Máquinas de Vector Soporte

Métodos basados en Kernels en ML

Gabriel Martos Venturini
gmartos@utdt.edu

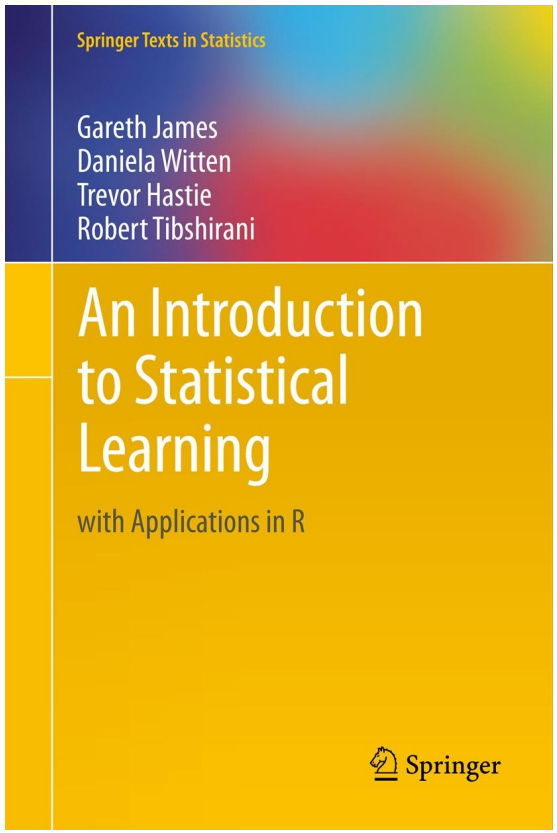
UNIVERSIDAD
TORCUATO DI TELLA

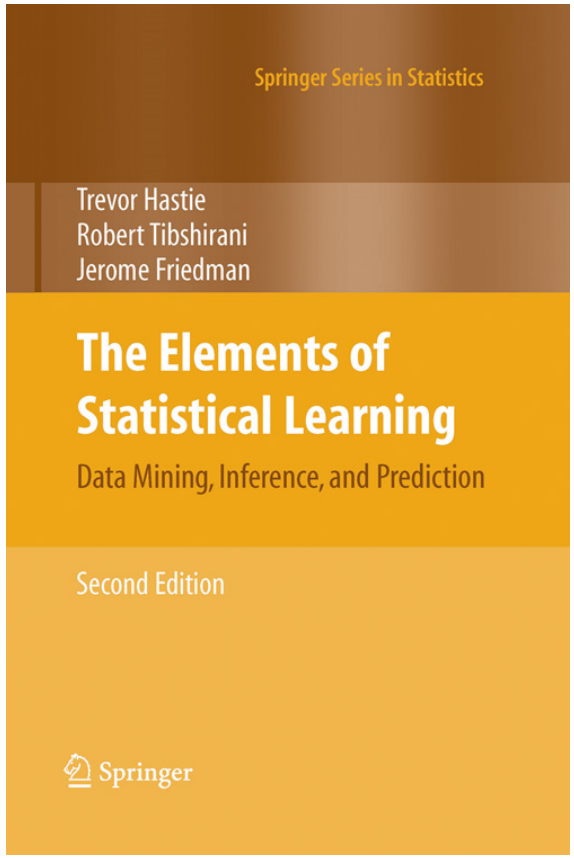
Agenda

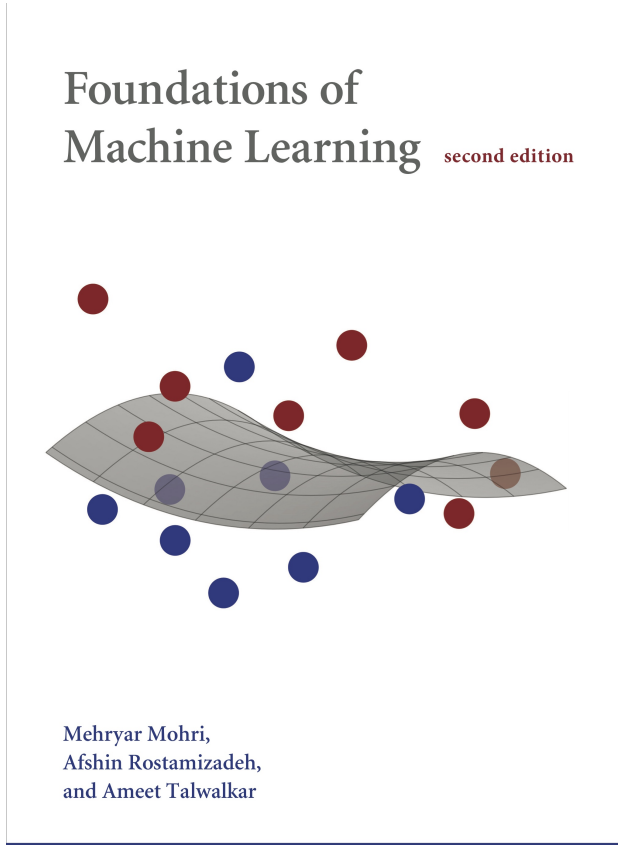
Clasificación con Máquinas de Vector Soporte

Regresiones con Máquinas de Vector Soporte

Bibliografía recomendada







ISL: Capítulo 9.
ESL: Sección 4.5 y Capítulo 12.
FML: Sección 5 .2 (detalles de los problemas de optimización).

Agenda

Clasificación con Máquinas de Vector Soporte

- Maximal margin classifier
- Support vector classifier
- Support vector machines

Regresiones con Máquinas de Vector Soporte

Hoja de ruta

- Clasificación:
  - Descripción del modelo y sus parámetros.
  - Clases separables: Estimación de parámetros vía maximización del margen (formulación del problema de optimización convexo).
  - Relajación del problema de optimización con variables de holgura.
    - Eliminamos el supuesto de separabilidad entre las clases.
  - El truco del kernel y los modelos de clasificación *no lineales*.
- Regresión con SVM.
- Implementación de los modelos en R: Librería `e1071`.

(recap)

- Un Hiperplano en $\mathbb{R}^p$ (parametrizado por $p+1$ parámetros $(\beta_0, \boldsymbol{\beta}) \equiv (\beta_0, \ldots, \beta_p)$) se corresponde con todos los puntos $(X_1, \ldots, X_p)$ para los que se cumple la ecuación:

$$\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p = 0.$$

- Un hiperplano parte el espacio de covariables en dos:

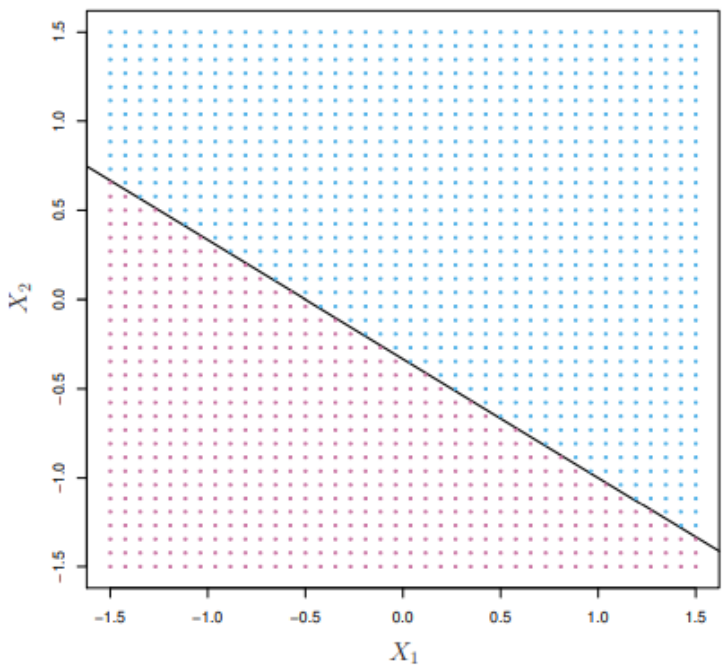


FIGURE 9.1. *The hyperplane* $1 + 2X_1 + 3X_2 = 0$ *is shown. The blue region is the set of points for which* $1 + 2X_1 + 3X_2 > 0$, *and the purple region is the set of points for which* $1 + 2X_1 + 3X_2 < 0$.

