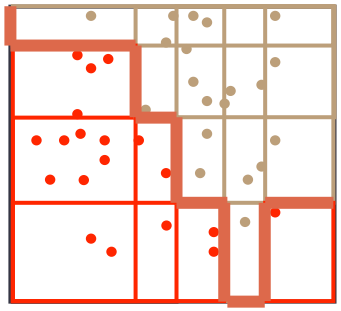
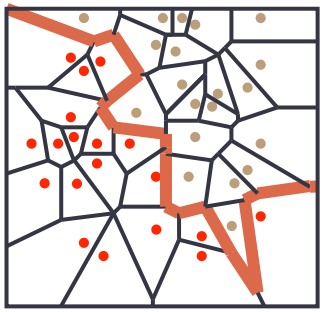
MÁQUINAS DE SOPORTE VECTORIAL

Minería de Datos: Preprocesamiento y clasificación

Salvador García
salvagl@decsai.ugr.es

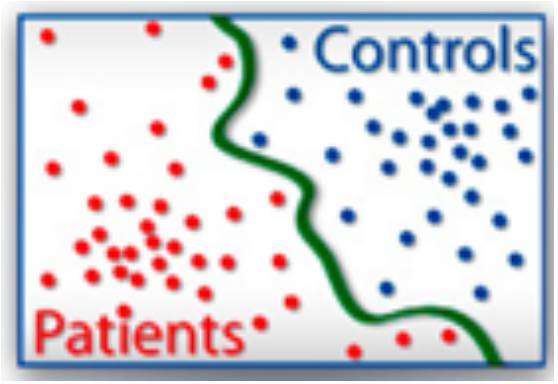
- Un *clasificador* puede ser un conjunto de reglas, un árbol de decisión, uso de técnias de vecino más cercano, …





- Una red neuronal, una máquina de vectores soporte …

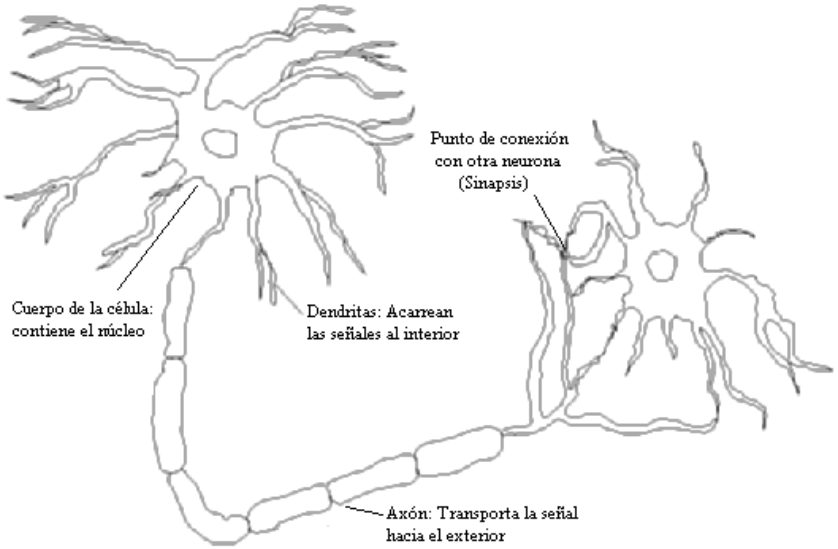
(algoritmos no interpretables pero de mucha precisión)

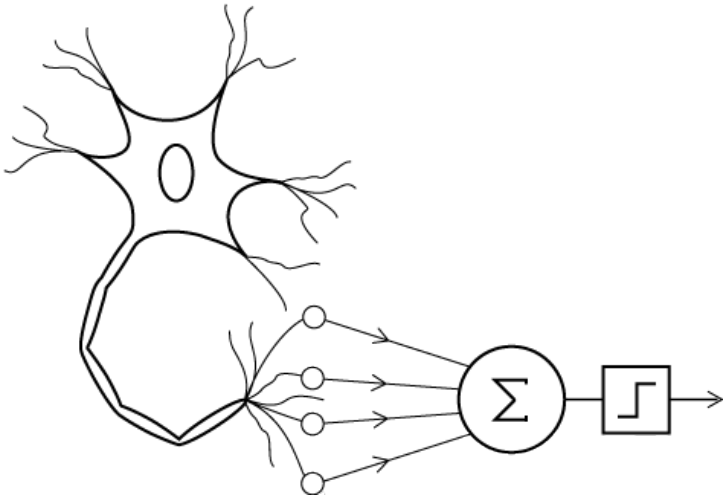


Clasificadores basados en redes neuronales

Redes neuronales

- Surgieron como un intento de emulación de los sistemas nerviosos biológicos
- Actualmente: computación modular distribuida mediante la interconexión de una serie de procesadores (neuronas) elementales

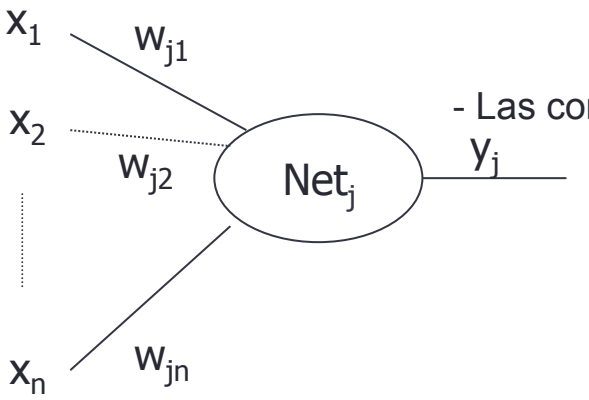




Clasificadores basados en redes neuronales y/o funciones

- **Ventajas:**
  - Habitualmente gran tasa de acierto en la predicción
  - Son más robustas que los árboles de decisión por los pesos
  - Robustez ante la presencia de errores (ruido, outliers,…)
  - Gran capacidad de salida: nominal, numérica, vectores,…
  - Eficiencia (rapidez) en la evaluación de nuevos casos
  - Mejoran su rendimiento mediante aprendizaje y éste puede continuar después de que se haya aplicado al conjunto de entrenamiento
- **Desventajas:**
  - Necesitan mucho tiempo para el entrenamiento
  - Entrenamiento: gran parte es ensayo y error
  - Poca (o ninguna) interpretabilidad del modelo (caja negra)
  - Difícil de incorporar conocimiento del dominio
  - Los atributos de entrada deben ser numéricos
  - Generar reglas a partir de redes neuronales no es inmediato
  - Pueden tener problemas de sobreaprendizaje
- **¿Cuándo utilizarlas?**
  - Cuando la entrada tiene una dimensión alta
  - La salida es un valor discreto o real o un vector de valores
  - Posibilidad de datos con ruido
  - La forma de la función objetivo es desconocida
  - La interpretabilidad de los resultados no es importante

Clasificadores basados en redes neuronales



- Las señales que llegan a las dendritas se representan como $x_1, x_2, \ldots, x_n$
- Las conexiones sinápticas se representan por unos pesos w_{j1}, w_{j2}, w_{jn} que ponderan (multiplican) a las entradas. Si el peso entre las neuronas *j* e *i* es:
  a) positivo, representa una sinapsis excitadora
  b) negativo, representa una sinapsis inhibidora
  c) cero, no hay conexión

- La acción integradora del cuerpo celular (o actividad interna de cada célula) se presenta por

$$Net_j = w_{j1} \cdot x_1 + w_{j2} \cdot x_2 + \ldots + w_{jn} \cdot x_n = \sum_{i=1}^{n} w_{ji} \cdot x_i$$

- La salida de la neurona se representa por y_j. Se obtiene mediante una función que, en general, se denomina función de salida, de transferencia o de activación. Esta función depende de Net_j y de un parámetro θ_j que representa el umbral de activación de la neurona

$$y_j = f(Net_j - \theta_j) = f\left(\sum w_{ji} \cdot x_i - \theta\right)$$

Clasificadores basados en redes neuronales

- **Función de escalón**. Representa una neurona con sólo dos estados de activación: activada (1) y inhibida (0 ó -1)

$$y_j = H(Net_j - \theta_j) = \begin{cases} 1, & si\ Net_j >= \theta_j \\ -1, & si\ Net_j < \theta_j \end{cases}$$

- **Función lineal**:

$$y_j = Net_j - \theta_j$$

- **Función lineal a tramos**:

$$y_j = \begin{cases} 1, & si\ Net_j >= \theta_j + a \\ Net_j - \theta_j, & si\ \left|Net_j - \theta_j\right| < a \\ -1, & si\ Net_j < \theta_j - a \end{cases}$$

- **Función sigmoidal**:

$$y_j = \frac{1}{1 + e^{-\lambda(Net_j - \theta_j)}} \qquad y_j = \frac{2}{1 + e^{-\lambda(Net_j - \theta_j)}} - 1$$

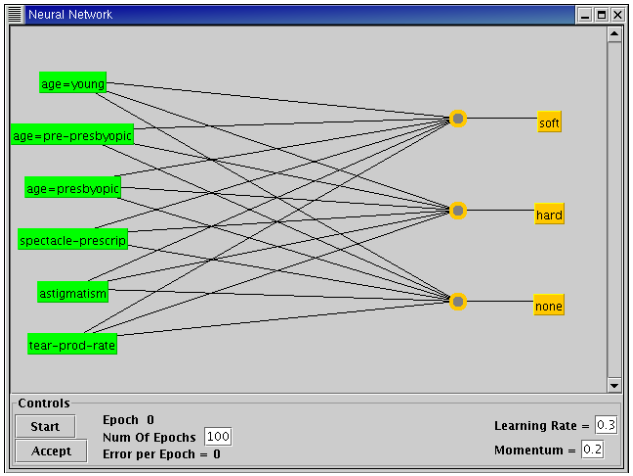
- **Función base radial**:

$$y_j = e^{-\left(\frac{Net_j - \theta_j}{\sigma}\right)^2}$$

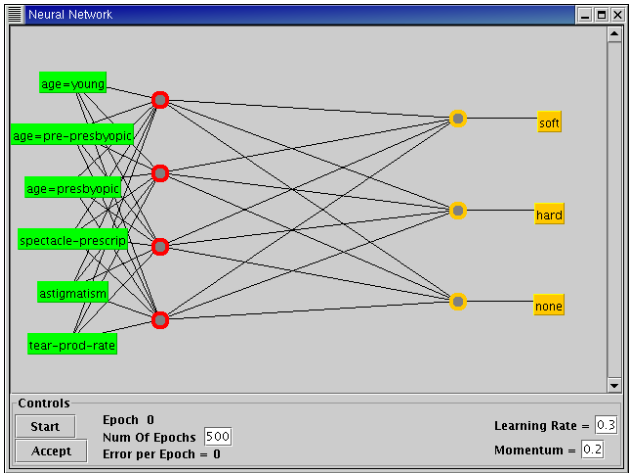
Clasificadores basados en redes neuronales

- El aprendizaje de la estructura de una red neuronal requiere experiencia, aunque existen algunas guías
- Entradas: Por cada variable numérica o binaria/booleana se pone una neurona de entrada. Por cada variable nominal (con más de dos estados) se pone una neurona de entrada por cada estado posible.
- Salidas: Si es para predicción se pone una única neurona de salida por cada valor de la variable clase
- Capas ocultas. Hay que indicar cuántas capas y cuantas neuronas hay que poner en cada capa.
- Cuando la red neuronal está entrenada, para clasificar una instancia, se introducen los valores de la misma que corresponden a las variables de los nodos de entrada.
  - La salida de cada nodo de salida indica la probabilidad de que la instancia pertenezca a esa clase
  - La instancia se asigna a la clase con mayor probabilidad

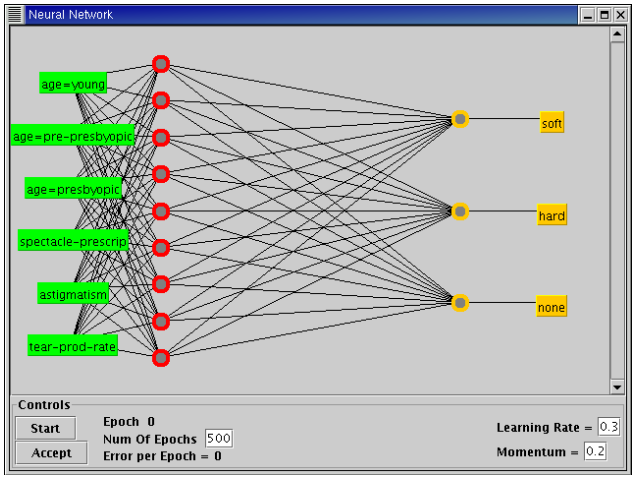
Clasificadores basados en redes neuronales



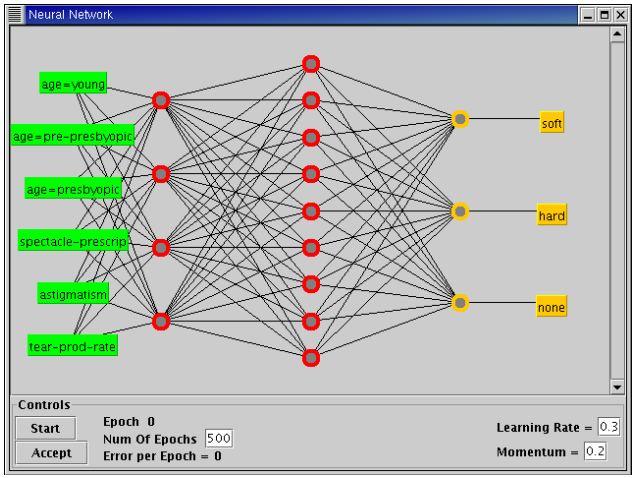
Epoch 0
Num Of Epochs 100
Error per Epoch = 0
Learning Rate = 0.3
Momentum = 0.2



Epoch 0
Num Of Epochs 500
Error per Epoch = 0
Learning Rate = 0.3
Momentum = 0.2



Epoch 0
Num Of Epochs 500
Error per Epoch = 0
Learning Rate = 0.3
Momentum = 0.2



Epoch 0
Num Of Epochs 500
Error per Epoch = 0
Learning Rate = 0.3
Momentum = 0.2

SVMs: Un poco de historia

- Las máquinas de vectores soporte (Support Vector Machines –SVMs-) se introducen en 1992 por Boser, Guyon, Vapnik en COLT-92

- Se populariza por la comunidad de RNAs, pero hoy día es un campo muy activo en todo el campo de Machine Learning

- ¡Ojo! Las SVMs son un caso particular de “Kernel Machines” (KM)← amplia familia de algoritmos de aprendizaje

Preliminares

- Objetivo de estos algoritmos: Detectar y explotar patrones complejos en los datos

- Problemas típicos:
  - ¿cómo representar patrones complejos?
  - ¿cómo excluir ejemplos espurios?

- El primero es un problema computacional. El segundo es un problema estadístico.

