Maestría en Explotación de Datos y Descubrimiento del Conocimiento

Unidad II, Clase 4

Aprendizaje Automático

Clasificación

Laura de la Fuente, Hernán Bocaccio
Ayudantes: Gastón Bujía, Diego Onna y Sofía Morena del Pozo

Dirección de e-mail de la materia:
datawillconfess@gmail.com

Itinerario de la clase

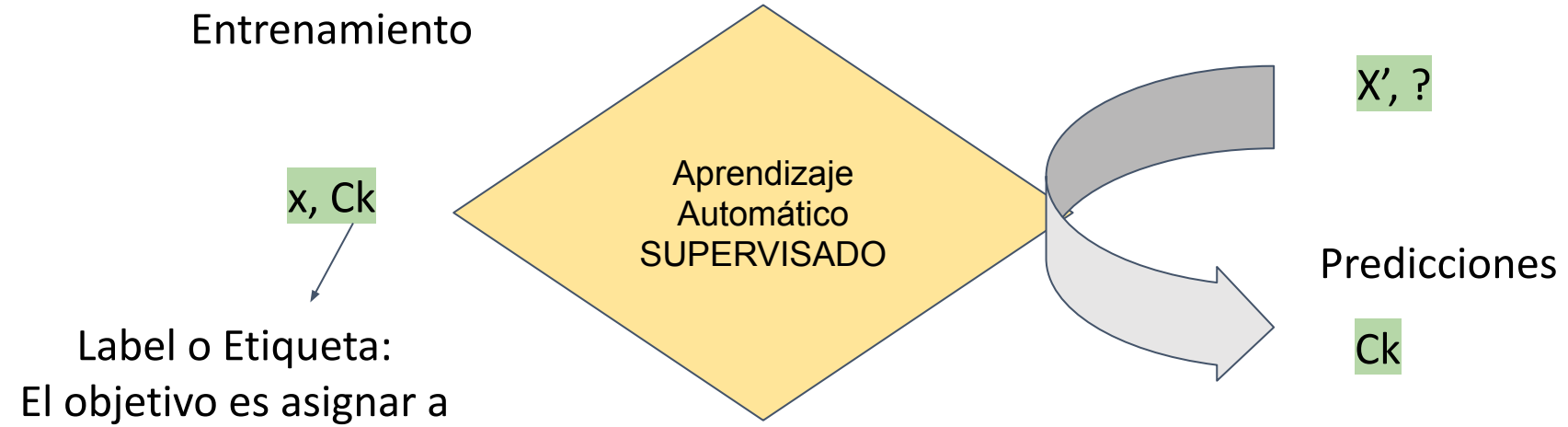
- Generalidades de clasificación
- Clasificación binaria y multiclase
- Regresión logística
- Sigmoidea
- Cross entropy
- SVM
- LDA
- KNN
- NB
- Desempeño multiclase

Clasificación

Aprendizaje supervisado:

Los **datos están anotados** con la respuesta correcta que quiero predecir.

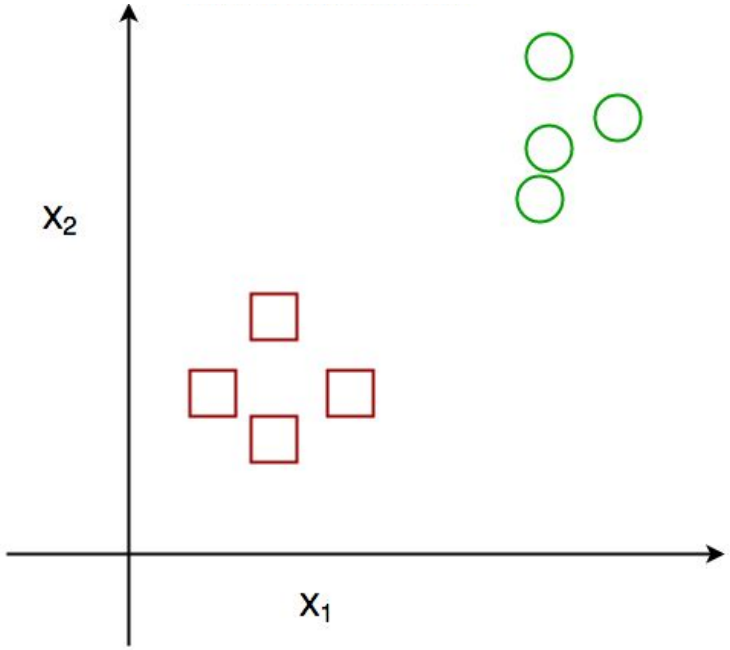
- Clasificación: Predecir una clase (variable categórica)
- Regresión: Predecir un valor numérico (variable numérica)



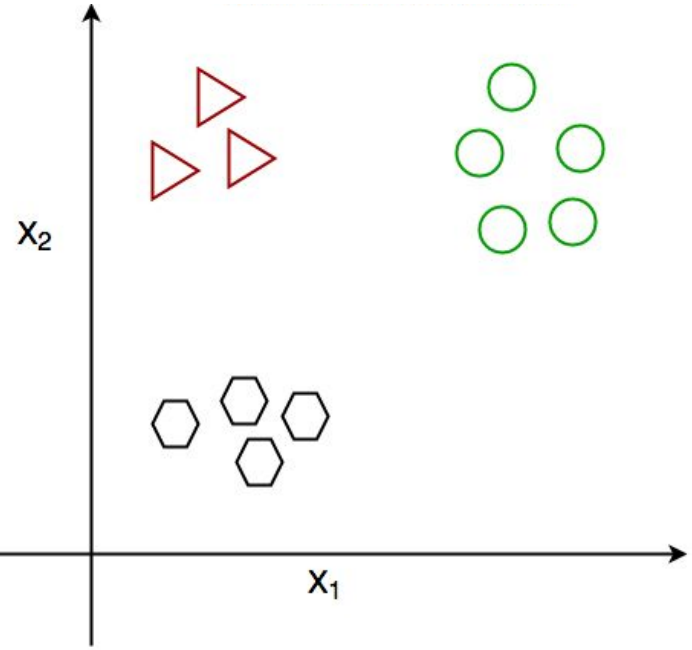
Clasificación

Cuando la **variable** anotada que quiero predecir es **categórica** (clase).

Clasificación binaria



Clasificación multi-clase



La **predicción** sobre un dato nuevo puede ser la **clase** a la que pertenece o la **probabilidad** de pertenecer a esa clase según el modelo y el algoritmo.

Fronteras (contornos) y regiones de decisión

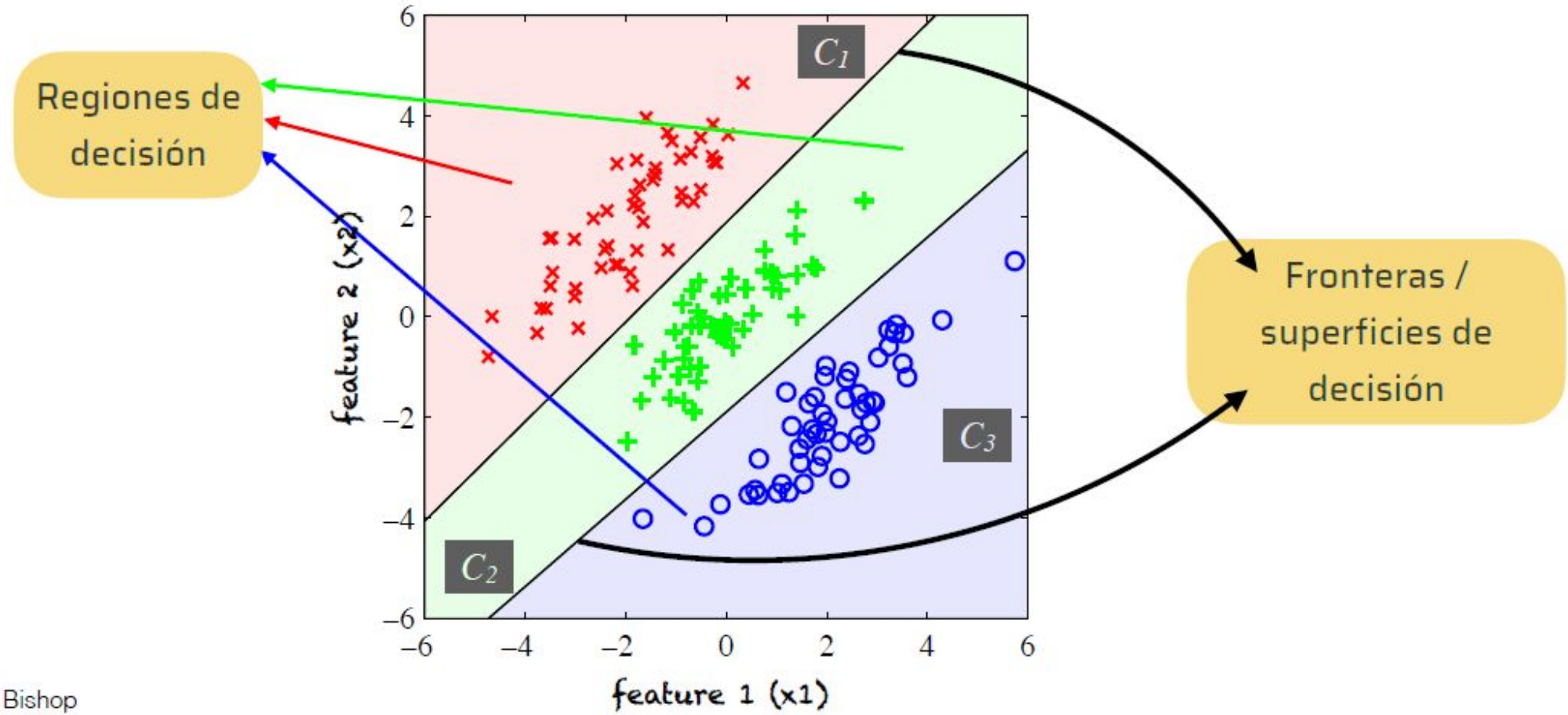


figure from Bishop

Regresión Logística

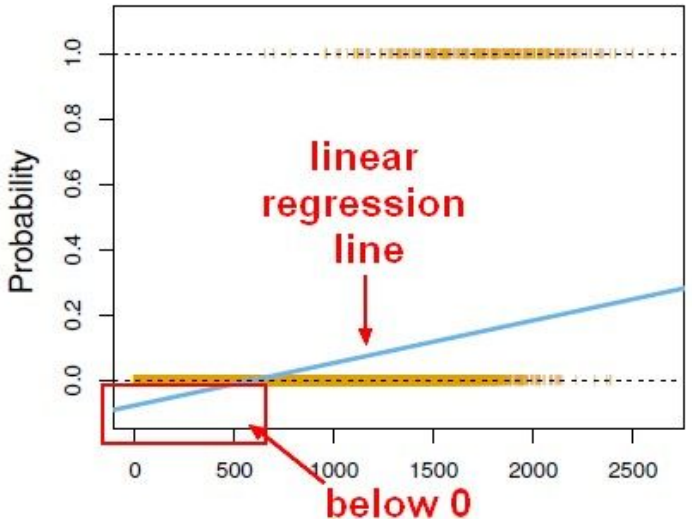
Los modelos lineales para clasificación usan una función de activación no lineal

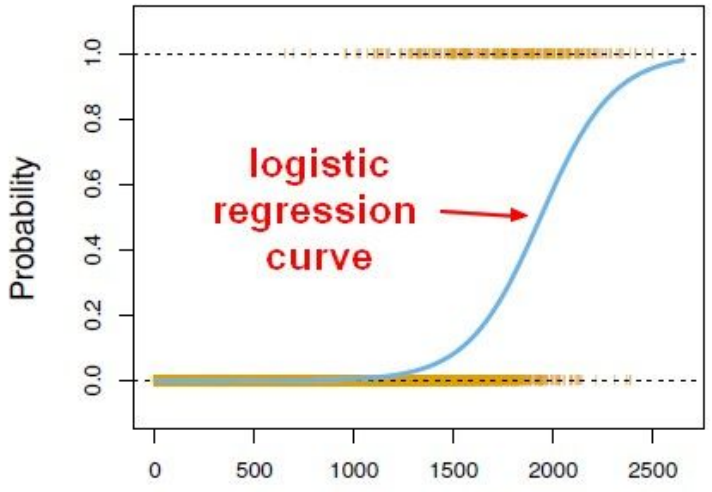
$$Y \approx f(X)$$

$$Y \approx f(X_1, X_2, ..., X_p)$$

$$\hat{y} = \hat{f}(x_1, x_2, ..., x_p)$$

Tradicionalmente se lo llama regresión por tratarse de un problema en el que busco estimar los parámetros que predicen una respuesta (variable independiente) en relación a los atributos (variables dependientes) de acuerdo a una forma funcional.





