Unidad IV - Máquinas de Soporte Vectorial (SVM)

Germán Braun

Facultado de Informática - Universidad Nacional del Comahue

german.braun@fi.uncoma.edu.ar

26 de septiembre de 2025

Agenda

ML en la práctica

¡Recordatorio!

El aprendizaje automático es un proceso de prueba y error.

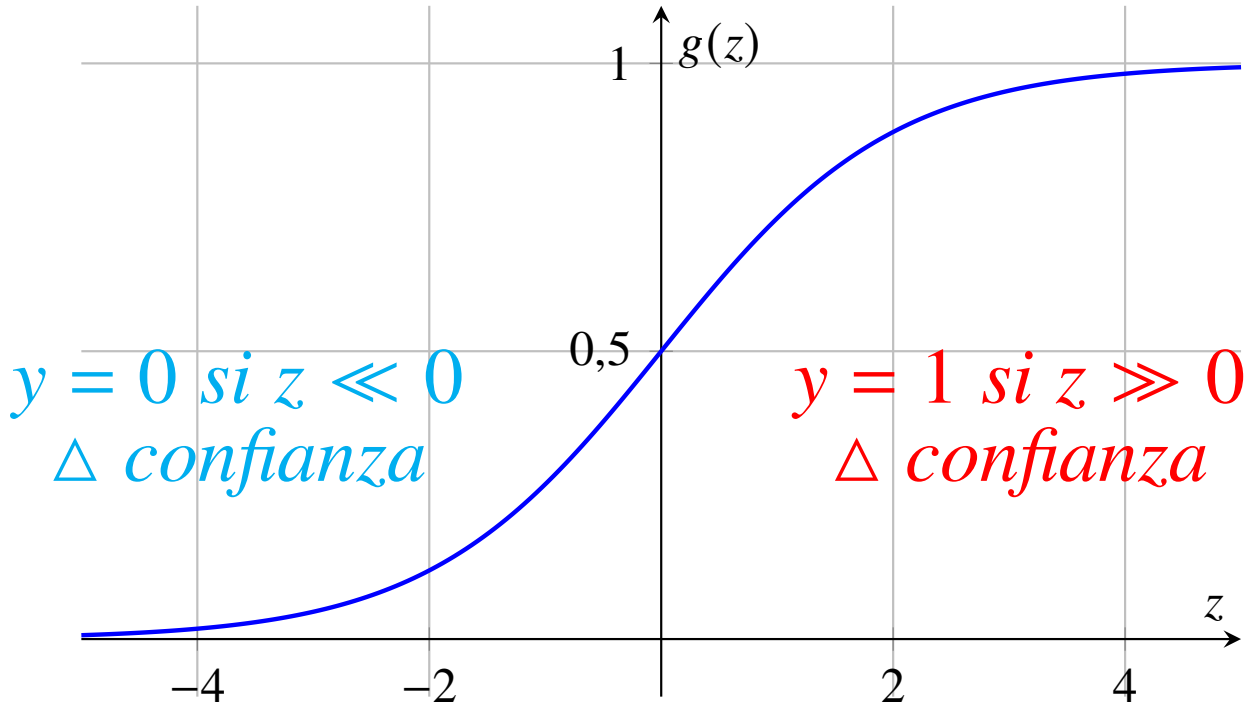
Intuición

Intuición I - Grado de Confianza

- Analicemos la función de regresión logística ...

Intuición I - Grado de Confianza

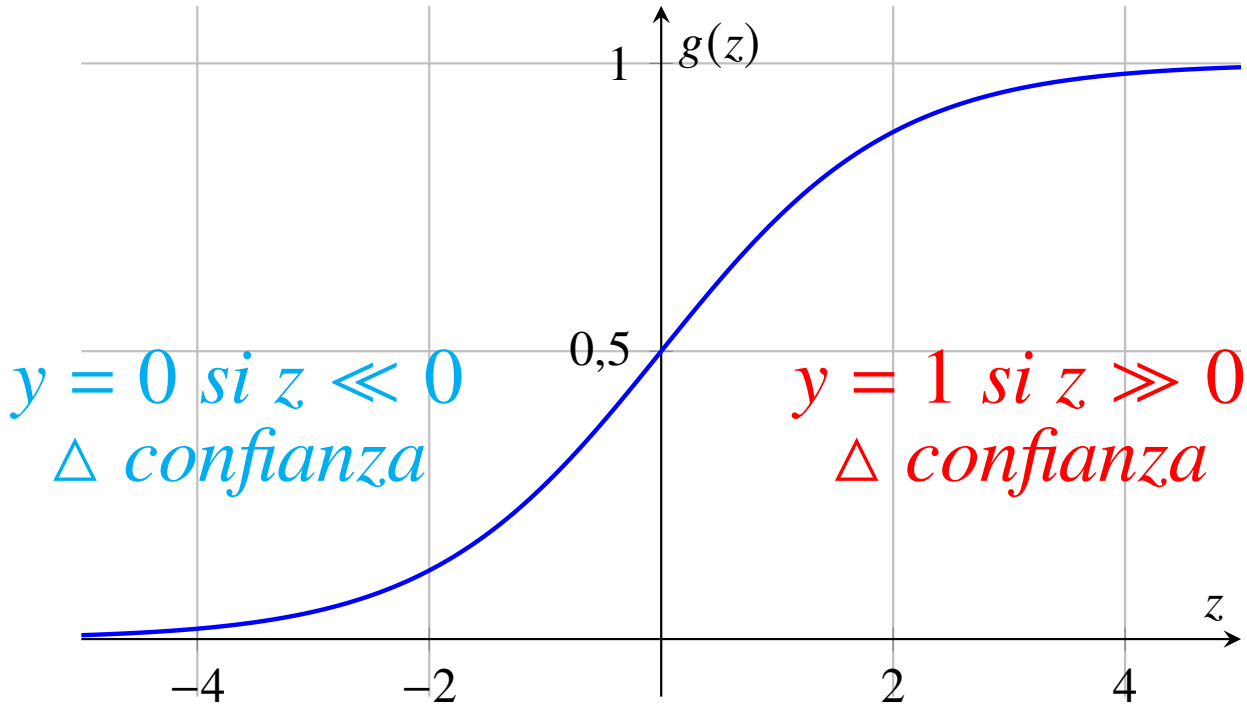
- Analicemos la función de regresión logística ...



- $y = 1 \iff h(x) \geq 0{,}5$ o $z \geq 0$. Por lo tanto, mientras más grande sea z, más alto será el grado de confianza de que $y = 1$. Entonces $y = 1$ si $z \gg 0$

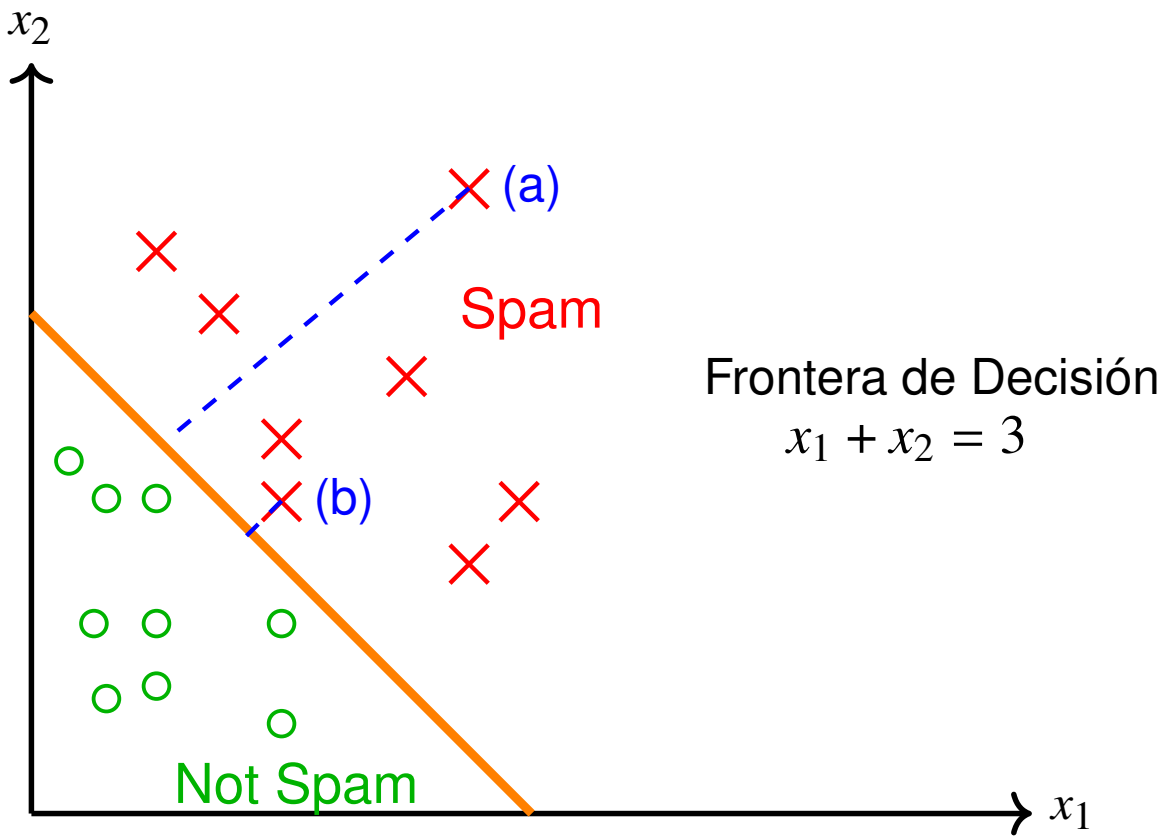
Intuición I - Grado de Confianza

- Analicemos la función de regresión logística ...



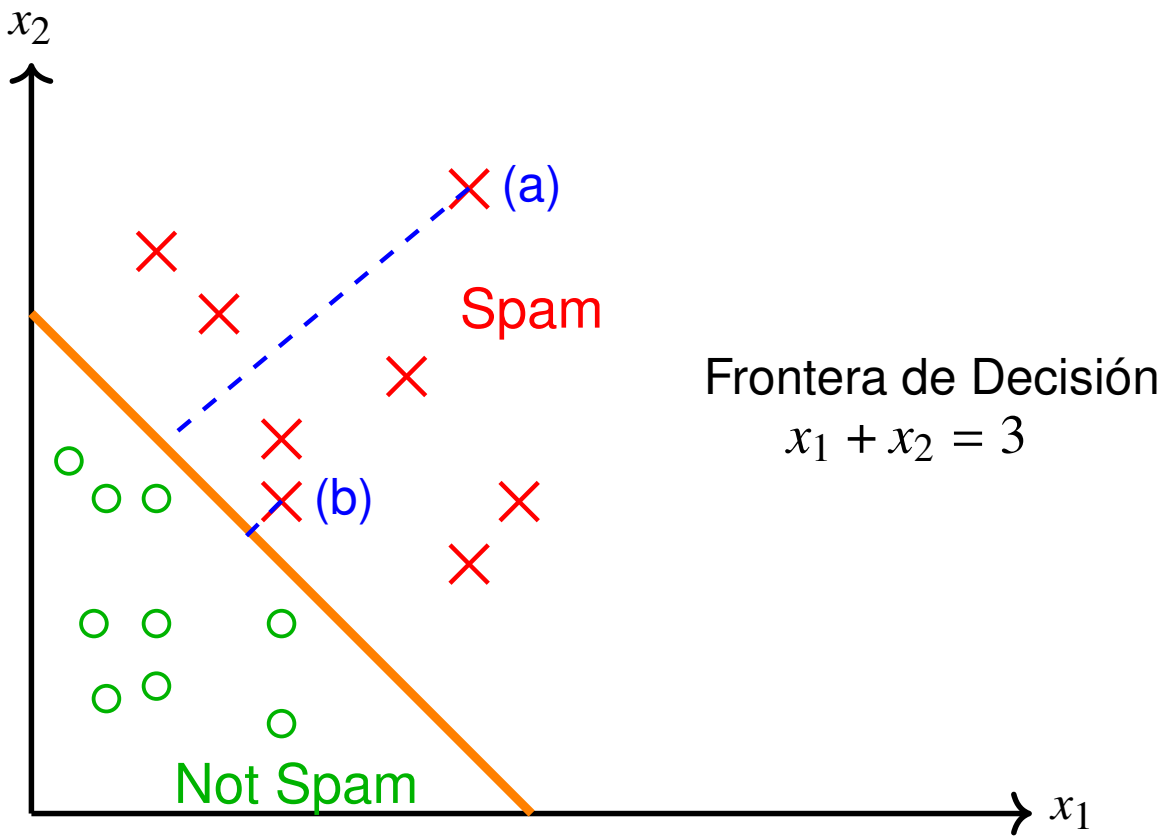
- $y = 1 \iff h(x) \geq 0{,}5$ o $z \geq 0$. Por lo tanto, mientras más grande sea z, más alto será el grado de confianza de que $y = 1$. Entonces $y = 1$ si $z \gg 0$
- De manera similar, podemos decir $y = 0$ si $z \ll 0$

Intuición II - Grado de Confianza



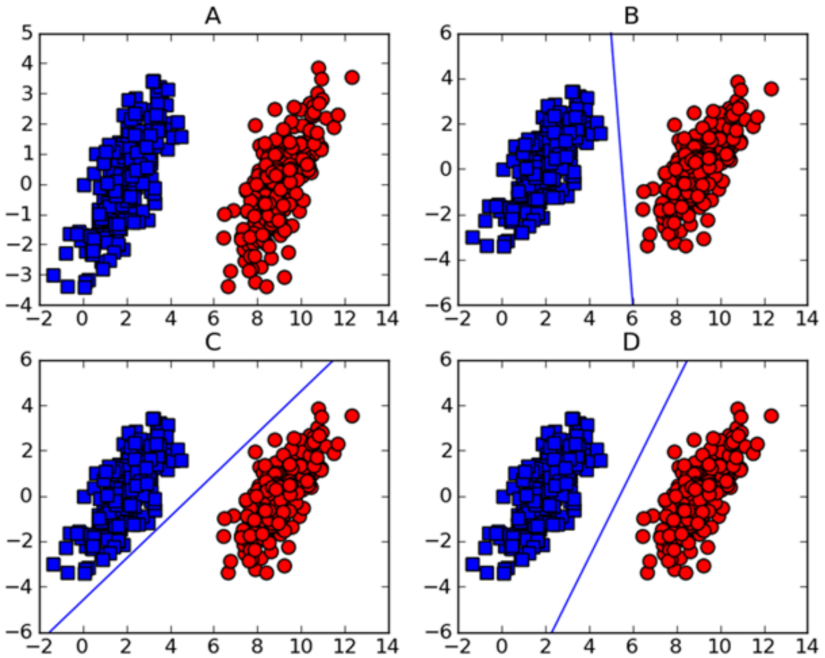
- El punto (a) está “lejos” del límite de decisión, entonces la predicción sobre ese valor tendrá un alto grado de confianza

Intuición II - Grado de Confianza



- El punto (a) está "lejos" del límite de decisión, entonces la predicción sobre ese valor tendrá un alto grado de confianza
- De manera opuesta, el punto (b) se encuentra "cerca" del límite de decisión. Por lo tanto, si bien la predicción es 1 (spam), ajustes en el límite podrían causar que la clasificación cambie a 0

