Tema 4. Introducción a las Máquinas de Vectores Soporte

Pedro Antonio Gutiérrez
Grupo de investigación AYRNA
Universidad de Córdoba
Correo electrónico: pagutierrez@uco.es

17 de septiembre de 2024

Supervisado vs No Supervisado

- A menudo, consideramos *aprendizaje supervisado…*
  - …donde el entrenamiento depende de *datos etiquetados*
  - Los datos de entrenamiento deben ser preprocesados.
- Por el contrario, en *aprendizaje no supervisado*…
  - …usamos datos no etiquetados
  - No necesitamos etiquetas para entrenar.
- También tenemos algoritmos *semi-supervisados*:
  - Supervisados, pero ¿no demasiado?

Aprendizaje supervisado

- Las SVM son uno de los algoritmos supervisados más populares
- Están diseñados para problemas binarios:
  - Es decir, dos clases, por ejemplo *spam* vs *ham*
- La SVM se puede generalizar a múltiples clases:
  - Tal y como pasa con el Análisis Discriminante Lineal o muchos otros algoritmos.

Máquina de vectores de soporte

- Según otro autor...
  - "Las SVM son un raro ejemplo de una metodología en la que se combinan la intuición geométrica, las matemáticas elegantes, las garantías teóricas y los algoritmos prácticos"
- Tenemos algo que decir sobre cada aspecto de esto...
  - Geometría, matemáticas, teoría y algoritmos.

Máquina de vectores de soporte

- SVM basado en cuatro GRANDES ideas:

1. Hiperplano separador
2. Maximizar el “margen”

   Maximizar la separación mínima entre clases
3. Trabajar en un espacio de dimensiones superiores.

   Más “espacio”, por lo que es más fácil separarlos
4. Truco del kernel

   Esto está íntimamente relacionado con 3

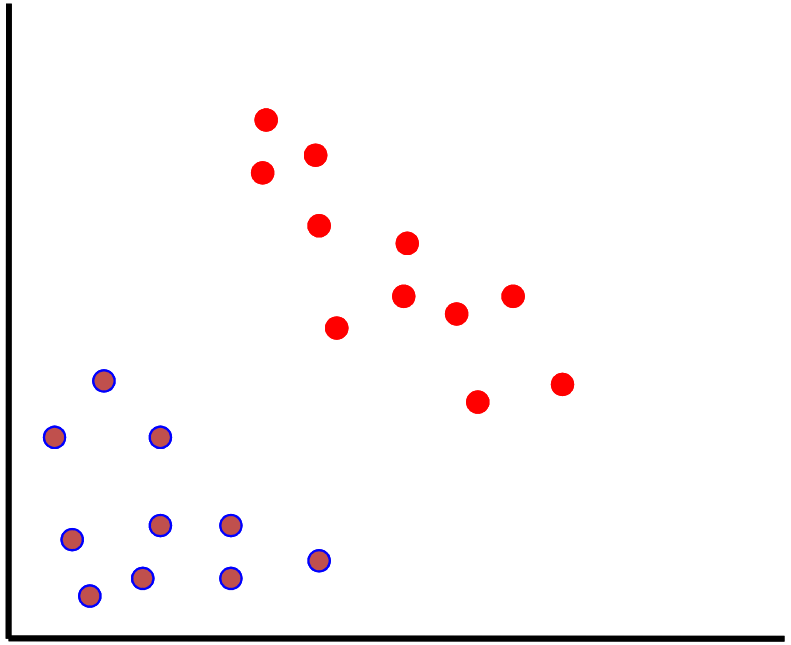
- Tanto 1 como 2 son bastante intuitivos

SVM

- Las SVM se pueden aplicar a cualquier dato de entrenamiento
  - Tener en cuenta que SVM produce un ***clasificador*** …
  - … no sólo una función de puntuación
- Con regresión logística, por ejemplo
  - Primero entrenamos un modelo...
  - …luego genera puntuaciones y establecemos un umbral para clasificar
- SVM hace clasificación directamente
  - Se salta el paso intermedio

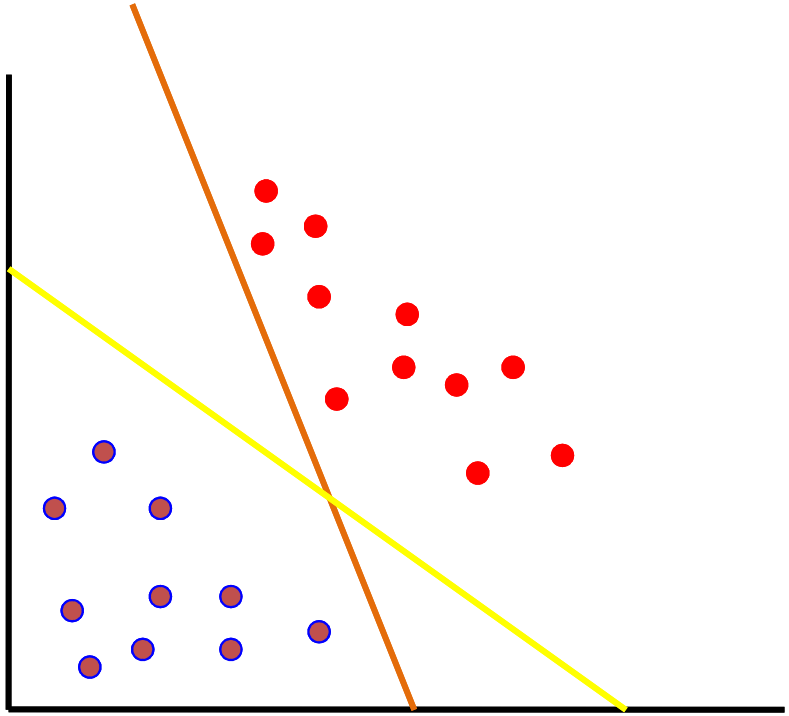
Separando clases

- Considere los datos etiquetados siguientes:
- Clasificador binario
  - La clase roja es tipo “1”
  - La clase azul es “-1”
  - (x,y) son características
- ¿Cómo separar?
  - Usaremos un “hiperplano separador”...
  - ...una línea en este caso



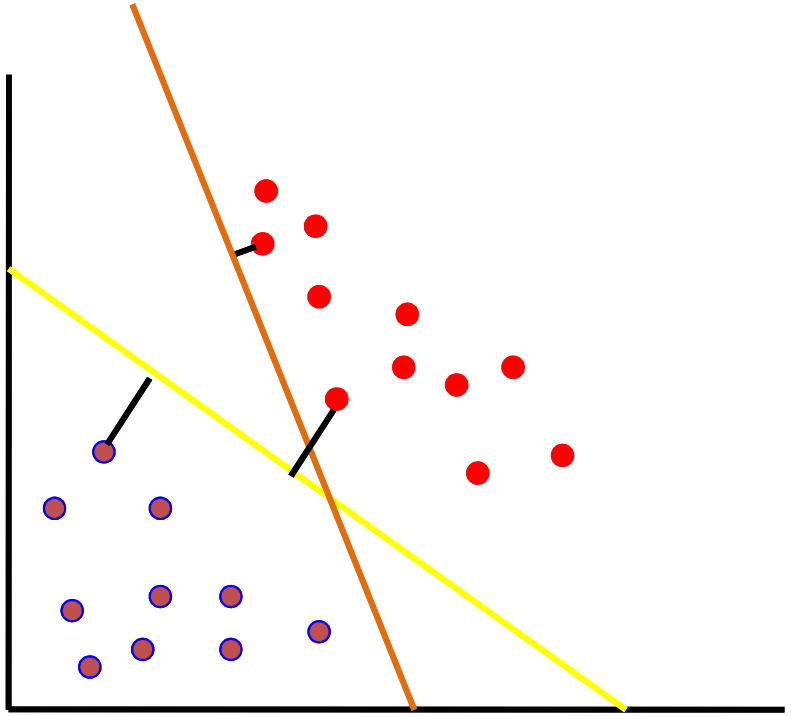
Separando hiperplanos

- Considere los datos etiquetados:
  - En este caso, es fácil
- Dibuja un hiperplano para separar puntos
  - Clasificar nuevos datos basándose en el hiperplano separador
  - ¿Qué hiperplano es mejor? ¿O cuál es el mejor de todos? ¿Por qué?



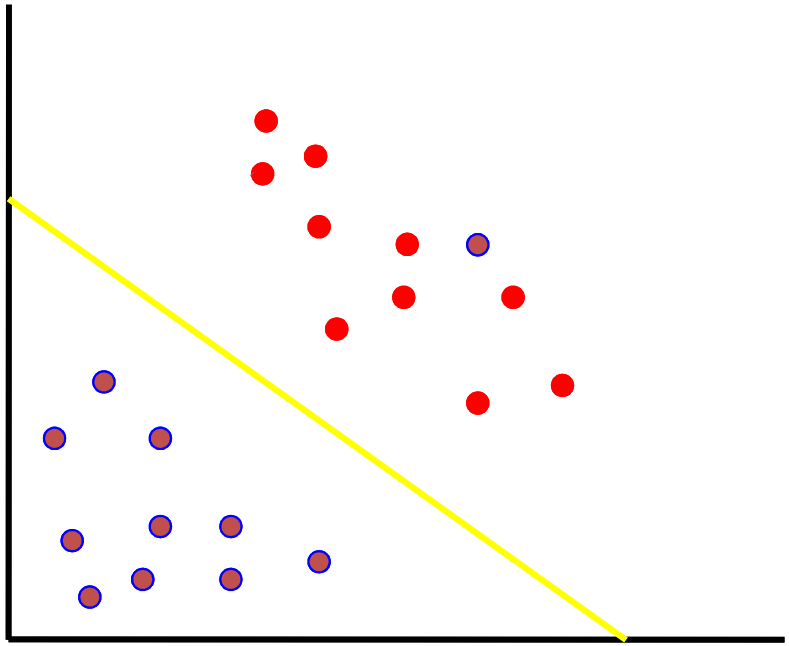
Maximizar margen

- ***Margen*** es la distancia mínima hasta la clasificación errónea
- Maximizar el margen
  - El hiperplano amarillo es mejor que el morado.
- Parece una buena idea
  - Pero puede que no sea posible.
  - Vea la siguiente diapositiva...



Separando… NO

- ¿Qué pasa con este caso?
- La línea amarilla no es una opción.
  - ¿Por qué no?
  - Ya no pueden ser "separadores"
- ¿Qué hacer?
  - ¿Permitir algunos errores?
  - Por ejemplo, el hiperplano no necesita separar todo



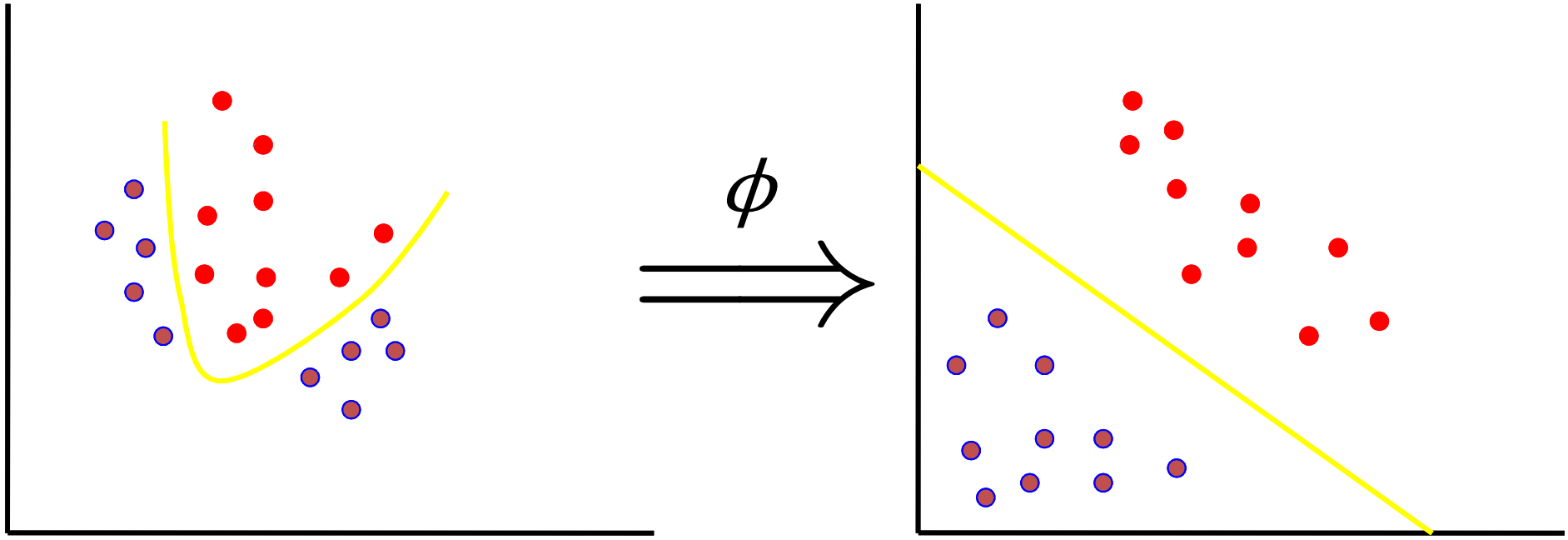
Margen suave

- Lo ideal es un gran margen y no cometer errores.
- Pero permitir algunas clasificaciones erróneas podría aumentar mucho el margen.
  - Es decir, relajar el requisito de “separación”
- ¿Cuántos errores permitir?
  - Este será un parámetro definido por el usuario.
  - ¿Equilibrio? Errores Vs margen mayor
  - En la práctica, prueba y error para determinar el equilibrio óptimo

Espacio de características

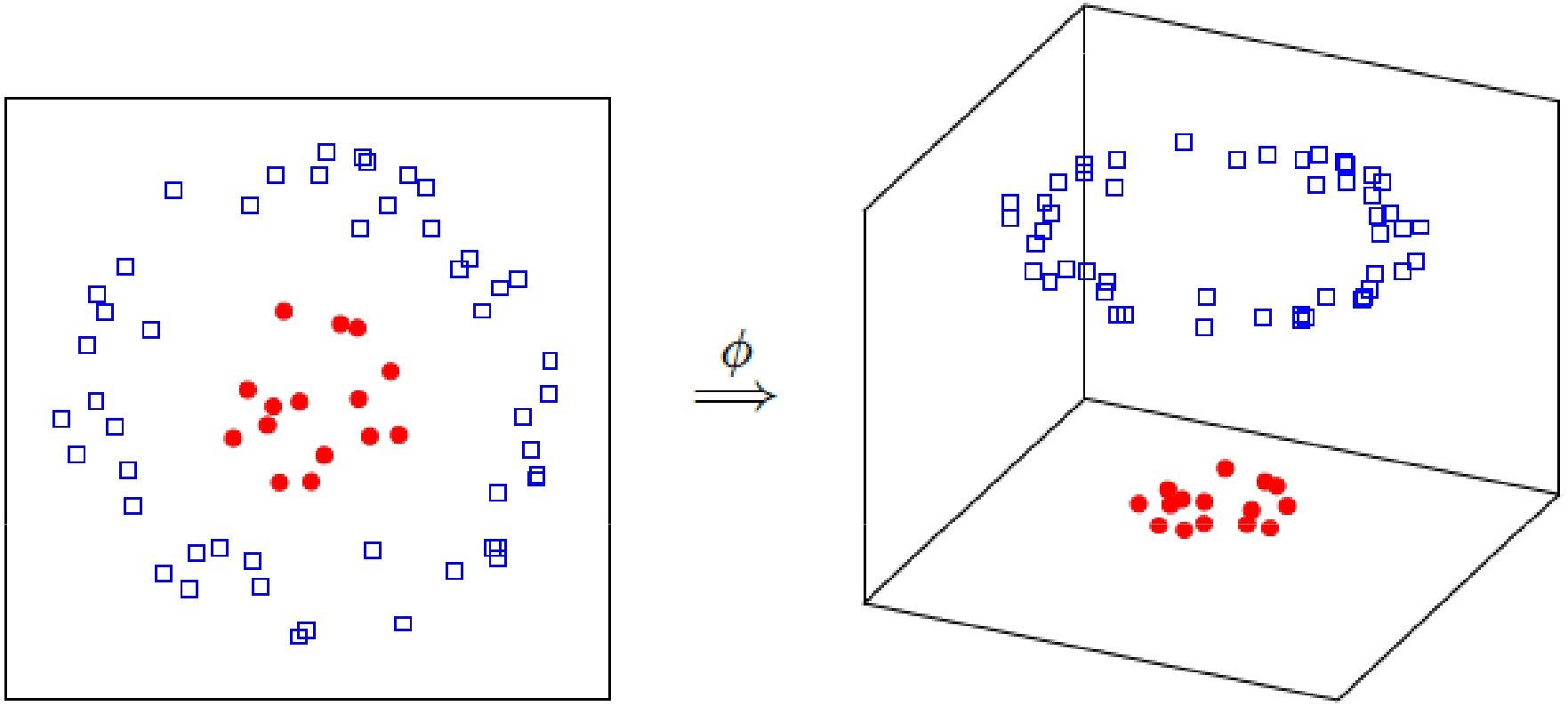
- Transformar datos hacia un “espacio de características”
  - Espacio de características generalmente en una dimensión superior.
  - Pero ¿qué pasa con la maldición de la dimensionalidad?
- **P**: ¿Por qué ***aumentar la*** dimensionalidad?
- **R**: más fácil de separar en el espacio de características.
- El objetivo es hacer que los datos sean “linealmente separables”
  - Queremos separar clases con un hiperplano
  - Pero no pagar el precio por la alta dimensionalidad.

- ¿Por qué transformar?
  - A veces lo no lineal puede volverse lineal...



Espacio de características en dimensión superior

- Un ejemplo de lo que puede pasar al transformar a una dimensión superior



Por favor, ver script Python en Moodle

Espacio de características

- Por lo general, mayor dimensionalidad es peor.
  - Desde el punto de vista de la complejidad computacional…
  - ...y desde un punto de vista de significación estadística
- Pero el espacio de características de mayor dimensión puede hacer que los datos sean linealmente separables
- ¿Podemos tener todo?
  - ¿linealmente separable **y** fácil de calcular?
- ¡Sí! Gracias al **truco del kernel o truco del núcleo**.

Truco del kernel

- Nos permite trabajar en el ***espacio de entrada***
  - Con resultados mapeados al espacio de características
  - Ningún cálculo realizado explícitamente en el espacio de características.
- Cálculos en el espacio de entrada.
  - Dimensión más baja, por lo que el cálculo es más fácil.
- Pero las cosas “suceden” en el espacio de características.
  - Mayor dimensión, por lo que es más fácil de separar
- ¡Un truco muy, muy genial!

Truco del kernel

- Desafortunadamente, para entender el truco del kernel, debemos profundizar un poco (¿quizás mucho?)
  - Aclarará todos los aspectos de las SVM
- No cubriremos todos los detalles aquí.
- Lo suficiente para tener una idea.
- Necesitaremos **multiplicadores de Lagrange**
  - Pero primero, ***optimización restringida***

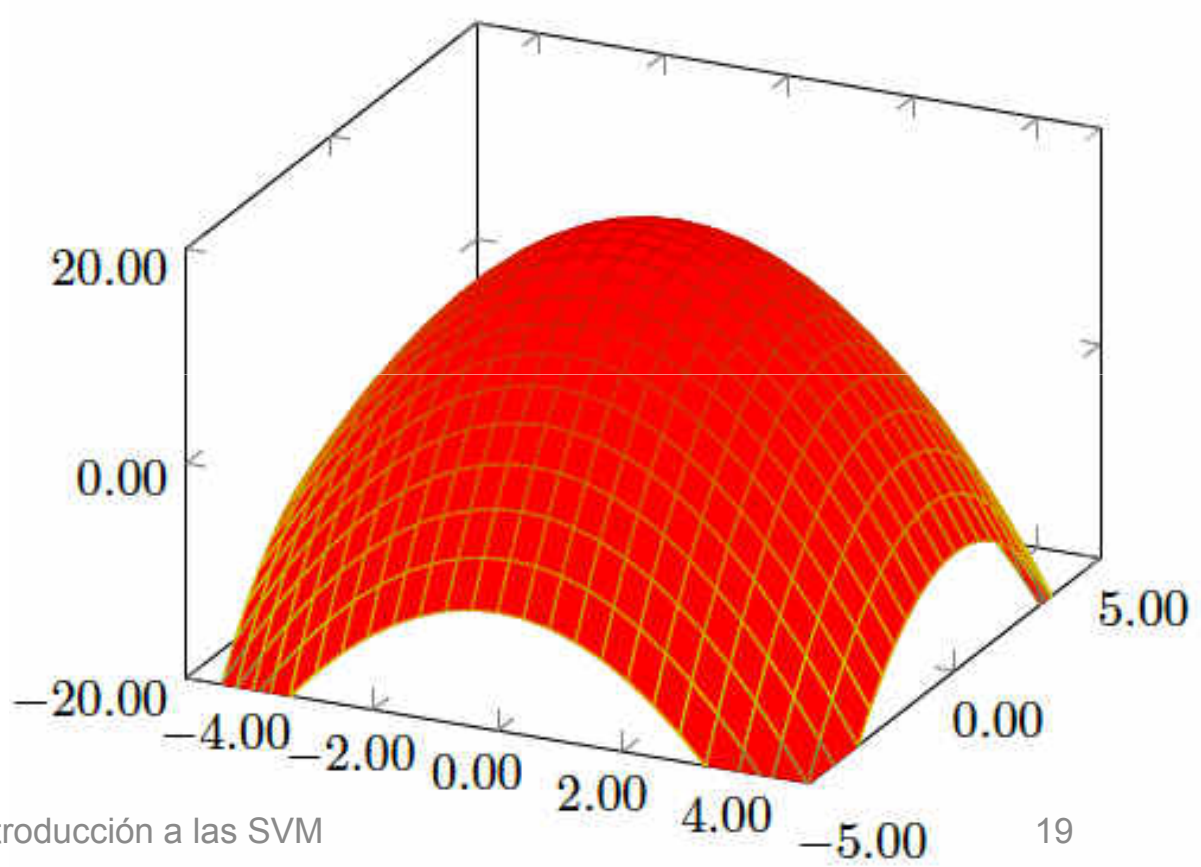
Optimización restringida

- Problema general (en 2 variables)
  - Maximizar: $f(x,y)$
  - Sujeto a: $g(x,y) = c$
- **Función objetivo** f y **restricción** g
- Por ejemplo,
  - Maximizar: $f(x,y) = 16 - (x^2 + y^2)$
  - Sujeto a: $2x - y = 4$
- Veremos este ejemplo en detalle.

