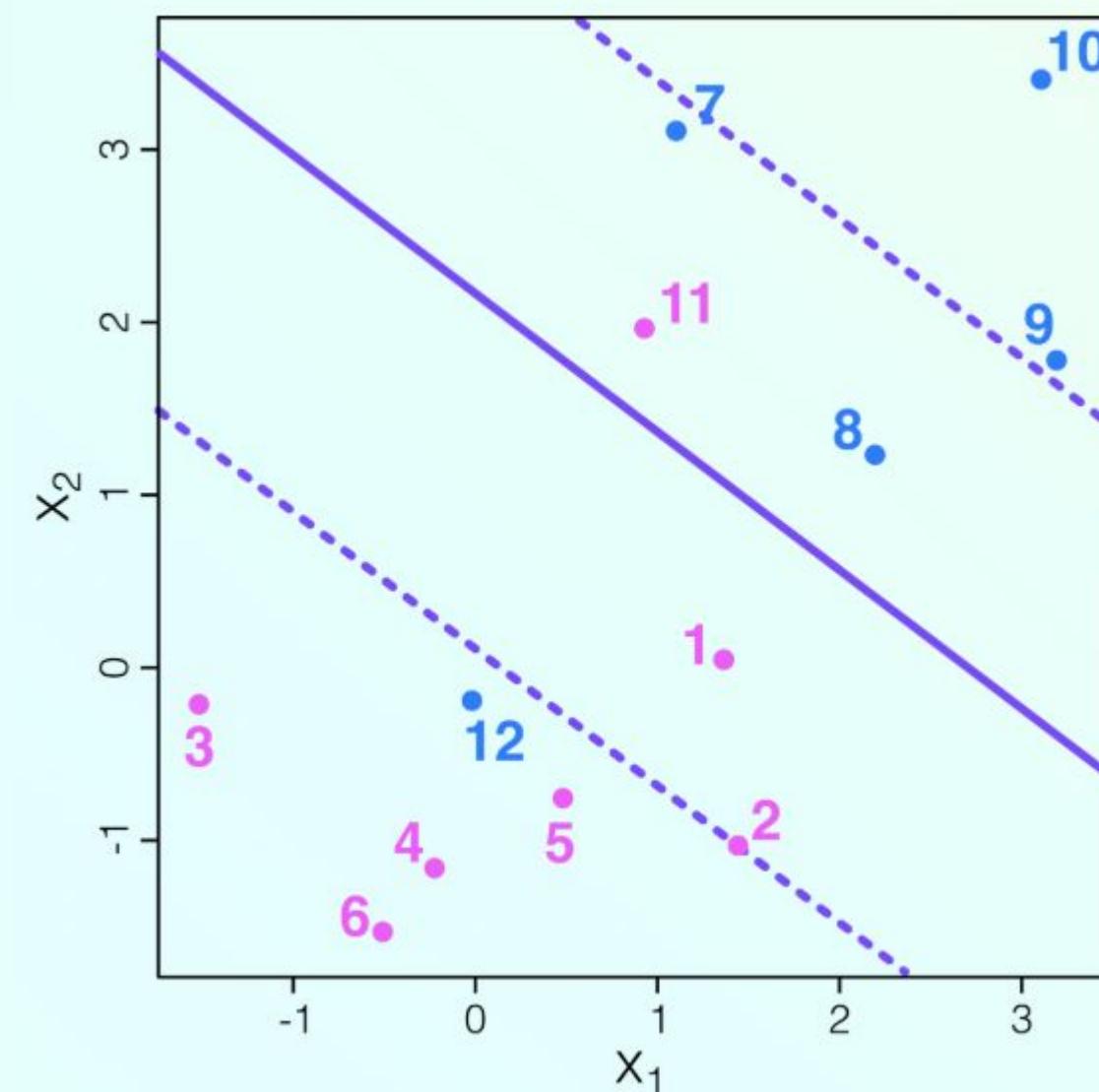
Dada una observación  $i$ :

- Si  $\epsilon_i = 0$ , la observación está *en el lado correcto*. (ej: 3, 10)
- Si  $0 < \epsilon_i < 1$ , la observación está *del lado correcto del hiperplano, pero en el lado equivocado del margen*. (ej: 1, 8)
- Si  $\epsilon_i > 1$ , la observación está *en el lado equivocado del hiperplano*. (ej: 11, 12)

# CLASIFICADOR DE VECTOR DE SOPORTES

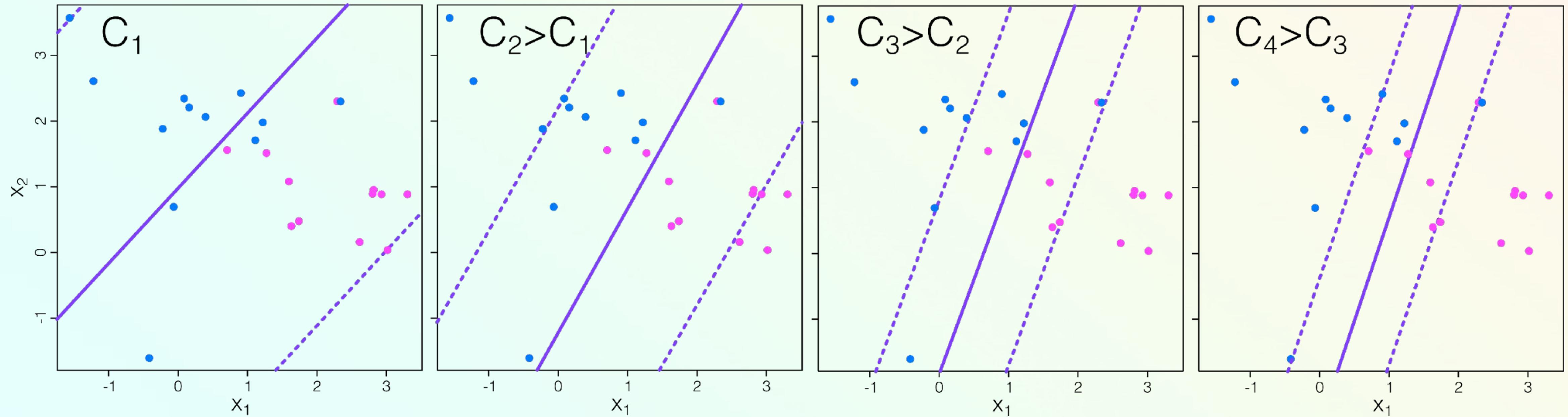
C es un hiper-parámetro que pone un límite a los errores.

$$y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) \geq M(1 - \epsilon_i) \quad \forall i = 1, \dots, n \quad \epsilon_i \geq 0 \quad \sum_{i=1}^n \epsilon_i \leq \frac{1}{C} \quad C > 0$$

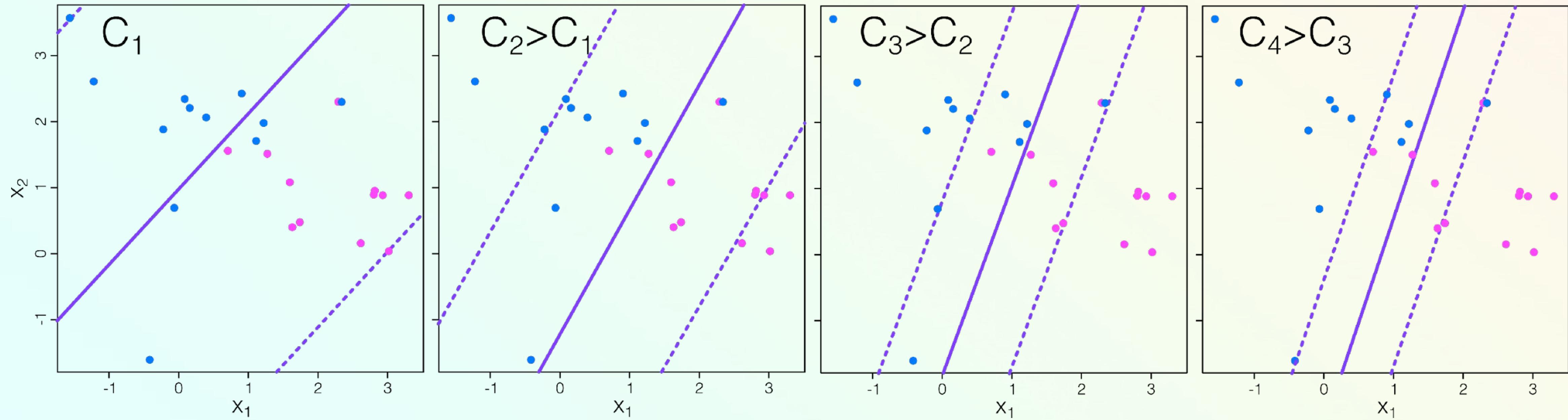


- C limita la suma de los  $\epsilon_i$  y, por tanto, determina el número y la gravedad de las violaciones del margen (y del hiperplano) que toleraremos.
- Si  $1/C > 0$ , no más que  $1/C$  observaciones pueden estar en el lado equivocado del hiperplano. Si C es más chico, más laxos son las exigencias y los márgenes serán más grandes.

