Métodos Kernel y Máquinas de Vectores Soporte

Juan José del Coz Velasco
Oscar Luaces Rodríguez
Centro de Inteligencia Artificial
Universidad de Oviedo en Gijón
33271 – Gijón
juanjo@aic.uniovi.es, oluaces@aic.uniovi.es

19 de julio de 2007

1. Introducción

En los últimos años hemos asistido al desarrollo de una familia de algoritmos denominados de forma genérica *métodos kernel* que han suscitado gran interés debido, entre otras cosas, al éxito alcanzado por el que se considera el primer método kernel: las *máquinas de vectores soporte* o *SVM* (iniciales en inglés de *Support Vector Machines*).

El elemento común y pieza fundamental de todo método kernel es la *función kernel* o simplemente kernel que sirve como mecanismo de representación de la información de entrada al algoritmo. Como veremos, las funciones kernel tienen que cumplir ciertas condiciones para conferir a los métodos kernel las ventajas que los hacen especialmente interesantes. Una de esas ventajas es que estos métodos son aplicables a cualquier tipo de datos, sin importar a priori cuál es su representación original. Para ello sólo es necesario contar con alguna función kernel adecuada que, de forma implícita, transforme los datos de entrada y efectúe operaciones con esas transformaciones.

La otra gran ventaja de los métodos basados en funciones kernel es que podemos sacar partido de esa transformación implícita para aplicar algoritmos lineales y obtener con ellos soluciones no lineales. Para ello necesitamos en primer lugar formular el algoritmo lineal de forma que las operaciones con los datos de entrada se expresen en términos de operaciones con funciones kernel. A este proceso se le denomina *kernelización* de un algoritmo. De esta forma el algoritmo efectúa las operaciones a través de la función kernel sobre una versión transformada (de forma implícita) del problema original, donde esperamos poder encontrar una solución lineal.

En este capítulo vamos a presentar en primer lugar el concepto de función kernel y a caracterizar formalmente las funciones kernel; veremos también algunos ejemplos de funciones kernel y mencionaremos alguna de sus características más peculiares. A continuación presentaremos los métodos kernel en términos generales e ilustraremos cómo se pueden convertir algunos algoritmos clásicos sencillos en métodos kernel.

La segunda parte del capítulo está dedicada al método kernel por excelencia, las SVM. En esta parte veremos con cierto detalle los fundamentos teóricos de

las SVM y su aplicación tanto en problemas de clasificación como en problemas de regresión.

2. Kernels

Una manera bastante intuitiva de presentar el concepto de función kernel y su utilidad en el análisis de datos es hacerlo desde el punto de vista de la representación de información.

Supongamos que tenemos un conjunto de objetos $\mathcal{S} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ donde cada $\boldsymbol{x}_i \in \mathcal{X}$ representa un objeto diferente, por ejemplo, una proteína, una imagen, un paciente, etc. Si pretendemos aplicar un algoritmo de análisis al conjunto $\mathcal{S}$ necesitamos representarlo de alguna forma y, tal vez, en función de la representación elegida, adaptar el algoritmo de análisis. Una manera obvia de representar los datos consiste en definir para cada objeto $\boldsymbol{x}_i \in \mathcal{X}$ una representación $\phi(\boldsymbol{x}_i) \in \mathcal{F}$ que puede consistir, por ejemplo, en un vector de números reales (en cuyo caso $\mathcal{F} = \mathbb{R}^p$), en una secuencia de longitud p de 0's y 1's ($\mathcal{F} = \{0,1\}^p$), en una cadena de caracteres de tamaño fijo ($\mathcal{F}$ será el conjunto de todas las posibles cadenas de caracteres de dicho tamaño), etc. Podemos entonces representar $\mathcal{S}$ como el conjunto de representaciones de cada objeto, $\phi(\mathcal{S}) = \{\phi(\boldsymbol{x}_1), \ldots, \phi(\boldsymbol{x}_n)\}$; la información será procesada a continuación por algún algoritmo de análisis que debe estar preparado para manejar los datos en este formato.

Los métodos basados en funciones kernel proponen una forma de representar la información radicalmente diferente. En lugar de asociar a cada elemento del dominio de entrada una representación mediante $\phi : \mathcal{X} \to \mathcal{F}$, se utiliza una función $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ que, en cierto sentido, calcula la similitud de cada par de objetos del conjunto de entrada. El conjunto $\mathcal{S}$ viene representado en este caso por la matriz K de $n \times n$ valores reales resultantes de aplicar la función k a cada posible par de objetos del espacio de entrada, es decir, $K_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

El uso de funciones kernel permite hacer una representación de los datos que ofrece algunas ventajas interesantes de cara a su posterior tratamiento por los algoritmos de análisis. La más evidente es la modularidad que confiere esta planteamiento, ya que se aísla el algoritmo de análisis de la representación de los datos: se pueden diseñar algoritmos genéricos que van a manipular la información de matrices de números reales independientemente de cuál sea la función k que se utilice y del tipo de objetos del dominio del problema. Sin embargo la ventaja más importante, aunque no tan evidente por el momento, es que el uso de funciones kernel puede permitir que algoritmos de análisis relativamente sencillos encuentren relaciones complejas entre objetos de forma eficiente. Esto va a depender, fundamentalmente, del diseño de la función kernel.