Función Logística o Sigmoide

$$f(t) = \frac{e^t}{1 + e^t}$$

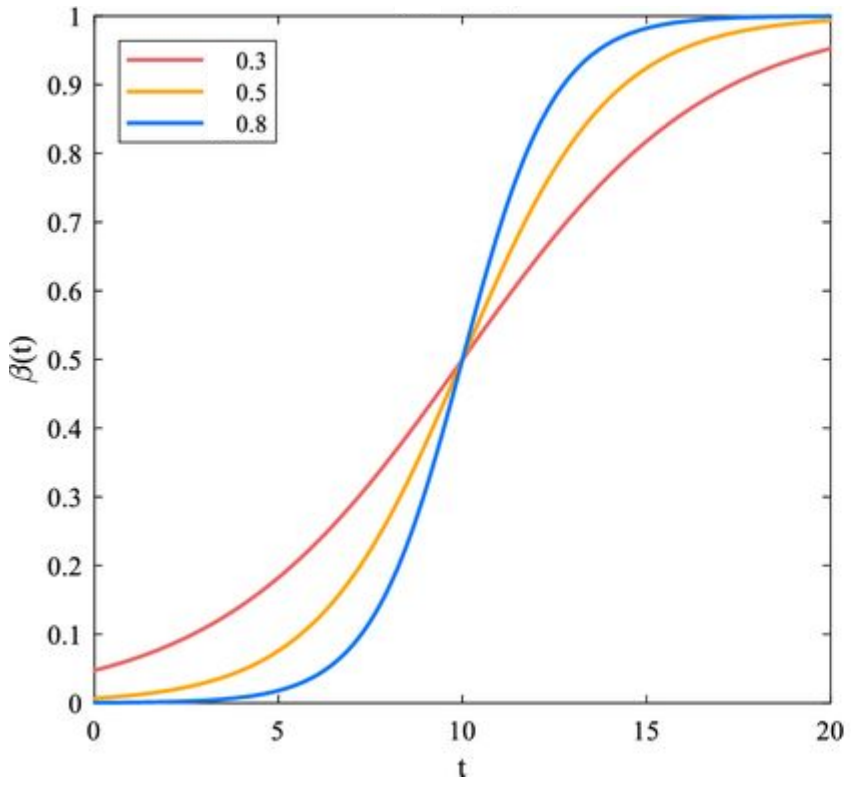
Estoy estimando la probabilidad de pertenecer a una clase

Regresión Logística

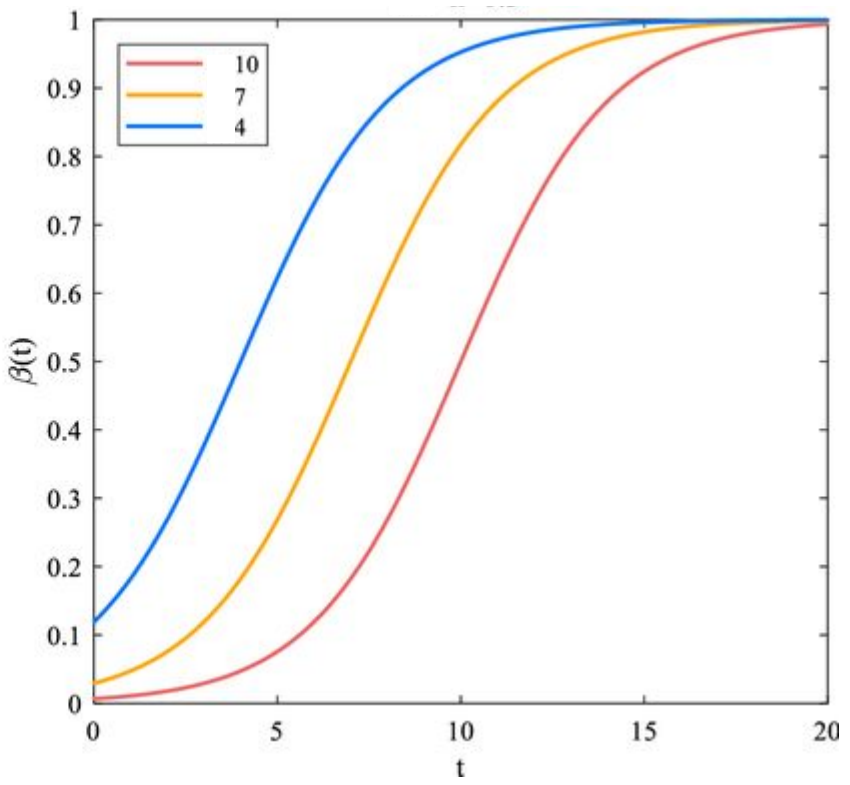
Función Logística o Sigmoide

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}$$

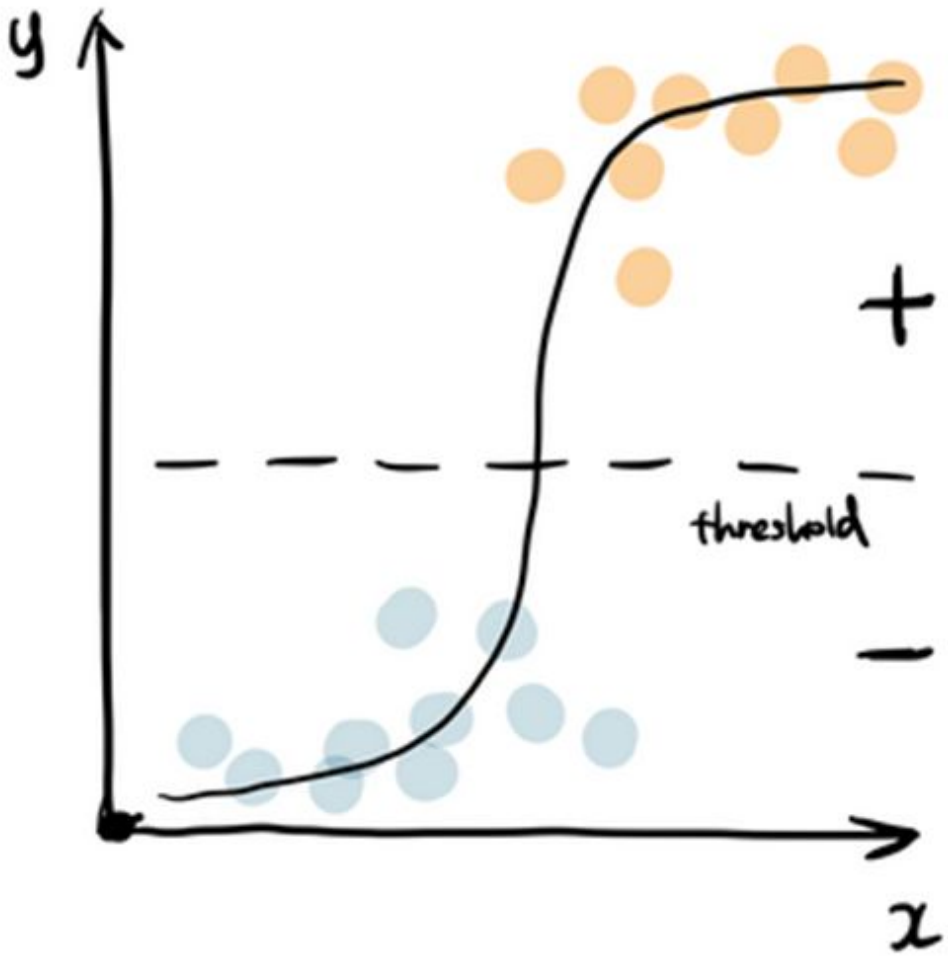
Beta 1



Beta 0



Regresión Logística



Class 1 when $y = 1$
Class 2 when $y = 0$

$$p(y = 1|x;\theta) + p(y = 0|x;\theta) = 1$$

$$p(y = 1) = \frac{1}{1 + e^{-\theta^T X}}$$

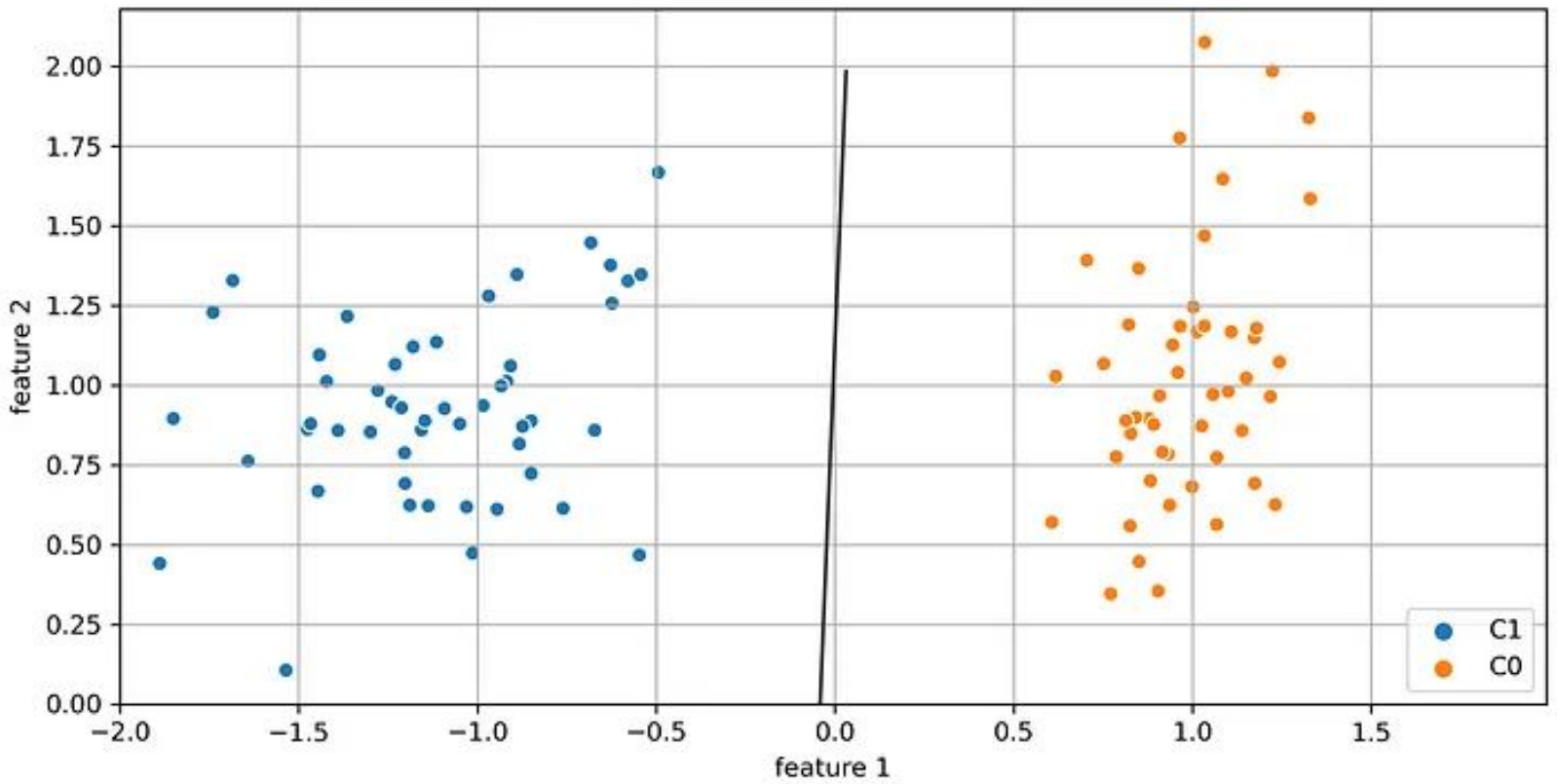
Problema de clasificación binaria

Obtengo probabilidad de pertenencia

Defino un umbral de probabilidad para delimitar el contorno de decisión

Regresión Logística

Contorno de decisión



Modelos lineales generalizados (clasificación)

Modelos lineales para regresión

$$y(x, w_0, w_1) = w_0 + w_1 x$$

Generalizando la idea:

$$y(\mathbf{x}, \mathbf{w}) = f\left(w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x}) \right)$$

donde **f es la función de activación**, y decimos que la superficie de decisión viene dada por funciones discriminantes.