# CLASIFICADOR DE VECTOR DE SOPORTES



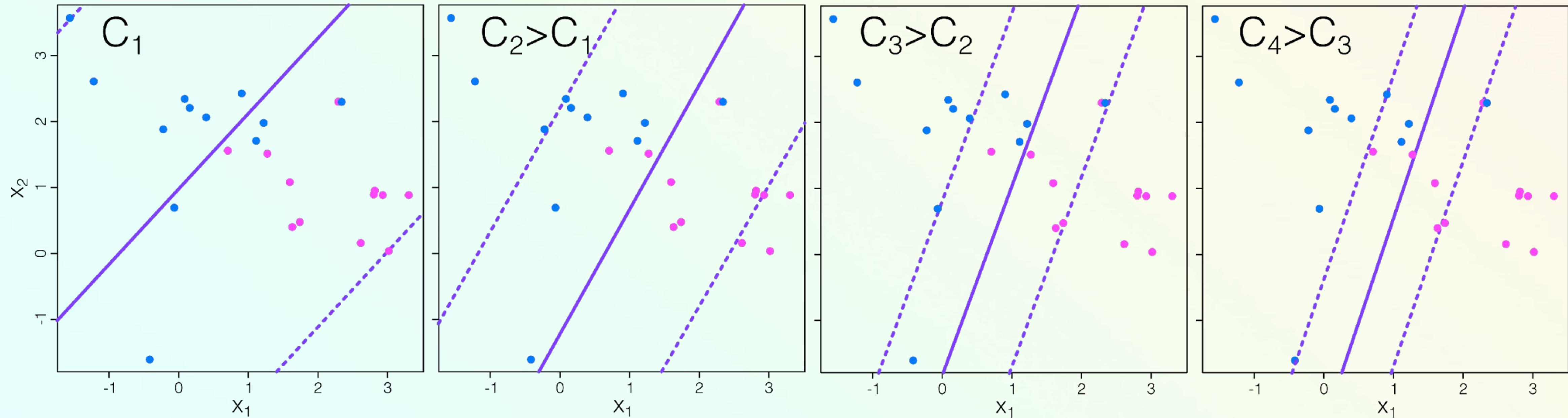
# CLASIFICADOR DE VECTOR DE SOPORTES



Hay un pequeño número de observaciones en el margen o dentro de él, que son los que determinan el hiperplano, esto se llaman vectores de soporte.

Las demás observaciones no tienen importancia para el modelo.

# CLASIFICADOR DE VECTOR DE SOPORTES



El valor de  $C$  nos da un trade-off (compensación) entre **sesgo** y **varianza**.

- Si  $C$  es chico, entonces el margen es grande y hay muchos vectores de soporte. Este clasificador tiene poca varianza, pero potencialmente poca exactitud.
- Si  $C$  es grande, hay menos vectores de soporte, y, por consiguiente, más varianza a expensas de una potencial mejor exactitud.

# CLASIFICADOR DE VECTOR DE SOPORTES

Podemos reescribir:

$$\begin{array}{ll} \text{maximizar}_M & \text{su}jeto\ a \sum_{j=1}^r \beta_j^2 = 1 \\ \beta_0, \beta_1, \dots, \beta_p & y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) \geq M(1 - \epsilon_i) \quad \forall i = 1, \dots, n \\ & \epsilon_i \geq 0 \quad \sum_{i=1}^n \epsilon_i \leq \frac{1}{C} \quad C > 0 \end{array}$$

usando el producto escalar:

$$\begin{array}{ll} \text{maximizar}_M & \text{su}jeto\ a \|\vec{\beta}\| = 1 \\ \beta_0, \beta_1, \dots, \beta_p & y_i(\beta_0 + \langle \vec{\beta}, X_i \rangle) \geq M(1 - \epsilon_i) \quad \forall i = 1, \dots, n \end{array}$$

# CLASIFICADOR DE VECTOR DE SOPORTES

Podemos reescribir:

$$\begin{array}{ll} \text{maximizar}_M & \text{su}jeto\ a \sum_{j=1}^r \beta_j^2 = 1 \\ \beta_0, \beta_1, \dots, \beta_p & y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) \geq M(1 - \epsilon_i) \quad \forall i = 1, \dots, n \\ & \epsilon_i \geq 0 \quad \sum_{i=1}^n \epsilon_i \leq \frac{1}{C} \quad C > 0 \end{array}$$

usando el producto escalar:

$$\begin{array}{ll} \text{maximizar}_M & \text{su}jeto\ a \|\vec{\beta}\| = 1 \\ \beta_0, \beta_1, \dots, \beta_p & y_i(\beta_0 + \langle \vec{\beta}, X_i \rangle) \geq M(1 - \epsilon_i) \quad \forall i = 1, \dots, n \end{array}$$

Si quitamos la restricción del versor y  
definimos  $M = 1/\|\vec{\beta}\|$

Entonces debemos modificar la medida de  
distancia en forma acorde.

# CLASIFICADOR DE VECTOR DE SOPORTES

Podemos reescribir:

$$\begin{array}{ll} \text{maximizar } M & \text{sujeto a } \sum_{j=1}^r \beta_j^2 = 1 \\ \beta_0, \beta_1, \dots, \beta_p & y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) \geq M(1 - \epsilon_i) \quad \forall i = 1, \dots, n \\ & \epsilon_i \geq 0 \quad \sum_{i=1}^n \epsilon_i \leq \frac{1}{C} \quad C > 0 \end{array}$$

usando el producto escalar:

$$\begin{array}{ll} \text{maximizar } M & \text{sujeto a } \|\vec{\beta}\| = 1 \\ \beta_0, \beta_1, \dots, \beta_p & y_i(\beta_0 + \langle \vec{\beta}, X_i \rangle) \geq M(1 - \epsilon_i) \quad \forall i = 1, \dots, n \end{array}$$

Si quitamos la restricción del versor y definimos  $M = 1/\|\vec{\beta}\|$

Entonces debemos modificar la medida de distancia en forma acorde.

$$\frac{y_i(\beta_0 + \langle \vec{\beta}, X_i \rangle)}{\|\vec{\beta}\|} \geq \frac{1}{\|\vec{\beta}\|} (1 - \epsilon_i) \quad \forall i = 1, \dots, n$$