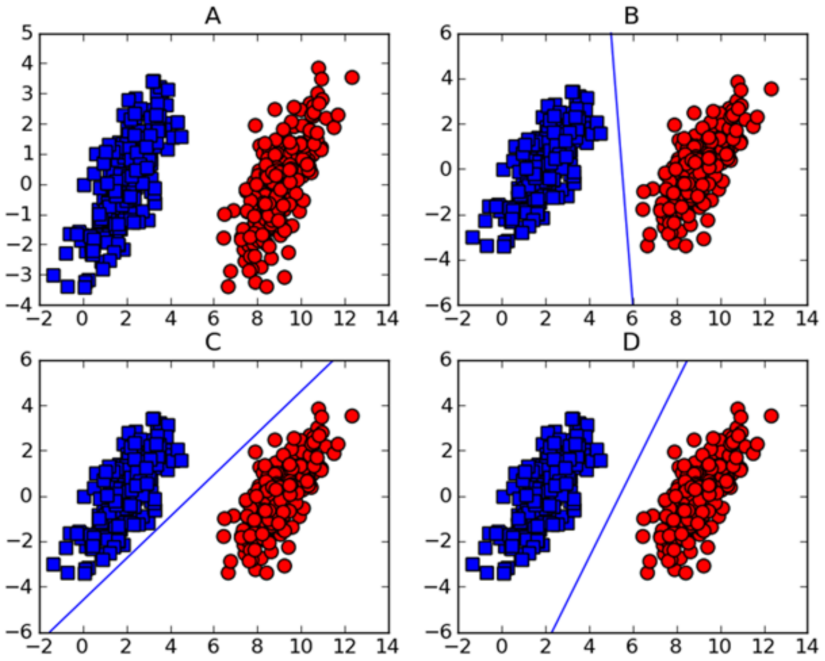
Intuición III



Separables linealmente

- Hiperplano → Frontera de Decisión

Intuición III

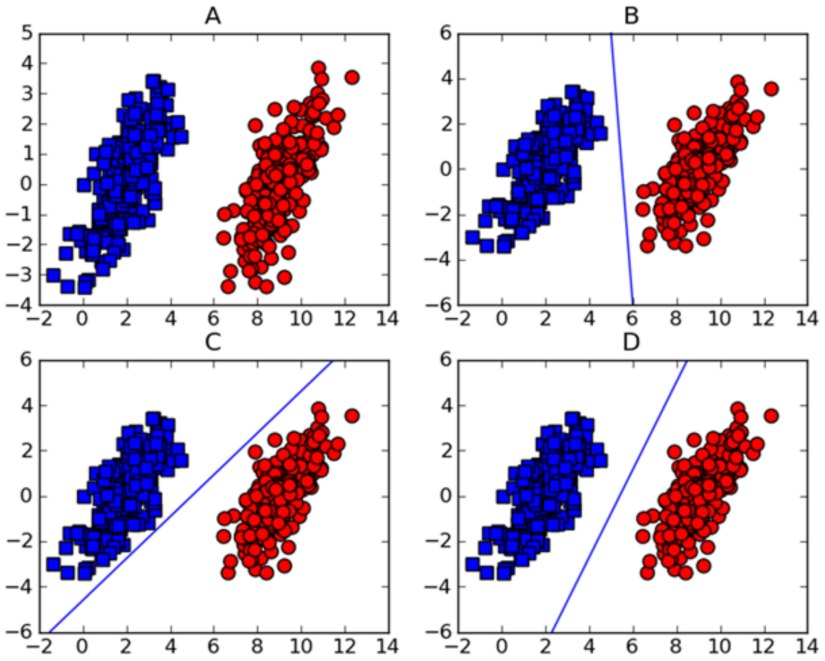


Separables linealmente

- Hiperplano → Frontera de Decisión

- La distancia entre la frontera y el punto más cercano es el **margen**

Intuición III

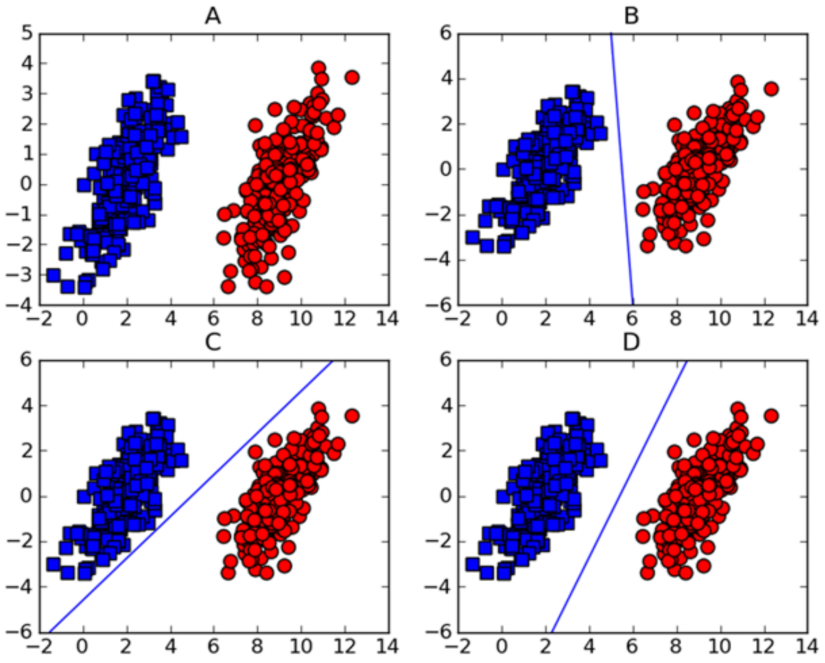


Separables linealmente

- Hiperplano → Frontera de Decisión

- La distancia entre la frontera y el punto más cercano es el **margen**
- El clasificador debe buscar que este margen sea “el mayor posible” → mayor grade de confianza en la predicción!

Intuición III

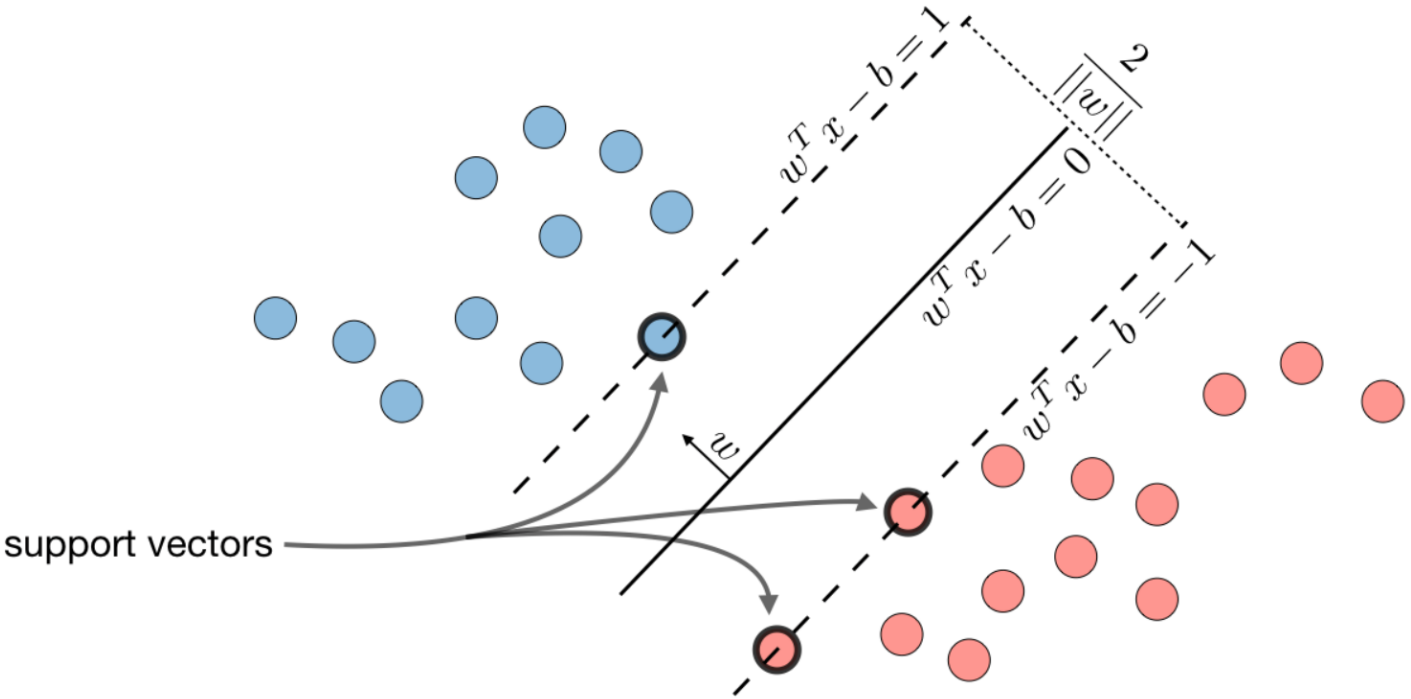


Separables linealmente

- Hiperplano → Frontera de Decisión

- La distancia entre la frontera y el punto más cercano es el **margen**
- El clasificador debe buscar que este margen sea “el mayor posible” → mayor grade de confianza en la predicción!
- Los puntos más cercanos al hiperplano de separación se denoninan **vectores de soporte**

Vectores de Soporte

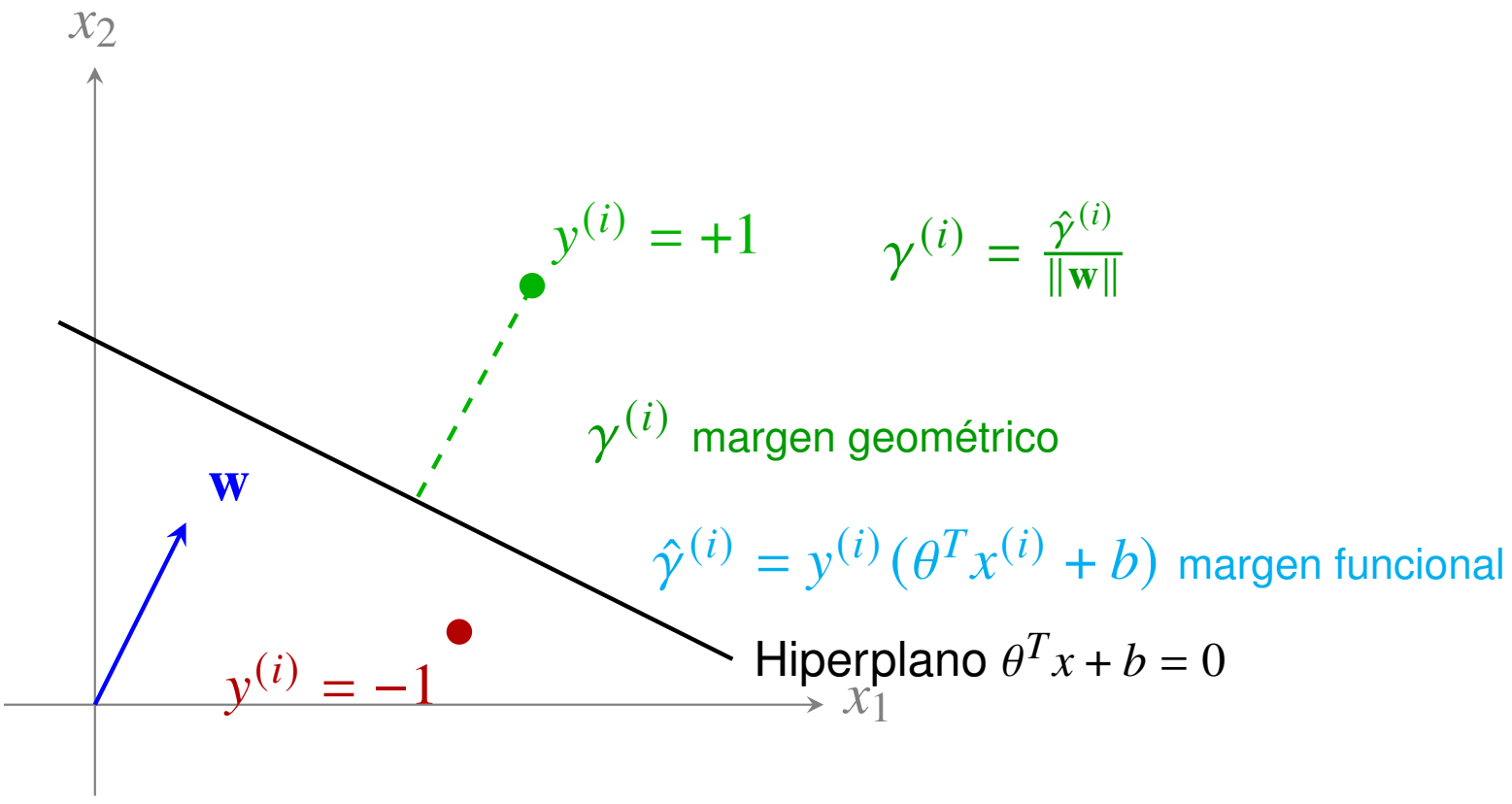


(*) imagen de

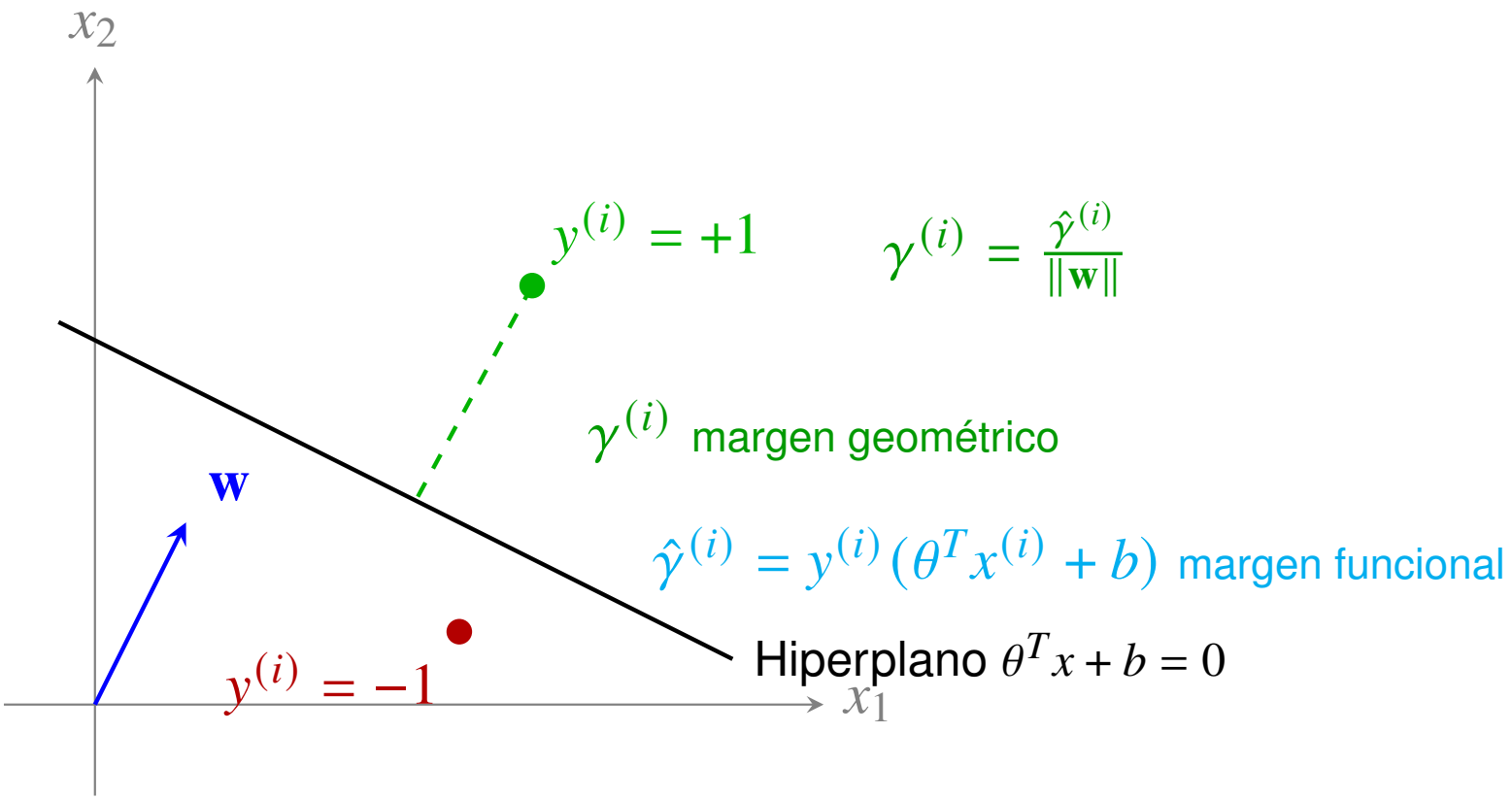
`https://stanford.edu/~shervine/teaching/cs-229/cheatsheet-supervised-learning`

Márgenes

Márgenes [Cortes,1995]