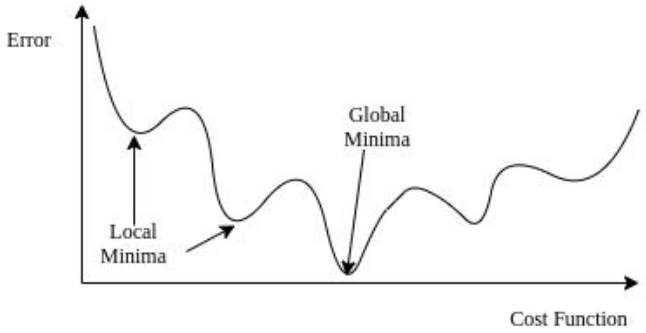
Regresión Logística

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Función costo:

Si uso una función costo como la de regresión, queda no convexa

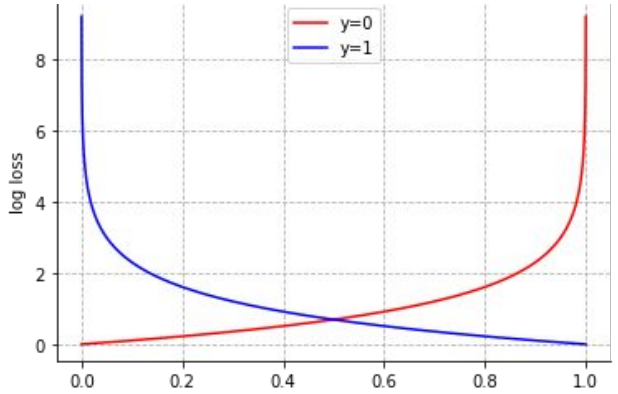
$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2.$$



Dificil encontrar mínimo global

Necesito redefinirla

$$Cost(h_\theta(x), y) = \begin{cases} -log(h_\theta(x)) & \text{if } y = 1 \\ -log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$



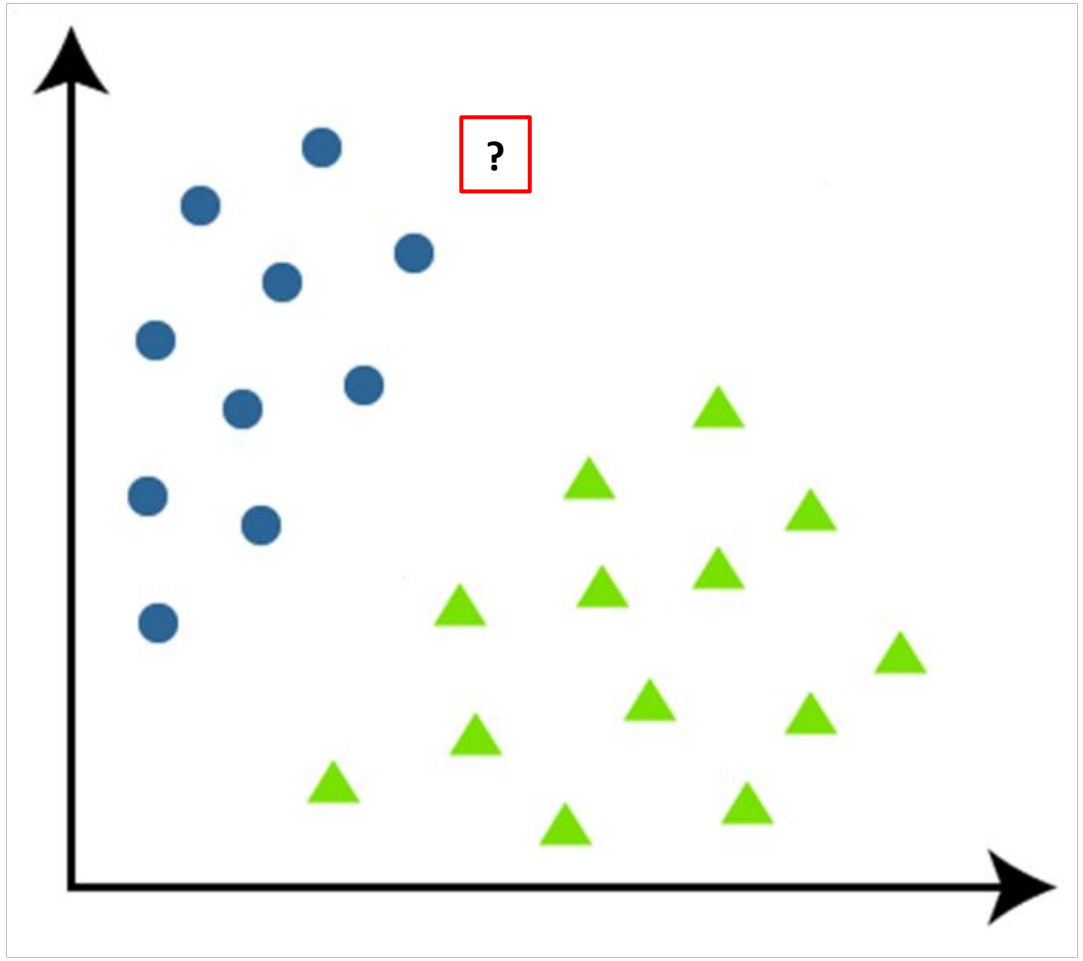
Penalizo más equivocarme en la predicción de clases

Likelihood $$\ell(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=0} (1 - p(x_{i'}))$$

Log-loss o Cross entropy

$$J(\theta) = -\frac{1}{m}\sum\left[y^{(i)}\log(h\theta(x(i))) + \left(1 - y^{(i)}\right)\log(1 - h\theta(x(i)))\right]$$

Support Vector Machines (SVM)



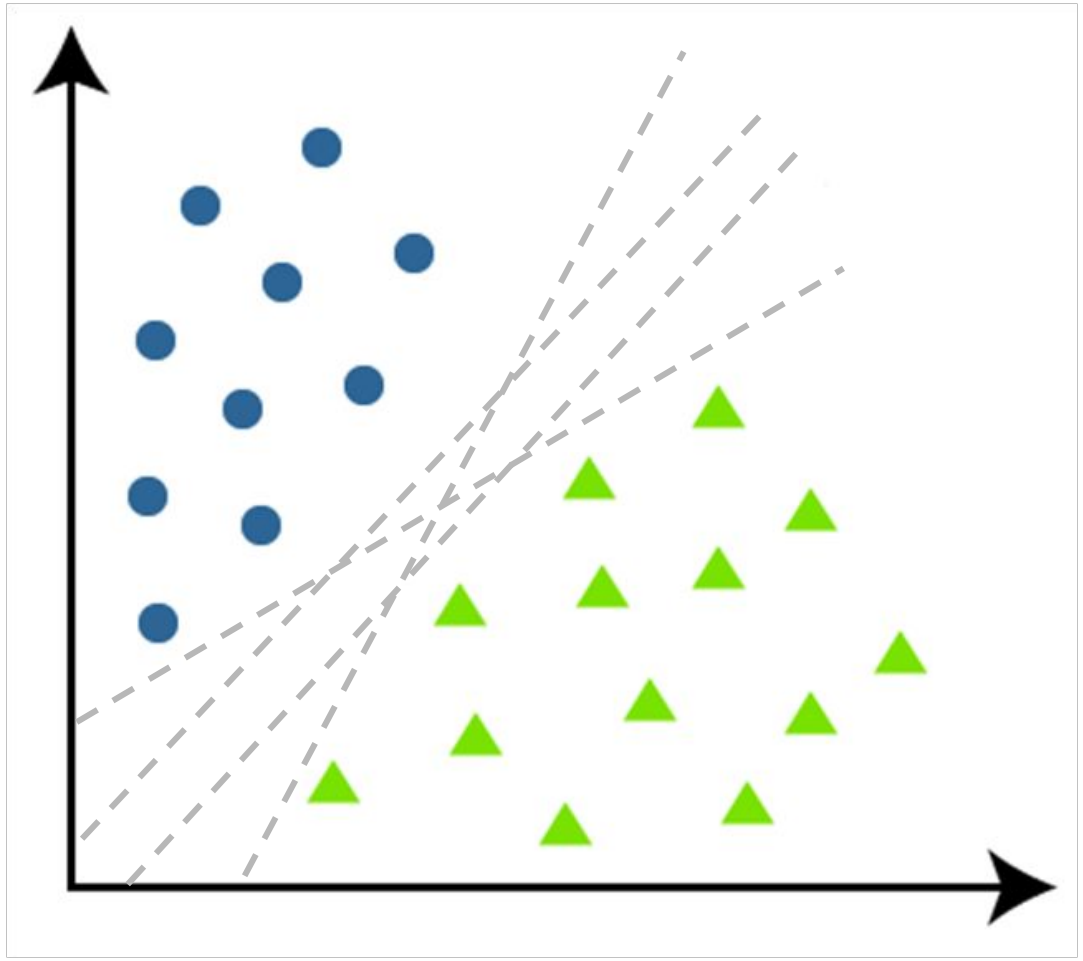
Tenemos N muestras [instancias].

Cada punto tiene 2 coordenadas: (x, y) [atributos].

Cada punto tiene un color (o figura): azul y verde (círculo o triángulo) [clases].

Queremos predecir la clase de un dato nuevo [?]

Support Vector Machines (SVM)



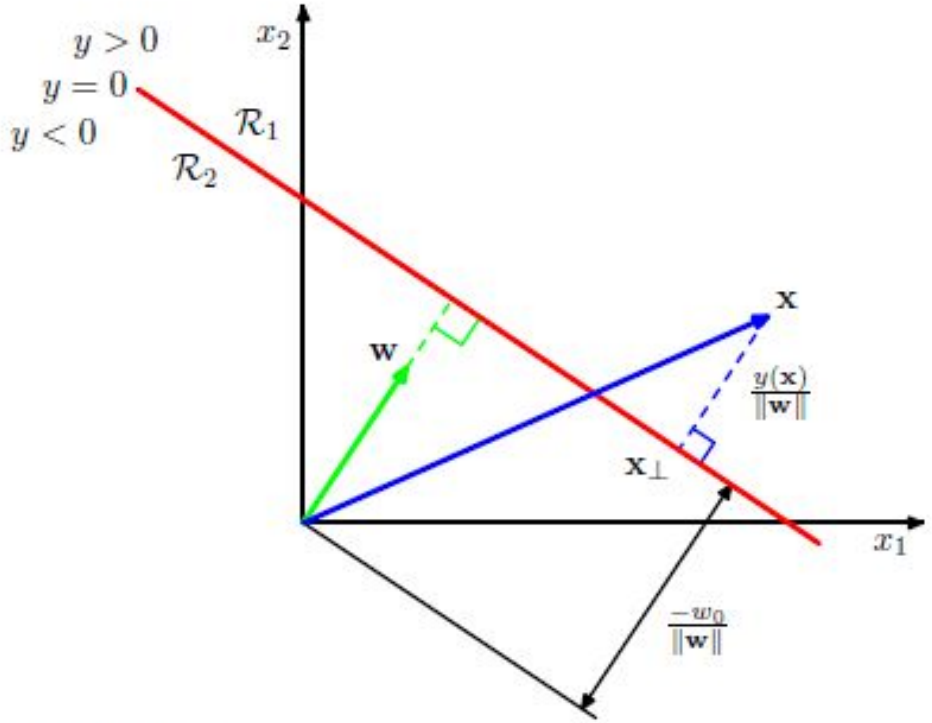
Buscamos una función

h(x, y) ⟶ color [hipótesis o modelo] que aproxime a la función objetivo.

Cómo elegimos h?

