

# Support Vector Machines

Curso de aprendizaje automático para  
el INE

IGMAT

Alberto Torres Barrán

2019-04-24



# Introducción

# Introducción

- Regresión logística y LDA son clasificadores lineales
- Lineal  $\Rightarrow$  la frontera de decisión es un hyperplano en  $\mathbb{R}^d$ ,

$$f(x) = w_0 + w^T x = 0$$

- Existen clasificadores que intentan separar las clases lo mejor posible:
  - Perceptron
  - SVM
- Dado  $y \in \{-1, 1\}$  y  $f(x)$ , podemos definir la regla de clasificación

$$G(x) = \text{sign } f(x)$$

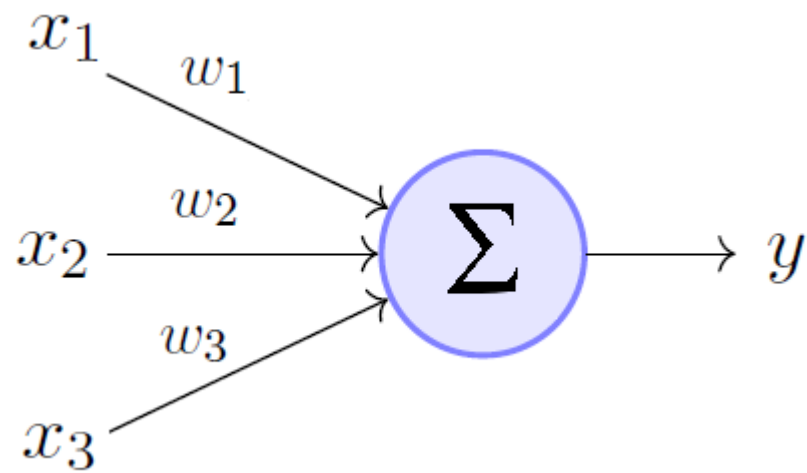
# Perceptron

- Rosenblatt, 1958
- Inspirado en las neuronas biológicas
- Inspiración a su vez de las redes neuronales de los 80
- Idea: minimizar la distancia de los puntos mal clasificados:

1. Si  $y_i = 1$ ,  $x_i^T w + b < 0$

2. Si  $y_i = -1$ ,  $x_i^T w + b > 0$

- Definimos  $D(b, w) = -\sum_{i \in M} y_i (x_i^T w + b)$ 
  - Positiva
  - Proporcional a la distancia de los puntos mal clasificados a la frontera



Fuente

# Algoritmo

- Descenso por gradiente estocástico
- Gradiente de un ejemplo mal clasificado:

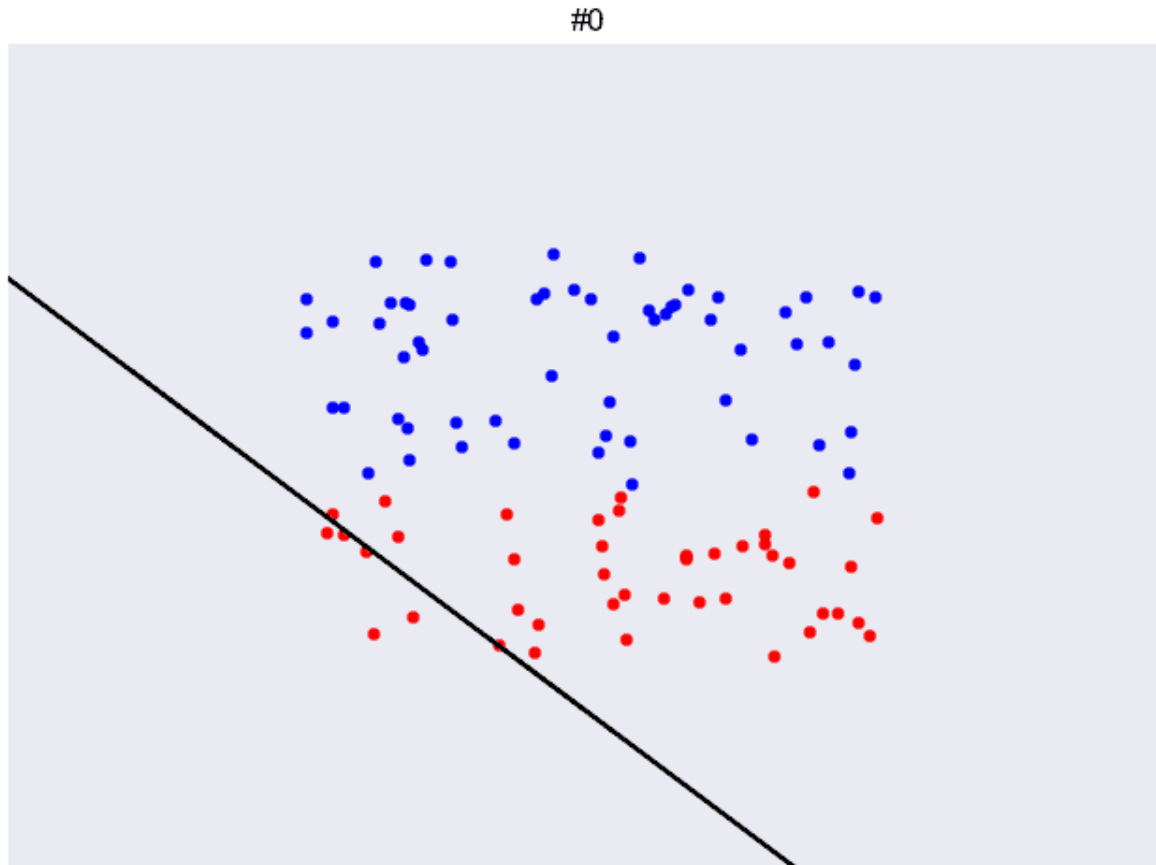
$$\partial_w D(w, b) = -y_i x_i, \quad \partial_b D(w, b) = -y_i$$

- Elegimos una observación  $i$  aleatoriamente:
  1. Si está bien clasificada, i.e.  $y_i(x_i^T w + b) > 0$ , no hacemos nada
  2. Si está mal clasificada, i.e.  $y_i(x_i^T w + b) \leq 0$ , actualizamos  $w$  y  $b$ ,