- Clasificación: $Y \in \{+1, -1\}$ y features $\mathbf{x} \in \mathbb{R}^p$ **numéricos**.

$$Y = \text{sign}\Big(\underbrace{\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p}_{f(\mathbf{x};\beta_0,\boldsymbol{\beta})} + \varepsilon\Big).$$

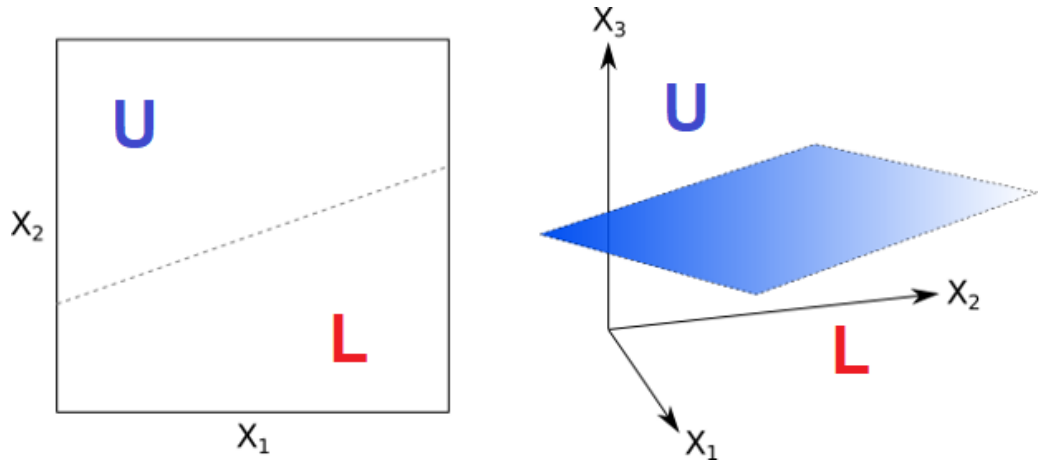


Figure: El hiperplano $H(\beta_0, \boldsymbol{\beta}) = \{\mathbf{x} \in \mathbb{R}^p : \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p = 0\}$ (parametrizado por $(\beta_0, \boldsymbol{\beta})$) parte el espacio de covariables en dos.

$$\underbrace{U_{\beta_0,\beta} = \{\mathbf{x} \in \mathbb{R}^p : \beta_0 + \sum_{i=1}^{p} \beta_i x_i > 0\}}_{\text{Epigrafo}} \text{ y } \underbrace{L_{\beta_0,\beta} = \{\mathbf{x} \in \mathbb{R}^p : \beta_0 + \sum_{i=1}^{p} \beta_i x_i < 0\}}_{\text{Hipografo}}.$$

- Si $\mathbf{x} \in U_{\beta_0,\beta} \Rightarrow$ Con una probabilidad alta: $Y = +1$.
- Si $\mathbf{x} \in L_{\beta_0,\beta} \Rightarrow$ Con una probabilidad alta: $Y = -1$.

- Dada una muestra de entrenamiento: $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$.

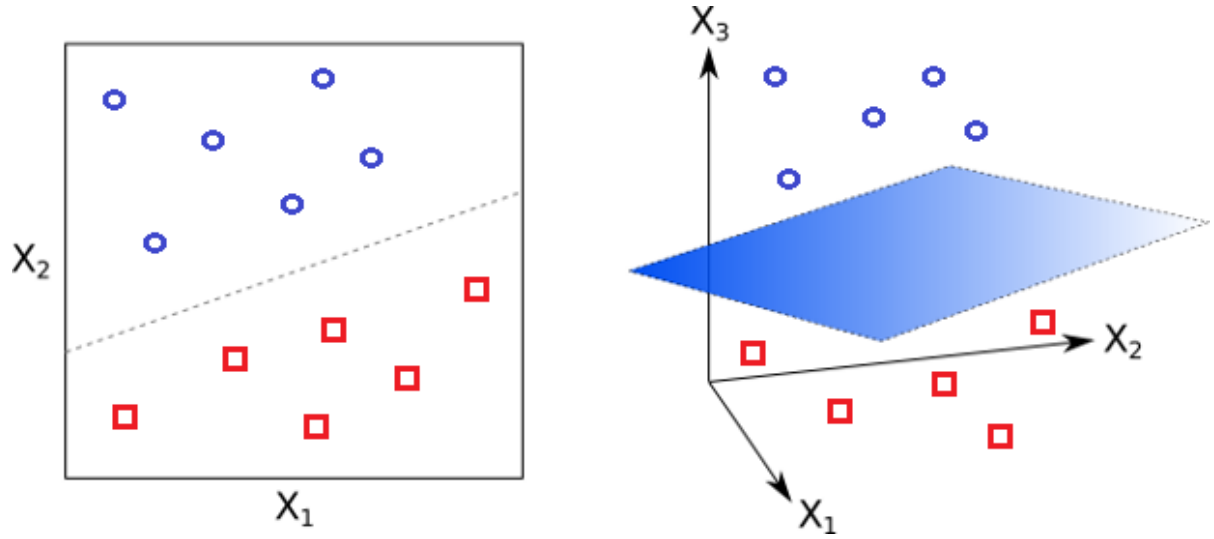


Figure: Aprendemos $(\beta_0, \boldsymbol{\beta})$ maximizando el margen entre las instancias.

  - Problema de optimización convexo.

- Una vez que aprendimos $(\beta_0, \boldsymbol{\beta})$, para $\mathbf{x}_{\text{new}}$ predecimos[1]

$$R(\mathbf{x}_{\text{new}}) = \text{sign}(\widehat{\beta}_0 + \widehat{\beta}_1 x_1 + \cdots + \widehat{\beta}_p x_p) \equiv \widehat{y}_{\text{new}}$$

  - $\widehat{y}_{\text{new}} = +1$ si $\mathbf{x}_{\text{new}} \in U_{\widehat{\beta}_0, \widehat{\boldsymbol{\beta}}}$, en cambio $\widehat{y}_{\text{new}} = -1$ si $\mathbf{x}_{\text{new}} \in L_{\widehat{\beta}_0, \widehat{\boldsymbol{\beta}}}$.

[1] De aquí en adelante: $\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p = \beta_0 + \boldsymbol{\beta}^T \mathbf{x}$.

Agenda

Maximal margin hyperplane

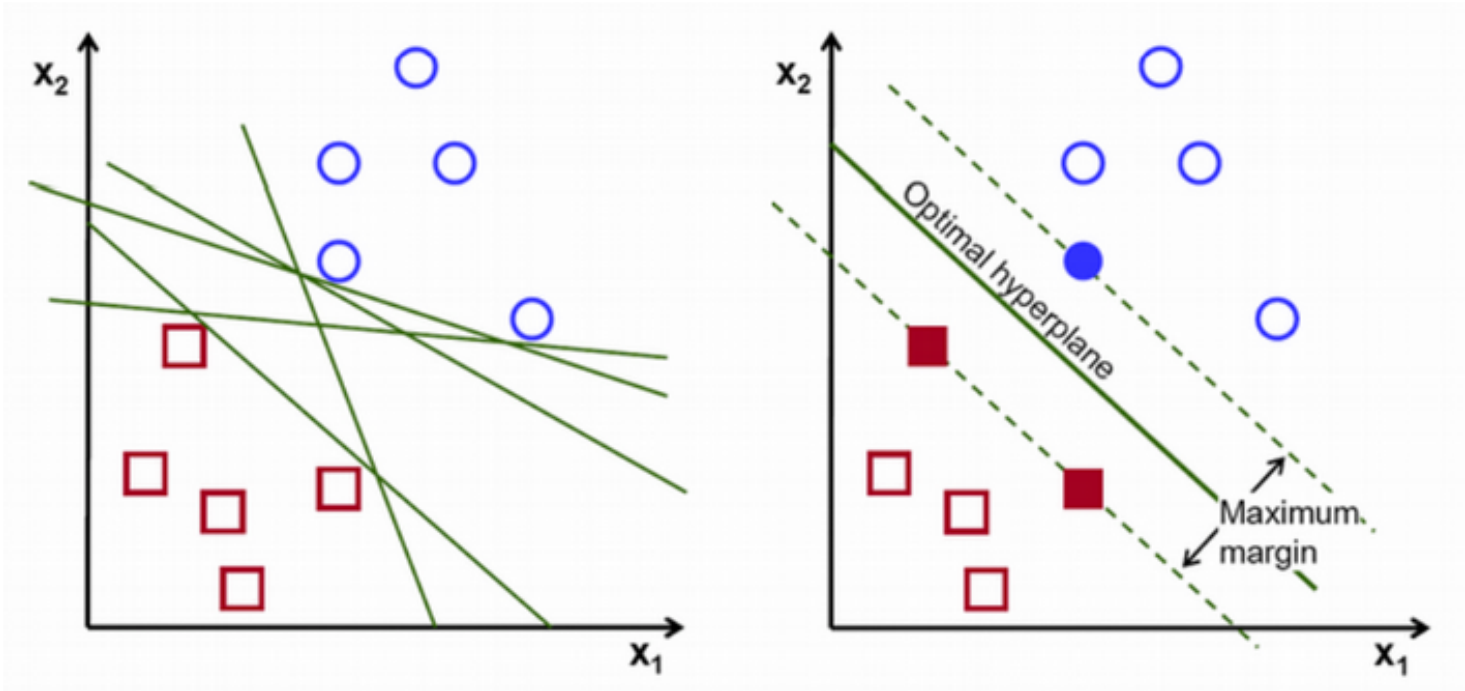


Figure: Muestra de entrenamiento ($n = 10$). Las instancias identificadas con $\square$ se corresponden con $Y = -1$ mientras que las instancias $\bigcirc$ con $Y = +1$.

- **Hiperplano óptimo**: Clasifica sin errores y esta más alejado de todas las observaciones de la muestra de entrenamiento.
- **Margen**: Espacio entre el hiperplano $H(b_0, \mathbf{b})$ y cada una de las *nubes de puntos*. El hiperplano óptimo maximiza el margen.

Elementos del problema de aprendizaje

- Llamemos $S_n = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ al conjunto de train.
- **Margen**: $M(b_0, \mathbf{b}, S_n) = 2 \min_{i=1,\ldots,n} d(\mathbf{x}_i, H(b_0, \mathbf{b}))$.

$$d(\mathbf{x}_i, H(b_0, \mathbf{b})) = \frac{\left| b_0, \mathbf{b}^T \mathbf{x}_i \right|}{\|\mathbf{b}\|}, \text{ para } i = 1, \ldots, n.$$

- Si $H(b_0, \mathbf{b})$ particiona correctamente el espacio de covariables:

$$y_i(b_0 + \mathbf{b}^T \mathbf{x}_i) \geq 0 \text{ para } i = 1, \ldots, n.$$

- El modelo plantea aprender los parámetros resolviendo:

$$\max_{b_0, \mathbf{b}} M(b_0, \mathbf{b}, S_n); \quad \text{sa: } y_i(b_0 + \mathbf{b}^T \mathbf{x}_i) \geq 0 \text{ para } i = 1, \ldots, n.$$

- Problema de optimización cuadrático y convexo.