Funciones discriminantes

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0 \qquad \mathbf{x} = \mathbf{x}_\perp + r\frac{\mathbf{w}}{\|\mathbf{w}\|} \qquad r = \frac{y(\mathbf{x})}{\|\mathbf{w}\|}$$

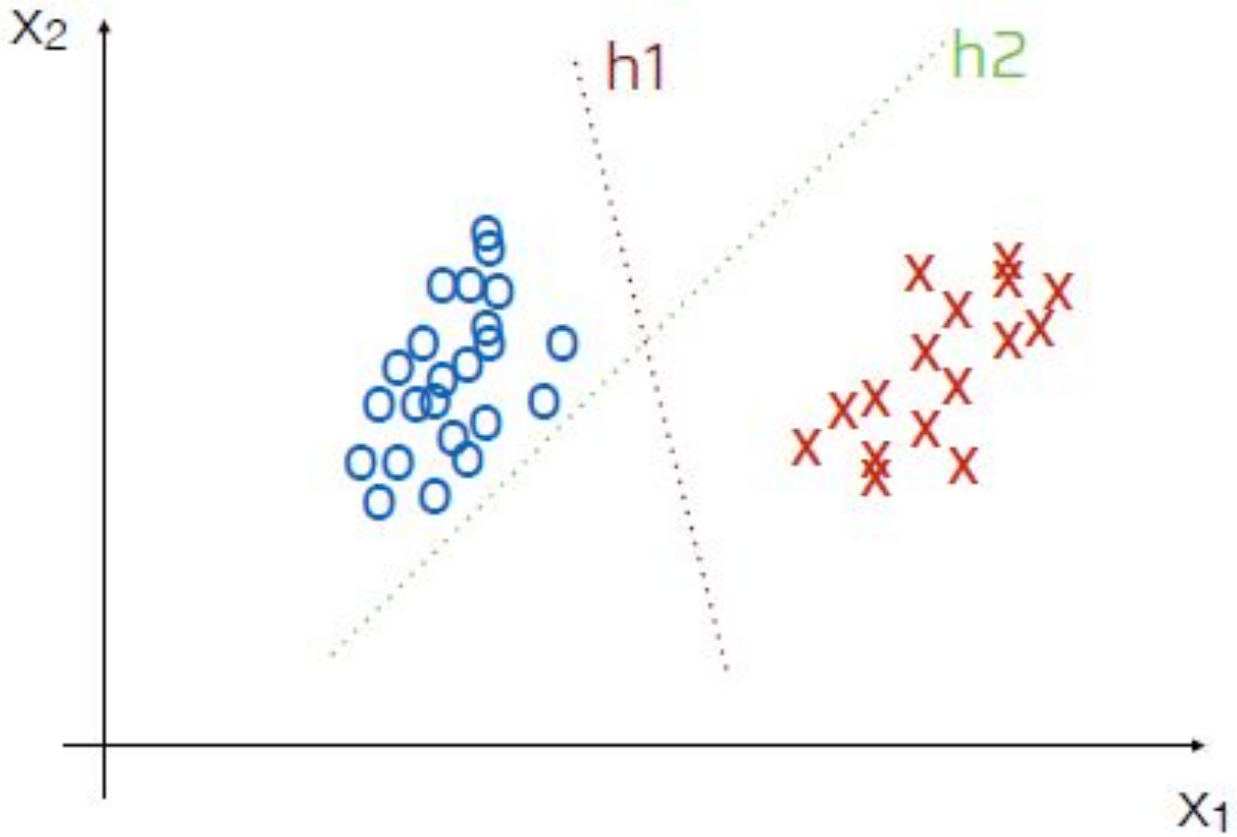


figuras del Bishop

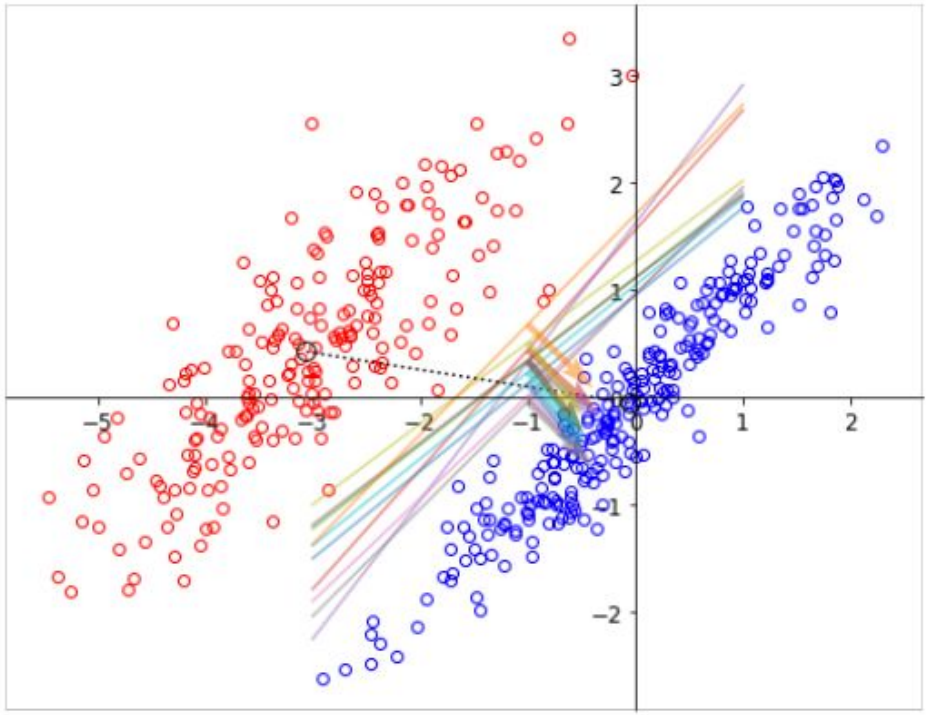
Algunas propiedades:

- El vector de pesos es perpendicular a la superficie de decisión
- La distancia al origen viene dada por el parámetro de “sesgo" w0 (por el umbral)
- La distancia de un punto a la superficie de decisión viene dada por y(x)

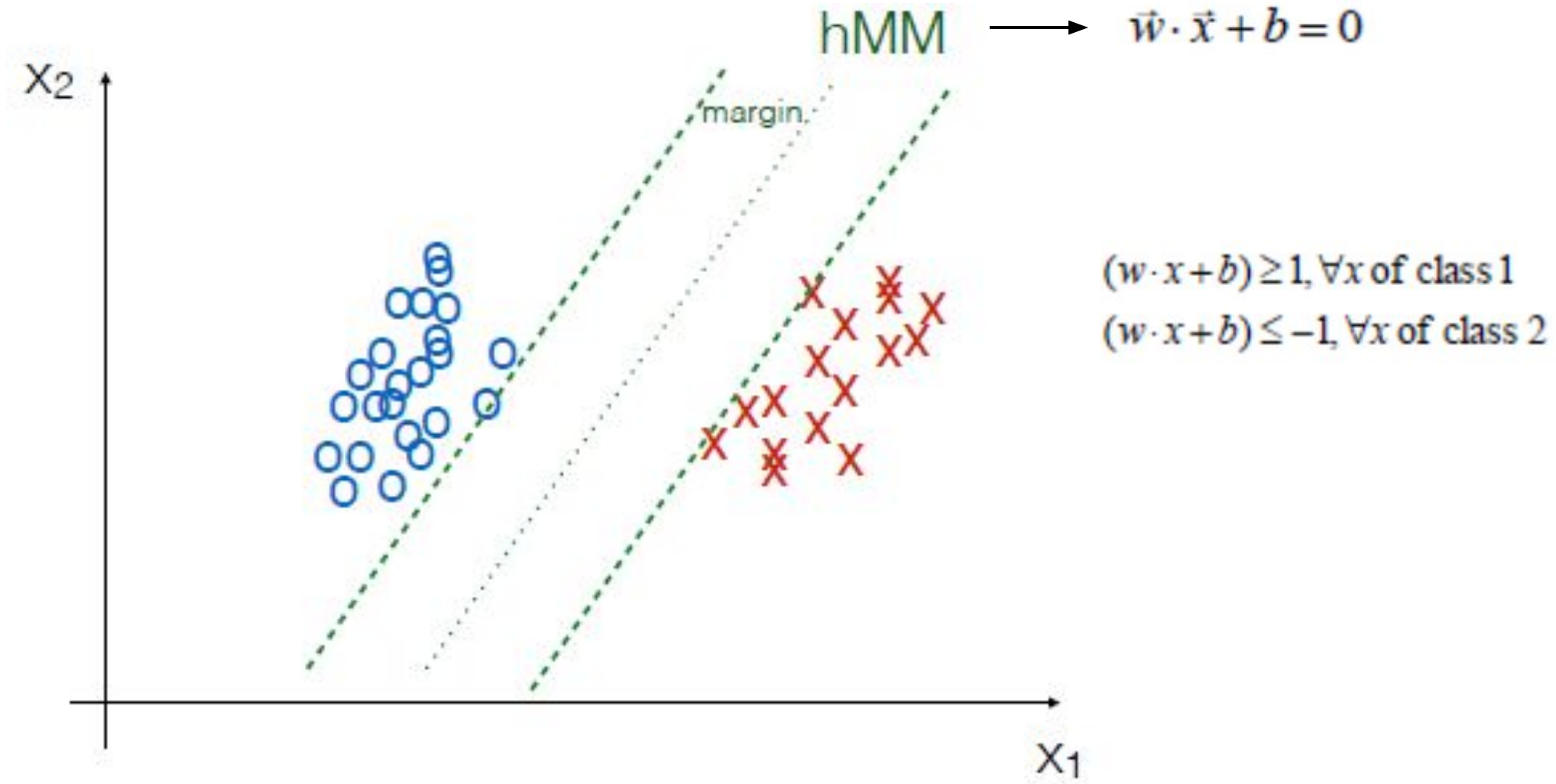
Modelos de maximización de márgenes



h1 y h2 don discriminantes

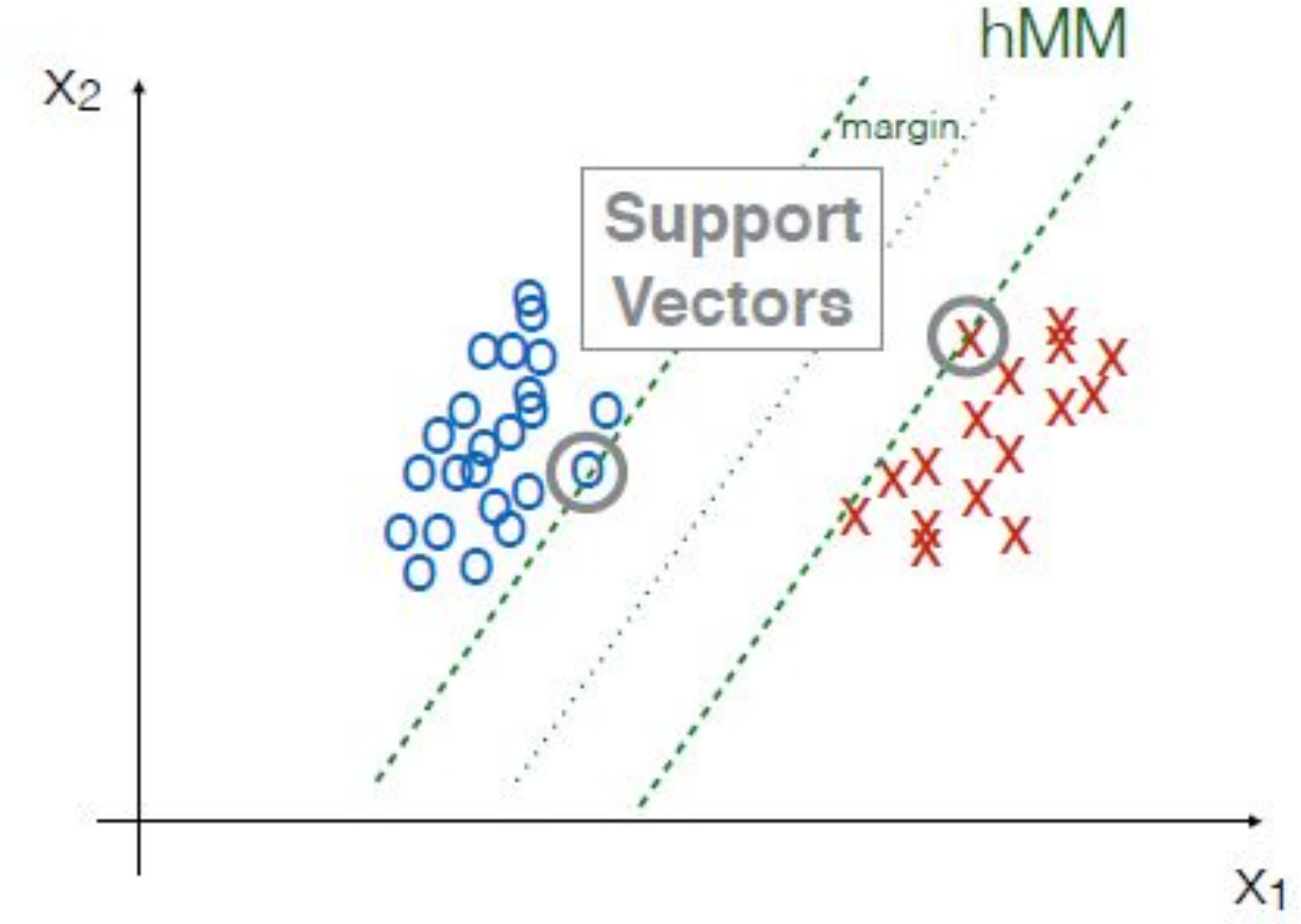


Modelos de maximización de márgenes



hMM minimiza el error de generalización (se puede demostrar [Vapnik, 1996])

