Máquinas de Soporte Vectorial

Eduardo Morales, Hugo Jair Escalante

INAOE

Contenido

Máquinas de Soporte Vectorial

- Se presentó en COLT-92 (Boser, Guyon, Vapnik)
- Por un tiempo “desbancó” a las redes neuronales artificiales
- Herramienta popular de aprendizaje con buenos resultados
- Desarrollo teórico
- Robusto a problemas con muchas variables y pocos datos
- Popularizó el *kernel trick*

Máquinas de Soporte Vectorial

Se puede usar para:

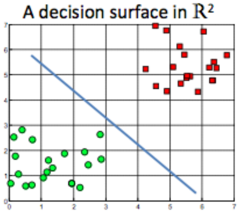
- Clasificación binaria (aplicación original)
- Clasificación multiclase
- Regresión
- Selección de variables
- Identificación de casos anómalos (*outliers*)
- *Clustering*

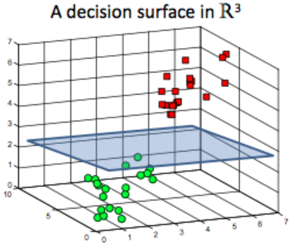
Máquinas de Soporte Vectorial

- Supongamos que tenemos un problema de clasificación binaria donde tenemos un conjunto de ejemplos $\mathcal{D} = \{(\vec{x}_1, y_1), \ldots, (\vec{x}_m, y_m)\}$ donde $y \in \{+1, -1\}$
- En ML, lo que queremos es generalizar para datos no vistos. Esto es, cuando nos llega un nuevo ejemplo $\vec{x}_i$, queremos saber si pertenece a una de dos posibles clases etiquetadas como $\{+1, -1\}$
- En términos generales, seleccionamos la clase y_i tal que $(\vec{x}_i, y_i)$ se parezca de alguna forma a los ejemplos de entrenamiento

Máquinas de Soporte Vectorial: Bases

- Los ejemplos los vamos a representar como vectores en un espacio m-dimensional (m = número de atributos)
- La ventaja de representarlos como vectores es que podemos usar una representación geométrica de lo que es una "superficie de decisión" que separa a dos grupos





Máquinas de Soporte Vectorial: Bases

Algunas operaciones:

- Dado un vector $\vec{a} = (a_1, a_2, \ldots, a_m)$
- L2-norma: $||\vec{a}||_2 = \sqrt{(a_1^2 + a_2^2 + \ldots + a_m^2)}$ (nos mide la longitud del vector)
- Producto punto: $\vec{a} \cdot \vec{b} = a_1 b_1 + a_2 b_2 + \ldots + a_m b_m = \sum_{i=1}^{m} a_i b_i$
- Notación equivalente: $< \mathbf{x}, \mathbf{x}' >= \sum_{i=1}^{N} [\mathbf{x}]_i [\mathbf{x}']_i$ donde $[\mathbf{x}]_i$ representa el i-ésimo elemento del vector $\mathbf{x}$

Máquinas de Soporte Vectorial: Bases

- La interpretación geométrica del producto punto es que calcula el coseno del ángulo entre los vectores $\vec{x_1}$ y $\vec{x_2}$ si están normalizados a una longitud de 1
- En general $< \vec{x_1}, \vec{x_2} >= (\vec{x_1} \cdot \vec{x_2}) = ||\vec{x_1}|| * ||\vec{x_2}|| cos(\theta)$
- Por lo mismo, cuando dos vectores son perpendiculares: $\vec{x_1} \cdot \vec{x_2} = 0$
- El producto punto se puede usar como medida de similaridad!

Máquinas de Soporte Vectorial: Bases

- El producto punto también nos sirve para calcular la longitud o norma del vector como: $||\vec{x}|| = \sqrt{< \vec{x}, \vec{x} >}$
- De la misma forma, la distancia entre dos vectores se puede calcular también como la longitud del vector diferencia

Máquinas de Soporte Vectorial: Kernels

- Definamos una medida de similitud genérica k, tal que dadas dos instancias, x y x', k nos regresa un valor real que caracteriza su similitud:

$$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$

$$(x, x') \to k(x, x')$$

con:

$$k(x, x') = k(x', x)$$

- A la función k la denominamos *kernel*

Máquinas de Soporte Vectorial: Bases

- Como $\mathcal{X}$ es el espacio de los *"objetos"*, necesitamos transformar los objetos originales a un espacio de producto punto $\mathcal{H}$
- Para ello, usamos el siguiente mapeo:

$$\Phi : \mathcal{X} \rightarrow \mathcal{H}$$

$$x \rightarrow \vec{x} := \Phi(x)$$

- Nota: Aun y cuando los *"objetos"* originales ya estén en $\mathbb{R}^m$ puede ser conveniente el mapeo.
- Así, podemos definir el siguiente kernel:

$$k(x, x') := \langle \vec{x}, \vec{x}' \rangle = \langle \Phi(x), \Phi(x') \rangle$$

SVM: Clasificador Simple

- Con estos elementos, podemos construir un clasificador simple. Una idea es tener dos clases y asignar la clase de un nuevo ejemplo al que tenga la media más cercana al ejemplo.
- La media de las dos clases es:

$$\mathbf{c}_+ = \frac{1}{m_+} \sum_{\{i|y_i=+1\}} \mathbf{x}_i$$

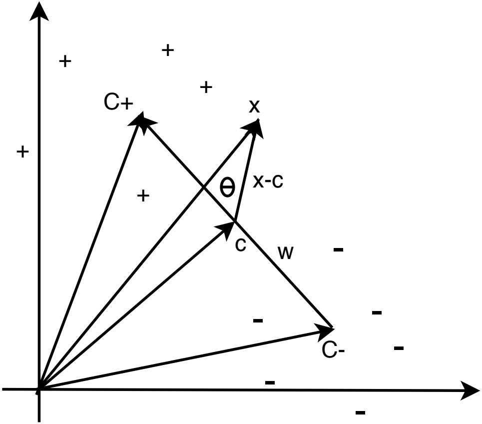
$$\mathbf{c}_- = \frac{1}{m_-} \sum_{\{i|y_i=-1\}} \mathbf{x}_i$$

donde m_+ y m_- son el número de ejemplos positivos y negativos

SVM: Clasificador Simple

- Asignamos la clase del nuevo punto a la clase promedio más cercana
- Se puede hacer en forma geométrica mediante el producto punto
- Sea $\mathbf{c}$ el punto medio entre las dos medias $\mathbf{c} = (\mathbf{c}_+ + \mathbf{c}_-)/2$
- Revisamos si el vector $\mathbf{x} - \mathbf{c}$ (que conecta a $\mathbf{c}$ con $\mathbf{x}$) tiene un ángulo menor a $\pi/2$ con el vector $\mathbf{w} = \mathbf{c}_+ - \mathbf{c}_-$ (que conecta a las dos medias)
- Si es menor se asigna la clase $+1$ y si es mayor, entonces se asigna -1

