Métodos No Lineales

Máquinas de Soporte Vectorial - SVM

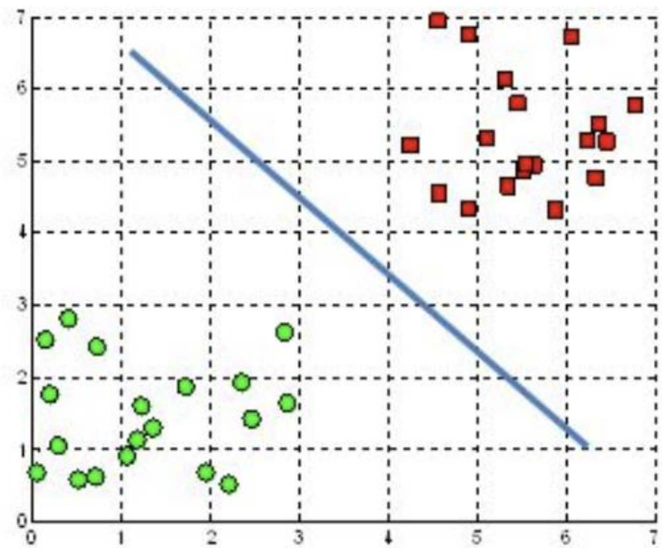
Universidad Católica del Uruguay

Support Vector Machines (SVM)

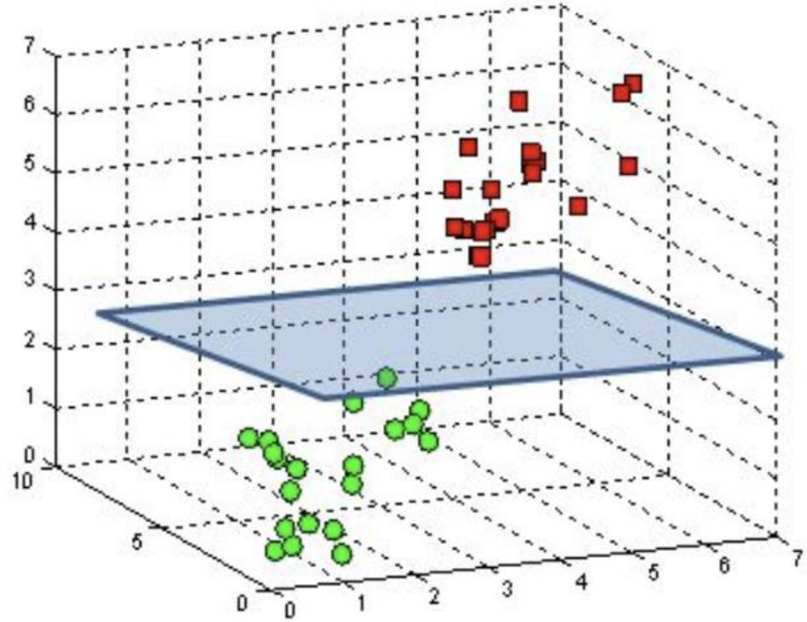
- Una muestra (o ejemplo) se describe como un vector de dimensión *p*. Nuestro objetivo será determinar un hiperplano que logre separar a todos los puntos de nuestro dataset en -en principio- dos clases.
- Luego, cuando se requiere determinar la clase de una nueva muestra simplemente se observa de qué lado queda en la separación indicada por el hiperplano.

Hiperplano de separación

A hyperplane in $\mathbb{R}^2$ is a line



A hyperplane in $\mathbb{R}^3$ is a plane



Maximal-Margin Classifier

- Caso donde las muestras son perfectamente separables:

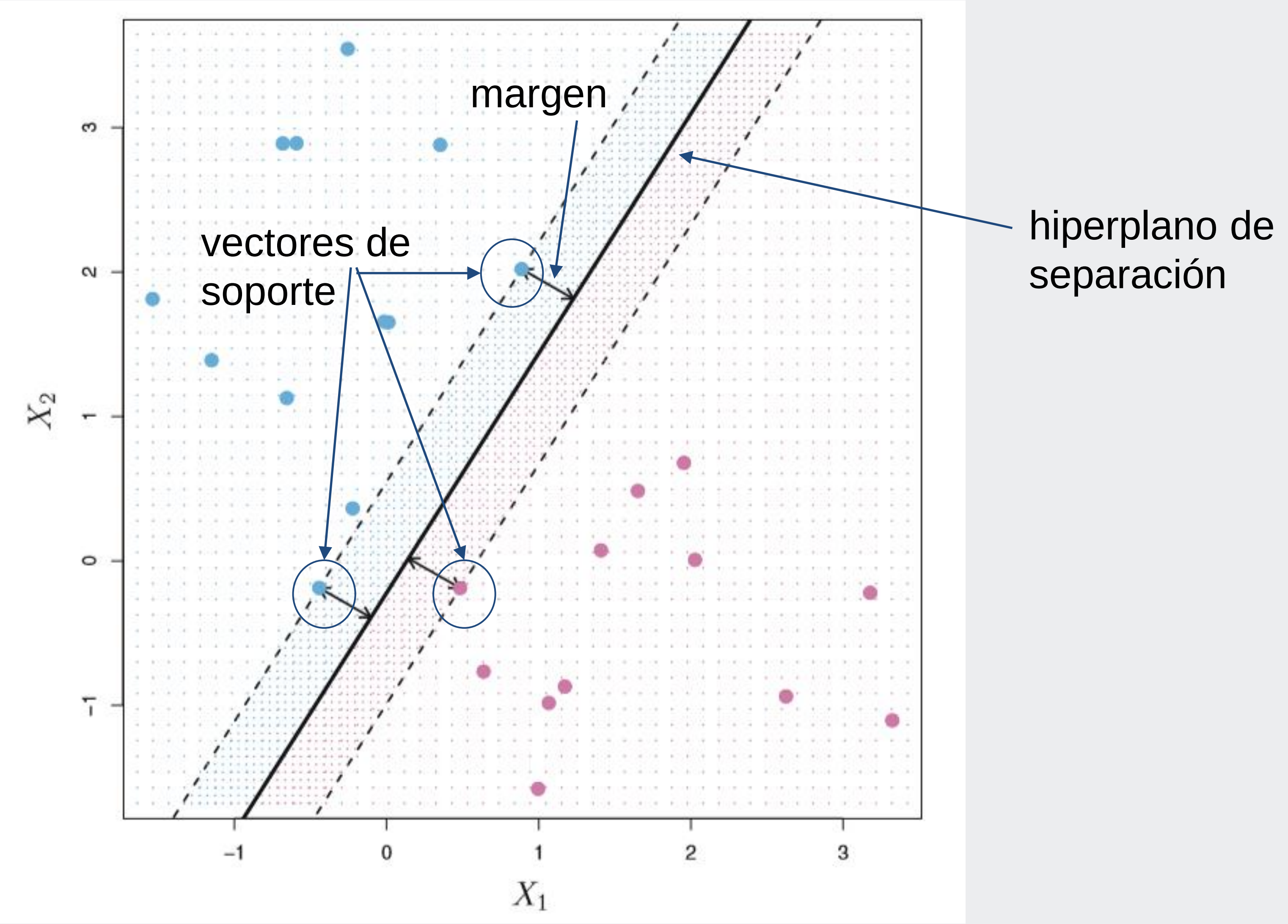
$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \ldots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}$$