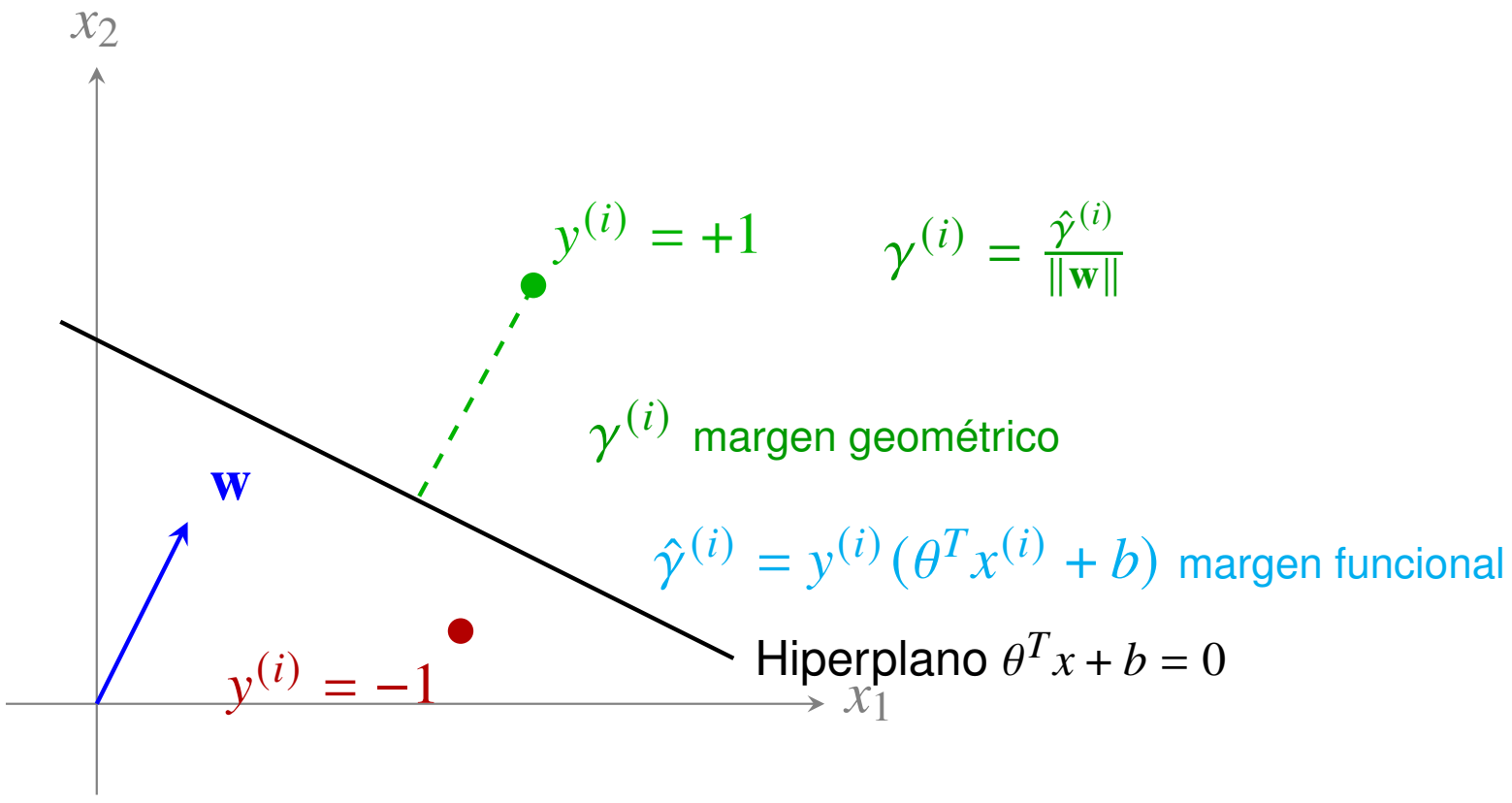


Márgenes [Cortes,1995]



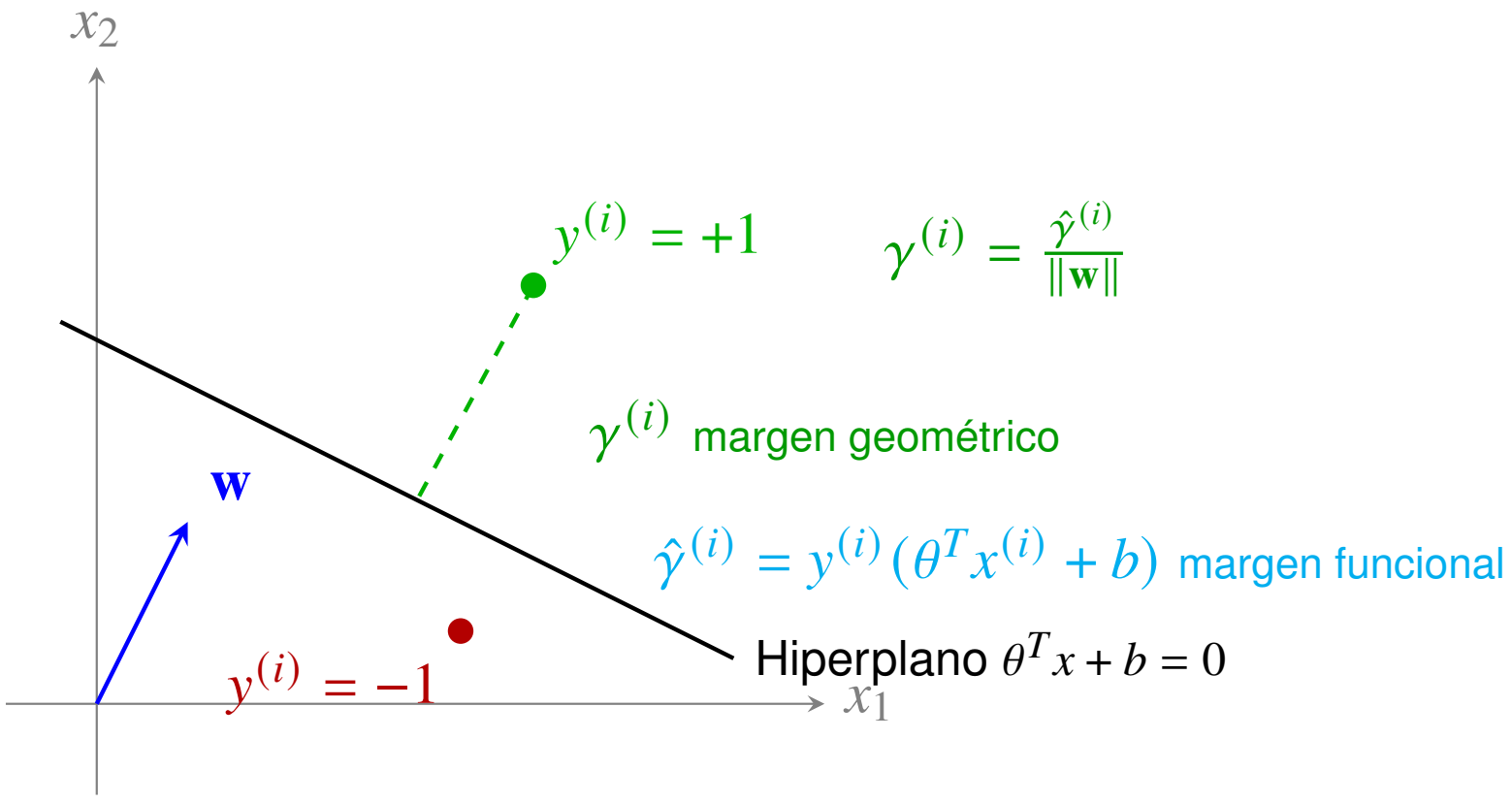
- El **margen funcional** representa “que tan bien clasificado” está un ejemplo de entrenamiento $y^{(i)}$

Márgenes [Cortes,1995]



- El **margen funcional** representa “que tan bien clasificado” está un ejemplo de entrenamiento $y^{(i)}$
- El **margen geométrico** es la **distancia real** de $y^{(i)}$ con respecto a la frontera de decisión

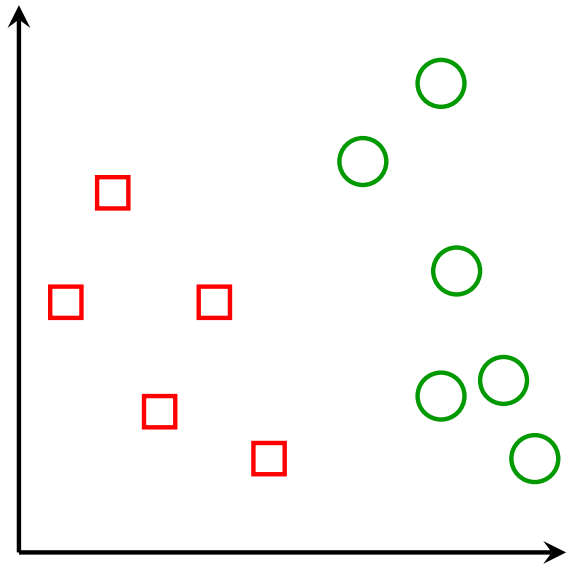
Márgenes [Cortes,1995]



- El **margen funcional** representa "que tan bien clasificado" está un ejemplo de entrenamiento $y^{(i)}$
- El **margen geométrico** es la **distancia real** de $y^{(i)}$ con respecto a la frontera de decisión
- *El objetivo del SVM es encontrar una frontera de decisión que maximice el margen (geométrico)*