Technicalities (BackUp slide)

- Formulación naive del problema de optimización:

$$\max_{b_0,\mathbf{b}} \left\{ 2 \min_{i=1,\ldots,n} \frac{\left| b_0 + \mathbf{b}^T \mathbf{x}_i \right|}{\|\mathbf{b}\|} \right\}, \quad \text{sa: } y_i(b_0 + \mathbf{b}^T \mathbf{x}_i) \geq 0, \ i = 1, \ldots, n.$$

- Invarianza de escala: $\min_{i=1,\ldots,n} |b_0 + \mathbf{b}^T \mathbf{x}_i| = 1.$
  - Luego se cumple que $\min_{i=1,\ldots,n} d(\mathbf{x}_i, H(b_0, \mathbf{b})) = 2/\|\mathbf{b}\|$.
- El problema queda planteado como:

$$\max_{b_0,\mathbf{b}} \left\{ \frac{2}{\|\mathbf{b}\|} \right\}, \quad \text{st: } y_i(b_0 + \mathbf{b}^T \mathbf{x}_i) \geq 1, \ \ i = 1, \ldots, n.$$

- Maximizar $2/\|\mathbf{b}\|$ equivale a minimizar $\|\mathbf{b}\|^2/2$, finalmente:

$$\min_{b_0,\mathbf{b}} \left\{ \frac{b_1^2 + \cdots + b_p^2}{2} \right\}, \quad \text{st: } y_i(b_0 + \mathbf{b}^T \mathbf{x}_i) \geq 1, \ \ i = 1, \ldots, n.$$

- Problema cuadrático y convexo (fácil de escalar en p).

Lagrangiano y condiciones de KKT (BackUp slide)

$$L(b_0, \mathbf{b}, \boldsymbol{\lambda}) = \frac{b_1^2 + \cdots + b_p^2}{2} - \sum_{i=1}^{n} \lambda_i (y_i(b_0 + \mathbf{b}^T \mathbf{x}_i) - 1).$$

Derivando e igualando a cero obtenemos que:

(1) $L'_{\mathbf{b}} : \widehat{\boldsymbol{\beta}} = \sum_{i=1}^{n} \lambda_i y_i \mathbf{x}_i$ (recuerda que $\mathbf{x}$ y $\boldsymbol{\beta} \in \mathbb{R}^p$),

(2) $L'_{b_0} : \sum_{i=1}^{n} \lambda_i y_i = 0$,

(3) $L_{\lambda_i} : \lambda_i (y_i(\widehat{\beta}_0 + \widehat{\boldsymbol{\beta}}^T \mathbf{x}_i) - 1) = 0$, con $\lambda_i \geq 0$, para $i = 1, \ldots, n$.

De (3) se deduce que:

- $\lambda_i > 0 \iff y_i(\widehat{\beta}_0 + \widehat{\boldsymbol{\beta}}^T \mathbf{x}_i) = 1$.
- Solo los *vectores soportes* (SV $= \{i \subseteq (1, \ldots, n) | \lambda_i > 0\}$) contribuyen a determinar los valores de $\mathbf{b}$ y b_0 a través de (1).

Vectores soportes

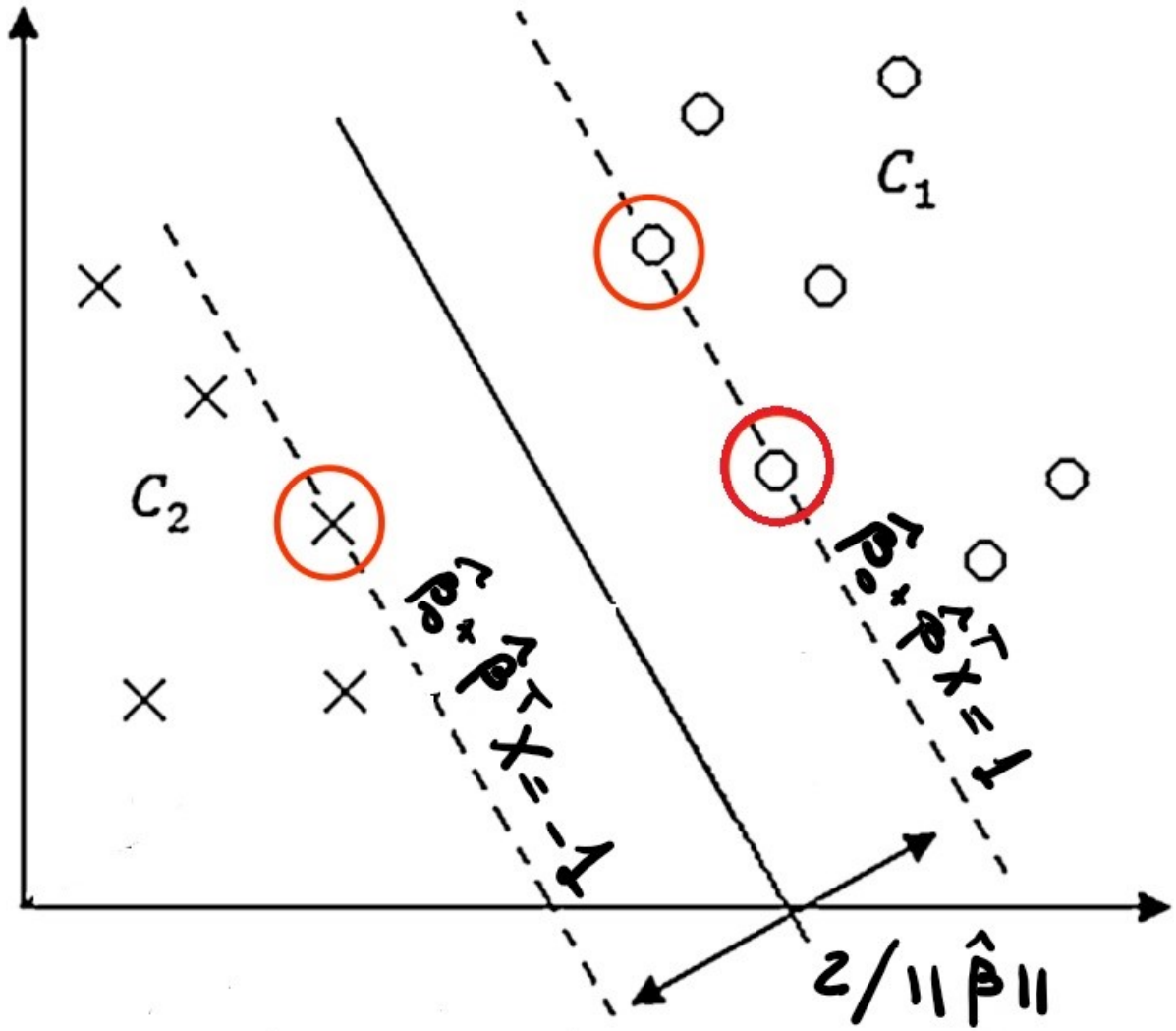


Figure: Las observaciones de la muestra de entrenamiento para las que se cumpla que $y_i(\widehat{\beta}_0 + \widehat{\boldsymbol{\beta}}^T \mathbf{x}_i) = 1$, determinan las estimaciones de $\widehat{\beta}_0$ y $\widehat{\boldsymbol{\beta}}$.

Formulación dual (BackUp slide)

- Permitirá modelar márgenes no lineales (*kernel–trick*).
- Introduciendo (1) y (2) en $L(\beta_0, \boldsymbol{\beta}, \lambda)$:

$$L_D(\lambda_1, \ldots, \lambda_n) = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j \underbrace{\mathbf{x}_i^T \mathbf{x}_j}_{\langle \mathbf{x}_i, \mathbf{x}_j \rangle} + \sum_{i=1}^{n} \lambda_i. \quad (\text{ver slide §–30})$$