Ejemplo específico

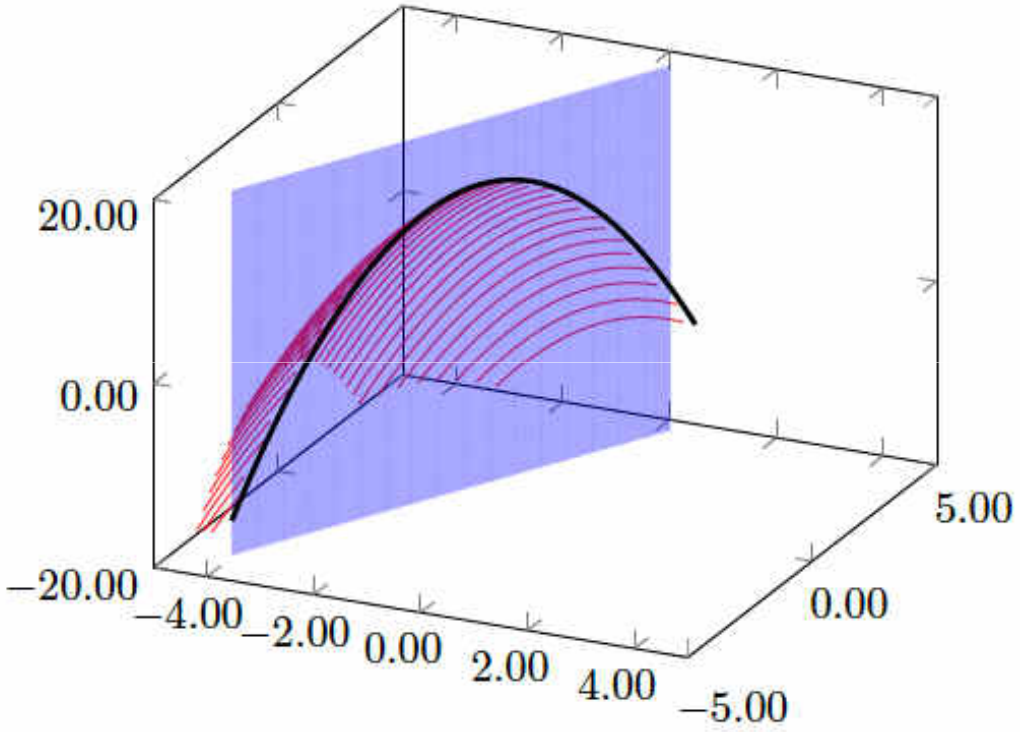
- Maximizar: $f(x,y) = 16 - (x^2 + y^2)$
- Sujeto a: $2x - y = 4$

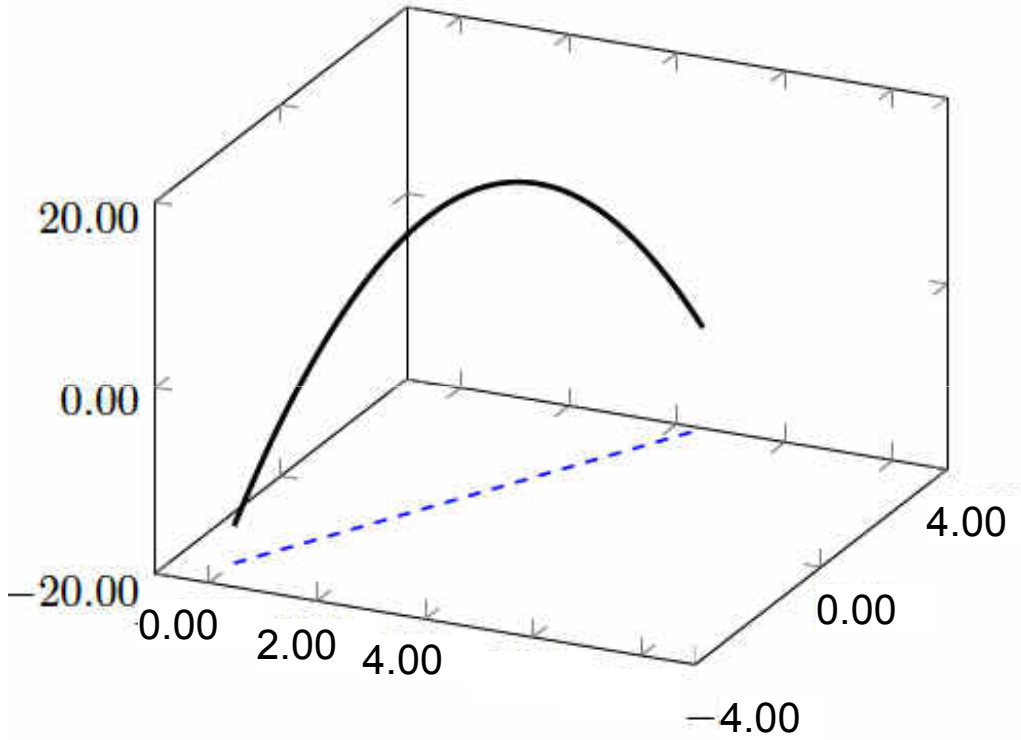
- Gráfico de $f(x,y)$



Intersección

- Intersección de f(x,y) y 2x–y=4
- ¿Cuál es la solución al problema?





Optimización restringida

- Este ejemplo parece fácil
- ¿Pero cómo solucionarlo en general?
- Recuerda, el caso general (en 2 variables) es:
  - Maximizar: $f(x,y)$
  - Sujeto a: $g(x,y) = c$
- ¿Cómo "simplificar"?
  - ¡Combinamos la función objetivo $f(x,y)$ y la restricción $g(x,y) = c$ en una ecuación!

Solución propuesta

- Definimos $J(x,y) = f(x,y) + I(x,y)$
  - Donde $I(x,y)$ es 0 siempre que $g(x,y) = c$ (la restricción se cumple) y $-\infty$ en caso contrario
- Recordemos el problema general...
  - Maximizar: $f(x,y)$
  - Sujeto a: $g(x,y) = c$
- La solución viene dada por $\max J(x,y)$
  - Aquí, $\max$ está sobre todo (x,y)

Solución propuesta

- Sabemos cómo resolver problemas de maximización (sin restricciones) usando cálculo.
- Entonces, usaremos cálculo para resolver el problema max $J(x,y)$, ¿verdad?
- ¡EQUIVOCADO!
- La función $J(x,y)$ es ***no*** en absoluto "agradable"
  - Esta función no es diferenciable.
  - ¡Ni siquiera es continuo!
  - ¿Por qué? → El problema es con $I(x,y)$

Solución propuesta

- Nuevamente, sea $J(x,y) = f(x,y) + I(x,y)$
  - Donde $I(x,y)$ es 0 siempre que $g(x,y) = c$ y $-\infty$ de lo contrario
- Entonces $\max J(x,y)$ es la solución al problema
  - Esto es **bueno**
- Pero no podemos resolver este problema máximo.
  - Esto es muy **malo**
- Que hacer???

Solución nueva y mejorada

- Reemplacemos $I(x,y)$ con una función "agradable"
- ¿Cuáles son las funciones más bonitas de todas?
  - Función lineal (en la restricción)
- Para maximizar $f(x,y)$, sujeto a $g(x,y) = c$ primero definimos el **Lagrangiano**

  $L(x,y,\lambda) = f(x,y) + \lambda (g(x,y) - c)$, con $\lambda>0$
- $g(x,y) - c \rightarrow 0$ si se cumple la restricción
- Buena función en λ , por lo que se aplica el cálculo
  - Pero no es sólo un problema de max (siguiente diapositiva...)

Solución nueva y mejorada

- Maximizar: $f(x,y)$, sujeto a: $g(x,y) = c$
- De nuevo, el lagrangiano es

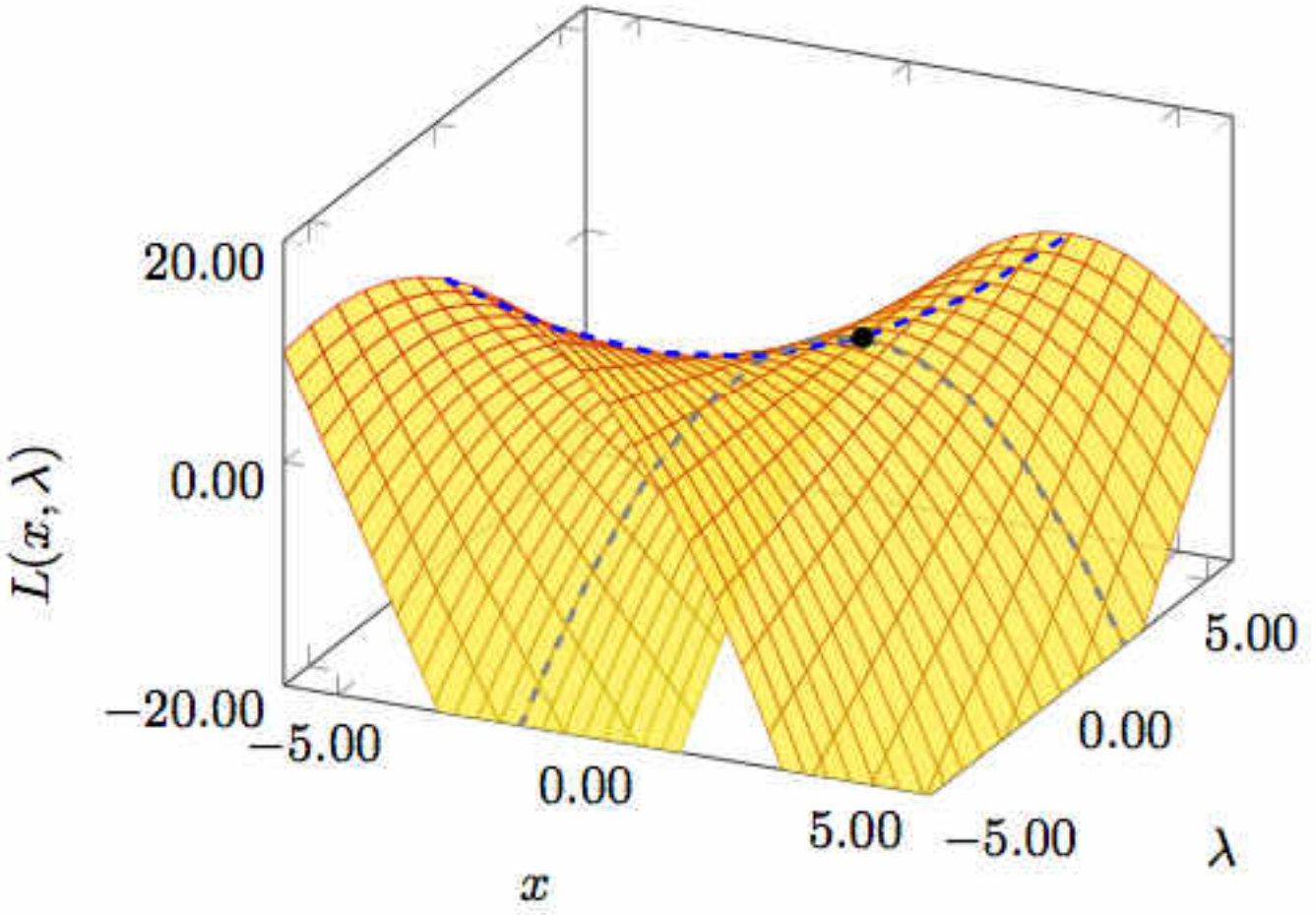
  $L(x,y,\lambda) = f(x,y) + \lambda (g(x,y) - c)$
- Observe que $\min L(x,y,\lambda) = J(x,y)$
  - Donde min ha terminado λ
- Recuerde que $\max J(x,y)$ resuelve el problema
- $\max_{x,y} \min_{\lambda} L(x,y,\lambda)$ también resuelve el problema
- ¿Ventaja de esta forma de problema?

Multiplicadores de Lagrange

- Maximizar: $f(x,y)$, sujeto a: $g(x,y) = c$
- Lagrangiano: $L(x,y,\lambda)=f(x,y)+\lambda\,(g(x,y)-c)$
- Solución dada por $\max_{x,y} \min_{\lambda} L(x,y,\lambda)$
  - Tenga en cuenta que esto es max wrt (x,y) variables...
  - ...y min es el parámetro λ
- Entonces, la solución está en un "punto de silla" con respecto a la función general, es decir, (x,y,λ) variables
  - Por definición de punto de silla

Puntos de silla o *saddle point*

- Gráfica de $L(x,\lambda) = 4 - x^2 + \lambda(x - 1)$
  - Nota , $f(x) = 4 - x^2$ y la restricción es $x = 1$



Solución nueva y mejorada

- Maximizar: $f(x,y)$, sujeto a: $g(x,y) = c$
- El lagrangiano es $L(x,y,\lambda)=f(x,y)+\lambda(g(x,y)-c)$
- Resuelto por $\max_{x,y} \min_{\lambda} L(x,y,\lambda)$
- ¡Cálculo al rescate!

$$\frac{\partial L}{\partial x} = \frac{\partial f(x,y)}{\partial x} + \lambda \frac{\partial g(x,y)}{\partial x} = 0 \qquad \frac{\partial L}{\partial y} = \frac{\partial f(x,y)}{\partial y} + \lambda \frac{\partial g(x,y)}{\partial y} = 0$$

- Y $\frac{\partial L}{\partial \lambda} = g(x,y) - c = 0$ lo que implica $g(x,y) = c$
- Langrangiano: optimización restringida convertida en optimización sin restricciones

Más, más, más

- El Lagrangiano se puede generalizar a más variables y/o más restricciones

$$L(x_1, x_2, \ldots, x_n, \lambda_1, \lambda_2, \ldots, \lambda_m) = f(x_1, x_2, \ldots, x_n) + \sum_{i=1}^{m} \lambda_i (g_i(x_1, x_2, \ldots, x_n) - c_i)$$

- O, más sucintamente

$$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i (g_i(x) - c_i)$$

  - Donde $x=(x_1, x_2, \ldots, x_n)$ y $\lambda=(\lambda_1, \lambda_2, \ldots, \lambda_m)$

Otro ejemplo

- Muchos buenos ejemplos geométricos.
- Primero, hacemos un ejemplo no geométrico.
- Considere la distribución de probabilidad discreta en n puntos: p_1, p_2, p_3, …, p_n
- ¿Qué distribución tiene entropía máxima?
  - Queremos maximizar la función de entropía.
  - Sujeto a la restricción de que p_j forme una distribución de probabilidad

Maximizar la entropía

- Entropía de Shannon: $\Sigma\ p_j \log_2 p_j$
- Tenemos una distribución de probabilidad, por lo que:
  - Se cumple $0 \leq p_j \leq 1$ para j y $\Sigma p_j = 1$
- Resolveremos este problema simplificado:
  - Maximizar: $f(p_1, ..., p_n) = \Sigma\ p_j \log_2 p_j$
  - Sujeto a restricción: $\Sigma\ p_j = 1$
- ¿Cómo deberíamos solucionar esto?
  - ¿Realmente tienes que preguntar?

Ejemplo de entropía

- Recuerde $L(x,y,\lambda) = f(x,y) + \lambda\,(g(x,y) - c)$
- Definición del problema
  - Maximizar $f(p_1, .., p_n) = \Sigma\, p_j \log_2 p_j$
  - Sujeto a restricción $\Sigma p_j = 1$
- En este caso, el lagrangiano es

  $L(p_1, \ldots, p_n, \lambda) = \Sigma\, p_j \log_2 p_j + \lambda(\Sigma p_j - 1)$
- Calcular las derivadas parciales con respecto a cada p_j y la derivada parcial con respecto a λ

Ejemplo de entropía

- $L(x, y, \lambda) = \Sigma\, p_j \log_2 p_j + \lambda(\Sigma\, p_j - 1)$
- Derivadas parciales con respecto a cualquier p_j:

  $\log_2 p_j + 1/\ln(2) + \lambda = 0$ Ec. (1)
- Y con respecto a λ produce la restricción:

  $\Sigma p_j - 1 = 0$ o $\Sigma p_j = 1$ Ec. (2)
- La ecuación (1) implica que todos los p_j son iguales
- Con la ecuación (2), todos $p_j = 1/n$
- ¿Conclusión?

Notación

- Sea $x=(x_1, x_2, \ldots, x_n)$ y $\lambda=(\lambda_1, \lambda_2, \ldots, \lambda_m)$
- Nuevamente escribimos lagrangiano como

  $L(x,\lambda) = f(x) + \Sigma\, \lambda_i\, (g_i(x) - c_i)$
- Nota: L es una función de $n+m$ variables
- Puede ver el problema como...
  - Las restricciones g_i definen una región factible
  - Maximizar la función objetivo f sobre esta región factible

Dualidad lagrangiana

- Para multiplicadores de Lagrange...
- **Problema primario:** problema original, reformulado como: $\max_{(x,y)} \min_{\lambda} L(x,y,\lambda)$
  - Donde max sobre (x,y) y min sobre λ
- **Problema dual:** $\min_{\lambda} \max_{(x,y)} L(x,y,\lambda)$
  - Como arriba, max sobre (x,y) y min sobre λ
- Afirmamos que es fácil demostrar

  mín máx $L(x,y,\lambda) \geq$ máx mín $L(x,y,\lambda)$
- ¿Por qué es esto cierto? No lo vamos a demostrar

Problema dual

- El problema dual proporciona un límite superior del problema primal.
  - mín máx $L(x,y,\lambda) \geq$ máx mín $L(x,y,\lambda)$
  - Es decir, solución dual ≥ solución primaria
- Pero es incluso mejor que eso
- Para el Lagrangiano, la ***igualdad*** es verdadera
- ¿Por qué?
  - Porque la función lagrangiana es convexa

Problema primal

- Maximizar: $f(x,y) = 16 - (x^2 + y^2)$
- Sujeto a: $2x - y = 4$
- Entonces $L(x,y,\lambda) = 16 - (x^2 + y^2) + \lambda(2x - y - 4)$
- Calcular derivadas parciales...

  $dL/dx = -2x + 2\lambda = 0$

  $dL/dy = -2y - \lambda = 0$

  $dL/d\lambda = 2x - y - 4 = 0$
- Resultado: $(x,y,\lambda) = (8/5,-4/5,8/5)$
  - Lo que produce max de $f(x,y) = 64/5$

Problema dual

- Maximizar: $f(x,y) = 16 - (x^2 + y^2)$
- Sujeto a: $2x - y = 4$
- Entonces $L(x,y,\lambda) = 16-(x^2+y^2)+\lambda(2x-y-4)$
- Recuerde que el problema dual es

  $\text{mín máx } L(x,y,\lambda)$

  - Donde max es sobre (x,y), min es sobre λ
- ¿Cómo podemos solucionar esto?

Problema dual

- Problema dual: $\min_{\lambda} \max_{(x,y)} L(x,y,\lambda)$
- Entonces, primero podemos tomar max de L sobre (x,y)
  - Entonces nos queda la función L **solo en λ**
- Para resolver el problema, buscamos $\min_{\lambda} L(\lambda)$
- En la siguiente diapositiva, ilustramos esto para

  $L(x,y,\lambda) = 16 - (x^2 + y^2) + \lambda(2x - y - 4)$

  - Mismo ejemplo considerado anteriormente

Problema dual

- Dado

  $L(x,y,\lambda) = 16 - (x^2 + y^2) + \lambda(2x - y - 4)$

- Maximizar sobre (x,y) calculando

  $dL/dx = -2x + 2\lambda = 0$

  $dL/dy = -2y - \lambda = 0$

- Lo que implica $x = \lambda$ y $y = -\lambda/2$

- Sustitúyalos en L para obtener

  $L(\lambda) = 5/4\ \lambda^2 + 4\lambda + 16$

Problema dual

- Problema original
  - Maximizar: $f(x,y) = 16 - (x^2 + y^2)$
  - Sujeto a: $2x - y = 4$
- La solución se puede encontrar minimizando

  $L(\lambda) = 5/4\ \lambda^2 + 4\lambda + 16$
- Entonces $L'(\lambda) = 5/2\ \lambda + 4 = 0$, lo que da $\lambda = -8/5$ y $(x,y) = (-8/5,4/5)$
- ¡La misma solución que el problema primal!