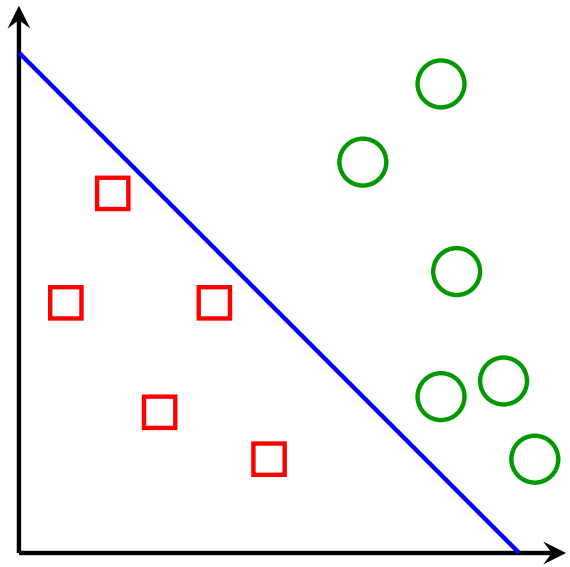
Hiperplano óptimo

Hiperplano óptimo



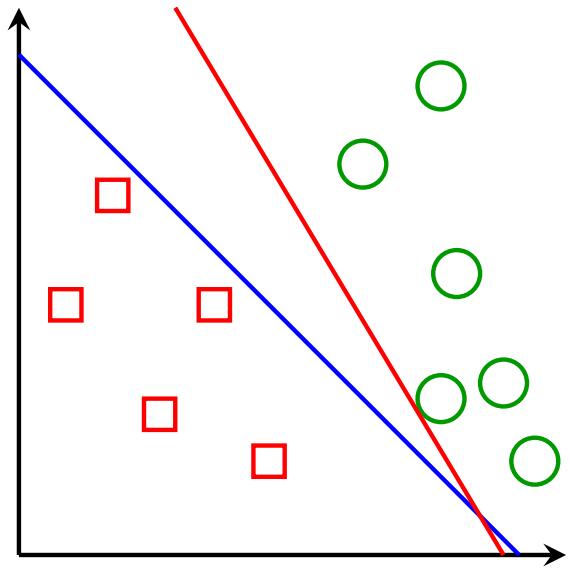
Hiperplano óptimo

Posibles hiperplanos



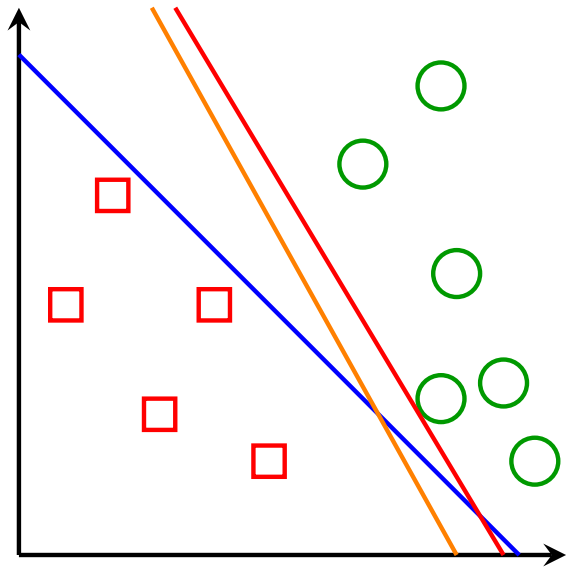
Hiperplano óptimo

Posibles hiperplanos



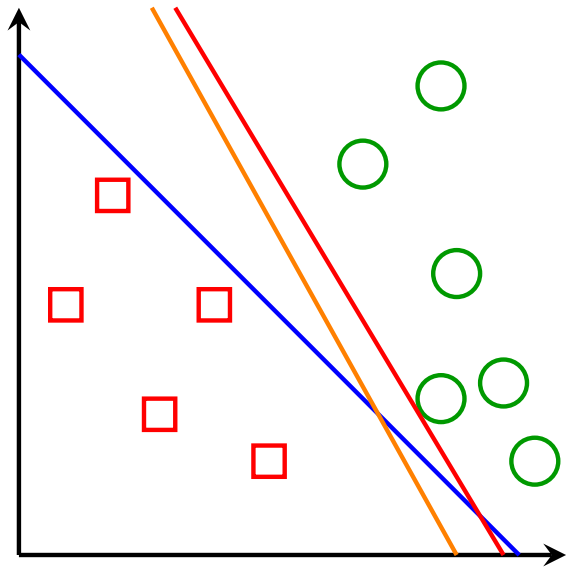
Hiperplano óptimo

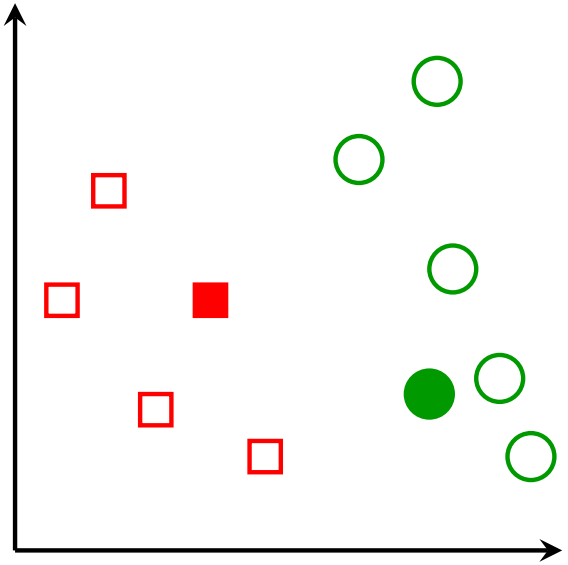
Posibles hiperplanos



Hiperplano óptimo

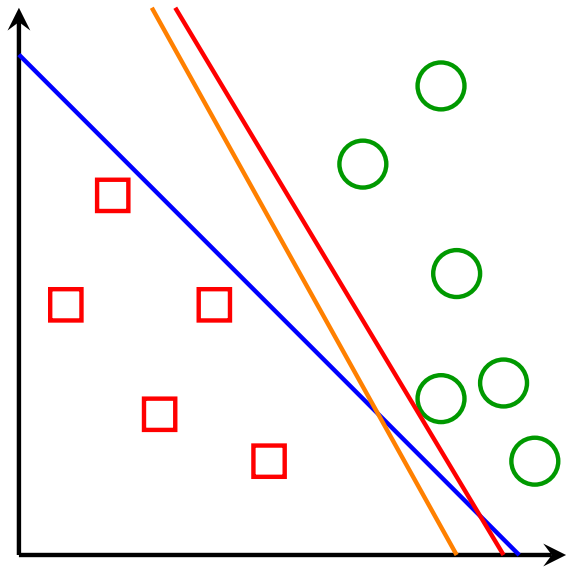
Posibles hiperplanos

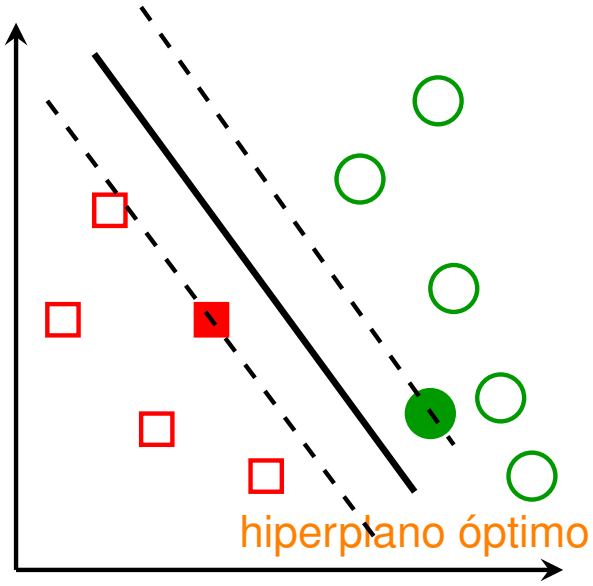




Hiperplano óptimo

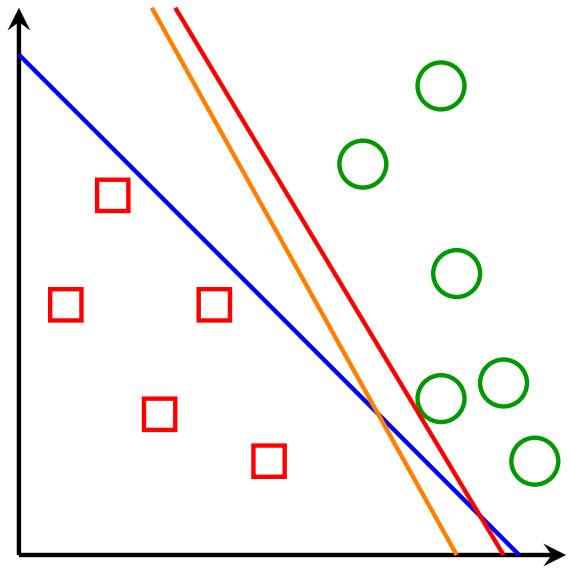
Posibles hiperplanos

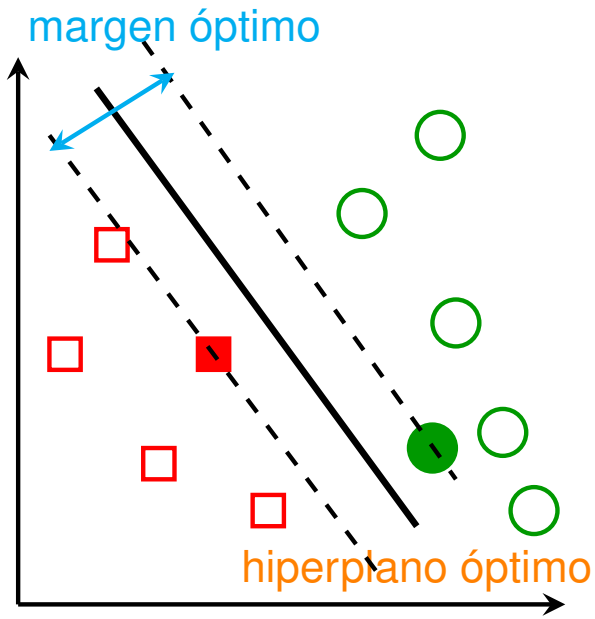




Hiperplano óptimo

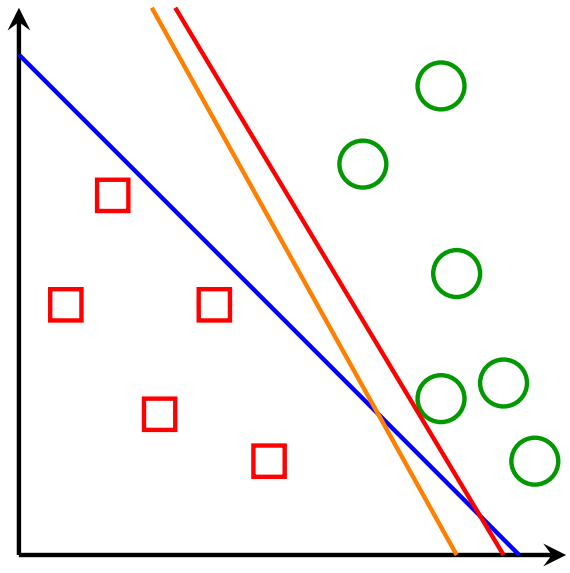
Posibles hiperplanos

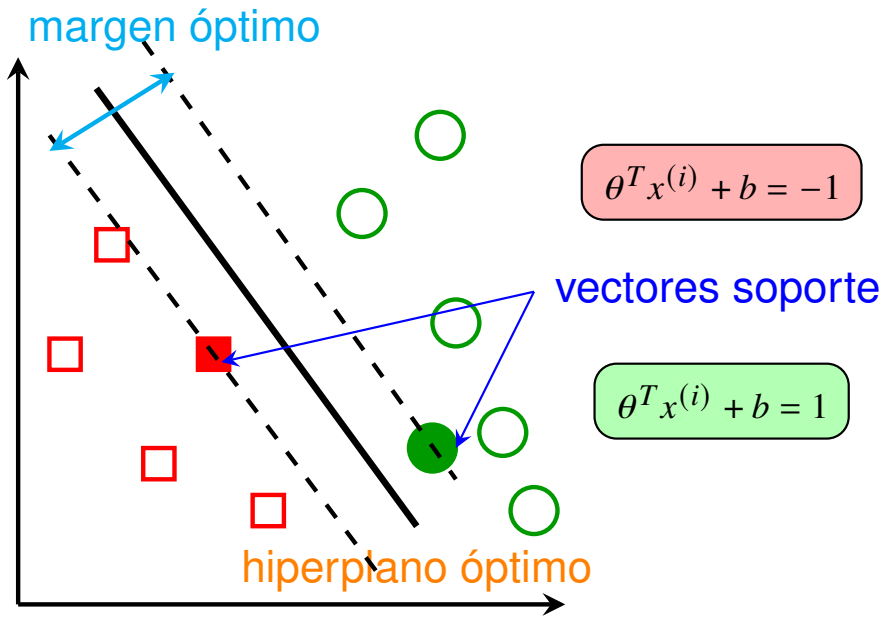




Hiperplano óptimo

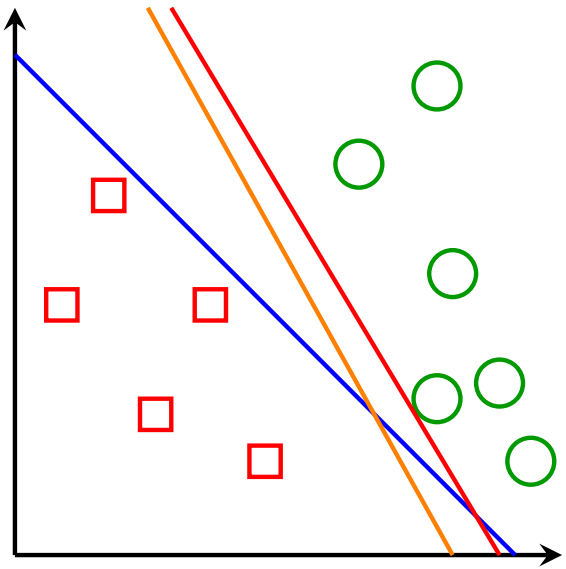
Posibles hiperplanos

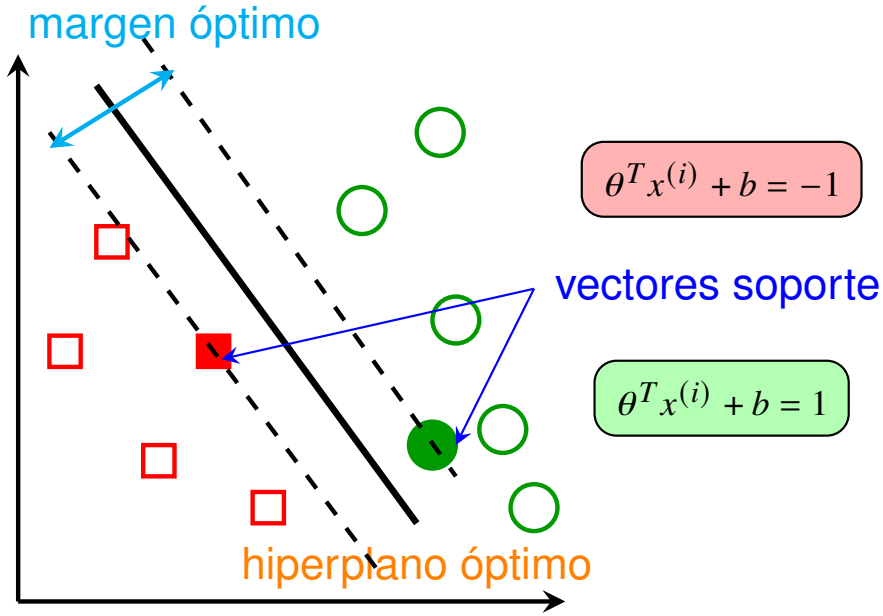




Hiperplano óptimo

Posibles hiperplanos



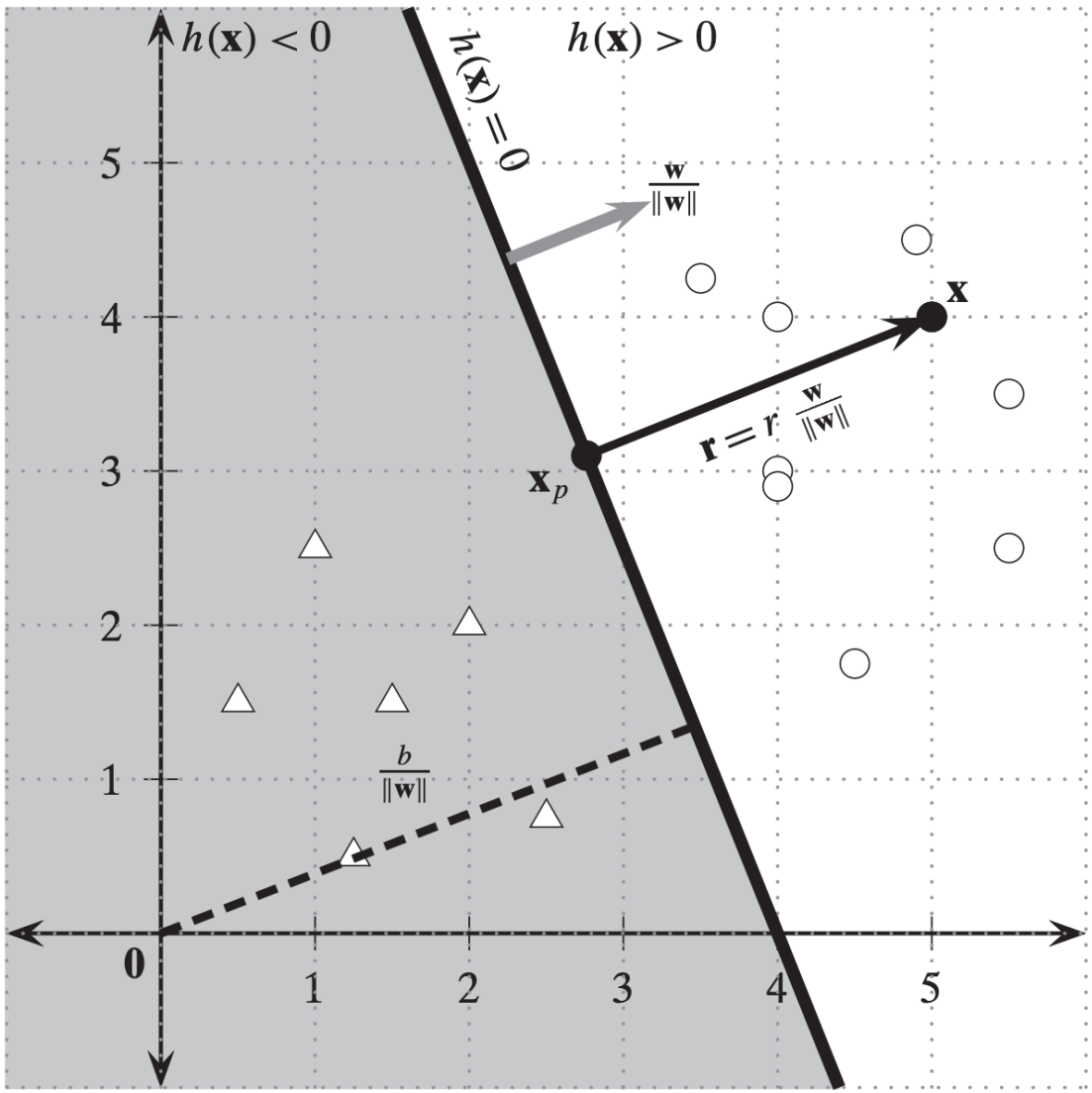


- Los vectores de soporte son especiales debido a que son los ejemplos de entrenamiento que definen el margen máximo del hiperplano al set de datos.
- El hiperplano óptimo es la frontera de decisión con el máximo margen entre los vectores de las dos clases
- El clasificador de márgenes óptimo (*Optimal margin classifier*) es la solución al problema de optimización:

$$min \frac{1}{2} ||w||^2 \; tal \; que \; (y^{(i)} \, \theta^T x^{(i)} + b) \geq 1$$

(*) https://stanford.edu/~shervine/teaching/cs-229/cheatsheet-supervised-learning

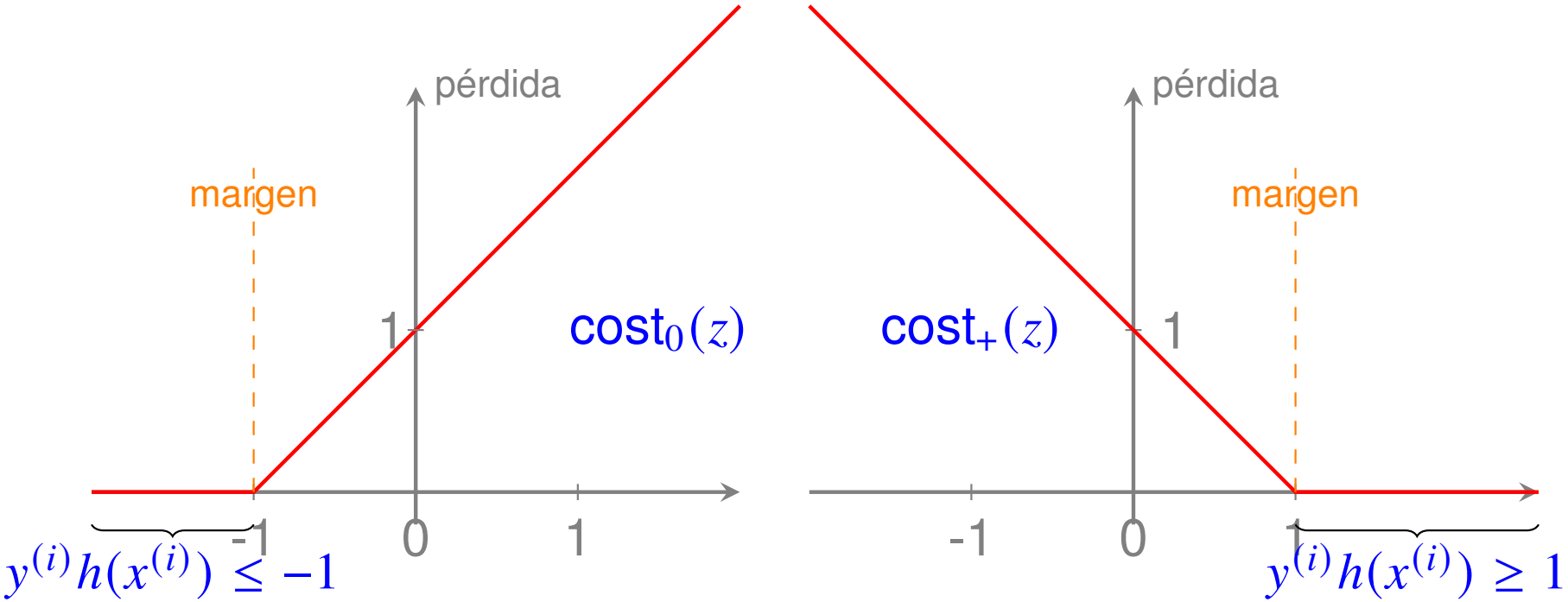
Geometría del hiperplano separador (final)