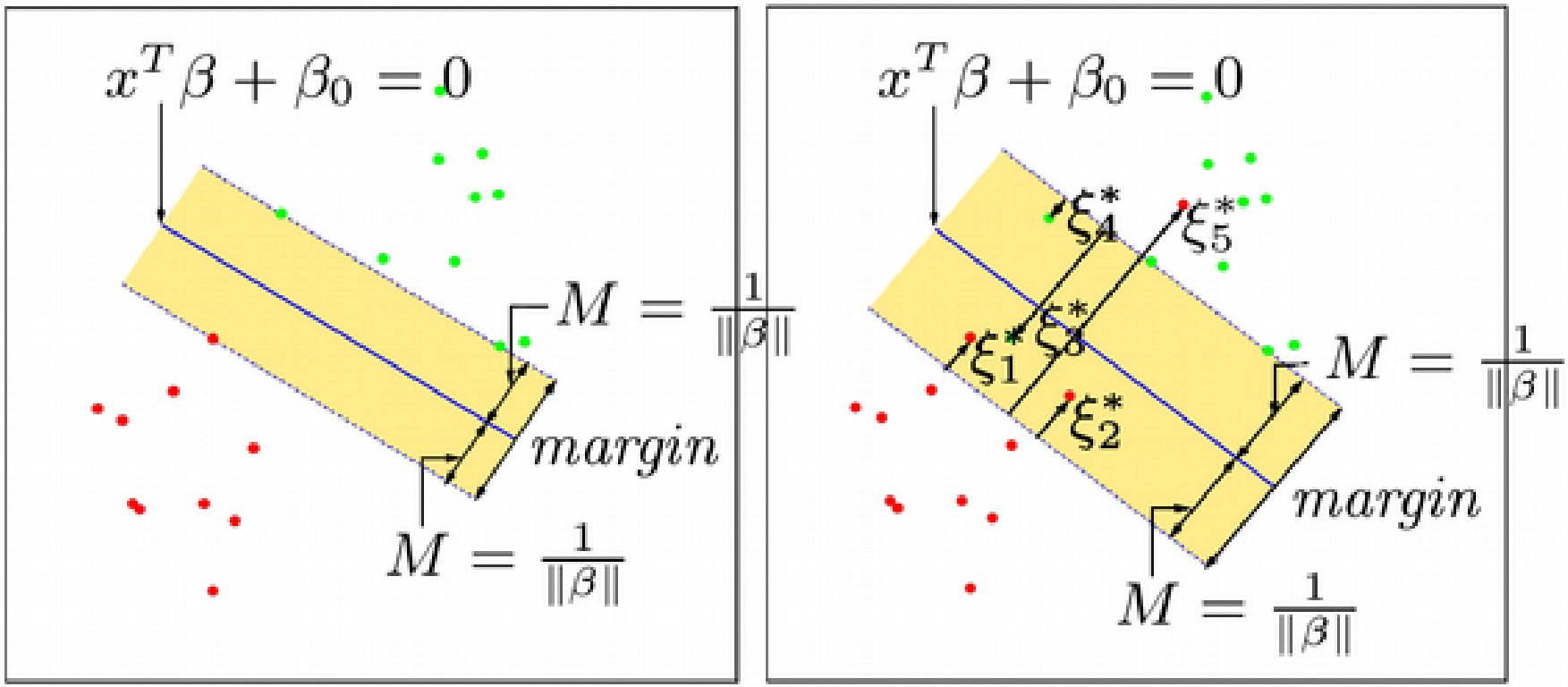
- La maximización del dual (de Wolfe) es también un problema convexo y cuadrático (existencia, unicidad y escalabilidad).
- El solver te devuelve $\lambda_1, \ldots, \lambda_n$ con $\lambda_i > 0 \Leftrightarrow i \in \text{SV}$.

$$\widehat{\boldsymbol{\beta}} = \sum_{i \in \text{SV}} \lambda_i y_i \mathbf{x}_i \text{ con } |\text{SV}| \text{ pequeño.}$$

- $\widehat{\beta}_0 = \frac{1}{|\text{SV}|} \sum_{i \in \text{SV}} (y_i - \sum_{j=1}^{n} \lambda_j y_j \mathbf{x}_j^T \mathbf{x}_i)$.

Agenda

- Introduciendo $\{\xi_1, \ldots, \xi_n\}$ relajamos supuesto de *separabilidad*.



$$\min_{b_0,\mathbf{b}} \frac{1}{2}\|\mathbf{b}\|^2, \quad \text{sujeto a:}$$

$$y_i(b_0 + \mathbf{b}^T\mathbf{x}_i) \geq 1, \; i = 1, \ldots, n.$$

$$\min_{b_0,\mathbf{b},\boldsymbol{\xi}} \frac{1}{2}\|\mathbf{b}\|^2 + C\sum_{i=1}^{n} \xi_i, \quad \text{sujeto a:}$$

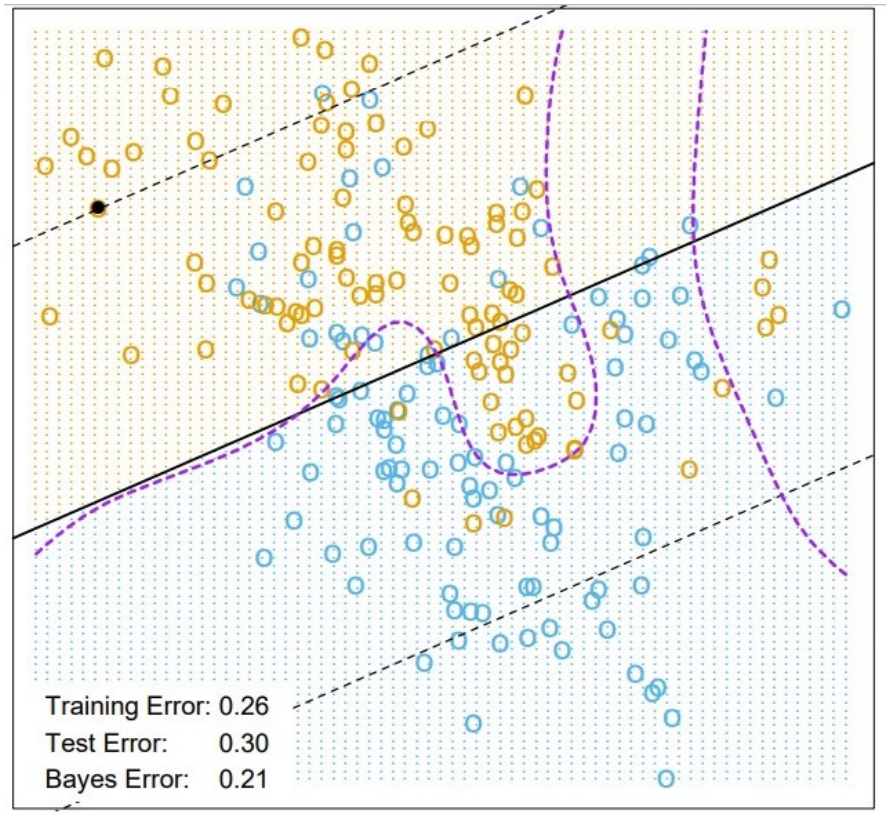
$$y_i(b_0 + \mathbf{b}^T\mathbf{x}_i) \geq 1 - \xi_i,$$

$$\xi_i \geq 0 \text{ para } i = 1, \ldots, n.$$

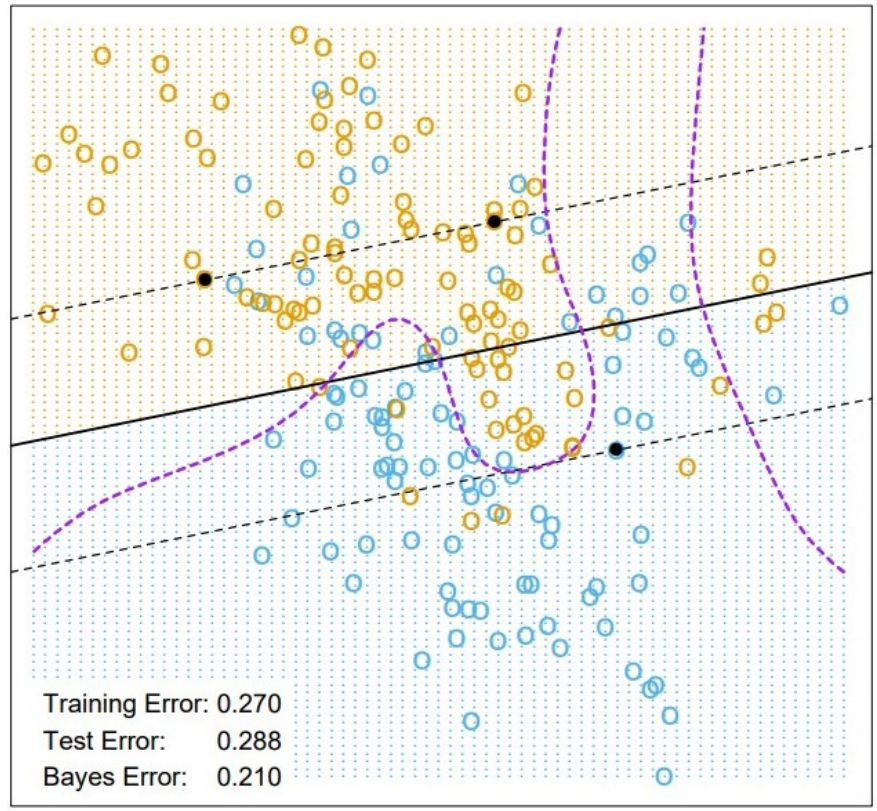
- C nos permite calibrar el trade–off entre bias y variance.

Interpretación del hiperparámetro 'C'

- C es el hiperparámetro *sensible* del modelo:
  - $\uparrow C$: Margen más pequeños $\Rightarrow -$ vectores soporte.
    - $\downarrow$ sesgo y $\uparrow$ varianza $\Rightarrow$ incrementa riesgo overfitting.
  - $\downarrow C$: Margen mas grandes $\Rightarrow +$ vectores soporte.
    - $\uparrow$ sesgo y $\downarrow$ varianza $\Rightarrow$ incrementa riesgo underfitting.
- Aprendemos C por validación cruzada.
- Sólo aquellas observaciones con parámetros de holgura positivos (más próximas a $H(\widehat{\beta}_0, \widehat{\boldsymbol{\beta}})$) determinan el hiperplano óptimo.
  - Además si $\widehat{\xi}_i > 1/\|\widehat{\boldsymbol{\beta}}\| \Rightarrow y_i \neq \widehat{y}_i$.
  - Detalles del problema de optimización en ESL § 12.2.1 (pp 420).