$$w \leftarrow w + y_i x_i, \quad b \leftarrow b + y_i$$

- Paramos cuando todas las observaciones están bien clasificadas

# Ejemplo perceptron

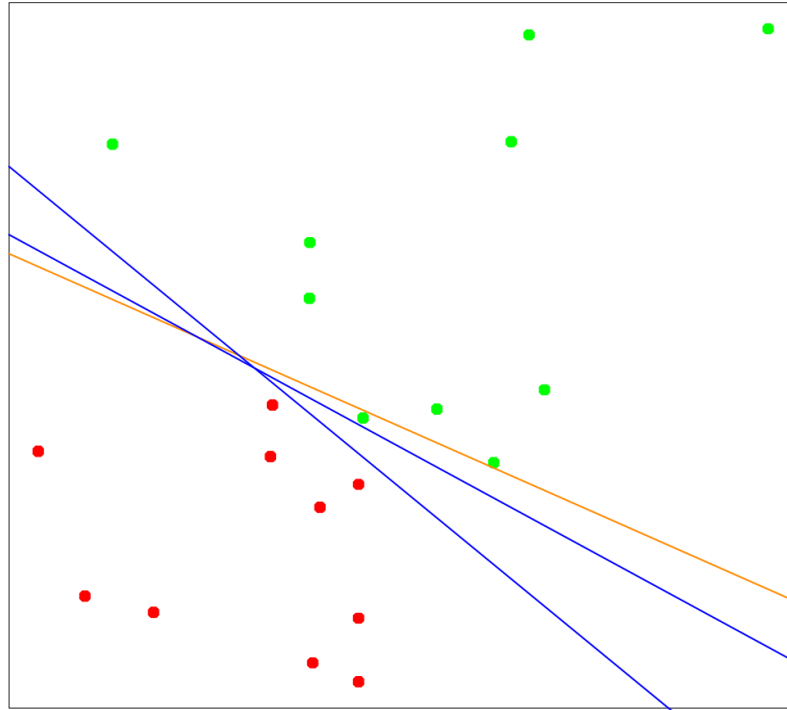




# Problemas perceptron

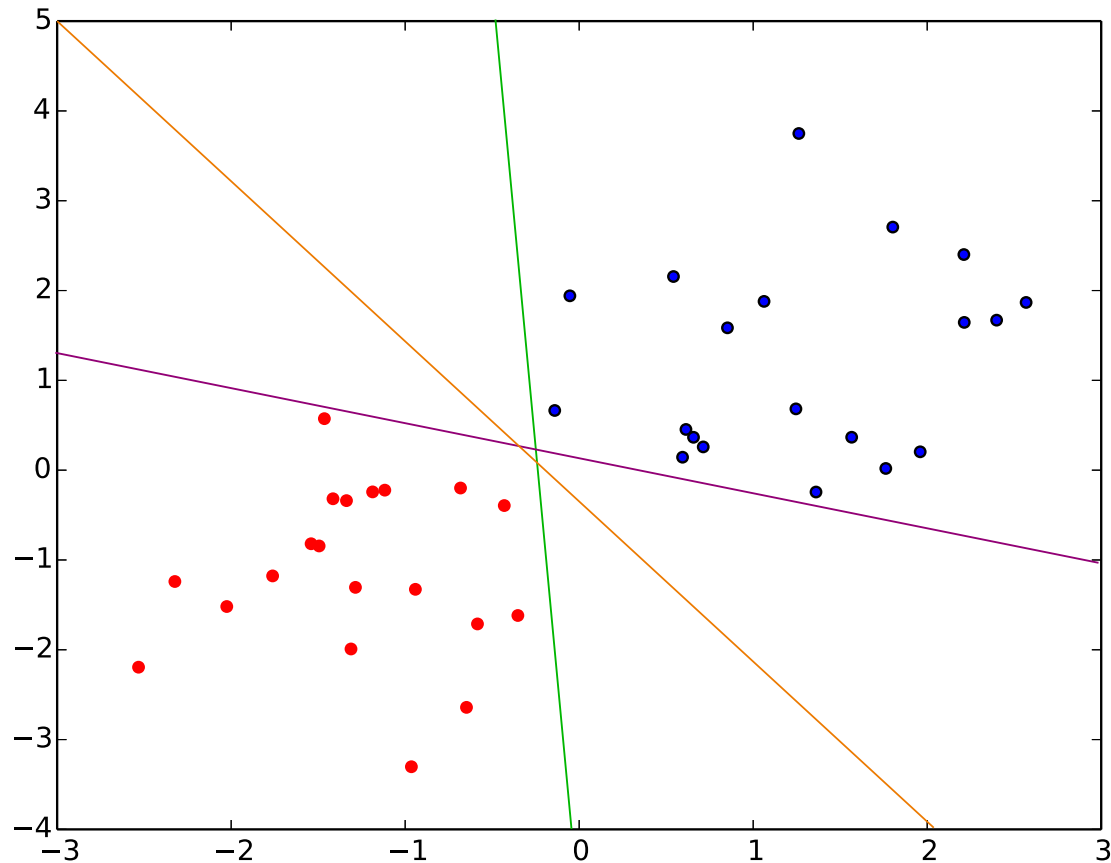
- Si las clases son linealmente separables:
  1. Converge a una solución
  2. Número de iteraciones finito
- Pero:
  1. Muchas soluciones posibles, dependen del punto inicial
  2. Convergencia lenta
  3. No converge si las clases no son linealmente separables

# Perceptron vs mínimos cuadrados



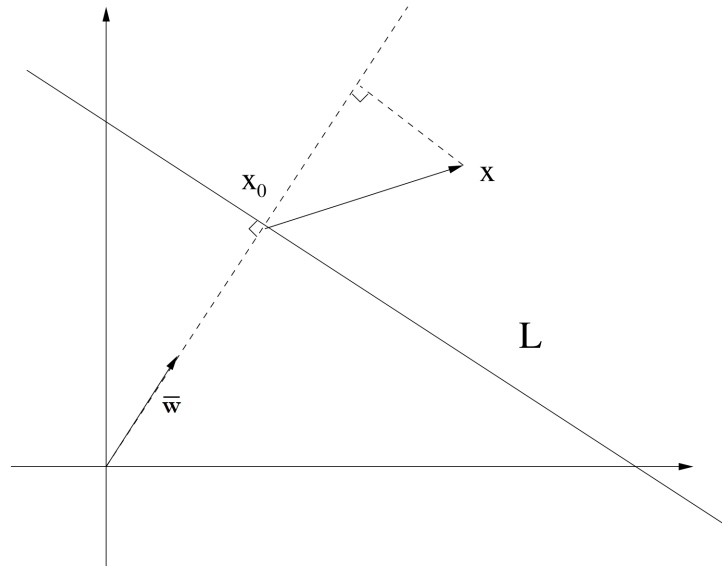
**FIGURE 4.14.** *A toy example with two classes separable by a hyperplane. The orange line is the least squares solution, which misclassifies one of the training points. Also shown are two blue separating hyperplanes found by the perceptron learning algorithm with different random starts.*

# ¿Qué solución escogemos?



# Support Vector Machines (SVM)

# Margen geométrico



- Dos puntos  $x_1$  y  $x_2$  de  $L$ ,  $w^T(x_1 - x_2) = 0$  y  $\bar{w} = w/||w||$  es el vector normal
- Para cualquier punto  $x_0$  de  $L$ ,  $w^T x_0 = -b$
- Distancia de un punto  $x$  a  $L$  (con signo),  $\bar{w}^T(x - x_0) = 1/||w|| (w^T x + b)$

# Clasificación de máximo margen

- Margen: distancia entre el hiperplano y el punto más cercano de una de las clases
- Idea: maximizar el margen

1. solución única

2. mejor generalización en el conjunto de test

- $M$  es el margen, queremos encontrar el hiperplano:

$$\begin{array}{ll} \max_{w,b} & M \\ \text{s.t} & y_i \frac{1}{\|w\|_2} (x_i^T w + b) \geq M \quad \forall i \end{array}$$

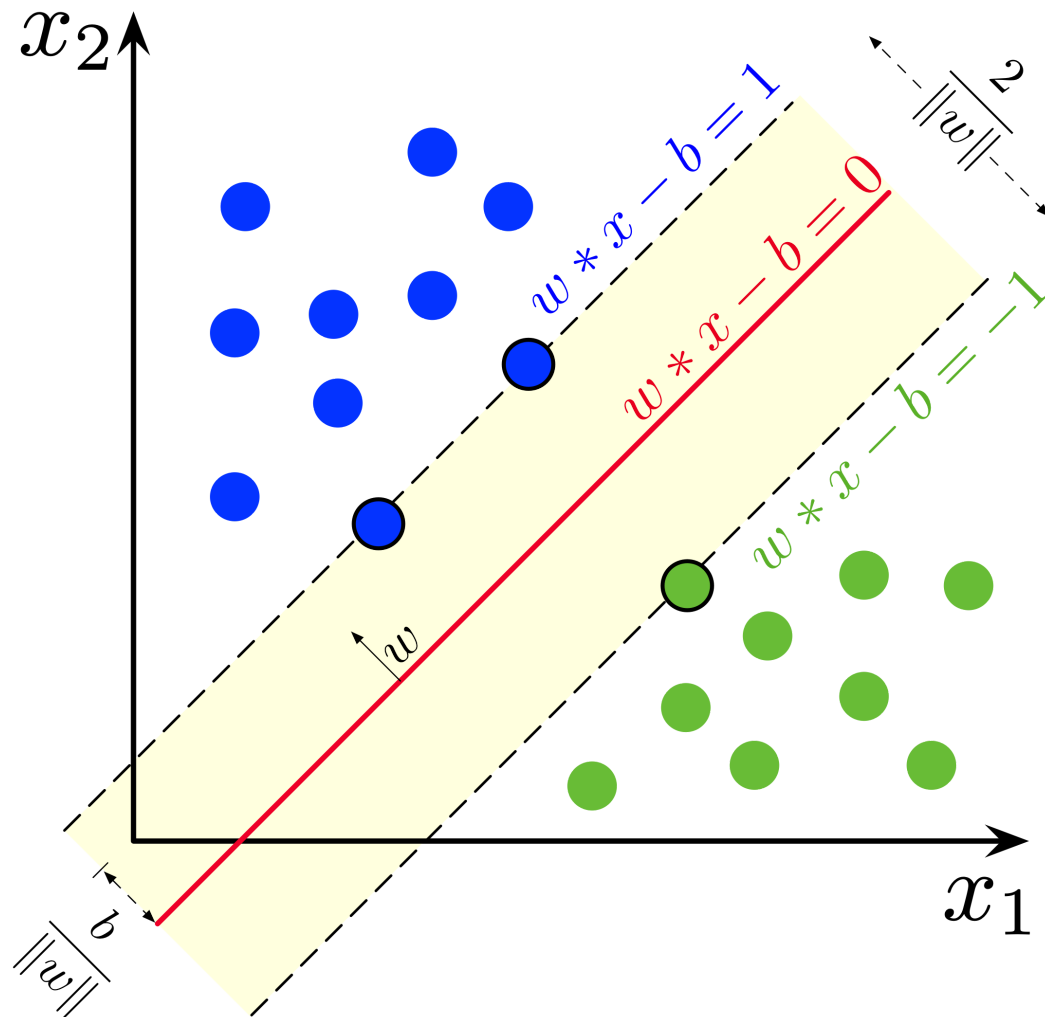
- La restricción nos asegura que todos los puntos están en el lado correcto del hiperplano y a una distancia mayor que  $M$  (fuera del margen)