Vectores de soporte



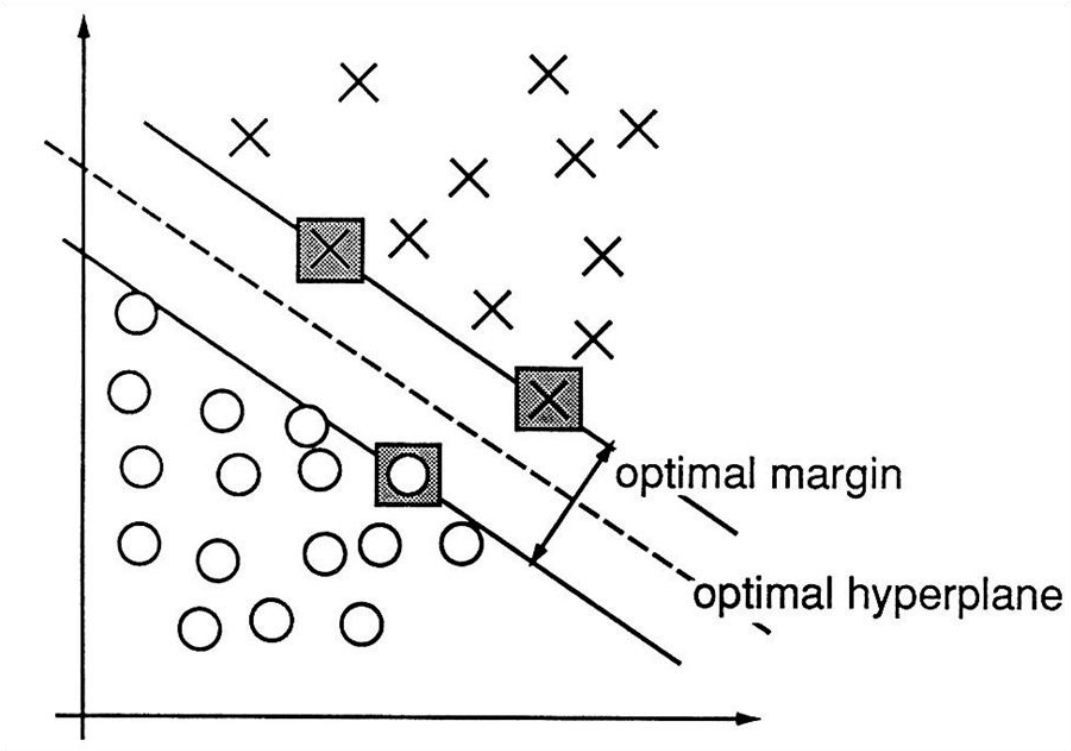
Los puntos que son difíciles de clasificar

Los más cercanos al márgen

Minimizar su distancia al márgen

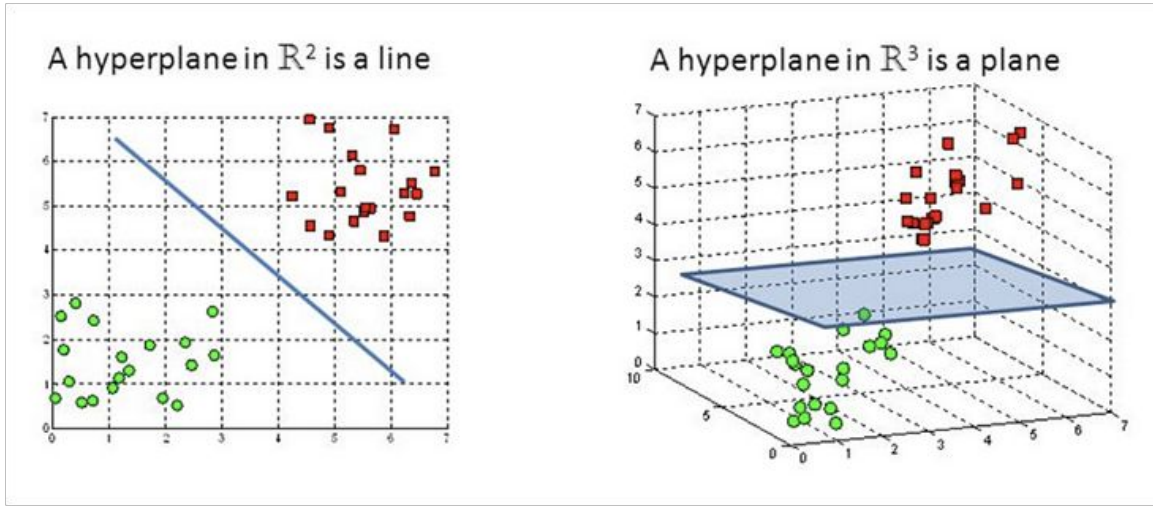
$$\min \frac{1}{2}\|w\|^2$$

Support Vector Machines (SVM)



C. Cortes & V. Vapnik 1995.

Machine Learning, 20, 273-297



En el hiperespacio M-dimensional de atributos, el contorno de decisión es un hiperplano

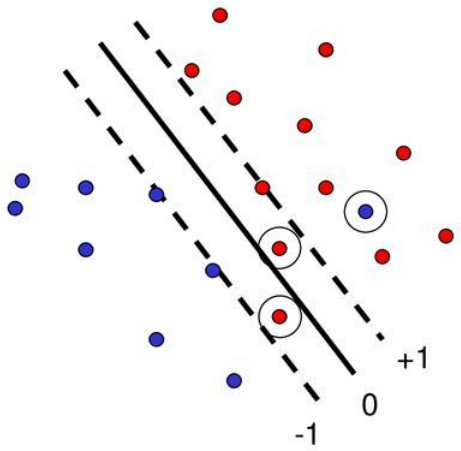
Support Vector Machines (SVM)

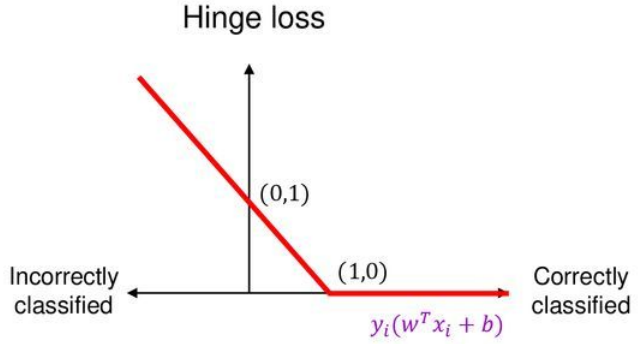
Función costo

"Soft margin" formulation

- Penalize margin violations using *hinge loss*:

$$\min_{w,b} \frac{\lambda}{2}\|w\|^2 + \sum_{i=1}^{n} \max[0, 1 - y_i(w^T x_i + b)]$$





Hinge-loss

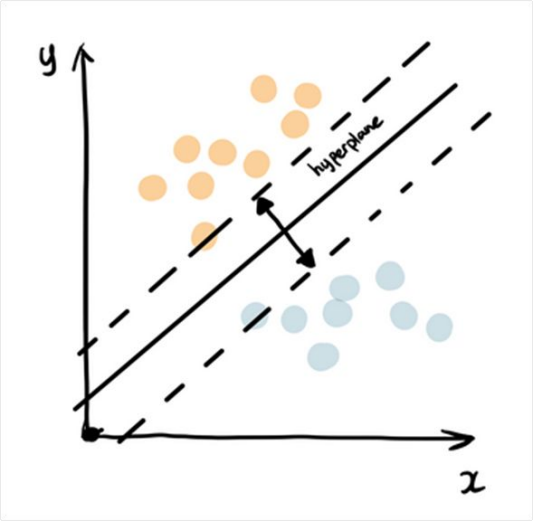
$$C \sum_{i=1}^{n} \underbrace{\max\left[1 - y_i(\mathbf{w}^T \mathbf{x}_i + b), 0\right]}_{\text{hinge loss}}$$

Estimo los pesos que maximizan el márgen con los vectores soporte

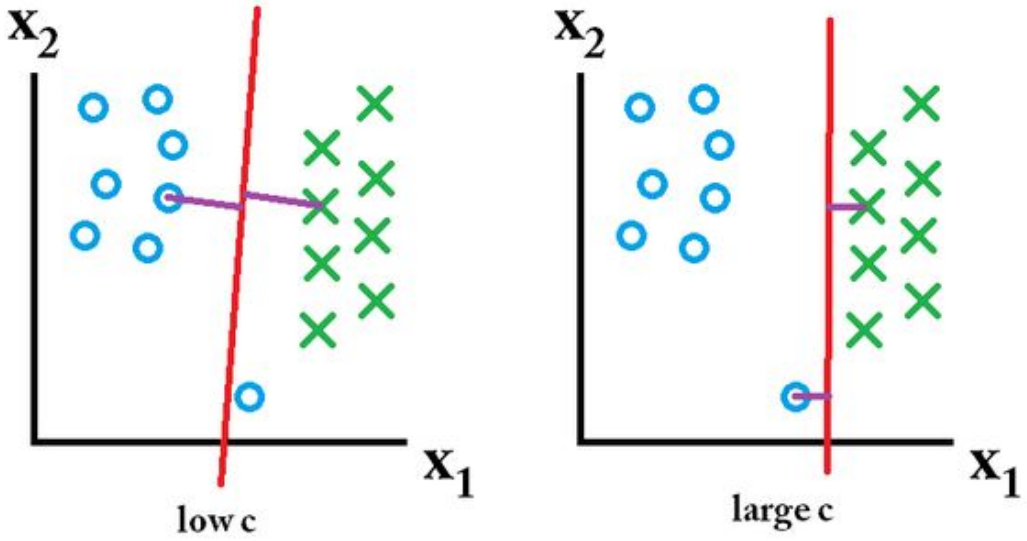
Penalizo errores de predicción

Agrego un hiperparámetro C para definir la flexibilidad de penalización

Support Vector Machines (SVM)

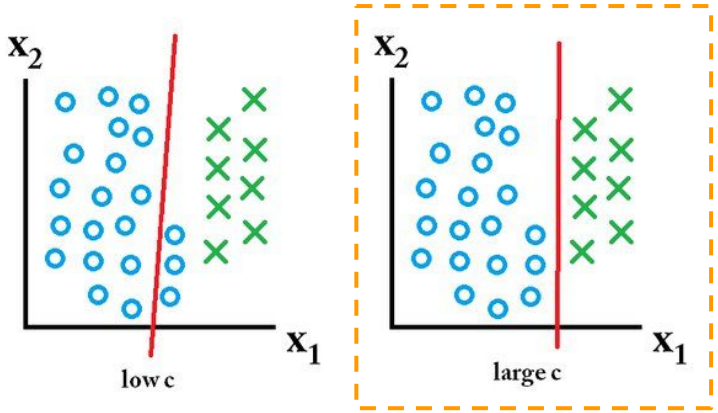


Hiperparámetro C: flexibilidad de márgenes

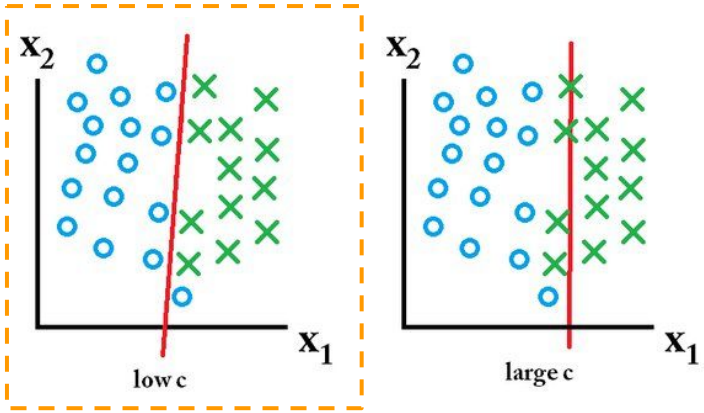


La selección del valor C dependerá del desempeño en validación cruzada, de acuerdo a la estructura de los datos