SVM: Clasificador Simple



SVM: Clasificador Simple

- Esto se puede hacer con el producto punto entre $(\mathbf{x} - \mathbf{c})$ y $(\mathbf{c}_{+} - \mathbf{c}_{-})$ ó $\mathbf{w}$
- Si el coseno es positivo el ángulo es menor a 90^{o} y si es negativo, el ángulo es mayor a 90^{o}.

SVM: Clasificador Simple

$$\begin{aligned}
y &= sgn < (\mathbf{x} - \mathbf{c}), \mathbf{w} > \\
&= sgn < (\mathbf{x} - (\mathbf{c}_+ + \mathbf{c}_-)/2, \mathbf{c}_+ - \mathbf{c}_- > \\
&= sgn((\mathbf{x} - \frac{\mathbf{c}_+ + \mathbf{c}_-}{2}) \cdot \mathbf{c}_+ - (\mathbf{x} - \frac{\mathbf{c}_+ + \mathbf{c}_-}{2}) \cdot \mathbf{c}_-) \\
&= sgn((\mathbf{x} \cdot \mathbf{c}_+) - (\mathbf{x} \cdot \mathbf{c}_-) + \frac{\mathbf{c}_+ + \mathbf{c}_-}{2} \cdot (\mathbf{c}_- - \mathbf{c}_+)) \\
&= sgn(< \mathbf{x}, \mathbf{c}_+ > - < \mathbf{x}, \mathbf{c}_- > + b)
\end{aligned}$$

- Donde: $b = \frac{1}{2}(||\mathbf{c}_-||^2 - ||\mathbf{c}_+||^2)$ y la norma $||\mathbf{x}|| = \sqrt{< \mathbf{x}, \mathbf{x} >}$
- b vale cero si las medias de las clases tienen la misma distancia al origen

SVM: Clasificador Simple

- Entre más pequeño el ángulo entre $\mathbf{x}$ y el centro de la clase, más grande es el coseno y la clase se vuelve positiva
- Lo anterior nos representa una frontera en forma de hiperplano que satisface una restricción que se puede expresar como una ecuación lineal

SVM: Clasificador Simple

- Si substituimos las definiciones de $\mathbf{c}_+$ y $\mathbf{c}_-$ en:

$$y = sgn(<\mathbf{x}, \mathbf{c}_+> - <\mathbf{x}, \mathbf{c}_-> + b)$$

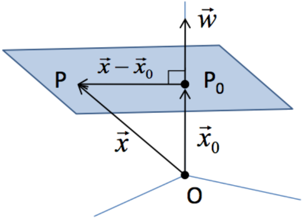
- Obtenemos:

$$y = sgn(\frac{1}{m_+} \sum_{\{i|y_i=+1\}} <\mathbf{x}, \mathbf{x}_i> - \frac{1}{m_-} \sum_{\{i|y_i=-1\}} <\mathbf{x}, \mathbf{x}_i> + b)$$

- Una solución en términos de producto punto!

Máquinas de Soporte Vectorial: Bases

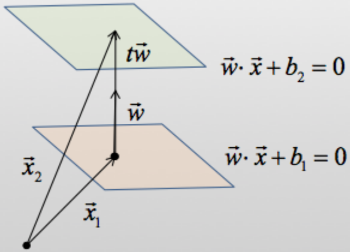
- La ecuación de un hiperplano la podemos definir con un punto P_0 y un vector perpendicular al plano $\vec{w}$ en ese punto:



- Si definimos el vector $\vec{x} = \vec{OP}$, donde P es un punto arbitrario en el hiperplano
- Una condición para que P esté en el plano es que el vector $\vec{x} - \vec{x_0}$ sea perpendicular a $\vec{w}$
- Esto es: $\vec{w} \cdot (\vec{x} - \vec{x_0}) = 0$ ó $(\vec{w} \cdot \vec{x} - \vec{w} \cdot \vec{x_0}) = 0$
- Si definimos: $b = -\vec{w} \cdot \vec{x_0}$, entonces: $\vec{w} \cdot \vec{x} + b = 0$

Máquinas de Soporte Vectorial: Bases

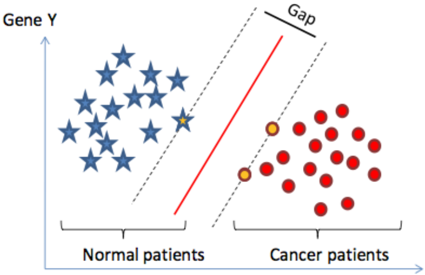
La distancia entre planos la podemos calcular como:



$$\vec{x}_2=\vec{x}_1+t\vec{w}$$
$$D=\|t\vec{w}\|=|t|\|\vec{w}\|$$
$$\vec{w}\cdot\vec{x}_2+b_2=0$$
$$\vec{w}\cdot(\vec{x}_1+t\vec{w})+b_2=0$$
$$\vec{w}\cdot\vec{x}_1+t\|\vec{w}\|^2+b_2=0$$
$$(\vec{w}\cdot\vec{x}_1+b_1)-b_1+t\|\vec{w}\|^2+b_2=0$$
$$-b_1+t\|\vec{w}\|^2+b_2=0$$
$$t=(b_1-b_2)/\|\vec{w}\|^2$$
$$\boxed{\Rightarrow D=|t|\|\vec{w}\|=|b_1-b_2|/\|\vec{w}\|}$$

Máquinas de Soporte Vectorial: Bases

- Representando los ejemplos como vectores
- Sabiendo como calcular hiperplanos
- Sabiendo como calcular la distancia entre planos
- La pregunta es ¿cómo podemos construir el hiperplano que separa dos clases con la máxima distancia entre los ejemplos de las clases?