Algunas notas sobre el razonamiento

- Los KM definen la clase de los posibles ejemplos introduciendo una noción **IMPLÍCITA** entre los datos
  - Esto es en cierto sentido similar a k-NN
  - Es fundamental elegir las características relevantes
- Los KM usan el producto escalar (**inner product**) entre los patrones de entrada
  - Este producto escalar es usado como una forma de establecer la similitud entre elementos
- Ventaja → muchos algoritmos estándar se pueden reescribir de forma que sólo necesiten el producto escalar entre ítems de entrada

Recordando: Producto escalar

- Producto escalar entre vectores:

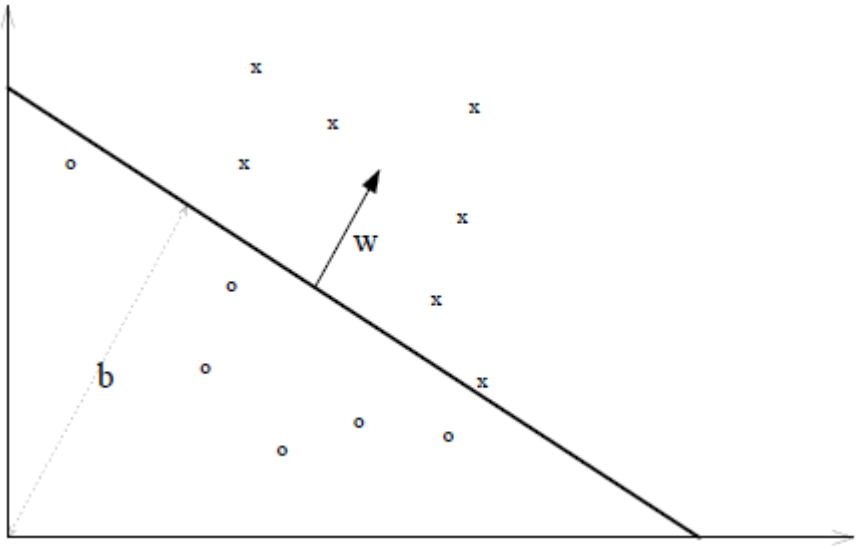
$$\langle \bar{x}, \bar{z} \rangle = \sum_i x_i z_i$$

- Este producto escalar también puede usarse para definir un plano en el espacio

Recordando: Hiperplano

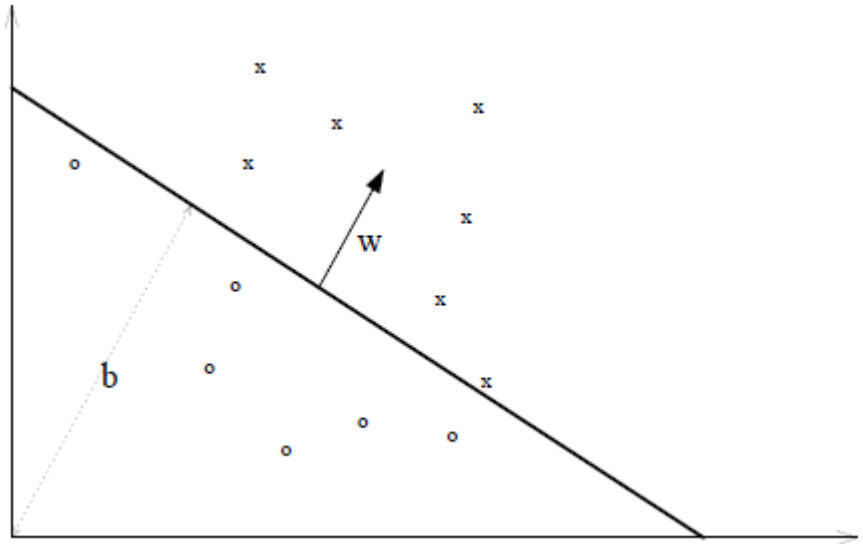
- *w* es el vector normal al plano. *b* es la distancia al origen como referencia al vector *w*

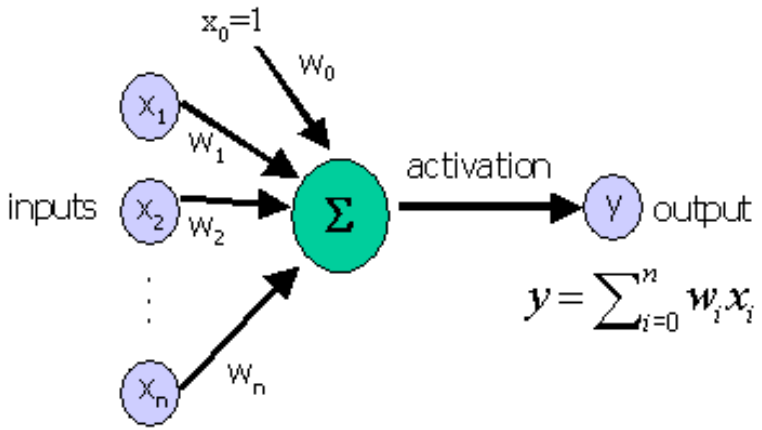
$$\langle w, x \rangle + b = 0$$



Linear Learning Machines (LLM)

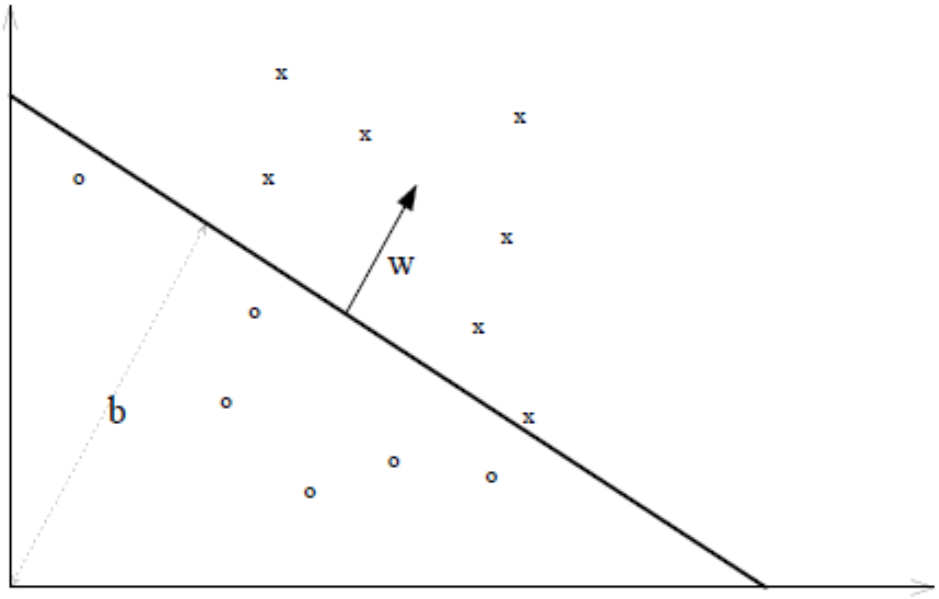
- El caso más simple es establecer un hiperplano como función de decisión en el espacio





- ¡Un ejemplo típico es el perceptrón!

Perceptron



- Produce una separación lineal del espacio de entrada

$$f(x) = \langle w, x \rangle + b$$

$$h(x) = sign(f(x))$$

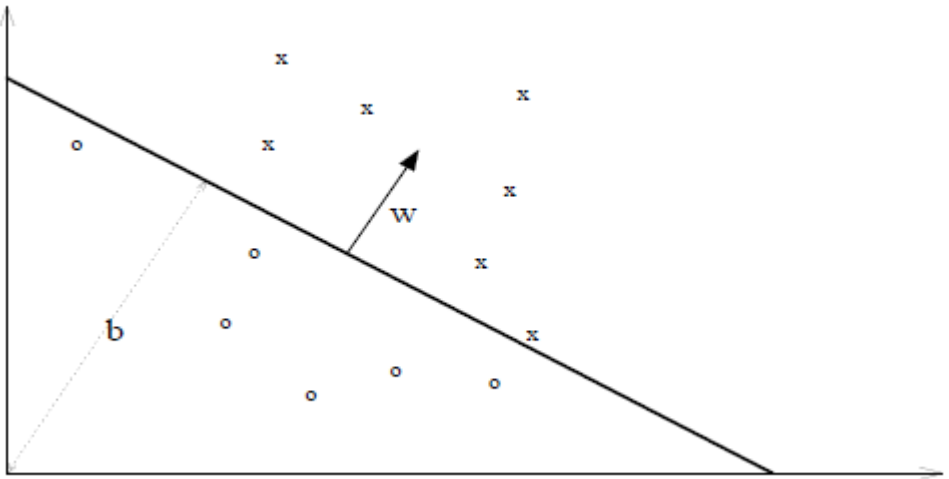
Algoritmo del perceptrón

- Regla de actualización (ignorando el umbral de activación):

if $y_i(\langle w_k, x_i \rangle) \leq 0$ then

$$w_{k+1} \leftarrow w_k + \eta y_i x_i$$

$$k \leftarrow k+1$$



- ¡El valor *b* no se actualiza!

Observaciones

- La solución es la combinación lineal de los puntos de entrenamiento de entrada

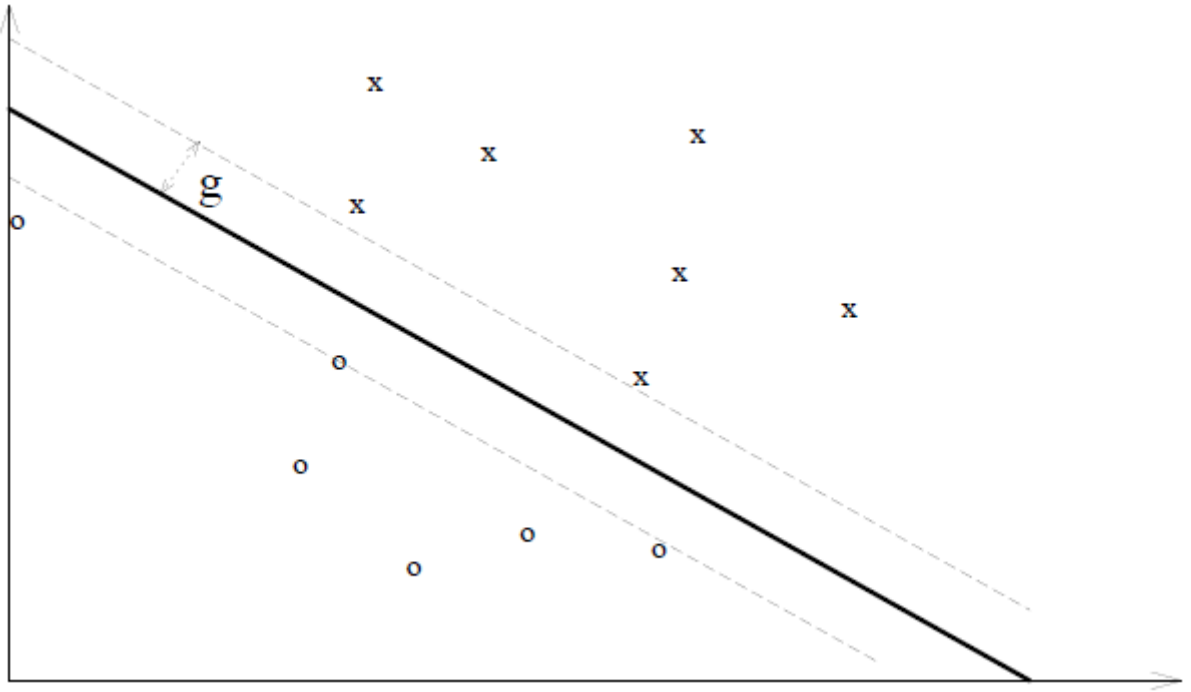
$$w = \sum \alpha_i y_i x_i$$

$$\alpha_i \geq 0$$

- Sólo se utilizan los ejemplos etiquetados
- El coeficiente de un punto en la combinación refleja su dificultad → más peso = más difícil

Observaciones

- Márgen de error:



- Los coeficientes son positivos
- Podemos reescribir el algoritmo para usar esta representación alternativa

Representación dual

- La función de decisión se puede reescribir como sigue:

$$f(x) = \langle w, x \rangle + b = \sum \alpha_i y_i \langle x_i, x \rangle + b$$

$$w = \sum \alpha_i y_i x_i$$

- Es muy importante porque nos permite definir la función de decisión a partir de las entradas

Representación dual

- La regla de actualización de pesos se puede reescribir también:

$$\text{if } y_i\left(\sum \alpha_j y_j \langle x_j, x_i \rangle + b\right) \leq 0 \quad \text{then } \alpha_i \leftarrow \alpha_i + \eta$$

Importante

En la representación dual los DATOS DE ENTRADA aparecen sólo en PRODUCTOS ESCALARES

Dualidad: primera propiedad de las SVM

- Las SVM son Linear Learning Machines (LLMs) representadas de forma dual

$$f(x) = \langle w, x \rangle + b = \sum \alpha_i y_i \langle x_i, x \rangle + b$$

- Los datos de entrada sólo actúan en productos escalares:
  - En la función de decisión
  - En el algoritmo de entrenamiento (actualización en transp. anteriores)

Limitaciones

- Las LLMs no pueden funcionar con:
  - Datos no separables linealmente
  - Datos con ruido
- Además, esta notación sólo funciona con datos representados de forma vectorial (atributo-valor)

¿Clasificadores no lineales?

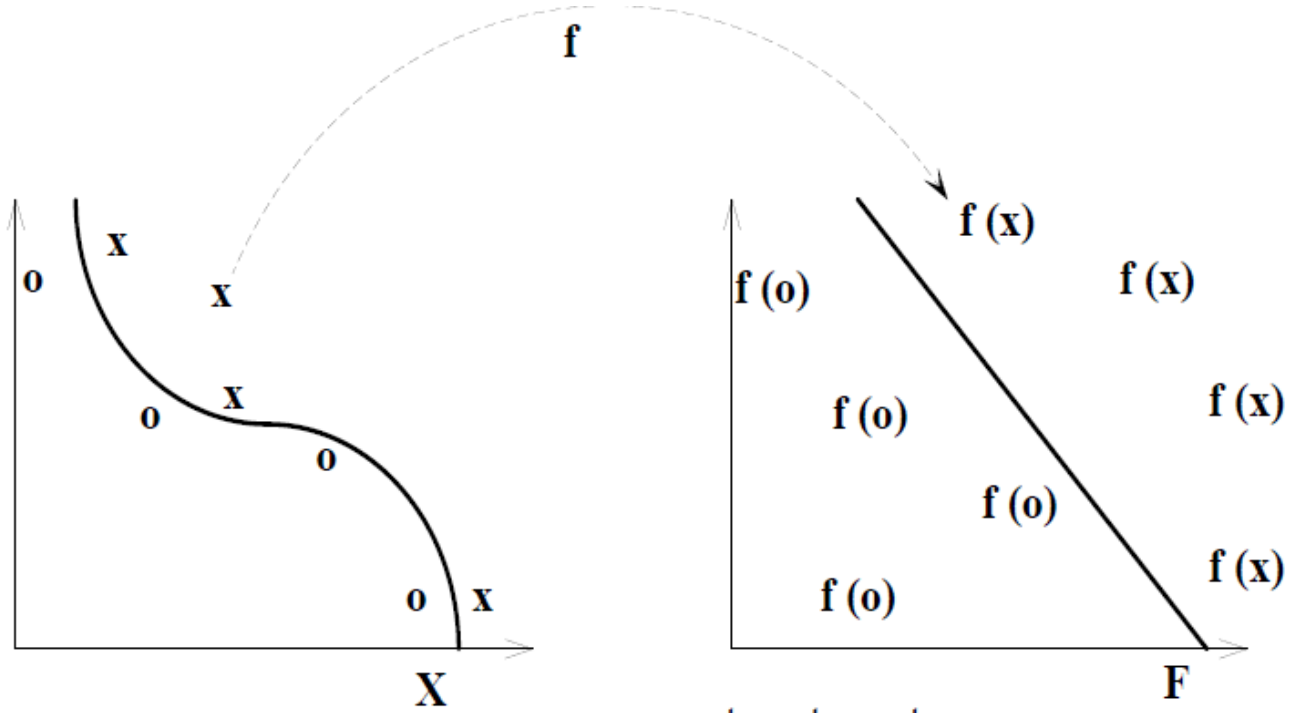
- **Solución 1**:
  - Crear una red de clasificadores lineales (neuronas) → RNA
  - Problemas: mínimos locales, alto número de parámetros, heurísticas necesarias para su entrenamiento, etc.