# SVM lineal

- Si  $w$  y  $b$  satisfacen las restricciones, podemos multiplicar por cualquier número positivo
- Por comodidad escogemos  $\|w\|_2 = 1/M$
- Problema de optimización

$$\begin{array}{ll}\min_{w,b} & \frac{1}{2} \|w\|_2^2 \\ \text{s.t} & y_i(x_i^T w + b) - 1 \geq 0 \quad \forall i\end{array}$$

- Maximizar se convierte en minimizar porque  $w$  está en el denominador
- $1/2$  y el cuadrado se añaden por conveniencia (no cambian la solución)
- Ningún punto de entrenamiento está dentro del margen (por construcción)
- Esperamos: margen grande  $\Rightarrow$  mayor separación en test



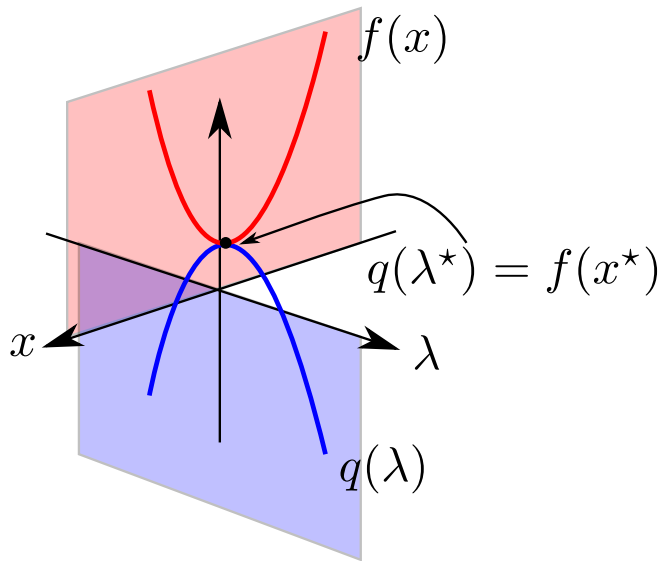
Wikipedia



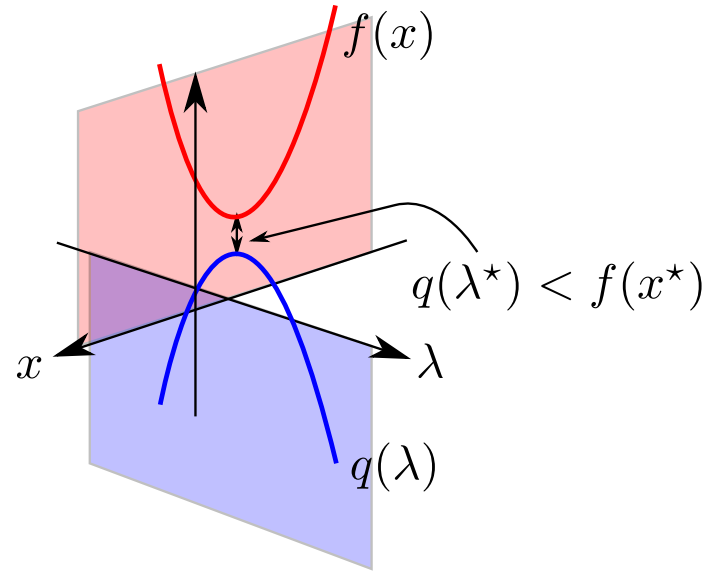
# Dualidad (Lagrange)

- Dado un problema de optimización (problema **primal**), se puede definir otro relacionado (problema **dual**)
- A menudo tiene propiedades complementarias
- Dualidad fuerte: los problemas primal y dual tienen la misma solución:
  - un ejemplo: función convexa y restricciones afines  $h(x) = Ax - b$
- En esos casos podemos elegir resolver uno u otro indistintamente

# Dualidad fuerte



strong duality



weak duality

Fuente

# SVM: derivación problema dual

- Lagrangiano

$$L_P = \frac{1}{2} \|w\|_2^2 - \sum_{i=1}^n \alpha_i y_i (x_i^T w + b) + \sum_{i=1}^n \alpha_i$$

- Problema dual:

$$\max_{\alpha} \left\{ \min_{w, b} L_P \right\} \quad \text{s.t.} \quad \alpha > 0$$

- Resolvemos el problema interior igualando derivadas a 0:

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

- Finalmente sustituimos en  $L_P$

# SVM: problema dual

- Problema optimización:

$$\begin{aligned} \max_{\alpha} \quad & \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j \\ \text{s.t} \quad & \alpha_i \geq 0 \quad \text{y} \quad \sum_i \alpha_i y_i = 0 \end{aligned}$$

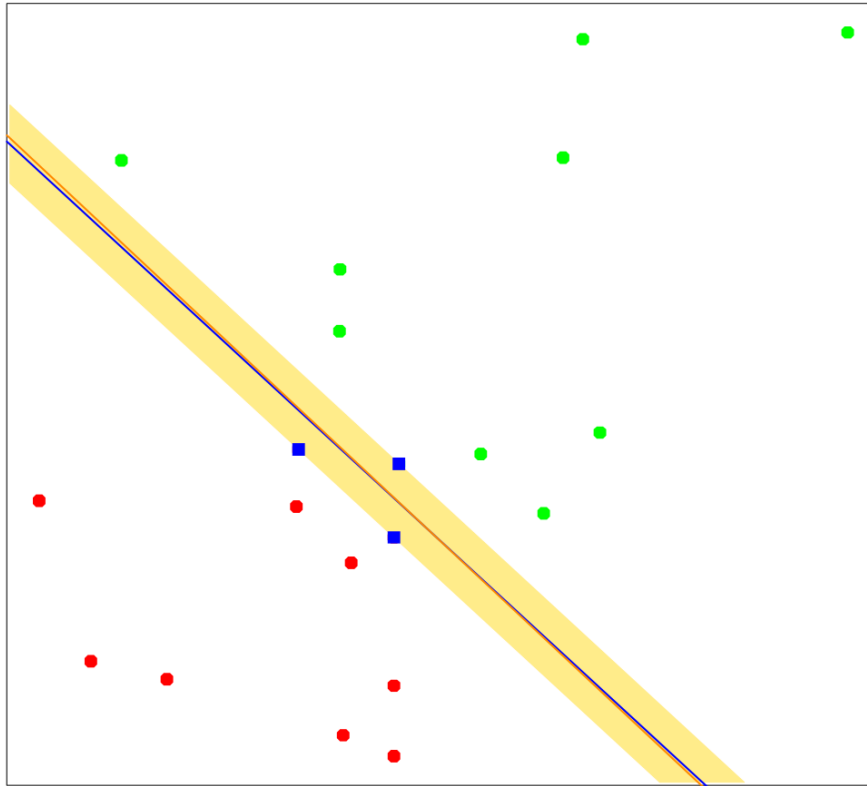
- La solución tiene que satisfacer las condiciones KKT, nos interesa una:

$$\alpha_i [y_i (x_i^T w + b) - 1] = 0$$

- Dos casos:

1.  $\alpha_i > 0$ , entonces  $y_i (x_i^T w + b) = 1$  y  $x_i$  está justo en el margen
2.  $\alpha_i = 0$ , y por tanto  $y_i (x_i^T w + b) > 1$

- $w^*$  solo depende de los  $x_i$  asociados a  $\alpha_i > 0 \Rightarrow$  **vectores de soporte**



**FIGURE 4.16.** *The same data as in Figure 4.14. The shaded region delineates the maximum margin separating the two classes. There are three support points indicated, which lie on the boundary of the margin, and the optimal separating hyperplane (blue line) bisects the slab. Included in the figure is the boundary found using logistic regression (red line), which is very close to the optimal separating hyperplane (see Section 12.3.3).*

# SVM: solución

- Dada una solución del problema dual  $\alpha^*$ , los coeficientes originales son

$$w^* = \sum_{i=1}^n \alpha_i^* y_i x_i$$

- Para cualquier vector de soporte ( $\alpha_i > 0$ ):

$$y_i(x_i^T w^* + b^*) = 1 \quad \Rightarrow \quad b^* = 1/y_i - x_i^T w^*$$

- En la práctica se hace la media para todos los vectores de soporte (estabilidad numérica)

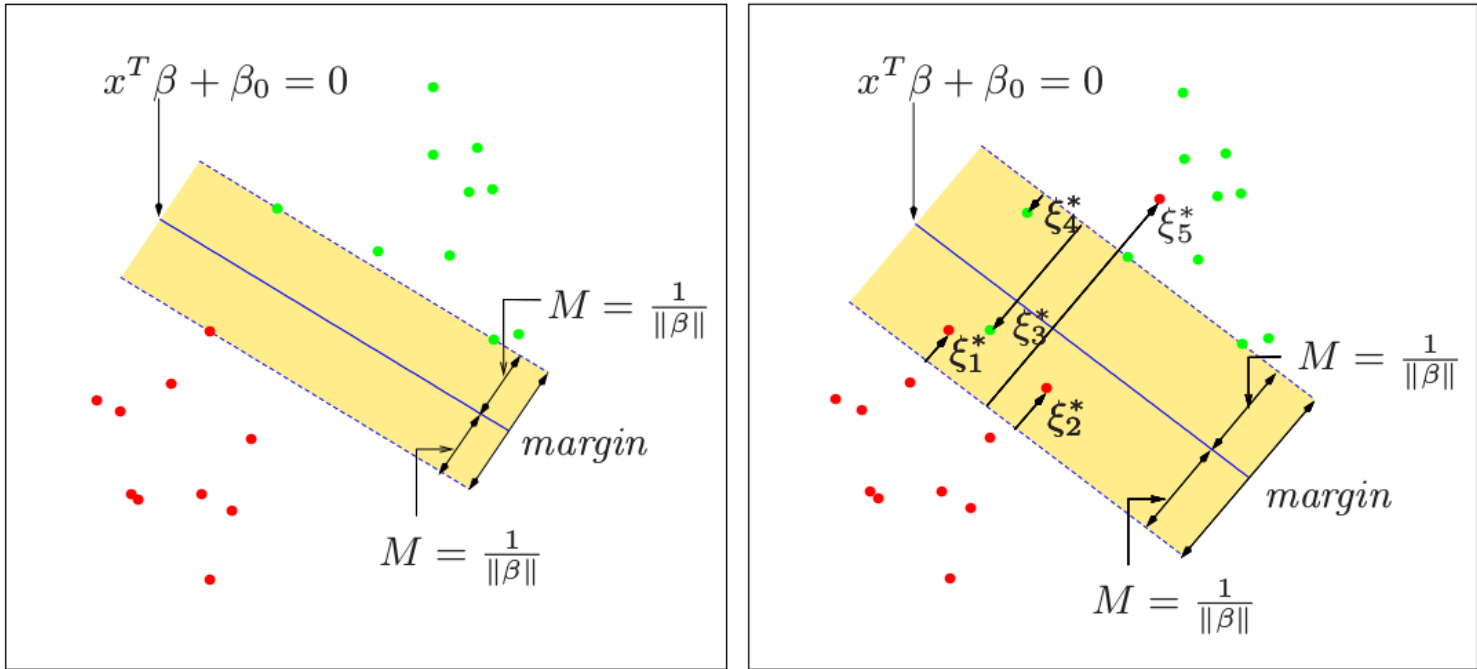
# SVM: margen flexible

- ¿Que pasa si ambas clases no son separables por un hiperplano?
- Idea: maximizar el margen, pero permitir que algunos puntos estén en el lado incorrecto
- Cambiar las restricciones por:

$$y_i(x_i^T w + b) \geq M(1 - \xi_i)$$

con  $\xi_i \geq 0$  y  $\sum \xi_i \leq \text{cte.}$

- $\xi_i$  indican la cantidad en proporcion por la que una variable está en el lado incorrecto:
  1.  $\xi_i = 0$ , punto clasificado correctamente
  2.  $0 < \xi_i \leq 1$ , punto **dentro** del margen
  3.  $\xi_i > 1$ , punto clasificado incorrectamente



**FIGURE 12.1.** Support vector classifiers. The left panel shows the separable case. The decision boundary is the solid line, while broken lines bound the shaded maximal margin of width  $2M = 2/\|\beta\|$ . The right panel shows the nonseparable (overlap) case. The points labeled  $\xi_j^*$  are on the wrong side of their margin by an amount  $\xi_j^* = M\xi_j$ ; points on the correct side have  $\xi_j^* = 0$ . The margin is maximized subject to a total budget  $\sum \xi_i \leq \text{constant}$ . Hence  $\sum \xi_j^*$  is the total distance of points on the wrong side of their margin.



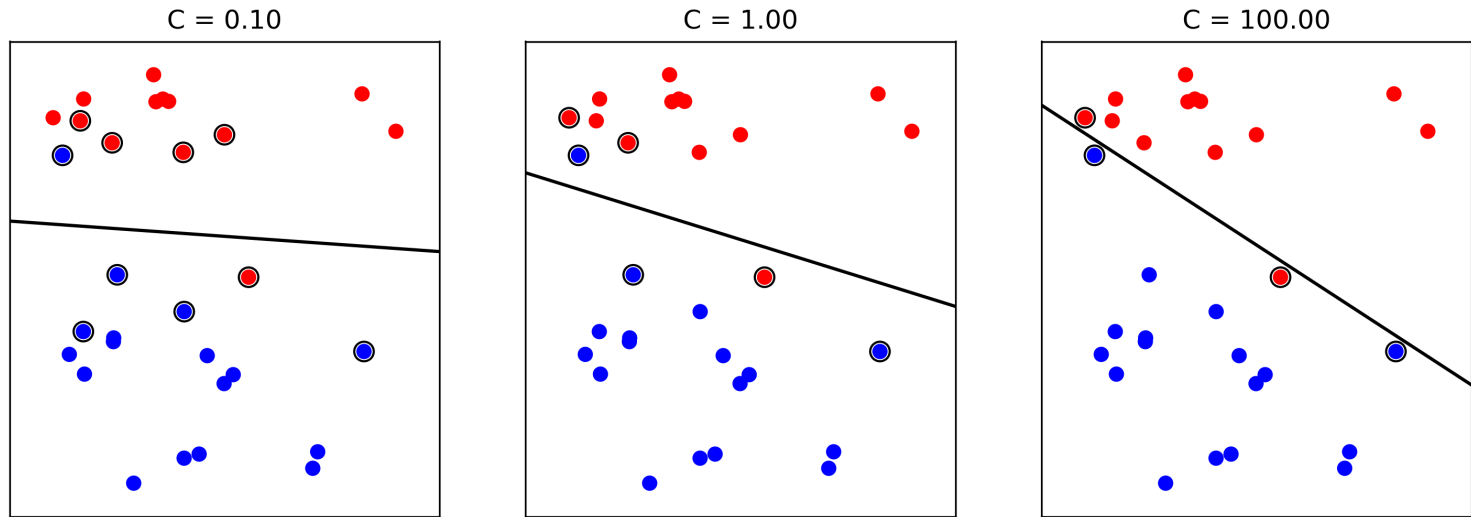
# C-SVM: formulación

- Número de puntos mal clasificados acotado superiormente por  $\sum \xi_i$
- Maximizar el margen y minimizar el número de errores de clasificación:

$$\begin{array}{ll}\min_{w,b,\xi} & \frac{1}{2} \|w\|_2^2 + C \sum \xi_i \\ \text{s.t} & y_i(x_i^T w + b) \geq 1 - \xi_i \quad \forall i \\ & \xi_i \geq 0 \quad \forall i.\end{array}$$