Resumen del problema dual

- Maximizar L para encontrar (x,y) en términos de λ
- Luego rescribir L en función de λ únicamente
- Finalmente, minimizar $L(\lambda)$ para resolver el problema.
- Pero, ¿por qué tanto alboroto?
  - El problema dual nos permite escribir el problema SVM de una manera mucho más práctica.
- En SVM, consideraremos el dual de $L(\lambda)$

Multiplicadores de Lagrange y SVM

- Los multiplicadores de Lagrange son realmente geniales.
  - ¿Pero qué tiene esto que ver con SVM?
- Puede ver el cálculo del margen (suave) como un problema de optimización restringido
  - De esta forma, el truco del kernel queda más claro.
- Podemos matar 2 pájaros de 1 tiro.
  - Hacer que el cálculo del margen sea más claro
  - Dejar el truco del kernel perfectamente claro

Configuración del problema

- Sean $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ puntos de entrenamiento (vectores)
  - Supongamos que cada $\mathbf{x}_i = (x_i, y_i)$ es un punto en el plano.
  - En general, podría ser una dimensión más alta.
- Sean $z_1, z_2, \ldots, z_n$ etiquetas de clase correspondientes, donde cada $z_i \in \{-1, 1\}$
  - Por ejemplo, $z_i = 1$ si se clasifica como tipo "rojo"
  - y $z_i = -1$ si se clasifica como tipo "púrpura"
- Tenga en cuenta que esta es una clasificación binaria.

Perspectiva

- Tendremos las siguientes posibilidades:
  - SVM lineal:
    - Caso separable (margen estricto).
    - Caso no separable (margen suave).
  - SVM no lineal (basado en kernel):
    - Caso separable (margen estricto).
    - Caso no separable (margen suave).
- Para cada problema, tendremos una formulación del primal y una formulación del dual .

Perspectiva geométrica

Los parámetros w1, w2 y b se configurarán de tal forma que:

- Ecuación de la línea amarilla (*H*):

$$w_1 x + w_2 y + b = 0$$

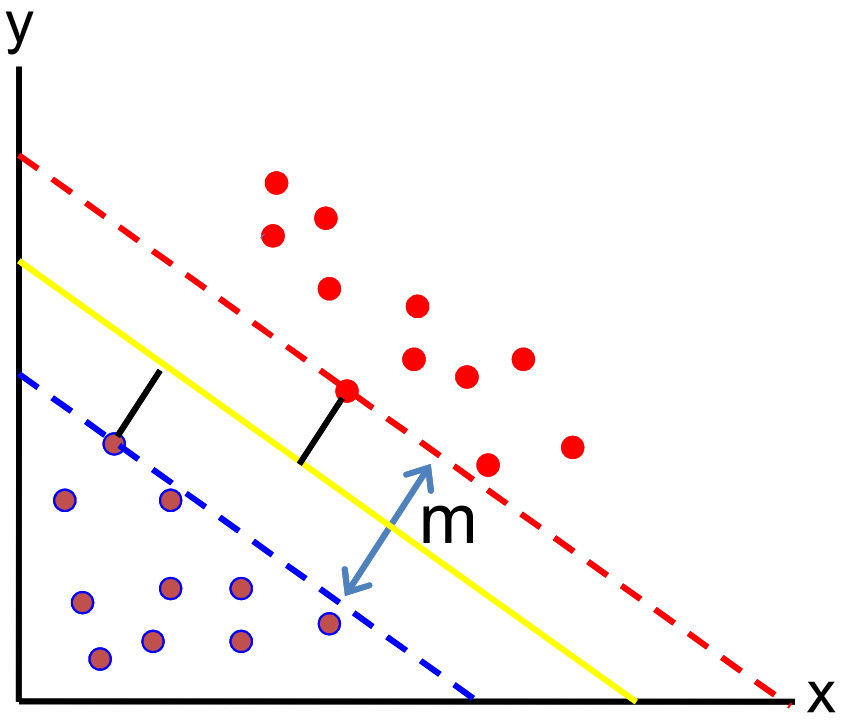
- Ecuación de la línea roja (H^+):

$$w_1 x + w_2 y + b = 1$$

- Ecuación de la línea azul (H^-):

$$w_1 x + w_2 y + b = -1$$

- Margen *m* es la longitud de la línea gris



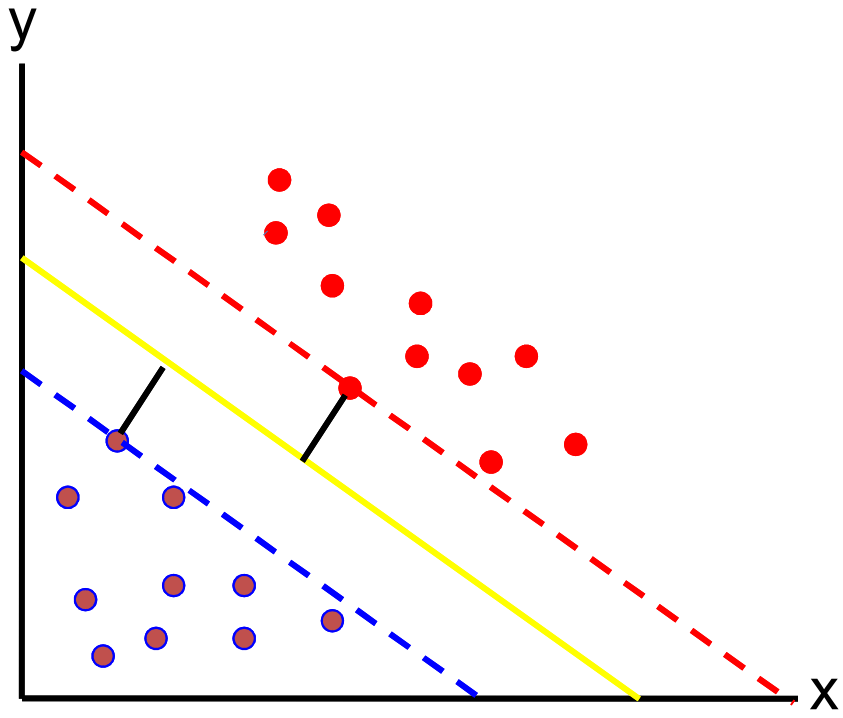
- Cualquier punto rojo **debe** satisfacer

$$w_1 x + w_2 y + b \geq 1$$

- Cualquier punto violeta **debe** satisfacer

$$w_1 x + w_2 y + b \leq -1$$

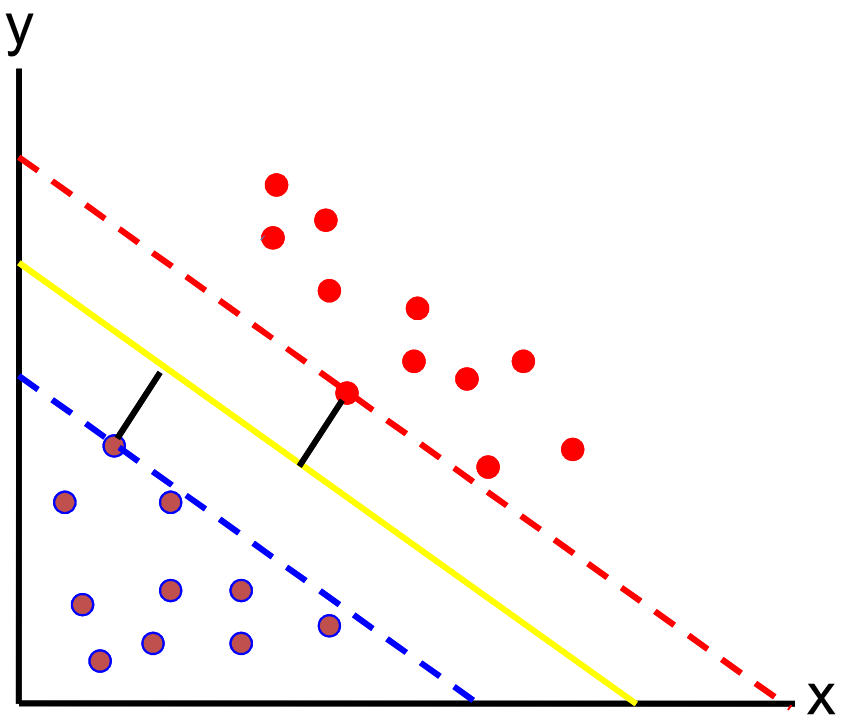
- Queremos que ***todas las*** desigualdades sean válidas después del entrenamiento



Puntuación

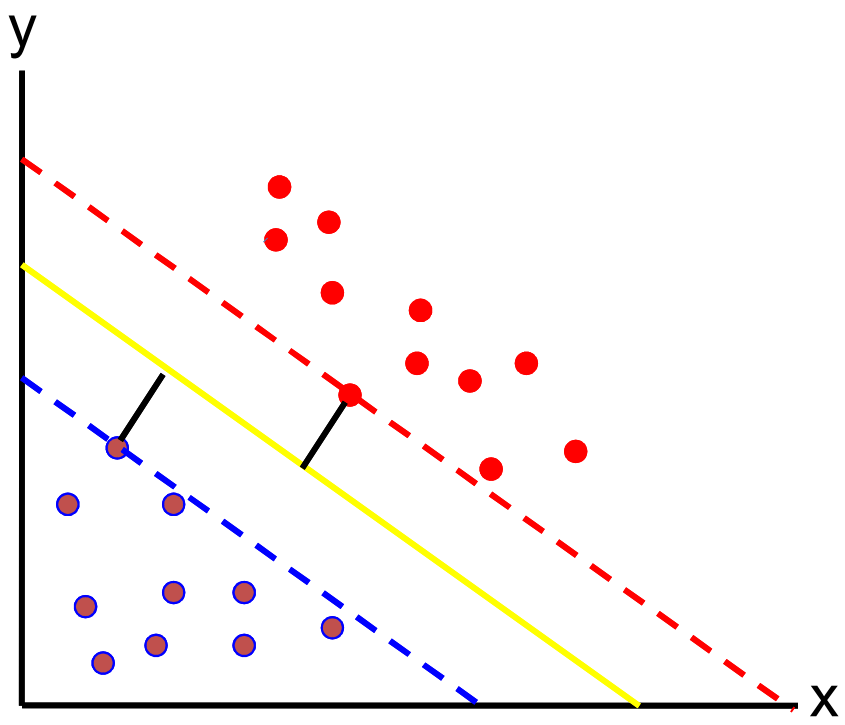
- Una vez tengamos las líneas…
- Dado un nuevo dato $\mathbf{x} = (x, y)$ para clasificar
  - "Rojo" siempre que
    $$w_1 x + w_2 y + b > 0$$
  - "Púrpura" siempre que
    $$w_1 x + w_2 y + b < 0$$
- Fase de puntuación.

  (fase de test)

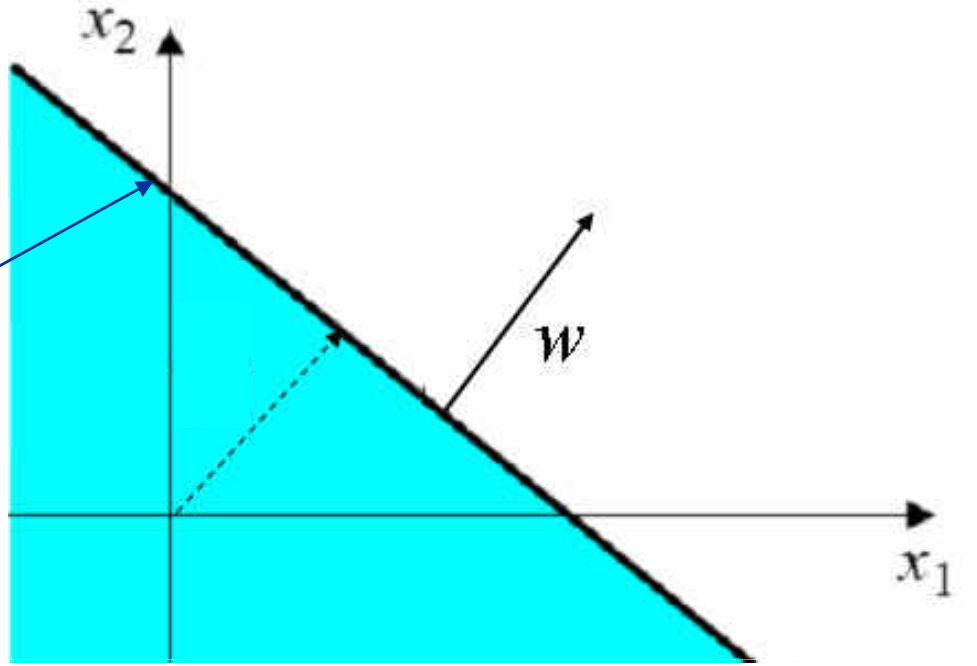


¿Cómo entrenar?

- La verdadera pregunta es...
- ¿Cómo encontrar la ecuación de la recta amarilla?
  - Dado $\mathbf{x}_i$ y z_i
  - $\mathbf{x}_i$ son puntos en el plano...
  - ...y z_i son las etiquetas de clase
- Encontrar la línea amarilla es la fase de entrenamiento de SVM



Hiperplano separador

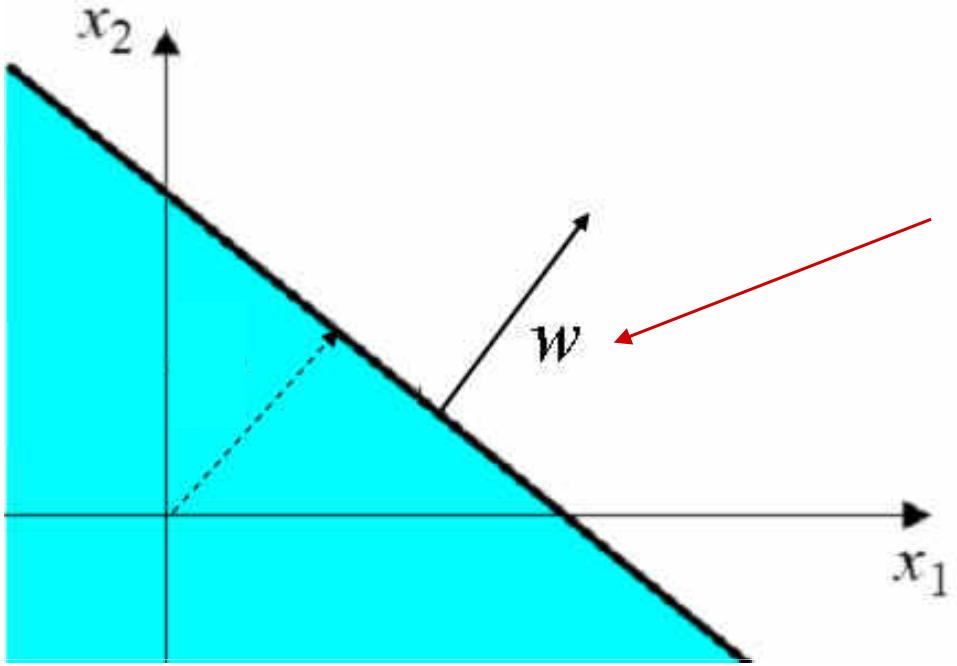


hiperplano

Hiperplano separador definido por (**w**,*b*)

$$\langle \mathbf{w}, \mathbf{x} \rangle + b = 0, \text{ donde } \langle -, - \rangle \text{ es el producto escalar}$$

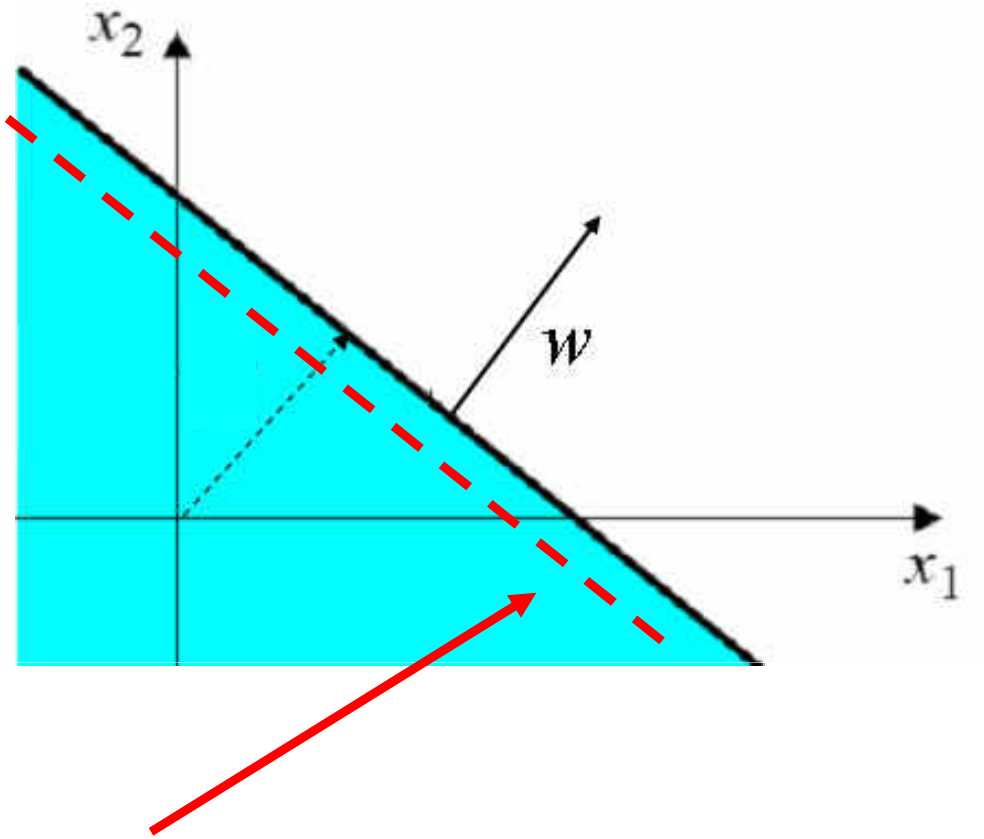
Hiperplano separador



w define una dirección perpendicular al hiperplano

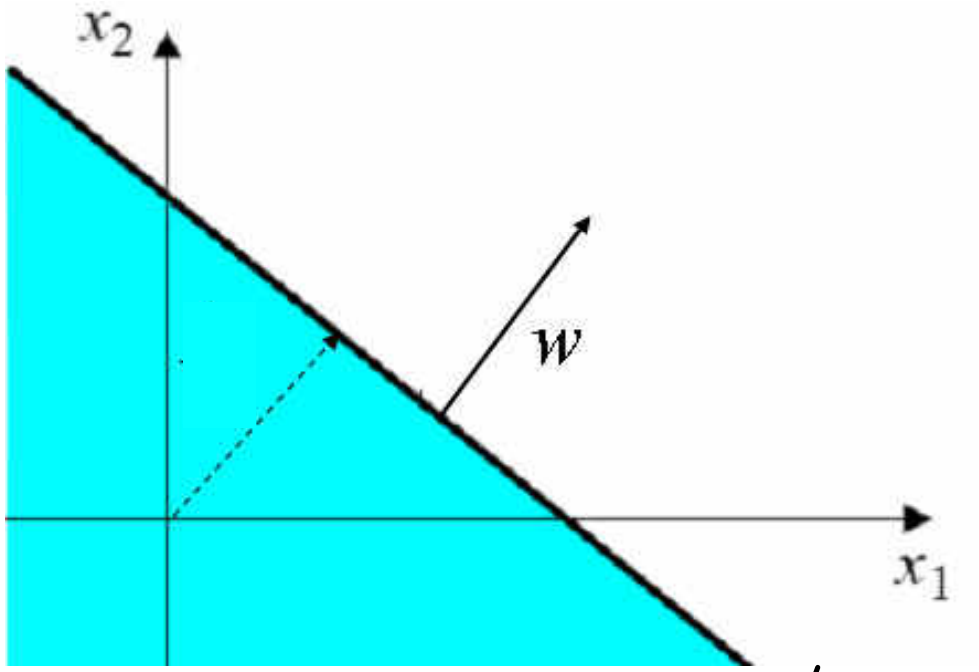
Hiperplano de separador definido por (**w** , *b*)

Hiperplano separador



Número de parámetros libres *n* +1, donde *n* es el número de características de entrada (en este caso, 3 parámetros para ajustar)

Distancia perpendicular desde x_i



Distancia del punto $\mathbf{x}_i$ a la recta $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$

$$\frac{\left| \langle \mathbf{w}, \mathbf{x}_i \rangle + b \right|}{\|\mathbf{w}\|}$$

https://brilliant.org/wiki/dot-product-distance-between-point-and-a-line/

Maximizar el margen

- Distancia perpendicular desde el origen a la recta $w_1x+w_2y+b=0$:

$$\frac{|b|}{\sqrt{w_1^2+w_2^2}}$$

- Desde origen hasta línea discontinua roja:

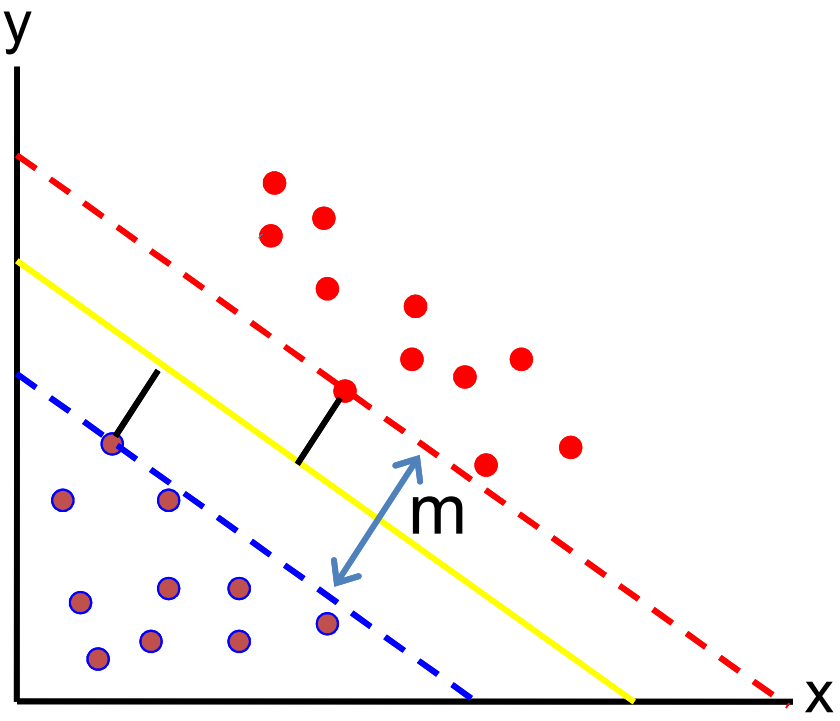
$$\frac{|1-b|}{\sqrt{w_1^2+w_2^2}}$$

- Origen a línea discontinua azul:

$$\frac{|-1-b|}{\sqrt{w_1^2+w_2^2}}$$

- Margen *m*:

$$\frac{|1-b|}{\sqrt{w_1^2+w_2^2}}-\frac{|-1-b|}{\sqrt{w_1^2+w_2^2}}=\frac{2}{\sqrt{w_1^2+w_2^2}}$$



Fase de entrenamiento

- Dados $\mathbf{x}_i$ y z_i , encontrar el mayor margen *m* que clasifica todos los puntos correctamente
- Es decir, encontrar líneas rojas y azules en la imagen.
- Recordar que la línea roja tiene la forma
  $w_1 x + w_2 y + b = 1$
- La línea azul es de la forma
  $w_1 x + w_2 y + b = -1$

❑ Y queremos margen máximo: $m = \dfrac{2}{\sqrt{w_1^2 + w_2^2}}$

- Dado que las etiquetas son $z_i \in \{-1,1\}$, se produce clasificación correcta siempre que

$$z_i(w_1 x + w_2 y + b) \geq 1,\ i \in \{1,2,\ldots,n\}$$

- Problema de entrenamiento a resolver:

$$\max\ m = \frac{2}{\sqrt{w_1^2 + w_2^2}}$$

$$\text{s.t. } z_i(w_1 x + w_2 y + b) \geq 1,\ i \in \{1,2,\ldots,n\}$$

- ¿Podemos determinar $\mathbf{w} = (w_1, w_2)$ y b ?