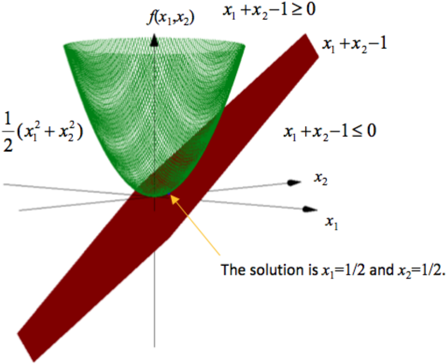


Máquinas de Soporte Vectorial: Bases

- Si tenemos una función convexa el mínimo local es igual al óptimo
- Un problema de optimización cuadrático es aquel en donde la función objetivo es cuadrática y está sujeta a restricciones lineales
- Estos problemas tienen un espacio de solución convexa y se pueden resolver de forma eficiente usando un enfoque voraz

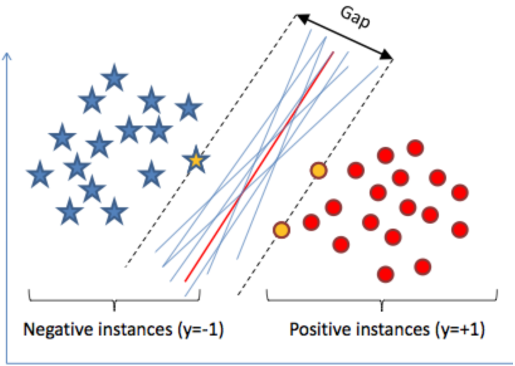
Máquinas de Soporte Vectorial: Bases

- Por ejemplo, si tenemos un problema con 2 atributos: $\vec{x} = (x_1, x_2)$
- El problema es: $\min \frac{1}{2}||\vec{x}||_2^2$ (o $\min \frac{1}{2}(x_1^2 + x_2^2)$) sujeto a $x_1 + x_2 - 1 \geq 0$

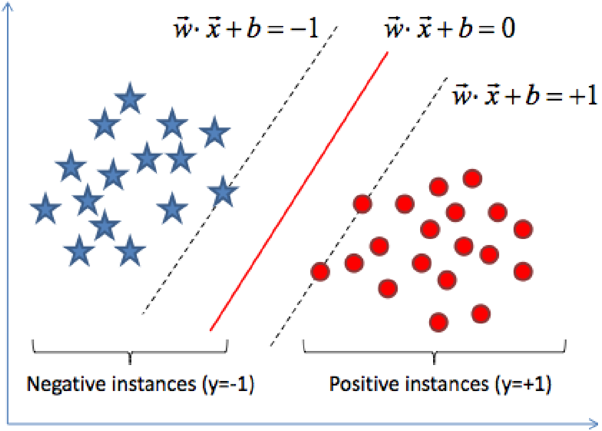


Máquinas de Soporte Vectorial

- Dado un conjunto de datos de entrenamiento: $\{\vec{x_1}, \vec{x_2}, \ldots, \vec{x_N}\} \in \mathbb{R}^M$ y $y_1, y_2, \ldots, y_N \in \{-1, +1\}$



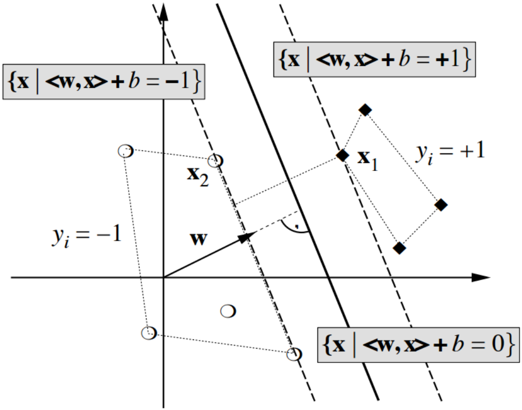
Máquinas de Soporte Vectorial



Máquinas de Soporte Vectorial

- Nos interesa la distancia entre los hiperplanos:
$$\vec{w} \cdot \vec{x} + b = -1, \vec{w} \cdot \vec{x} + b = +1$$
- O de forma equivalente:
$$\vec{w} \cdot \vec{x} + (b+1) = 0, \vec{w} \cdot \vec{x} + (b-1) = 0$$
- Sabemos que $D = \frac{b_1 - b_2}{||\vec{w}||}$
- Por lo tanto $D = 2/||\vec{w}||$
- Como queremos maximizar la separación, necesitamos minimizar: $||\vec{w}||$ o equivalentemente, minimizar $\frac{1}{2}||\vec{w}||^2$

Máquinas de Soporte Vectorial



Note:

$$\langle \mathbf{w}, \mathbf{x}_1 \rangle + b = +1$$
$$\langle \mathbf{w}, \mathbf{x}_2 \rangle + b = -1$$
$$\Rightarrow \quad \langle \mathbf{w}, (\mathbf{x}_1 - \mathbf{x}_2) \rangle = 2$$
$$\Rightarrow \quad \left\langle \frac{\mathbf{w}}{\|\mathbf{w}\|}, (\mathbf{x}_1 - \mathbf{x}_2) \right\rangle = \frac{2}{\|\mathbf{w}\|}$$

Máquinas de Soporte Vectorial

- También tenemos que poner restricciones para que los ejemplos esten bien clasificados:

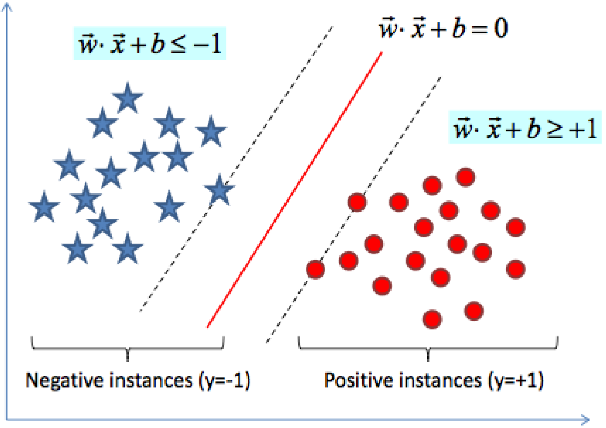
$$\vec{w} \cdot \vec{x}_i + b \leq -1, si : y_i = -1$$

$$\vec{w} \cdot \vec{x}_i + b \geq +1, si : y_i = +1$$

- De forma equivalente:

$$y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1$$

Máquinas de Soporte Vectorial



Máquinas de Soporte Vectorial

- Resumiendo, queremos:

$$\min \tau(\vec{w}) = \frac{1}{2}||\vec{w}||^2$$

s.t.