$$y_1, \ldots, y_n \in \{-1, 1\}$$

targets: clases

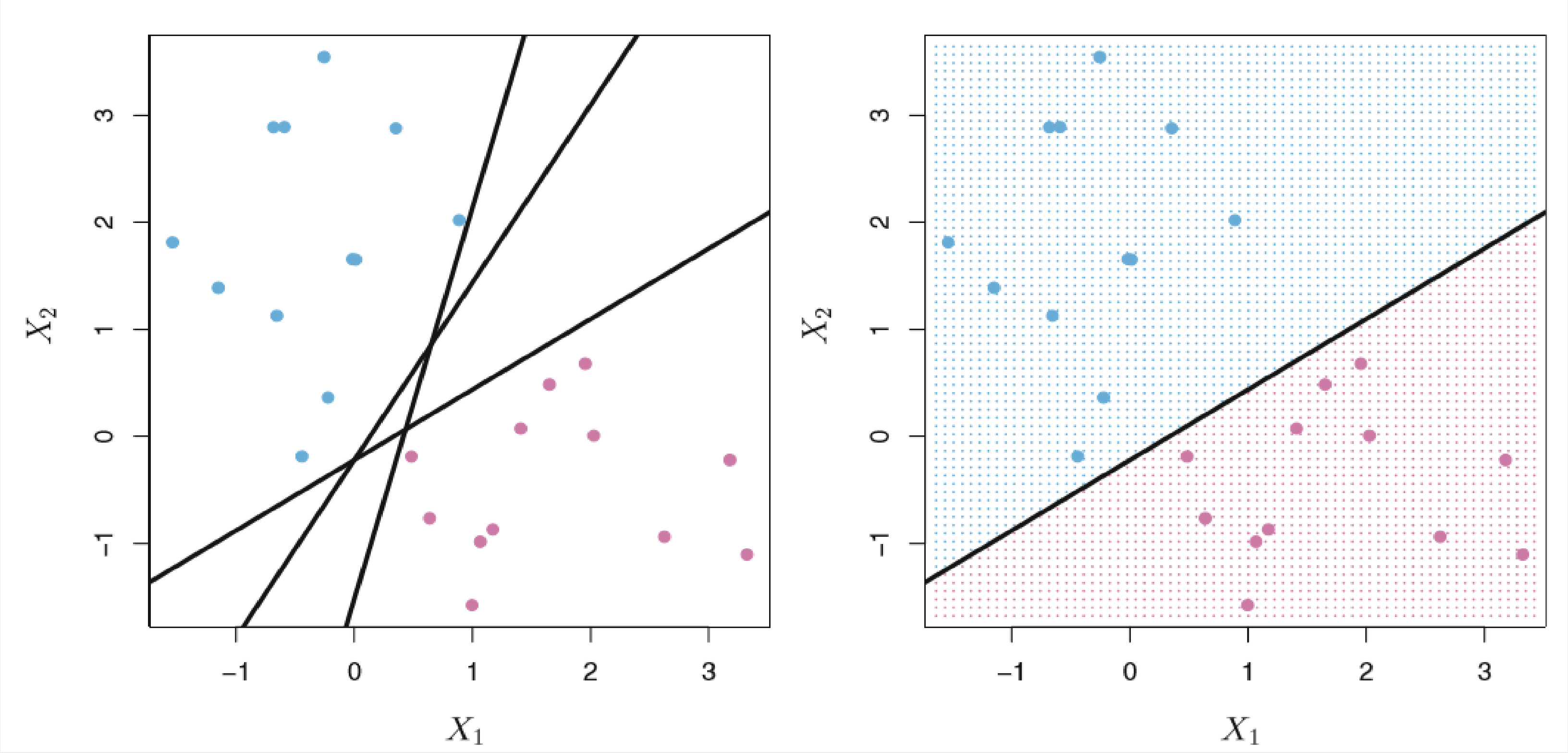
- En un problema lineal, un hiperplano que separa los datos será: $\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}$
  - si es >0, $y_i = 1$
  - si es <0, $y_i = -1$
  - la magnitud nos puede decir que tan cerca está del hiperplano (y de esa manera indicar una medida de confianza de predicción)

Maximal-Margin Classifier



Maximal-Margin Classifier

- ¿Cuál es la mejor separación?



Maximal-Margin Classifier

- ¿Cómo se define el mejor hiperplano?

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{maximize}}\, M$$

$$\text{subject to} \sum_{j=1}^{p} \beta_j^2 = 1, \quad \textbf{(A)}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M \quad \forall\, i = 1, \ldots, n$$