- El problema de la diapositiva anterior es equivalente al siguiente:

$$\min \frac{w_1^2 + w_2^2}{2}$$

$$\text{s.t. } 1\text{-}z_i(w_1 x + w_2 y + b) \leq 0,\ i \in \{1, 2, \ldots, n\}$$

- Debería empezar a resultarte familiar...

- Considerando las desigualdades como **igualdades** …

$$L(w_1,w_2,b,\lambda)= \frac{w_1^2+w_2^2}{2} + \sum_{i\in\{1,2,\ldots,n\}} \lambda_i \left(1-z_i(w_1x+w_2y+b)\right)$$

- Trabajaremos con el problema dual, de modo que comencemos calculando las derivadas cra w_i y b. Calcular

$dL/dw_1 = w_1 - \Sigma\, \lambda_i z_i x_i = 0$

$dL/dw_2 = w_2 - \Sigma\, \lambda_i z_i y_i = 0$

$dL/db = \Sigma\, \lambda_i z_i = 0$

- Las derivadas dan lugar a estas restricciones:

$$\mathbf{w} = \left(w_1, w_2\right)^{\mathrm{T}} = \sum_i \lambda_i z_i \mathbf{x}_i \qquad \sum_i \lambda_i z_i = 0$$

- Sustituir la primera en L produce

$$L(\lambda) = \sum_i \lambda_i - \frac{1}{2} \sum_i \sum_j \lambda_i \lambda_j z_i z_j \left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle$$

donde <,> es el producto escalar $\left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle = x_i x_j + y_i y_j$

- Aquí, L es sólo una función de λ
  - Todavía tenemos la segunda restricción.
  - Nota: Si encontramos λ_i entonces sabemos $\mathbf{w}$

Problema nuevo y mejorado

$$\max L(\lambda) = \sum_i \lambda_i - \frac{1}{2}\sum_i \sum_j \lambda_i \lambda_j z_i z_j \left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle$$

$$\text{s.t.} \sum_i \lambda_i z_i = 0 \text{ and } \lambda_i \geq 0, i \in \{1, \ldots, n\}$$

- La solución es exacta sólo para funciones convexas (que es el caso). Consulte el teorema de Kuhn-Tucker para obtener más información.
- ¿Por qué maximizar L(λ)? Intuitivamente…
  - El objetivo es minimizar F($\mathbf{w}$) = ($w_1^2 + w_2^2$) / 2
  - Sujeto a restricciones en la función L(λ)
  - Maximizar L(λ) encuentra los "mejores" parámetros λ
  - Y el "mejor" λ resolverá el problema de minimización
- Recuerde, este es el problema dual.

- Podemos también eliminar la restricción $\sum_i \lambda_i z_i = 0$ introduciendo un nuevo multiplicador de Lagrange, llamado α para distinguirlo de los originales:

$$\max L(\lambda, \alpha) = \sum_i \lambda_i - \frac{1}{2} \sum_i \sum_j \lambda_i \lambda_j z_i z_j \left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle - \\ -\alpha \left(\sum_i \lambda_i z_i \right)$$

$$\text{s.t. } \lambda_i \geq 0, i \in \{1, \ldots, n\}, \alpha \geq 0$$

Versión dual del problema

$$\max L(\lambda) = \sum_i \lambda_i - \frac{1}{2} \sum_i \sum_j \lambda_i \lambda_j z_i z_j \left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle$$

$$\text{s.t.} \sum_i \lambda_i z_i = 0 \text{ and } \lambda_i \geq 0, i \in \{1, \ldots, n\}$$

- De nuevo, este es el problema dual.
- Siempre puedo resolverlo (si existe solución)
  - Y encontraremos un máximo ***global***.
- ¡No hay nada mejor que eso!

Todo junto ahora: entrenamiento

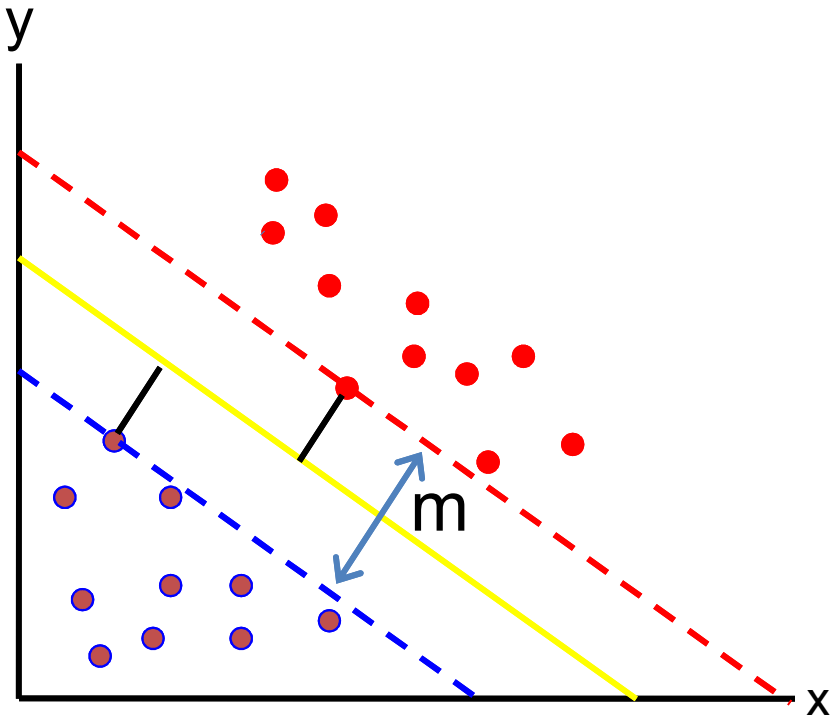
- Dados los puntos $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$
- Etiquetado cada $\mathbf{x}_i$ con $z_i \in \{-1, 1\}$
- Resuelves el problema dual (diapositiva anterior)
  - Al resolverlo se obtiene λ_i
  - Una vez conocido λ_i, calculas $\mathbf{w} = (w_1, w_2)$ y b
  - Obtienes la ecuación de la recta: $w_1 x + w_2 y + b$
- ¿Qué hemos logrado?

Todo junto ahora: puntuación

- Del entrenamiento, calculamos λ_i
  - Produce $\mathbf{w} = (w_1, w_2)$ y b en $w_1x + w_2y + b$
- Dado un nuevo punto $\mathbf{x} = (x,y)$
  - Es decir, **x** no está en el conjunto de entrenamiento
- Calcular $w_1x + w_2y + b$
  - Si es mayor que 0, clasifica **x** como “rojo” (+1)
  - De lo contrario, clasifique **x** como “púrpura” (-1)
- ¿Qué pasó en términos de imagen?

Punto de vista geométrico

- ¿Entrenamiento?
  - Encuentre la ecuación de la línea amarilla, $f(\mathbf{x})$
- ¿Puntuación de $\mathbf{x}=(x,y)$?
  - Si $f(\mathbf{x}) > 0$, entonces $\mathbf{x}$ está por encima de la línea amarilla (clasificar como roja)
  - sino $\mathbf{x}$ debajo de la línea (clasificar como morado)



Puntuación revisada

- Utilizar la línea amarilla para puntuar...
- Hay una forma alternativa (mejor)
  - Tenemos $f(\mathbf{x}) = w_1 x + w_2 y + b = \langle \mathbf{w}, \mathbf{x} \rangle + b$
  - recuerda que $\mathbf{w} = \sum_i \lambda_i z_i \mathbf{x}_i$

    → Entonces, $f(\mathbf{x}) = \left(\sum_i \lambda_i z_i \langle \mathbf{x}_i, \mathbf{x} \rangle \right) + b$
- ¿Por qué es esto mejor?
  - No es necesario calcular explícitamente **w**
  - ¿Alguna mejor razón por la que es mejor? (truco del kernel)

Vectores soporte

- Al resolver L(λ), nos damos cuente de que la mayoría de $\lambda_i = 0$
- Específicamente, $\lambda_i=0$ para todos los $\mathbf{x}_i$ para los cuales

$$z_i(w_1x + w_2y + b) > 1$$

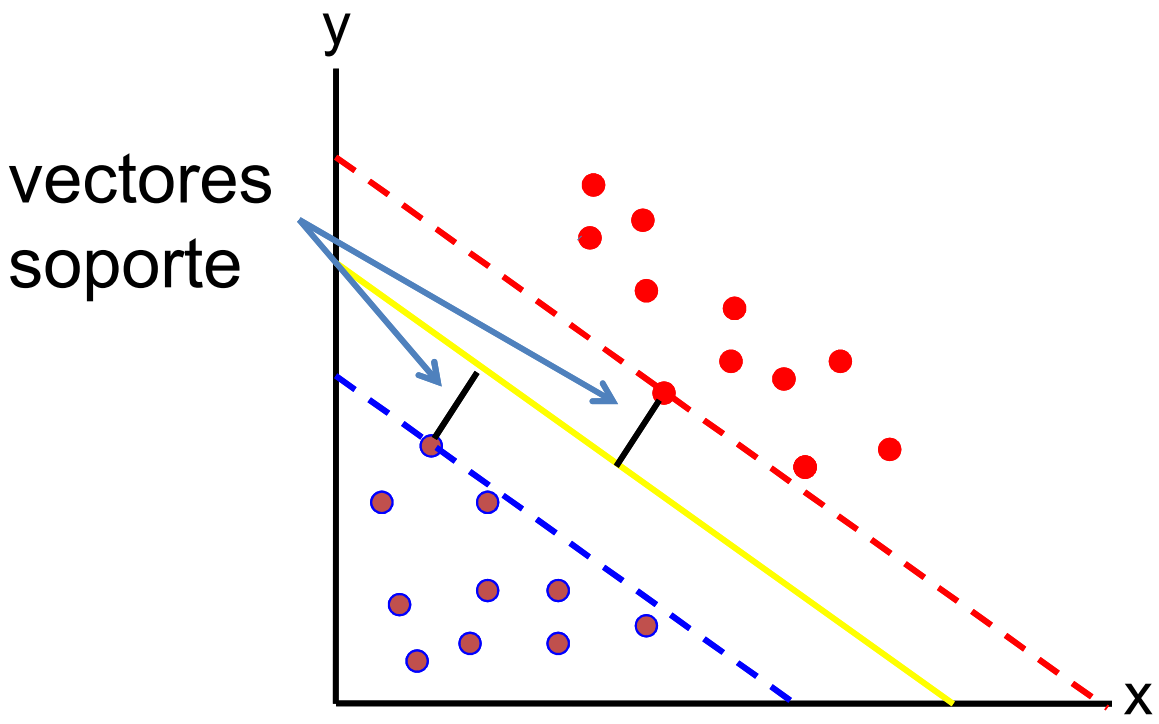
- Las únicas restriccionesque finalmente influyen son:

$$z_i(w_1x + w_2y + b) = 1$$

- Los puntos asociados a esas restricciones son los ***vectores soporte***.
  - No conocidos de antemano → el entrenamiento determina los vectores de soporte

Vectores soporte

- ¿Una imagen vale más que 1000 palabras?
- ¿Dónde están los vectores de soporte?
  - Otros vectores (puntos de entrenamiento) no influyen
  - ¿Por qué no?



Puntuación revisada nuevamente

- Puntuación **x** usando $f(\mathbf{x}) = \left(\sum_i \lambda_i z_i \langle \mathbf{x}_i, \mathbf{x} \rangle \right) + b$

- Generalmente, la mayoría de los λ_i serán *0*
- Entonces, la suma no es realmente para todo *i* $\sum$
  - En cambio, la suma es para vectores de soporte → $\sum_{i \in S}$
  - donde *S* es el conjunto de vectores de soporte ($\lambda_i \neq 0$)
- ¿Qué importancia tiene esto?
  - Normalmente, *n* es grande, |*S*| es pequeño, la puntuación es rápida.
  - Por tanto, este f(X) sin **w** es útil por más razones.

Cómo establecer el sesgo

- Resolviendo el problema dual obtenemos los λ_i
- ¿Cómo obtener el vector de proyección y el sesgo?
  - Para la proyección , usando las derivadas, sabemos que:

$$\mathbf{w} = \sum_{i \in S.} \lambda_i z_i \mathbf{x}_i$$

  - Para el sesgo , podemos usar el hecho de que el modelo tiene que predecir un +1 o un -1 para los vectores de soporte (dos opciones):

$$f(\mathbf{x}) = \left(\sum_{i \in S} \lambda_i z_i \left\langle \mathbf{x}_i, \mathbf{x} \right\rangle \right) + b$$

$$\begin{cases} b = 1 - \left(\sum_{i \in S} \lambda_i z_i \left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle \right), \text{ para un patrón } \mathbf{x}_j \text{ t.q. } z_j = 1 \text{ y } \lambda_j \neq 0 \\ b = -1 - \left(\sum_{i \in S} \lambda_i z_i \left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle \right), \text{ para un patrón } \mathbf{x}_j \text{ t.q. } z_j = -1 \text{ y } \lambda_j \neq 0 \end{cases}$$

Es posible promediar estos valores para todos los s.v. para tener una solución más estable!!

Ejemplo: solución dual con vectores soporte

Clasificación unidimensional: $g(x)= wx + w_0$

– Conjunto de entrenamiento: (-1, 1) (0, 1) de la primera clase; (2, -1) (5, -1) de la segunda clase

Primal →

$$\left.\begin{array}{l} \min\limits_{\mathbf{w}} \ \frac{1}{2}\langle \mathbf{w}, \mathbf{w} \rangle \\ s.t. \ \ z_i\left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right) \geq 1, \quad i = 1 \ldots n \end{array}\right\} \Rightarrow \begin{array}{c} -w + b \geq 1 \\ b \geq 1 \\ -2w - b \geq 1 \\ -5w - b \geq 1 \end{array}$$

Seleccionamos dos vectores soporte (0,1) y (2,-1). Entonces $x_1=0$, $x_2=2$, $z_1=1$ y $z_2=-1$

$$\left.\begin{array}{l} 0w + b = 1 \\ 2w + b = -1 \end{array}\right\} \begin{array}{l} b = 1 \\ w = -1 \end{array}$$

Ejemplo: solución dual con vectores soporte

Seleccionamos dos vectores soporte (0,1) y (2,-1). Entonces x_1=0, x_2=2, z_1=1 y z_2 =-1

Dual → $$\left.\begin{aligned} &\max_{\lambda} \sum_{i\in S}\lambda_i - \frac{1}{2}\sum_{i,j\in S}\lambda_i\lambda_j z_i z_j \left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle \\ &\text{s.t.} \quad \sum_{i\in S}\lambda_i z_i = 0, \quad \lambda_i \geq 0,\ i = 1,\ldots,n \end{aligned}\right\} \Rightarrow$$

$$\max \lambda_1 + \lambda_2 - \frac{1}{2}\left(0\lambda_1^2 + 0\lambda_1\lambda_2 + 0\lambda_2\lambda_1 + 4\lambda_2^2\right)$$

$$\text{s.t.} \quad \lambda_1 - \lambda_2 = 0,\ \lambda_1 \geq 0,\ \lambda_2 \geq 0$$

$$L = \lambda_1 + \lambda_2 - 2(\lambda_2^2) - \alpha(\lambda_1 - \lambda_2)$$

Ejemplo: solución dual con vectores soporte

Derivamos cra λ_i y α

$$1)\ \frac{\partial L}{\partial \lambda_1} = 1-\alpha = 0 \qquad 2)\ \frac{\partial L}{\partial \lambda_2} = 1-4\lambda_2+\alpha = 0 \qquad 3)\ \frac{\partial L}{\partial \alpha} = -\lambda_1+\lambda_2 = 0$$

$$\text{Solución} \equiv\ \alpha = 1,\ \lambda_1 = 1/2,\ \lambda_2 = 1/2,$$

- Vector de proyección:

$$w = \sum_{i \in S} \lambda_i z_i \mathbf{x}_i = (1/2)\times 1\times 0 + (1/2)\times(-1)\times 2 = -1$$

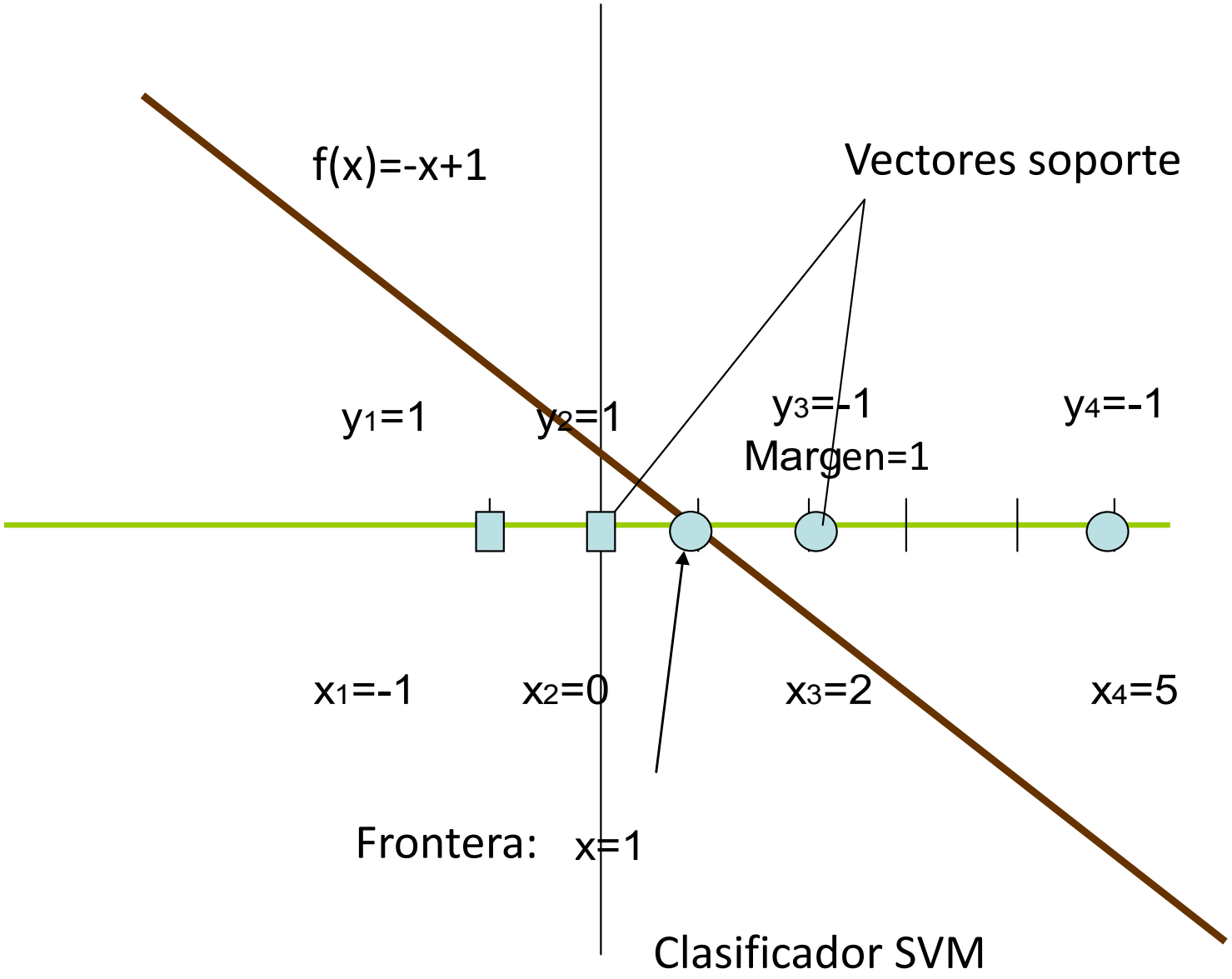
- Sesgo óptimo:

$$b = 1\text{-}wx_1 = 1-(-1)\times 0 = 1$$

- Clasificador:

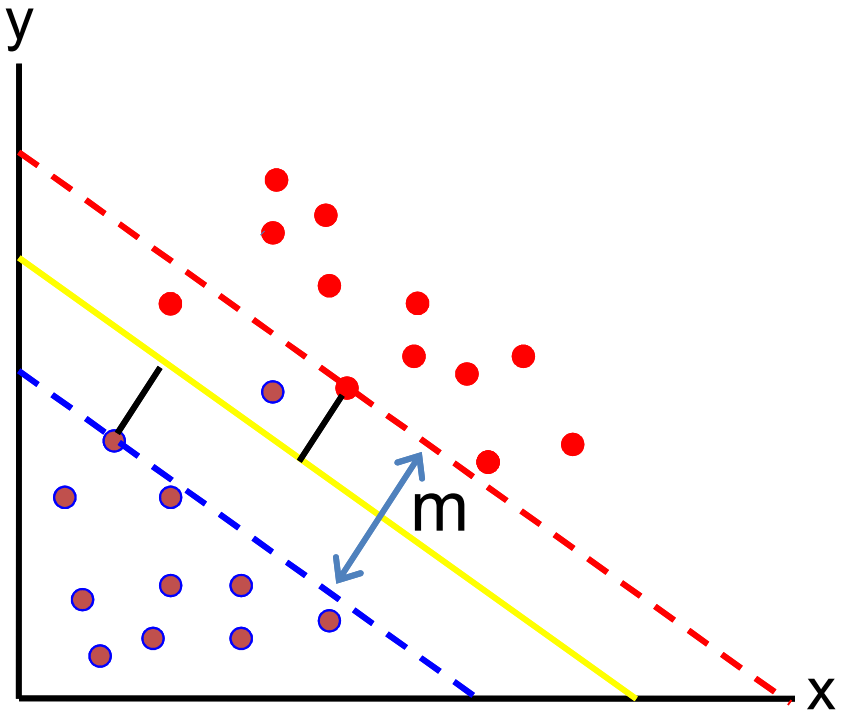
– El clasificador SVM es $f(x)= -x+1$ y la frontera de decisión para la cual $g(x)=0$ es $x=1$

Ejemplo: solución dual con vectores soporte



Entrenamiento: Margen Suave

- Supongamos que relajamos lo de "linealmente separable"
- Errores de compensación para distancia mayor *m*
  - Más errores, pero ganamos margen
- Aquí se ilustran dos tipos de errores...



Errores

- Para tener en cuenta los errores, introducimos "variables de holgura" $\xi_i \geq 0$ en la optimización.
- Para el punto rojo $\mathbf{x}_i = (x_i, y_i)$, la restricción es:

$$w_1 x + w_2 y + b > 1 - \xi_i$$

- Para el punto morado $\mathbf{x}_i = (x_i, y_i)$, la restricción es

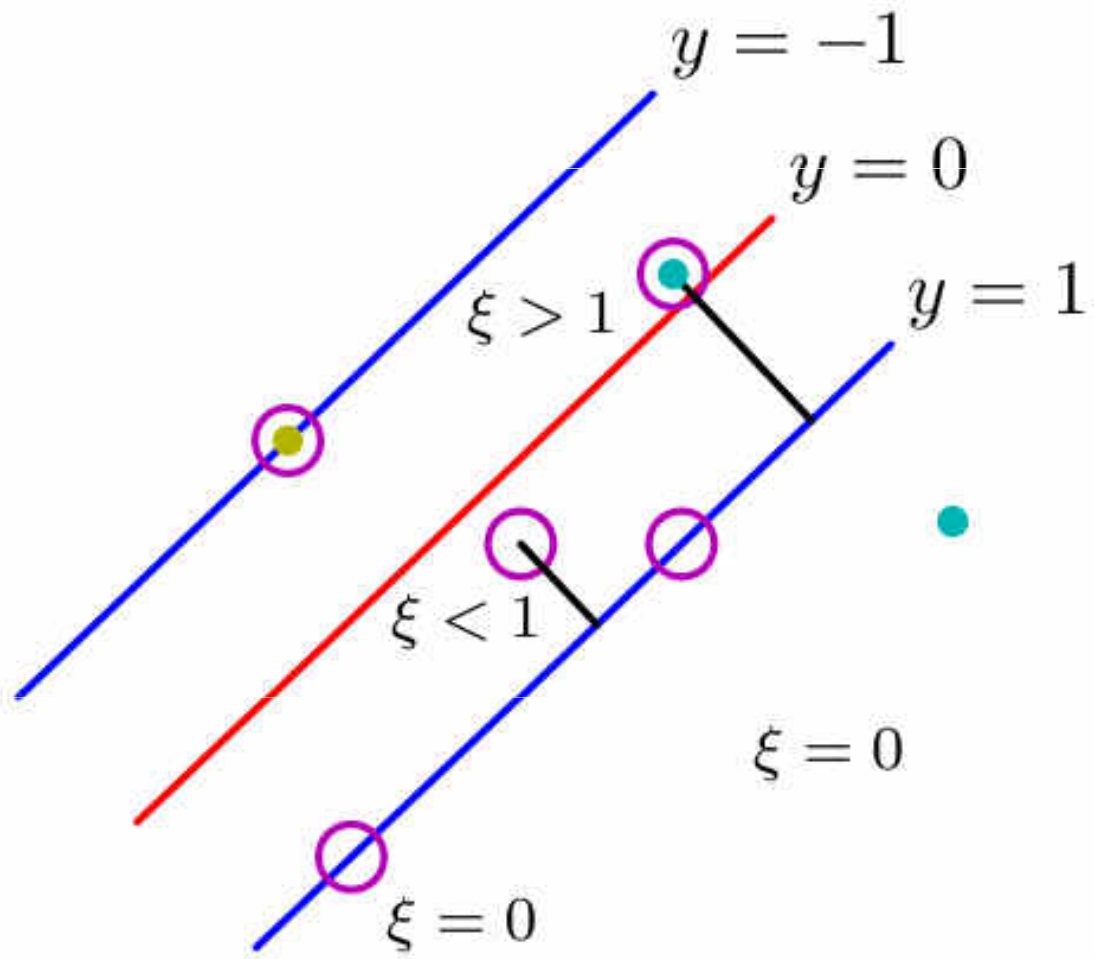
$$w_1 x + w_2 y + b \leq -1 - \xi_i$$

- Problema primal es ahora:

$$\min \frac{w_1^2 + w_2^2}{2} + C \sum_i \xi_i$$

$$\text{s.t. } z_i \left(w_1 x + w_2 y + b \right) \geq 1 - \xi_i$$

Variables de holgura



Problema dual

- Trabajando los detalles, el problema dual es...

$$\max L(\lambda) = \sum_i \lambda_i - \frac{1}{2}\sum_i \sum_j \lambda_i \lambda_j z_i z_j \left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle$$

$$\text{s.t.} \sum_i \lambda_i z_i = 0 \text{ and } C \geq \lambda_i \geq 0, i \in \{1, \ldots, n\}$$

  - La única diferencia es la condición $C \geq \lambda_i$
  - Especificamos C antes del entrenamiento (hiperparámetro)
- ¿Conclusión?
  - Permitir errores cambia las restricciones

Entrenamiento y puntuación revisados nuevamente

- Entrenamiento

$$\max L(\lambda) = \sum_i \lambda_i - \frac{1}{2}\sum_i \sum_j \lambda_i \lambda_j z_i z_j \left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle$$

$$\text{s.t.} \sum_i \lambda_i z_i = 0 \text{ and } C \geq \lambda_i \geq 0, i \in \{1, \ldots, n\}$$

- Puntuación: dado un patrón de test **x**=(*x*,*y*)
  - Calcular: $$f(\mathbf{x}) = \left(\sum_{i \in S} \lambda_i z_i \left\langle \mathbf{x}_i, \mathbf{x} \right\rangle \right) + b$$

    sumando en el conjunto de vectores de soporte
  - Si $f(\mathbf{x}) < 0$, entonces **x** es “azul”; de lo contrario es "púrpura"

Papel de C

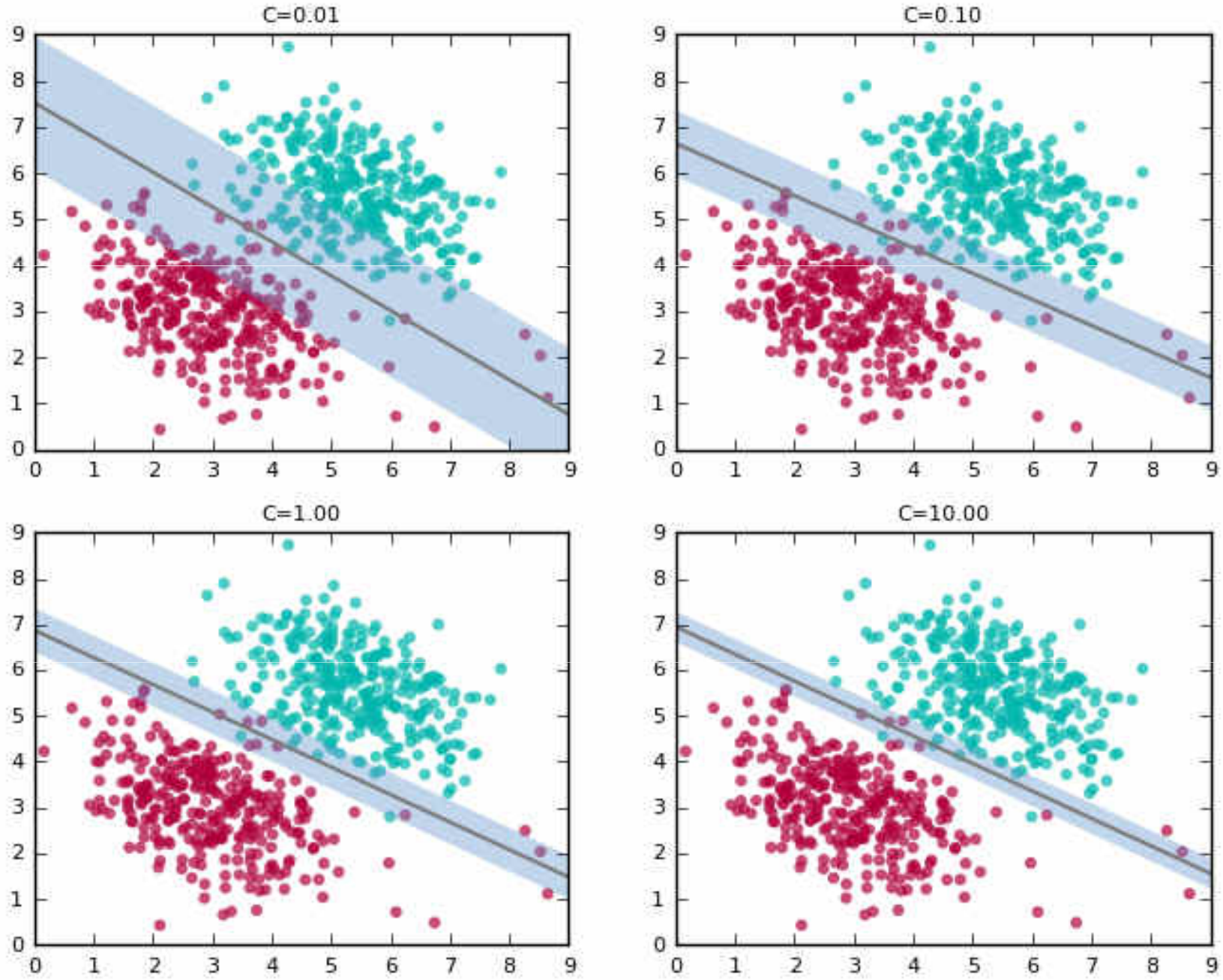
- Recuerda el primal:

$$\min \frac{w_1^2 + w_2^2}{2} + C \sum_i \xi_i$$

$$\text{s.t. } z_i \left(w_1 x + w_2 y + b \right) \geq 1 - \xi_i$$

- El primer término maximiza el margen, mientras que el segundo minimiza el número de errores. De esta forma, *C* es un parámetro de penalización, definido por el usuario y que depende del conjunto de datos.
- Mayor margen → mayor regularización.
- *C* actúa como el inverso de un **término de regularización** (cuanto mayor *C* menor regularización y menor error; cuanto menor *C* mayor regularización y el error mayor).

Papel de C



Truco del kernel

- Finalmente, podemos entender el truco del kernel.
- $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ son los vectores de entrenamiento:
  - Para **entrenamiento**, $\mathbf{x}_i$ solo aparece como $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$
  - Cuando **puntuamos** $\mathbf{x}$, $\mathbf{x}_i$ solo aparece como $\langle \mathbf{x}_i, \mathbf{x} \rangle$
- Podemos transformar los datos de entrada a un espacio de características.
  - Y calcular productos escalares en el espacio de características.
- El efecto es reemplazar "$\langle -, - \rangle$ " con algo definido en dimensiones superiores
  - Y dado que el producto escalar está en el espacio de características ...
  - …es fácil de calcular (en términos de espacio de entrada)

Ejemplo de kernel

- Por ejemplo, supongamos que definimos un espacio de características:

$$\phi(X) = \phi(x, y) = (1, \sqrt{2}x, \sqrt{2}y, x^2, y^2, \sqrt{2}xy)$$

- Entonces ϕ asigna cada punto 2D a 6D:
- Para $\mathbf{x}_i = (x_i, y_i)$ y $\mathbf{x}_j = (x_j, y_j)$, tenemos:

  $<\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)> = (1 + x_i x_j + y_i y_j)^2 \quad = (<\mathbf{x}_i, \mathbf{x}_j> + 1)^2$
- En este caso, la función de kernel K es:

  $K(\mathbf{x}_i, \mathbf{x}_j) = (<\mathbf{x}_i, \mathbf{x}_j> + 1)^2$
- **Nota** : K es una composición de ϕ y "<,>"

Ejemplo de kernel

- En el espacio de entrada, $\mathbf{x}_i$ y $\mathbf{x}_j$ son 2D
- Mapeamos $\mathbf{x}_i$ y $\mathbf{x}_j$ a un espacio de características de 6D
  - Imágenes $\phi(\mathbf{x}_i)$ y $\phi(\mathbf{x}_j)$
- Realizamos el producto escalar en el espacio de características:
  - Es decir, calculamos $<\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)>$
- Las matemáticas funcionan tanto para la entrada 2D usando el kernel $K(\mathbf{x}_i, \mathbf{x}_j)$ como para el espacio de características 6D usando $<\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)>$
  - ¡Podemos usar el kernel para tener un producto escalar en el espacio de características usando operaciones en el espacio de entrada!

El panorama general

- Los datos de entrenamiento viven en el espacio de entrada:
  - Donde los datos pueden ***no*** ser linealmente separables
- Asigamos el espacio de entrada al espacio de características de mayor dimensión usando la función ϕ
- Realizamos el entrenamiento y la puntuación en el espacio de características:
  - Donde los datos pueden ser linealmente separables
- Pero no quiero sufrir una penalización en el rendimiento debido a una dimensión mayor (usamos el kernel):
  - Elije sabiamente la función del kernel...

Entrenamiento y puntuación con Kernel

- Simplemente remplazamos $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ con $K(\mathbf{x}_i, \mathbf{x}_j)$
- Entrenamiento :

$$\max L(\lambda) = \sum_i \lambda_i - \frac{1}{2} \sum_i \sum_j \lambda_i \lambda_j z_i z_j K\left(\mathbf{x}_i, \mathbf{x}_j\right)$$

$$\text{s.a.} \sum_i \lambda_i z_i = 0 \text{ y } C \geq \lambda_i \geq 0, i \in \{1, \ldots, n\}$$

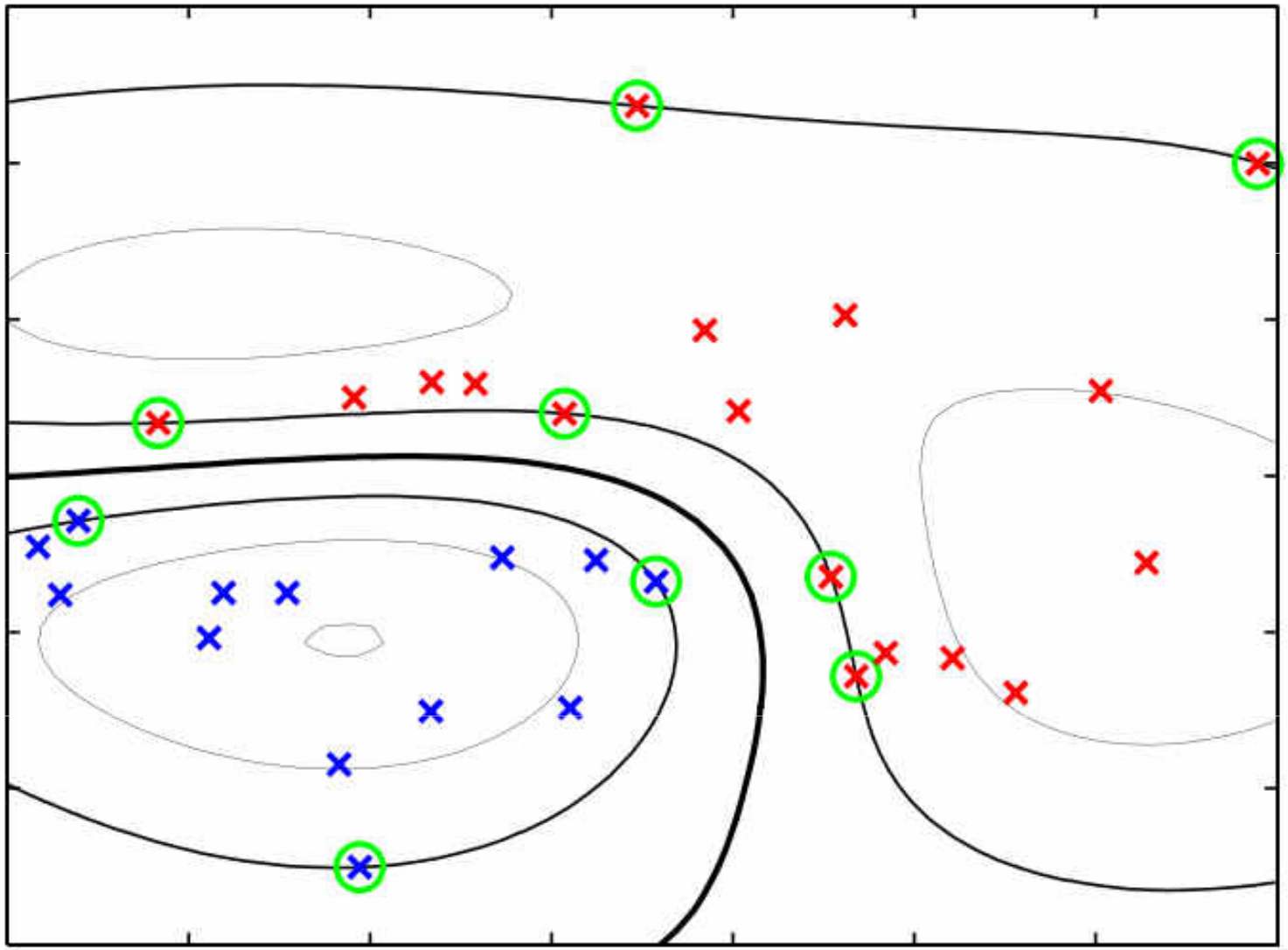
donde C es un parámetro del usuario

- Puntuación : dado $\mathbf{x}=(x,y)$

$$f(\mathbf{x}) = \left(\sum_{i \in S} \lambda_i z_i K\left(\mathbf{x}_i, \mathbf{x}\right) \right) + b$$

– Si $f(\mathbf{x}) < 0$, entonces $\mathbf{x}$ es “azul” ; más “púrpura”

SVM después del truco del kernel (RBF)



Truco del kernel

- No es necesario asignar explícitamente datos de entrada al espacio de características.
- Ni siquiera necesitamos saber ϕ
  - Sólo se necesita la función kernel resultante, K
- En pocas palabras
  - Obtener el beneficio de trabajar en un espacio de dimensiones superiores (linealmente separable)...
  - ...sin penalización significativa en el rendimiento
  - ¡¡¡Ese es un truco realmente increíble!!!