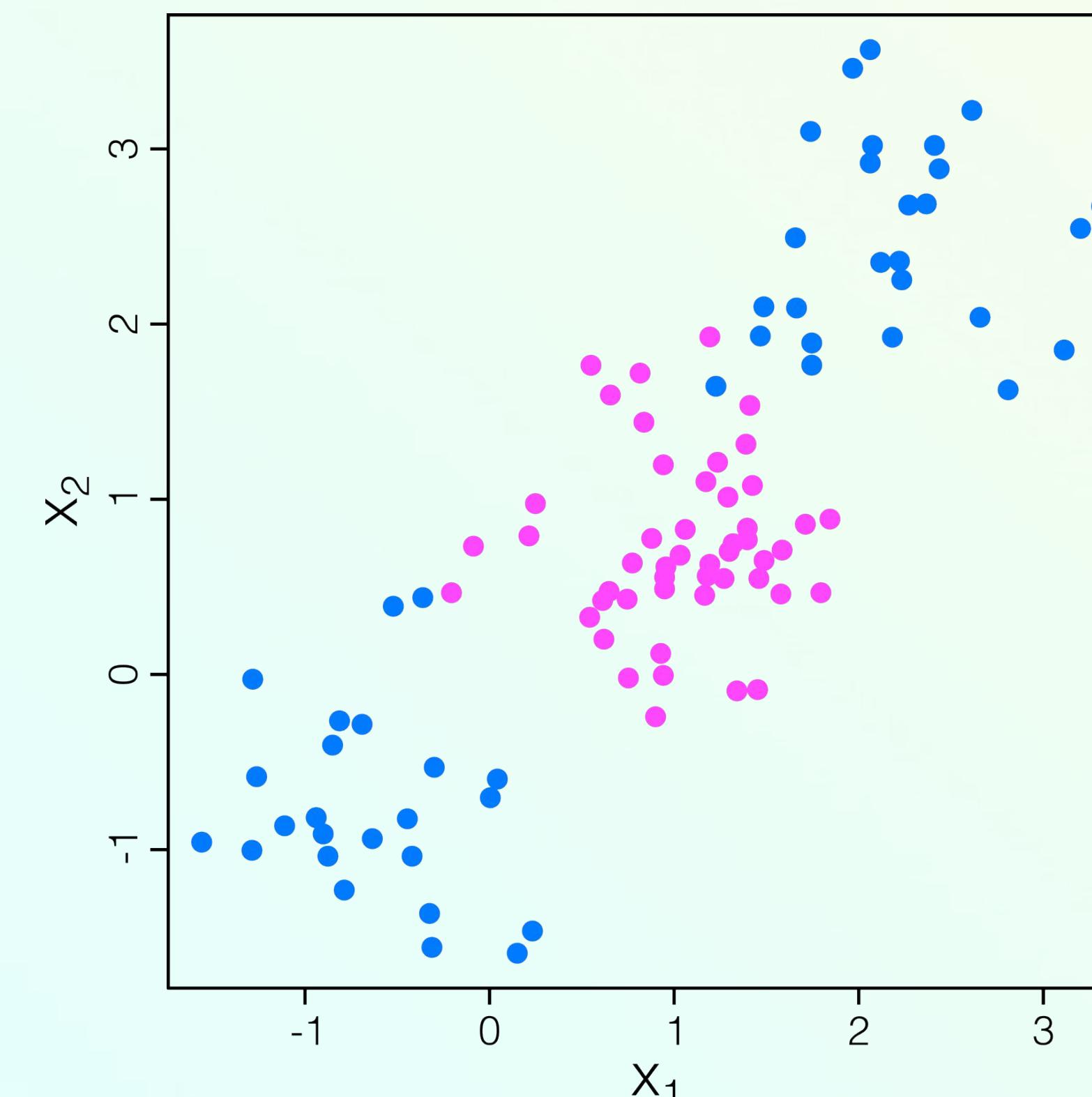
$$y_i(\beta_0 + \langle \vec{\beta}, X_i \rangle) \geq (1 - \epsilon_i) \quad \forall i = 1, \dots, n$$

*VAMOS A PRACTICAR UN POCO...*

# ***SUPPORT VECTOR MACHINE***

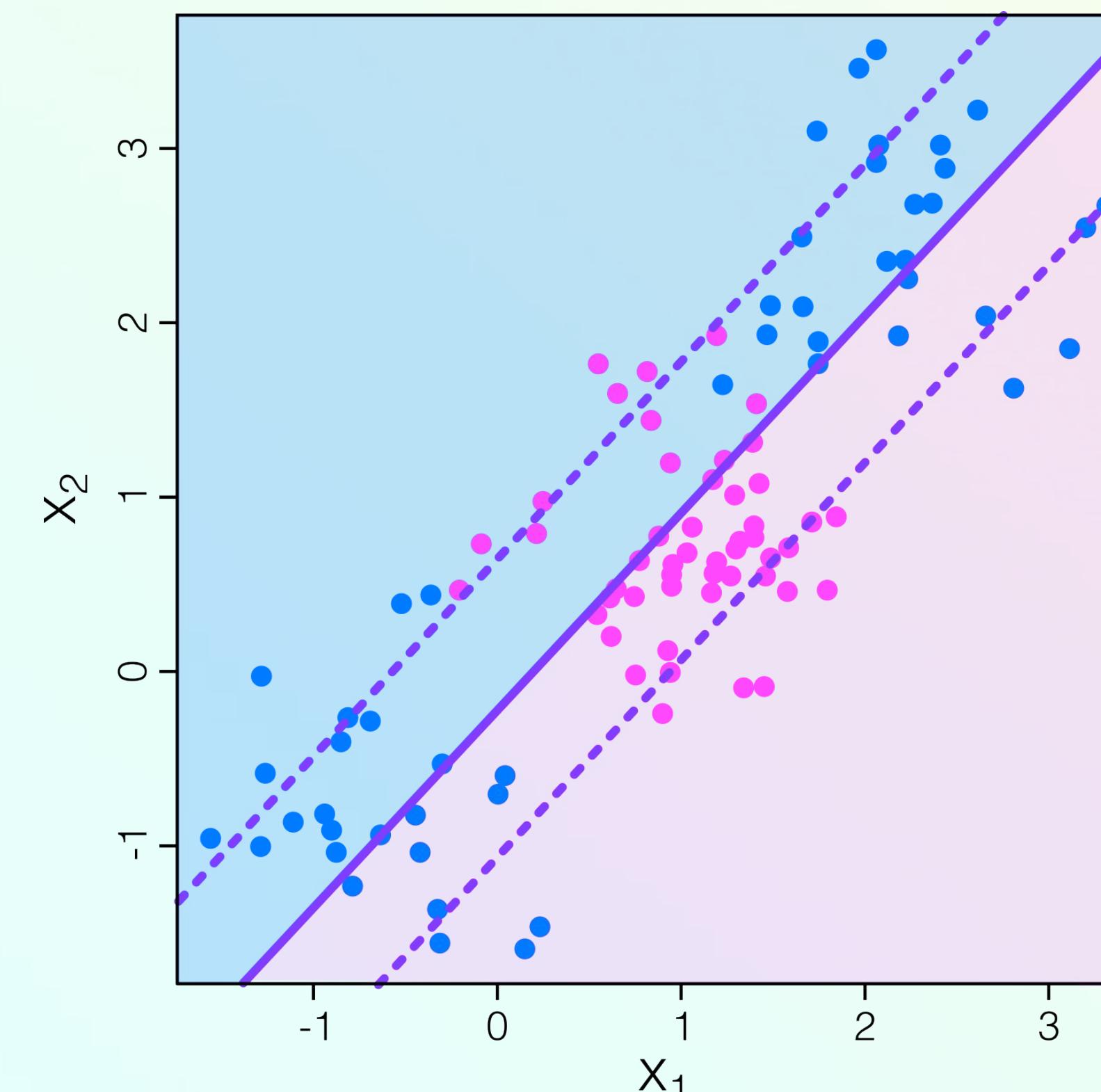
# SUPPORT VECTOR MACHINE

El clasificador que hemos visto hasta ahora solo funciona para casos de separación lineal, pero en casos de fronteras no lineales es completamente inútil.



# SUPPORT VECTOR MACHINE

El clasificador que hemos visto hasta ahora solo funciona para casos de separación lineal, pero en casos de fronteras no lineales es completamente inútil.



# SUPPORT VECTOR MACHINE

Podríamos, similar al caso de regresión lineal y polinomial, modificar a nuestros predictores, agregando la versión cuadrática, cúbica, etc.

Es decir, en vez de para cada observación entrenar con los atributos  $x_1, x_2, \dots, x_p$

Entrenamos con  $x_1, x_1^2, x_2, x_2^2, x_2^3, \dots, x_p, x_p^2$

El modelo será lineal en el espacio extendido, pero cuando volvamos al espacio dado por  $x_1, x_2, \dots, x_p$ , la frontera de decisión será polinómica.

Esto lo podríamos llevar a otro tipo de relaciones, así podemos usar otro tipo de fronteras, pero el problema es que de ajuste se vuelve **inmanejable**.