$C = 0.01$



$C = 10000$

Figure: Los vectores soporte se corresponden con los datos del conjunto de entrenamiento que caen dentro del margen y los mal clasificados. Valores más grande de C producen soluciones con márgenes más pequeños (ESL).

- Warning: En ISL § 9.2.2. (pp 377, V2) se define el problema de optimización de otra manera (equivalente) y el parámetro C se interpreta exactamente al revés de como lo hacemos aquí.

(BackUp)

$$L_P(b_0, \mathbf{b}, \boldsymbol{\xi}) = \frac{1}{2}\|\mathbf{b}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\lambda_i[y_i(b_0 + \mathbf{b}^T\mathbf{x}_i) - (1-\xi_i)] + \mu_i\sum_{i=1}^{n}\xi_i.$$

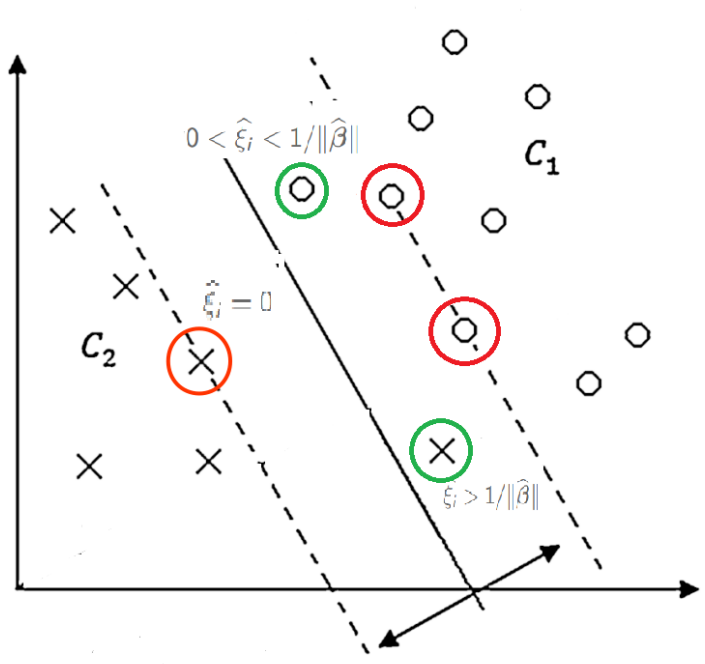


Figure: $\text{SV} = \{i \in \{1, \ldots, n\} \mid \lambda_i > 0 \,\&\, \xi_i \geq 0\}$, luego $\widehat{\boldsymbol{\beta}} = \sum_{i \in \text{SV}} \lambda_i y_i \mathbf{x}_i$

- M. Mohri et al: *Foundations of Machine Learning* (pp 89 § 5.3.2).

Agenda

El paquete e1071 en R

```
library(e1071)
svm(y ~ . , data , kernel = "radial", cross , type,
 gamma , cost , scale, na.action = na.omit)
```

- Librería que implementa varios modelos de aprendizaje automático, entre ellos las máquinas de vector soporte.
- Nos ayuda a hacer validación cruzada para los parámetros de la SVM de manera eficiente (ver función `tune.svm()`).
- Modelos no lineales: Implementa de manera automática el uso de núcleos polinómicos, Gaussiano y sigmoide (ver próximas slides).
- En problemas multiclase utiliza la estrategia de uno contra uno.

Predicción del rango de precio de teléfonos celulares

```
battery_power: Total energy a battery (in mAh).
blue: Has bluetooth or not.
clock_speed:  microprocessor speed.
dual_sim: Has dual sim support or not.
fc: Front Camera mega pixels.
four_g: Has 4G or not.
int_memory: Internal Memory (in Gbt).
m_dep: Mobile Depth in cm.
mobile_wt: Weight of mobile phone
n_cores: Number of cores of processor.
pc: Primary Camera mega pixels.
... (algunas variables más) ...

price_range: 0(low cost), 1(medium cost),
             2(high cost) and 3(very high cost).
```

Agenda

Clasificación con Máquinas de Vector Soporte

- Maximal margin classifier
- Support vector classifier
- Support vector machines

Regresiones con Máquinas de Vector Soporte

Márgenes no lineales

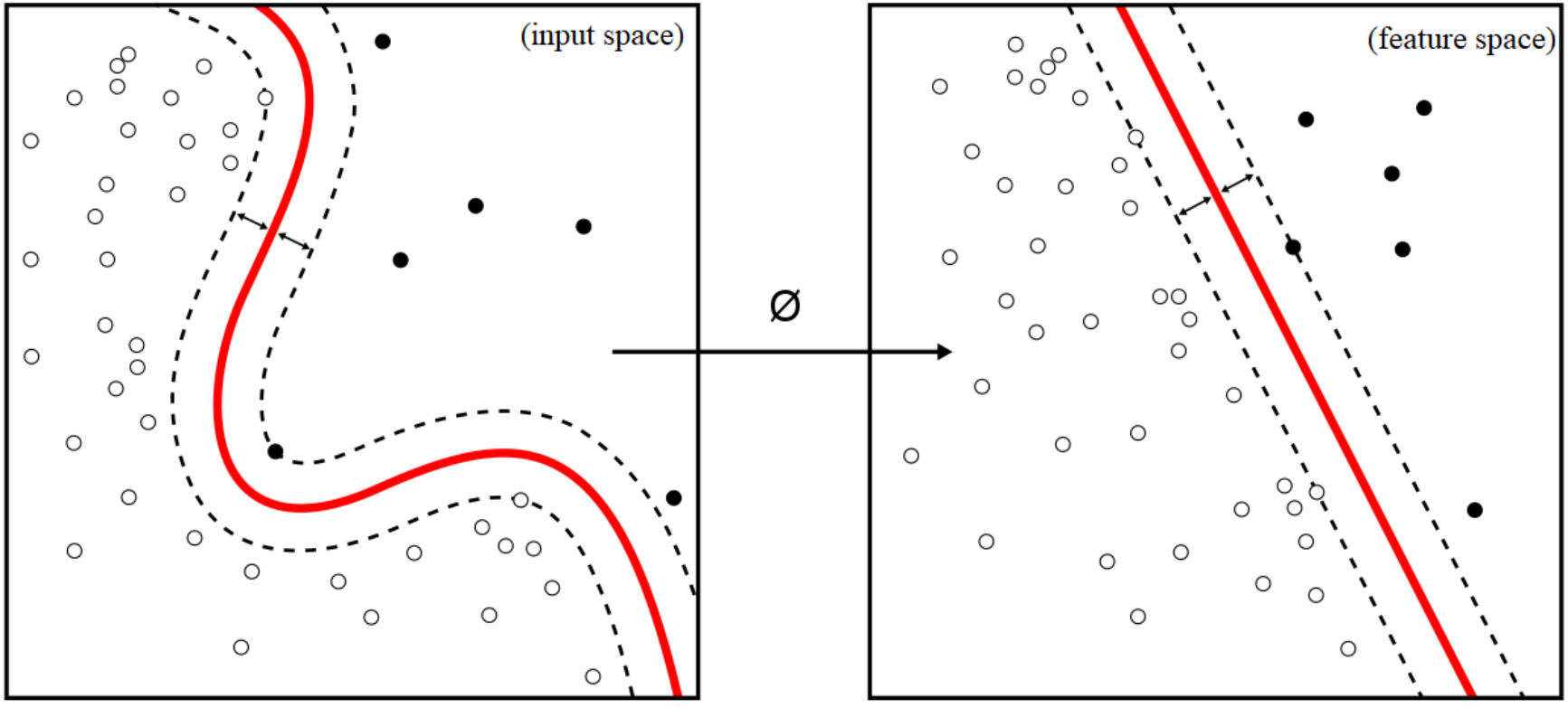


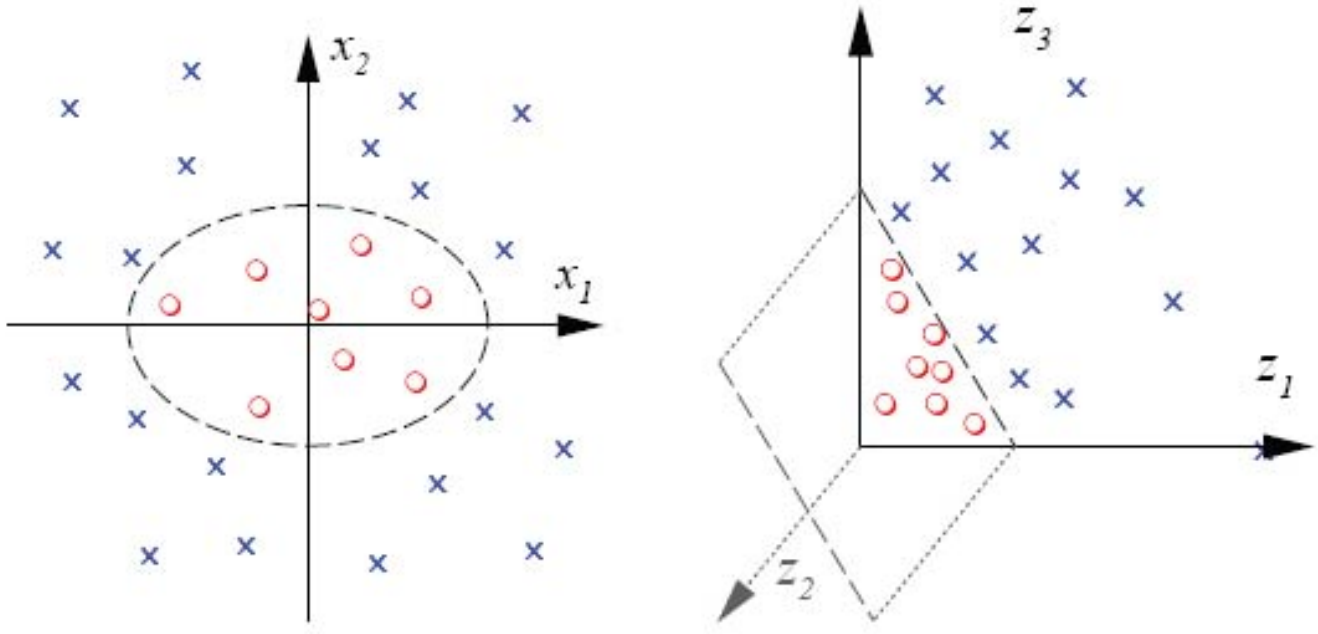
Figure: Mapeamos los datos y modelamos (linealmente) en el *feature space*.