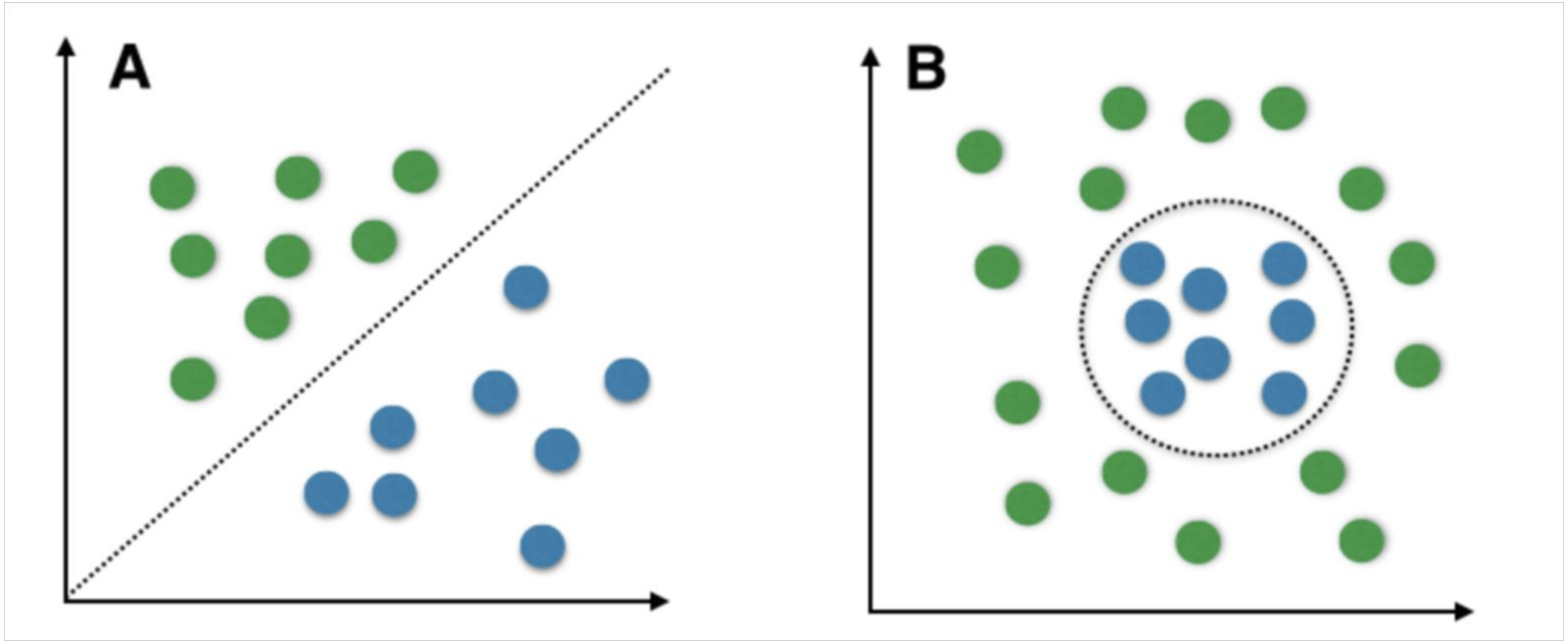
Escenario 1 (validación)



Escenario 2 (validación)



Support Vector Machines (SVM)



Linealmente separable

No linealmente separable

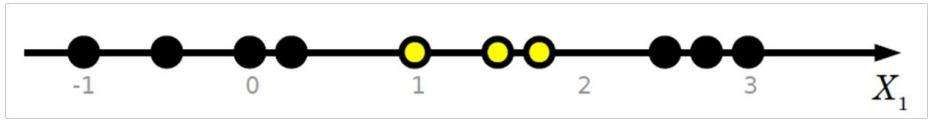
Support Vector Machines (SVM)

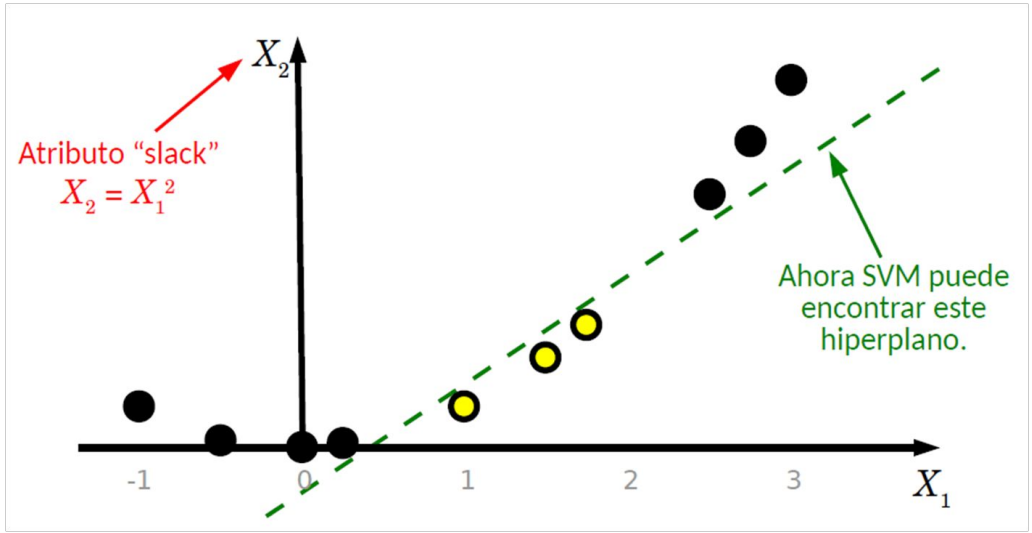
Transformación de vectores de atributos. Ej: Φ([*X1]) = ([X1, X1^*2])

👎 Expandir las transformaciones explícitamente es muy costoso

👍 Lo evitamos mediante kernels

Kernel trick





<u>Kernel</u>:
Generalización del producto interno que nos permite operar con nuevos atributos en forma implícita.
K(*x(1), x(2)) = <Φ(x(1)), Φ(x(2))>,* [*x(1), x(2)* instancias]

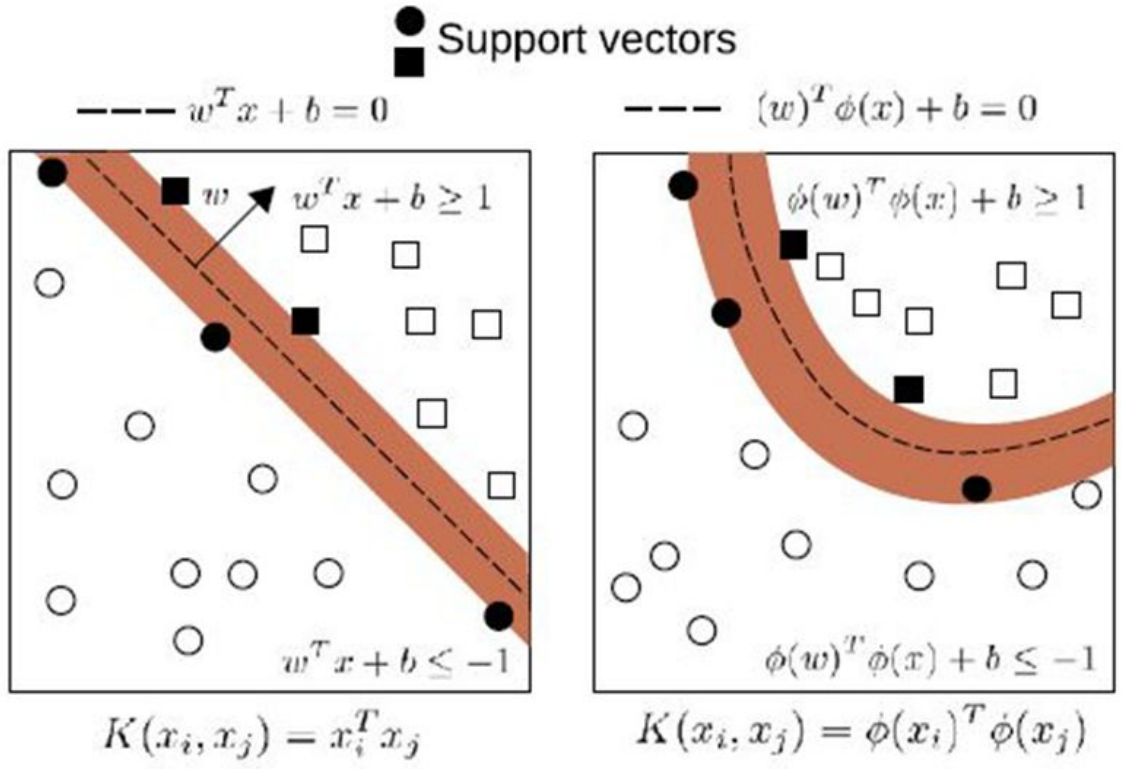
Si un algoritmo (ej. SVM) puede expresarse en términos de productos internos entre instancias, reemplazamos las apariciones de *<x(1), x(2)> por K(x(1), x(2))*

Asi, ejecutamos SVM implícitamente en dimensiones superiores *x(1), x(2)>* por *K(x(1), x(2))*

Support Vector Machines (SVM)

Kernels mas usados: lineal, polinomial, sigmoideo, RBF.

– Souza (2010), “Kernel Functions for Machine Learning Applications”



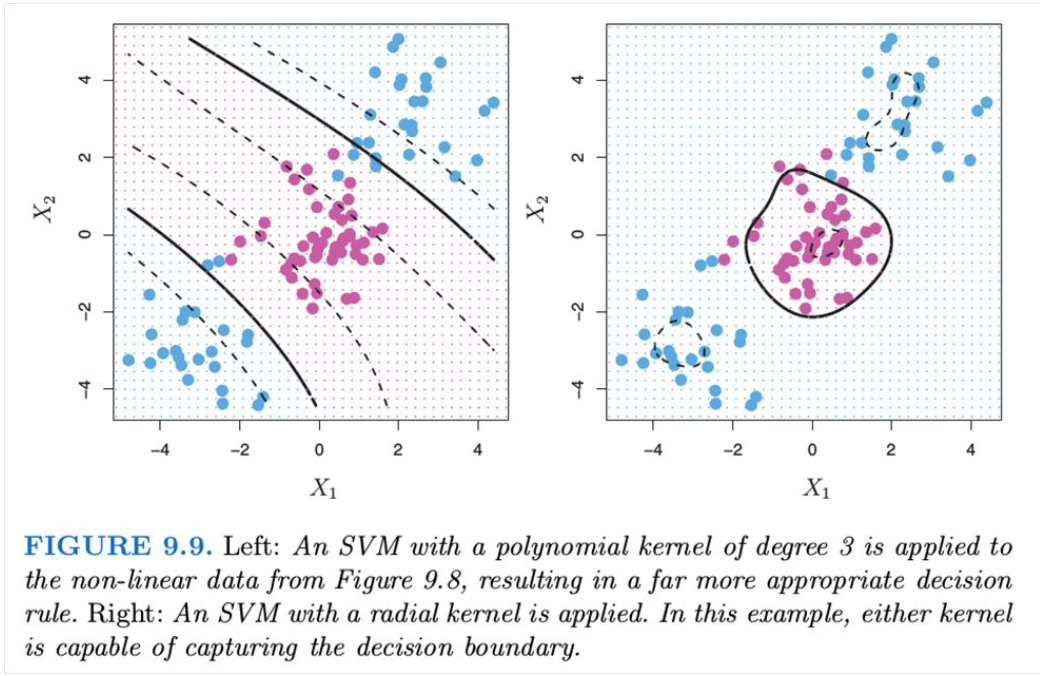
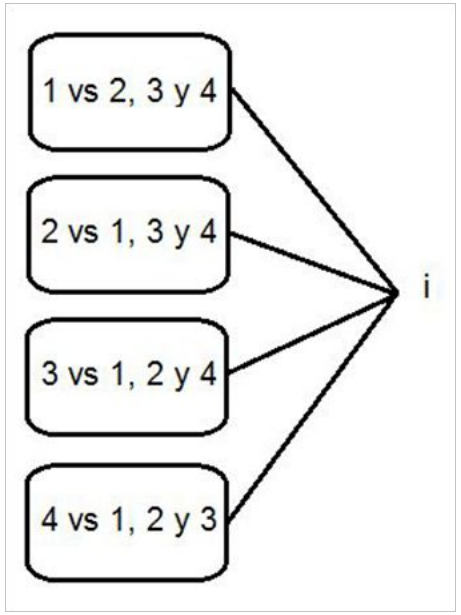


FIGURE 9.9. Left: *An SVM with a polynomial kernel of degree 3 is applied to the non-linear data from Figure 9.8, resulting in a far more appropriate decision rule.* Right: *An SVM with a radial kernel is applied. In this example, either kernel is capable of capturing the decision boundary.*

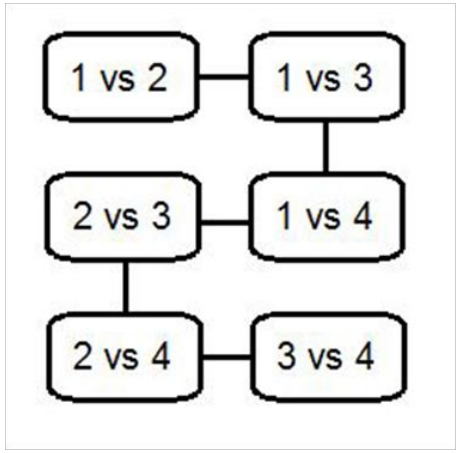
Support Vector Machines (SVM)

SVM sirve para clasificación binaria. Puedo pasar a multi-clase?