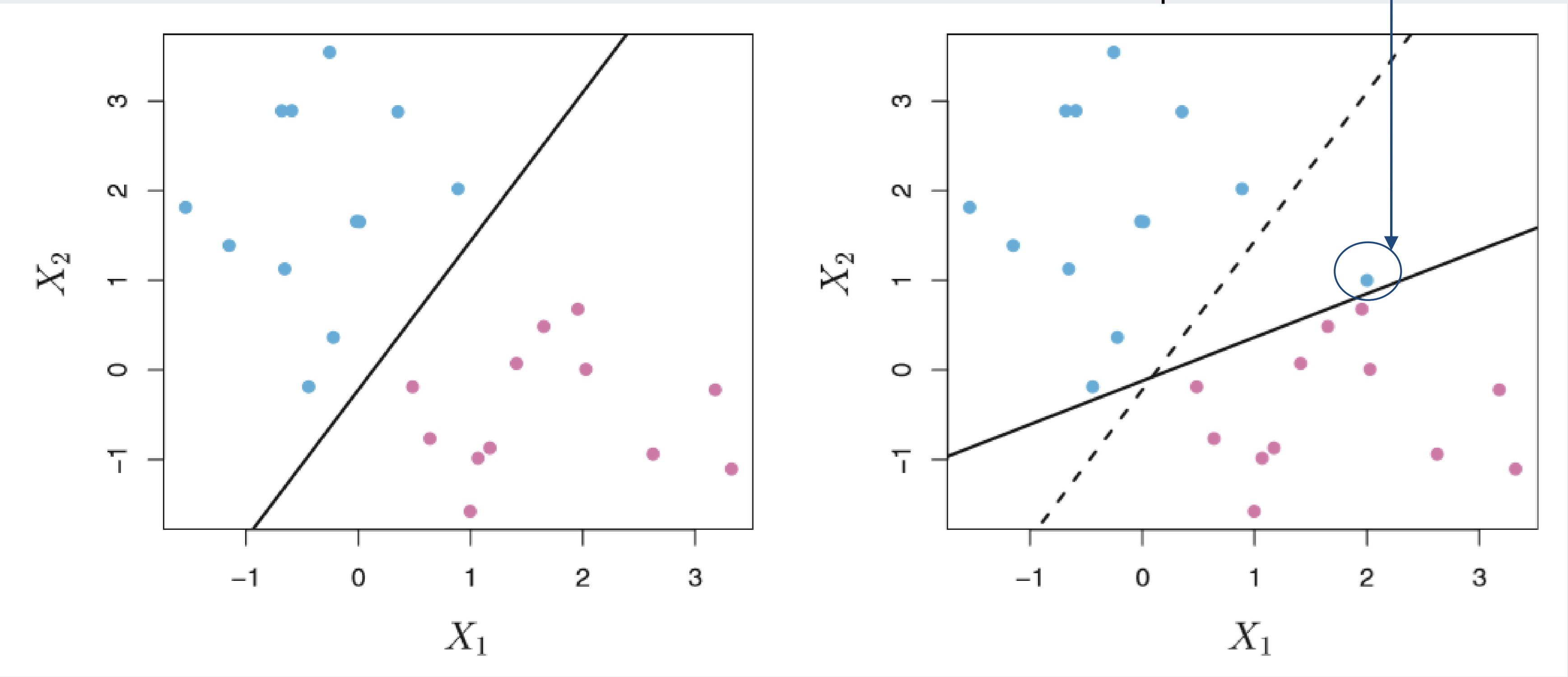
- (A) asegura que la distancia del punto al hiperplano en forma perpendicular es

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

- Si los ejemplos del dataset no son separables, el problema de optimización no tiene solución

Maximal-Margin Classifier

la introducción de este punto modifica completamente la separación



- Maximal-Margin genera este problema
  - reduce el margen de los vectores de soporte
  - es sensible a overfitting en entrenamiento

Soft-Margin Classifier

- Es mejor “errar” en la clasificación de algunos ejemplos para favorecer un modelo más robusto y generalizable

