SVM – Support Vector Machines (Máquinas de Vectores Soporte)

Francisco José Ribadas Pena

ribadas@uvigo.es

28 de abril de 2021

6.1 Introducción

SVM *(Support Vector Machines)*, máquinas de vectores soporte

- Pertenece a la familia de métodos *kernel machines*
- Método de $\begin{cases} \text{clasificación (binaria)} \\ \text{regresión (predicción numérica)} \end{cases}$

Principios básicos

1. Uso de clasificadores lineales de "margen máximo"
2. Uso de <u>funciones kernel</u>
   - "describen" el problema en un "espacio de caracteríticas" de mayor dimensión
   - permiten aplicar "algoritmos lineales" sobre "problemas no lineales"

https://scikit-learn.org/stable/modules/svm.html

PREVIO

- Producto escalar de 2 vectores
  Con $\vec{x} = \{x_1, x_2, ..., x_n\}$ y $\vec{z} = \{z_1, z_2, ..., z_n\}$

$$\cdot : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$$

$$\vec{x} \cdot \vec{z} = \sum_{i=1}^{n} x_i z_i = x_1 z_1 + x_2 z_2 + ... + x_n z_n$$

- Norma de un vector (módulo)

$$\|\vec{x}\| = \sqrt{\vec{x} \cdot \vec{x}} = \sqrt{\sum_{i=1}^{n} x_i^2} = \sqrt{x_1^2 + x_2^2 + ... + x_n^2}$$

6.2 Clasificadores lineales

Espacio de entrada: X ($\equiv \mathbb{R}^n$, dimensión n)

Espacio de salida: $\mathrm{Y} = \{-1, +1\}$

- Cada ejemplo de entrenamiento será un par $(\vec{x}_i, y_i)$ con $\begin{cases} \vec{x}_1 \in \mathrm{X} \\ y_i \in \mathrm{Y} \end{cases}$
- Conjunto de entrenamiento $L = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots, (\vec{x}_l, y_l)\}$

Objetivo:

Encontrar un hiperplano $\mathbf{h}$ de dimensión $(n-1)$ que separe los ejemplos etiquetados con -1 de los etiquetados con $+1$ con un "margen máximo"

Espacio de hipótesis (H):

Conjunto de hiperplanos de decisión definidos por $\begin{cases} \vec{w} \text{ (≈ vector de "pesos")} \\ b \text{ (≈ umbral)} \end{cases}$

$H : \mathbb{R} \to \mathrm{Y}$

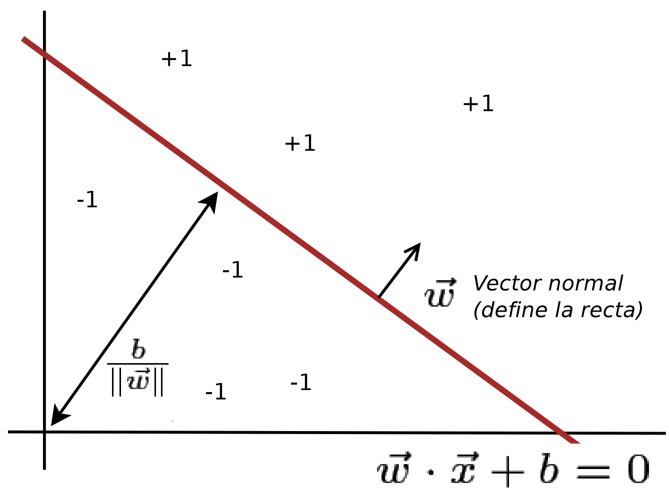
$h(\vec{x}) = signo\left(\sum_{i=1}^{n} w_i x_i + b\right) = \begin{cases} +1 & \text{si } \sum_{i=1}^{n} w_i x_i + b > 0 \\ -1 & \text{en otro caso} \end{cases}$

Reescrito en forma de producto escalar: $h(\vec{x}) = signo\,(\vec{w} \cdot \vec{x} + b)$