Elecciones comunes para el kernel

- Kernel lineal (problema original): $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$
- Máquina de aprendizaje polinomial

  $K(\mathbf{x}_i, \mathbf{x}_j) = (\langle \mathbf{x}_i, \mathbf{x}_j \rangle + 1)^p$

  donde p es un hiperparámetro.
- Kernel gaussiano (el más popular, siguiente diapositiva)
- Perceptrón de dos capas

  $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\beta_0 \langle \mathbf{x}_i, \mathbf{x}_j \rangle + \beta_1)$
- Muchas otras posibilidades
  - Seleccionar el kernel "correcto" es el verdadero truco

Elecciones comunes para el kernel

- Función de base radial gaussiana (RBF)

$$K\left(\mathbf{x}_i, \mathbf{x}_j\right) = \exp\left(-\frac{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2}{2\sigma^2}\right)$$

  donde σ es un hiperparámetro.

- Definición equivalente de RBF:

$$K\left(\mathbf{x}_i, \mathbf{x}_j\right) = \exp\left(-\gamma\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2\right) \quad \text{donde } \gamma = \frac{1}{2\sigma^2}$$

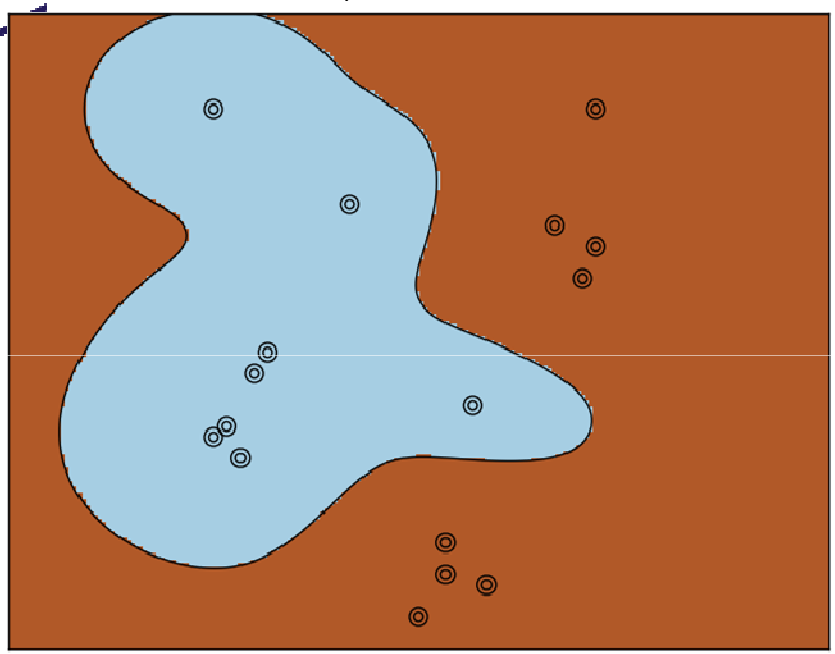
Parámetros de un kernel RBF

C	Superficie decisión	Modelo	Sesgo	Varianza
Pequeño	Suave	Simple	Alto	Bajo
Grande	Irregular	Complejo	Bajo	Alto

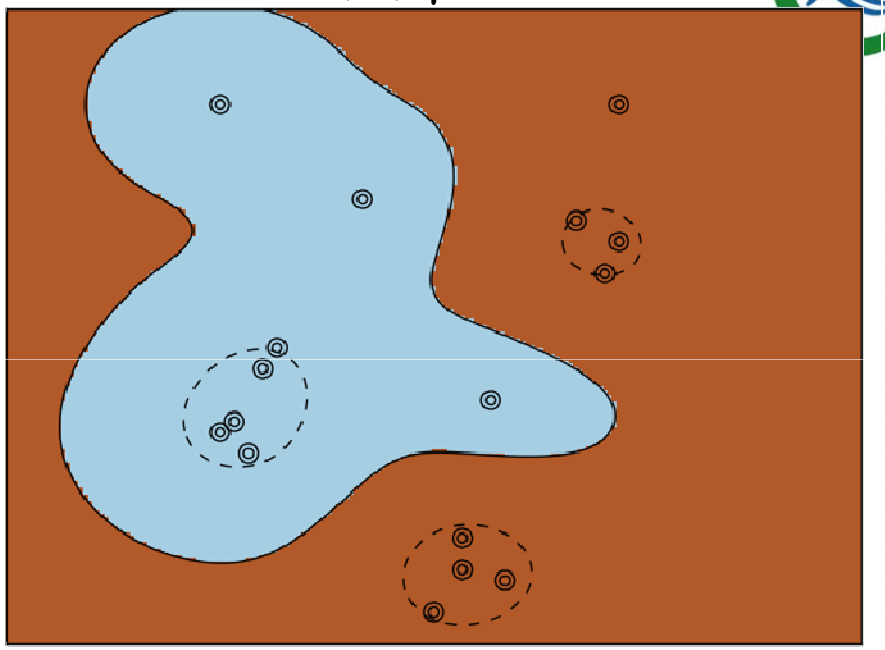
Gamma	Puntos afectados	Superficie de decisión	Modelo	Sesgo	Varianza
Pequeño→ alto sigma	Lejos de los datos	Suave	Simple	Alto	Bajo
Grande → bajo sigma	Cerca de los datos	Irregular	Complejo	Bajo	Alto

- Un valor más alto de gamma aumenta la varianza del modelo.
- La selección de gamma es muy importante para el desempeño de SVM, por lo que la opción más común es aplicar una búsqueda en grilla guiada por validación cruzada.
- Los valores se suelen considerar en escala logarítmica.
- Se debe considerar una amplia gama.