2.1. La función kernel

Antes de caracterizar formalmente las funciones kernel vamos a presentar de forma intuitiva este concepto a través de un ejemplo sencillo. Supongamos que la información a analizar viene dada en forma de vectores de números reales, esto es, nuestro *espacio de entrada* es $\mathcal{X} = \mathbb{R}^d$. Puesto que, como hemos dicho, el espíritu de la función kernel es el de expresar en algún sentido la similitud entre dos objetos, una aproximación razonable para comparar pares de vectores

puede ser utilizar el *producto escalar*, que relaciona las normas de los vectores y el coseno del ángulo que forman: $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \|\boldsymbol{x}\| \|\boldsymbol{y}\| \cos(\angle(\boldsymbol{x}, \boldsymbol{y}))$. De esta manera podemos definir una función de comparación como la que sigue:

$$k_L(\boldsymbol{x}, \boldsymbol{y}) = \langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i=1}^{d} x_i y_i \tag{1}$$

que, como veremos más adelante, es una función kernel que se conoce como *kernel lineal*.

La utilidad de esta función parece, en principio, limitada, ya que requiere que los datos de entrada sean vectores de números reales. Sin embargo, podemos generalizar su uso si consideramos una representación alternativa de los datos, de forma que cada objeto del espacio de entrada, $\boldsymbol{x} \in \mathcal{X}$, tenga una representación $\phi(\boldsymbol{x}) \in \mathcal{F}$.

Definición 1 *Una función kernel es una función $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, que asigna a cada par de objetos del espacio de entrada, $\mathcal{X}$, un valor real correspondiente al producto escalar de las imágenes de dichos objetos en un espacio $\mathcal{F}$, que denominaremos* espacio de características*, es decir,*

$$k(\boldsymbol{x}, \boldsymbol{y}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{y}) \rangle, \tag{2}$$

donde $\phi : \mathcal{X} \to \mathcal{F}$.

De esta manera ni tan siquiera se requiere que el espacio de entrada $\mathcal{X}$ sea un espacio vectorial, puesto que el producto escalar se calcula en el espacio de características. Nótese también que, a diferencia del planteamiento clásico de representación de los datos, no necesitamos conocer cuál es la imagen $\phi(\boldsymbol{x})$ de cada objeto, sino únicamente el resultado del producto escalar de cada par de imágenes de los datos de entrada[1]. Lo que necesitamos es tener la garantía de que la función k que vayamos a utilizar se corresponde con un producto escalar en algún espacio $\mathcal{F}$. Es en este momento cuando debemos caracterizar formalmente las funciones kernel para lo que necesitamos dos definiciones previas

Definición 2 *Una función $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ es* simétrica *si $k(\boldsymbol{x}, \boldsymbol{y}) = k(\boldsymbol{y}, \boldsymbol{x})$ para cualquier par de objetos $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$.*

Definición 3 *Una función $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ es* semi-definida positiva *si*

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0$$

para cualquier conjunto de n objetos $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ de $\mathcal{X}$ y cualquier conjunto de valores reales $c_1, \ldots, c_n$, donde $n > 0$.

Tras estas definiciones podemos enunciar el siguiente teorema, debido a Aronszajn [1]

[1] De hecho, en algunos casos, el espacio de características es de dimensión infinita, mientras que el resultado de la función kernel se obtiene de forma relativamente simple.

Teorema 1 *Para cualquier función $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ que sea simétrica y semi-definida positiva existe un espacio de Hilbert $\mathcal{F}$ y una función $\phi : \mathcal{X} \to \mathcal{F}$ tal que*

$$k(\boldsymbol{x}, \boldsymbol{y}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{y}) \rangle, \text{ para cualquier } \boldsymbol{x}, \boldsymbol{y} \in \mathcal{X},$$

donde $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$ representa el producto escalar en el espacio de Hilbert entre dos puntos cualesquiera, $\boldsymbol{u}, \boldsymbol{v} \in \mathcal{F}$.