Hiperplano de decisión definido por

la ecuación $\boxed{\sum_{i=1}^{n} w_i x_i + b = 0}$

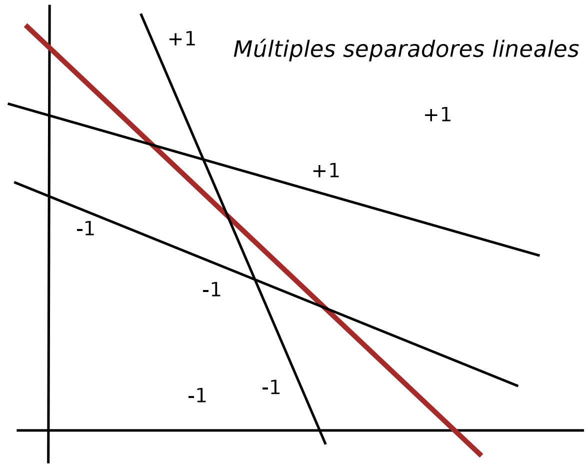
(ó $\underline{\underline{\vec{w} \cdot \vec{x} + b = 0}}$ en forma vectorial)



6.2 Margen máximo y vectores soporte

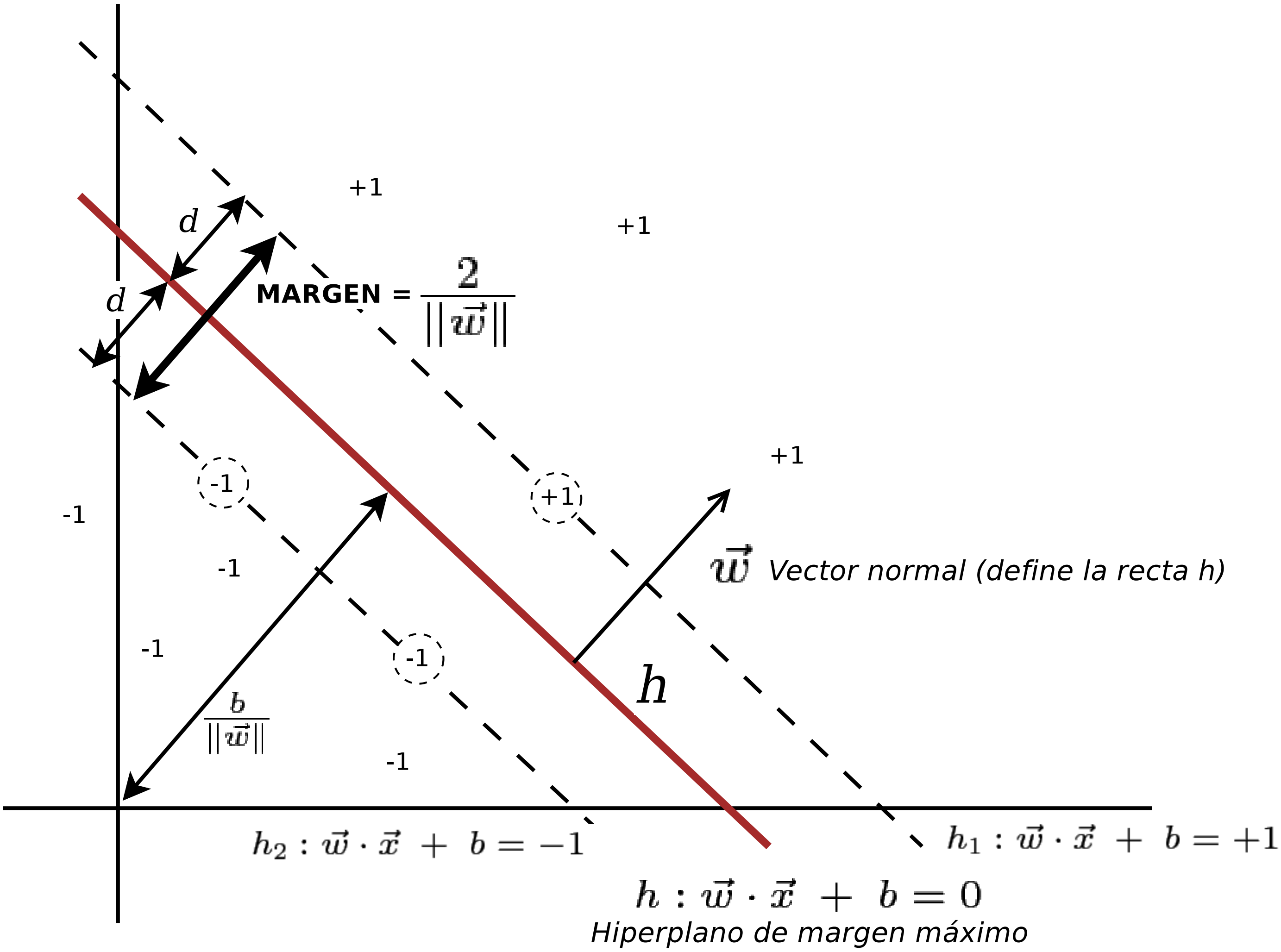
Si el problema (definido por el conjunto de ejemplos L) es linealmente separable existen infinitos hiperplanos que separan los ejemplos de entrenamiento.