Soft-Margin Classifier

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n}{\text{maximize}} \quad M$$

$$\text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1 - \epsilon_i),$$

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

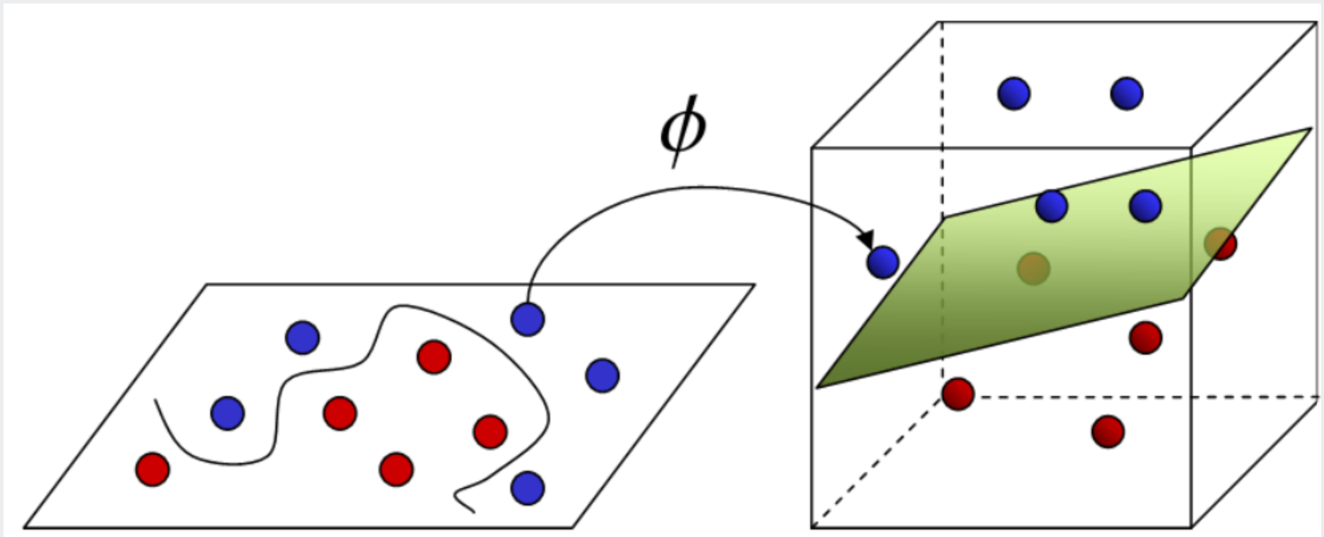
Variables de holgura (slack) - permiten que los ejemplos violen el hiperplano de separación
Si ∈i=0, la muestra está en el lado correcto del margen,
Si ∈i>1, la muestra está violando el hiperplano,
en otro caso ha violado solamente el margen

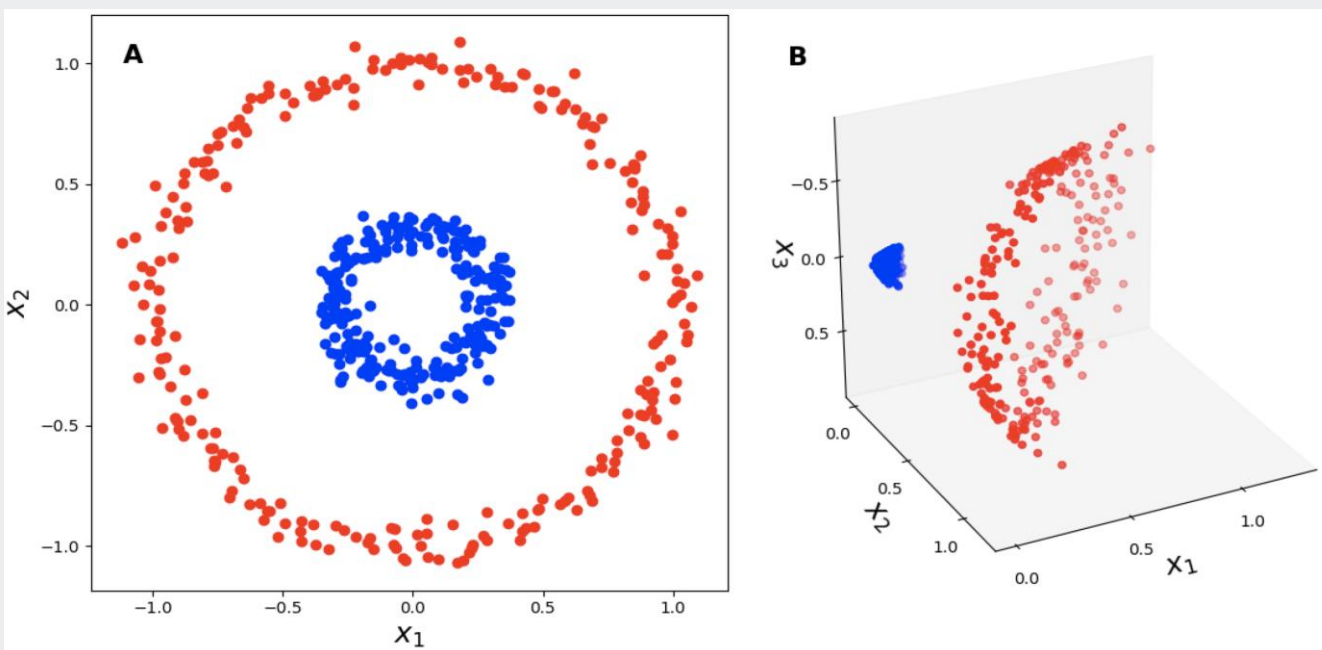
- C es un parámetro del modelo - Permite indicar la tolerancia que se va a tener para que los ejemplos violen la separación del hiperplano
- Si C=0 el modelo actúa como Maximal-Margin
- Si C>0, no más de C observaciones se permitirán que violen el hiperplano