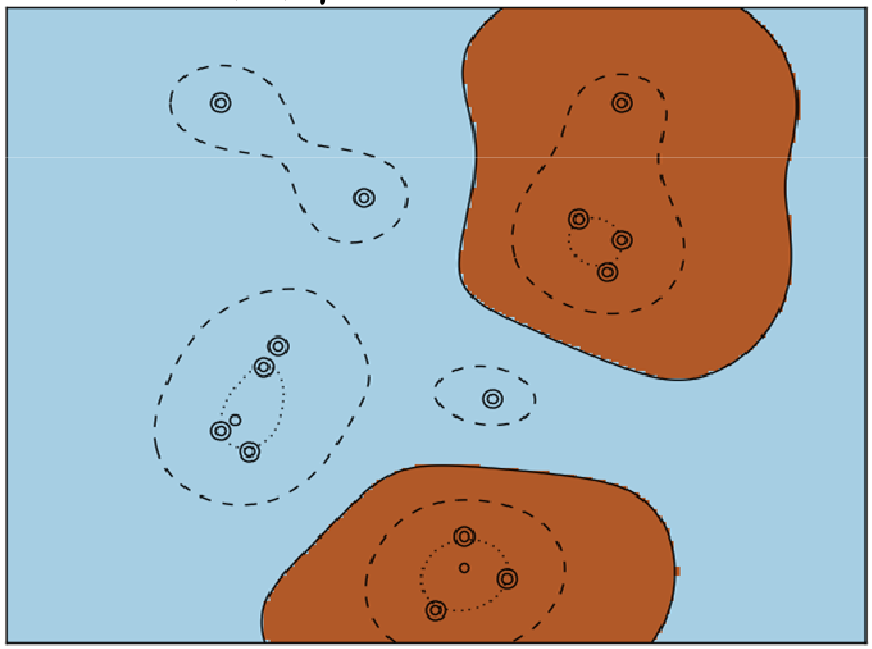
C=0,05, γ =2



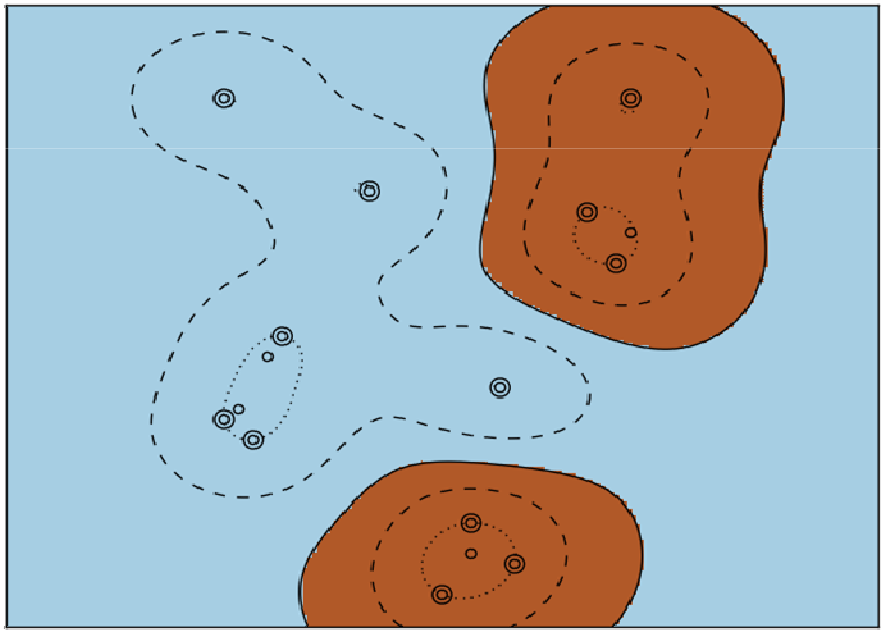
C=0,2, γ =2



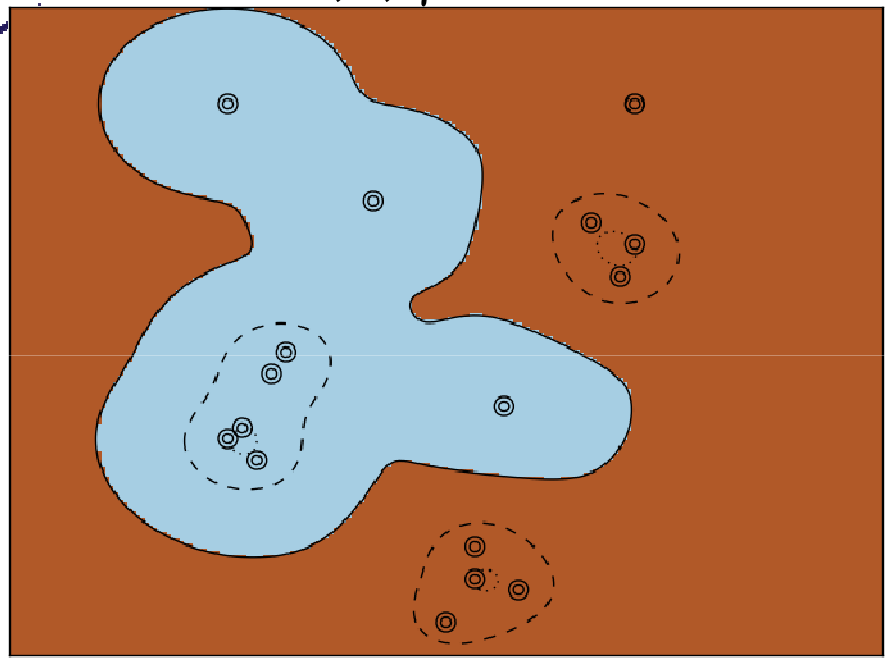
C=0,6, γ =2



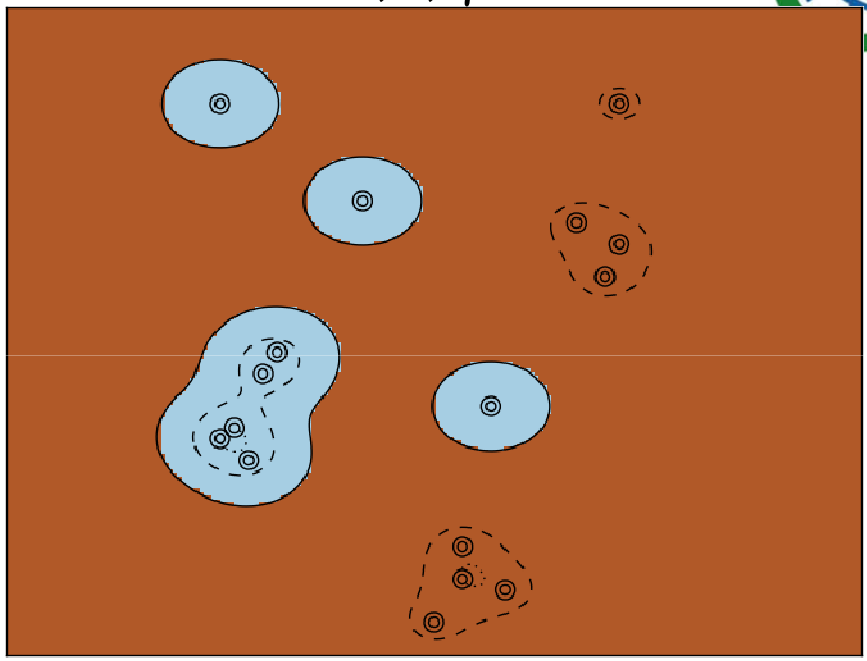
C=2, γ =2



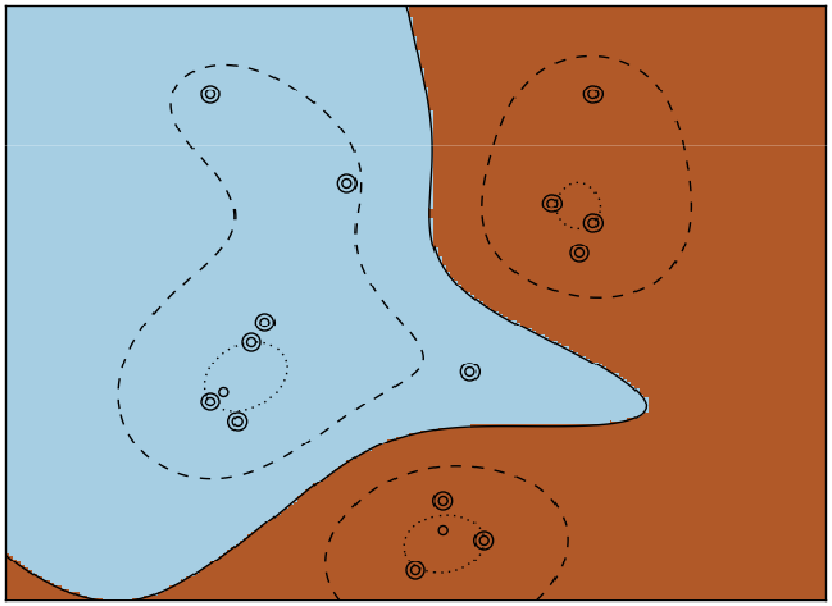
C=0,5, γ =5



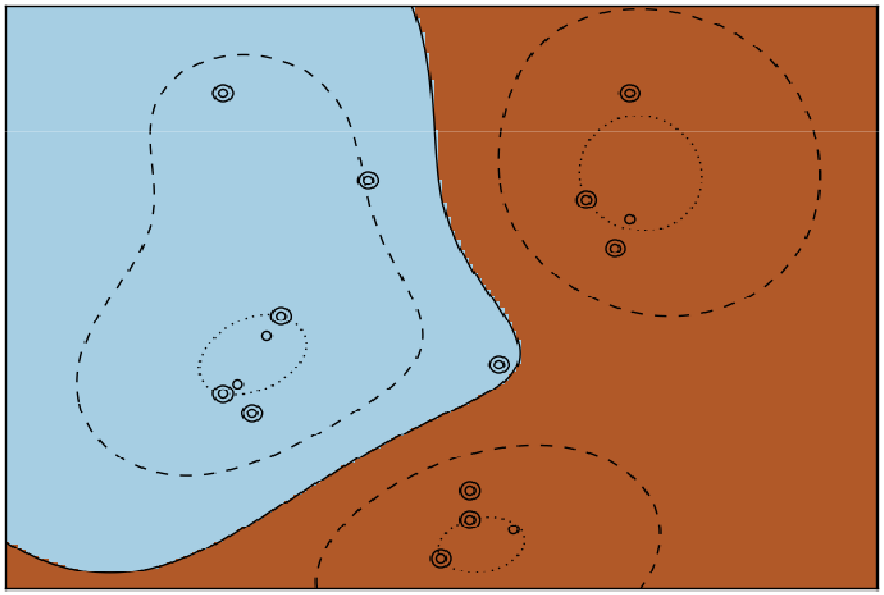
C=0,5, γ =10



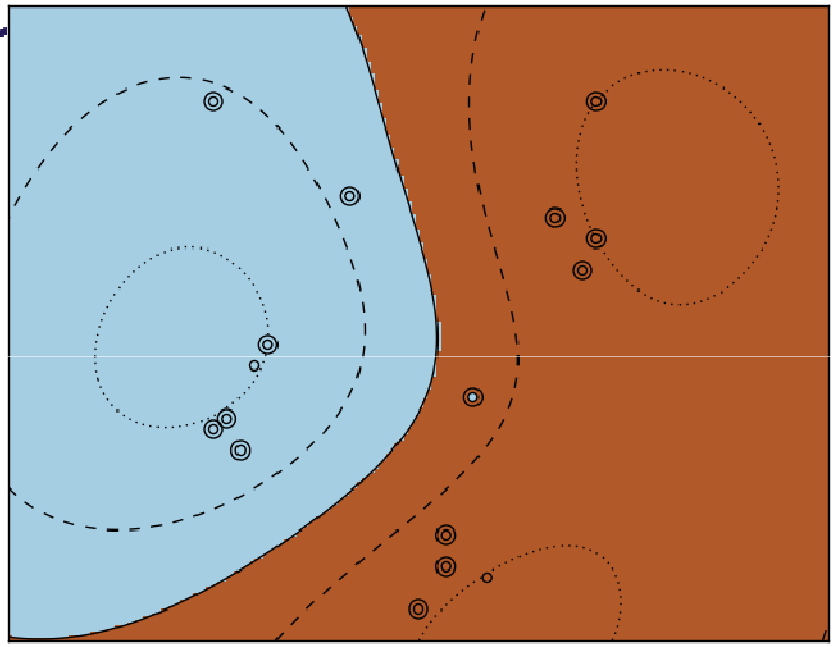
C=0,5, γ =1



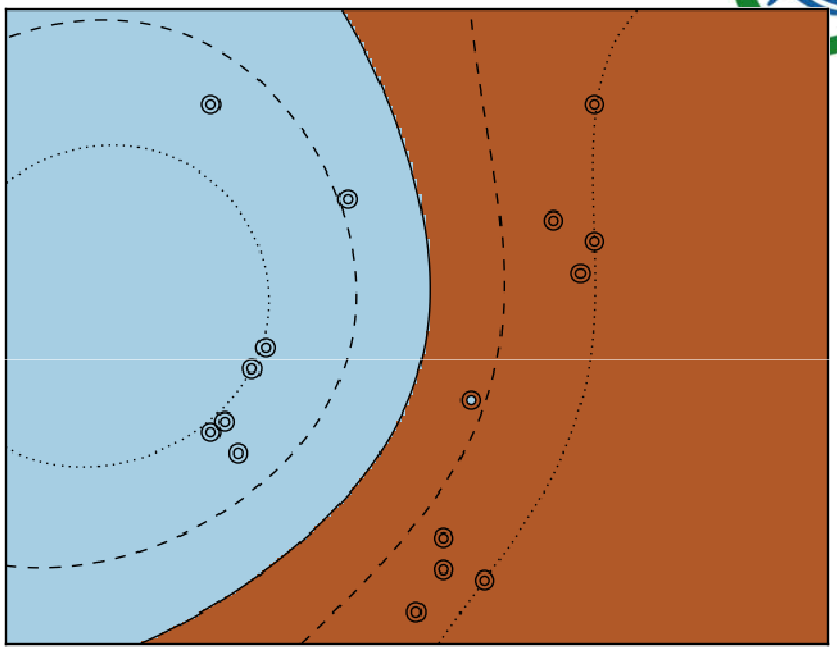
C=0,5 γ =0,5



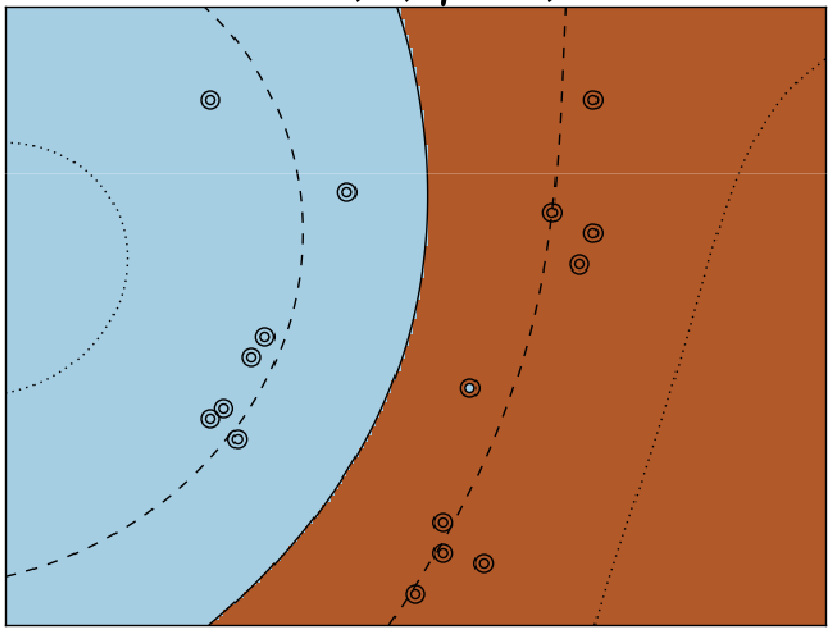
C=0,5, γ =0,2



C=0,5, γ =0,1

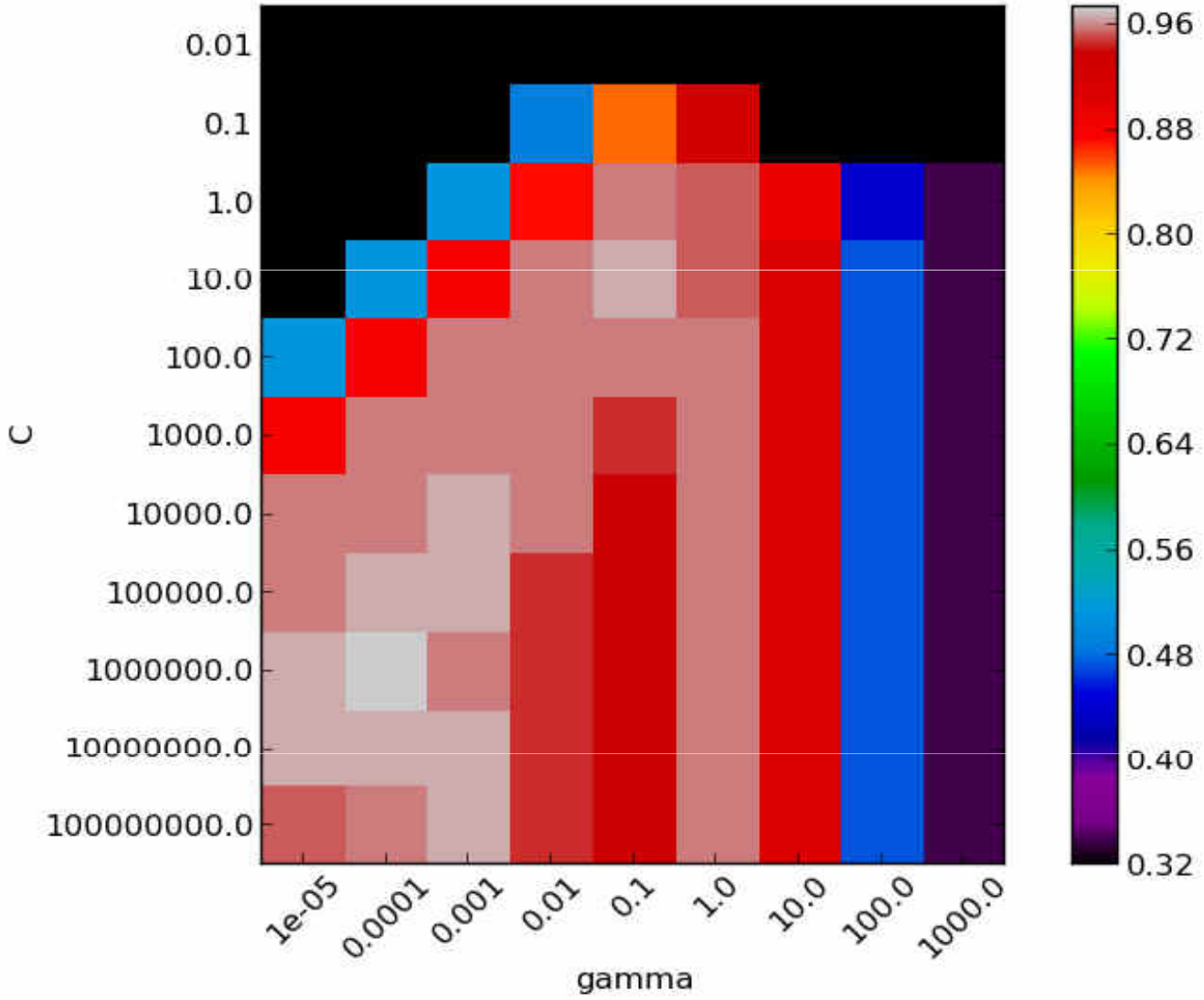


C=0,5, γ =0,05



Parámetros de un kernel RBF

Mapa de validación cruzada



http://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html

- SVM lineal de margen estricto

Primal

$$\min_{\mathbf{w}} \; m = \frac{\langle \mathbf{w}, \mathbf{w} \rangle}{2}$$

$$\text{s.a. } z_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b \right) \geq 1, \; i \in \{1, 2, \ldots, N\}$$

Dual

$$\max_{\boldsymbol{\lambda}, \alpha} L(\boldsymbol{\lambda}, \alpha) = \sum_i \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j z_i z_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle - \alpha \left(\sum_i \lambda_i z_i \right)$$

$$\text{s.a. } \alpha \geq 0, \quad \lambda_i \geq 0, \; i = 1, \ldots, N$$

$$\mathbf{w} = \sum_{i \in S} \lambda_i z_i \mathbf{x}_i \qquad b = 1 - \langle \mathbf{w}, \mathbf{x}_{i | z_i = 1, \lambda_i \neq 0} \rangle$$

- SVM lineal de margen suave

Primal

$$\min_{\mathbf{w},\varepsilon_i} \; m = \frac{\langle \mathbf{w},\mathbf{w} \rangle}{2} + C\sum_i \xi_i$$

$$\text{s.a. } z_i\left(\langle \mathbf{w},\mathbf{x}_i \rangle + b\right) \geq 1-\xi_i, \; i \in \{1,2,\ldots,N\}$$

Dual

$$\max_{\boldsymbol{\lambda},\alpha} L(\boldsymbol{\lambda},\alpha) = \sum_i \lambda_i - \frac{1}{2}\sum_{i,j}\lambda_i\lambda_j z_i z_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle - \alpha\left(\sum_i \lambda_i z_i\right),$$

$$\text{s.a. } \; C \geq \lambda_i \geq 0, \; i = 1,\ldots,N$$

$$\mathbf{w} = \sum_{i\in S}\lambda_i z_i \mathbf{x}_i \qquad b = 1 - \langle \mathbf{w}, \mathbf{x}_{i|z_i=1,\lambda_i\neq 0} \rangle$$

- SVM no lineal de margen estricto

Primal

Con la versión no lineal, normalmente trabajamos en el dual.

$$\min_{\mathbf{w}} \; m = \frac{\langle \mathbf{w}, \mathbf{w} \rangle}{2}$$

$$\text{s.a. } z_i \left(\langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle + b \right) \geq 1, \; i \in \{1, 2, \ldots, N\}$$

Dual

$$\max_{\boldsymbol{\lambda}, \alpha} L(\boldsymbol{\lambda}, \alpha) = \sum_i \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j z_i z_j K(\mathbf{x}_i, \mathbf{x}_j) - \alpha \left(\sum_i \lambda_i z_i \right)$$

$$\text{s.a. } \alpha \geq 0, \quad \lambda_i \geq 0, \; i = 1, \ldots, N$$

$$f(\mathbf{x}) = \left(\sum_{i \in S} \lambda_i z_i K(\mathbf{x}_i, \mathbf{x}) \right) + b \qquad b = 1 - \left(\sum_{i \in S} \lambda_i z_i K\left(\mathbf{x}_i, \mathbf{x}_{j | z_i = 1, \lambda_i \neq 0} \right) \right)$$

- SVM no lineal de margen suave

Primal

Con la versión no lineal, normalmente trabajamos en el dual.

$$\min_{\mathbf{w},\varepsilon_i} \; m = \frac{\langle \mathbf{w},\mathbf{w} \rangle}{2} + C\sum_i \xi_i$$

$$\text{s.a. } z_i\left(\langle \mathbf{w}, \Phi(\mathbf{x}_i)\rangle + b\right) \geq 1 - \xi_i,\; i \in \{1,2,\ldots,N\}$$

Dual

$$\max_{\boldsymbol{\lambda},\alpha} L(\boldsymbol{\lambda},\alpha) = \sum_i \lambda_i - \frac{1}{2}\sum_{i,j}\lambda_i\lambda_j z_i z_j K(\mathbf{x}_i,\mathbf{x}_j) - \alpha\left(\sum_i \lambda_i z_i\right)$$

$$\text{s.a. } \; C \geq \lambda_i \geq 0,\; i = 1,\ldots,N$$

$$f(\mathbf{x}) = \left(\sum_{i\in S}\lambda_i z_i K(\mathbf{x}_i,\mathbf{x})\right) + b \qquad b = 1 - \left(\sum_{i\in S}\lambda_i z_i K\left(\mathbf{x}_i, \mathbf{x}_{j|z_j=1,\lambda_j\neq 0}\right)\right)$$

SVM: aspectos prácticos

- A la hora de clasificar conviene seguir esta guía:
  - Escalar las entradas :
    - Las variables deben tener media 0 y desviación típica 1.
    - Sino, algunas características dominan a las otras.
  - Selección de modelo :
    - Por lo general, consideramos RBF como la primera opción para el kernel, porque tiene menos parámetros.
    - Tenemos que configurar *C* y gamma.
    - Es común considerar estos rangos:

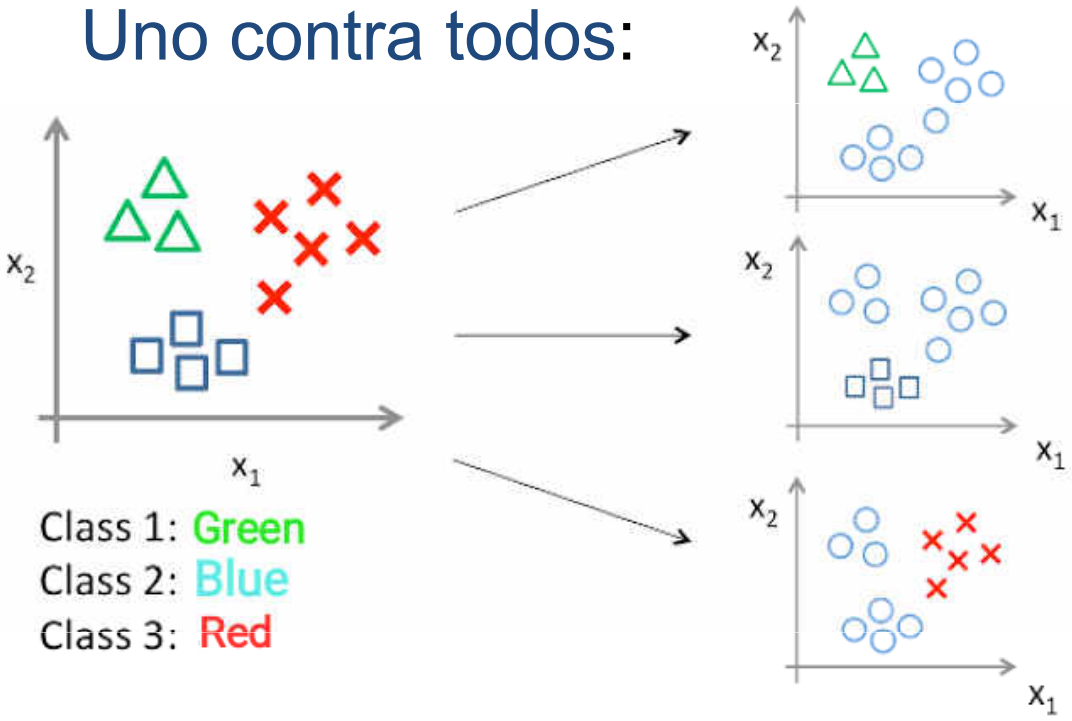
$$C \in \left\{10^{-3}, 10^{-2}, ..., 10^{3}\right\} \text{ y } \gamma \in \left\{10^{-3}, 10^{-2}, ..., 10^{3}\right\}$$

SVM: clasificación multiclase

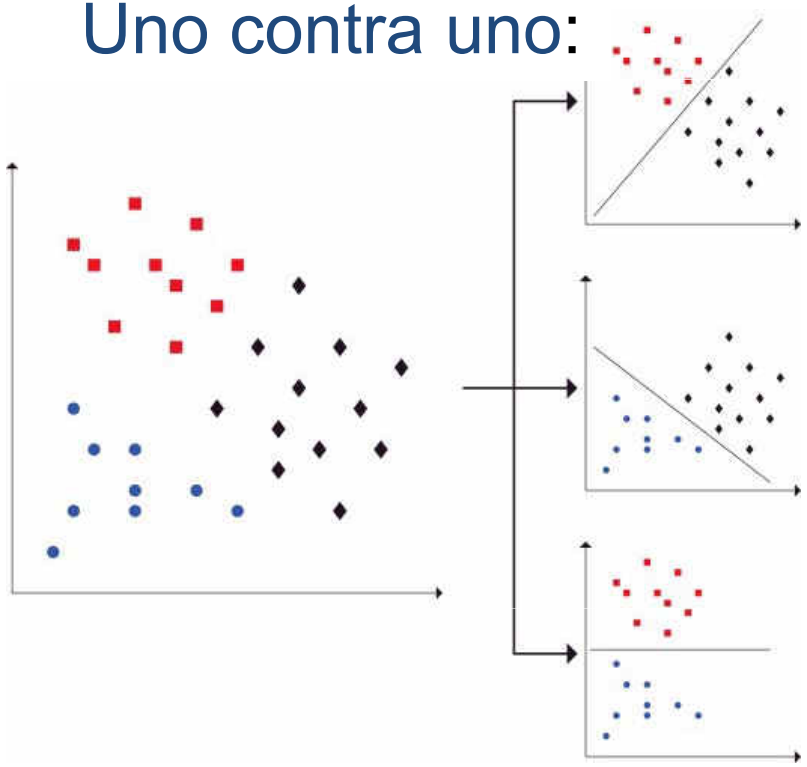
- Las SVM se definen naturalmente para la clasificación binaria.
- Se pueden adaptar a la clasificación multiclase mediante dos estrategias:
  - Uno contra todos:
    - Se construye un clasificador binario para cada clase, donde los ejemplos de la clase son positivos y todos los demás ejemplos son negativos → Q clasificadores , donde Q es el número de clases.
    - La decisión final es la decisión positiva con la puntuación máxima.
  - Uno contra uno:
    - Un clasificador binario con ejemplos de dos pares de clases → Q*(Q-1)/2 Clasificadores binarios
    - La decisión final se basa en una estrategia de votación después de aplicar todos los clasificadores (votación mayoritaria).

SVM: clasificación multiclase

Uno contra todos:



Uno contra uno:



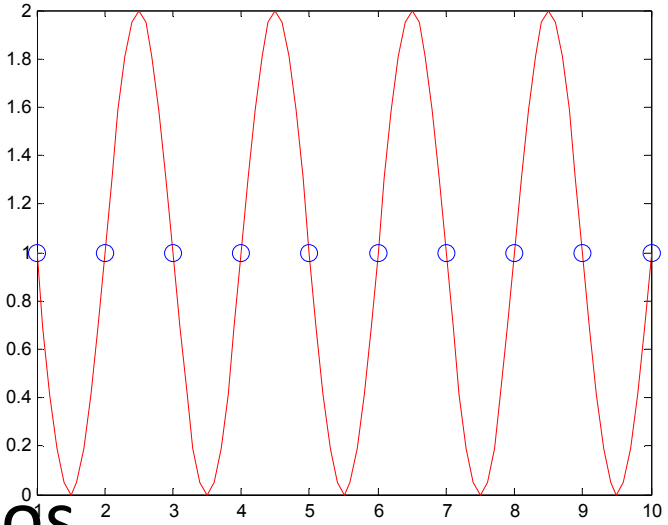
Regresión de vectores soporte (SVR)

- Para entrenar un regresor, minimizamos una medida de error, también llamada “función de pérdida”.
- También queremos que la función sea sencilla:
  - Menos funciones de base
  - Funciones de base simples
  - La planicidad es deseable

  (dado y=1 para x=i, i=1,10

  no ajustamos un seno a los datos.

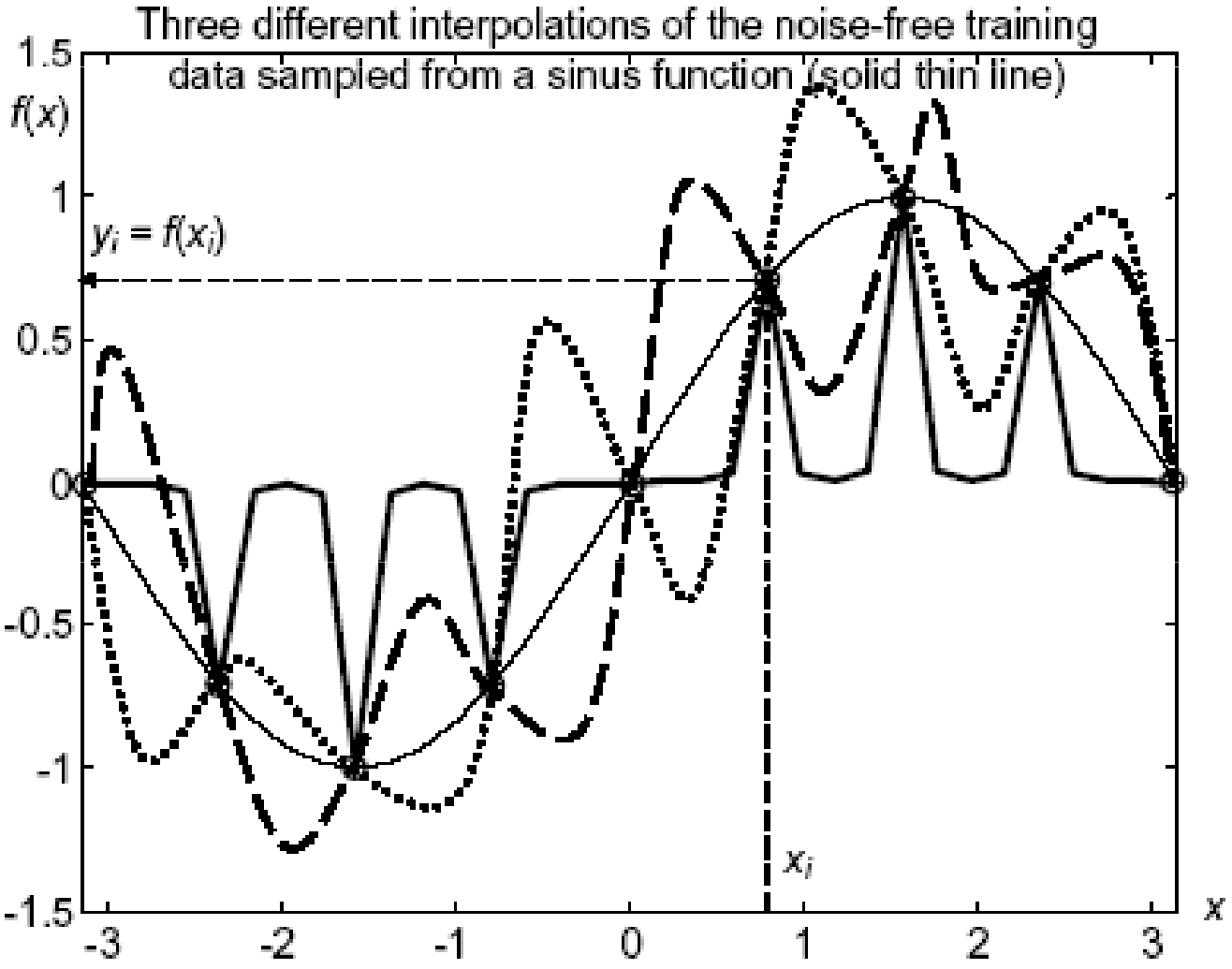
  ¿Por qué?)