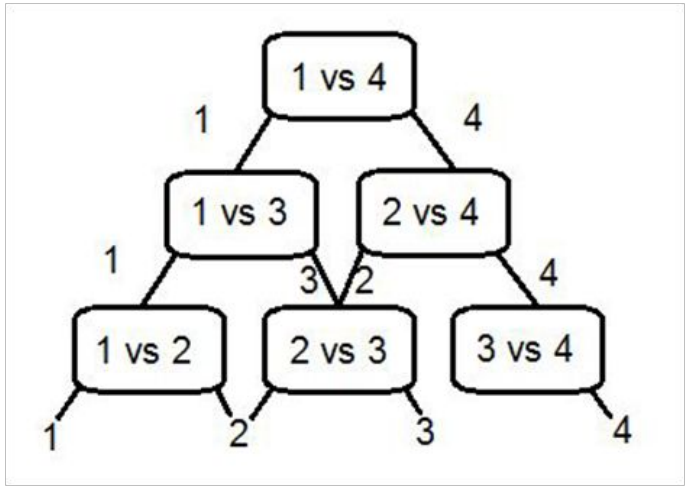
1 vs rest

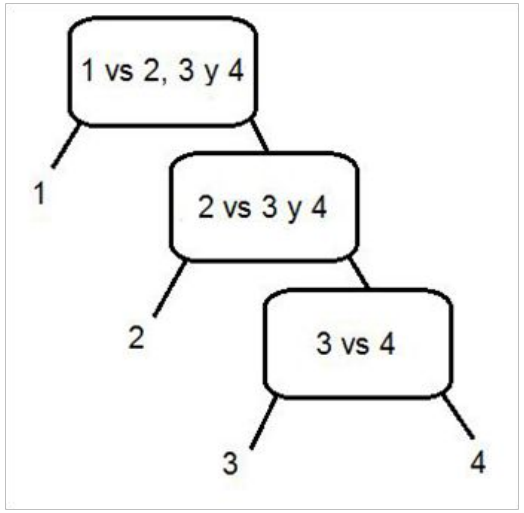


1 vs 1



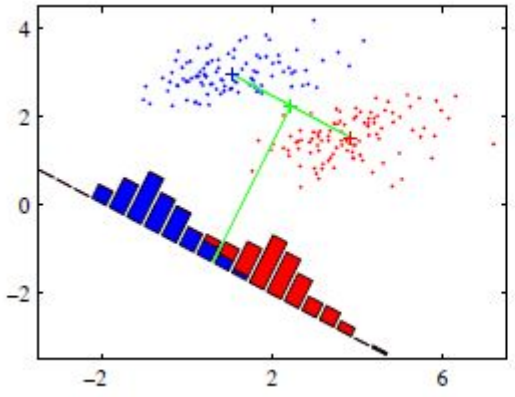
Directed Acyclic Graph (DAG)

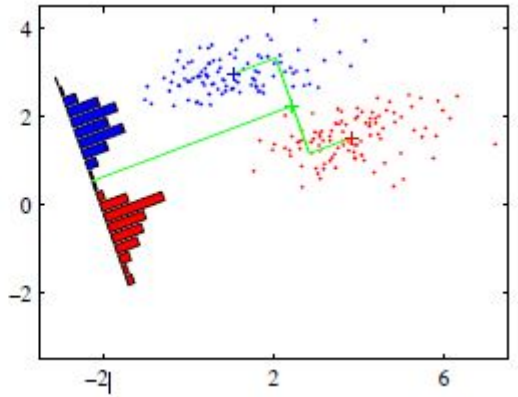


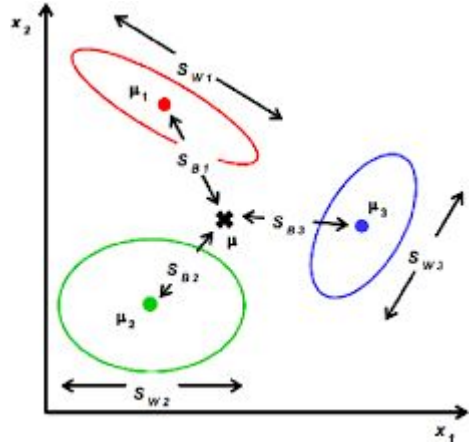


Linear Discriminant Analysis (LDA)

Calculo otra vez función discriminante, pero no maximizo el márgen, maximizo distancia entre centroides (Generalización de Discriminante Lineal de Fischer)







+ Problema de clasificación, admite multi-clase

+ Implica reducción de dimensionalidad y elegir la dirección correcta para aumentar la separabilidad

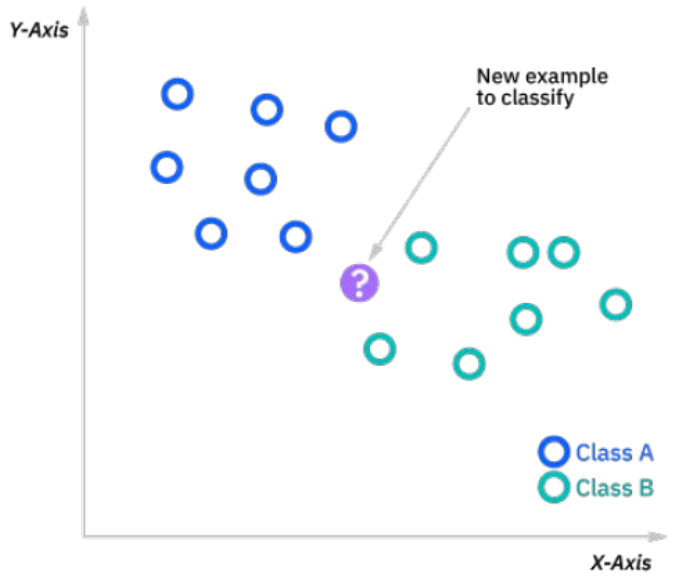
- Asume normalidad de las distribuciones

- No es muy flexible, pero sirve como base

K-Nearest Neighbors (KNN)

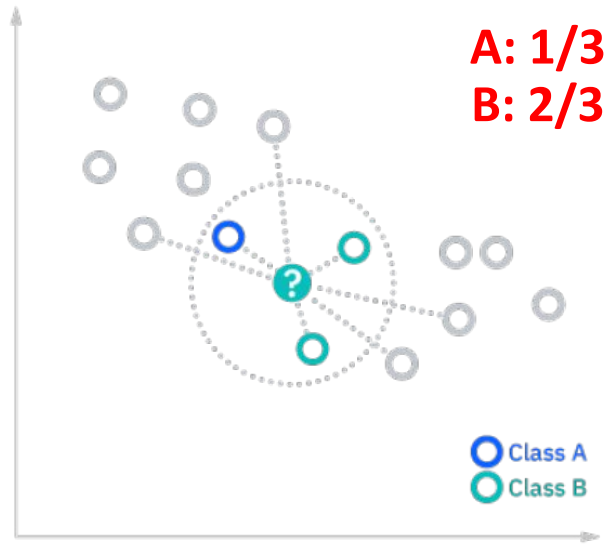
Mismo problema, predecir la clase de un dato nuevo (?), otro modelo

N instancias, 2 atributos. Cada punto tiene un color o clase



Puedo calcular las distancias del punto nuevo a los otros puntos

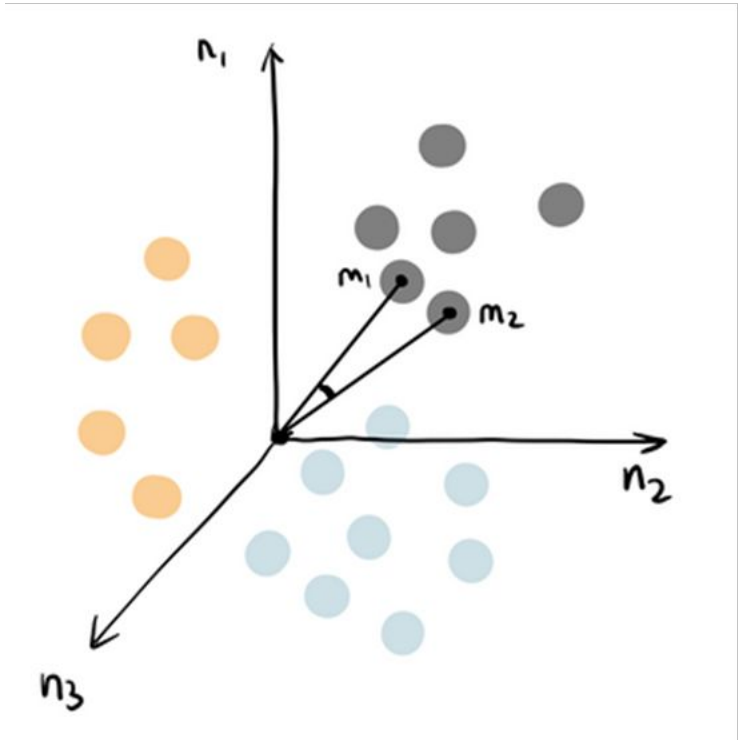
Me quedo con los K puntos que más cerca están (K vecinos más cercanos), y me fijo su color



Defino el color del nuevo punto de acuerdo al más probable por sus vecinos (votos).
También puedo normalizar y definir **probabilidades** de pertenecer a cada clase.