(*) Créditos de la imagen: https://titan.dcs.bbk.ac.uk/

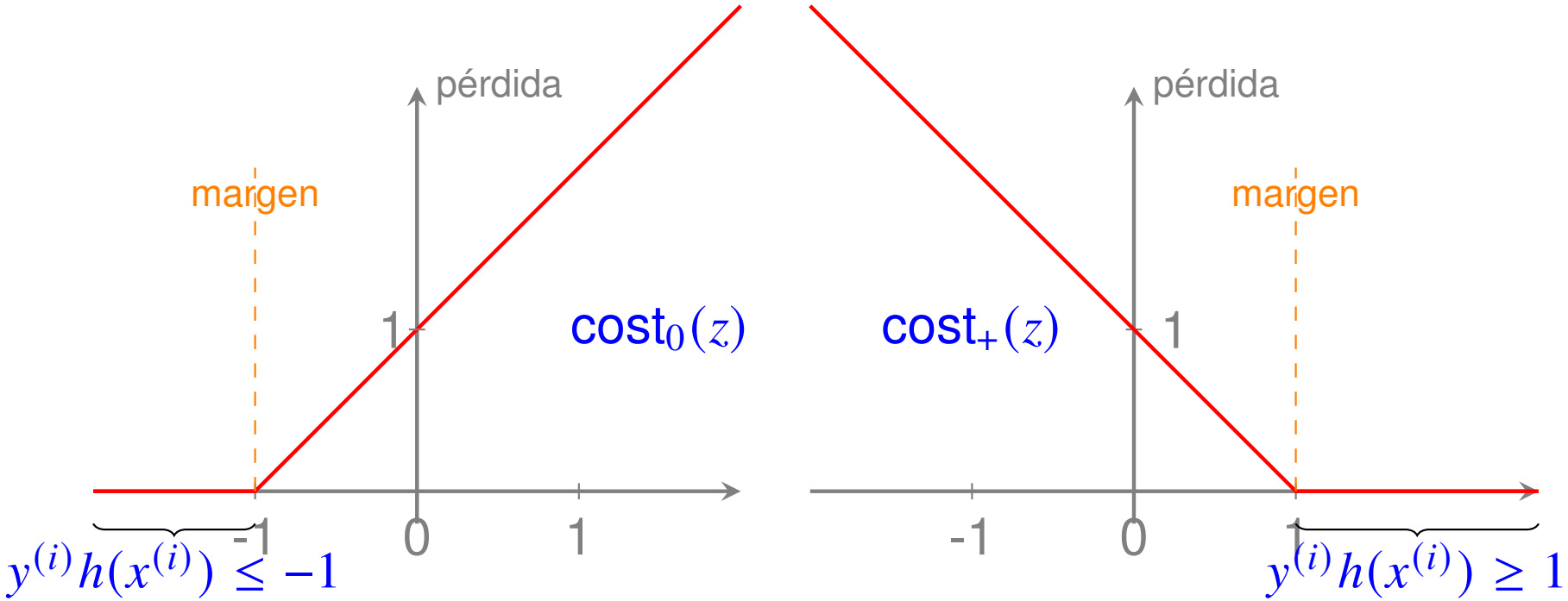
Función de Pérdida

Función de Pérdida en SVM



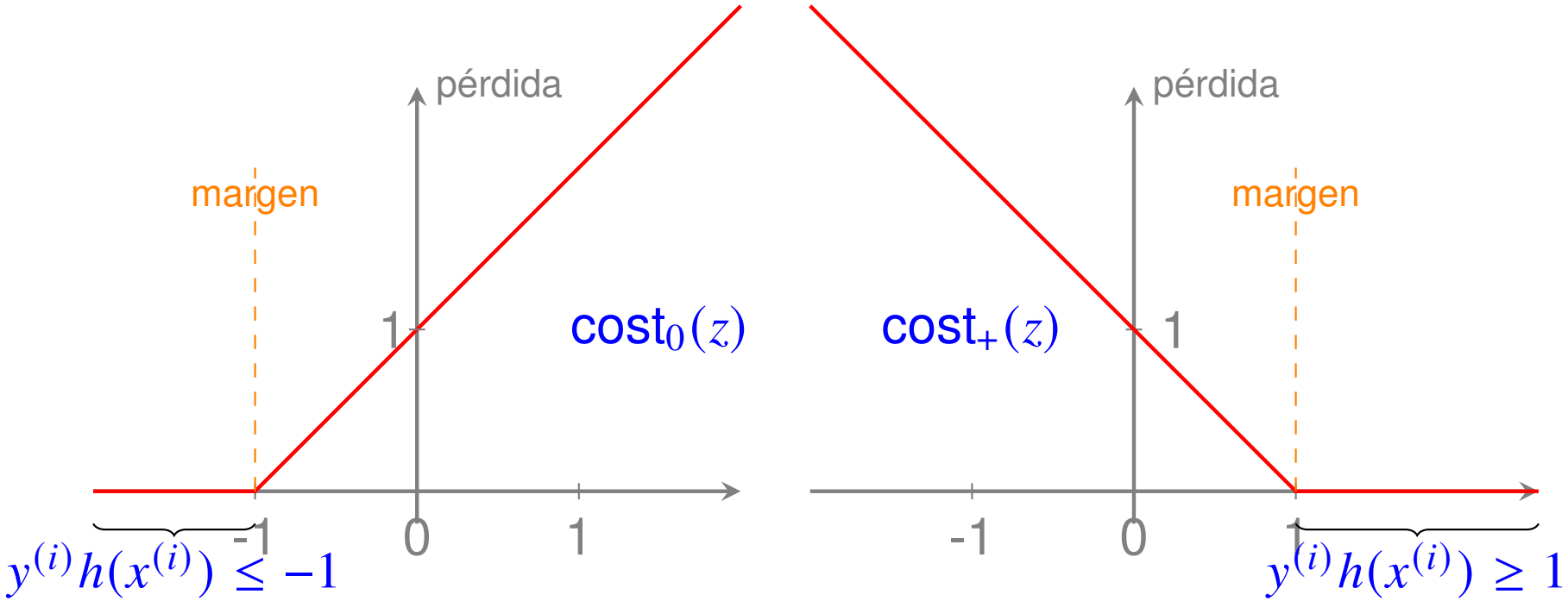
- El eje x representa correctitud de la predicción

Función de Pérdida en SVM



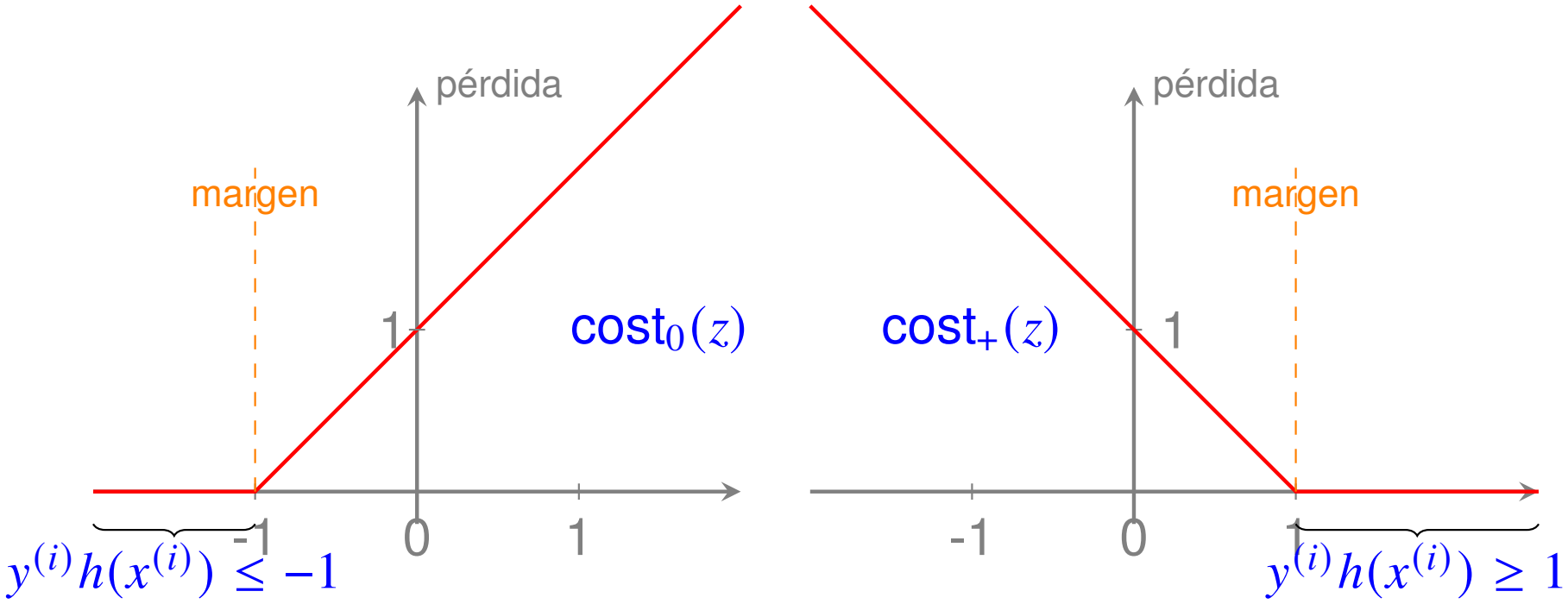
- El eje x representa correctitud de la predicción
- Si $y^{(i)}h(x^{(i)}) \geq 1$, la pérdida es 0

Función de Pérdida en SVM



- El eje x representa correctitud de la predicción
- Si $y^{(i)}h(x^{(i)}) \geq 1$, la pérdida es 0
- Si $y^{(i)}h(x^{(i)}) < 1$, hay pérdida (o penalidad)

Función de Pérdida en SVM



- El eje x representa correctitud de la predicción
- Si $y^{(i)}h(x^{(i)}) \geq 1$, la pérdida es 0
- Si $y^{(i)}h(x^{(i)}) < 1$, hay pérdida (o penalidad)

En la literatura, a esta función se la denomina *Hinge-Loss*

$$max(0, 1 - y^{(i)}h(x^{(i)}))$$

Regularización en SVM

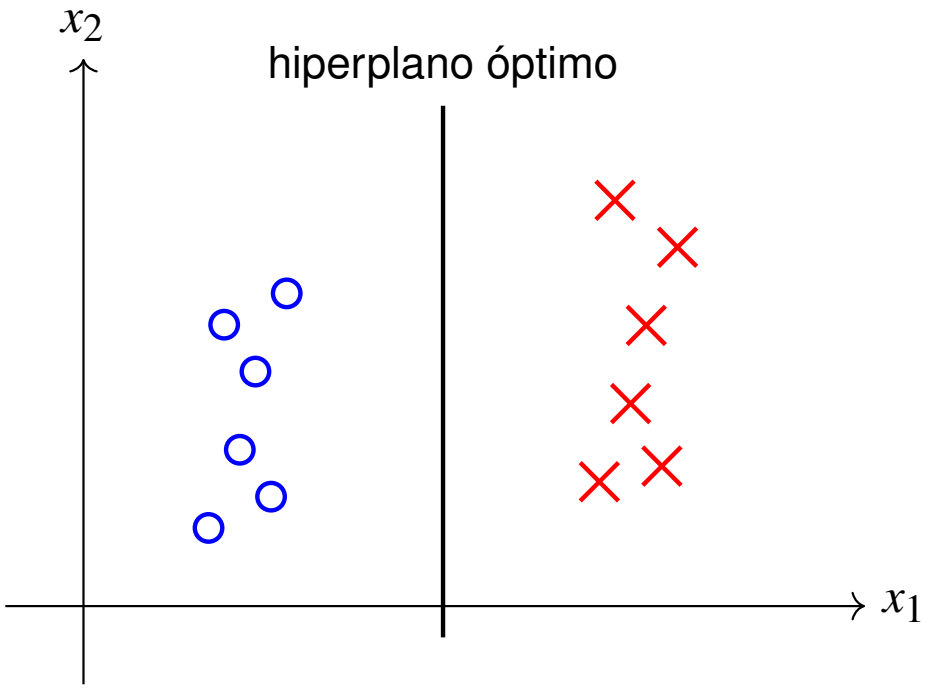
- La formalización del SVM (hasta ahora) asume que el dataset es **separable linealmente**

Regularización en SVM

- La formalización del SVM (hasta ahora) asume que el dataset es **separable linealmente**
- Sin embargo, encontrar el margen óptimo puede ser sensible a *outlier*

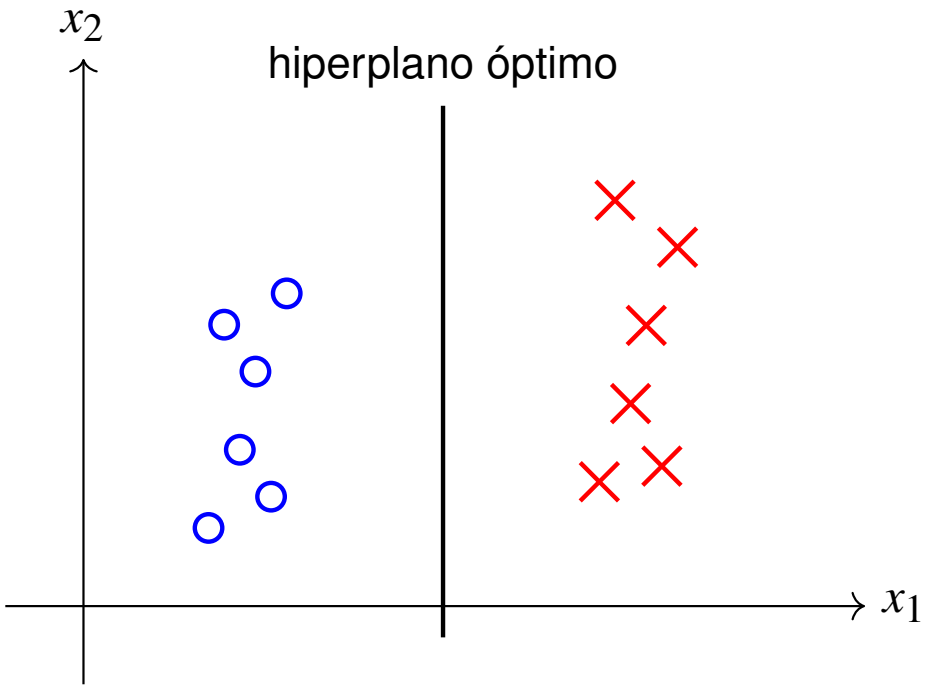
Regularización en SVM

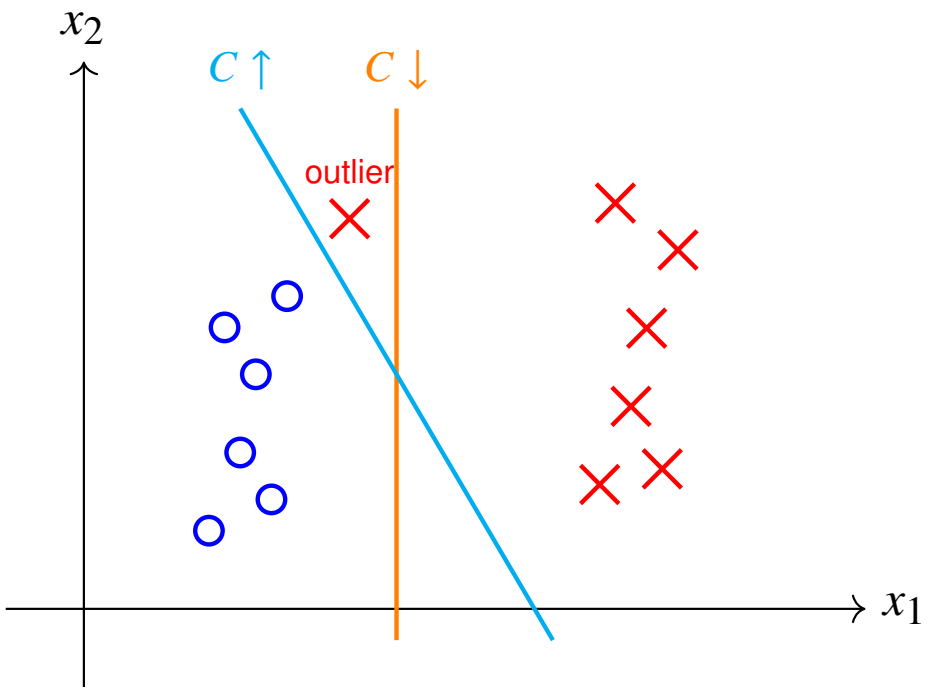
- La formalización del SVM (hasta ahora) asume que el dataset es **separable linealmente**
- Sin embargo, encontrar el margen óptimo puede ser sensible a *outlier*



Regularización en SVM

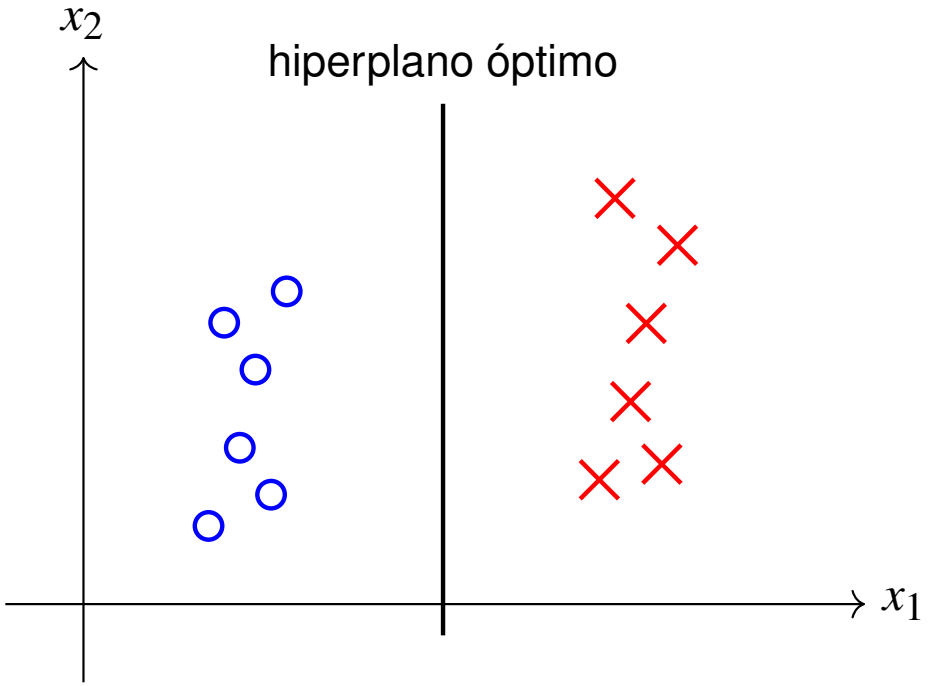
- La formalización del SVM (hasta ahora) asume que el dataset es **separable linealmente**
- Sin embargo, encontrar el margen óptimo puede ser sensible a *outlier*