Regresión de vectores soporte (SVR)

- Combina función de pérdida y planicidad como un único objetivo.
- Regresión lineal en el espacio de características.
- Planicidad en la predicción :
  - Por ejemplo, predecimos el precio de una casa basado en el número de habitaciones , y tenemos 2 funciones :
    - f1: # habitaciones *10000 + 10000
    - f2: # habitaciones *20000 -10000
  - Ambos están de acuerdo en su predicción para una casa de dos habitaciones, que cuesta 30000
    - Pero f1 es más plano que f2; f1 es menos sensible al ruido
  - Normalmente , la planitud es se mide usando ||w|| que es 20000 para f2 y 10000 para f1; cuanto menor sea||w||, más plana es f…
  - En consecuencia, ||w|| se minimiza para SVR; sin embargo, en la mayoría de los casos, usamos$||w||^2$ y nos quitamos la raíz

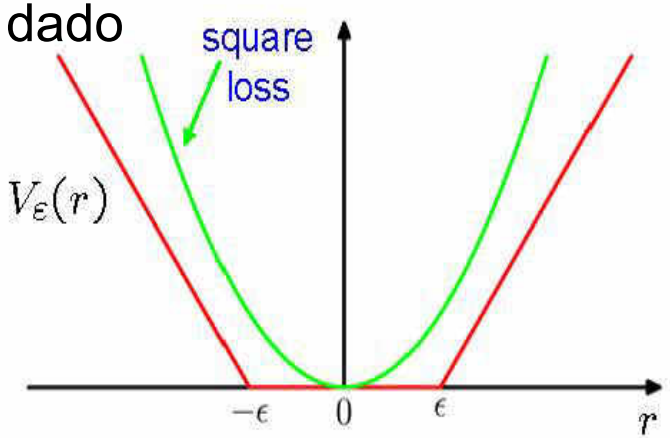
- Sobreajuste en regresión no lineal

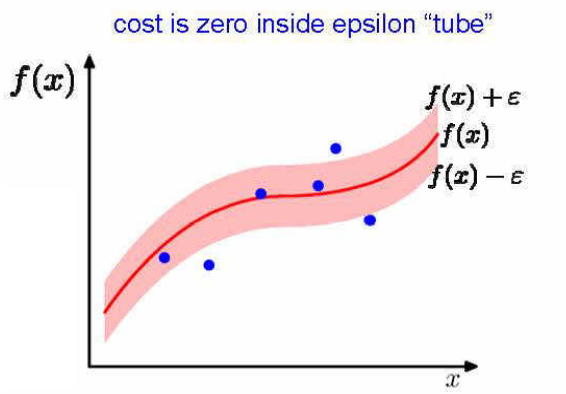


SVR: pérdida insensible a épsilon

- Con SVR se puede utilizar cualquier función de pérdida, pero la que se asocia con mayor frecuencia es la función épsilon-insensible.
- Es menos sensible a un punto de datos con ruído.

r : Error para un punto dado





Gráficos tomados de Gunn's *Support Vector Machines for Classification and Regression*

SVR: problema de optimización

- Dado: un conjunto con entradas $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ y valores objetivo $y_1, y_2, \ldots, y_N, y_i \in \mathbb{R}$
- El problema de optimización primal es:

Primal

$$\min_{\mathbf{w}, \varepsilon_i} \; m = \frac{\langle \mathbf{w}, \mathbf{w} \rangle}{2} + C \sum_i \left(\xi_i + \xi_i^* \right),$$

$$\text{s.a.} \quad \begin{cases} y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle - b \leq \varepsilon + \xi_i, \\ \langle \mathbf{w}, \mathbf{x}_i \rangle + b - y_i \leq \varepsilon + \xi_i^*, \\ \xi_i, \xi_i^* \geq 0, \end{cases}$$

$$i \in \{1, 2, \ldots, N\}.$$

La constante *C* determina el compromiso entre la planitud de *f* y la cantidad hasta la cual se toleran desviaciones mayores que ε

SVR: problema de optimización

- El lagrangiano correspondiente es el siguiente:

$$L = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) - \sum_{i=1}^{N}(\lambda_i\xi_i + \lambda_i^*\xi_i^*)$$

$$-\sum_{i=1}^{N}\lambda_i(\varepsilon + \xi_i - y_i + \langle \mathbf{w}, \mathbf{x}_i\rangle + b)$$

$$-\sum_{i=1}^{N}\lambda_i^*(\varepsilon + \xi_i^* + y_i - \langle \mathbf{w}, \mathbf{x}_i\rangle - b)$$

Donde $\alpha_i, \lambda_i, \alpha_i^*, \lambda_i^*$ son los multiplicadores de Lagrange que deben satisfacer restricciones de positividad.

SVR: problema de optimización

Dual

$$\max_{\boldsymbol{\lambda},\alpha} L(\boldsymbol{\lambda},\boldsymbol{\lambda}^*,\alpha) = -\frac{1}{2}\sum_{i,j}\left(\lambda_i-\lambda_j\right)\left(\lambda_i^*-\lambda_j^*\right)\left\langle \mathbf{x}_i,\mathbf{x}_j\right\rangle -$$

$$-\varepsilon\sum_i(\lambda_i+\lambda_i^*)+\sum_i y_i(\lambda_i-\lambda_i^*),$$

$$\text{s.t. } \sum_i^{\ell}(\lambda_i-\lambda_i^*)=0, \quad 0\le\lambda_i\le C, \quad 0\le\lambda_i^*\le C.$$

$$\mathbf{w}=\sum_{i\in S}\left(\lambda_i-\lambda_i^*\right)\mathbf{x}_i$$

SVR: problema de optimización

- Resulta que:

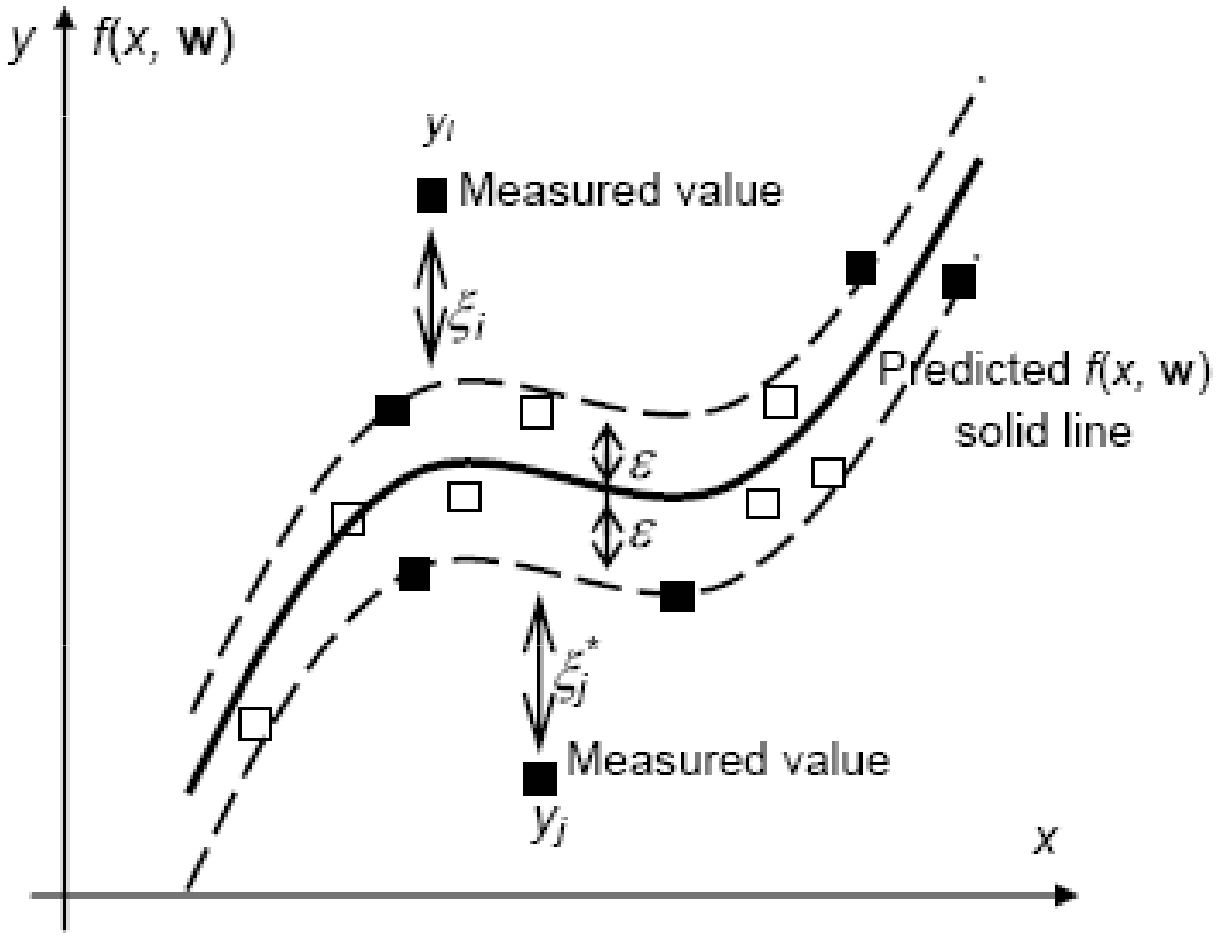
$$\lambda_i > 0, \quad \langle \mathbf{w}, \mathbf{x}_i \rangle + b - y_i + \varepsilon = 0,$$
$$\lambda_i^* < C, \quad -\langle \mathbf{w}, \mathbf{x}_i \rangle - b + y_i + \varepsilon = 0.$$

- Por lo tanto, para todos los datos que cumplen $y - f(x) = +\varepsilon$, las variables duales $0 < \lambda_i < C$, y para $y - f(x) = -\varepsilon$, las variables duales $0 < \lambda_i^* < C$. Estos puntos de datos se denominan vectores soporte no acotados.
- Los vectores soporte no acotados permiten calcular el valor del término de sesgo *b* :

$$b = y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle - \varepsilon, \text{ for } 0 < \lambda_i < C,$$
$$b = y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle + \varepsilon, \text{ for } 0 < \lambda_i^* < C.$$

- El cálculo de un término de sesgo b es numéricamente muy sensible y es mejor calcular el sesgo b promediando todos los vectores soporte.

SVR: problema de optimización



SVR: problema de optimización

- Para todos los puntos de datos dentro del ε-tubo, es decir,

$$\left| y_i - (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \right| < \varepsilon$$

- La solución será dispersa (muchos λ_i, λ_i^* serán cero y los puntos estarán fuera del tubo).
- Comparación con SVC:
  - El tamaño del problema SVR, con respecto al tamaño de una tarea de diseño de clasificador SV, ahora se duplica. Hay *2N* variables duales desconocidas (*N* y *N*) para una regresión de vector de soporte.
  - Además del parámetro de penalización C y los parámetros de forma de las funciones del núcleo (como las varianzas de un núcleo gaussiano, o el orden del polinomio), la zona de insensibilidad ε también debe establecerse de antemano al construir máquinas SV para regresión.

SVR no lineal

Primal

$$\min_{\mathbf{w},\varepsilon_i} \; m = \frac{\langle \mathbf{w},\mathbf{w} \rangle}{2} + C\sum_i \left(\xi_i + \xi_i^*\right),$$

$$s.t. \quad \begin{cases} y_i - \langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle - b \le \varepsilon + \xi_i, \\ \langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle + b - y_i \le \varepsilon + \xi_i^*, \\ \xi_i, \xi_i^* \ge 0, \end{cases}$$

$$i \in \{1, 2, \ldots, N\}.$$

Dual

$$\max_{\boldsymbol{\lambda},\alpha} L(\boldsymbol{\lambda}, \boldsymbol{\lambda}^*, \alpha) = -\frac{1}{2}\sum_{i,j}\left(\lambda_i - \lambda_j\right)\left(\lambda_i^* - \lambda_j^*\right)K(\mathbf{x}_i, \mathbf{x}_j) -$$

$$-\varepsilon\sum_i (\lambda_i + \lambda_i^*) + \sum_i y_i(\lambda_i - \lambda_i^*),$$

$$\text{s.t.} \quad \sum_i^{\ell} (\lambda_i - \lambda_i^*) = 0, \quad 0 \le \lambda_i \le C, \quad 0 \le \lambda_i^* \le C.$$

$$f(\mathbf{x}) = \left(\sum_{i \in S}\left(\lambda_i - \lambda_i^*\right)K\left(\mathbf{x}_i, \mathbf{x}\right)\right) + b \qquad b = y_i - \left(\sum_{i \in S}\left(\lambda_i - \lambda_i^*\right)K\left(\mathbf{x}_i, \mathbf{x}_{j|0<\lambda_j<C}\right)\right) - \varepsilon$$

SVR no lineal

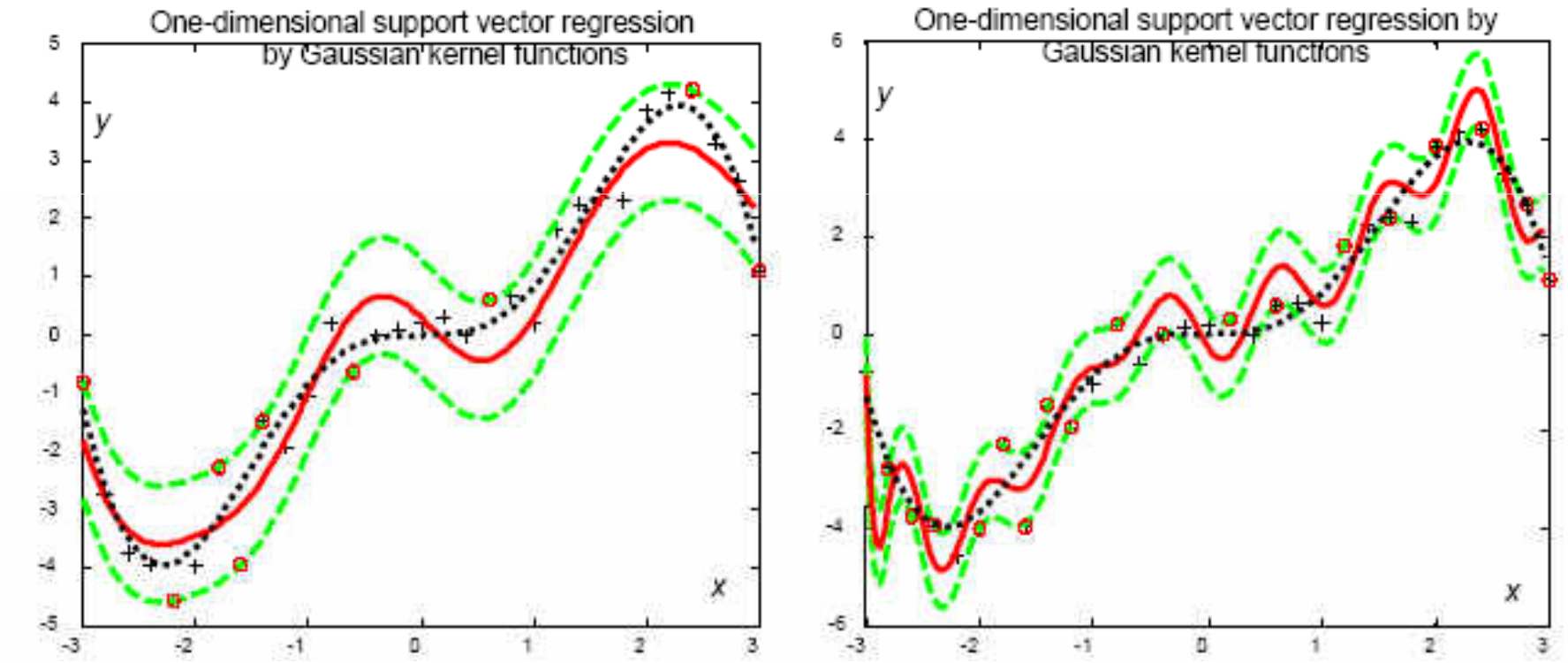


Figure 20 The influence of an insensitivity zone ε on the model performance. *A* nonlinear SVM creates a regression function *f* with Gaussian kernels and models a highly polluted (25% noise) function $x^2\sin(x)$ (dotted). 31 training data points (plus signs) are used. *Left*: $\varepsilon = 1$; 9 SVs are chosen (encircled plus signs). *Right*: $\varepsilon = 0.75$; the 18 chosen SVs produced a better approximation to noisy data and, consequently, there is the tendency of overfitting.

Una breve historia de SVM

- SVM se inventó en 1962
  - Pero en realidad no fue útil hasta la década de 1990
- No linealidad agregada en 1992.
  - Es decir, el truco del kernel.
- Márgenes suaves desarrollados en 1993
  - Aunque no se publicó hasta 1995
- SVM es relativamente nuevo
  - Sólo es útil si podemos entrenar eficientemente...

¿Algoritmo para entrenar SVM?

- Para entrenar SVM (lineal) debemos resolver un problema de **programación cuadrática** :

$$\text{Maximize: } L(\lambda) = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j z_i z_j (X_i \bullet X_j)$$

$$\text{Subject to: } C \geq \lambda_i \geq 0 \text{ for } i = 1, 2, \ldots, n \text{ and } \sum_{i=1}^{n} \lambda_i z_i = 0$$

- Una vez que se conocen los λ_i , clasificamos las muestras usando

$$f(X) = \sum_{i=1}^{s} \lambda_i z_i (X_i \bullet X) + b$$

¿Cómo resolver el problema de QP?

- Muchas técnicas generales disponibles para resolver problemas de QP:
  - Punto interior, gradiente conjugado,…
- Pero el problema de SVM es muy especial:
  - La solución es “dispersa” (muchos $\lambda_i = 0$)
  - Tenemos restricciones de desigualdad.
  - El problema es GRANDE (más sobre esto más adelante...)
- Entonces, SVM no es un caso de QP estándar.

Peculiaridades del entrenamiento SVM

- En comparación con los problemas de QP "estándar", SVM ***no*** requiere gran precisión
  - Verdadero para la mayoría de los algoritmos de ML
- La solución al problema de SVM es dispersa
  - Es decir, la mayoría de los λ_i serán 0…
  - …pero no sé cuál será 0 de antemano
- El número de muestras de entrenamiento ***y el tamaño*** de cada vector de entrenamiento pueden ser ENORMES

Solucionadores de QP

- Solucionadores generales de QP
  - Matriz del kernel precalculada

    $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$ para todo i y j
  - Almacenar toda la matriz **K** en la memoria
- Hace que los algoritmos sean eficientes...
- …pero no apto para entrenamiento SVM
- ¿Qué hacer?
  - Recuerde que la solución SVM es “dispersa”
  - ¿Podemos aprovechar la dispersión?

Primeros solucionadores SVM

- "...todos los primeros resultados de SVM se obtuvieron utilizando algoritmos ad hoc"
  - "Primeros" significa hasta mediados o finales de los años 1990
- Una idea: combinar la "fragmentación iterativa" con la "búsqueda de dirección simple"
- Fragmentar significa que solo tratamos con una parte de los datos de entrenamiento a la vez.
  - Luego optimizamos mediante búsqueda de dirección, según cálculos de gradiente.

Fragmentación iterativa

- No puedo abordar todo el problema a la vez
- Entonces, un posible plan de ataque es…
- Resolver SVM en un “conjunto de trabajo”, es decir, un subconjunto (o ***fragmento***) de los datos de entrenamiento
  - Consideramos los puntos de datos dentro del margen.
  - ¡Estos son vectores de apoyo a los candidatos!
  - Entonces, úselos como conjunto de trabajo y repita
- ¡Suena muy plausible!

SMO

- Los mejores solucionadores de SVM actuales utilizan SMO...
  - Optimización mínima secuencial (SMO)
  - Propuesto por primera vez en 1999
- Dividir en ***subproblemas mínimos*** QP
  - "Conjunto de trabajo" mínimo (solo 2 variables, es decir, puntos de datos, recuerde que estamos resolviendo λ_i)
- Facilita el cálculo de la dirección
- Las propiedades de convergencia/terminación de SMO son buenas y se comprenden bien.

SVM + y –

- Principales ventajas de las SVM:
  - Tiene fundamento teórico riguroso.
  - Realiza una clasificación con mayor precisión que la mayoría de métodos en aplicaciones, especialmente para datos de alta dimensión.
  - También se pueden aplicar a problemas de clasificación no lineal mediante el uso de funciones del núcleo. El “truco del kernel” permite que estos problemas sean manejables computacionalmente
  - Se pueden conectar diferentes núcleos a la misma maquina de aprendizaje y estudiarlos independientemente de ella.

SVM + y –

- Principales limitaciones de SVM:
  - Funciona sólo en un espacio de valor real:
    - Para un atributo categórico, necesitamos convertir sus valores categóricos a valores numéricos.
  - Sólo hace una clasificación de dos clases:
    - Para problemas de varias clases, se pueden aplicar algunas estrategias, por ejemplo, uno contra el resto.
  - El hiperplano producido por SVM es difícil de entender para los usuarios humanos.
    - El problema empeora con el kernel, por lo que SVM se usa comúnmente en aplicaciones que no requieren comprensión humana.
  - Dependen mucho de los datos de límites.

Referencias

- C. Bishop, Pattern Recognition and Machine Learning, Springer, 2006. *Chapter 7. Sparse Kernel Machines* (p. 325-339.
- R. Berwick, An idiot's guide to support vector machines
- E. Kim, Everything you wanted to know about the kernel trick (but were too afraid to ask)
- M. Law, A simple introduction to support vector machines
- W.S. Noble, What is a support vector machine?, Nature Biotechnology, 24(12):1565-1567, 2006

Referencias: Multiplicadores de Lagrange

- C. Bishop, Pattern Recognition and Machine Learning, Springer, 2006. *Appendix E. Lagrange Multipliers* (p. 707-710).
- D. Klein, Lagrange multipliers without permanent scarring
- Wikipedia, Lagrange multiplier

Referencias: Algoritmo SMO

- A. Ng, Simplified SMO algorithm, http://cs229.stanford.edu/materials/smo.pdf
- L. Bottou and C.-J. Lin, Support vector machine solvers, https://www.csie.ntu.edu.tw/~cjlin/papers/bottou_lin.pdf

Tema 4. Introducción a las Máquinas de Vectores Soporte

Pedro Antonio Gutiérrez
Grupo de investigación AYRNA
Universidad de Córdoba
Correo electrónico: pagutierrez@uco.es

17 de septiembre de 2024