- Existen algoritmos para encontrar/construir esos hiperplanos
- Ejemplo: algoritmo de aprendizaje de perceptrones simples



Nos interesará el hiperplano que mejor separe los ejemplos de entrenamiento (minimiza las posibilidades de sobreajuste)

Objetivo: buscar/construir el hiperplano de margen máximo



Donde tenemos:

$$h_1: \quad \vec{w} \cdot \vec{x} \ + \ b = +1 \quad \text{(delimita ejemplos +1)}$$

$$h_2: \quad \vec{w} \cdot \vec{x} \ + \ b = -1 \quad \text{(delimita ejemplos -1)}$$

y el hiperplano de margen máximo definido por la ecuación:

$$h: \quad \vec{w} \cdot \vec{x} \ + \ b = 0$$

- Los **vectores soporte** son aquellos ejemplos de entrenamiento que definen los hiperplanos de separación h_1 y h_2 (señalados con un círculo en la figura anterior).

Como $\mathrm{Y} = \{-1, +1\}$, el ejemplo de entrenamiento $(\vec{x}_i, y_i) \in L$ estará bien clasificado si se verifica:

$$\begin{array}{l} \vec{w} \cdot \vec{x}_i \ + \ b \geq +1 \quad \text{para } y_i = +1 \\ \text{ó} \\ \vec{w} \cdot \vec{x}_i \ + \ b \leq -1 \quad \text{para } y_i = -1 \end{array}$$

Ambas expresiones pueden combinarse de la forma:

$$\boxed{y_i(\vec{w} \cdot \vec{x}_i \ + \ b) \geq 1 \ \ \forall (\vec{x}_i, y_i) \in L}$$

$\Longrightarrow$ Este será el conjunto de restricciones que deberá cumplir el hiperplano objetivo h

Puede demostrarse que la distancia entre el hiperplano objetivo h y cada hiperplano "separador", h_1 y h_2, es $\frac{1}{\|\vec{w}\|}$, por lo que el **margen** es $\boxed{\dfrac{2}{\|\vec{w}\|}}$

Objetivo: Buscar los valores de $\vec{w}$ y b que:

1. maximicen el margen (distancia entre h_1 y h_2 [$=\frac{2}{\|\vec{w}\|}$])
2. garanticen que se clasifiquen correctamente todos los ejemplos del conjunto de entrenamiento $L = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots, (\vec{x}_l, y_l)\}$

Por conveniencia matemática, en los clasificadores SVM se utiliza la siguiente equivalencia.

$$\textsc{Maximizar}\ \frac{2}{\|\vec{w}\|} \quad \equiv \quad \textsc{Minimizar}\ \frac{1}{2}\|\vec{w}\|^2$$

ENUNCIADO (forma primal)

Dado un conjunto de ejemplos de entrenamiento previamente clasificados $L = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots, (\vec{x}_l, y_l)\}$.
Encontrar b y $\vec{w}$ que:

$$\begin{array}{ll} \text{MINIMICEN} & \frac{1}{2}\|\vec{w}\|^2 \qquad \text{(equivale a maximizar el margen)} \\ \text{SUJETO A :} & y_1(\vec{w} \cdot \vec{x}_1 + b) \geq 1 \\ & y_2(\vec{w} \cdot \vec{x}_2 + b) \geq 1 \\ & \ldots \\ & y_l(\vec{w} \cdot \vec{x}_l + b) \geq 1 \end{array}$$

(todos los ejemplos correctamente clasificados)

Es un problema de optimización cuadrática (*Quadratic Programming (QP)*).