- No es necesario explicitar Φ. Para aprender los parámetros (y hacer predicciones) sólo necesitas computar el producto interior:

$$K_\sigma(x, x') \equiv \Phi(x)^T \Phi(x')$$

Ejemplo

- Problema de clasificación con $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$:



- Si consideramos el mapa:

$$\Phi(x_1, x_2) \equiv (z_1 = x_1^2, z_2 = x_2^2, z_3 = \sqrt{2}x_1x_2).$$

- ... en las coordenadas $\{(\mathbf{z}_1, y_1), \ldots, (\mathbf{z}_n, y_n)\}$ el problema se puede resolver utilizando técnicas lineales (por ejemplo con SVM's!).

- Considerando $\Phi(\mathbf{x} = (x_1, x_2)) \equiv (z_1 = x_1^2, z_2 = x_2^2, z_3 = \sqrt{2}x_1x_2)$:

$$\begin{aligned} \Phi(\mathbf{x})^T \Phi(\mathbf{x}') &= x_1^2 x_1'^2 + x_2^2 x_2'^2 + 2x_1 x_1' x_2 x_2' \\ &= \underbrace{(x_1 x_1' + x_2 x_2')^2}_{K(\mathbf{x},\mathbf{x}')} \end{aligned}$$

  - Existe una relación biunívoca entre K y Φ (condiciones).

- La función $K(\mathbf{x}, \mathbf{x}')$ codifica relaciones de similaridad en el feature–space entre pares de observaciones $\mathbf{x}$ y $\mathbf{x}'$.

- Para estimar el hiperplano en el feature–space (partir de manera no lineal el imput–space), las SVM's sólo necesitan evaluar:

$$K(\mathbf{x}_i, \mathbf{x}_j) \equiv \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j), \text{ con } i, j \in \{1, \ldots, n\}.$$

Núcleo K y su relación con Φ

- Si $K : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}^+$ es un mapa simétrico y definido positivo (technicalities) entonces existe una única función Φ tal que

$$K(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x})^T \Phi(\mathbf{x}'), \quad \forall\, \mathbf{x}, \mathbf{x}' \in \mathbb{R}^p.$$

- Esta *dualidad* permite modelar el problema en el *feature–space* sin necesidad de definir explícitamente la función Φ (elegimos K).
  - El algoritmo sólo necesita computar $\Phi(\mathbf{x})^T \Phi(\mathbf{x}') = K(\mathbf{x}, \mathbf{x}')$ cada vez que estimo los parámetros del modelo y/o hago predicciones.
- K tendrá *parámetros* que *aprenderemos* (por VC).
  - Cambiando parámetros de K $\to$ diferentes grados de linealización.
  - El costo de la linealización lo pagamos en términos de complejidad.
    - Bias–variance trade off.
- ... algunos ejemplos de K's y sus parámetros...

Kernels utilizados habitualmente en ML

- Kernel Polinómico de parámetros σ y c:

$$K_{c,\sigma}(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}' + c)^\sigma.$$

  - σ determina la dimensión del feature space $((p+\sigma)!/(\sigma! p!))$
  - c localiza la media en el feature space.

- Kernel Sigmoide de parámetros σ y c:

$$K_{\sigma,c}(\mathbf{x}, \mathbf{x}') = \frac{\exp[2(\sigma \mathbf{x}^T \mathbf{x}' + c)] + 1}{\exp[2(\sigma \mathbf{x}^T \mathbf{x}' + c)] - 1}.$$

- **Kernel Gaussiano** (Radial Basis Function) de parámetro σ:

$$K_\sigma(\mathbf{x}, \mathbf{x}') = \exp(-\sigma ||\mathbf{x} - \mathbf{x}'||^2).$$

  - σ: Parámetro que tunea la complejidad del mapa.
  - Kernel suficientemente flexible como para lidiar con muchos de los problemas de aprendizaje supervisado (aprendemos σ por VC).

Aprendizaje de parámetros (BackUp)

- Dado Φ y el data set de TRAIN: $\{(\Phi(\mathbf{x}_1), y_1), \ldots, (\Phi(\mathbf{x}_n), y_n)\}$.
- De la formulación dual (▸ ver slide §–15):

$$L(\lambda_1, \ldots, \lambda_n) = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i,j=1}^{n} \lambda_i \lambda_j y_i y_j \underbrace{\Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)}_{K_\sigma(\mathbf{x}_i, \mathbf{x}_j)}.$$

- Para aprender los parámetros $\{\lambda_i\}_{i=1}^n$ (y con estos $(\beta_0, \boldsymbol{\beta})$ en el feature space) solo necesito poder computar $K_\sigma(\mathbf{x}_i, \mathbf{x}_j)$.
- Tu solución dependerá de los hiperparámetros σ de K_σ (VC).
  - Y del hiperparámetro C (omitido en el planteo para simplificar).
- Una vez aprendidos $(\widehat{\beta}_0, \widehat{\boldsymbol{\beta}})$, $H(\widehat{\beta}_0, \widehat{\boldsymbol{\beta}})$ en el feature space se corresponde con una **partición no lineal del input space**: $H_{\widehat{f}} = \{\mathbf{x} : \widehat{f}(\mathbf{x}) = 0\}$ (Epi $= \{\mathbf{x} : \widehat{f}(\mathbf{x}) > 0\}$ e Hipo$\{\mathbf{x} : \widehat{f}(\mathbf{x}) < 0\}$).

Quadratic programming (BackUp slide)

- El 'solver' resuelve:

$$\min_{\lambda_1,\ldots,\lambda_n} \boldsymbol{\lambda}^T \begin{bmatrix} y_1^2 K(x_1,x_1) & y_1 y_2 K(x_1,x_2) & \cdots & y_1 y_n K(x_1,x_n) \\ \vdots & \ddots & \cdots & \vdots \\ y_1 y_n K(x_n,x_1) & y_n y_2 K(x_n,x_2) & \cdots & y_n^2 K(x_n,x_n) \end{bmatrix} \boldsymbol{\lambda} - \mathbf{1}^T \boldsymbol{\lambda}$$

$$s.a: \quad \mathbf{y}^T \boldsymbol{\lambda} = \mathbf{0} \text{ y } \lambda \geq \mathbf{0}.$$

- No necesitamos conocer Φ para aprender $\lambda_1, \ldots, \lambda_n$.
- El algoritmo escala bien en p y mal en n.
- Aprendemos los hiperparámetros σ y C por VC.
- Solución rala en el feature space:

$$\lambda_i > 0 \rightarrow x_i \in \text{SV y } \lambda_i = 0 \text{ en otro caso,}$$

donde SV es el conjunto de vectores soportes (en el feature space).

(ver simulación en R)