K-Nearest Neighbors (KNN)

Para M atributos, calculo la distancia de los puntos en el hiper-espacio de M-dimensiones (1 para cada atributo)

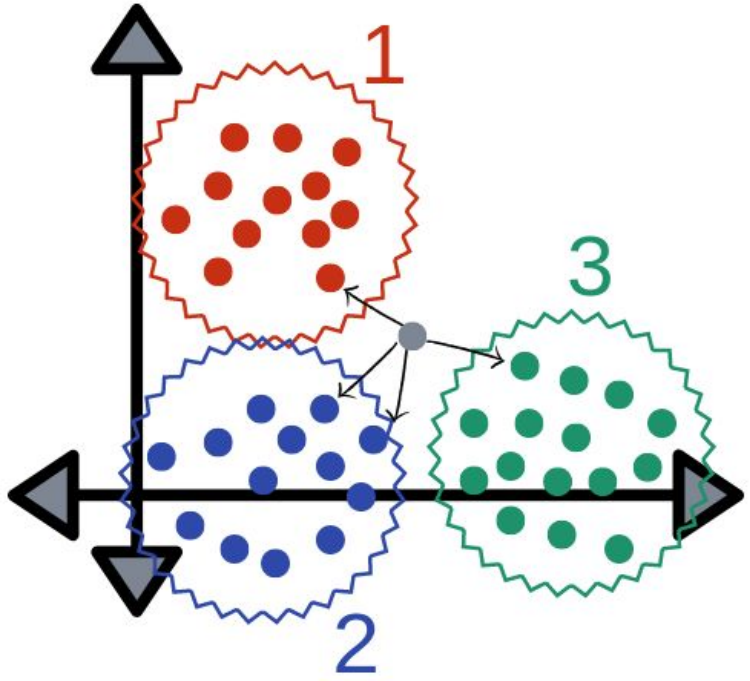


Puedo usar distintas métricas de distancia: Euclídea, Manhattan

$$\sum_{i=1}^{n} |x_i - y_i|$$

$$\sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

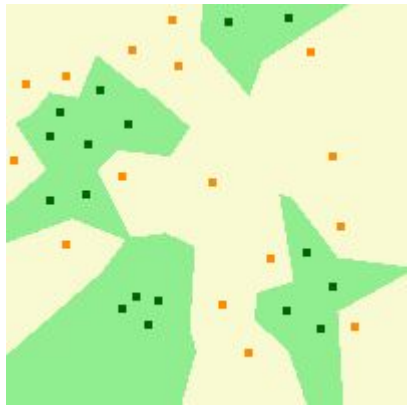
Admite naturalmente la posibilidad de resolver un problema multi-clase



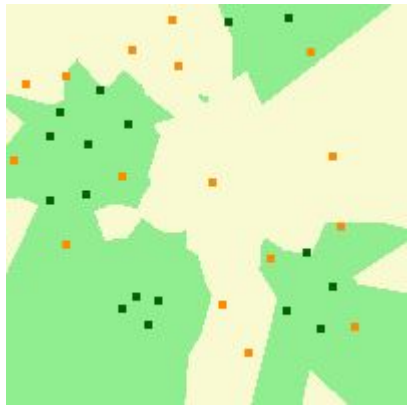
K-Nearest Neighbors (KNN)

Contornos de decisión

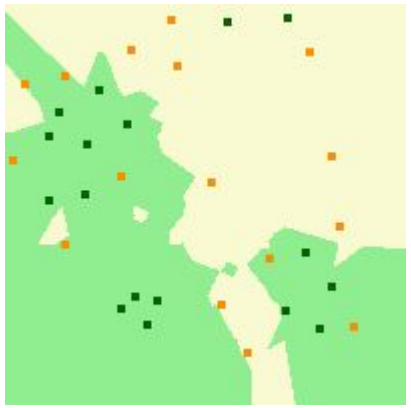
K = 1



K = 3



K = 5



K = 7

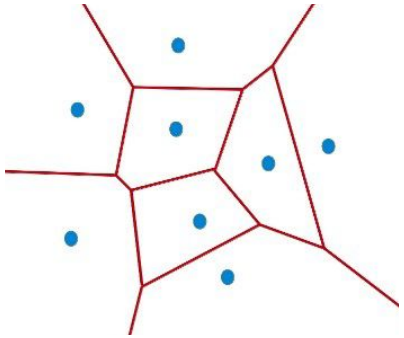


Diagrama de Voronoi

El K es un hiperparámetro del modelo, busco el óptimo con validación.

Para K bajo, puede sobreajustar a los datos de entrenamiento.

A medida que aumento el K, contornos más suaves, pero más errores.

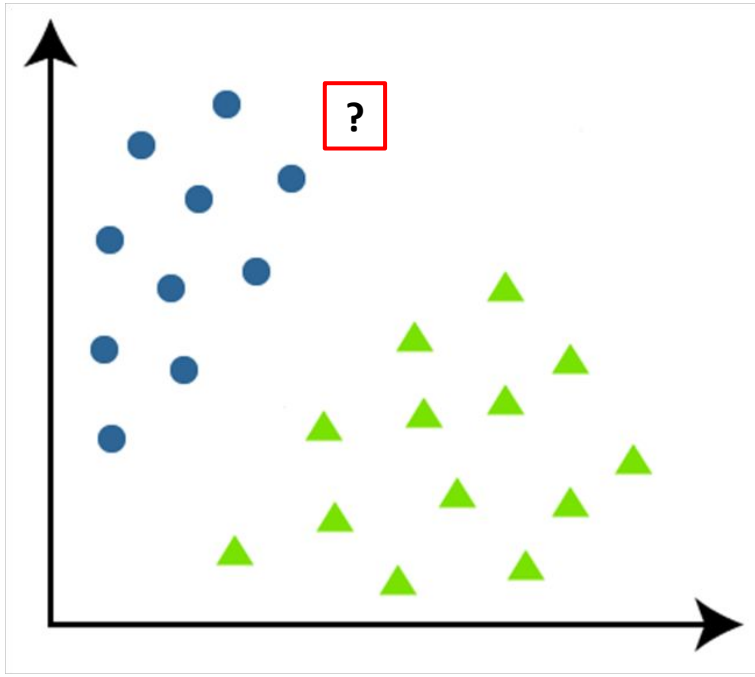
Si K es igual a N (instancias), en definitiva voy a elegir la clase que más tengo como la más probable, no necesito calcular distancias.

K-Nearest Neighbors (KNN)

+ La técnica es simple
+ El entrenamiento es rápido
+ Permite resolver problemas multi-clase
+ Permite obtener probabilidades de pertenecer a cada clase
+ Podría usar pesos y que los votos de los vecinos más cercanos pesen más
- El modelo ocupa mucho espacio en disco (cálculo de todas las distancias)
- Requiere mayores esfuerzos en ingeniería de atributos
    - Escala de atributos afecta a las distancias (normalización)
    - Atributos irrelevantes afectan a las distancias (selección de atributos)

Naive Bayes (NB)

Mismo problema, predecir la clase de un dato nuevo [?], otro modelo



N instancias, 2 atributos.
Cada punto tiene un color o clase

Si tuviera la probabilidad de pertenecer a una clase k, dados ciertos atributos x, es la más probable a posteriori

$$k_{\text{MAP}} = \underset{k}{\operatorname{argmax}}\ P(Y = k \mid X = x)$$

Aplicando Teorema de Bayes sobre probabilidad condicional

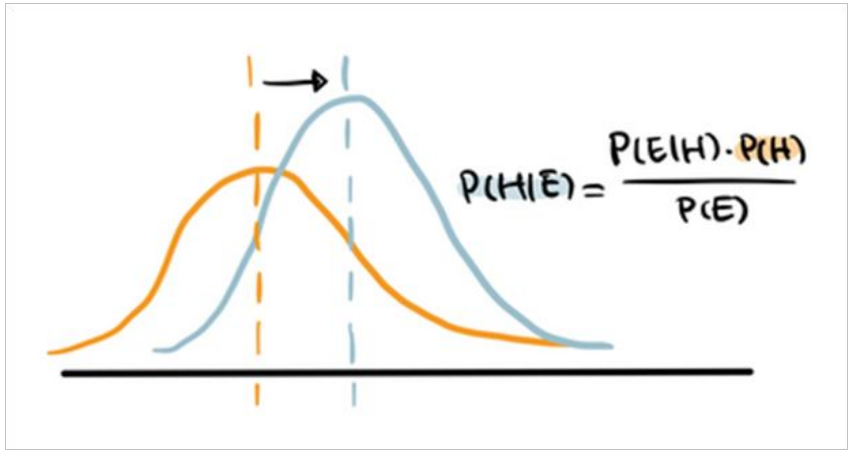
$$k_{\text{MAP}} \overset{\text{Bayes}}{=} \underset{k}{\operatorname{argmax}}\ \frac{P(Y = k) \cdot P(X = x \mid Y = k)}{\cancel{P(X = x)}}$$

→ no depende de k

Clasificador de Bayes

$$k_{\text{MAP}} = \underset{k}{\operatorname{argmax}}\ P(Y = k) \cdot P(X = x \mid Y = k)$$

Naive Bayes (NB)



Clasificador de Bayes

$$k_{\text{MAP}} = \underset{k}{\operatorname{argmax}}\ P(Y = k) \cdot P(X = x \mid Y = k)$$

Entonces, para clasificar una nueva instancia dada por x, alcanza con calcular P(Y = k) (prior de la clase k) y P(X = x | Y = k) (distribución de instancias en la clase k), y listo...

Problema: conocer P(X = x | Y = k) es casi siempre imposible

Lo que hago es suponer que los atributos son independientes (suposición naive)

$$k_{\text{MAP}} = \underset{k}{\operatorname{argmax}}\ P(Y = k) \cdot P(X_1 = x_1 \wedge X_2 = x_2 \ldots \wedge X_p = x_p | Y = k)$$

producto de probabilidades

Clasificador de Naive Bayes

$$k_{\text{NB}} = \underset{k}{\operatorname{argmax}}\ P(Y = k) \cdot \prod_{i=1}^{p} P(X_i = x_i \mid Y = k)$$

Puedo calcular probabilidad de ocurrencia a partir de los datos de entrenamiento

Naive Bayes (NB)

	Atributos				Clase
Instancia	Cielo	Temperatura	Humedad	Viento	Va a correr?
1	sol	calor	alta	débil	No
2	sol	calor	alta	fuerte	No
3	nublado	calor	alta	débil	Sí
4	lluvia	templado	alta	débil	Sí
5	lluvia	frío	normal	débil	Sí
6	lluvia	frío	normal	fuerte	No
7	nublado	frío	normal	fuerte	Sí
8	sol	templado	alta	débil	No
9	sol	frío	normal	débil	Sí
10	lluvia	templado	normal	débil	Sí
11	sol	templado	normal	fuerte	Sí
12	nublado	templado	alta	fuerte	Sí
13	nublado	calor	normal	débil	Sí
14	lluvia	templado	alta	fuerte	No

P(Sí) = 9/14

P(No) = 5/14

Cielo				
	Sí	No	P(Sí)	P(No)
sol	2	3	2/9	3/5
nublado	4	0	4/9	0/5
lluvia	3	2	3/9	2/5
total	9	5	100%	100%

Temperatura				
	Sí	No	P(Sí)	P(No)
calor	2	2	2/9	2/5
templado	4	2	4/9	2/5
frío	3	1	3/9	1/5
total	9	5	100%	100%

Y así con los atributos Humedad y Viento también

Naive Bayes (NB)

Cielo				
	Sí	No	P(Sí)	P(No)
sol	2	3	2/9	3/5
nublado	4	0	4/9	0/5
lluvia	3	2	3/9	2/5
total	9	5	100%	100%

Temperatura				
	Sí	No	P(Sí)	P(No)
calor	2	2	2/9	2/5
templado	4	2	4/9	2/5
frío	3	1	3/9	1/5
total	9	5	100%	100%

P(Sí) = 9/14

P(No) = 5/14

P(Cielo=sol |Corre=Sí) = 2/9

P(Temperatura=templado | Corre=Sí) = 4/9

…

P(Cielo=sol | Corre=No) = 3/5

P(Temperatura=templado | Corre=No) = 2/5

…

Para clasificar (sol, templado, alta, fuerte), se calcula

P(Corre=Sí | x) = P(x | Corre=Sí) * P(Sí) = 2/9 * 4/9 * … * … * 9/14

P(Corre=No | x) = P(x | Corre=No) * P(No) = 3/5 * 2/5 * … * … * 5/14

Observación: Si las variables fuesen numéricas, podría calcular P como la ocurrencia en bins

Naive Bayes (NB)

+ El entrenamiento es rápido
+ No requiere mucho almacenamiento en disco
+ Permite obtener probabilidades de pertenecer a cada clase
+ Permite resolver problemas multi-clase
- Asume independencia de atributos
- Problemas con probabilidades cero (zero-frequency problem)
    - No es muy bueno si los datos son esparsos
    - No es muy bueno en datos muy desbalanceados
    - Pocos datos
* Muy usado en clasificación de textos (NLP), tradicionalmente

Medidas de desempeño

TVP/TVN; precision/recall; curva ROC y AUC → Requieren discriminar positivos y negativos → Clasificación binaria

Accuracy se salva

Puedo generalizar para el problema multi-clase?

1 vs rest ó 1 vs 1

Medidas para cada comparación en particular

Puedo tener 1 por clase (1vsR) o muchas (1vs1)

Puedo definir medidas globales llamadas micro-average y macro-average

Ejemplo Precision, vale para las otras

$$P_{micro} = \frac{TP_a + TP_b + \ldots + TP_n}{(TP_a + TP_b + \ldots + TP_n) + (FP_a + FP_b + \ldots + FP_n)}$$

$$P_{macro} = \frac{P_a + P_b + \ldots + P_n}{k}$$

A los Colabs…