- $C > 0$  es un hiper-parámetro que controla la complejidad:
  1.  $\uparrow C$ , más importancia a clasificar correctamente todos los puntos (menos regularización)
  2.  $\downarrow C$ , más importancia a maximizar el margen (más regularización)

# Efecto de C



Andreas C. Müller, [Linear Models for Classification](#)

# C-SVM: formulaci3n dual

- La formulaci3n dual es:

$$\begin{aligned} \max_{\alpha} \quad & \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j \\ \text{s.t} \quad & 0 \leq \alpha_i \leq C \quad \forall i \\ & \sum_i \alpha_i y_i = 0. \end{aligned}$$

- Notaci3n vectorial:

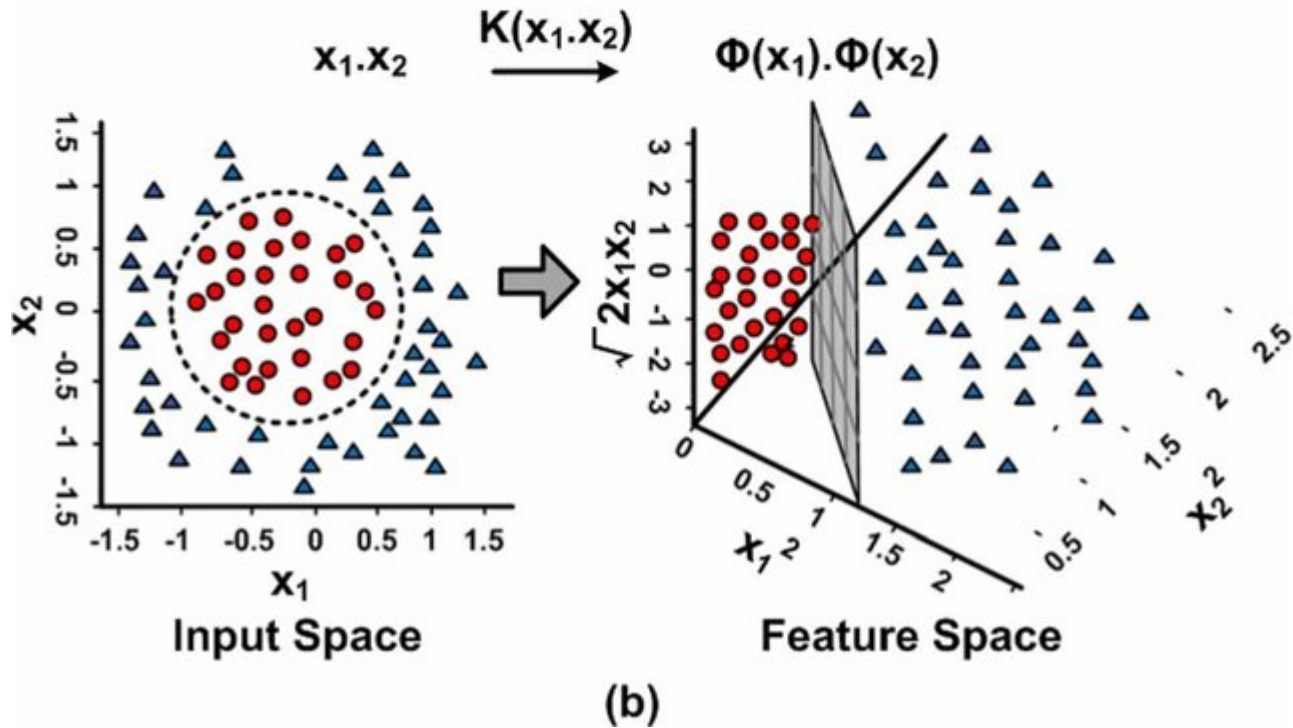
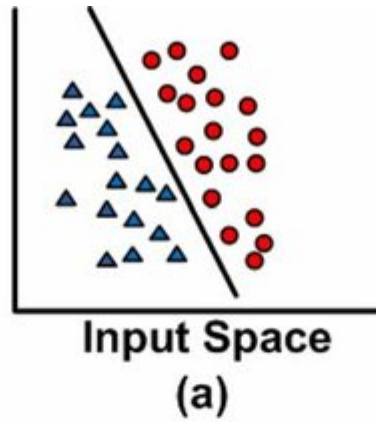
$$\min_{\alpha} \left\{ \frac{1}{2} \alpha^T \mathbf{Q} \alpha - \alpha^T \mathbf{1} \right\} \quad \text{s.t.} \quad \alpha^T \mathbf{y} = 0 \quad \text{y} \quad 0 \leq \alpha_i \leq C, \quad \forall i$$

donde  $\mathbf{Q}$  es una matriz  $n \times n$  con elementos  $Q_{ij} = y_i y_j x_i^T x_j$ .

- Con respecto a SVM *hard margin* solo cambia la restricci3n  $0 \leq \alpha \leq C$

# SVM no lineal

- C-SVM acepta datos no separables linealmente, pero la frontera de decisión es lineal (hiperplano)
- Idea: transformar variables originales en otras variables de mayor dimensión
- 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
- Similar a lo que vimos en regresión lineal de añadir expansiones polinómicas



# SVM no lineal: problemas

- $\phi(x_i) : R^d \rightarrow R^D$ , con  $D \gg d$
- Si intentamos calcular la función  $\phi(x_i)$  explícitamente:
  1. El espacio ampliado puede tener dimensión  $D$  muy grande, incluso infinita
  2. Computacionalmente muy costoso calcular la función  $\phi$  cada vez que sea necesario
  3. Complicado de almacenar en memoria

# Kernel trick

- Elementos de la matriz  $\mathbf{Q}$ ,  $Q_{ij} = y_i y_j x_i^T x_j$
- Solo depende del producto escalar de  $x_i$  y  $x_j$
- Podemos reemplazar el producto escalar por una función de kernel:

$$k(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

- No es necesario calcular  $\phi$  explícitamente, solo  $k$
- $k$  cualquier función simétrica (semi-) definida positiva
- Ejemplos:

1. Kernel polinómico:  $k(x, x') = (1 + x^T x')^p$

2. Kernel RBF:  $k(x, x') = \exp(-\gamma \|x - x'\|_2^2)$

# Ejemplo kernel polinómico

- Si  $d = 2$  y  $p = 2$ ,

$$\begin{aligned}k(x, x') &= (1 + x^T x')^2 = (1 + x_1 x'_1 + x_2 x'_2)^2 \\&= 1 + 2x_1 x'_1 + 2x_2 x'_2 + (x_1 x'_1)^2 + (x_2 x'_2)^2 + 2x_1 x'_1 x_2 x'_2\end{aligned}$$

- Por tanto

$$\phi(x) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1 x_2)$$

- Dimensión espacio original  $d = 2$
- Dimensión espacio ampliado  $D = 6$
- En otros kernels (por ej. RBF),  $\phi(\cdot)$  no se puede calcular explícitamente