Kernel trick

- ¿Cómo podemos extender estos modelos para problema no lineales?
  - Una forma es mapeando los datos originales en p-dimensiones (no separables) en m (m>p) dimensiones, donde sí son linealmente separables

Kernel trick





Kernel trick

- Si bien mapear a una mayor dimensión es posible, el cómputo es prohibitivo
- Ej: sean

$$\mathbf{x} = (x_1, x_2, x_3)^T$$
$$\mathbf{y} = (y_1, y_2, y_3)^T$$

3d

- Si necesitamos mapear a un espacio de 9 dimensiones:

$$\phi(\mathbf{x}) = (x_1^2, x_1x_2, x_1x_3, x_2x_1, x_2^2, x_2x_3, x_3x_1, x_3x_2, x_3^2)^T$$
$$\phi(\mathbf{y}) = (y_1^2, y_1y_2, y_1y_3, y_2y_1, y_2^2, y_2y_3, y_3y_1, y_3y_2, y_3^2)^T$$

9d

Kernel trick

- ¿Qué sucede si debemos calcular el producto vectorial de ambos vectores?

$$\phi(\mathbf{x})^T\phi(\mathbf{y}) = \sum_{i,j=1}^{3} x_i x_j y_i y_j$$

- Esta expresión es $O(n^2)$

Kernel trick

- Si usamos la función kernel k(x, y), en lugar de hacer cálculos en el espacio de 9 dimensiones obtenemos el mismo resultado operando en 3 dimensiones - O(n)

$$
\begin{aligned}
k(\mathbf{x}, \mathbf{y}) &= (\mathbf{x}^T \mathbf{y})^2 \\
&= (x_1 y_1 + x_2 y_2 + x_3 y_3)^2 \\
&= \sum_{i,j=1}^{3} x_i x_j y_i y_j
\end{aligned}
$$

- ¿Pero por qué esto nos interesa?

Kernel trick

- El problema original (maximal-margin classifier) puede escribirse como:

$$\begin{gathered}\text{maximize } M\\ M, \beta, b\\ y_i(\beta^T x_i + b) \geq M\\ \beta^T \beta = 1\end{gathered}$$

- Y luego:

$$\begin{gathered}min \frac{1}{2}|w|^2\\ y_i(w^T x_i + d) \geq 1\end{gathered}$$

- ahora este problema tiene función objetivo cuadrática y restricciones lineales

Kernel trick

- Se plantea el Lagrangiano de la formulación anterior
- Esa formulación incluye productos vectoriales que se reemplazan por kernels k(x,y) con las propiedades anteriores, los cuales tienen la propiedad de evaluar los puntos proyectados en el espacio de dimensiones mayor (pero a nivel de cálculo sin salir de la dimensión original)