- Se trata de identificar los parámetros que optimicen (maximicen ó minimicen) una ecuación de segundo grado sujetos a una serie de restriciones lineales sobre dichos parámetros.
- Existen algoritmos razonablemente eficientes para resolverlos, tanto de forma exacta como aproximada.
  - Weka usa el método SMO (*Sequential Minimal Optimization*)
    J. Platt: *Fast Training of Support Vector Machines using Sequential Minimal Optimization*. In Advances in Kernel Methods Support Vector Learning, 1998.

6.2.1 Forma dual

En la práctica se usa la *forma dual* del problema de optimización anterior.

Permite expresar el problema de optimización en función de productos escalares entre los vectores de entrenamiento (necesario para poder aplicar funciones *kernel*)

Reformulación:

- $\vec{w}$ puede expresarse como una combinación lineal de los ejemplos de entrenamiento $[(\vec{x}_i, y_i) \in L]$ en la forma: $\boxed{\vec{w} = \sum_{i=1}^{l} \alpha_i y_i \vec{x}_i}$
  - Cada ejemplo de entrenamiento $(\vec{x}_i, y_i) \in L$ tiene asociada una variable α_i que describe su *"influencia"* en el hiperplano de margen máximo.
  - Sólo los vectores soporte participan en la definición del vector $\vec{w}$
    - $\alpha_i > 0$ para los $\vec{x}_i$ que sean vectores soporte
    - $\alpha_i = 0$ para los $\vec{x}_i$ que no sean vectores soporte
- El hiperplano de margen máximo quedaría definido por la ecuación:

$$h: \quad \vec{w} \cdot \vec{x} + b = \sum_{i=1}^{l} \alpha_i y_i \underline{\underline{(\vec{x}_i \cdot \vec{x})}} + b$$

ENUNCIADO (forma dual)

Dado un conjunto de ejemplos de entrenamiento previamente clasificados $L = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), ..., (\vec{x}_l, y_l)\}$.
Encontrar los valores $\alpha_1, \alpha_2, \ldots, \alpha_l$ que:

$$\begin{array}{ll} \text{MAXIMICEN} & \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j (\vec{x}_i \cdot \vec{x}_j) \\ \text{SUJETO A}: & \sum_{i=1}^{l} \alpha_i y_i = 0 \\ & \text{y} \\ & \alpha_i \geq 0 \ \forall i \in \{1, \ldots, l\} \end{array}$$