# SVM no lineal: formulación

- Problema optimización:

$$\min_{\alpha} \left\{ \frac{1}{2} \alpha^T \mathbf{Q} \alpha - \alpha^T \mathbf{1} \right\} \quad \text{s.t.} \quad \alpha^T \mathbf{y} = 0 \quad \text{y} \quad 0 \leq \alpha_i \leq C, \forall i$$

con  $Q_{ij} = y_i y_j k(x_i, x_j)$

- **Q** es la **matriz de kernel**
- Solo cambia **Q**, el resto idéntico
- Valor de hiper-parámetros es crítico para buen rendimiento:
  1.  $C$ , parámetro de complejidad
  2. Parámetros del kernel, por ejemplo  $\gamma$  en el kernel RBF

# Cálculo de $w$ y $b$

- El valor de  $w$  es ahora:

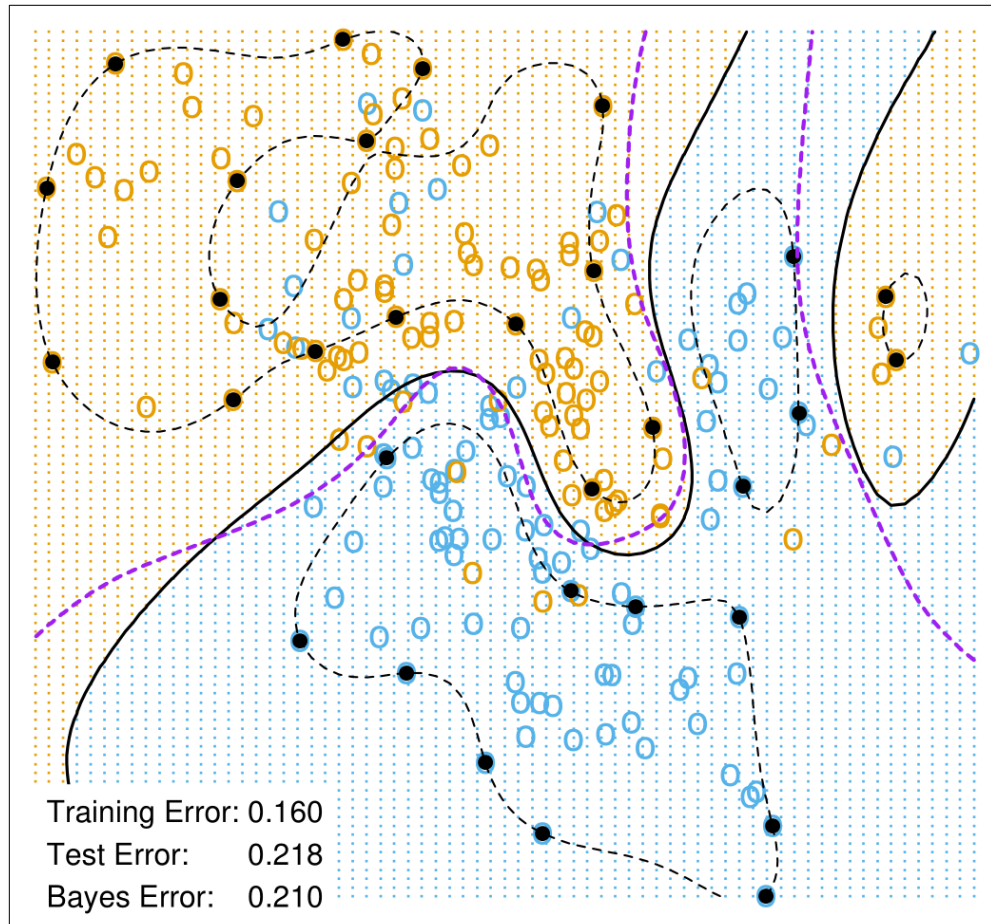
$$w^* = \sum_{i=1}^n \alpha_i^* y_i \phi(x_i)$$

- No podemos calcular su valor explícitamente, pero dado un nuevo  $x'$ :

$$\begin{aligned} f(x') &= \phi(x')^T w^* + b^* = \sum_{i=1}^n \alpha_i^* y_i \phi(x')^T \phi(x_i) + b^* \\ &= \sum_{i=1}^n \alpha_i^* y_i k(x', x_i) + b^* \end{aligned}$$

- $b^*$  se puede calcular como antes, resolviendo  $y_i f(x_i) = 1$  para cualquier vector de soporte ( $\alpha_i > 0$ )

# Ejemplo kernel RBF



# SVM como regularización

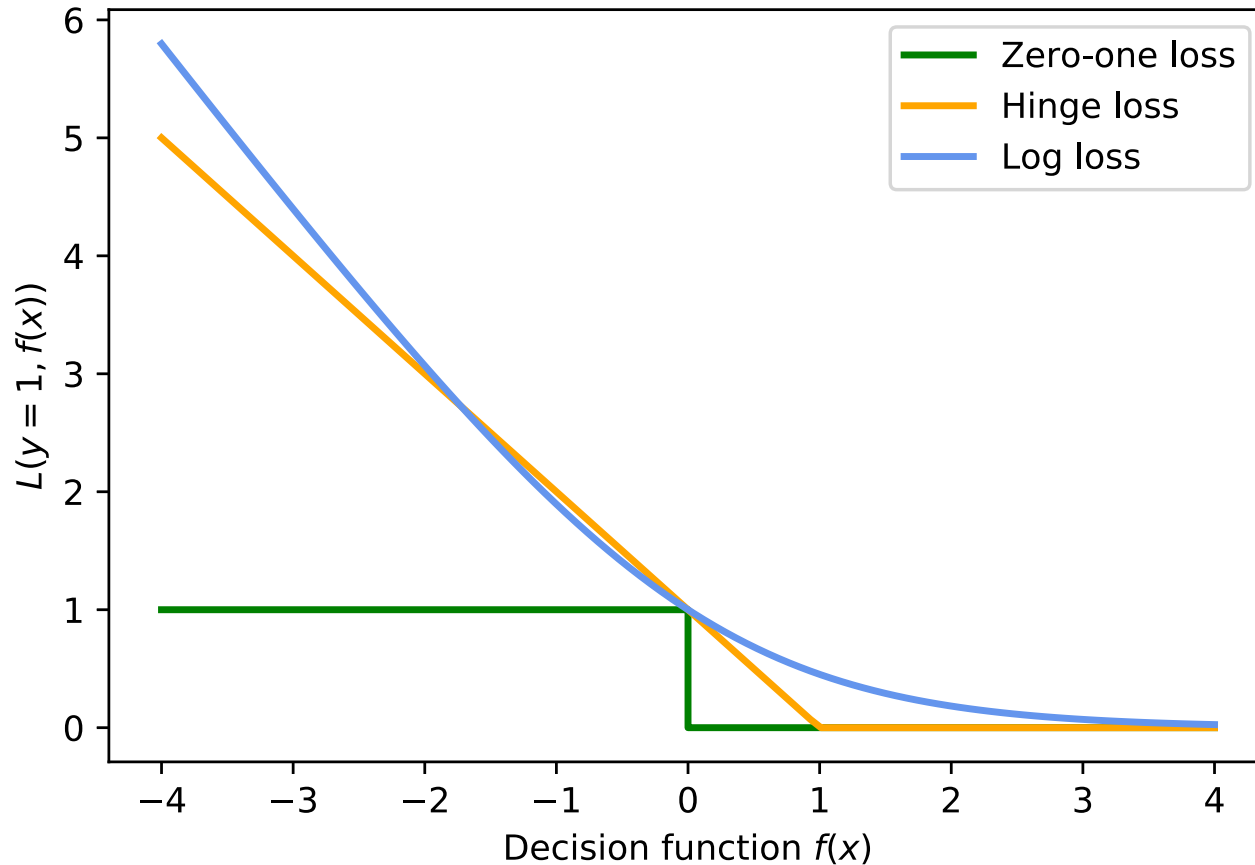
- Sea  $f(x) = \phi(x)^T w + b$
- Considerar el problema de optimización

$$\min_{w,b} \sum_{i=1}^n \max\{1 - y_i f(x_i), 0\} + \frac{\lambda}{2} \|w\|_2^2$$