- **Solución 2:**
  - Asignar los datos en un espacio más complejo, con características no lineales, y usar un clasificador lineal en este espacio

Aprendizaje en un espacio más complejo

- **¡Idea!** → vamos a trasladar los datos a otro espacio de características donde sean linealmente separables

$$x \rightarrow \phi(x)$$



Problemas de la asignación

- ✓ Trabajar en un espacio de alta dimensionalidad nos evita el problema de expresar funciones no lineales complejas

PERO

- × Tienen un alto coste computacional → operaciones con vectores muy grandes
- × Es difícil obtener funciones de decisión generales por la *maldición de la dimensionalidad*

- Intentaremos dar solución a estos problemas en las transparencias siguientes

¿Cómo asignar los ejemplos a un espacio de mayor dimensionalidad?

Aquí entran en juego los Kernels:

- Solucionan el problema de la dimensionalidad!!
- Incluso pueden manejar espacios de dimensionalidad infinita
- Y más ventajas!

Espacios obtenidos mediante kernels

- Usando la **representación dual**, los ejemplos sólo aparecían en los productos escalares

$$f(x) = \sum \alpha_i y_i \langle \phi(x_i), \phi(x) \rangle + b$$

- La dimensionalidad del espacio F no es necesariamente importante. Ni siquiera necesitamos saber la función Φ

Kernels

¿Qué es un kernel?

- Una función que devuelve el valor del producto escalar entre **las imágenes** de los dos argumentos

$$K(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle$$

- **Lo más importante:**
  - No necesito la función Φ para dar la función K
  - Puedo verificar que una función K es un kernel sin Φ

Kernels

- Se pueden usar LLMs en un espacio de características de mayor dimensionalidad **simplemente sustituyendo los productos escalares por una función kernel**

$$\langle x_1, x_2 \rangle \leftarrow K(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle$$

La matriz kernel

- También se conoce como la matriz de Gram
- Define el resultado de la función kernel para cada par de vectores de entrada

$K=$

K(1,1)	K(1,2)	K(1,3)	…	K(1,m)
K(2,1)	K(2,2)	K(2,3)	…	K(2,m)
…	…	…	…	…
K(m,1)	K(m,2)	K(m,3)	…	K(m,m)

La matriz kernel

- Es la estructura principal en las KM
- Es el aspecto fundamental:
  - Contiene toda la información necesaria para el algoritmo de aprendizaje
- Fusiona la información de los datos de entrada y del kernel
- Muchas propiedades interesantes:

Teorema de Mercer

- La matriz kernel es una matriz definida positiva simétrica
- De igual forma, cualquier matriz definida positiva simétrica se puede considerar una matriz kernel → el producto escalar en un espacio cualquiera.
- Cualquier función simétrica (semi) positiva definida es un kernel, es decir, existe una función Φ tal que es posible escribir.
- ¿Qué es positiva definida?

Pos. Def. $\int K(x,z) f(x) f(z) dx dz \geq 0$

$\forall f \in L_2$

Ejemplos de kernels

- Ejemplos de kernels:

Polinomial $K(x,z) = \langle x,z \rangle^d$

Radial Basis $K(x,z) = e^{-\|x-z\|^2/2\sigma}$

Ejemplo: kernel polinomial

$$x = (x_1, x_2);$$

$$z = (z_1, z_2);$$

$$\langle x, z \rangle^2 = (x_1 z_1 + x_2 z_2)^2 =$$

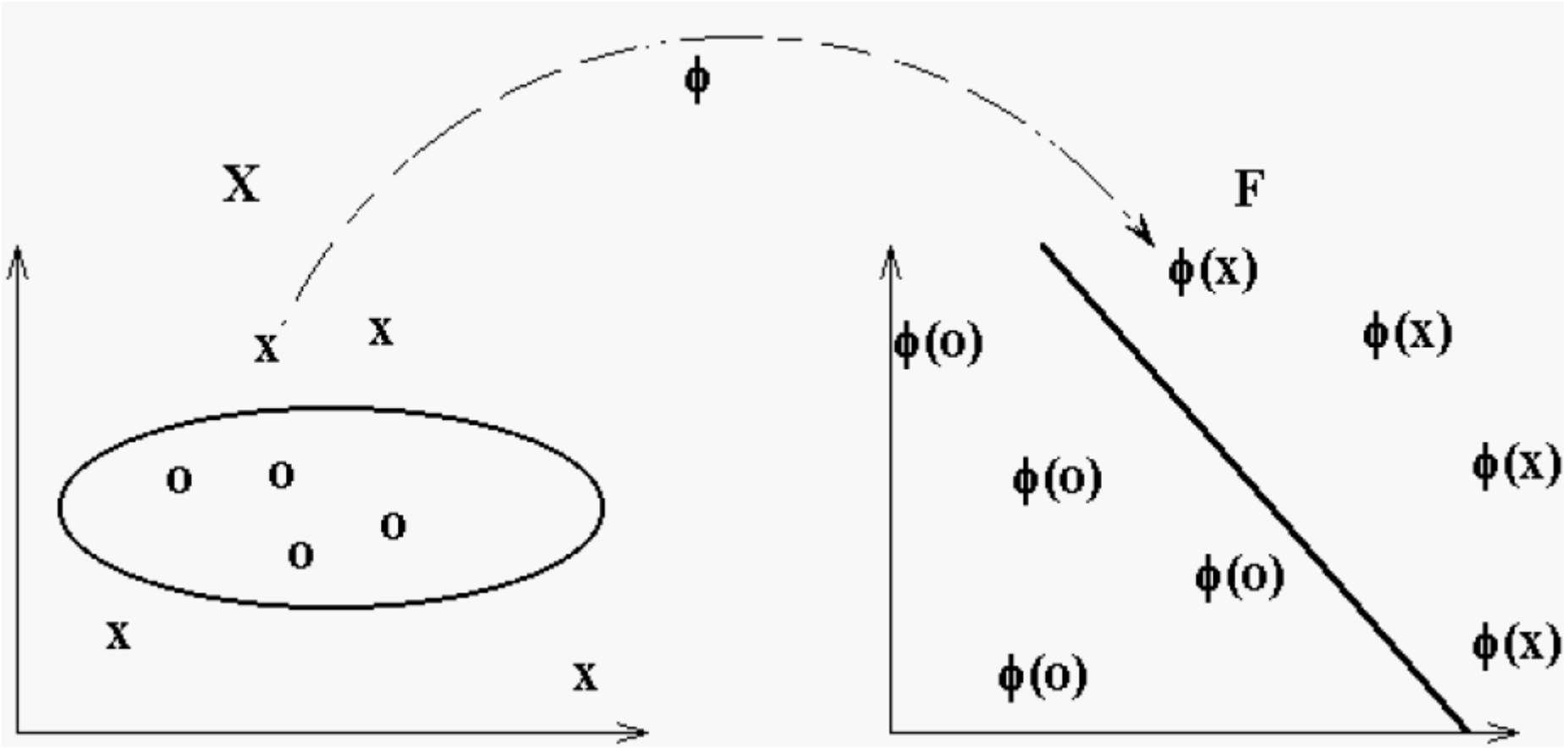
$$= x_1^2 z_1^2 + x_2^2 z_2^2 + 2 x_1 z_1 x_2 z_2 =$$

$$= \left\langle (x_1^2, x_2^2, \sqrt{2} x_1 x_2), (z_1^2, z_2^2, \sqrt{2} z_1 z_2) \right\rangle =$$

$$= \langle \phi(x), \phi(z) \rangle$$

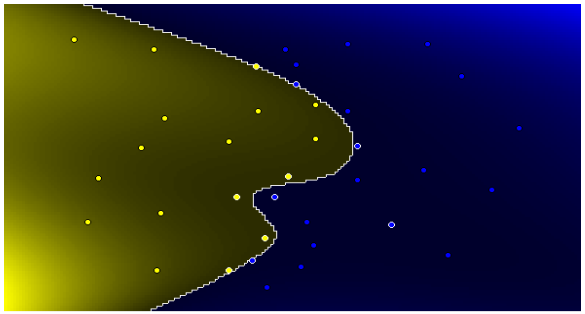
Usando el binomio de Newton se puede observar que el espacio al que se 'mapea' la entrada tiene 3 dimensiones frente a las 2 originales

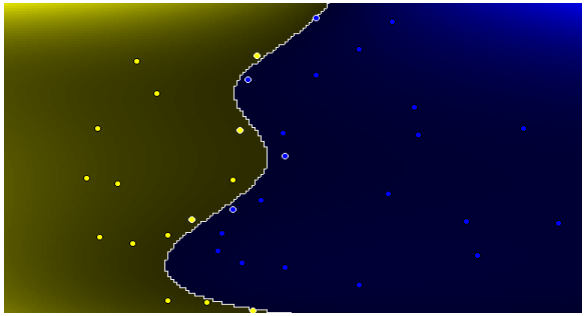
Ejemplo: kernel polinomial

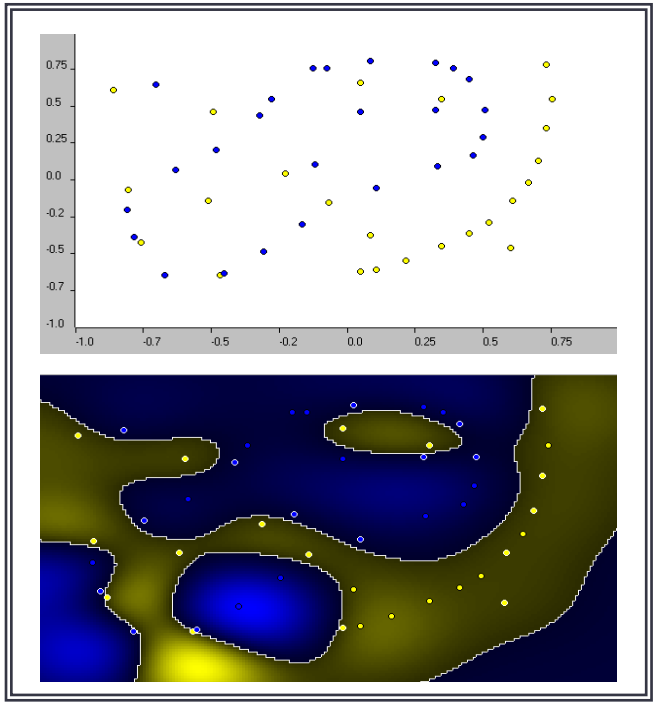


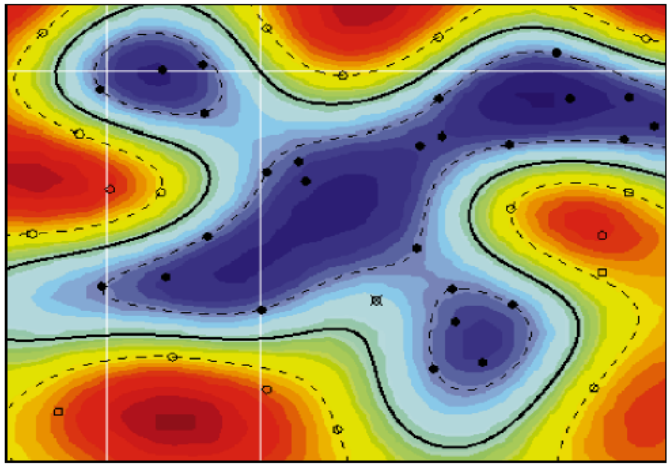
Máquinas de soporte vectorial (SVM)

Separación polinómica y con RBF









Construyendo kernels

- El conjunto de kernels es cerrado bajo ciertas operaciones. Si K y K' son kernels entonces:
  - K+K' es kernel
  - cK es kernel si c > 0
  - aK+bK' es kernel para a,b > 0
  - Etc.

- Se pueden construir funciones kernels complejas a partir de módulos más sencillos

Segunda propiedad de las SVM

- Las SVMs son LLMs que:
  - Usan representación DUAL

Y

  - Operan un espacio de mayor dimensionalidad inducido por un kernel, esto es, f(x) es una función de decisión lineal en el espacio definido implícitamente por K

$$f(x) = \sum \alpha_i y_i \langle \phi(x_i), \phi(x) \rangle + b$$

Un mal kernel…

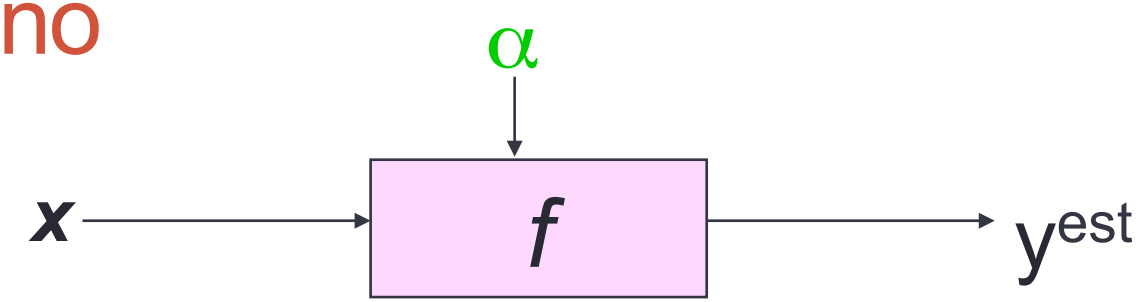
- … es aquél cuya matriz kernel es mayormente diagonal: todos los puntos son ortogonales entre sí, no hay información implícita en los datos

1	0	0	…	0
0	1	0	…	0
		1		
…	…	…	…	…
0	0	0	…	1

El algoritmo de aprendizaje genérico: El problema de la generalización

- Maldición de la dimensionalidad: es fácil sobreajustar en espacios de alta dimensionalidad
  - Mayor probabilidad de encontrar estructuras de información en el conjunto de entrenamiento que son accidentales → no se encuentran en el test!
- El problema definido por las SVMs está mal definido: existen muchos hiperplanos que separan los datos
- Necesitamos alguna manera de encontrar el mejor hiperplano