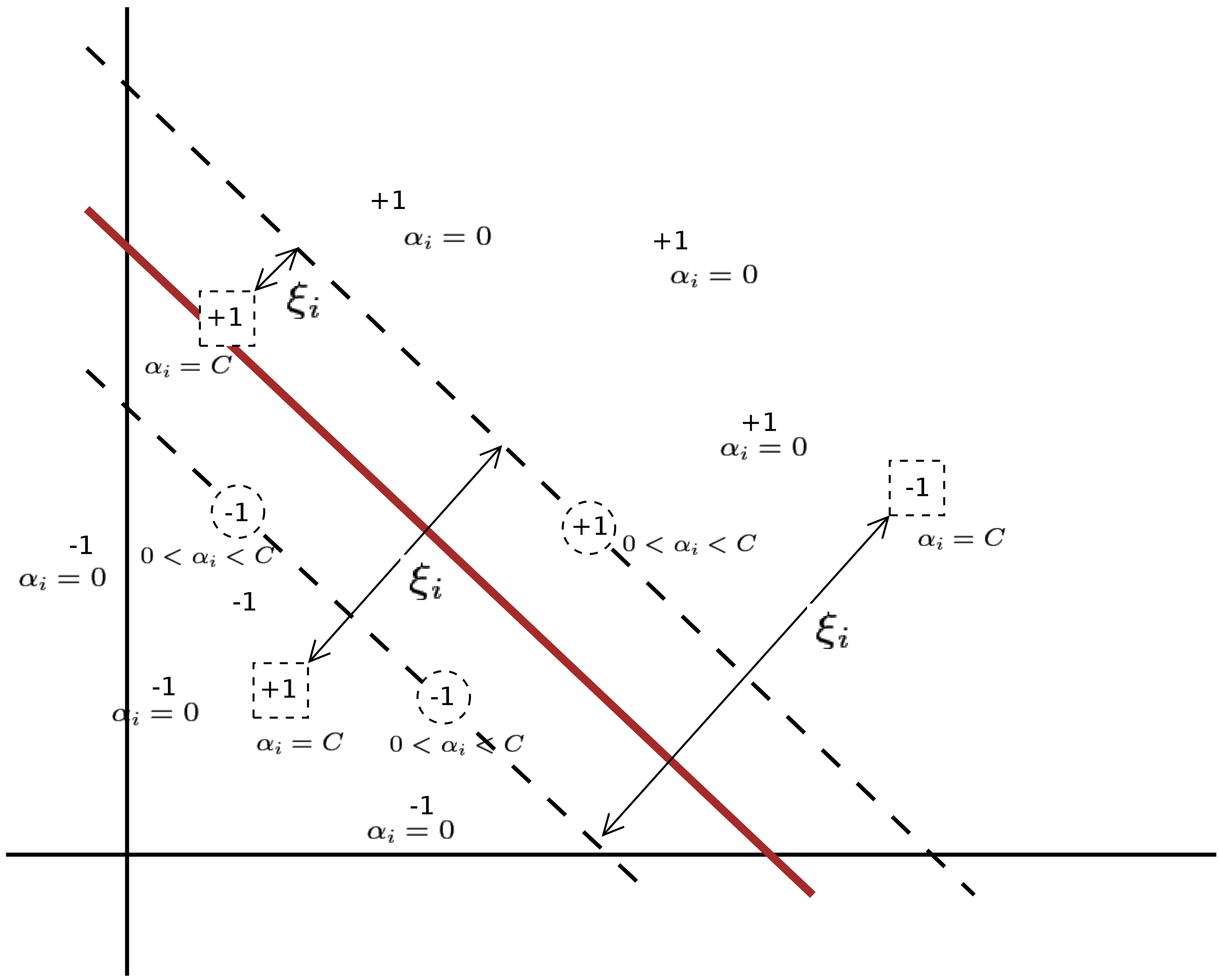
Nota: la única operación donde intervienen los vectores de entrenamiento es el producto escalar $\underline{\underline{(\vec{x}_i \cdot \vec{x}_j)}}$ presente en la expresión a maximizar.

Esto permitirá posteriormente *"kernelizar"* el algoritmo sustituyendo ese producto escalar por una función *kernel* adecuada

6.3 Margen máximo con holgura

Para mitigar aún más las posibilidades de sobreajuste $\Rightarrow$ permitir cierto grado de error en el hiperplano de separación de margen máximo

- Admite problemas no totalmente linealmente separables.
- ξ_i : pérdida/holgura admitida para el ejemplo $(\vec{x}_i, y_i) \in L$



Se relajan las restricciones de lo que es considerado un *"ejemplo bien clasificado"*

- El ejemplo de entrenamiento $(\vec{x}_i, y_i) \in L$ se considera como bien clasificado si se verifica:

$$\begin{array}{ll} \vec{w} \cdot \vec{x}_i \ + \ b \geq +1 - \xi_i & \text{para } y_i = +1 \\ \vec{w} \cdot \vec{x}_i \ + \ b \leq -1 + \xi_i & \text{para } y_i = -1 \end{array}$$

$$\text{con } \xi_i \geq 0 \ \forall i \in \{1, \ldots, l\}$$

- La cantidad máxima de "pérdidas" admitidas sobre el conjunto de entrenamiento se acota mediante el parámetro C

ENUNCIADO (forma primal)