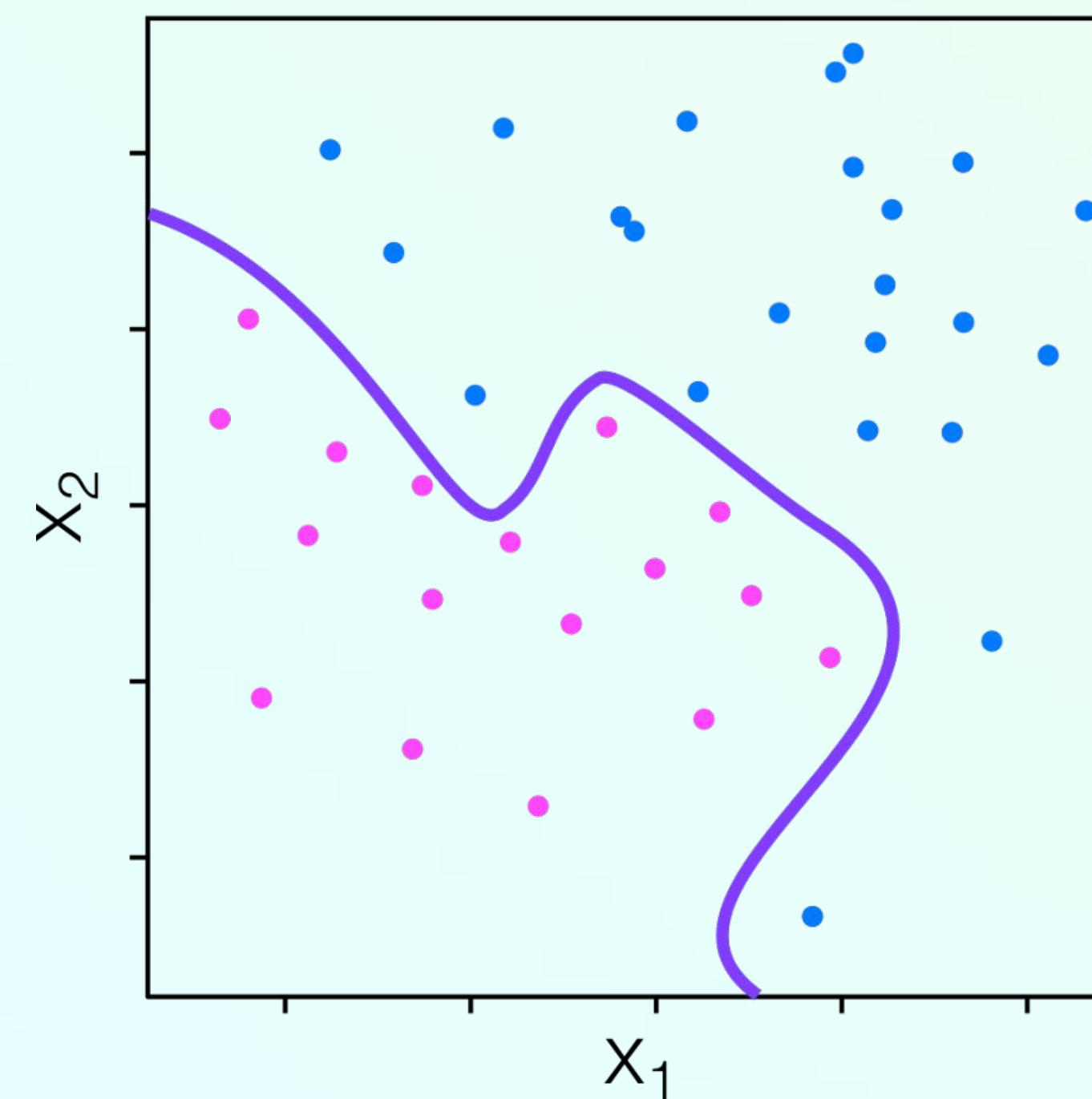
# SUPPORT VECTOR MACHINE

Por eso existe el modelo llamado **Support Vector Machine (SVM)** o máquina de vector de soportes que extiende el Clasificador de Vector de Soportes permitiendo ampliar el espacio de atributos, usando funciones *kernels*.

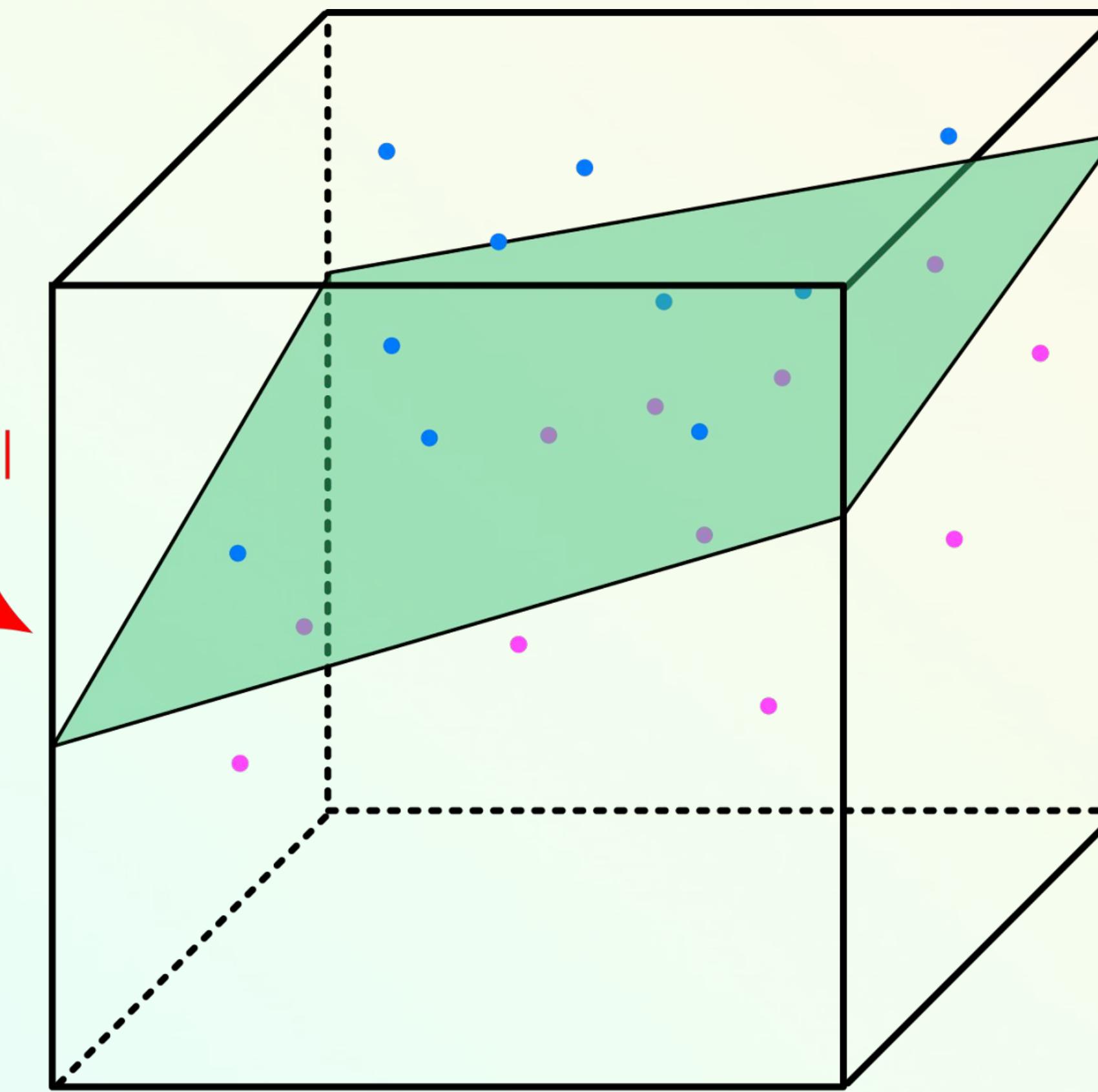
Este tipo de extensión es eficiente computacionalmente.

Veamos cómo lo hace...

# SUPPORT VECTOR MACHINE



Función Kernel



# SUPPORT VECTOR MACHINE

La función que usábamos para clasificar usando el **clasificador de vector de soportes** era:

$$f(X) = \beta_0 + \langle \vec{\beta}, X \rangle$$

Se puede trabajar para llegar a la siguiente expresión:

$$f(X) = \beta_0 + \sum_{i=1}^n \alpha_i \langle X_i, X \rangle$$

- $\langle X_i, X \rangle$  es el producto interno (o producto escalar) entre la observación de entrada y la i-esima observación de entrenamiento.
- Hay n parámetros  $\alpha_i$ , uno por observación de entrenamiento.
- El entrenar, para estimar los parámetros  $\alpha_1, \alpha_2, \dots, \alpha_n$ , y  $\beta_0$ , se necesita  $\binom{n}{2}$  productos internos entre todos los pares de entrenamiento.