$$y_i(\vec{w} \cdot \vec{x_i} + b) \geq 1, para : i = 1, \ldots, N$$

- Dada una nueva instancia, $\vec{x}$, la clasificación es:

$$f(\vec{x}) = sign(\vec{w} \cdot \vec{x} + b)$$

Máquinas de Soporte Vectorial

- $\tau(\vec{w})$ es la función objetivo a optimizar, sujeta a N restricciones: $y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1$
- Se llama la formulacion primal de SVMs lineales
- Es un problema de optimización convexo de programación cuadrática (QO) con m variables $(w_i, i = 1, \ldots, m)$ donde m es el número de atributos en los datos

Máquinas de Soporte Vectorial

- La formulación lineal se puede resolver mediante multiplicadores de Lagrange, y entonces $\tau(\vec{w})$ se convierte en:

$$L(\vec{w}, b, \alpha) = \frac{1}{2}||\vec{w}||^2 - \sum_{j=1}^{N} \alpha_i (y_i(\langle x_i, w \rangle + b) - 1)$$

- $\vec{w}$ tiene m elementos (atributos) y $\vec{\alpha}$ tiene N elementos (ejemplos)
- Tenemos que minimizar el Lagrangiano con respecto a $\vec{w}, b$ y al mismo tiempo que las derivadas con respecto a $\vec{\alpha}$ sean cero

Máquinas de Soporte Vectorial

- Las derivadas de $L(\vec{w}, b, \alpha)$ con respecto a **w** y b son cero en el punto de silla (*saddle point*):

$$\frac{\partial L(\vec{w}, b, \alpha)}{\partial b} = 0$$

$$\frac{\partial L(\vec{w}, b, \alpha)}{\partial \mathbf{w}} = 0$$

- Lo que nos lleva a:

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

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i x_i$$

Máquinas de Soporte Vectorial

- Recordando, tenemos:

$$L(\vec{w}, b, \alpha) = \frac{1}{2}||\vec{w}||^2 - \sum_{i=1}^{N} \alpha_i (y_i(\langle x_i, w \rangle + b) - 1)$$

- Substituyendo $\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i x_i$ nos da:

$$L(\vec{w}, b, \alpha) = \frac{1}{2}(\sum_{i=1}^{m} \alpha_i y_i x_i) \cdot (\sum_{i=j}^{m} \alpha_j y_j x_j) - \sum_{i=1}^{N} \alpha_i (y_i (\sum_{i=j}^{m} \alpha_j y_j x_j) \cdot x_i + b) - 1$$

$$L(\vec{w}, b, \alpha) = \frac{1}{2} \sum_{i=1}^{m} \sum_{i=j}^{m} \alpha_i \alpha_j y_i y_j x_i \cdot x_j - \sum_{i=1}^{N} \alpha_i y_i (\sum_{i=j}^{m} \alpha_j y_j x_j) \cdot x_i + b) + \sum_{i=1}^{m} \alpha_i$$

Máquinas de Soporte Vectorial

$$
\begin{aligned}
L(\vec{w}, b, \alpha) = \quad & \tfrac{1}{2} \textstyle\sum_{i=1}^{m} \sum_{i=j}^{m} \alpha_i \alpha_j y_i y_j x_i \cdot x_j - \\
& \textstyle\sum_{i=1}^{N} \sum_{i=j}^{m} \alpha_i \alpha_j y_i y_j x_i \cdot x_j - \\
& b \textstyle\sum_{i=1}^{m} \alpha_i y_i + \sum_{i=1}^{m} \alpha_i
\end{aligned}
$$

Finalmente:

$$
L(\vec{w}, b, \alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} (\sum_{i=1}^{m} \sum_{i=j}^{m} \alpha_i \alpha_j y_i y_j x_i \cdot x_j)
$$

Máquinas de Soporte Vectorial

- Lo anterior está sujeto a:

$$\alpha_i \geq 0, \forall i = 1, \ldots, N, y \sum_{i=1}^{N} \alpha_i y_i = 0$$

- Donde queremos maximizar con respecto a $\vec{\alpha}$
- La decisión es:

$$f(\vec{x}) = sign(\sum_{i=1}^{N} \alpha_i y_i \vec{x}_i \cdot \vec{x} + b)$$

¿Porqué usar la formulación dual?

- No se requiere acceder a los datos originales sino sólo a los productos puntos:
  - Función objetivo: $\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \vec{x_i} \cdot \vec{x_j}$ sujeta a: $\alpha_i \geq 0$ y $\sum_{i=1}^{N} \alpha_i y_i = 0$ (restricciones)
  - Solución: $f(\vec{x}) = sign(\sum_{i=1}^{N} \alpha_i y_i \vec{x_i} \cdot \vec{x} + b)$
- El número de variables libres está acotado por el número de vectores de soporte y no por el número de atributos

El método del Lagrangiano

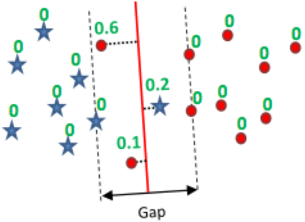
- Se usa el método dual para resolver problemas de optimización con restrucciones de desigualdad
- El lagrangiano minimiza con respecto a las variables primales $\vec{w}, b$ y maximiza con respecto a las variables duales $\vec{\alpha}$
- Esto crea un punto silla, en este punto las derivadas del lagrangiano con respecto a las variables primales debe de ser cero (condiciones Karush-Kuhn-Tucker o KKT)

SVM resumen

- SVM es un clasificador que intenta encontrar el híper plano que maximiza la separación de ejemplos de ambas clases
- Se minimiza $||\vec{\mathbf{w}}||$, sujeto a clasificar correctamente todos los ejemplos
- El híper plano de decisión depende únicamente de algunos ejemplos
- En la formulación dual se trabaja con productos punto o funciones de kernel
- Garantizado encontrar el óptimo si los datos son linealmente separables

SVM para datos no separables linealmente

- Qué hacer si los datos no son linealmente separables?