Dado un conjunto de ejemplos de entrenamiento previamente clasificados $L = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), ..., (\vec{x}_l, y_l)\}$ y una cota máxima de pérdidas permitidas, C.
Encontrar b y $\vec{w}$ que:

$$\begin{array}{ll} \text{MINIMICEN} & \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{l} \xi_i \\ \text{SUJETO A :} & y_1(\vec{w} \cdot \vec{x}_1 \ + \ b) \geq 1 - \xi_1 \\ & y_2(\vec{w} \cdot \vec{x}_2 \ + \ b) \geq 1 - \xi_2 \\ & \ldots \\ & y_l(\vec{w} \cdot \vec{x}_l \ + \ b) \geq 1 - \xi_l \\ & \text{y} \\ & \xi_i \geq 0 \ \ \forall i \in \{1, 2, \ldots, l\} \end{array}$$

ENUNCIADO (forma dual)

Dado un conjunto de ejemplos de entrenamiento previamente clasificados $L = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), ..., (\vec{x}_l, y_l)\}$ y una cota máxima de pérdidas permitidas, C.

Encontrar los valores $\alpha_1, \alpha_2, \ldots, \alpha_l$ que:

$$\text{MAXIMICEN} \quad \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \underline{\underline{(\vec{x}_i \cdot \vec{x}_j)}}$$

$$\text{SUJETO A :} \quad \sum_{i=1}^{l} \alpha_i y_i = 0$$

y

$$0 \leq \alpha_i \leq C \ \forall i \in \{1, \ldots, l\}$$

- Los vectores soporte están asociados a valores de α_i que verifiquen $0 < \alpha_i < C$.
- Los vectores correspondientes a los errores de clasficación admitidos tienen asociado un $\alpha_i = C$.

6.4 El "truco" del kernel

Problema:

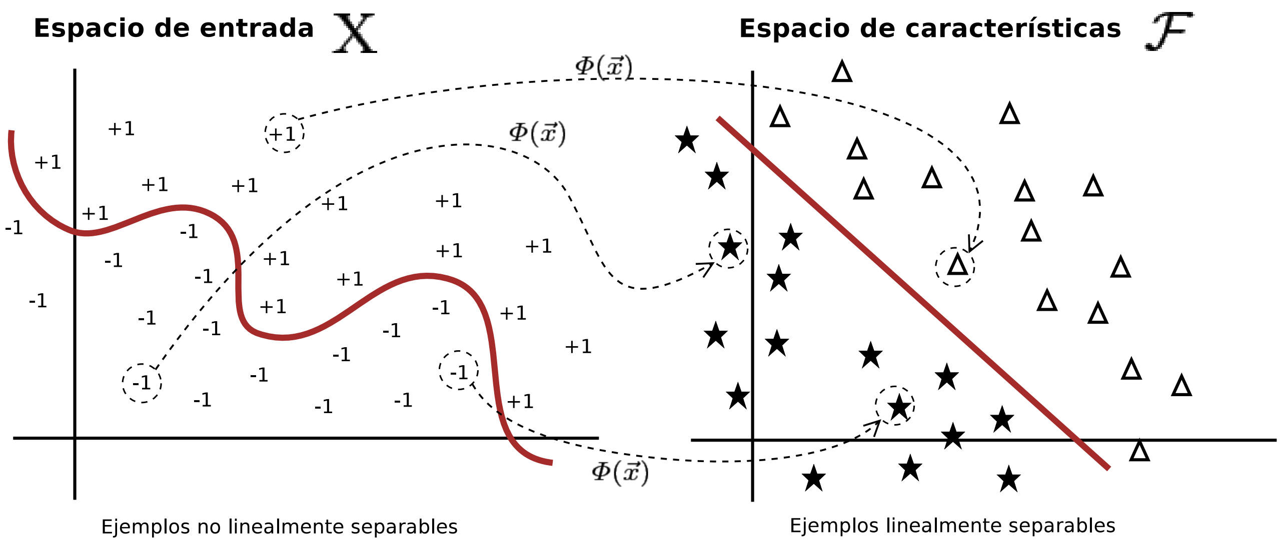
La mayoría de los "problemas reales" no son linealmente separables.

Idea: Transformar los ejemplos de entrenamiento a un espacio vectorial de alta dimensión ($N \gg n$) (denominado espacio de características), donde sí sea posible la separación lineal.