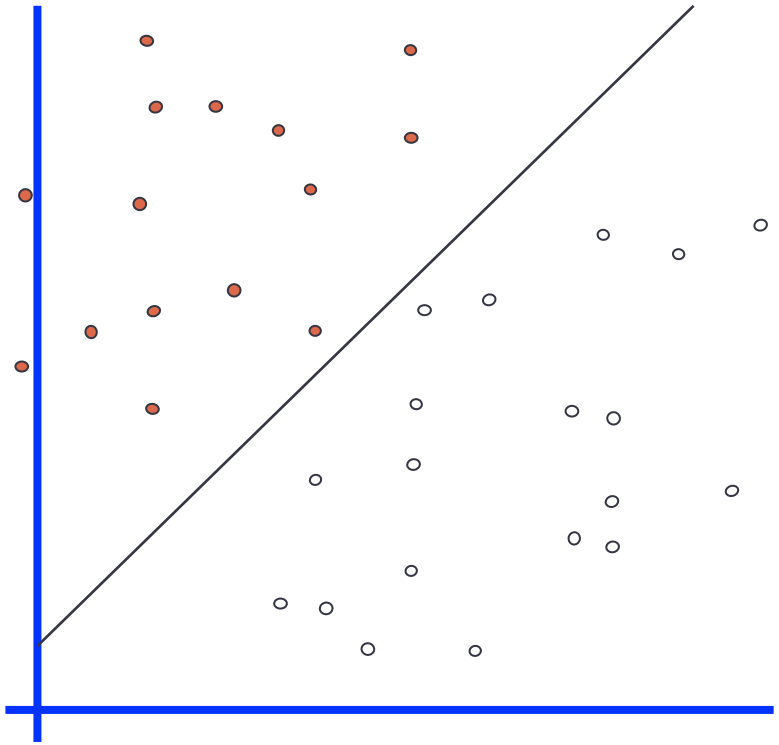
Elegir el hiperplano



$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$

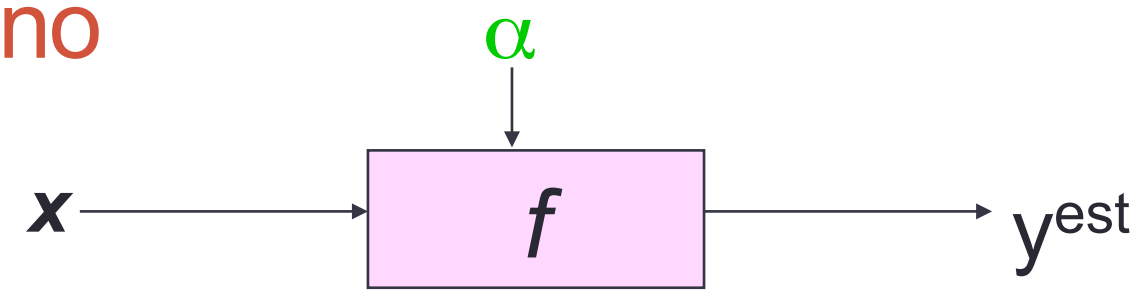
denotes +1

denotes -1



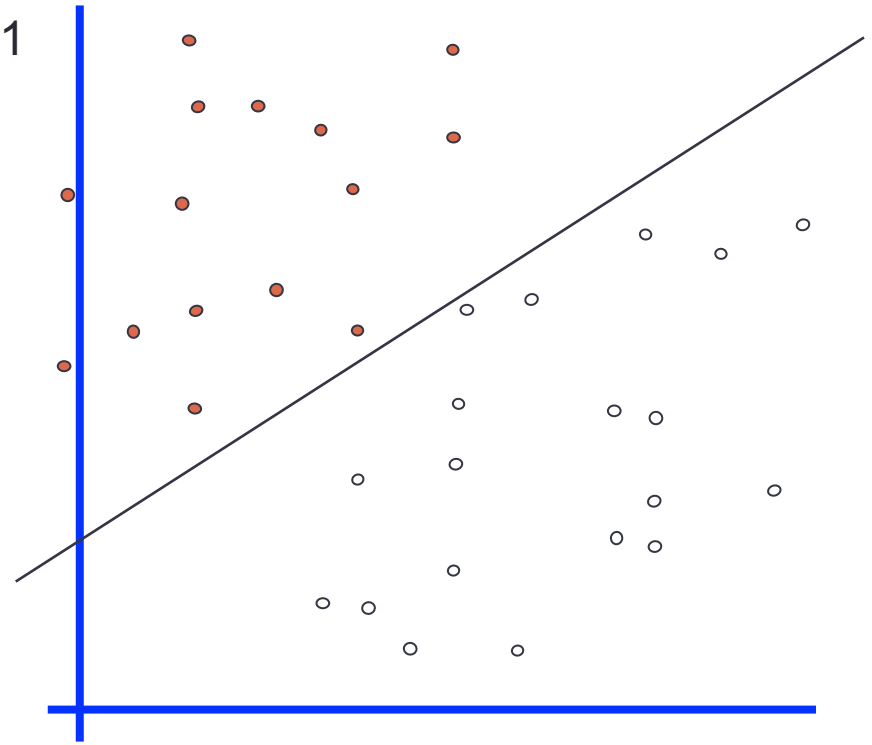
How would you classify this data?

Elegir el hiperplano



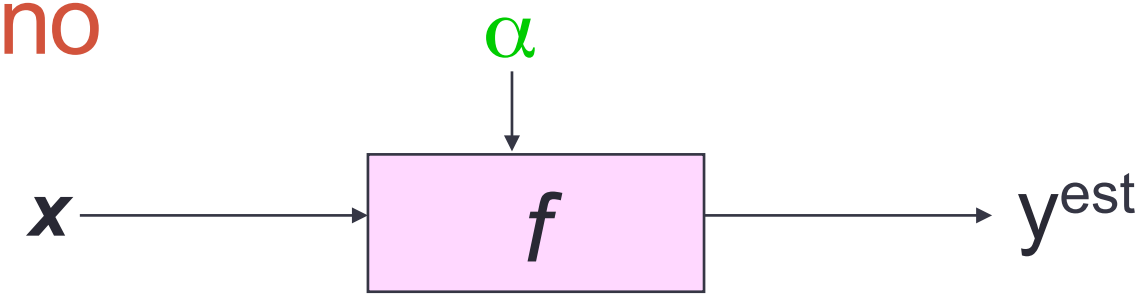
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$

- denotes +1
- denotes -1

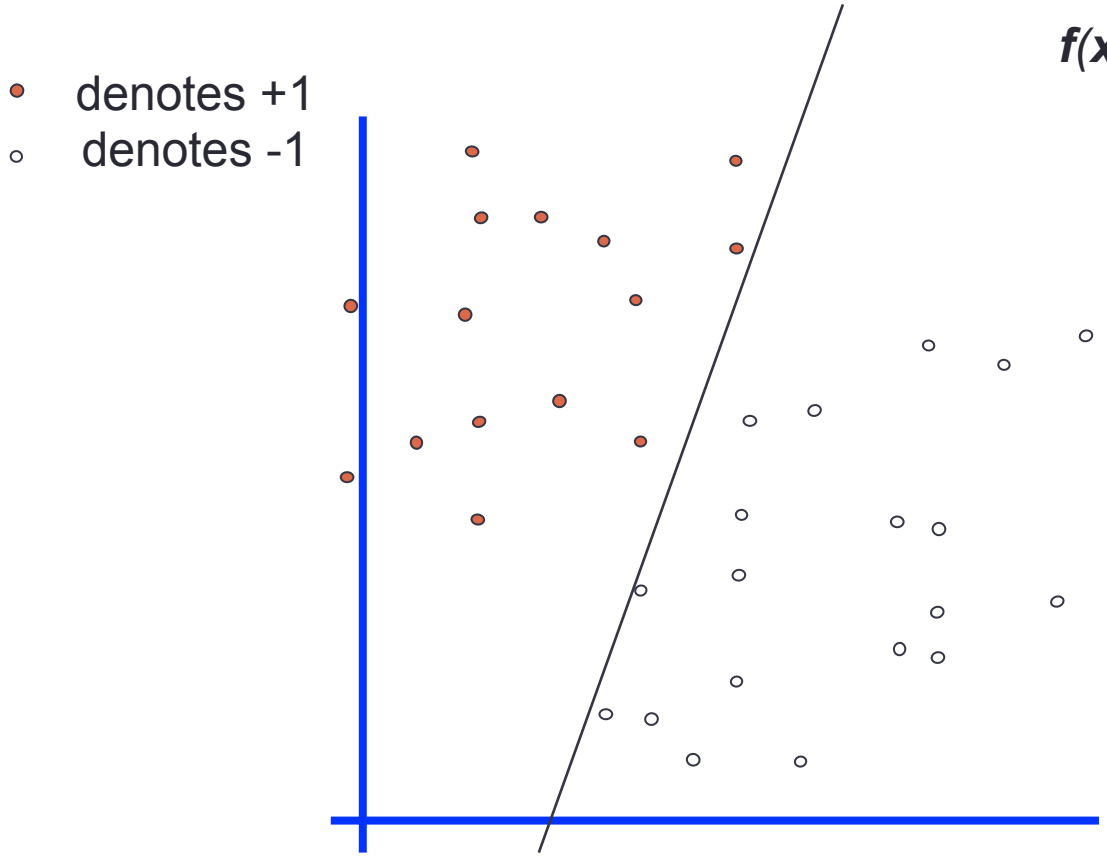


How would you classify this data?

Elegir el hiperplano

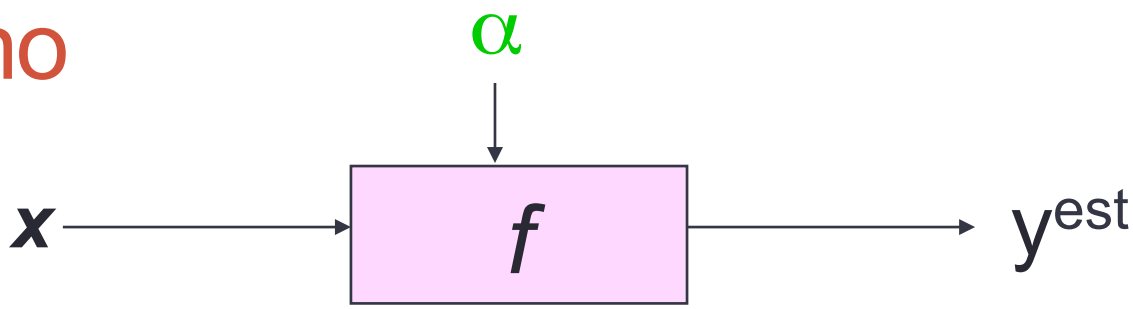


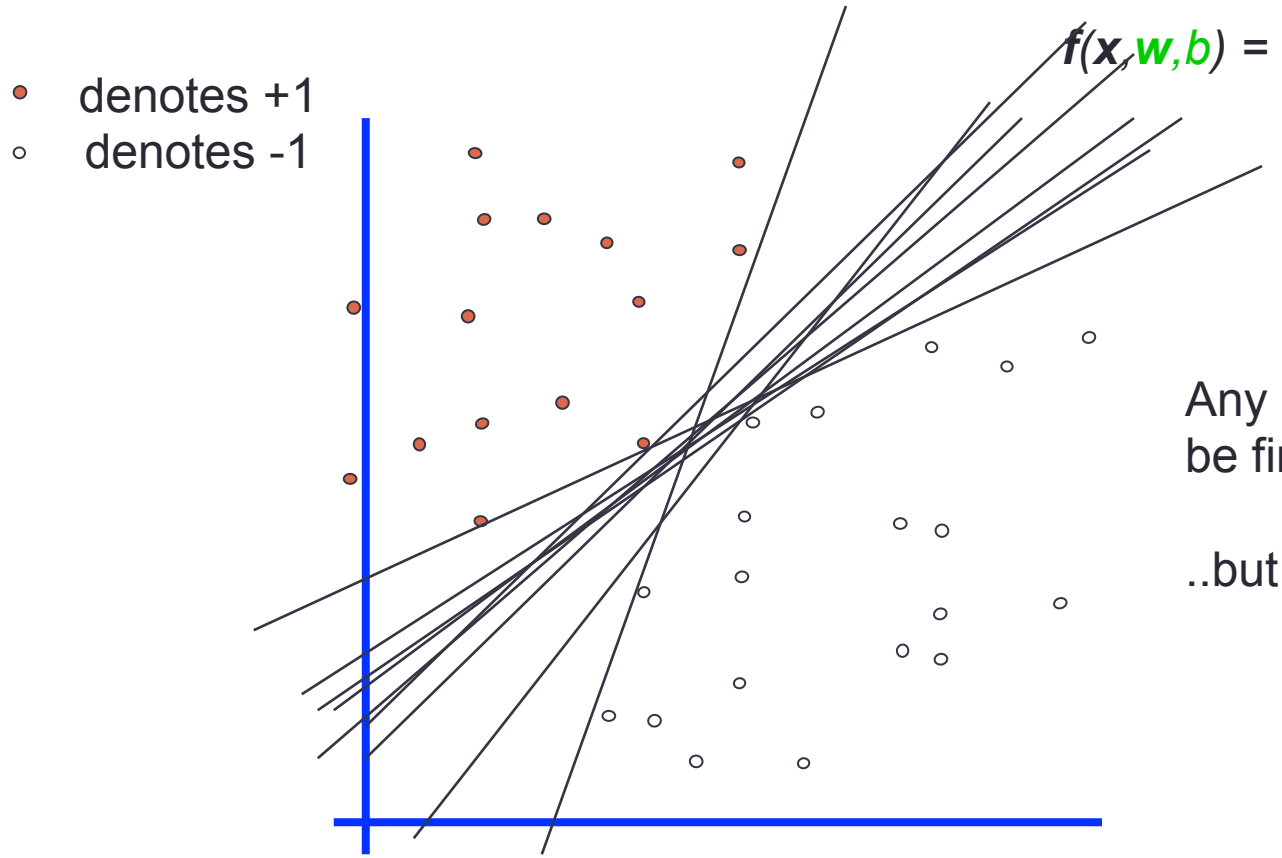
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$



How would you classify this data?

Elegir el hiperplano





Any of these would be fine..

..but which is best?

El algoritmo de aprendizaje genérico: El problema de la generalización

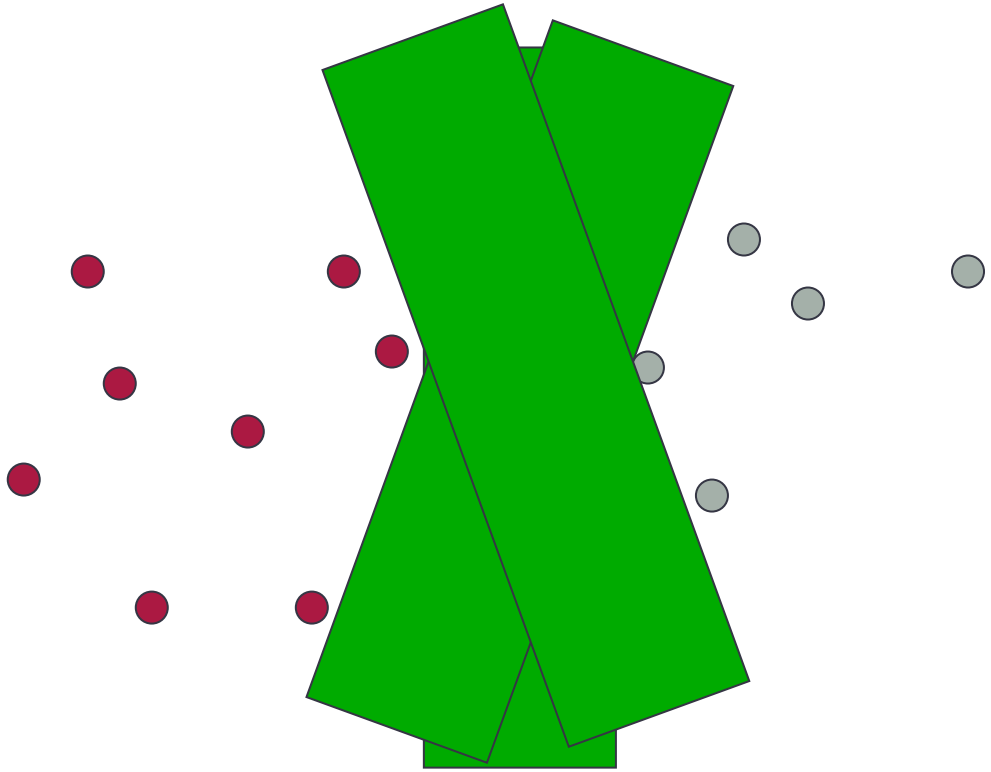
- Hay muchos métodos buenos para encontrar un buen hiperplano
  - Bayes, aprendizaje estadístico, MDL, etc.

- Se puede usar cualquiera, pero en SVMs se suele usar el caso motivado por la teoría de aprendizaje estadístico.