- Tiene la forma *perdida* + *regularización*
- La solución es la misma que la de C-SVM con  $\lambda = 1/C$
- La función de pérdida es la *hinge loss*,

$$L(y, f(x)) = \max\{1 - yf(x), 0\}$$

# Comparación funciones pérdida



# SVM para regresión

- Empezamos por el caso lineal,  $f(x) = x^T w + b$
- Formulación regularizada,

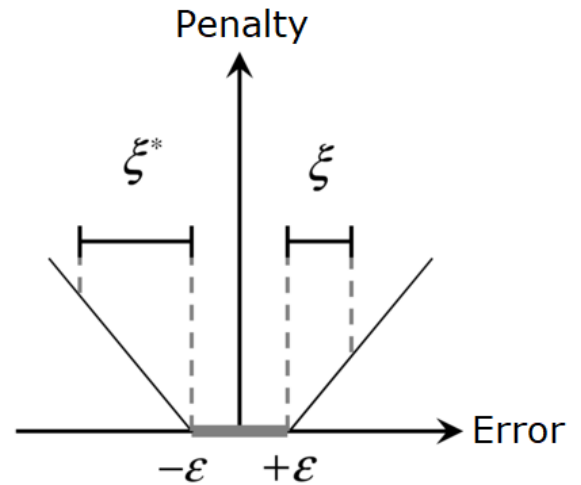
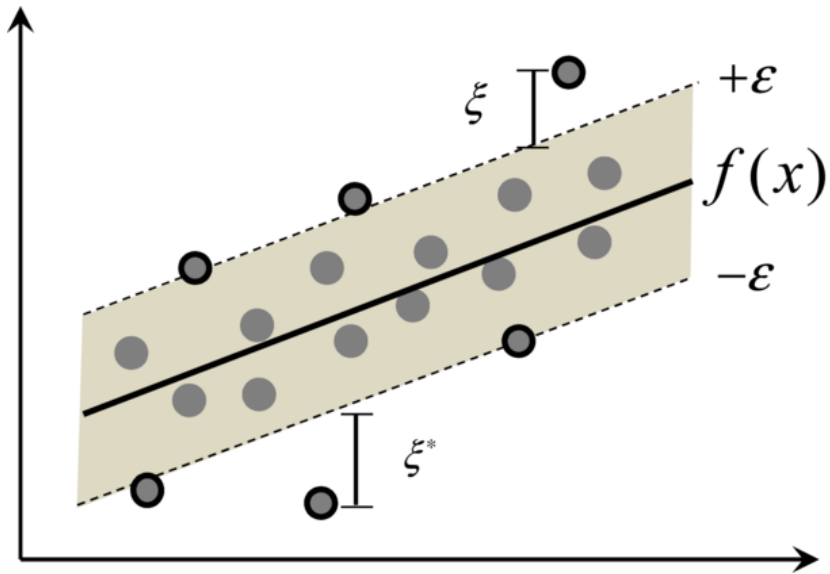
$$\sum_{i=1}^n L(y_i, f(x_i)) + \frac{\lambda}{2} \|w\|_2^2$$

con

$$L_{\epsilon}(y, f(x)) = \begin{cases} 0 & \text{si } |y - f(x)| \leq \epsilon \\ |y - f(x)| - \epsilon & \text{otro caso} \end{cases}$$

- Pérdida  $\epsilon$ -insensitiva
- Ignora errores menores que  $\epsilon$  (en valor absoluto)

# Pérdida $\epsilon$ -insensitiva



Yu et. al, 2012

# SVR: problema dual

- $w^*$  y  $b^*$  óptimos del problema anterior tienen la forma:

$$w^* = \sum_{i=1}^n (\alpha_i^* - (\alpha'_i)^*) x_i$$
$$f(x) = \sum_{i=1}^n (\alpha_i^* - (\alpha'_i)^*) k(x_i, x) + b$$

- $\alpha^*$  y  $(\alpha')^*$  son las soluciones del problema:

$$\begin{aligned} \min_{\alpha, \alpha'} \quad & \frac{1}{2}(\alpha - \alpha')^T \mathbf{Q}(\alpha - \alpha') + \epsilon \sum_{i=1}^n (\alpha_i + \alpha'_i) + \sum_{i=1}^n y_i(\alpha_i - \alpha'_i) \\ \text{s.t} \quad & (\alpha - \alpha')^T \mathbf{1} = 0, \\ & 0 \leq \alpha_i, \alpha'_i \leq C, \quad \forall i \end{aligned}$$

- $\mathbf{Q}$  es la matriz de kernel,  $Q_{ij} = k(x_i, x_j)$



# SVR como problema de clasificación

- Con las transformaciones;

$$\begin{aligned}\tilde{\alpha} &= \begin{bmatrix} \alpha' \\ \alpha \end{bmatrix} \in \mathbb{R}^{2n}, \\ \tilde{\mathbf{Q}} &= \begin{bmatrix} \mathbf{Q} & -\mathbf{Q} \\ -\mathbf{Q} & \mathbf{Q} \end{bmatrix} \in \mathbb{R}^{2n \times 2n}, \\ p &= \begin{bmatrix} \epsilon \mathbf{1}_n - y \\ \epsilon \mathbf{1}_n + y \end{bmatrix} \in \mathbb{R}^{2n}, \\ \tilde{y} &= [1, \dots, 1, -1, \dots, -1]^T \in \mathbb{R}^{2n}.\end{aligned}$$

- El problema de la SVR se convierte en uno de clasificación:

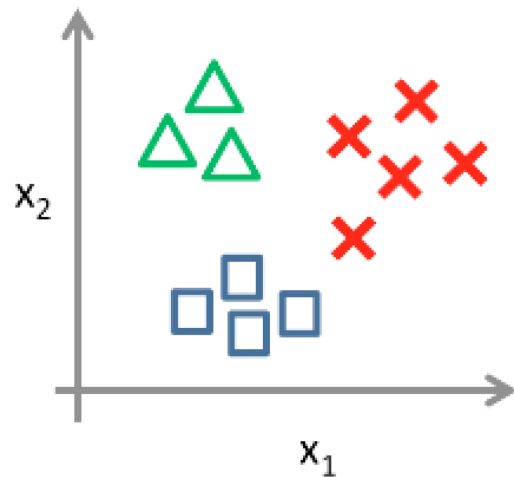
$$\begin{aligned}\min_{\tilde{\alpha}} \quad & \frac{1}{2} \tilde{\alpha}^T \tilde{\mathbf{Q}} \tilde{\alpha} + p^T \tilde{\alpha} \\ \text{s.t} \quad & \tilde{\alpha}^T \tilde{y} = 0, \\ & 0 \leq \tilde{\alpha}_i \leq C, \quad i = 1, \dots, 2n.\end{aligned}$$




- En el caso de la SVC,  $p = \mathbf{1}$

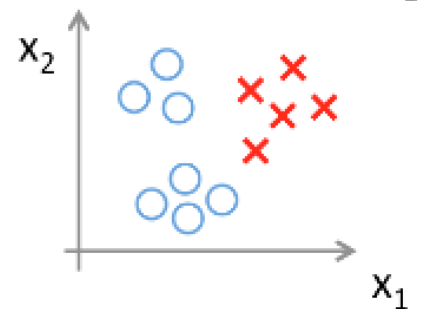
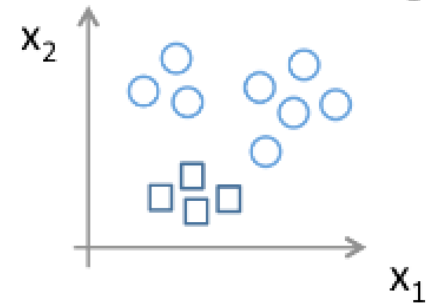
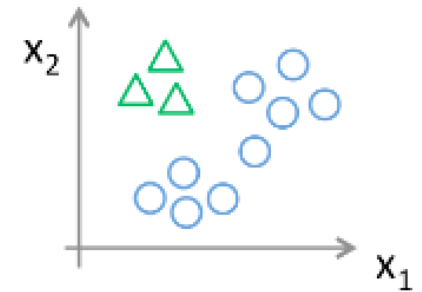
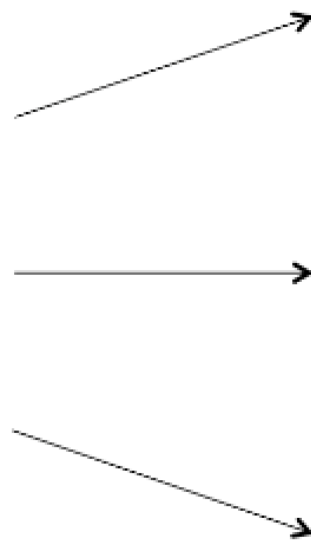
# SVM multiclase