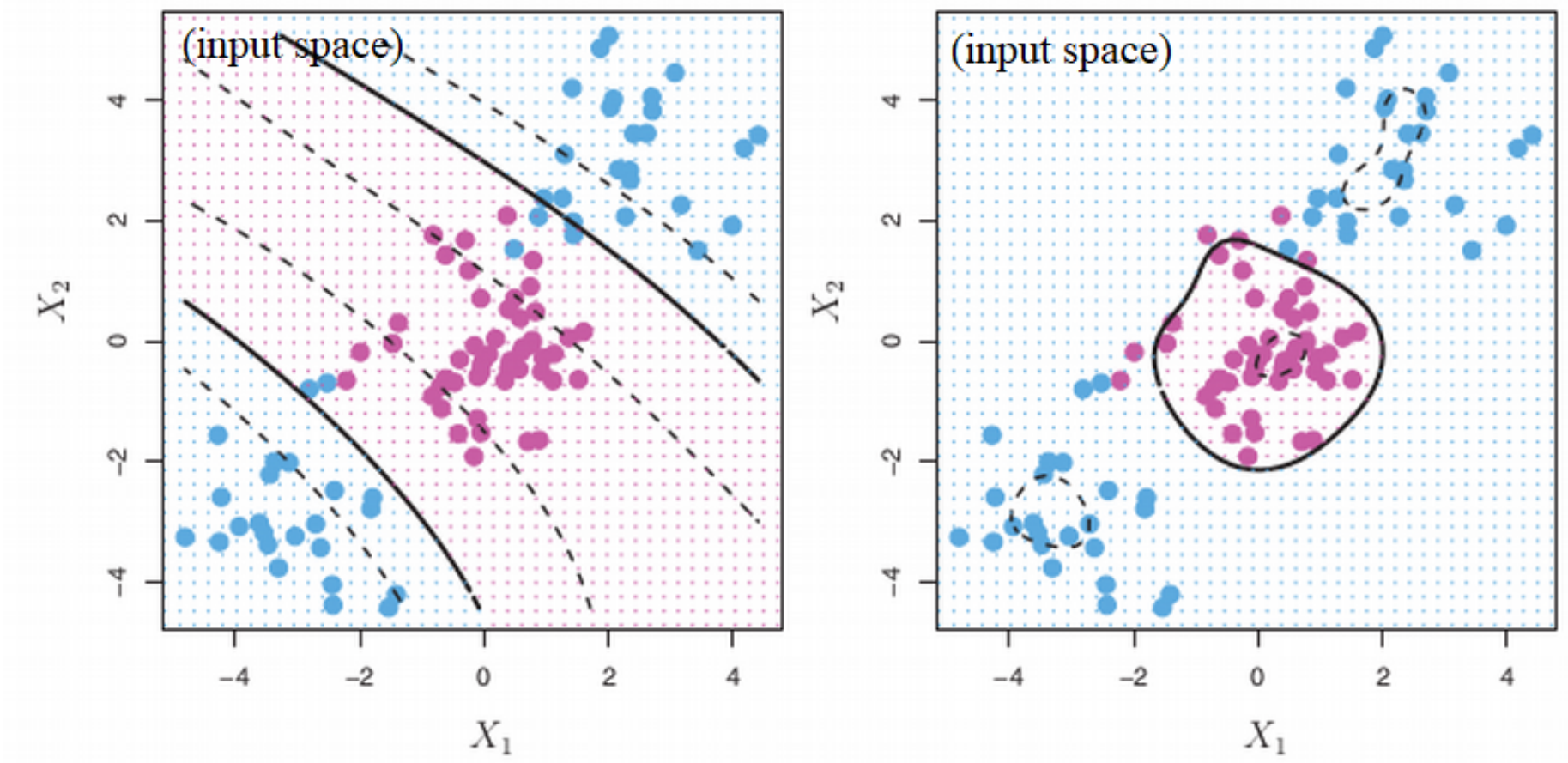


FIGURE 9.9. Left: *An SVM with a polynomial kernel of degree 3 is applied to the non-linear data from Figure 9.8, resulting in a far more appropriate decision rule.* Right: *An SVM with a radial kernel is applied. In this example, either kernel is capable of capturing the decision boundary.*

... se puede demostrar que (BackUp)

- En el feature space:

$$\widehat{\beta} = \sum_{i \in \text{SV}} \lambda_i y_i \Phi(\mathbf{x}_i) \text{ y por tanto } R(\mathbf{x}) = \text{sign}[\widehat{\beta}^T \Phi(\mathbf{x}) + \widehat{\beta}_0].$$

- Utilizando el planteo dual:

$$R(\mathbf{x}) = \text{sign}\Big[\underbrace{\sum_{i \in \text{SV}} \lambda_i y_i K_\sigma(\mathbf{x}_i, \mathbf{x}) + \widehat{\beta}_0}_{\widehat{f}(\mathbf{x})}\Big].$$

- En el espacio original (input), la SVM tiene asociada una **frontera de clasificación** que **no** es **lineal** (ver slide anterior).

$$H(\widehat{f}, \sigma, C) = \{\mathbf{x} : \widehat{f}(\mathbf{x}) = \sum_{i \in \text{SV}} \lambda_i y_i K_\sigma(\mathbf{x}_i, \mathbf{x}) + \widehat{\beta}_0 = 0\}.$$

Informalmente, el hiperplano que separa en dos partes el espacio de covariables ya no es una función lineal (un hiperplano).

Recapitulación

+ Aprendemos parámetros vía optimización cuadrática y convexa.
  - No hay mínimos locales :)
  - El algoritmo escala sin problemas en p.
  - Estandarizar features acelera la velocidad de convergencia.
+ Flexibilidad: Podemos modelar problemas no lineales.
+ Podemos fitear el modelo aún cuando $p \gg n$.
+ No hay especificaciones fuertes.

o Atípicos: Funciones de riesgo *robustas*.

o One-hot o método equivalente para features cualitativos.

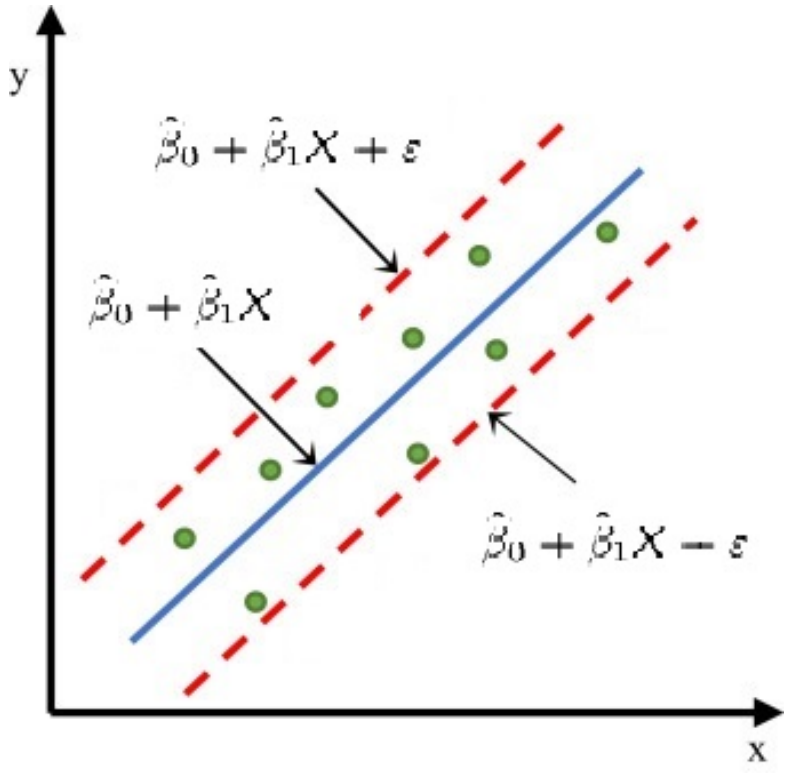
- Modelo caja negra (¿importancia de cada feature?).
- No gestiona de forma automática los datos faltantes.

Agenda

Clasificación con Máquinas de Vector Soporte

Regresiones con Máquinas de Vector Soporte

- El modelo más simple con svm plantea: $Y = \underbrace{\beta_0 + \beta_1 X}_{\text{Modelo para } f(X)} + \delta.$



- Dada una muestra de train y un parámetro $\varepsilon > 0$, aprendemos (β_0, β_1) resolviendo el siguiente problema:

$$\min_{b_0, b_1} \frac{1}{2}\|b_1\|^2, \text{ s.a: } |y_i - b_0 - b_1 x_i| \leq \varepsilon.$$

- Vía optimización convexa y cuadrática:

$$\widehat{f}(X) = \widehat{\beta}_0 + \widehat{\beta}_1 X$$

  - No se corresponde con OLS.

- Las restricciones garantizan un ECM pequeño (si ε es pequeño).
- La función objetivo intenta *regularizar* el modelo.
- Veamos ahora el planteo más general.

- SVM (y kernel) lineal: $Y = \underbrace{\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p}_{\text{Modelo para } f(X_1,\ldots,X_p)} + \delta$.

- Dados $\varepsilon > 0$ y una muestra de train $S_n = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, aprendemos los parámetros del modelo de forma tal que:

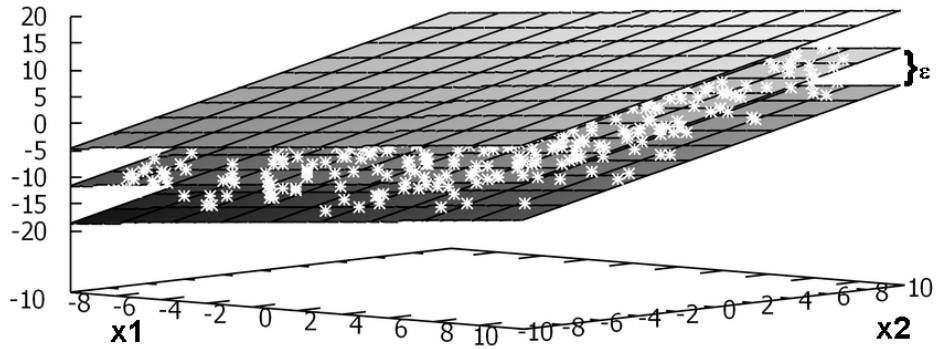


Figure: Para $i = 1, \ldots, n : |y_i - b_0 - \mathbf{b}^T \mathbf{x}_i| \leq \varepsilon$.

- Problema de optimización convexo para aprender $(\beta_0, \ldots, \beta_p)$.