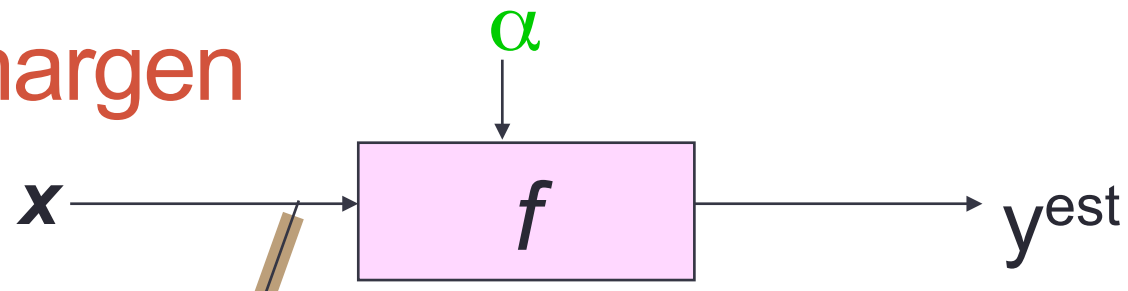
- **Vapnik–Chervonenkis (VC) learning theory**

Una intuición:

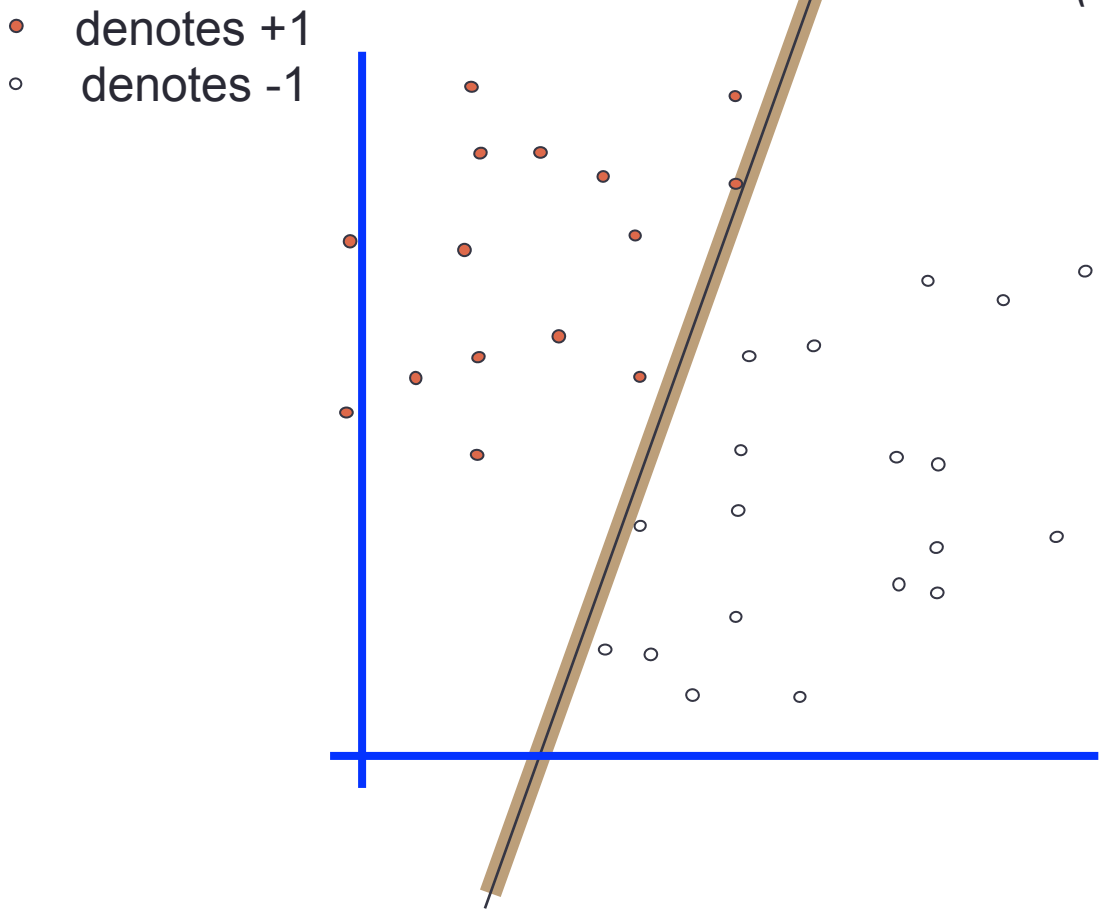
- Si se coloca un separador “gordo” entre las clases, existen menos opciones, y así la capacidad del modelo decrece



Clasificador de margen máximo

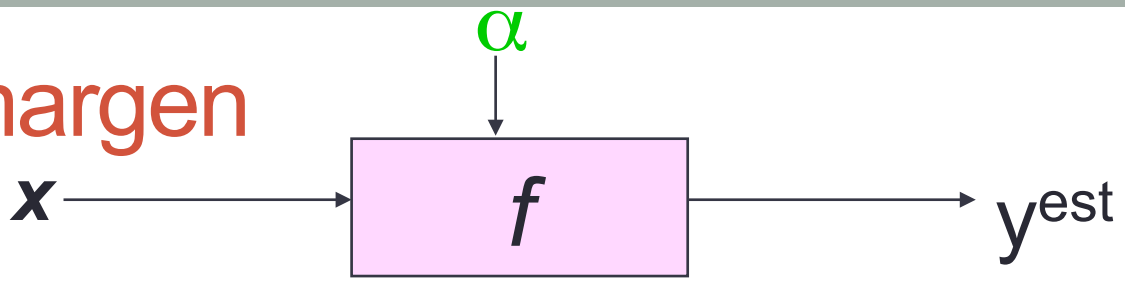


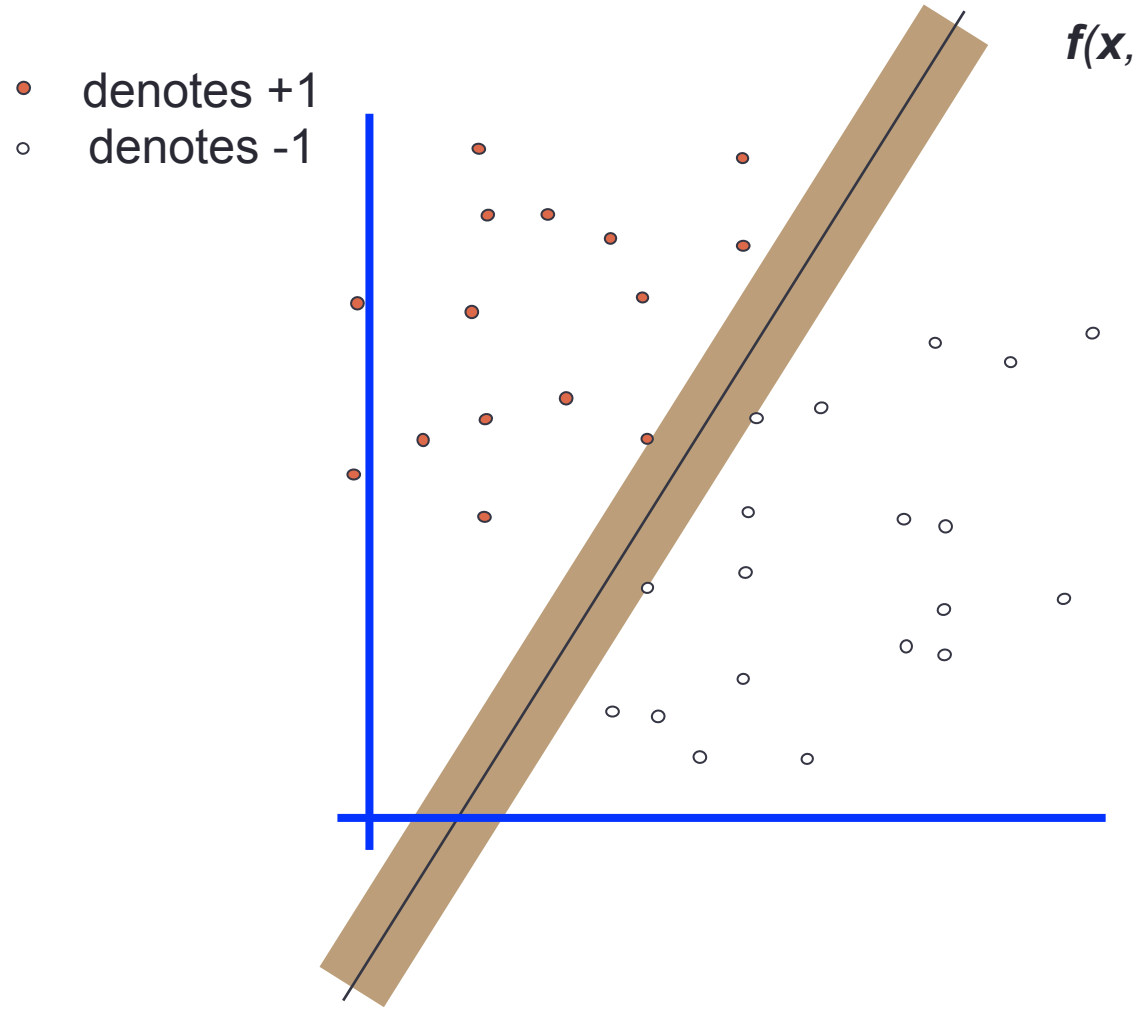
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$



Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

Clasificador de margen máximo



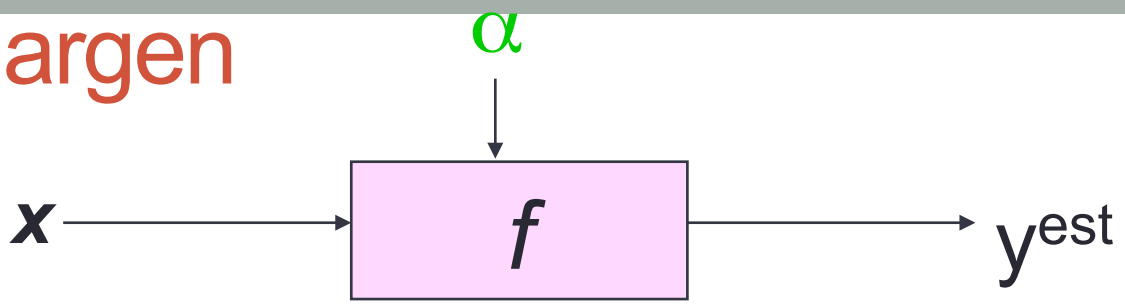


$f(x,w,b) = sign(w.\ x - b)$

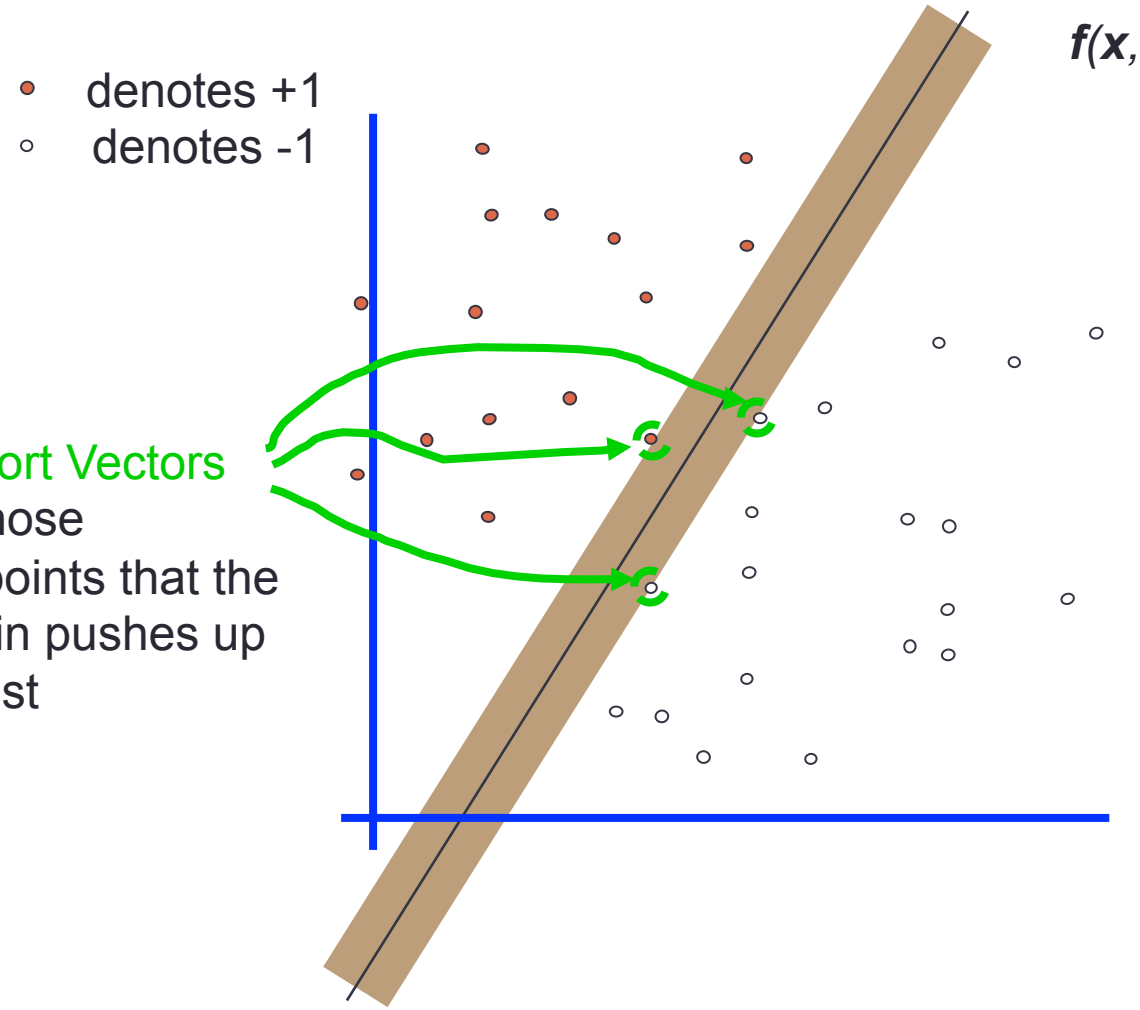
The maximum margin linear classifier is the linear classifier with the maximum margin.
This is the simplest kind of SVM (Called an LSVM)

Linear SVM

Clasificador de margen máximo



$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\mathbf{x} - b)$



Support Vectors are those datapoints that the margin pushes up against

The maximum margin linear classifier is the linear classifier with the maximum margin.
This is the simplest kind of SVM (Called an LSVM)