- Dos aproximaciones directas para extender a  $K$  clases:
  1. *One-vs-all* (OVA): se construyen  $|K|$  clasificadores con los datos de una clase contra el resto
  2. *One-vs-one* (OVO): se construyen  $|K|(|K| - 1)/2$  clasificadores por cada par de clases, y se elige la clase que predice la mayoría
- OVO construye más clasificadores, pero cada uno se entrena con menos datos
- Otras aproximaciones más sofisticadas
- Comparación OVO y OVA: Hsu y Lin, 2002

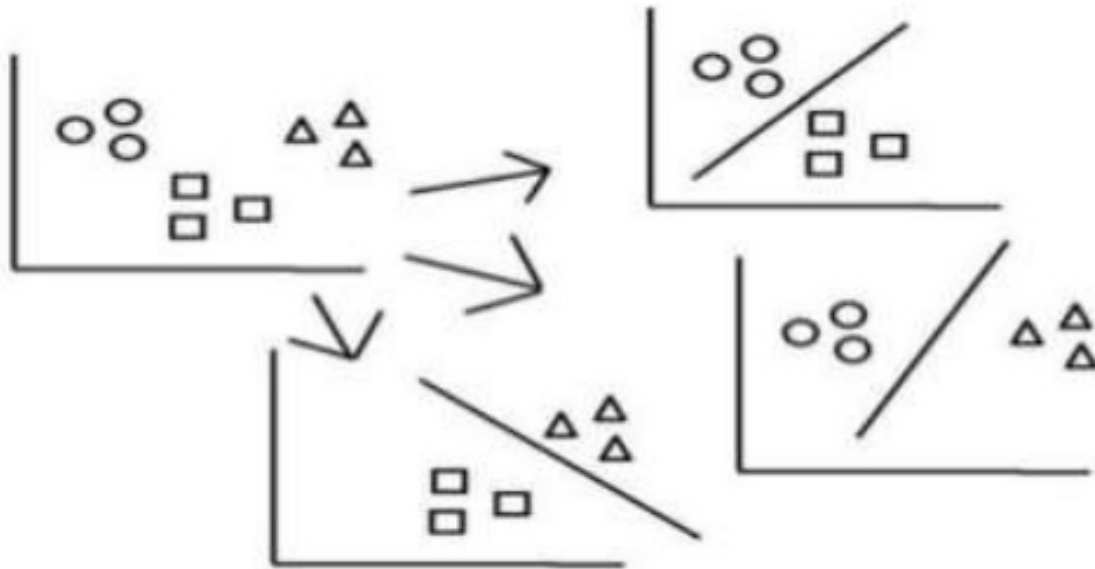
## One-vs-all (one-vs-rest):



Class 1:   
Class 2:   
Class 3: 



## One-vs-One (OVO)



Sergey Ivanov, Multiclass classification

# SVM: selección de hiperparámetros

- El valor de los hiper-parámetros es crítico para el rendimiento de la SVM
- El kernel RBF es el más popular, aunque se podría considerar como otro hiper-parámetro
- Generalmente se seleccionan usando búsqueda exhaustiva:
  - comparar error de validación cruzada en una rejilla de valores (escala logarítmica base 2)
- Clasificación:
  1.  $C$ , coste: de -5 a 15, paso 2
  2.  $\gamma$ , parámetro kernel RBF: de -15 a 3, paso 2
- Regresión: los anteriores y, además,
  1.  $\epsilon$ , parámetro de la función de pérdida: de -8 a -1, paso 1

# Algoritmos

- Históricamente los algoritmos para optimizar la SVM resuelven el problema dual:
  1. Restricciones más sencillas
  2. Sencillo extender al caso no lineal
  3. Escala mal con el número de patrones  $n$
- Alternativamente se puede resolver la formulación de Lagrange (perdida + regularización):
  1. No tiene restricciones
  2. Pérdida no diferenciable
  3. Complicado extender al caso no lineal
- Existen muchos algoritmos, pero vamos a centrarnos en el más popular: SMO

# Sequential Minimal Optimization (SMO)

# Introducción

- Principal complicación del problema dual: calcular matriz de kernel  $\mathbf{Q}$ 
  - Tamaño  $n \times n$
  - Costoso computacionalmente
- Solo depende de los datos  $\{\mathbf{X}, y\}$  pero precalcular no es buena idea:
  - Solo hacen falta entre 15-50% de los valores  $k(x_i, x_j)$  para calcular el óptimo
  - Al aumentar  $n$ , puede no haber espacio en memoria para almacenarla



# Características

- Descenso coordinado en el problema dual
- Actualiza dos coeficiente por iteración
- Calcular el kernel a medida que va siendo necesario
- Los coeficientes se eligen usando reglas heurísticas
  - elegir los que maximizan la disminución de la función objetivo

# Subproblema

- Seleccionar coeficientes a optimizar  $\alpha_i$  y  $\alpha_j$
- Eliminamos términos que no dependen de esos coeficientes:

$$\begin{aligned} \min_{\alpha_i, \alpha_j} \quad & \frac{1}{2} (\alpha_i^2 Q_{ii} + 2\alpha_i \alpha_j Q_{ij} + \alpha_j^2 Q_{jj}) + \alpha_i \sum_{k \notin \{i, j\}} Q_{ik} \alpha_k + \alpha_j \sum_{k \notin \{i, j\}} Q_{jk} \alpha_k - \alpha_i - \alpha_j \\ \text{s.t} \quad & y_i \alpha_i + y_j \alpha_j = - \sum_{k \notin \{i, j\}} y_k \alpha_k, \\ & 0 \leq \alpha_i, \alpha_j \leq C. \end{aligned}$$

- Hay que optimizar 2 coeficientes por iteración debido a restricción
- Ahora solo aparecen las filas (o columnas)  $\mathbf{Q}_i$  y  $\mathbf{Q}_j$ 
  - se calculan para esta iteración y no tenemos que almacenar toda la matriz
- Este subproblema tiene solución analítica!!

# Actualizar coeficientes

- La actualización de los coeficientes es

$$\alpha^{k+1} = \alpha^k + \rho(y_i e_i - y_j e_j) = \alpha^k + \rho d$$

donde  $e_k$  es el vector con todo 0 excepto un 1 en la posición  $k$ .

- Solo cambian dos coeficientes!!
- $\rho$  se calcula minimizando la función en la dirección  $d$ 
  1. sencillo, es un problema convexo cuadrático y solo con una dimensión
  2. hay que asegurarse de que al avanzar  $\rho$  se satisfacen las restricciones
- Se itera hasta converger al óptimo  $\alpha^*$

# Implementación

Dos técnicas principales para implementar SMO de forma eficiente:

## 1. Caching:

- almacenar las filas de la matriz de kernel  $\mathbf{Q}_i, \mathbf{Q}_j$  a medida que se calculan
- si se eligen posteriormente los coeficientes  $i$  o  $j$  las recuperamos
- si se agota el almacenamiento (*cache*), eliminamos las filas más antiguas
- típicamente se indica un valor máximo en Mb

## 2. Shrinking:

- a lo largo del entrenamiento, muchos  $\alpha_i$  tendrán valor 0 o  $C$
- el valor de esos coeficientes se mantiene hasta el final
- se pueden identificar (aprox.) y eliminar del problema

# SVM en R

- La implementación más popular de SMO es **LIBSVM**
- Soporta SVMs:
  1. Lineales y no lineales
  2. Clasificación y regresión
  3. Múltiples kernels
  4. Multiclase (one vs one)
- Implementado en C++, con interfaces para múltiples lenguajes
- La interfaz de R está en el paquete **e1071**
- Existen otros algoritmos específicos, por ej. **LIBLINEAR** para SVMs lineales
- La interfaz de R es **LiblineaR**