# SUPPORT VECTOR MACHINE

$$f(X) = \beta_0 + \sum_{i=1}^n \alpha_i \langle X_i, X \rangle$$

Esta función para predecir necesita de todas las observaciones del set de entrenamiento.

Sin embargo, resulta que  $\alpha_i$  es **cero para aquellas observaciones que no son vectores de soporte**.  
De tal forma que realmente la función se puede reescribir:

$$f(X) = \beta_0 + \sum_{i \in \mathcal{H}} \alpha_i \langle X_i, X \rangle$$

Que normalmente involucra muchos menos términos.

Es decir, para usar el modelo de clasificador de vector de soportes y computar sus coeficientes, todo lo que se necesita es el **producto interno**.

# SUPPORT VECTOR MACHINE

Ahora, podemos reemplazar al producto interno por una generalización del producto interno:

$$K(X_i, X_{i'})$$

donde K es una función que llamamos **función kernel**. Un kernel o núcleo es una función que cuantifica la similitud entre dos vectores. Por ejemplo, si tenemos:

$$K(X_i, X_{i'}) = \sum_{j=1}^p x_{ij} x_{i'j}$$

Volvemos al clasificador de vector de soportes con frontera lineal. Este kernel lineal en esencia está midiendo la similitud de las observaciones mediante una correlación de Pearson.

# SUPPORT VECTOR MACHINE

Pero también podemos obtener un kernel polinomial de grado d ( $d > 1$ ):

$$K(X_i, X_{i'}) = \left( 1 + \sum_{j=1}^p x_{ij} x_{i'j} \right)^d$$

Lo que nos generará una frontera de decisión polinomial.

Otro kernel popular es kernel radial definido como:

$$K(X_i, X_{i'}) = \exp \left( -\gamma \sum_{j=1}^p (x_{ij} - x_{i'j})^2 \right)$$

Donde  $\gamma$  es una constante positiva. Este kernel nos permite obtener fronteras radiales. Este kernel se basa en la **distancia euclíadiana**, con un efecto muy local.

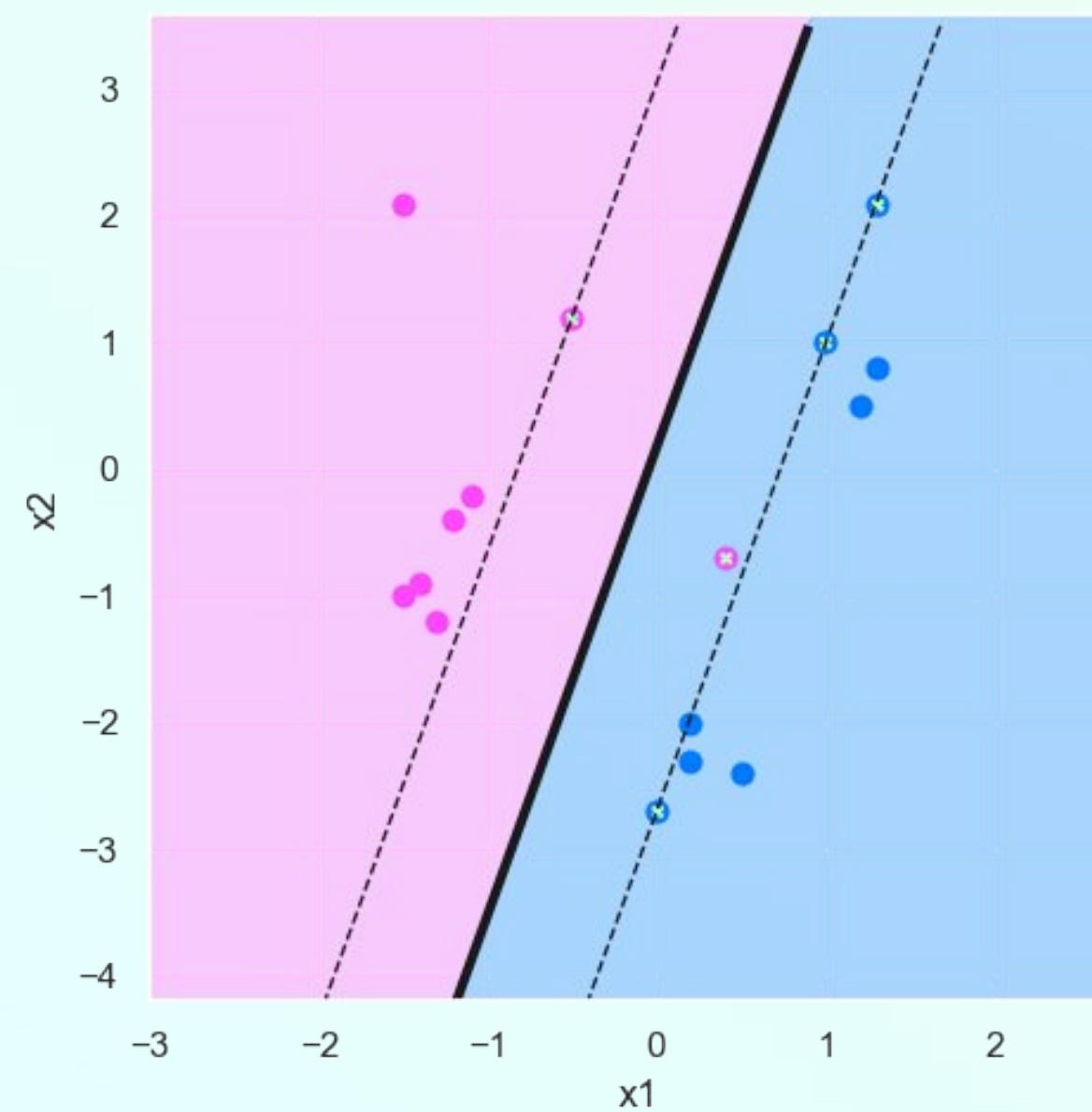
# SUPPORT VECTOR MACHINE

Por lo que podemos escribir a la función de decisión de la **máquina de vector de soportes** como:

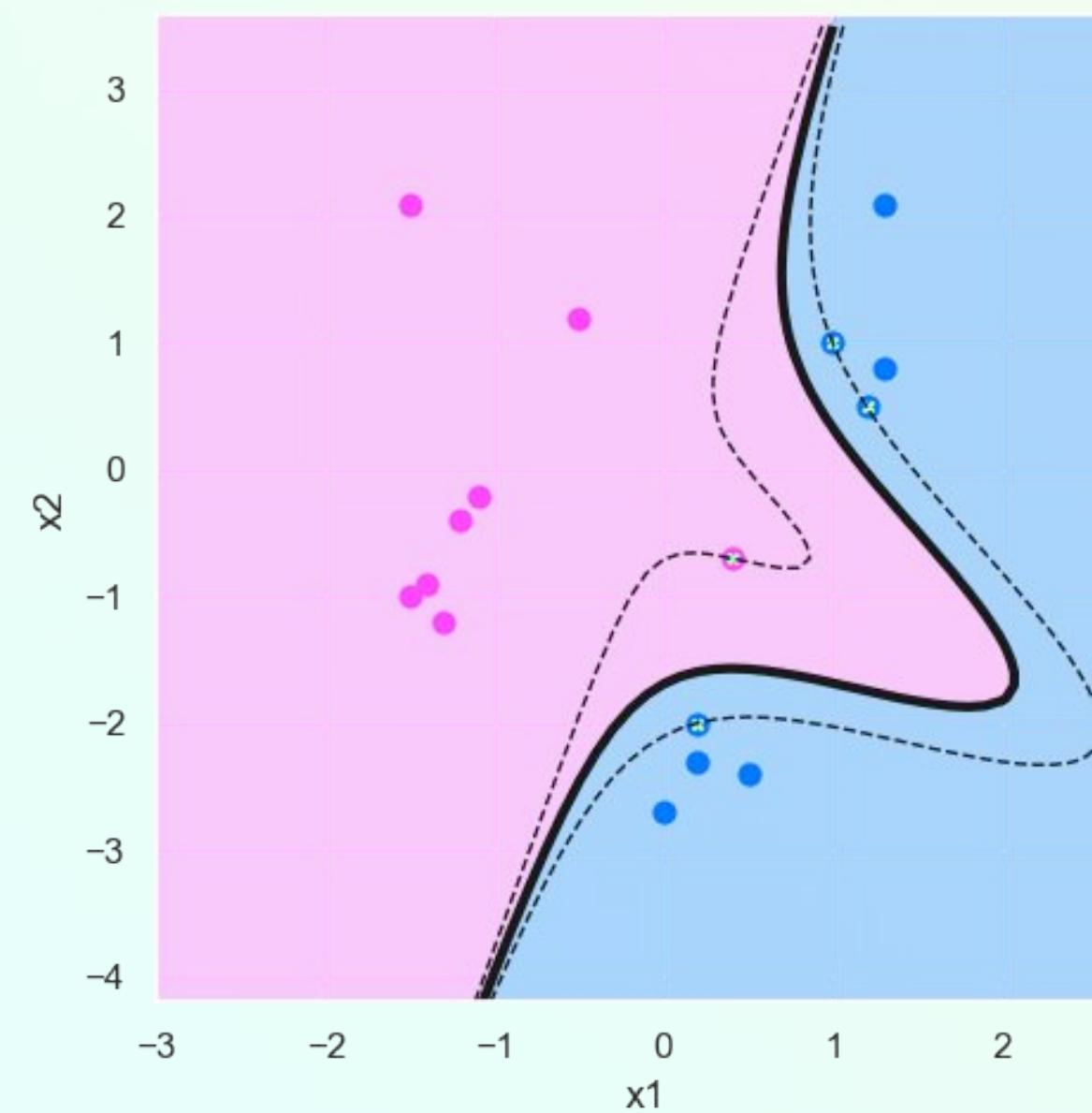
$$f(X) = \beta_0 + \sum_{i \in \mathcal{H}} \alpha_i K(X_i, X)$$