Linear SVM

Clasificador de margen máximo

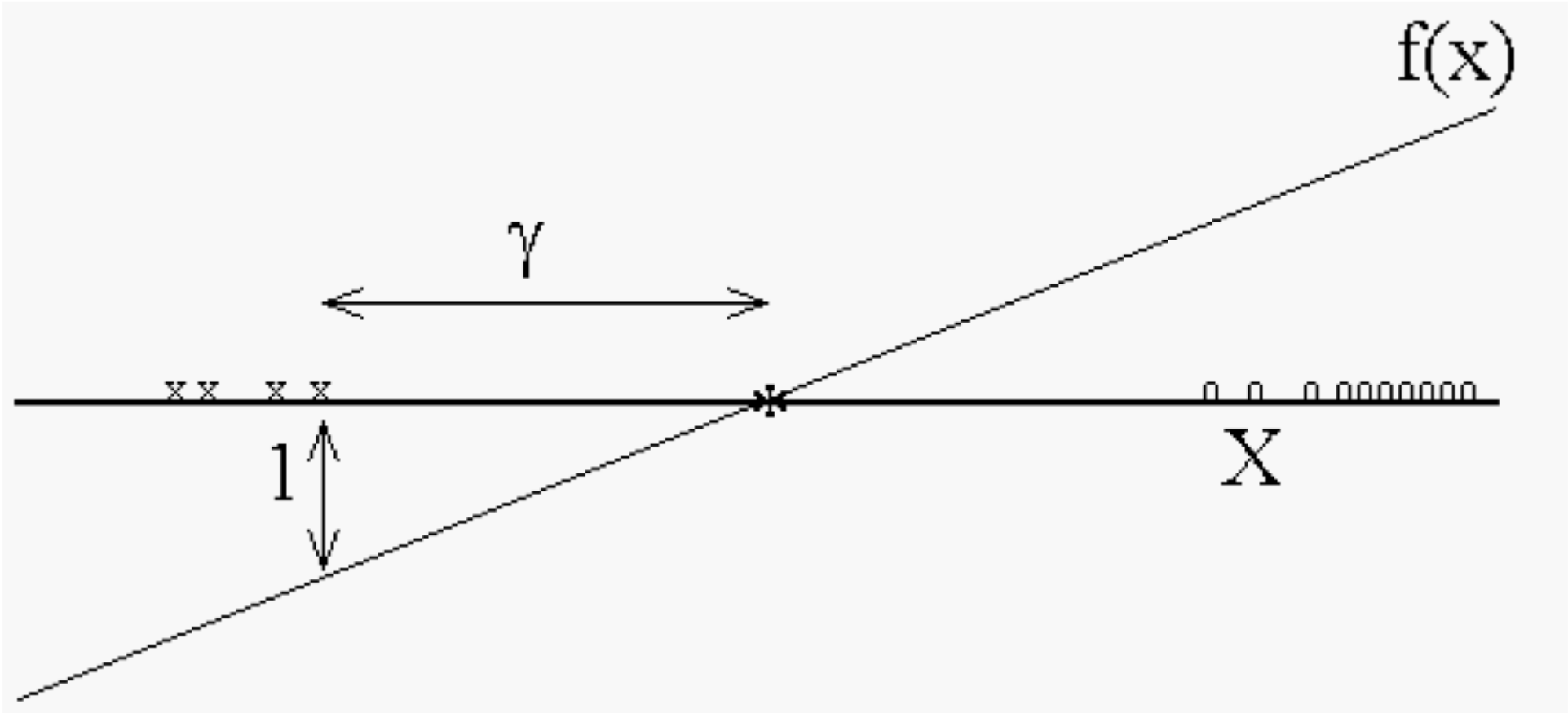
- Se minimiza el riesgo de sobreajuste eligiendo el hiperplano de máximo margen en el espacio de alta dimensionalidad

- **3ª característica de las SVM**: maximizan el margen

- Las SVMs tienen la capacidad de controlar el margen para adaptarse, en vez de los grados de libertad del espacio (no dependen de la dimensionalidad).

Dos tipos de margen

- **Funcional** → f(x)* signo = |f(x)|(valor absoluto). Distancia paralela al eje
- **Geométrica** → |f(x)| ponderado por la norma (distancia al plano)



Margen máximo = norma mínima

- Si fijamos el margen funcional a 1, el margen geométrico es igual a 1/||w||

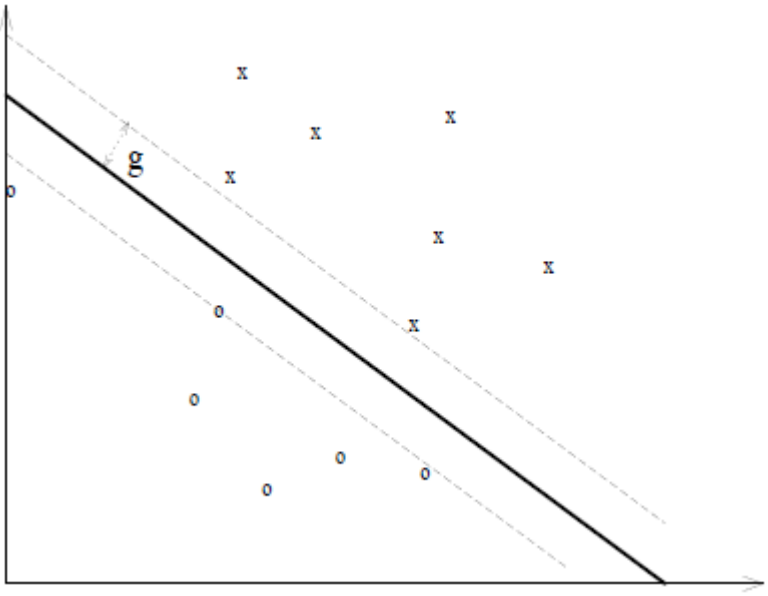
- El margen funcional no lo controlamos:
  - Maximizamos el margen minimizando la norma
  - La cantidad del margen funcional no nos importa. Nos interesa el signo

Margen máximo = norma mínima

- Distancia entre los límites convexos
  - x^+,x^- son ejemplos en el límite del margen (vectores soporte)

$$\langle w, x^+ \rangle + b = +1$$

$$\langle w, x^- \rangle + b = -1$$



•Se desea que el resultado para los vectores soporte sea +1 y -1 según a la clase que pertenezcan.
•Otros ejemplos más distantes del hiperplano *w* que los vectores soporte tendran un

|<w,x>+b| > 1

Margen máximo = norma mínima

Sabemos que

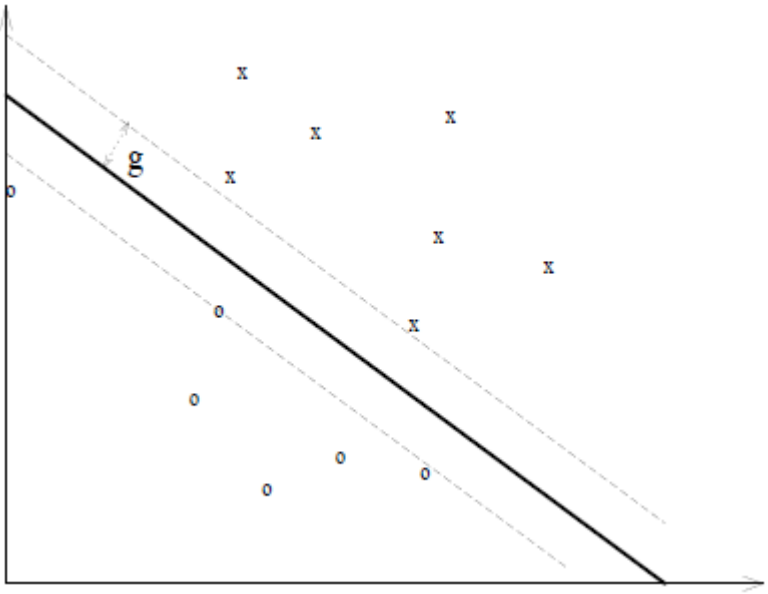
<w,x>+b= 0

es equivalente a

c(<w,x>+b) = 0

$$\langle w, x^{+} \rangle + b = +1$$

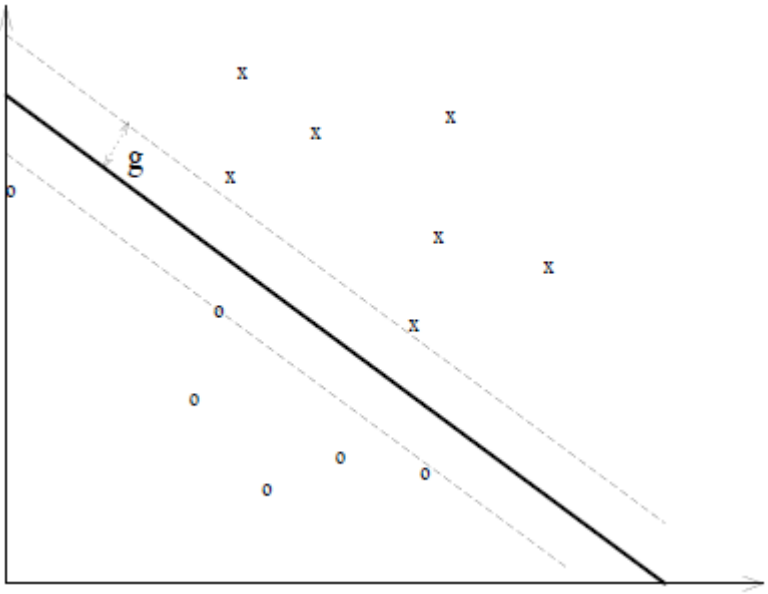
$$\langle w, x^{-} \rangle + b = -1$$



•Entonces debemos elegir una **normalización tal que se verifique el resultado de +1 y -1 para los vectores soporte**

Margen máximo = norma mínima

- La normalización que permite este resultado está dada por ||w||



$$\langle w, x^+ \rangle + b = +1$$

$$\langle w, x^- \rangle + b = -1$$

$$\langle w, (x^+ - x^-) \rangle = 2$$

$$\left\langle \frac{w}{\|w\|}, (x^+ - x^-) \right\rangle = \frac{2}{\|w\|}$$

Problema primordial

- El margen es 2/||w||
- Entonces → minimizar la norma = maximizar margen:

$$\langle w, w \rangle$$

- Sujeto a:

$$y_i(\langle w, x_i \rangle + b) \geq 1$$

Siempre es mayor que uno porque al multiplicar por y_i, negativo por negativo da positivo (si el ejemplo era de clase negativa el producto por el plano daba negativo, e y_i es negativo porque es la etiqueta!) ← **mantiene la condición de plano de decision**

Teoría de optimización

- El problema de encontrar el hiperplano de margen máximo es un problema de optimización condicionado (**programación cuadrática** o *QP*)
- Se suele usar la teoría de Lagrange (o teoría de Kuhn-Tucker)
- Lagrange:

$$\frac{1}{2}\langle w, w\rangle - \sum \alpha_i y_i \left[\left(\langle w, x_i\rangle + b\right) - 1\right]$$

$$\alpha \geq 0$$

  - Se trata de hallar unos *w* y *b* de manera que los α calculados para cada ejemplo permitan que ese valor de <w,w>=||w|| sea mínimo

Quadratic Programming

Find $\arg\min_{\mathbf{u}} \; c + \mathbf{d}^T\mathbf{u} + \frac{\mathbf{u}^T R\mathbf{u}}{2}$ ← Quadratic criterion

Subject to

$$\begin{aligned} a_{11}u_1 + a_{12}u_2 + \ldots + a_{1m}u_m &\le b_1 \\ a_{21}u_1 + a_{22}u_2 + \ldots + a_{2m}u_m &\le b_2 \\ &\vdots \\ a_{n1}u_1 + a_{n2}u_2 + \ldots + a_{nm}u_m &\le b_n \end{aligned}$$

n additional linear inequality constraints

And subject to

$$\begin{aligned} a_{(n+1)1}u_1 + a_{(n+1)2}u_2 + \ldots + a_{(n+1)m}u_m &= b_{(n+1)} \\ a_{(n+2)1}u_1 + a_{(n+2)2}u_2 + \ldots + a_{(n+2)m}u_m &= b_{(n+2)} \\ &\vdots \\ a_{(n+e)1}u_1 + a_{(n+e)2}u_2 + \ldots + a_{(n+e)m}u_m &= b_{(n+e)} \end{aligned}$$

e additional linear equality constraints

Quadratic Programming

Find $\arg\min_{\mathbf{u}} \; c + \mathbf{d}^T\mathbf{u} + \frac{\mathbf{u}^T R\mathbf{u}}{2}$

Quadratic criterion

Subject to $a_{11}u_1$ ... a_1

...ditional linear
...equality
...straints

And subject

Existen algoritmos para encontrar el óptimo de un problema cuadrático con restricciones mucho más eficientes y fiables que el gradiente ascendente.
(Pero son muy complejos y probablemente no quieres escribir uno por tí mismo…)

...+1)

a_m $b_{(n+2)}$

$a_{(n+e)1}u_1 + a_{(n+e)2}u_2 + \ldots + a_{(n+e)m}u_m = b_{(n+e)}$

e additional linear
equality
constraints

El problema dual

- Maximizar

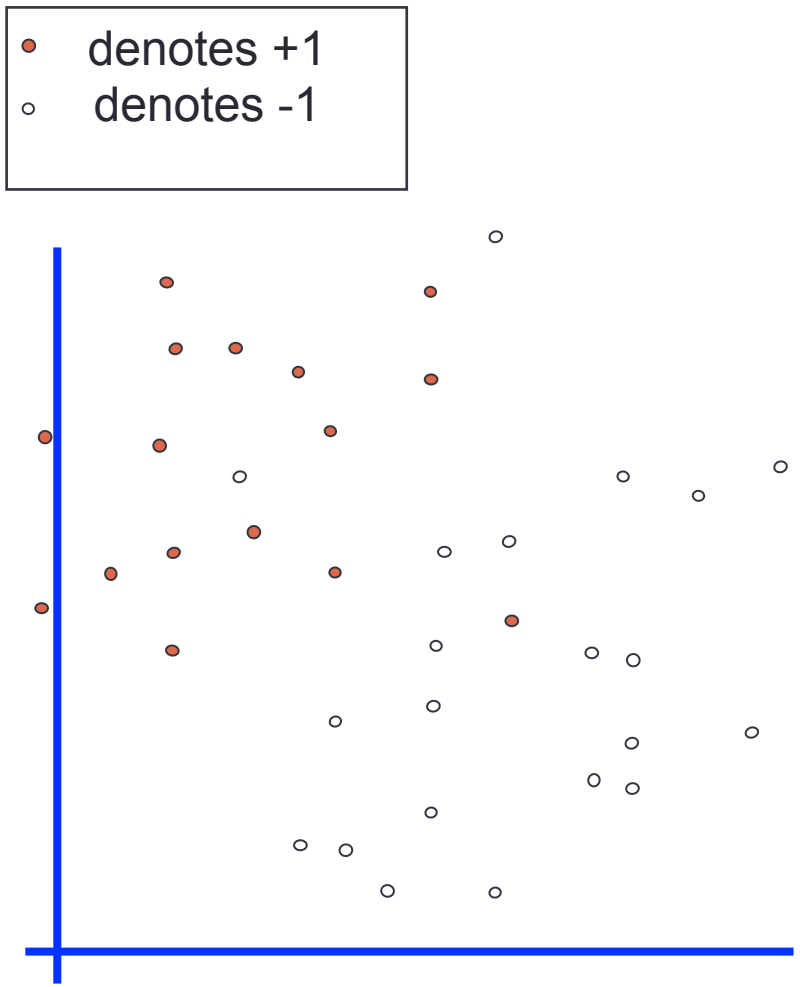
$$W(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$$

- Sujeto a: $\alpha_i \geq 0$

$$\sum_i \alpha_i y_i = 0$$

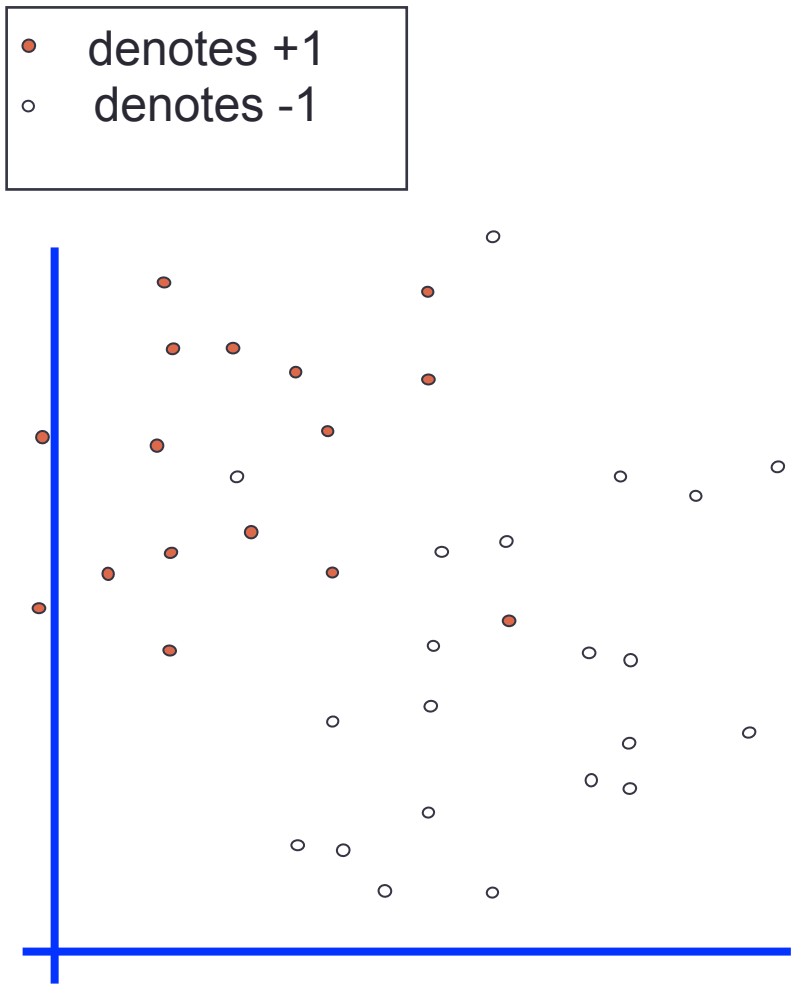
- Esta formulación no utiliza ***w ni b*** → a partir de los *vectores soporte* se obtiene el plano
- ¡¡Esta dualidad nos permite usar kernels!!