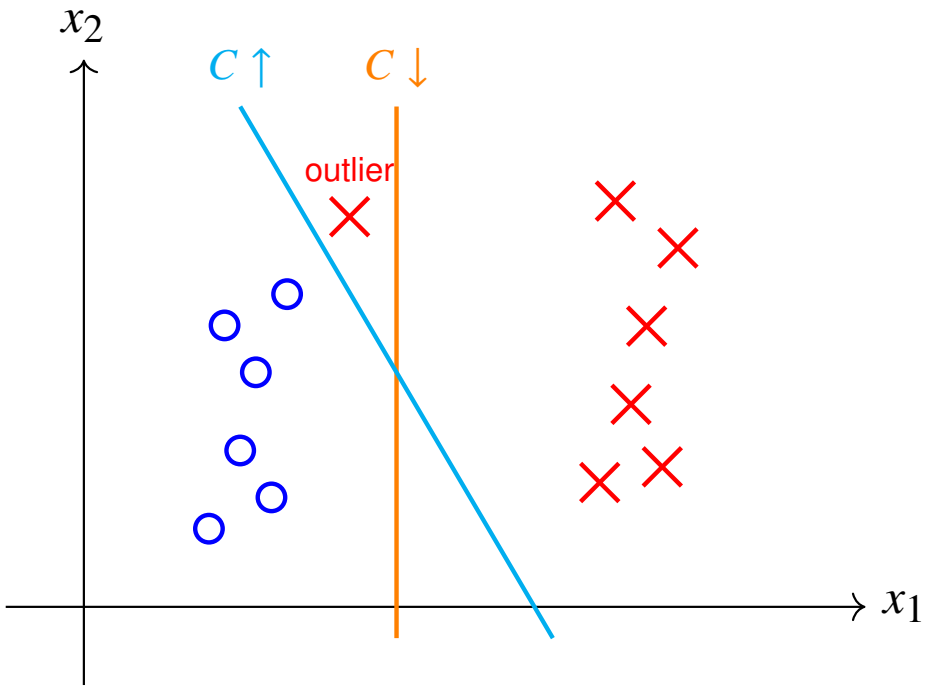




Regularización en SVM

- La formalización del SVM (hasta ahora) asume que el dataset es **separable linealmente**
- Sin embargo, encontrar el margen óptimo puede ser sensible a *outlier*

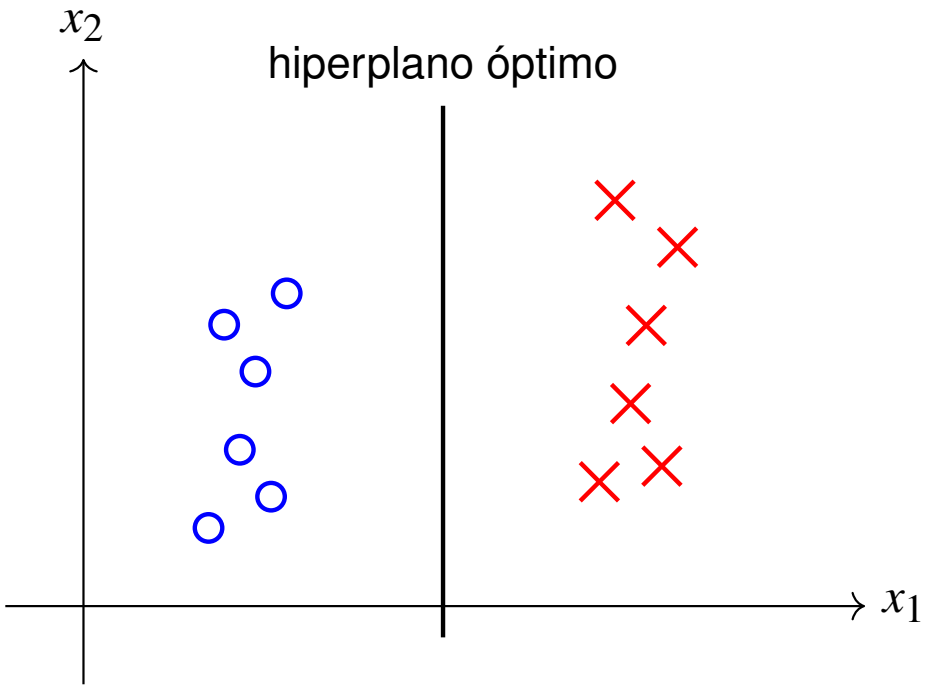


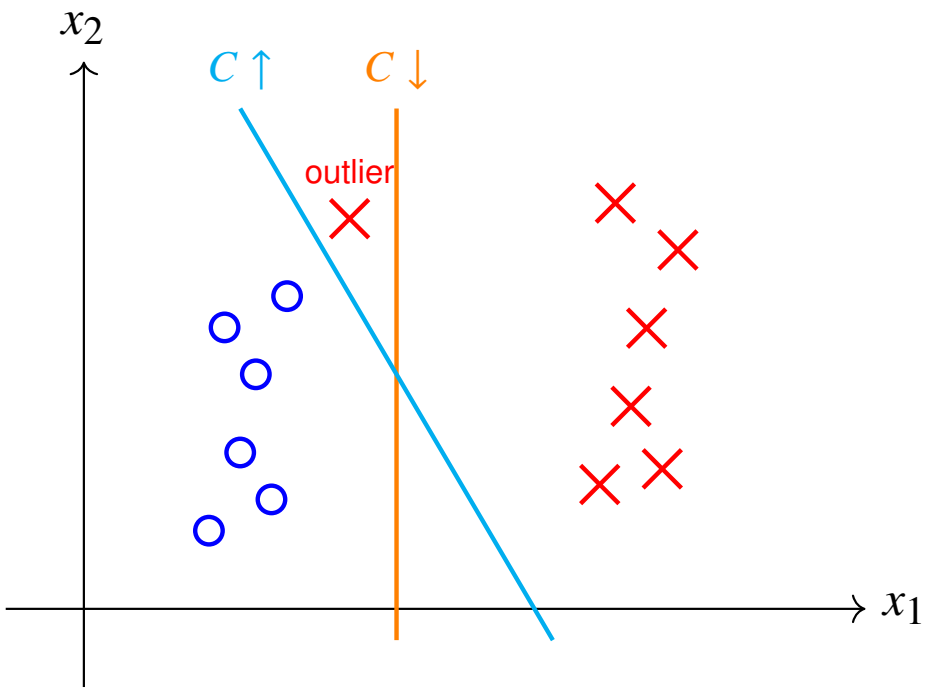


- $C \downarrow$ Se da más relevancia a la maximización de los márgenes (***hard-margin***)

Regularización en SVM

- La formalización del SVM (hasta ahora) asume que el dataset es **separable linealmente**
- Sin embargo, encontrar el margen óptimo puede ser sensible a *outlier*





- $C \downarrow$ Se da más relevancia a la maximización de los márgenes (***hard-margin***)
- $C \uparrow$ Se da más relevancia a la clasificación $>$ ajusta los datos (***soft-margin***)

Regularización en SVM (cont'd)

Queremos encontrar el hiperplano que maximiza el margen y minimiza los errores de clasificación:

$$\min_{w,b,\xi} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$$

sujeto a:

$$y_i\left(w^\top x^{(i)} + b\right) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \ldots, n$$

- $\|w\|^2$ controla la **anchura del margen**
- ξ_i son las **variables de holgura** que permiten errores.
- $C > 0$ es el **parámetro de regularización**:
  - $C \uparrow \rightarrow$ **pocos errores**, margen más pequeño. Penaliza mucho errores (riesgo de *overfitting*)
  - $C \downarrow \rightarrow$ margen más amplio, **tolera errores** (mejor generaliza)

Regularización en SVM (cont'd)

Queremos encontrar el hiperplano que maximiza el margen y minimiza los errores de clasificación:

$$\min_{w,b,\xi} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$$

sujeto a:

$$y_i\left(w^\top x^{(i)} + b\right) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \ldots, n$$

- $\|w\|^2$ controla la **anchura del margen**
- ξ_i son las **variables de holgura** que permiten errores.
- $C > 0$ es el **parámetro de regularización**:
  - $C \uparrow \rightarrow$ **pocos errores**, margen más pequeño. Penaliza mucho errores (riesgo de *overfitting*)
  - $C \downarrow \rightarrow$ margen más amplio, **tolera errores** (mejor generaliza)

- Intuitivamente, los márgenes (funcionales) pueden ser < 1

Regularización en SVM (cont'd)

Queremos encontrar el hiperplano que maximiza el margen y minimiza los errores de clasificación:

$$\min_{w,b,\xi} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$$

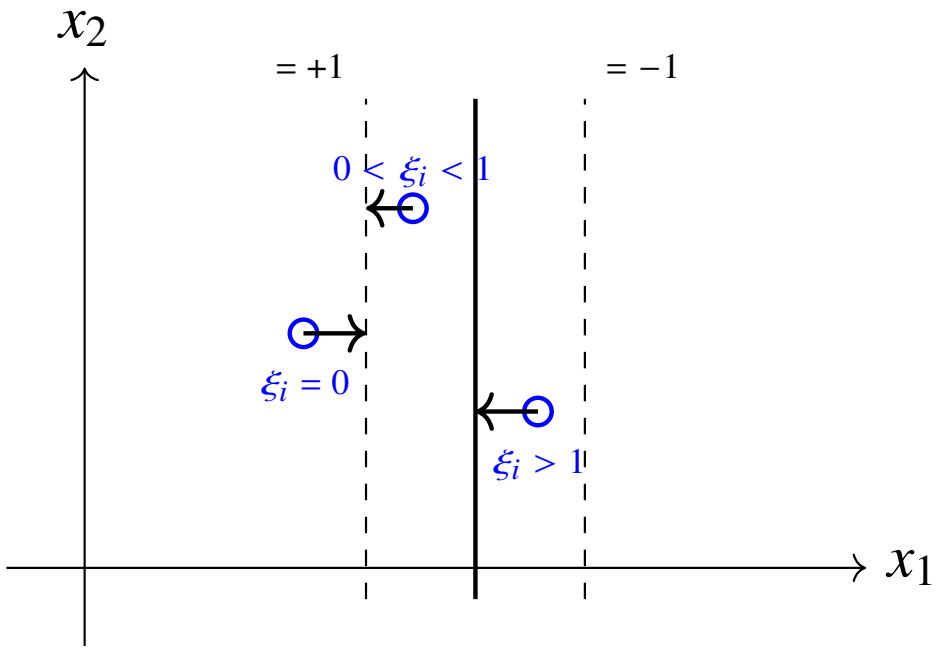
sujeto a:

$$y_i\left(w^\top x^{(i)} + b\right) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \ldots, n$$

- $\|w\|^2$ controla la **anchura del margen**
- ξ_i son las **variables de holgura** que permiten errores.
- $C > 0$ es el **parámetro de regularización**:
  - $C \uparrow \rightarrow$ **pocos errores**, margen más pequeño. Penaliza mucho errores (riesgo de *overfitting*)
  - $C \downarrow \rightarrow$ margen más amplio, **tolera errores** (mejor generaliza)

- Intuitivamente, los márgenes (funcionales) pueden ser < 1
- C controla el *peso* relativo entre la meta de minimizar $\|w\|^2$ (haciendo márgenes grandes), y asegurar que muchos ejemplos tengan un margen funcional de (al menos) 1

Regularización en SVM (cont'd)



- $\xi_i = 0$: punto correctamente clasificado y fuera del margen.
- $0 < \xi_i < 1$: dentro del margen, pero sigue del lado correcto.
- $\xi_i > 1$: mal clasificado (cruza el hiperplano).

***Soft-margin* SVM**

Formulación dual

- El problema de clasificador de márgenes óptimo es conocido como el problema **primal** del SVM

[1] Dual of support vector machine

Formulación dual

- El problema de clasificador de márgenes óptimo es conocido como el problema **primal** del SVM
  - involucra el peso de los atributos

[1] Dual of support vector machine

Formulación dual

- El problema de clasificador de márgenes óptimo es conocido como el problema **primal** del SVM
  - involucra el peso de los atributos
- Puede ser también reescrito como un problema de maximización, denominado problema **dual**[1] del SVM

[1] Dual of support vector machine

Formulación dual

- El problema de clasificador de márgenes óptimo es conocido como el problema **primal** del SVM
  - involucra el peso de los atributos
- Puede ser también reescrito como un problema de maximización, denominado problema **dual**[1] del SVM
  - el número de variables a optimizar es ahora independiente de la dimensionalidad
  - w puede representarse como combinación lineal los $x^{(i)}$

$$w = \sum_{i=1}^{n} \alpha_i y^{(i)} x^{(i)}$$

[1] Dual of support vector machine

Formulación dual

- El problema de clasificador de márgenes óptimo es conocido como el problema **primal** del SVM
  - involucra el peso de los atributos
- Puede ser también reescrito como un problema de maximización, denominado problema **dual**[1] del SVM
  - el número de variables a optimizar es ahora independiente de la dimensionalidad
  - w puede representarse como combinación lineal los $x^{(i)}$

$$w = \sum_{i=1}^{n} \alpha_i y^{(i)} x^{(i)}$$

- La nueva función **dual** depende solamente de los vectores de soporte y será maximizada para estos ejemplos de entrenamiento (no necesitamos aproximar pesos de w)
- Es la base para el uso de ***kernels*** en datasets no lineales, ya que calcula el grado de similitud entre vectores (producto punto, geométrico)

[1] Dual of support vector machine

Función Objetivo (dual)

$$\max_{\boldsymbol{\alpha}} \ \mathcal{L}_D(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} \mathbf{x}^{(i)} \mathbf{x}^{(j)}$$

Sujeto a:

$$0 \leq \alpha_i \leq C, \quad \forall i$$

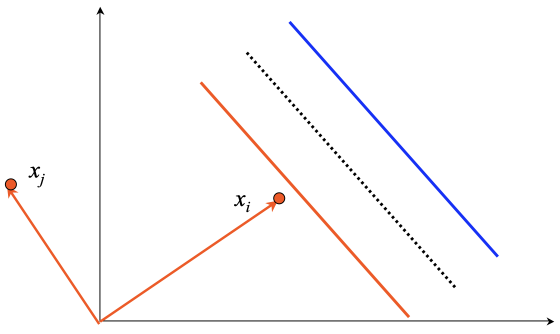
$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

- α_i son los multiplicadores de Lagrange.
- C controla el trade-off entre margen ancho y errores de clasificación.

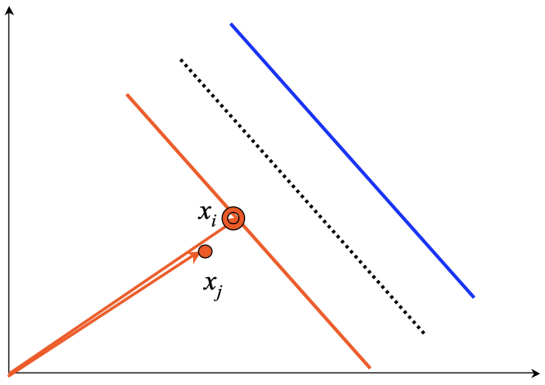
Formulación dual (casos)

- Dados los vectores $x^{(i)}$, $x^{(j)}$:

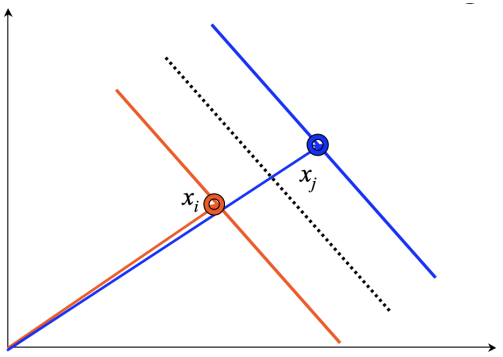
Ortogonales



Redundantes



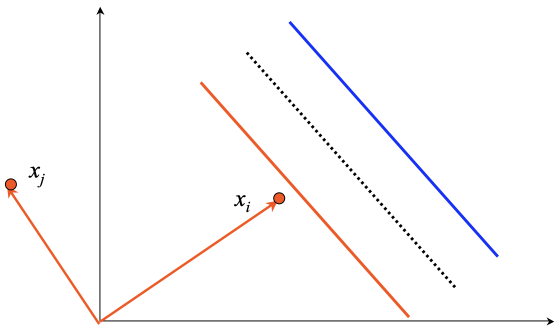
Maximizan el margen



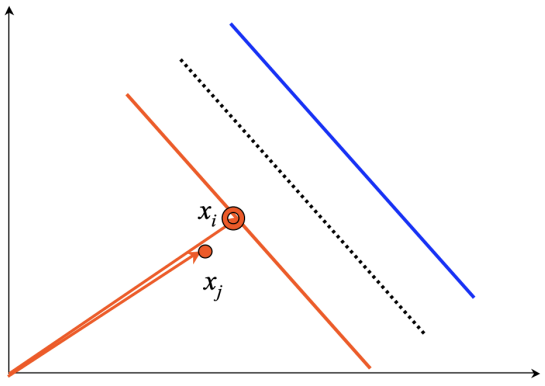
Formulación dual (casos)

- Dados los vectores $x^{(i)}$, $x^{(j)}$:

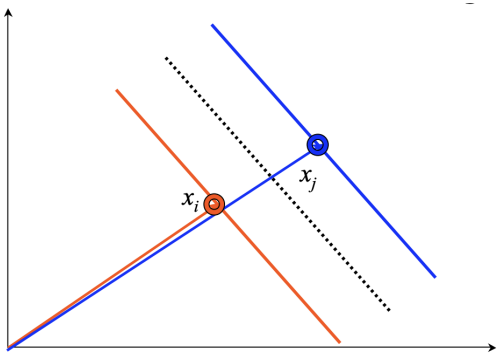
Ortogonales



Redundantes



Maximizan el margen



- Si vectores **redundantes** (similares, misma clase), se descarta $x^{(i)}$ o $x^{(j)}$
- Si vectores similares y clase opuesta, se ajustan para mantener el margen

(*) Créditos de R. Berwick

Dual (Final)

- Los α_i indican **cuánto contribuye** cada punto de entrenamiento para encontrar el hiperplano

Dual (Final)

- Los α_i indican **cuánto contribuye** cada punto de entrenamiento para encontrar el hiperplano
- $\alpha_i > 0$ indica que $x^{(i)}$ es un **vector soporte**

Dual (Final)

- Los α_i indican **cuánto contribuye** cada punto de entrenamiento para encontrar el hiperplano
- $\alpha_i > 0$ indica que $x^{(i)}$ es un **vector soporte**
- $\alpha_i = 0$ indica que el punto no afecta al hiperplano

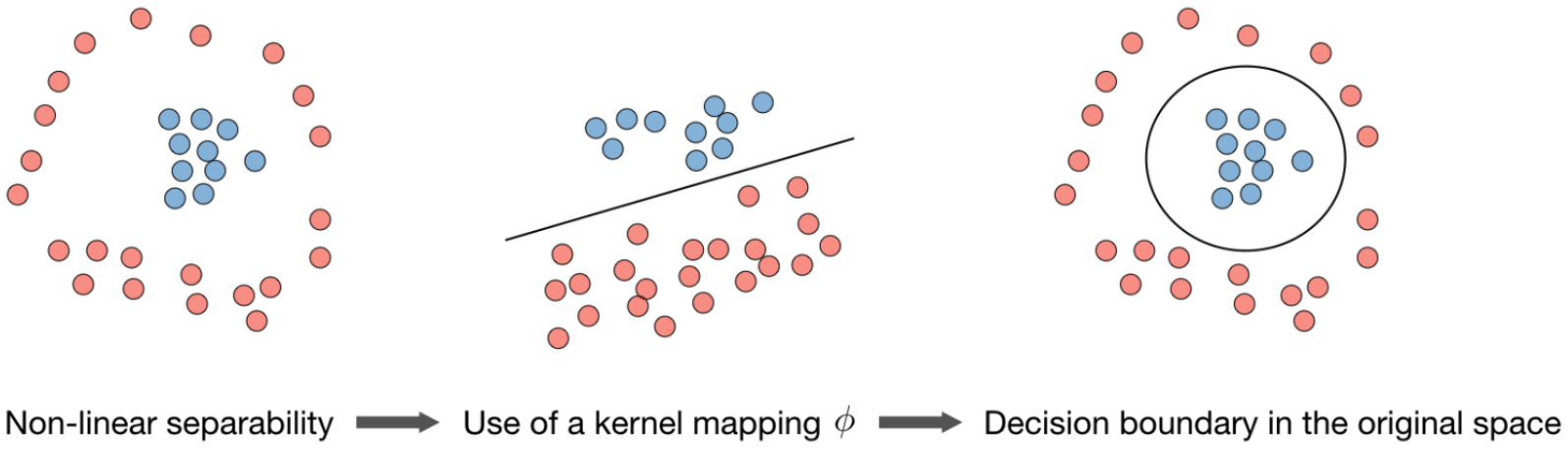
Dual (Final)

- Los α_i indican **cuánto contribuye** cada punto de entrenamiento para encontrar el hiperplano
- $\alpha_i > 0$ indica que $x^{(i)}$ es un **vector soporte**
- $\alpha_i = 0$ indica que el punto no afecta al hiperplano
- **el nuevo algoritmo depende solo de un subconjunto de puntos permitiendo trabajar con espacios de alta dimensión!**

Kernels

Intuición

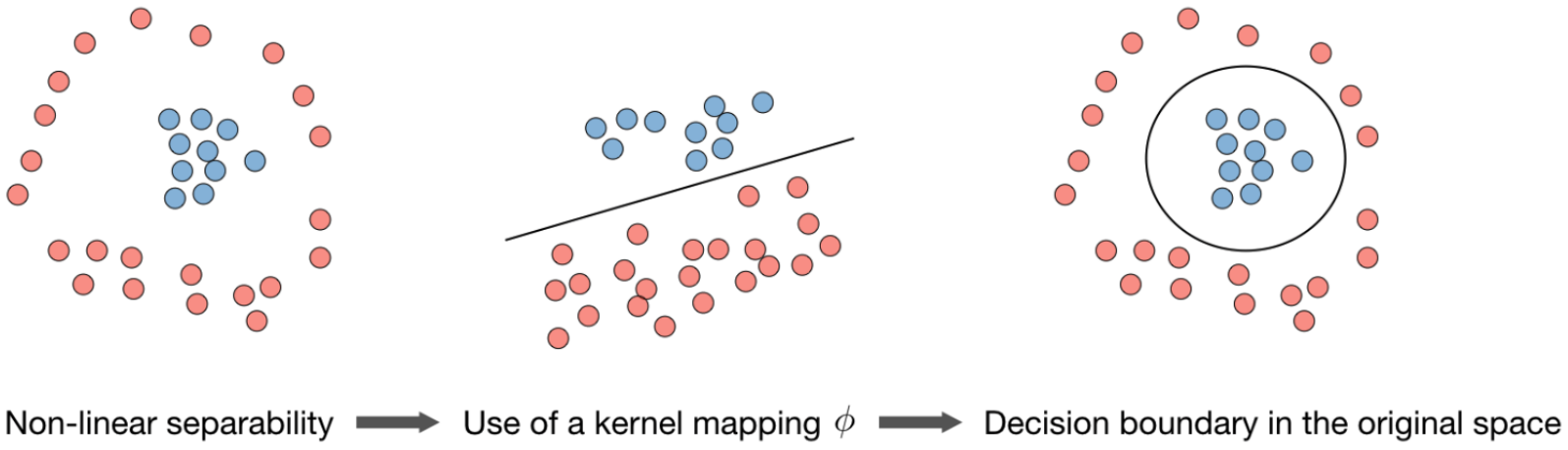
Clasificación no lineal



- **Idea**: mapear los datos desde un espacio original a otro de mayor dimensionalidad

Intuición

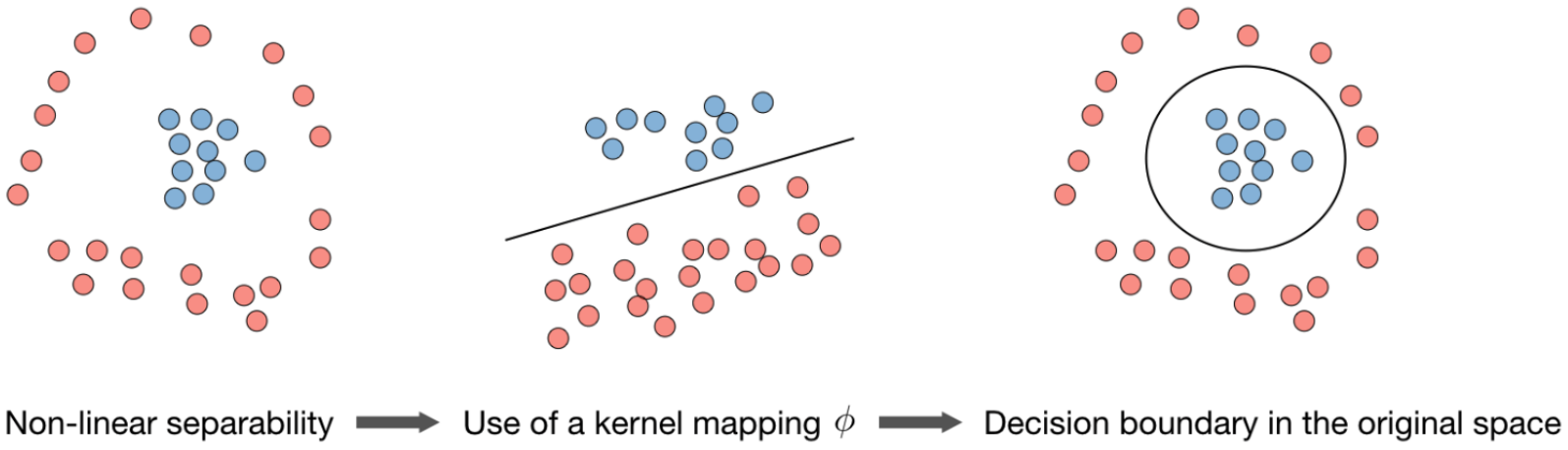
Clasificación no lineal



- **Idea**: mapear los datos desde un espacio original a otro de mayor dimensionalidad
- En el espacio ampliado esperamos que las clases sean separables linealmente

Intuición

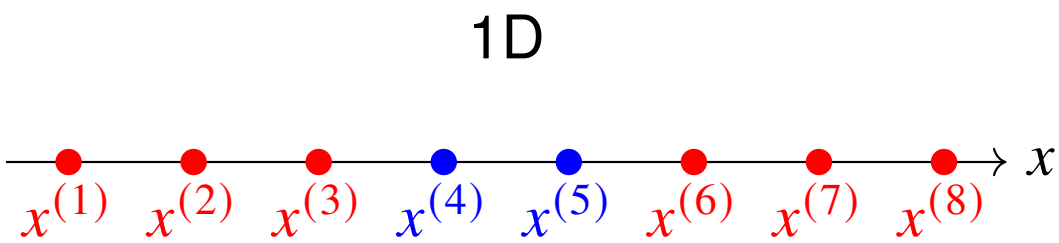
Clasificación no lineal



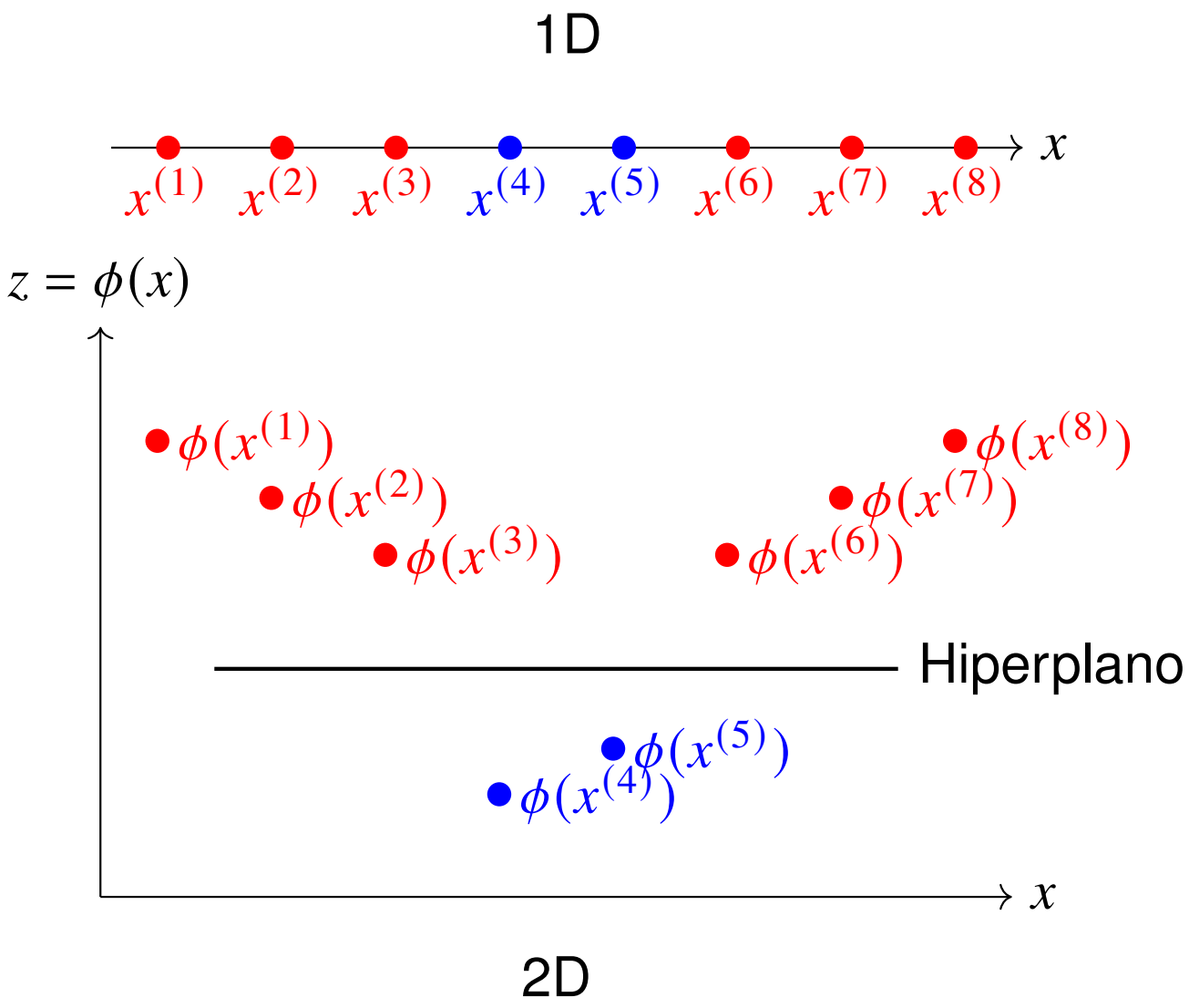
- **Idea**: mapear los datos desde un espacio original a otro de mayor dimensionalidad
- En el espacio ampliado esperamos que las clases sean separables linealmente
- Si las transformaciones son no lineales, se traduce en una frontera de decisión no lineal en el espacio original

(*) Créditos de la imagen: cheatsheet-supervised-learning

Intuición (cont'd)



Intuición (cont'd)



Truco del Kernel (*Kernel Trick*)

- Un enfoque posible es mapear cada $x^{(i)}$ a otro espacio, usando algun transformación $\phi(x^{(i)})$ tal que:

$$k(x^{(i)}, x^{(j)}) = \phi(x^{(i)})^T \phi(x^{(j)})$$

Truco del Kernel (*Kernel Trick*)

- Un enfoque posible es mapear cada $x^{(i)}$ a otro espacio, usando algun transformación $\phi(x^{(i)})$ tal que:

$$k(x^{(i)}, x^{(j)}) = \phi(x^{(i)})^T \phi(x^{(j)})$$

- Suponiendo

$$\phi(x^{(i)}) : \mathbb{R}^2 \rightarrow \mathbb{R}^n \; n >> 2$$

el computo sería muy costoso!

Truco del Kernel (*Kernel Trick*)

- Un enfoque posible es mapear cada $x^{(i)}$ a otro espacio, usando algun transformación $\phi(x^{(i)})$ tal que:

$$k(x^{(i)}, x^{(j)}) = \phi(x^{(i)})^T \phi(x^{(j)})$$

- Suponiendo

$$\phi(x^{(i)}) : \mathbb{R}^2 \rightarrow \mathbb{R}^n \; n >> 2$$

el computo sería muy costoso!

- Una función de *kernel* es una funcion equivalente a un producto interno en algun espacio de caracteristicas

$$k(x^{(i)}, x^{(j)}) = \left(\phi(x^{(i)}), \phi(x^{(j)})\right)$$

Truco del Kernel (*Kernel Trick*)

- Un enfoque posible es mapear cada $x^{(i)}$ a otro espacio, usando algun transformación $\phi(x^{(i)})$ tal que:

$$k(x^{(i)}, x^{(j)}) = \phi(x^{(i)})^T \phi(x^{(j)})$$

- Suponiendo

$$\phi(x^{(i)}) : \mathbb{R}^2 \rightarrow \mathbb{R}^n \; n >> 2$$

el computo sería muy costoso!

- Una función de *kernel* es una funcion equivalente a un producto interno en algun espacio de caracteristicas

$$k(x^{(i)}, x^{(j)}) = \left(\phi(x^{(i)}), \phi(x^{(j)})\right)$$

- Dicha función mapea implicitamente datos a un espacio de mayor dimension (sin necesidad de calcular cada $\phi(x^{(i)})$ explicitamente)

Truco del Kernel (*Kernel Trick*)

- Un enfoque posible es mapear cada $x^{(i)}$ a otro espacio, usando algun transformación $\phi(x^{(i)})$ tal que:

$$k(x^{(i)}, x^{(j)}) = \phi(x^{(i)})^T \phi(x^{(j)})$$

- Suponiendo

$$\phi(x^{(i)}) : \mathbb{R}^2 \rightarrow \mathbb{R}^n \; n >> 2$$

el computo sería muy costoso!