Support Vector Machines
Tipos de Kernels

- Lineal: $$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}$$

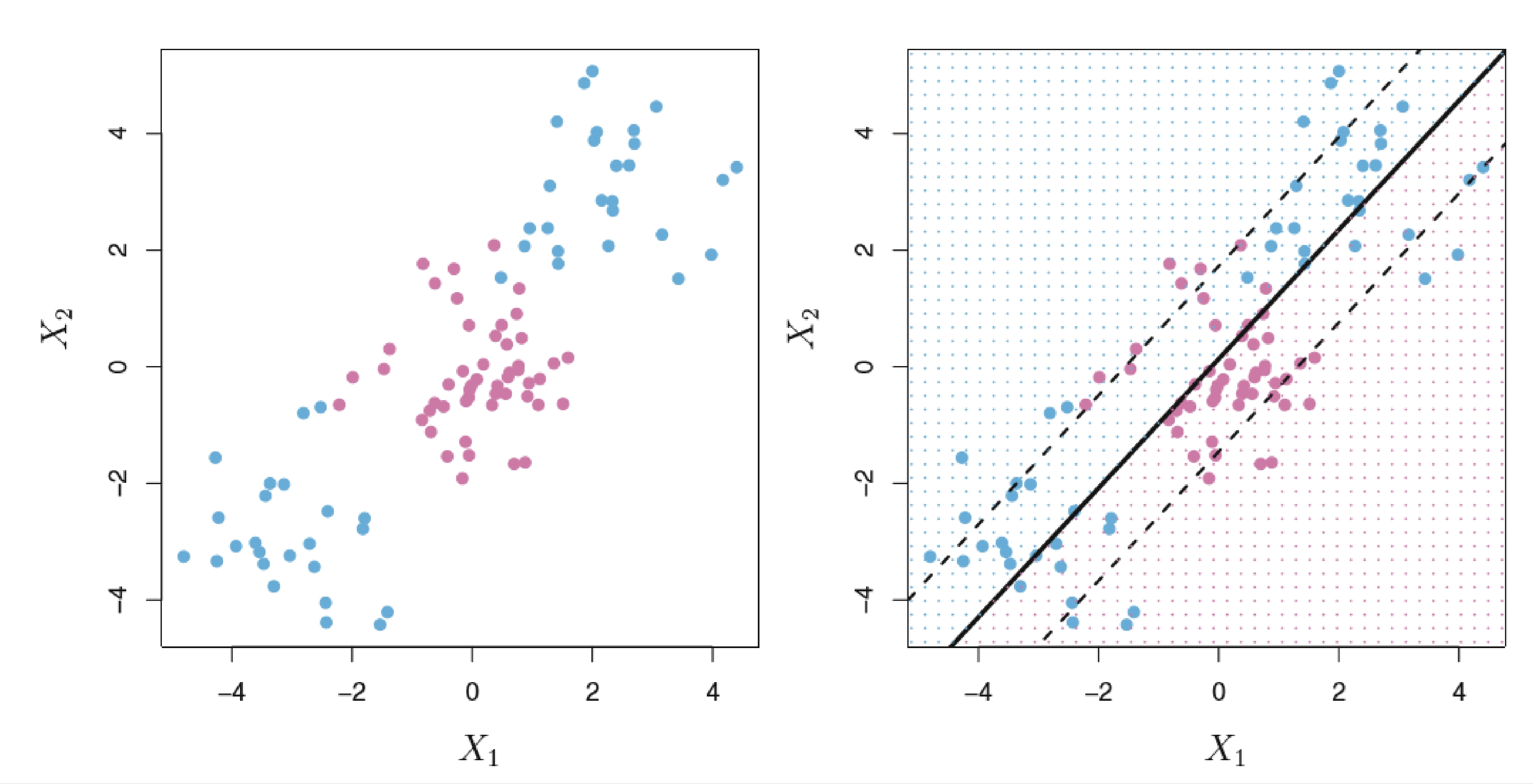
Polinómico: $$K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d$$