Noise!! Uh-oh!



This is going to be a problem!
What should we do?

Uh-oh!



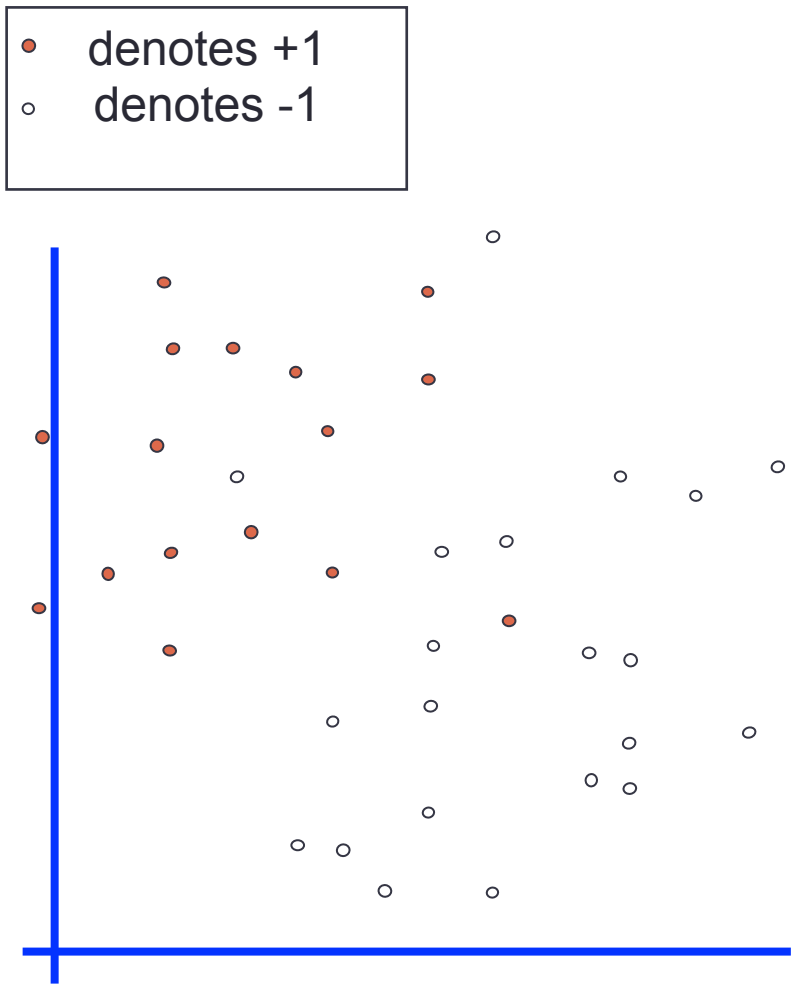
This is going to be a problem!
What should we do?

Idea 1:

Find minimum $||\boldsymbol{w}||$, while minimizing number of training set errors.

Problemette: Two things to minimize makes for an ill-defined optimization

Uh-oh!



This is going to be a problem! What should we do?

Idea 1.1:

Minimize

w.w + *C (#train errors)*

Tradeoff parameter

There's a serious practical problem that's about to make us reject this approach. Can you guess what it is?

Uh-oh!

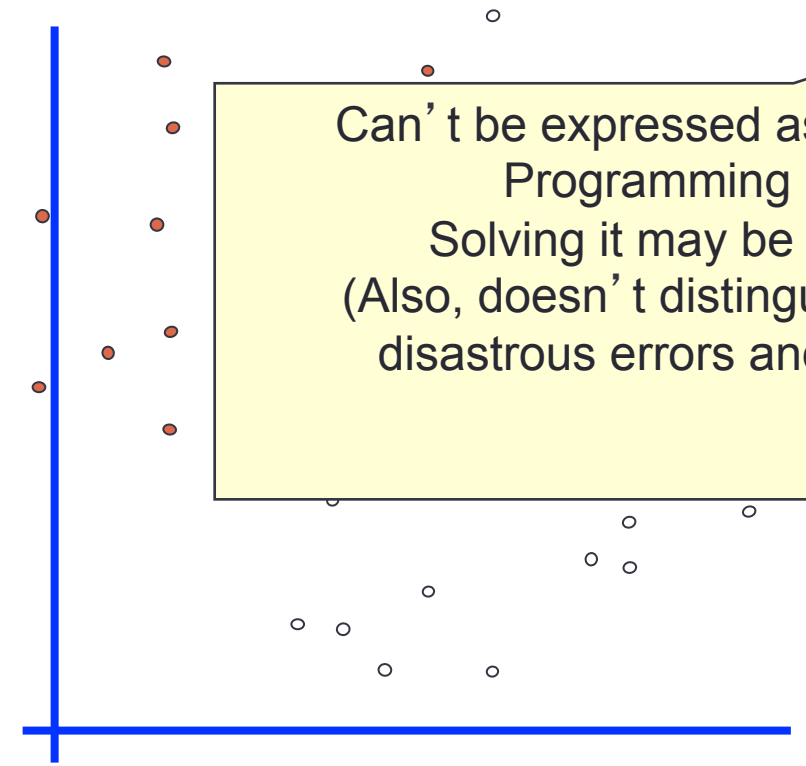
This is going to be a problem!
What should we do?

Idea 1.1:

Minimize

w.w + *C (#train errors)*

Tradeoff parameter

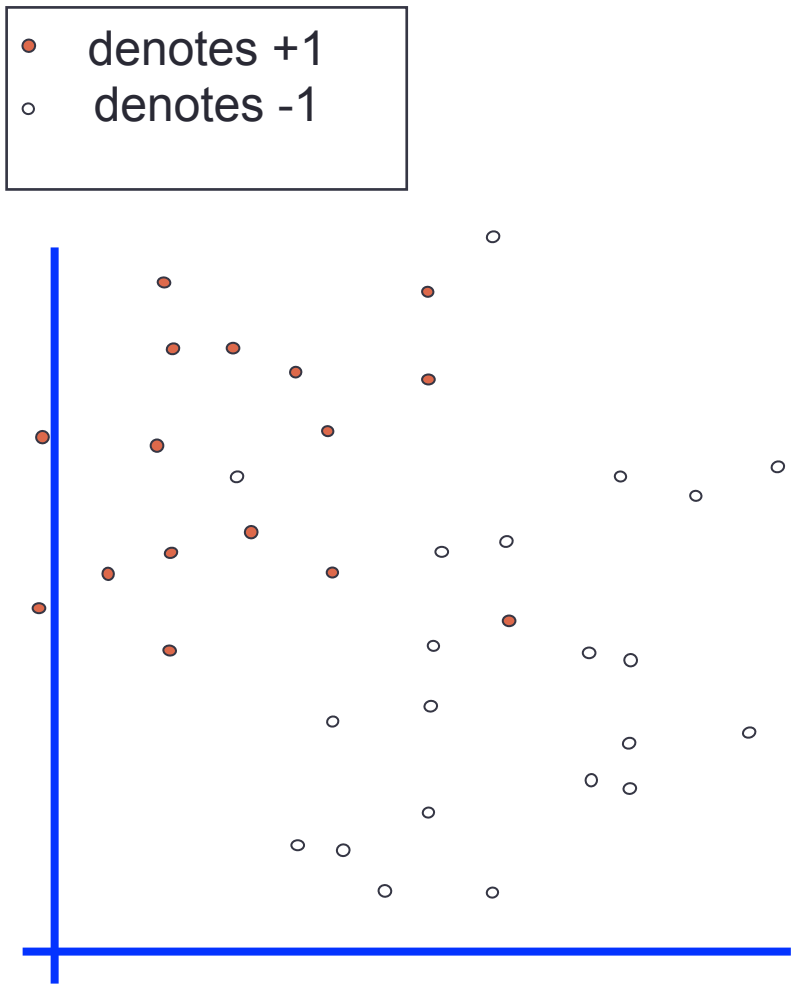


Can't be expressed as a Quadratic Programming problem.
Solving it may be too slow.
(Also, doesn't distinguish between disastrous errors and near misses)

serious practical
at's about to make
his approach. Can
what

So… any other ideas?

Uh-oh!



This is going to be a problem!
What should we do?

Idea 2.0:

Minimize

w.w + *C (distance of error points to their correct place)*

Support Vector Machine (SVM) for Noisy Data

•Ahora mimimizamos la norma ||w|| penalizada por los valores ε de los ejemplos erróneos

$$\{\vec{w}^*, b^*\} = \min_{\vec{w},b,\vec{\varepsilon}} \sum_{i=1}^{d} w_i^2 + c\sum_{j=1}^{N} \varepsilon_j$$

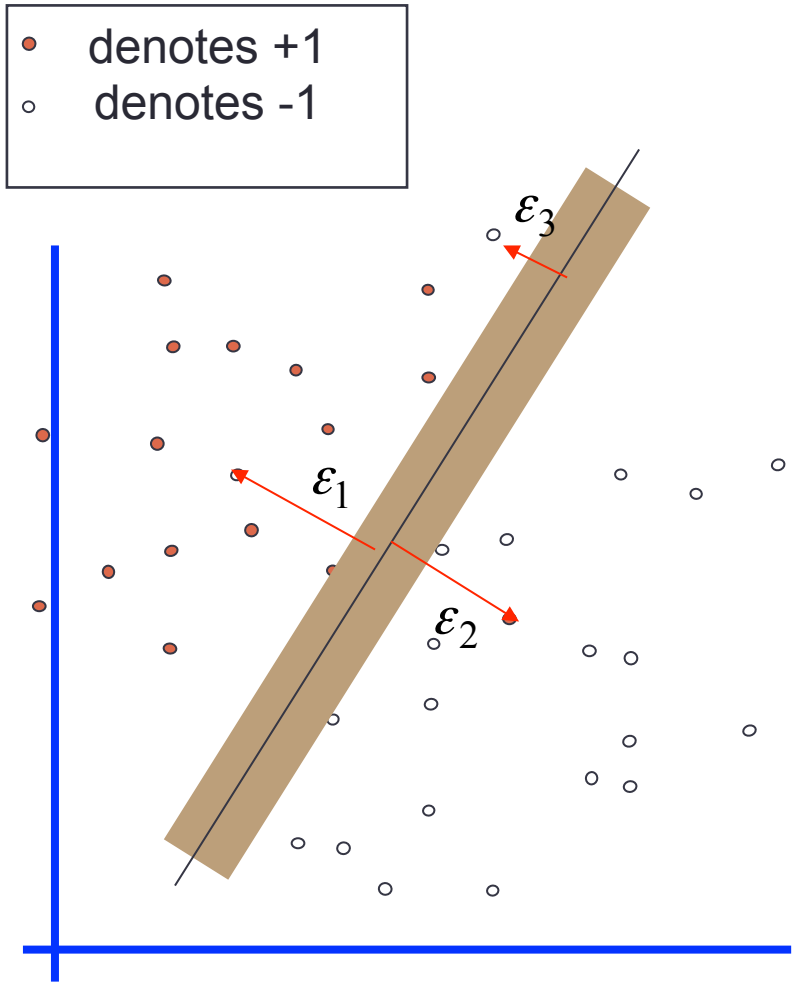
$$y_1\left(\vec{w}\cdot\vec{x}_1 + b\right) \geq 1 - \varepsilon_1$$

$$y_2\left(\vec{w}\cdot\vec{x}_2 + b\right) \geq 1 - \varepsilon_2$$

...

$$y_N\left(\vec{w}\cdot\vec{x}_N + b\right) \geq 1 - \varepsilon_N$$

- Any problem with the above formulism?
  - Balance the trade off between margin and classification errors ← **PARAMETER C**



El clasificador con margen *suave*

- Minimizar $$\frac{1}{2}\langle w, w\rangle + C\sum_i \xi_i$$
- O: $$\frac{1}{2}\langle w, w\rangle + C\sum_i \xi_i^2$$
- Sujeto a: $$y_i(\langle w, x_i\rangle + b) \geq 1 - \xi_i$$

Representación dual de Lagrange para el clasificador de margen suave

- Mediante

$$W(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$$

Restricciones de caja:

$$0 \leq \alpha_i \leq C$$

$$\sum_i \alpha_i y_i = 0$$

- Mediante

$$\sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle - \frac{1}{2C} \sum \alpha_j \alpha_j$$

Diagonal:

$$0 \leq \alpha_i$$

$$\sum_i \alpha_i y_i \geq 0$$

¿Cómo se implementa esto?