En el entrenamiento, la importancia de los vectores de soporte y el hiperparámetro C se mantienen. La gran diferencia es que ahora las fronteras de decisión no son necesariamente lineales y determinada por la función **kernel elegida**.

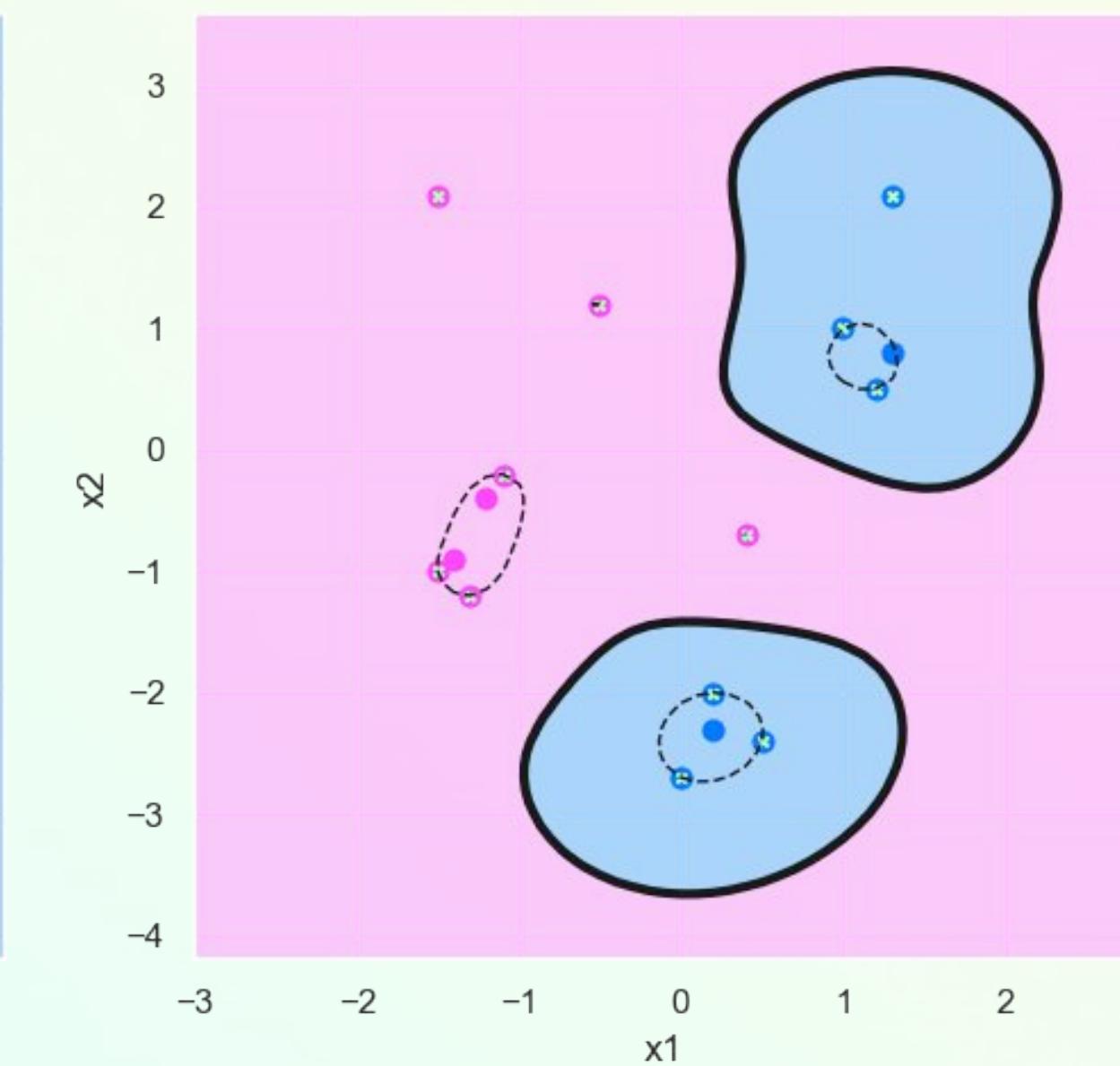
# SUPPORT VECTOR MACHINE



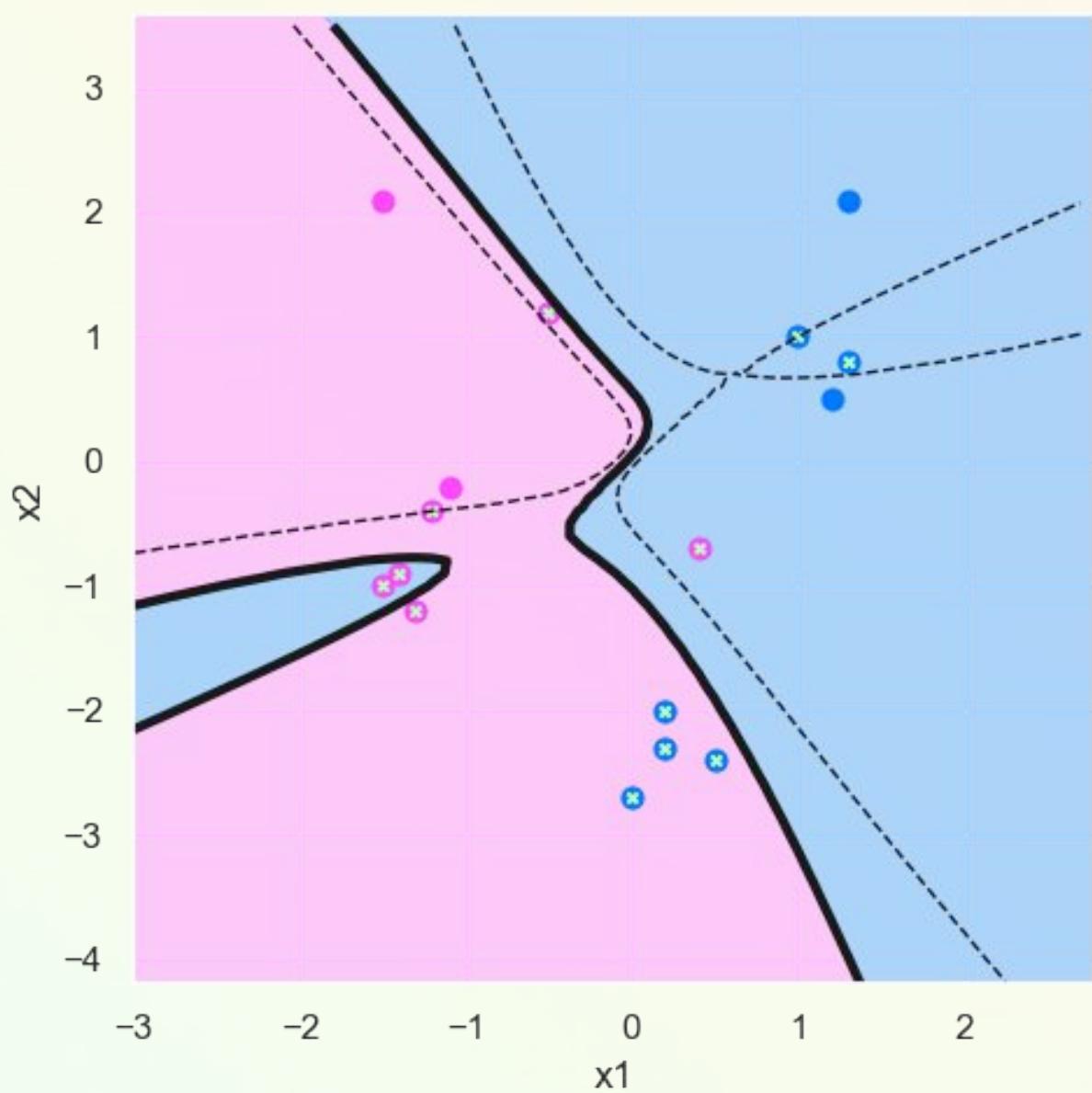
Lineales



Polinomial



Radial



Sigmoidea

# SUPPORT VECTOR MACHINE

Este clasificador que estuvimos viendo es binario. Ese es su punto fuerte y está preparado para clasificar entre dos clases.

Cómo hacemos para clasificar con más de dos clases:

- One-vs-one: En este caso es comparar a cada clase con otra, construyendo un SVM para cada par posible de clases. Una vez que tenemos entrenados los SVMs, la clasificación final de una predicción en una clase particular es dada por la clase que más votada por los clasificadores.
- One-vs-all: En este caso comparamos una clase contra el resto de las clases. Por lo que tendremos  $K$  SVMs, cada uno entrenado para comparar una clase en particular contra las restantes como un solo conjunto. Se asigna la clase a una predicción para el modelo que predice la clase con mayor confidencia usando los  $f(X)$  de los SVMs.