Radial: $$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2)$$

Kernels
no lineales

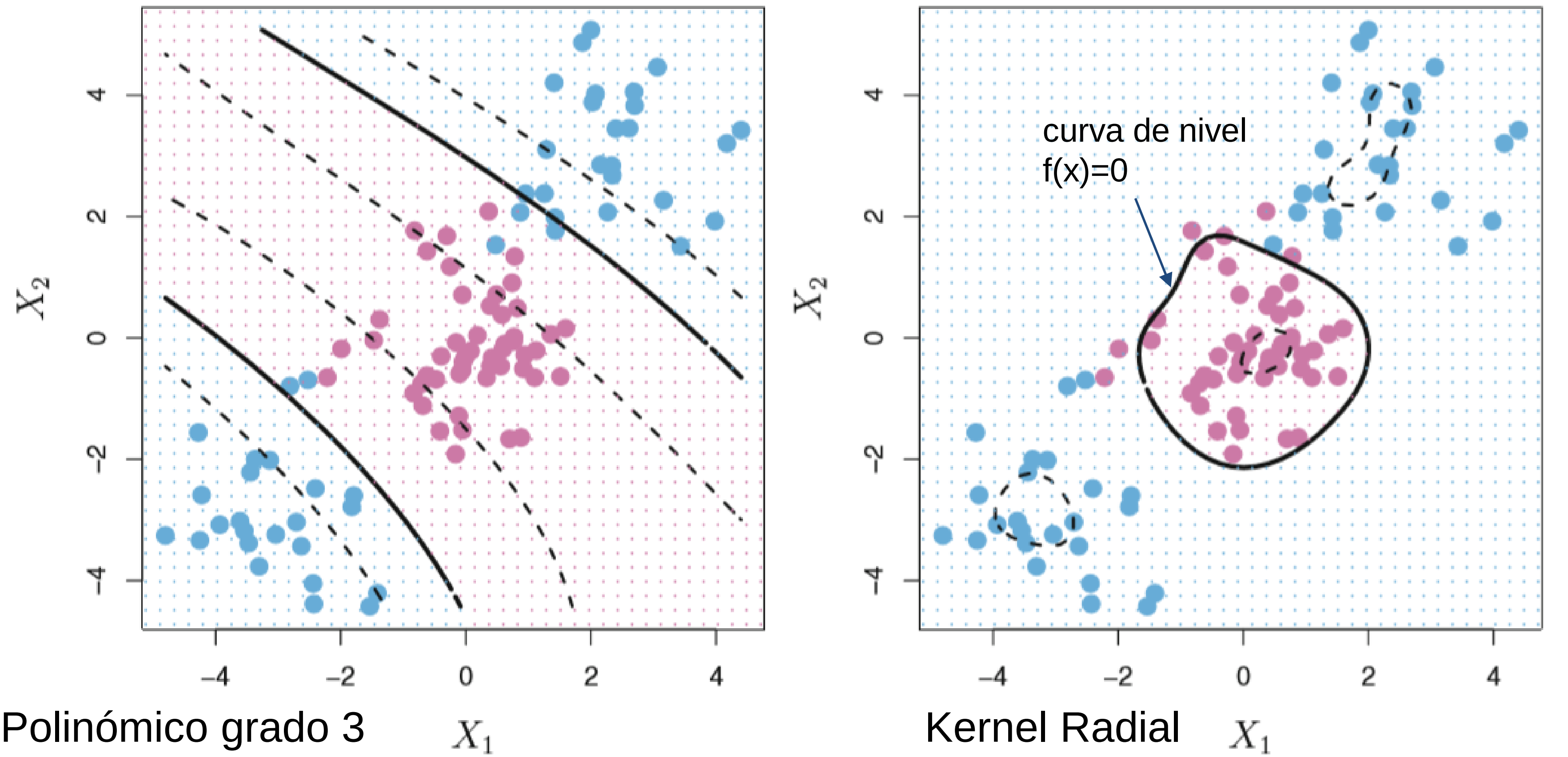
Support Vector Machines
Ejemplos

- Utilizando un kernel lineal



Support Vector Machines
Ejemplos

- Utilizando kernels no lineales



Polinómico grado 3

Kernel Radial

Support Vector Machines
Consideraciones

- Dada la estructura de los kernels es importante que los atributos estén normalizados
- Los parámetros del método, el kernel utilizado (y parámetros del kernel eventualmente) no deberían ser sobreajustados

Support Vector Machines
Ejemplo

base

radial