- Maximizar una función cuadrática sujeta a restricciones de igualdad lineales (así como desigualdades)

$$W(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

$$\alpha_i \geq 0$$

$$\sum_i \alpha_i y_i = 0$$

Primera aproximación

- Inicialmente se usaban paquetes de QP complejos:
  - Gradiente ascendente estocástico (actualiza un peso en cada paso). Da buenos resultados

$$\alpha_i \leftarrow \alpha_i + \frac{1}{K(x_i, x_i)}\left(1 - y_i \sum \alpha_i y_i K(x_i, x_j)\right)$$

- Se descompone la matriz kernel obviando filas y columnas con ceros
- ¿Por qué un peso en cada paso?
  - Si el problema es grande, la matriz kernel es inmensa y no cabe en memoria

Solución completa: SMO

- SMO actualiza **DOS** pesos en cada paso
- Puede hacer gradiente descendente satisfaciendo las restricciones lineales (J. Platt)
- LibSVM emplea el método de SMO con una heurística de selección de los dos pesos/parámetros de Lagrange a optimizar en cada paso

Solución completa: SMO

- SMO resuelve el problema más pequeño de optimización en cada caso:
  - El problema mínimo implica dos multiplicadores de Lagrange α
- ¡La ventaja de este tamaño de problema dos es que se puede resolver analíticamente!
  - Aunque tenga que hacer más pasos de optimización, cada paso es muy rápido de resolver →más rápido que el QP de paso simple

Solución completa: LibSVM

- SMO escoge los dos multiplicadores α de manera desinformada
- LibSVM propone escoger dos multiplicadores de forma más inteligente para mejorar la convergencia del algoritmo
  - Utiliza información de segundo orden
- Este algoritmo es **C-SVM**

- LibSVM incluye también **v-SVM**
  - El parámetro **v** actúa de forma inversamente proporcional al parámetro C.
  - Mayor **v** → menor C y viceversa
  - Formulación originaria para **regresión**

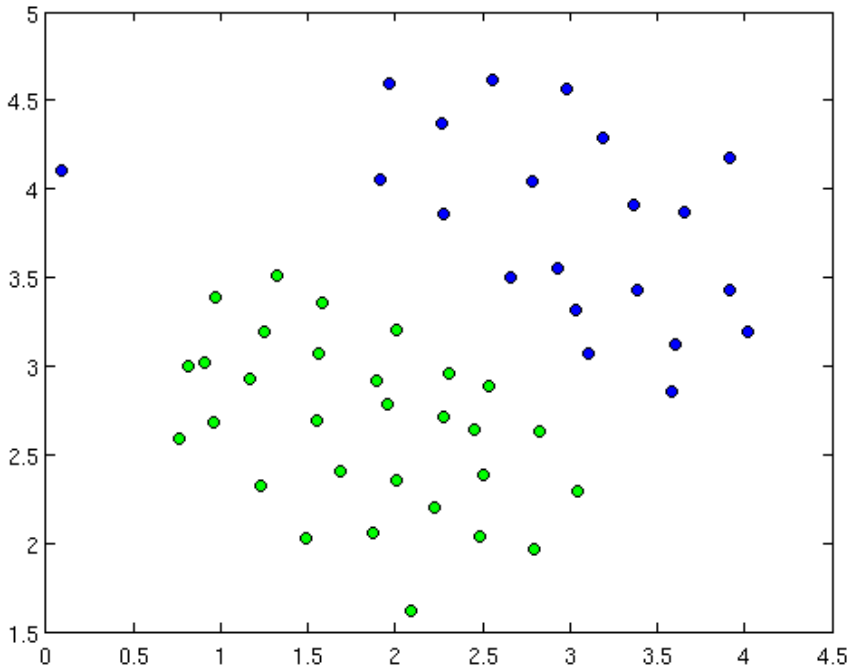
Efectos de los parámetros en SVM

Las SVM son **MUY SENSIBLES** a los parámetros

- El valor de C modifica el comportamiento de la SVM:
  - Un valor bajo hace a la SVM más permisiva con los outliers
  - Un valor alto hace a la SVM menos permisiva con los ejemplos extraños
- Valores altos de C → pesos α distintos de cero para mayor número de ejemplos (aumenta el nº de vectores de soporte) y valores más bajos en media (vectores soporte menos significativos)
- ¿Por qué? Las clasificaciones erróneas tienen un coste más elevado si C es mayor ← **peligro de sobreajuste**

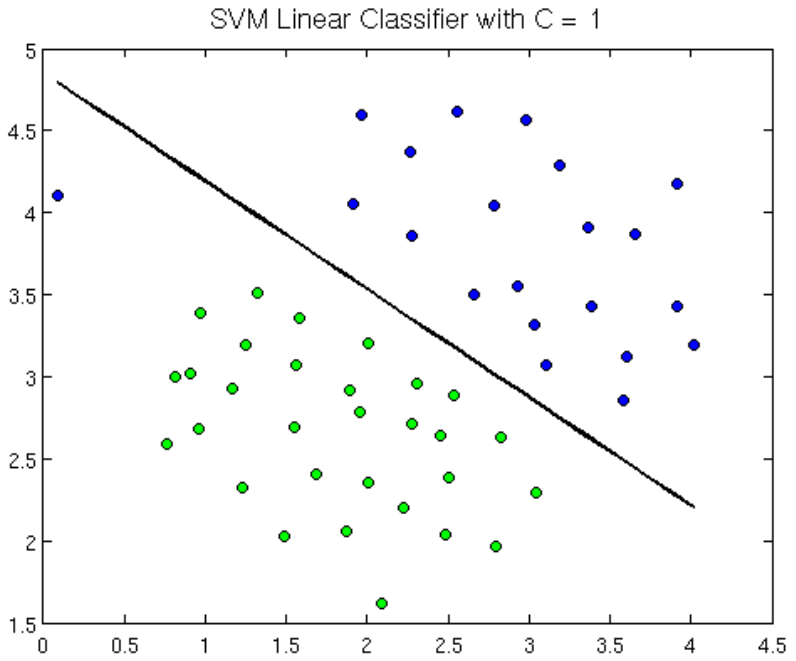
C bajo frente a C alto

- ¿Dónde trazarías el hiperplano de separación?



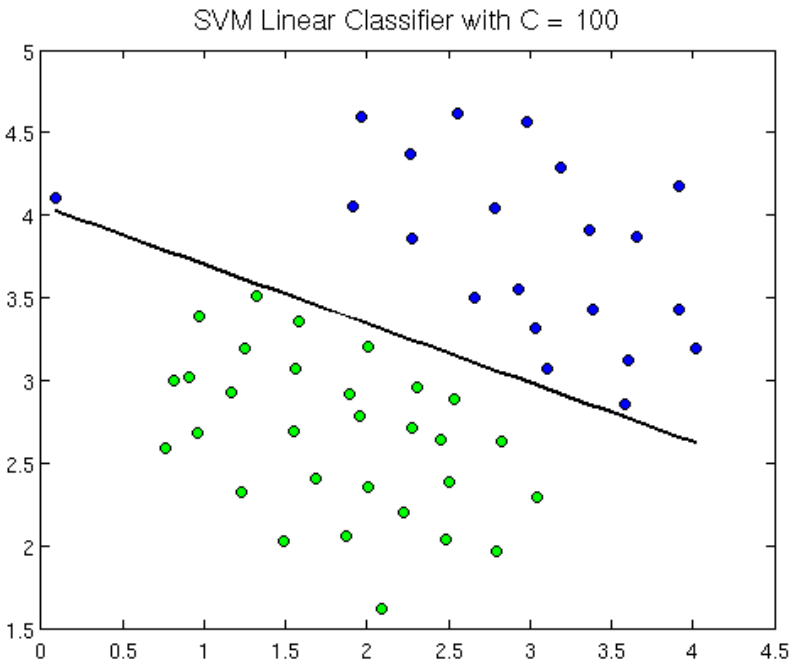
C bajo frente a C alto

- C = 1
- Parece razonable si consideramos el outlier como ruido



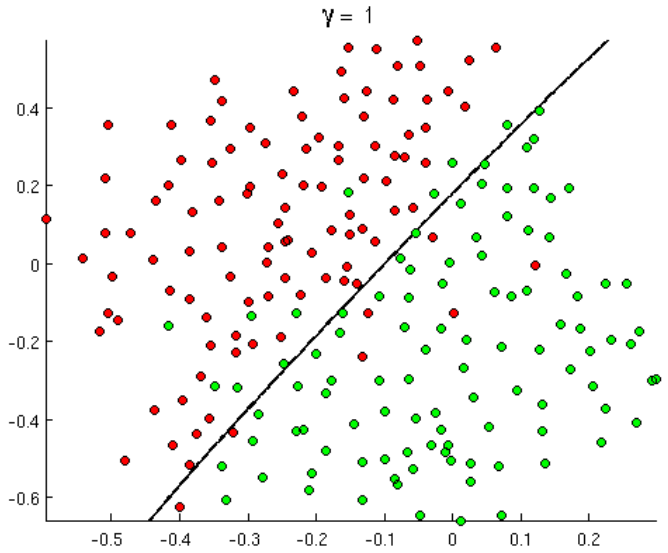
C bajo frente a C alto

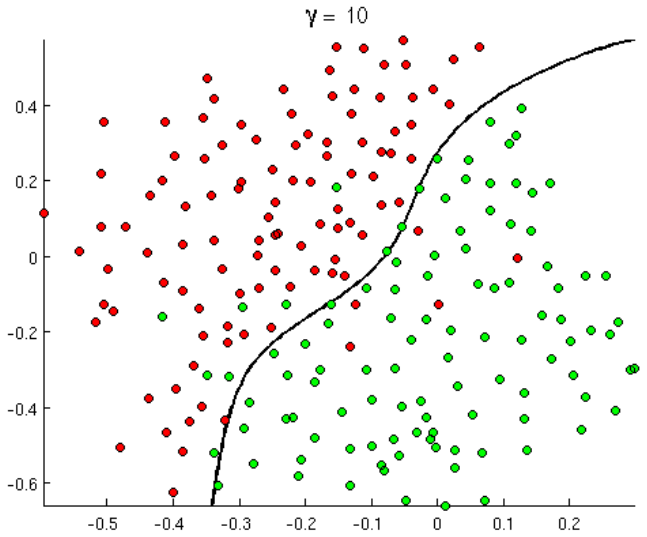
- C = 100
- La separación no parece tan natural.
- Puede sobreajustar.
-
- Muchos vectores de soporte implicados



Efectos de los parámetros en SVM

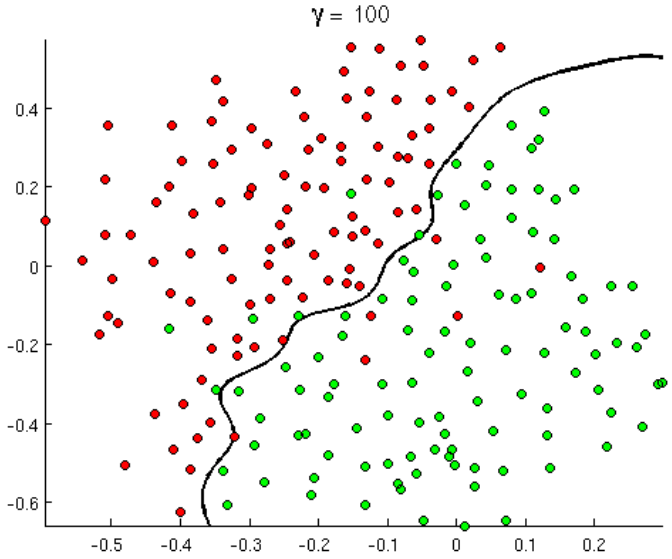
- ¡No hay que menospreciar los parámetros del kernel!
- En un kernel RBF se pueden apreciar como valores altos de gamma γ también pueden conducir a un sobreajuste

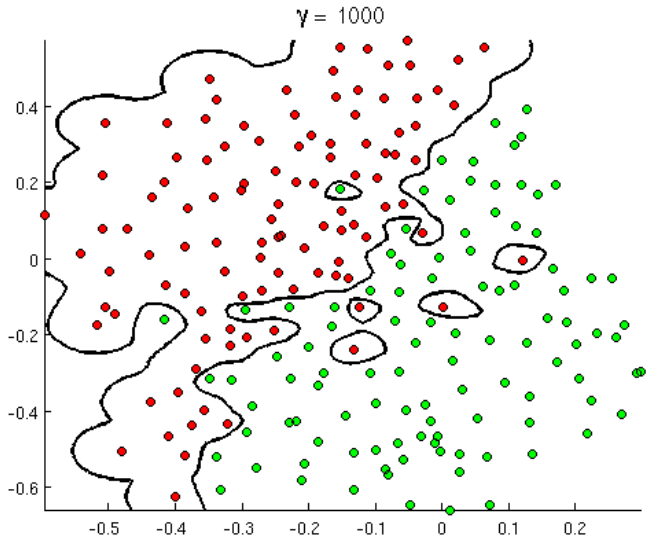




Efectos de los parámetros en SVM

- ¡No hay que menospreciar los parámetros del kernel!
- En un kernel RBF se pueden apreciar como valores altos de gamma γ también pueden conducir a un sobreajuste





Algunos pasos necesarios antes de usar SVMs

- **Recordemos**: los kernels establecen una medida implícita de similitud entre ejemplos → es conveniente normalizar los atributos
  - Rangos usuales: [-1,1] ó [0,1]
- Si hay valores nominales, hay que transformarlos:
  - La opción de usar índices de números enteros no es recomendable → matrices densas
  - Es mejor convertir cada atributo nominal a un conjunto de atributos binarios:
    - ✓ Matrices dispersas que para SMO van bien
    - x Mayor dimensionalidad → más lentitud

Resumen Máquinas de soporte vectorial (SVM)

"Una **SVM (support vector machine)** es un modelo de aprendizaje que se fundamenta en la *Teoría de Aprendizaje Estadístico.* La idea básica es encontrar un hiperplano canónico que maximice el margen del conjunto de datos de entrenamiento, esto nos garantiza una buena capacidad de generalización."

Representación dual de un problema

\+

Funciones Kernel

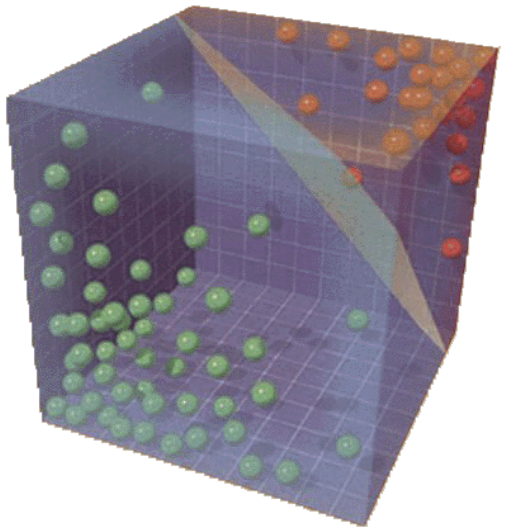
\+

Teoría de Aprendizaje Estadística

\+

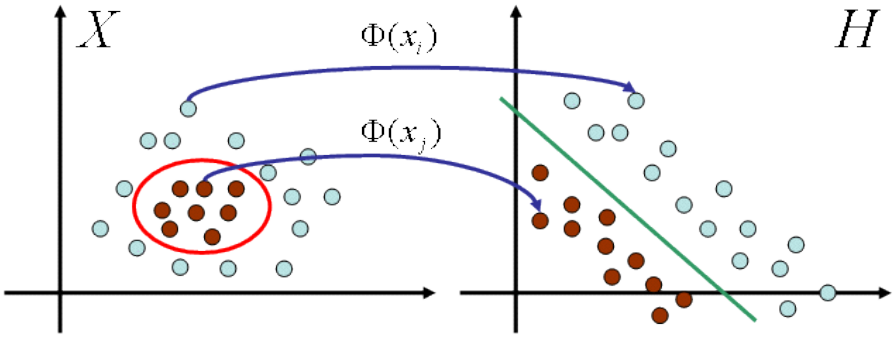
Teoría Optimización Lagrange

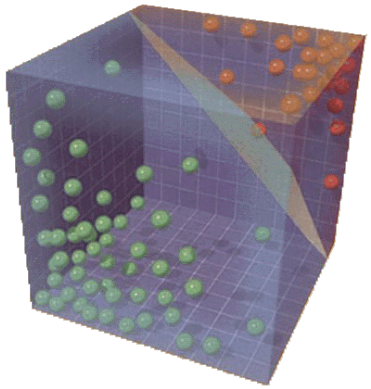
=



Resumen Máquinas de soporte vectorial (SVM)

- Una **SVM** es un **máquina de aprendizaje lineal** (requiere que los datos sean linealmente separables).
- Estos métodos explotan la información que proporciona el **producto interno (escalar)** entre los datos disponibles.
- La idea básica es:





Máquinas de soporte vectorial (SVM)

- Originariamente el **modelo de aprendizaje basado en SVMs** fue diseñado para **problemas de clasificación binaria**.
- La extensión a multi-clases se realiza mediante combinación de clasificadores binarios

(se estudia en la siguiente sección, modelos One vs One).