Función de transformación $\boxed{\Phi(x) : \mathrm{X} \rightarrow \mathcal{F}}$ con $\left\{ \begin{array}{l} |\mathrm{X}| = n, \quad |\mathcal{F}| = N \\ \text{y} \quad N \gg n \end{array} \right.$

- Recibe vectores del espacio de entrada, X, y los transforma en vectores del espacio de características, $\mathcal{F}$

$$\Phi(x) : \mathrm{X} \rightarrow \mathcal{F}$$



-1

Inconvenientes potenciales:

- Difícil encontrar/definir una función de transformación $\Phi(\vec{x})$ adecuada
- Costoso convertir vectores de X en vectores de $\mathcal{F}$ (vectores muy grandes)
- Costoso calcular productos escalares en $\mathcal{F}$ sobre vectores tan grandes.

Solución: Uso de funciones *kernel*

- Se aplican sobre vectores de X y su resultado es un producto escalar sobre "algún" espacio de características $\mathcal{F}$
- Definen una función de transformación $\Phi(\vec{x})$ implícita (no es necesario construirla ni calcularla)

Definición: (función *kernel*)

> Una función *kernel* $k(x, y) : \mathrm{X} \times \mathrm{X} \rightarrow \mathbb{R}$ asigna a cada par de objetos de entrada, x e y, un valor real que se corresponde con el producto escalar de sus respectivas imágenes en el espacio de caracterísitcas $\mathcal{F}$

Es decir, $\boxed{k(x, y) = \Phi(X) \cdot \Phi(u)}$ para alguna función de transformación implícita, $\Phi(x) : \mathrm{X} \rightarrow \mathcal{F}$.

Las funciones *kernel* **permiten:**

- Calcular productos escalares en $\mathcal{F}$ (espacio de características) aplicando la respectiva función *kernel* sobre X (espacio de entrada).
- No es necesarios que los objetos de entrada estén definidos en un *espacio vectorial*.
  - Las entradas no tienen por que ser necesariamente vectores numéricos.
  - Ejemplo: funciones *kernel* aplicables sobre cadenas de texto (*string kernels*)

Conclusión: ("truco del *kernel*" / *kernel trick*)

Con una función *kernel* adecuada cualquier algoritmo que pueda expresarse en función de productos escalares sobre su espacio de entrada puede ser *kernelizado*

- En el algoritmo, se sustituye el producto escalar "original" por la función *kernel*.
- Implicitamente, se consigue que el algoritmo "original" pase a aplicarse sobre el espacio de caracterísitcas $\mathcal{F}$

Nota: el "truco del *kernel*" permite que algoritmos "lineales" se apliquen sobre problemas "no lineales".

Teorema de Mercer (caracterización de funciones *kernel*)

Definiciones previas

- $k : \mathrm{X} \times \mathrm{X} \rightarrow \mathbb{R}$ es simétrica si $k(x, y) = k(y, x)\ \forall x, y \in \mathrm{X}$
- $k : \mathrm{X} \times \mathrm{X} \rightarrow \mathbb{R}$ es semidefinida positiva si se verifica que $\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j k(x_i, y_j) > 0$ para cualquier conjunto de objetos $x_1, x_2, \ldots, x_n$ de X y cualquier conjunto de valores reales $c_1, c_2, \ldots, c_n$.

Teorema

Para cualquier función $k : \mathrm{X} \times \mathrm{X} \rightarrow \mathbb{R}$ que sea simétrica y semidefinida positiva existe un espacio de Hilbert $\mathcal{F}$ y una función $\Phi : \mathrm{X} \rightarrow \mathcal{F}$ tal que:

$$k(x, y) = \Phi(x) \cdot \Phi(y) \quad \forall x, y \in \mathrm{X}$$

Nota: Un espacio de Hilbert es un espacio vectorial de dimensión N con propiedades equivalentes a las de $\mathbb{R}^N$