- Una posibilidad es usar variables de holgura (*slack variables*) dejando *soft-margins*
- Al asignar variables de holgura a cada instancia $\psi_i \geq 0$ se puede pensar como la distancia que separa el hiperplano de la instancia que está mal clasificada (o 0 si está bien clasificada)

SVM para datos no separables linealmente

- El problema es entonces:

$$\min \frac{1}{2}||\vec{w}||^2 + C \sum_{i=1}^{N} \psi_i$$

- sujeto a:

$$y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1 - \psi_i, para, i = 1, \ldots, N$$

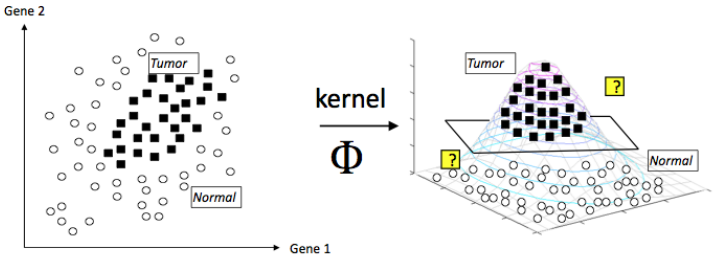
- La forma dual es:

$$\min \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \vec{x}_i \cdot \vec{x}_j$$

- sujeto a:

$$0 \leq \alpha_i \leq C, y \sum_{i=1}^{N} \alpha_i y_i = 0$$

Kernel Trick

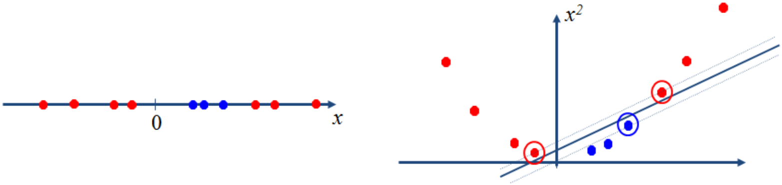


- Datos que no son linealmente separables en el espacio generado por los atributos de entrada pueden ser linealmente separables en el espacio de los atributos que se obtienen a aplicar un *kernel*

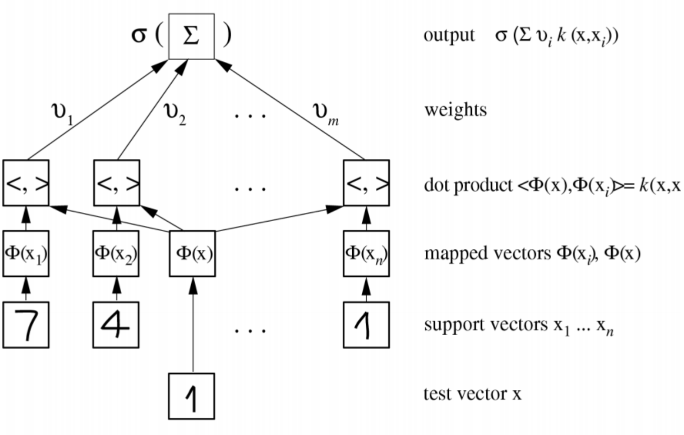
Kernel Trick

- Datos originales: $f(\vec{x}) = sign(\vec{w} \cdot \vec{x} + b)$ con $\vec{w} = \sum_{i=1}^{N} \alpha_i y_i \vec{x}_i$
- Datos en un espacio de atributos de mayor dimensión $\Phi(\vec{x})$:
  - $f(\vec{x}) = sign(\vec{w} \cdot \Phi(\vec{x}) + b)$
  - $\vec{w} = \sum_{i=1}^{N} \alpha_i y_i \Phi(\vec{x}_i)$
  - $f(\vec{x}) = sign(\sum_{i=1}^{N} \alpha_i y_i \Phi(\vec{x}_i) \cdot \Phi(\vec{x}) + b)$
  - $f(\vec{x}) = sign(\sum_{i=1}^{N} \alpha_i y_i K(\vec{x}_i, \vec{x}) + b)$
- Por lo que no necesitamos saber Φ directamente, sólo la función kernel $K(\cdot, \cdot) : R^N \times R^N \rightarrow R$

Kernel trick: Ejemplo



Kernel trick



Kernel Trick

- Un kernel es el producto punto en un espacio de atributos: $K(\vec{x_i}, \vec{x_j}) = \Phi(\vec{x_i}) \cdot \Phi(\vec{x_j})$
- Ejemplos:

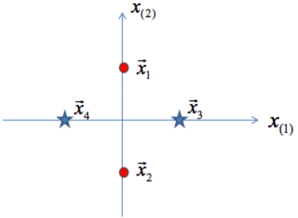
$K(\vec{x_i}, \vec{x_j}) = \vec{x_i} \cdot \vec{x_j}$	kernel lineal
$K(\vec{x_i}, \vec{x_j}) = exp(-\gamma \vert\vert\vec{x_i} \cdot \vec{x_j}\vert\vert^2)$	kernel gaussiano
$K(\vec{x_i}, \vec{x_j}) = exp(-\gamma \vert\vert\vec{x_i} \cdot \vec{x_j}\vert\vert)$	kernel exponential
$K(\vec{x_i}, \vec{x_j}) = (p + \vec{x_i} \cdot \vec{x_j})^q$	kernel polinomial
$K(\vec{x_i}, \vec{x_j}) = (p + \vec{x_i} \cdot \vec{x_j})^q exp(-\gamma \vert\vert\vec{x_i} \cdot \vec{x_j}\vert\vert^2)$	kernel híbrido
$K(\vec{x_i}, \vec{x_j}) = tanh(k\vec{x_i} \cdot \vec{x_j} - \delta)$	kernel sigmoidal

Kernel Trick

- Si consideramos un kernel polinomial de grado 3: $K(\vec{x}_i, \vec{x}_j) = (p + \vec{x}_i \cdot \vec{x}_j)^3$
- Si tenemos datos representados en dos dimensiones $\{x_1, x_2\}$
- Al aplicar el kernel nos queda un espacio de 10 dimensiones: $\{1, x_1, x_2, x_1x_2, x_1^2, x_2^2, x_1x_2^2, x_1^2x_2, x_1^3, x_2^3\}$