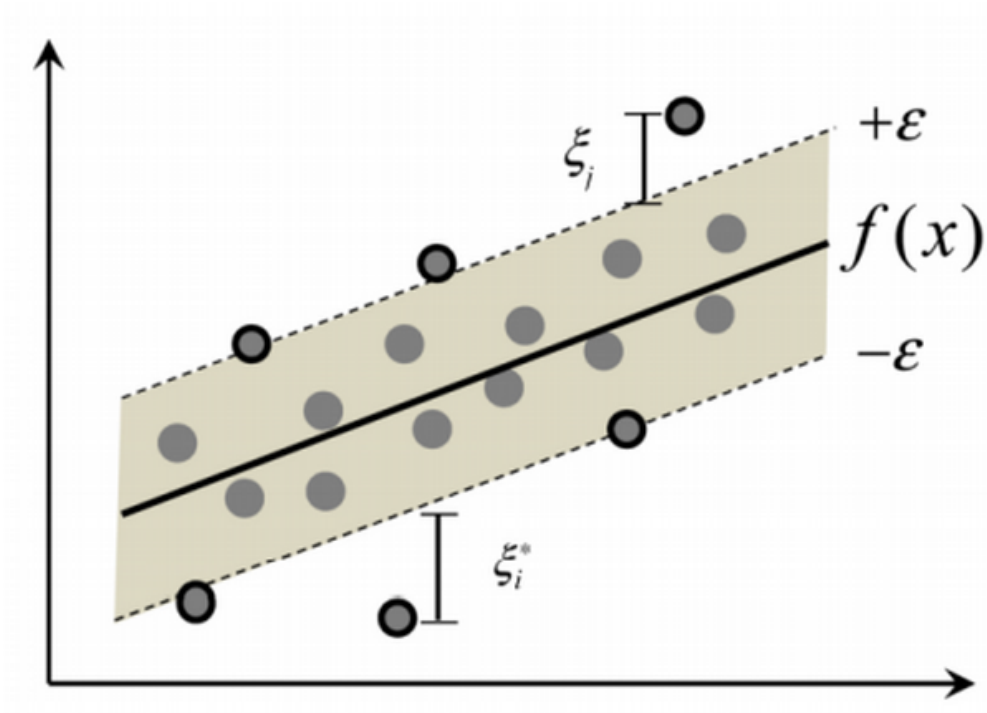
- ε es un hiperparámetro del modelo.

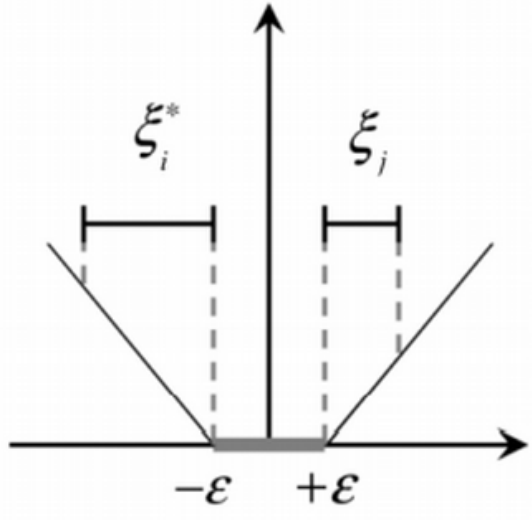
- La solución puede no existir si ε es pequeño.

A. Smola: *A Tutorial on Support Vector Regression*. Stat & Comp. (2003).

Relajación del modelo de regresión

- Al igual que en clasificación, introducimos parámetros de holgura $\{\xi_1, \ldots, \xi_n\}$ que nos permiten aprender los parámetros del modelo permitiendo que algunas observaciones estén a una distancia mayor que ε de la función de regresión.





Aprendizaje de parámetros (BackUp)

$$\min_{b_0,\mathbf{b},\boldsymbol{\xi},\boldsymbol{\xi}^*} C\sum_{i=1}^{n}(\xi_i+\xi_i^*)+\frac{1}{2}\|\mathbf{b}\|^2$$

subject to

$$y_i-(b_0+\mathbf{b}^T\mathbf{x}_i)\leq\varepsilon+\xi_i,\ i=1,\ldots,n.$$
$$(b_0+\mathbf{b}^T\mathbf{x}_i)-y_i\leq\varepsilon+\xi_i^*,\ i=1,\ldots,n.$$
$$\xi_i\geq 0,\quad \xi_i^*\geq 0,\ i=1,\ldots,n.$$

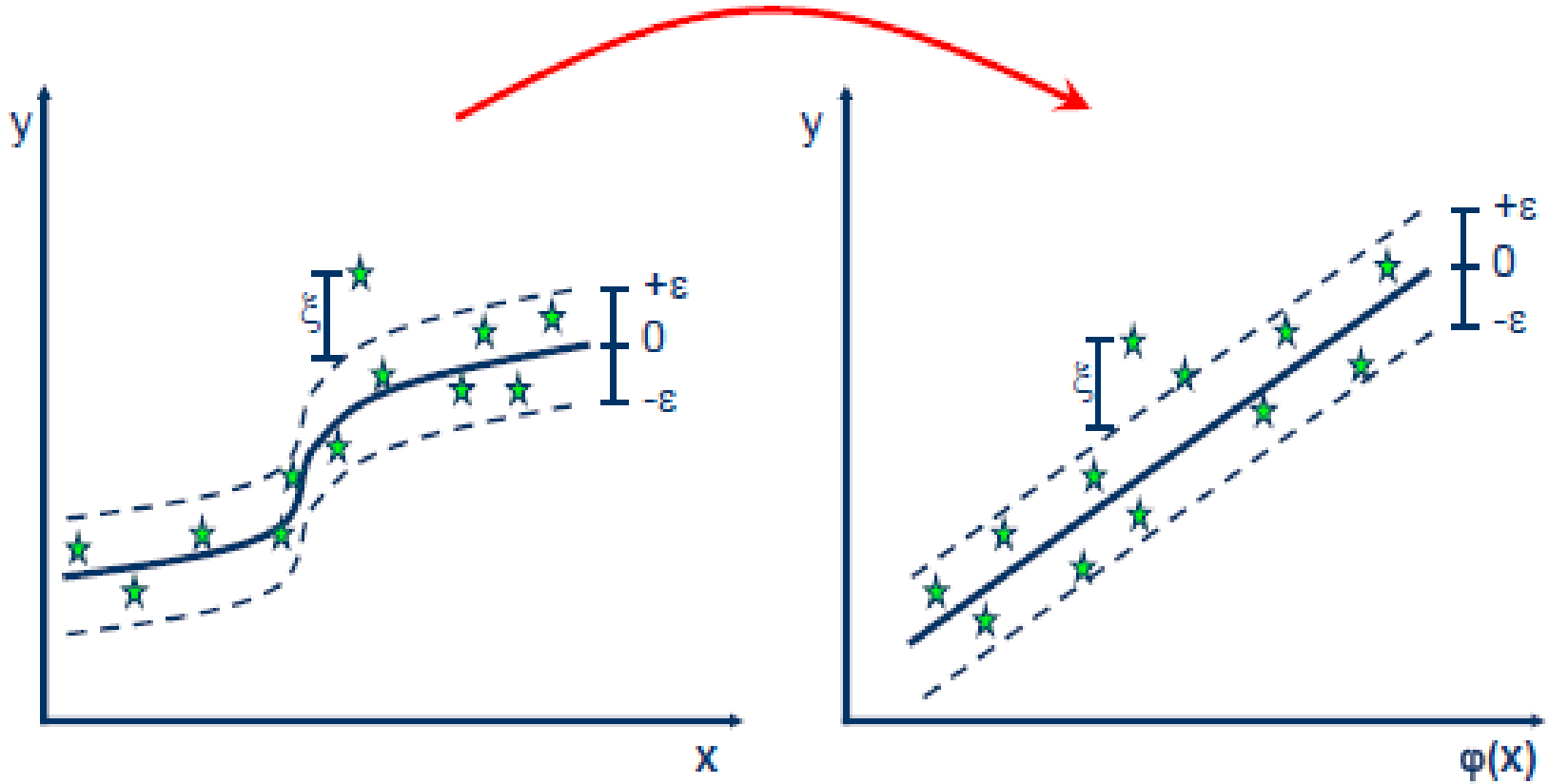
- A. Smola (A tutorial on support vector regression):

  C determines the trade-off between the flatness of f and the amount up to which deviations larger than ε are tolerated.

- $\uparrow C$: *Pendiente 'cambiante'* ($+$ varianza).
- $\downarrow C$: *Poca pendiente* ($+$ bias).
- Deslinealizamos el modelo introduciendo un K_σ de parámetro σ.
- Aprendemos (C,σ) por validación cruzada.

Extensiones al caso no lineal

- $\Phi : \mathbb{R}^p \to \mathbb{R}^q$ (con $q \gg p$ modelo lineal en el feature space).



$$\widehat{f}_\sigma(\mathbf{x}) = \sum_{i=1}^{n} c_i \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}) \rangle + \hat{\beta}_0 = \sum_{i=1}^{n} c_i K_\sigma(\mathbf{x}_i, \mathbf{x}) + \hat{\beta}_0.$$

- Las constantes $\{c_i\}_{i=1}^n$ se aprenden vía optimización convexa.

Modelando precios de pólizas de seguro

```
age: edad del beneficiario de la póliza.
sex: género declarado por el asegurado.
bmi: índice de masa corporal.
children: número de hijos cubiertos en la póliza.
smoker: si quien contrata la póliza es fumador o no.
region: lugar de residencia del beneficiario.
insuranceclaim: reclamación de costos (1 = si y 0 = no).

charges: costo registrado de los tratamientos médicos.
```

Dividí los datos en train y test: Aprende los parámetros de la SVM por VC con train (selección de modelo) y estima el ECM del modelo seleccionado sobre los datos de test. Este modelo servirá de benchmark para comprar contra las redes neuronales profundas (próxima clase).