Funciones kernel típicas

- *kernel* identidad: $k(\vec{x}, \vec{y}) = \vec{x} \cdot \vec{y}$
- *kernel* polinómico: $k(\vec{x}, \vec{y}) = (\vec{x} \cdot \vec{y} + r)^p$
- *kernel* gaussiano (función de base radial [RBF]): $k(\vec{x}, \vec{y}) = e^{\left(\frac{-\|\vec{x}-\vec{y}\|^2}{2\sigma^2}\right)}$

Combinación de kernels

Si k_1 y k_2 son funciones *kernel*, también lo serán:

- $k_1(x, y) + k_2(x, y)$
- $ak_1(x, y)$
- $k_1(x, y)k_2(x, y)$

Ejemplo: *kernel* polinómico de grado 2 sobre vectores en $\mathbb{R}^2$ $(\mathrm{X} = \mathbb{R}^2)$

$$\vec{y} = (y_1, y_2) \in \mathbb{R}^2$$
$$\vec{z} = (z_1, z_2) \in \mathbb{R}^2$$
$$k(\vec{y}, \vec{x}) = (\vec{y} \cdot \vec{z})^2$$

Induce la función de transformación $\Phi(\vec{x}) : \mathbb{R}^2 \to \mathbb{R}^3$, definida como:

$$\Phi(\vec{x}) = \Phi((x_1, x_2)) = (x_1^2, x_2^2, \sqrt{2}x_1x_2) \in \mathbb{R}^3$$

Comprobación

$$\begin{aligned}
k(\vec{y}, \vec{z}) \quad & = k\,(\vec{y} \cdot \vec{z})^2 = \\
& = ((y_1, y_2) \cdot (z_1, z_2))^2 = \\
& = (y_1z_1 + y_2z_2)^2 = \\
& = (y_1z_1)^2 + (y_2z_2)^2 + 2y_1z_1y_2z_2 = \\
& = y_1^2z_1^2 + y_2^2z_2^2 + \sqrt{2}y_1y_2\sqrt{2}z_1z_2 = \\
& = (y_1^2, y_2^2, \sqrt{2}y_1y_2) \cdot (z_1^2, z_2^2, \sqrt{2}z_1z_2) = \\
& = \Phi(\vec{y}) \cdot \Phi(\vec{z})
\end{aligned}$$

$k(\vec{y}, \vec{z})$ calculado sobre $\mathbb{R}^2$ (espacio de entrada X) da como resultado el producto escalar de 2 vectores de $\mathbb{R}^3$ (espacio de características $\mathcal{F}$) $\begin{cases} \Phi(\vec{y}) = (y_1^2, y_2^2, \sqrt{2}y_1y_2) \\ \Phi(\vec{z}) = (z_1^2, z_2^2, \sqrt{2}z_1z_2) \end{cases}$

6.5 SVMs

El método de aprendizaje en el que se basan las *Support Vector Machines (SVM)* no es más que la kernelización de un clasificador lineal de margen máximo con holgura.

- Se aplica un algoritmo para aprender un clasificador lineal de margen máximo con holgura de forma implícita sobre el espacio de caracterísitcas $\mathcal{F}$ inducido por la función *kernel* empleada (en lugar de sobre el espacio de entrada original X).

ENUNCIADO (forma dual *kernalizada*)

Dado un conjunto de ejemplos de entrenamiento previamente clasificados $L = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots, (\vec{x}_l, y_l)\}$, una cota máxima de pérdidas permitidas, C, y una función *kernel* $k(x, y)$.

Encontrar los valores $\alpha_1, \alpha_2, \ldots, \alpha_l$ que:

$$\text{MAXIMICEN} \quad \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \underline{\underline{k(\vec{x}_i, \vec{x}_j)}}$$

$$\text{SUJETO A}: \sum_{i=1}^{l} \alpha_i y_i = 0$$

y

$$0 \leq \alpha_i \leq C \ \forall i \in \{1, \ldots, l\}$$

$k(\vec{x}_i, \vec{x}_j)$ equivale al producto escalar , $\Phi(\vec{x}_i) \cdot \Phi(\vec{x}_i)$, en $\mathcal{F}$.

Es decir, la optimización $\sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j k(\vec{x}_i, \vec{x}_j)$ se aplica realmente de forma implícita sobre $\sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \Phi(\vec{x}_i) \cdot \Phi(\vec{x}_i))$