Ejemplo

- Datos que no son linealmente separables en un espacio (R^2) podemos hacerlos separables al mapearlos a otro espacio

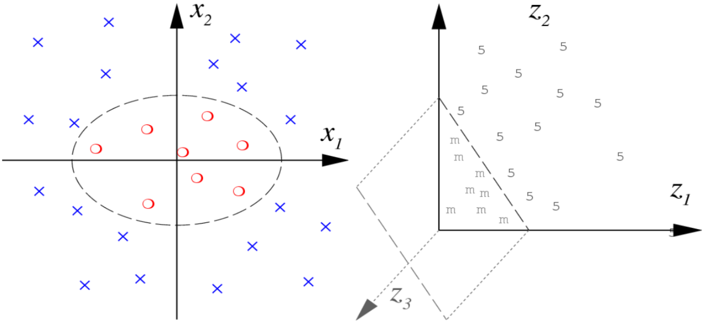


$$K(\vec{x},\vec{z}) = (\vec{x}\cdot\vec{z})^2 = \left[\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \cdot \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}\right]^2$$

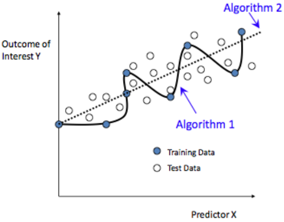
Ejemplo

- $= [x_1 z_1 + x_2 z_2]^2 = x_1^2 z_1^2 + 2x_1 z_1 x_2 z_2 + x_2^2 z_2^2$
- $= \begin{pmatrix} x_1^2 \\ \sqrt{2} x_1 x_2 \\ x_2^2 \end{pmatrix} \cdot \begin{pmatrix} z_1^2 \\ \sqrt{2} z_1 z_2 \\ z_2^2 \end{pmatrix}$
- $= \Phi(\vec{x}) \cdot \Phi(\vec{z})$
- El mapeo es: $\Phi(\vec{x}) = \begin{pmatrix} x_1^2 \\ \sqrt{2} x_1 x_2 \\ x_2^2 \end{pmatrix}$

Resultado:



Regularización



- Muchos algoritmos buscan una función de decisión resolviendo el siguiente problema de optimización: minimizar(Pérdida + λ Penalización)
- Pérdida (*loss*) = mide el error de ajuste en los datos
- Penalización = penaliza la complejidad del modelo
- λ = un parámetro de regularización que balancea *Error* y *Complejidad*

Regularización

- Existen diferentes modelos

Loss function	Penalty function	Resulting algorithm
Hinge loss: $\sum_{i=1}^{N}[1-y_i f(\vec{x}_i)]_+$	$\lambda\|\vec{w}\|_2^2$	SVMs
Mean squared error: $\sum_{i=1}^{N}(y_i - f(\vec{x}_i))^2$	$\lambda\|\vec{w}\|_2^2$	Ridge regression
Mean squared error: $\sum_{i=1}^{N}(y_i - f(\vec{x}_i))^2$	$\lambda\|\vec{w}\|_1$	Lasso
Mean squared error: $\sum_{i=1}^{N}(y_i - f(\vec{x}_i))^2$	$\lambda_1\|\vec{w}\|_1 + \lambda_2\|\vec{w}\|_2^2$	Elastic net
Hinge loss: $\sum_{i=1}^{N}[1-y_i f(\vec{x}_i)]_+$	$\lambda\|\vec{w}\|_1$	1-norm SVM

La función de pérdida o error (*loss*)

- $\sum_{i=1}^{N}[1 - y_i f(\vec{x_i})]_+$ donde $[...]_+$ indica la parte positiva
- La función de pérdida es diferente de cero si $1 - y_i f(\vec{x_i}) > 0$, o $y_i f(\vec{x_i}) < 1$
- Como $y_i = \{-1, +1\}$, la funcion de pérdida es no cero si: $f(\vec{x_i}) < 1$ para $y_i = +1$ o $f(\vec{x_i}) > -1$ para $y_i = -1$
- Esto es, para:
  - $\vec{w} \cdot \vec{x_i} + b < 1$ para $y_i = +1$
  - $\vec{w} \cdot \vec{x_i} + b > -1$ para $y_i = -1$

Riesgo empírico y riesgo real

- Para medir qué tan bien clasifica una función podemos usar lo que se conoce como *zero-one loss function*:

$$c(x, y, f(x)) = \frac{1}{2}|f(x) - y|$$

donde la "pérdida" es 0 si clasifica correctamente y 1 si no (recordemos que las clases pueden ser ± 1)

- Podemos tomar ésto para todos los datos y promediar el resultado:

$$R_{emp}[f] = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}|f(x_i) - y_i|$$

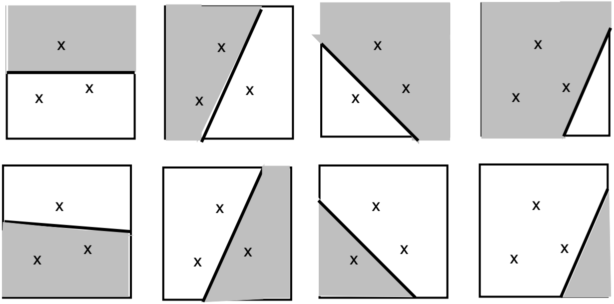
- Valores pequeños en el error de prueba o riesgo empírico (*empirical risk*) no necesariamente implican un error real ($R[f]$)

Dimensión VC

- Cada función que se elija separa los ejemplos de cierta forma. Como tenemos una etiquetación de ± 1, existen a lo más 2^m etiquetas para m ejemplos
- Una clase de funciones suficientemente expresiva podría generar las 2^m particiones. Si ese es el caso, se dice que esa clase de funciones despedaza o *shatters* los m ejemplos
- La dimensión VC se define como la m más grande tal que existe un conjunto de m puntos que la clase puede despedazar
- Se puede ver como un número que define la *capacidad* de un sistema de aprendizaje

Dimensión VC

- Por ejemplo, para 3 puntos y 2 clases en un plano existen 8 posibles asignaciones y una recta las puede generar; para 4 puntos ya no se puede usar una recta