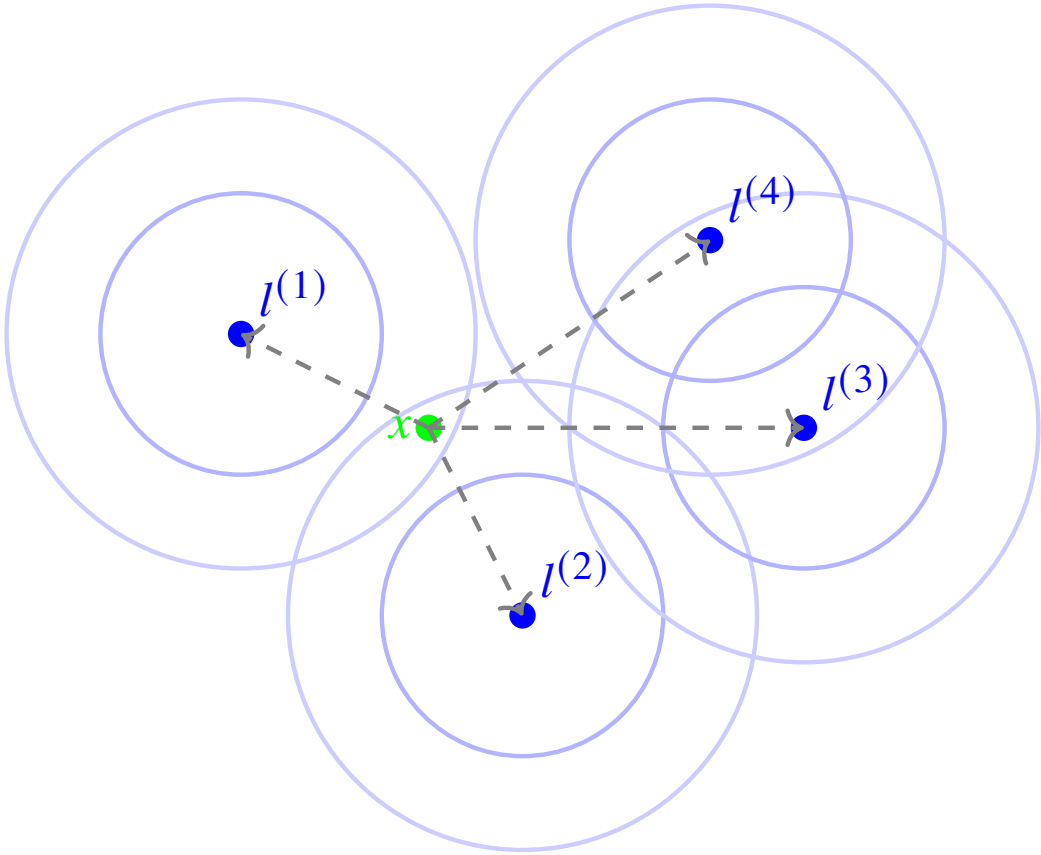
- Una función de *kernel* es una funcion equivalente a un producto interno en algun espacio de caracteristicas

$$k(x^{(i)}, x^{(j)}) = \left(\phi(x^{(i)}), \phi(x^{(j)})\right)$$

- Dicha función mapea implicitamente datos a un espacio de mayor dimension (sin necesidad de calcular cada $\phi(x^{(i)})$ explicitamente)

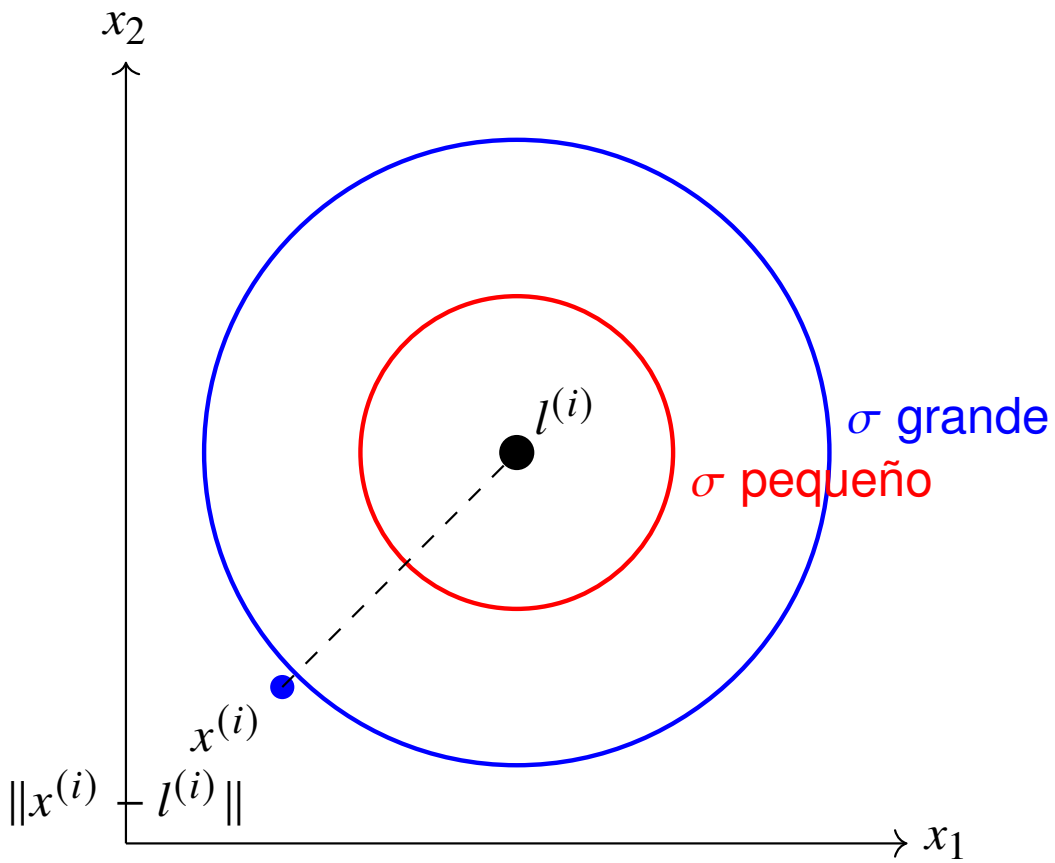
El **truco del kernel** es usar el SVM en espacios no lineales sin calcular explícitamente la transformación a un espacio de alta dimensión, reemplazando los productos punto por un *kernel* que devuelve el mismo resultado.

Kernel Gaussiano (Intuición)



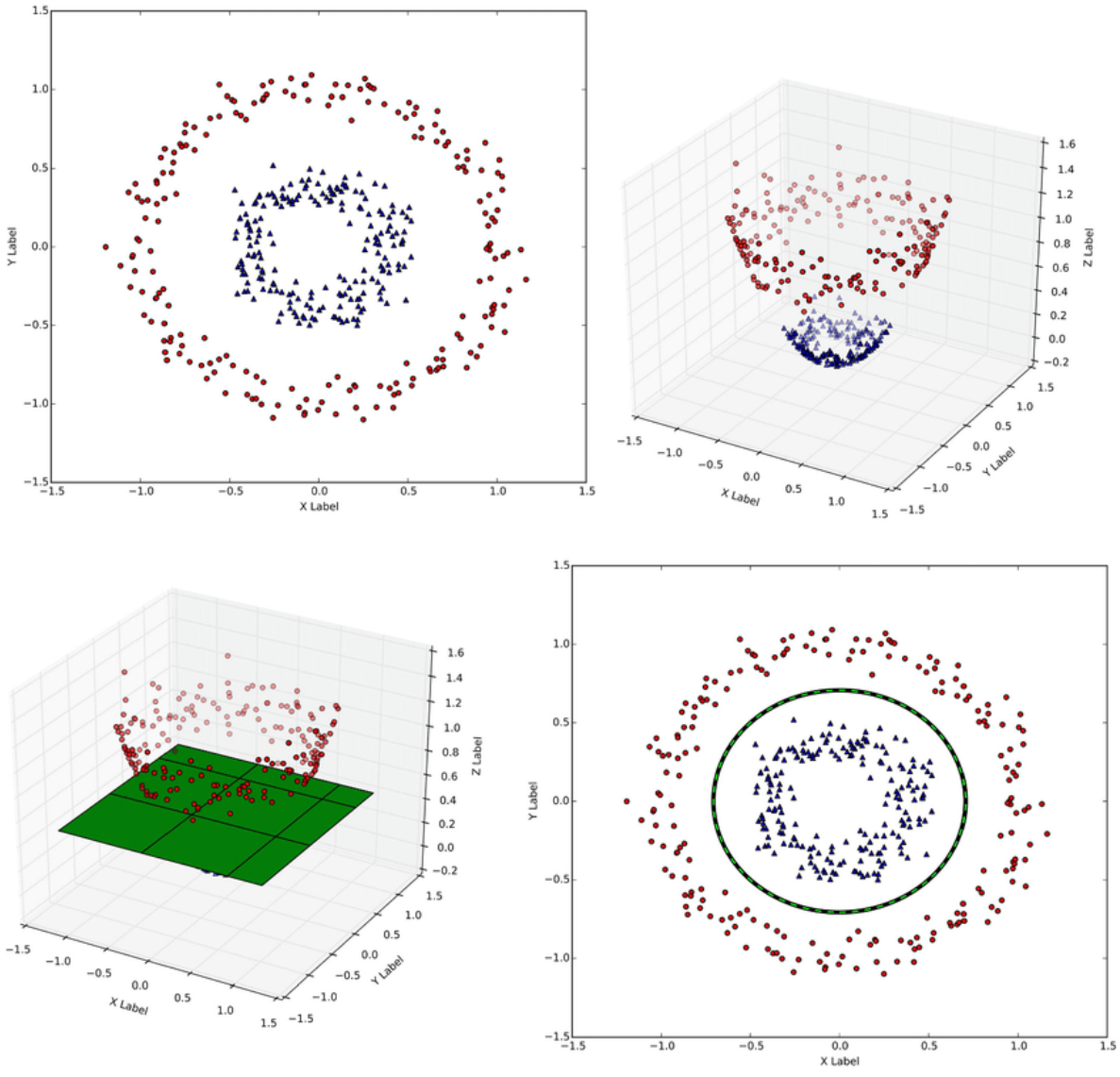
- cada $l^{(i)}$ es un *landmark* (o punto de referencia)
- para cada x, $f_j^{(i)} = similarity(x^{(i)}, l^{(i)}) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{l}_j\|^2}{2\sigma^2}\right)$
- si $x^{(i)} \approx l^{(i)} \Rightarrow f^{(i)} = \exp\left(-\frac{0^2}{2\sigma^2}\right) \approx 1$ > vectores similares en espacio transformado

Kernel Gaussiano (RBF): efecto de σ



- σ **grande:** la similitud decrece lentamente $\rightarrow$ puntos lejanos todavía influyen $\rightarrow$ frontera más suave.
- σ **pequeño:** la similitud decrece rápido $\rightarrow$ solo vecinos muy cercanos influyen $\rightarrow$ frontera más ajustada y compleja $\rightarrow$ riesgo de overfitting.

Kernel Polinómico



(*) Créditos de la imagen: Parameter investigation of support vector machine classifier with kernel functions

Kernels más comunes en SVM

Kernel	Fórmula
Lineal	$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j$
Polinómico	$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^\top \mathbf{x}_j + c)^d$
RBF / Gaussiano	$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$
Sigmoidal	$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\kappa \mathbf{x}_i^\top \mathbf{x}_j + c)$

- **Lineal:** Producto punto clásico; no transforma el espacio.
- **Polinómico:** Permite separar datos polinómicamente separables; d = grado del polinomio.
- **RBF / Gaussiano:** Mapea datos a un espacio de dimensión infinita; bueno para datos no lineales.
- **Sigmoidal:** Inspirado en redes neuronales; depende de los parámetros κ y c.

Usando a SVM

¿Cómo usar un SVM en la práctica?

- Usar una librería tal como `sklearn.svm` e importar `SVC`, `NuSVC` o `LinearSVC`
- Parámetros
  - `C` (regularización)
  - `kernel` (función de similaridad)
- Si no se define un kernel, el problema es el de predecir $y = 1$ si $\theta^T x \geq 0$ y usar `LinearSVC`
- Si `kernel` es Gaussiano (o RBF) $\Rightarrow$

$$f_i = \exp\left(- \frac{\|\mathbf{x} - l^{(i)}\|^2}{2\sigma^2} \right)$$

  dónde $l^{(i)} = x^{(i)}$, y el parámetro a especificar es σ^2 (o `gamma` - γ - el cual es $\frac{1}{\sigma^2}$)
  - `NuSVC` vs. `SVC`. Ambos son similares, `NuSVC` permite contralar el número de vectores de soporte
- Si multiclase $y \in 1, 2, 3, ..., K$, usar enfoques *one-vs-rest* (`ovr`) o *one-vs-one* (`ovo`)

(*) scikit-learn SVM

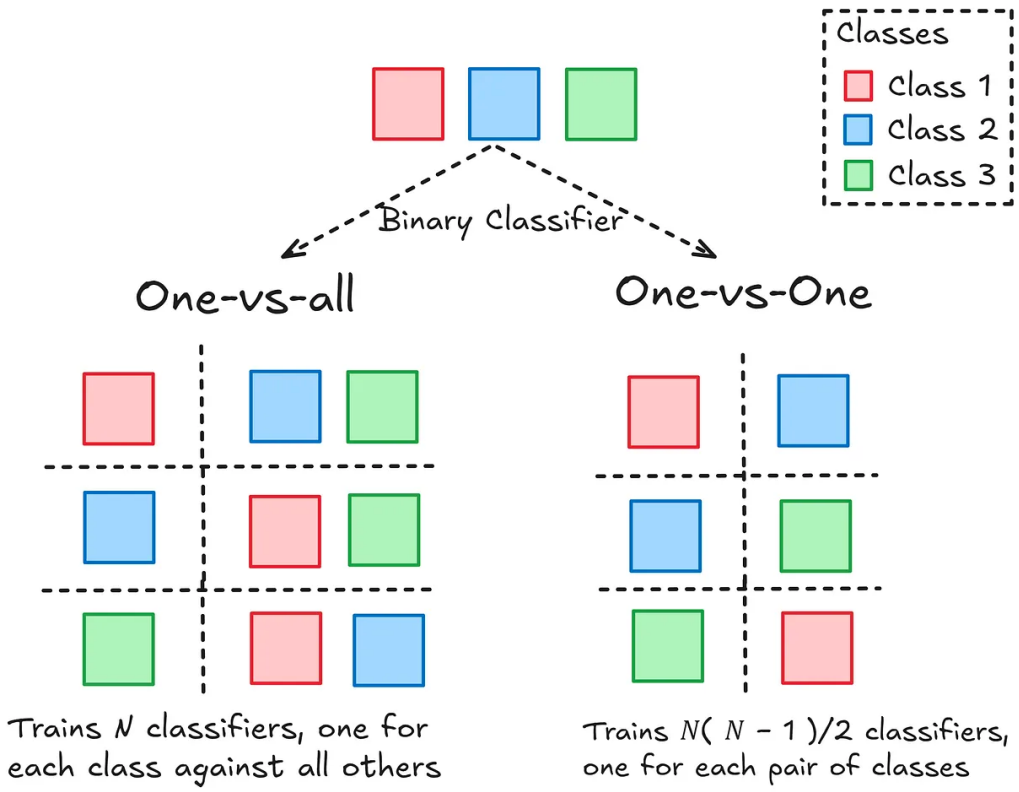
ovr **vs** ovo

ovr

* K, una clase vs todas
* Clase con mayor puntuación
+ Escalable, rápido
- Datos desbalanceados, – precisión

ovo

* $\frac{K(K-1)}{2}$, cada par de clases
* Voto mayoritario
+ Mejor precisión, balanceado
- Muchos modelos si K es grande



(*) image de ovr vs ovo

Próximas clases

- Redes Neuronales
- Aprendizaje no supervisado

¡Gracias!

Bibliografía y material de referencia


Harrington, Peter. Machine learning in action. *Simon and Schuster*, 2012.


Alpaydin, Ethem. Introduction to machine learning. 3era Edición *MIT Press*, 2020.


Brett Lantz. Machine Learning with R. *Packt Publishing*, 1997.


Tom M. Mitchell. Machine Learning. *WCB McGraw-Hill*, 1997.


Witten I., Frank E., Hall, M., Pal C.. Data Mining: Practical Machine Learning Tools and Techniques. 4th Edition*WMorgan Kaufmann. Elsevier*, 2017.


Michael A. Nielsen. Neural Networks and Deep Learning. 4th Edition *Determination Press*, 2015.
`http://neuralnetworksanddeeplearning.com`

Bibliografía y material de referencia


Afshine Amidi, Shervine Amidi. CS 229 — Machine Learning. https://stanford.edu/~shervine/teaching/cs-229/


Andrew Ng. Stanford CS229 - Machine Learning Course. https://www.youtube.com/playlist?list=PLoROMvodv4rMiGQp3WXShtMGgzqpfVfbU


Andrew Ng. Deep Learning AI. https://www.deeplearning.ai/resources/


Kilian Weinberger. Machine Learning for Intelligent Systems. https://www.cs.cornell.edu/courses/cs4780/2018fa/syllabus/


Corinna Cortes and Vladimir Vapnik. Support-vector networks. Machine Learning. Springer. 1995