*VAMOS A PRACTICAR UN POCO...*

# ***SUPPORT VECTOR MACHINE EN REGRESIÓN***

# SUPPORT VECTOR MACHINE EN REGRESIÓN

Hasta ahora vimos a los SVM como un modelo de clasificación, y son más popular como modelo de clasificación, pero podemos usar esta idea para realizar regresiones.

Con solo cambiar la restricción, entendiendo que ahora tenemos que predecir un target continuo.

Si cambiamos la restricción del **Maximal Margin Classifier**:

$$y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) \geq M \quad \forall i = 1, 2, \dots, n$$

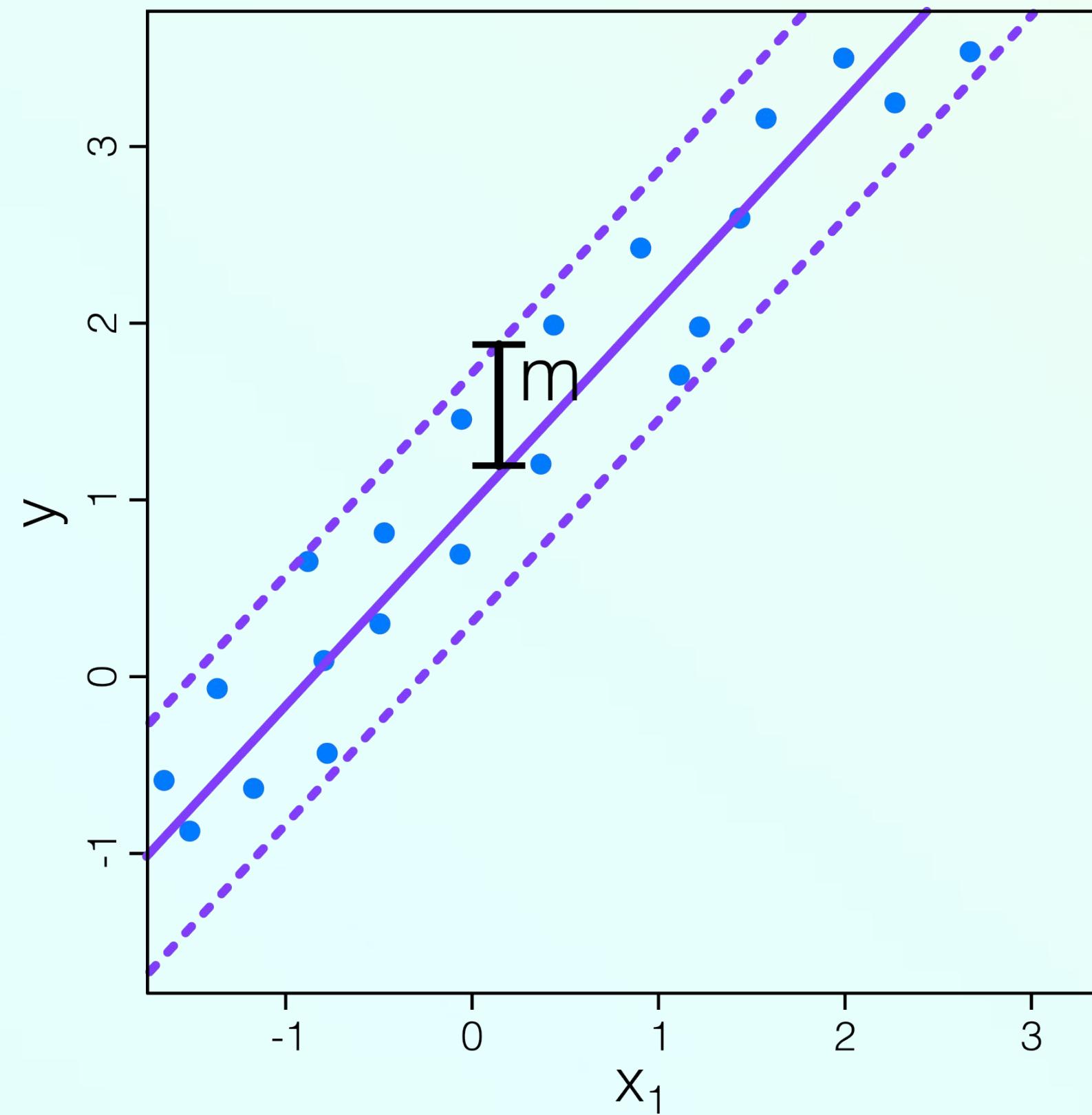
por

$$-m \leq b_0 + b_1 x_{i1} + \dots + b_p x_{ip} \leq m \quad \forall i = 1, 2, \dots, n$$

podemos realizar regresiones.

# SUPPORT VECTOR MACHINE EN REGRESIÓN

Lo que se optimiza ahora es el hiperplano que mejor que logra meter a todos los puntos de entrenamiento dentro del margen, minimizando el valor del margen.