Dimensión VC

- Si $h < m$ es la dimensión VC (Vapnik-Chervonenkis) de una clase de funciones que el algoritmo de aprendizaje implementa, entonces todas las funciones de la clase, independientemente de la distribución P que genera los datos, cumplen con la siguiente cota con probabilidad $1 - \delta$ sobre los datos de entrenamiento:

$$R[f] \leq R_{emp}[f] + \phi(h, m, \delta)$$

donde el término de confianza (*confidence*) o de capacidad (*capacity*) ϕ se define como:

Máquinas de Soporte Vectorial

$$\phi(h, m, \delta) = \sqrt{\frac{1}{m}\left(h\left(ln\frac{2m}{h} + 1\right) + ln\frac{4}{\delta}\right)}$$

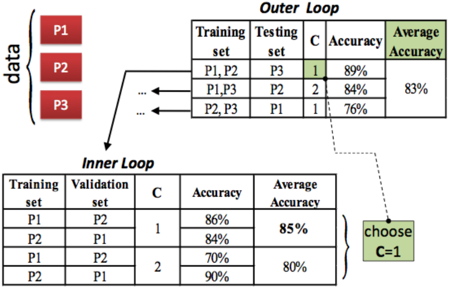
- Lo que se busca es seleccionar una clase de funciones suficientemente restrictiva (y por lo tanto simple) que al mismo tiempo tenga la capacidad de modelar las dependencias que existen en $P(x, y)$

Extensiones

- Selección de modelos
- SVM para problemas multiclase (no sólo binarios)
- *Support Vector Regression*
- Detección de “novedad” o *outliers*
- *Clustering*
- Selección de variables
- Cálculo de probabilidades de un clasificador SVM

Selección de Modelos

- Seleccionar qué Kernel usar y con qué parámetros
- Por ejemplo, qué valor usar para “C” y cuál para “p”
- Algunos autores han usado un validación cruzada anidada:



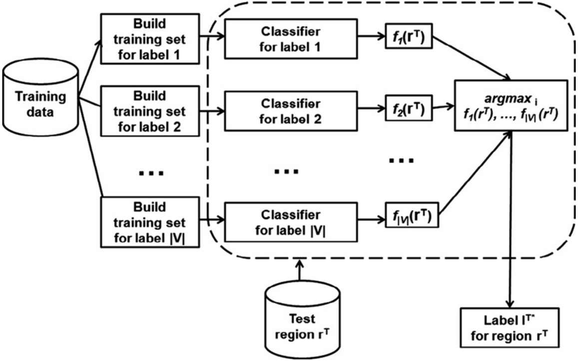
Outer Loop

Training set	Testing set	C	Accuracy	Average Accuracy
P1, P2	P3	1	89%	83%
P1,P3	P2	2	84%	
P2, P3	P1	1	76%	

Inner Loop

Training set	Validation set	C	Accuracy	Average Accuracy
P1	P2	1	86%	85%
P2	P1		84%	
P1	P2	2	70%	80%
P2	P1		90%	

SVM para multiclases

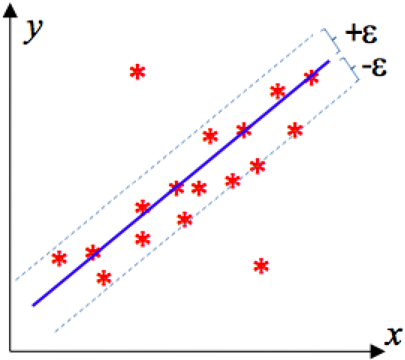


SVM para multiclases

- Si existe más de una clase, si buscamos un hiperplano que separe cada clase de las demás, pueden existir áreas en donde no queda clara cual sería la clasificación
- Algunos construyen un grafo de decisión que toma en cuenta las diferentes hiperplanos de decisión, pero el orden del grafo importa
- Se han propuesto otros metodos para problemas multiclase

SVM para regresión (ϵ-*SVM regression*)

- La idea es encontrar una función $f(\vec{x}) = \vec{w} \cdot \vec{x} + b$ que se acerque $y_1, \ldots, y_N$ lo más posible con un error de hasta ϵ

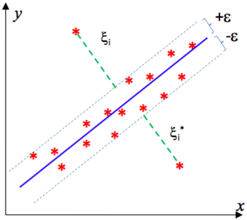


SVM para regresión (ϵ-*SVM regression*)

- Encontrar: $f(\vec{x}) = \vec{w} \cdot \vec{x} + b$
- minimizando: $\frac{1}{2}||\vec{w}||^2$
- Sujeto a las restricciones:
  $y_i - (\vec{w} \cdot \vec{x} + b) \leq \epsilon$
  $y_i - (\vec{w} \cdot \vec{x} + b) \geq -\epsilon$

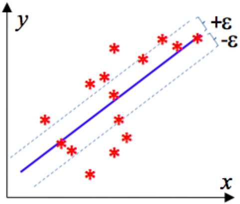
SVM para regresión (ϵ-*SVM regression*)

- Si tenemos valores lejanos, podemos introducir variables de holgura y penalizar a las soluciones que las contengan
- Encontrar: $f(\vec{x}) = \vec{w} \cdot \vec{x} + b$
- Minimizando: $\frac{1}{2}||\vec{w}||^2 + C\sum_{i=1}^{N} \psi_i$
- Sujeto a las restricciones:
  $y_i - (\vec{w} \cdot \vec{x} + b) \leq \epsilon + \psi_i$
  $y_i - (\vec{w} \cdot \vec{x} + b) \geq -\epsilon - \psi_i^*$

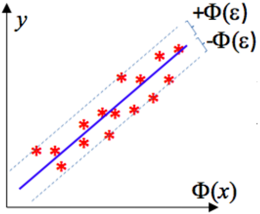


SVM para regresión (ϵ-*SVM regression*)

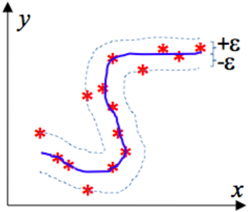
- Si no se aproxima bien una función para un ϵ pequeño, entonces se puede usar un kernel



Con kernel $\Phi(\epsilon)$:



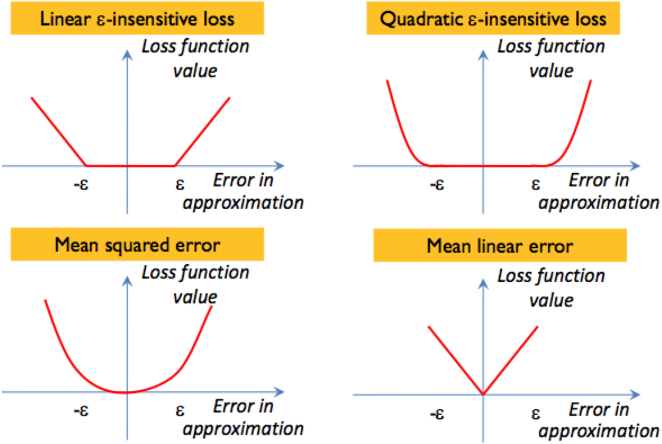
Y al revés:



Función de pérdida para ϵ-*SVM regression*

Loss function	Penalty function	Resulting algorithm
Linear ε-insensitive loss: $\sum_{i=1}^{N} \max(0, \lvert y_i - f(\vec{x}_i)\rvert - \varepsilon)$	$\lambda \lVert \vec{w} \rVert_2^2$	ε-SVR
Quadratic ε-insensitive loss: $\sum_{i=1}^{N} \max(0, (y_i - f(\vec{x}_i))^2 - \varepsilon)$	$\lambda \lVert \vec{w} \rVert_2^2$	Another variant of ε-SVR
Mean squared error: $\sum_{i=1}^{N} (y_i - f(\vec{x}_i))^2$	$\lambda \lVert \vec{w} \rVert_2^2$	Ridge regression
Mean linear error: $\sum_{i=1}^{N} \lvert y_i - f(\vec{x}_i)\rvert$	$\lambda \lVert \vec{w} \rVert_2^2$	Another variant of ridge regression

Función de pérdida para ϵ-*SVM regression*



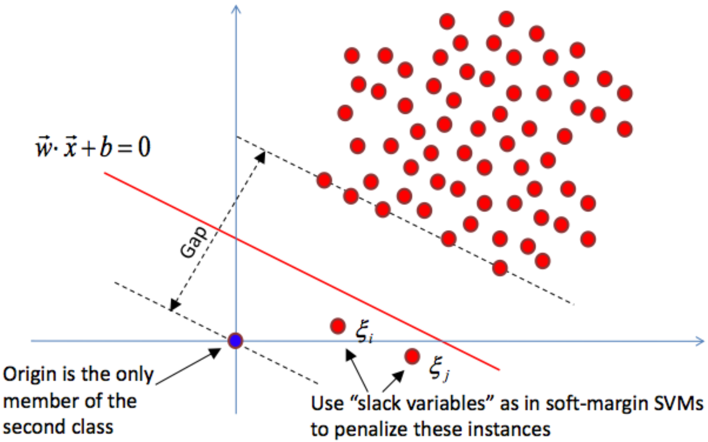
SVM para una clase

- La idea es encontrar la región más compacta que contenga a la mayoría de los ejemplos
- Encontrar una función de decisión que tome valores $+1$ dentro de la región y -1 fuera
- Nos puede servir para encontrar *outliers*

SVM para una clase

- No sabemos la clase/etiqueta de los ejemplos
- Todos los ejemplos positivos son parecidos, pero los negativos pueden ser diferentes entre sí
- Encontrar el hiperplano con máxima separación entre los datos y el origen

SVM para una clase



SVM para una clase

- De nuevo, encontrar: $f(\vec{x}) = sign(\vec{w} \cdot \vec{x} + b)$
- minimizar $\frac{1}{2}||\vec{w}||^2 + \frac{1}{\nu N} \sum_{i=1}^{N} \psi_i + b$ sujeta a: $(\vec{w} \cdot \vec{x} + b) \geq -\psi_i$ para $i = 1, \ldots, N$
- ν es la máxima fracción de *outliers* (o puntos fuera de la frontera) que se permiten en los datos

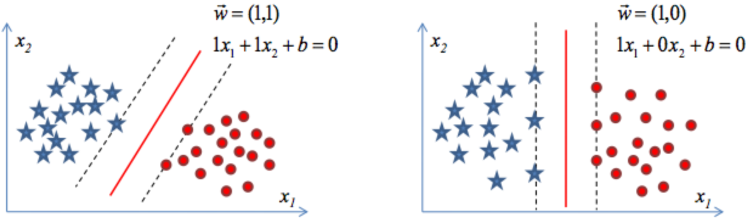
SVM para una clase

- Se puede hacer también su formulación dual y aplicar el “kernel trick”
- El parámetro ν afecta de manera importante la superficie de decisión
- La selección del origen es arbitraria y también afecta el resultado del algoritmo

SVM para seleccionar atributos

- En la formulación original de SVMs el vector de pesos $\vec{w}$ tiene tantos elementos como existen atributos en los datos
- La magnitud de cada elemento nos denota la importancia que tiene cada atributo en la tarea

SVM para seleccionar atributos



- En la figura del lado derecho x_1 y x_2 son igualmente importantes, en la de la izquierda sólo x_1 es importante

SVM para seleccionar atributos

Algoritmo:

- Entrena el SVM para clasificar los datos
- Ordena los atributos con base en la magnitud del vector de pesos $\vec{w}$
- Selecciona el subconjunto más pequeño con buena predicción

SVM para seleccionar atributos

- Los pesos no son localmente consistentes, por lo que al quitar una variable puede afectar los pesos originales
- En general se tiene que volver a estimar el vector $\vec{w}$ para decidir cuál sería el siguiente atributo a eliminar
- Al aplicar un kernel se puede tener un número grande de atributos y esta selección se vuelve más relevante
- Se puede combinar con otro metodos (e.g., seleccionar la cobija de Markov)

Otros desarrollos (Clustering y Probabilidad)

- Existen algoritmos para aplicar SVM en “clustering”
- En clasificadores SVM se puede estimar la probabilidad de salida del clasificador (calculando la distancia al hiperplano, obteniendo “bins” con frecuencia en los ejemplos de entrenamiento que reflejen la probabilidad)

Conclusiones

- Los SVMs tienen en general muy buenos resultados
- El “kernel trick” permite aprender funciones complejas no lineales y seguir resolviendo un problema de optimización cuadrático convexo
- Es robusto a ruido (variables de holgura) y la solución se define sólo por un subconjunto pequeño de puntos (vectores de soporte)