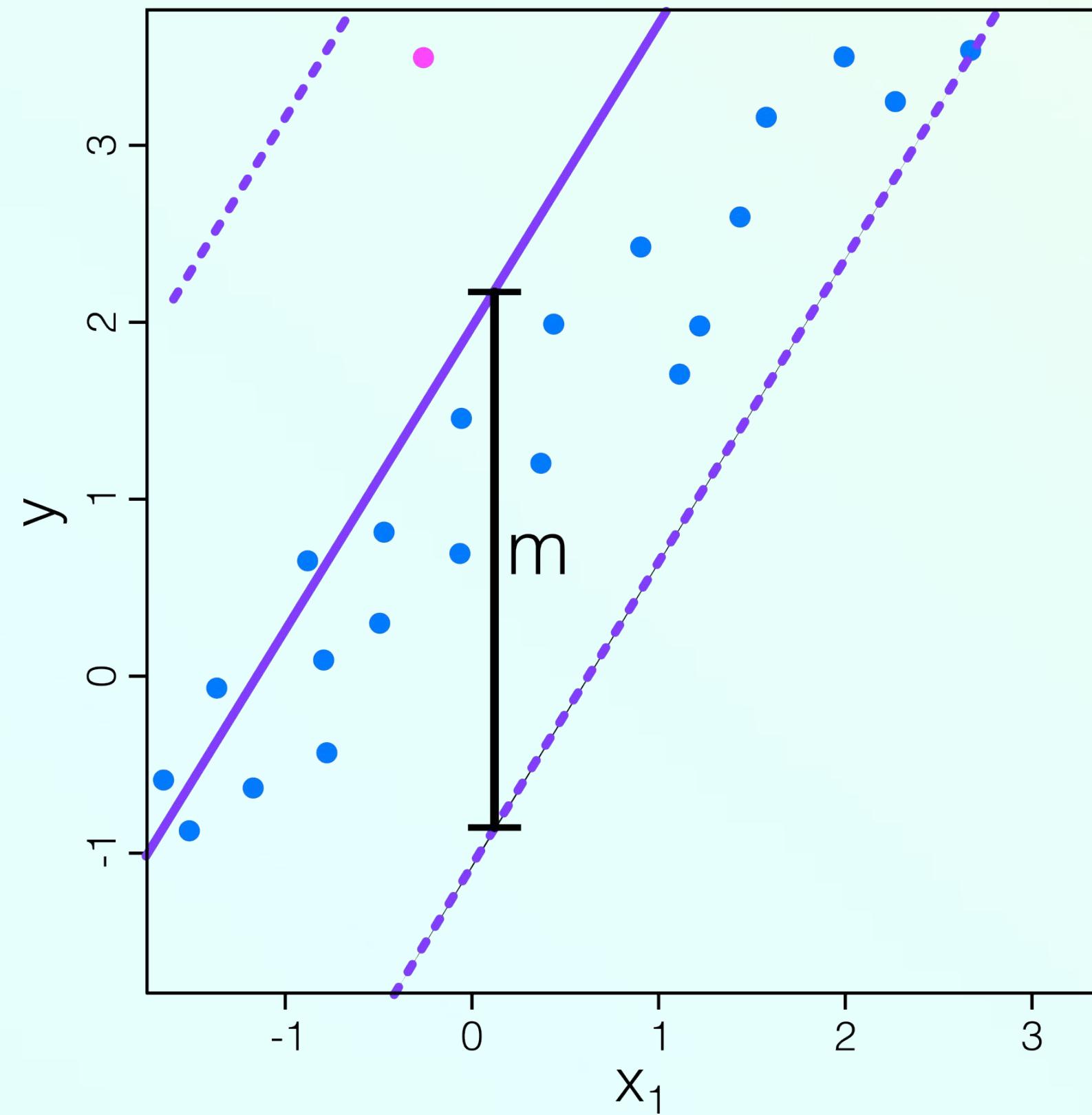


Entonces una vez que se tiene el modelo entrenado, se puede usar para regresión con la fórmula:

$$f(X) = b_0 + b_1 x_1 + \dots + b_p x_p$$

# SUPPORT VECTOR MACHINE EN REGRESIÓN

Lo que se optimiza ahora es el hiperplano que mejor que logra meter a todos los puntos de entrenamiento dentro del margen, minimizando el valor del margen.



Entonces una vez que se tiene el modelo entrenado, se puede usar para regresión con la fórmula:

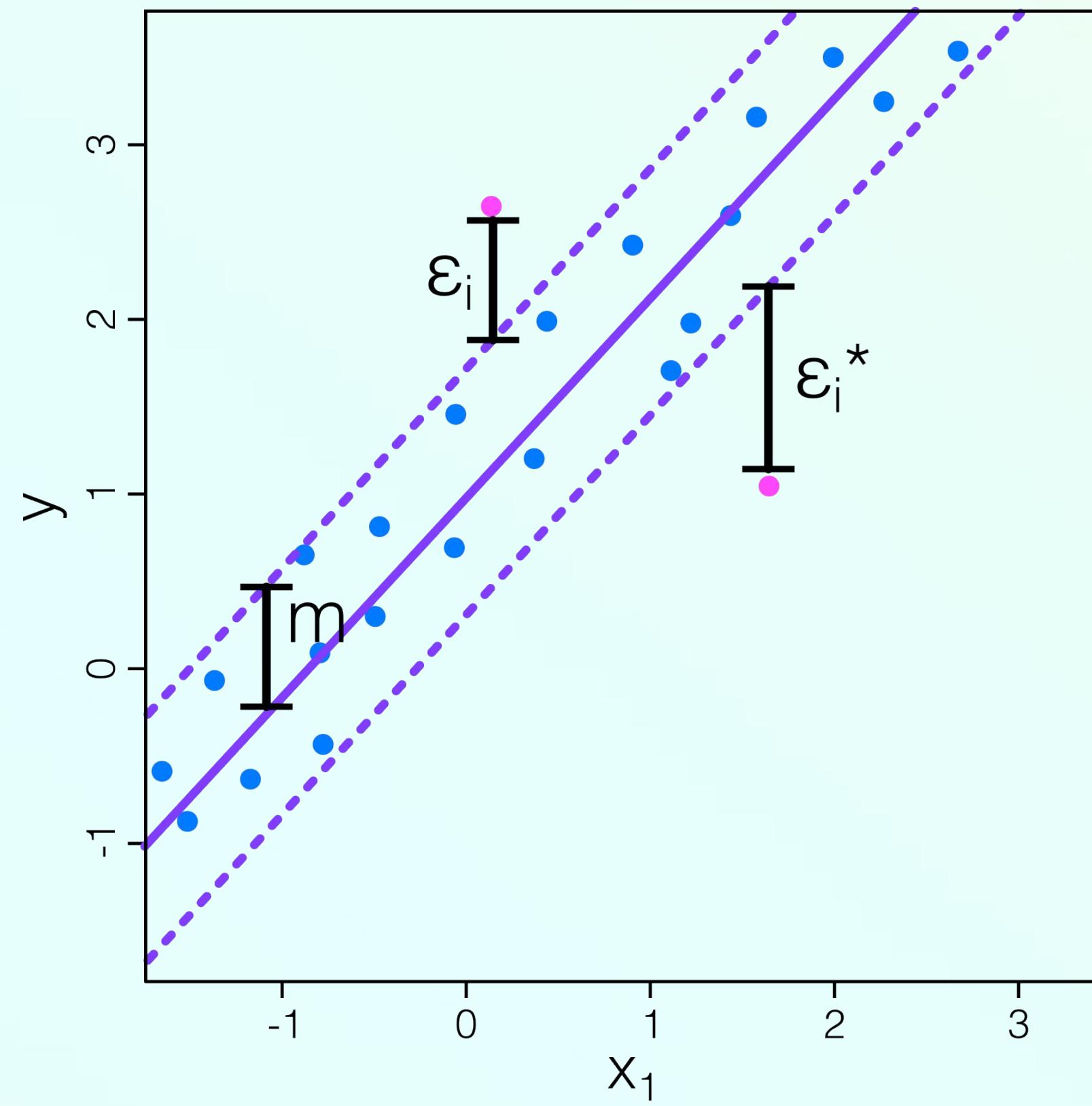
$$f(X) = b_0 + b_1x_1 + \dots + b_px_p$$

Igual que en el caso de clasificación, una observación de entrenamiento muy alejada del resto puede cambiar radicalmente a la regresión.

Por lo que se debe incorporar la misma idea de permitir que algunas observaciones puedan estar por fuera.

# SUPPORT VECTOR MACHINE EN REGRESIÓN

Entonces para evitar esto, agregamos un hiper-parámetro que nos permite tener algunas observaciones fuera de la zona de margen.



$$\sum_{i=1}^n (\epsilon_i + \epsilon_i^*) \leq \frac{1}{C} \quad \epsilon_i, \epsilon_i^* \geq 0 \quad \forall i = 1, \dots, n$$

Y la restricción es:

$$y_i - (b_0 + \langle \vec{b}, X \rangle) \leq m + \epsilon_i \quad \forall i = 1, \dots, n$$
$$(b_0 + \langle \vec{b}, X \rangle) - y_i \leq m + \epsilon_i^*$$

# SUPPORT VECTOR MACHINE EN REGRESIÓN

Y podemos extender a usar funciones kernels para extender a casos no lineales, extendiendo a un espacio dimensional mayor.

*VAMOS A PRACTICAR UN POCO...*