Es bastante simple comprobar como la función k_L presentada en la ecuación (1), es una función kernel, llamada *kernel lineal*, ya que satisface estas condiciones:

- es simétrica:
$$k_L(\boldsymbol{x}, \boldsymbol{y}) = \langle \boldsymbol{x}, \boldsymbol{y} \rangle = \langle \boldsymbol{y}, \boldsymbol{x} \rangle = k_L(\boldsymbol{y}, \boldsymbol{x})$$
- es semi-definida positiva:
$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j k_L(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle = \left\| \sum_{i=1}^{n} c_i \boldsymbol{x}_i \right\|^2 \geq 0 \qquad (3)$$

2.2. Ejemplos de funciones kernel

A continuación se muestran algunas de las funciones kernel más habituales, aplicables directamente cuando $\mathcal{X} \subseteq \mathbb{R}^n$:

Kernel polinómico Una función kernel polinómica de grado p puede ser

$$k_P(\boldsymbol{x}, \boldsymbol{y}) = (\langle \boldsymbol{x}, \boldsymbol{y} \rangle)^p \qquad (4)$$

aunque suele ser frecuente utilizar

$$k_{P'}(\boldsymbol{x}, \boldsymbol{y}) = (\langle \boldsymbol{x}, \boldsymbol{y} \rangle + r)^p \qquad (5)$$

Es relativamente sencillo comprobar que la función ϕ_P asociada al kernel de la ecuación (4) lleva a cada vector de entrada en un vector con todos los posibles monomios de grado p, mientras que la $\phi_{P'}$ asociada al kernel de la ecuación (5) los lleva a vectores compuestos por todos los monomios de grado menor o igual que p. No obstante, se puede comprobar fácilmente que las funciones kernel no necesariamente determinan una única función ϕ asociada. Consideremos, por ejemplo, un caso particular en el que $\mathcal{X} = \mathbb{R}^2$ y $p = 2$; si desarrollamos la expresión (4) obtenemos

$$\begin{aligned} k_P(\boldsymbol{x}, \boldsymbol{y}) &= (\langle (x_1, x_2), (y_1, y_2) \rangle)^2 = (x_1 y_1 + x_2 y_2)(x_1 y_1 + x_2 y_2) \\ &= x_1 y_1 x_1 y_1 + x_2 y_2 x_1 y_1 + x_1 y_1 x_2 y_2 + x_2 y_2 x_2 y_2 \\ &= \langle (x_1^2, x_1 x_2, x_2 x_1, x_2^2), (y_1^2, y_1 y_2, y_2 y_1, y_2^2) \rangle = \langle \phi_P(\boldsymbol{x}), \phi_P(\boldsymbol{y}) \rangle \end{aligned}$$

es decir, que $\phi_P(\boldsymbol{x}) = (x_1^2, x_1 x_2, x_2 x_1, x_2^2)$. Sin embargo, también podemos simplificar la expresión dado que uno de los monomios se repite

$$\begin{aligned} k_P(\boldsymbol{x}, \boldsymbol{y}) &= x_1 y_1 x_1 y_1 + 2 x_1 y_1 x_2 y_2 + x_2 y_2 x_2 y_2 \\ &= \langle (x_1^2, \sqrt{2} x_1 x_2, x_2^2), (y_1^2, \sqrt{2} y_1 y_2, y_2^2) \rangle = \langle \hat{\phi}_P(\boldsymbol{x}), \hat{\phi}_P(\boldsymbol{y}) \rangle, \end{aligned}$$

luego, $k_P(\boldsymbol{x}, \boldsymbol{y}) = \langle \phi_P(\boldsymbol{x}), \phi_P(\boldsymbol{y}) \rangle = \langle \hat{\phi}_P(\boldsymbol{x}), \hat{\phi}_P(\boldsymbol{y}) \rangle$.

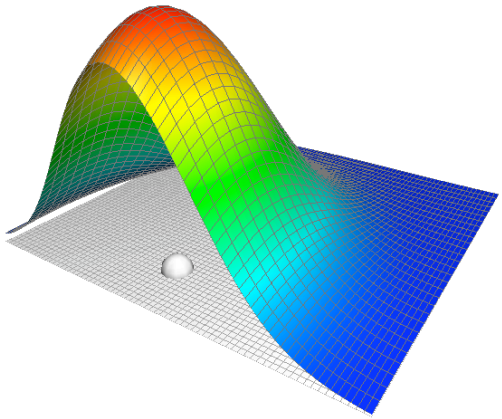


Figura 1: El kernel gaussiano en $\mathbb{R}^2$. La similitud entre el punto de $\mathbb{R}^2$ representado por una pequeña esfera y otros puntos de su entorno viene dada por el valor de la función gaussiana, que disminuye a medida que nos alejamos de él.

Kernel gaussiano Función de base radial gaussiana o RBF

$$k(\boldsymbol{x}, \boldsymbol{y}) = e^{-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{2\sigma^2}} \tag{6}$$

El hecho de que una función kernel representa en algún modo la similitud entre dos vectores se puede constatar de forma muy clara en el caso del kernel RBF. En este caso la similitud entre dos vectores del espacio de entrada viene dada por la valoración de uno de los vectores en la gaussiana centrada en el otro. La Figura 1 muestra un ejemplo para un espacio de entrada de dos dimensiones, en la que puede verse un punto en el plano XY y la función gaussiana centrada en él. La similitud entre dicho punto y otros puntos de su entorno es menor a medida que nos alejamos del punto en cualquier dirección.

El kernel gaussiano presenta una peculiaridad, y es que $k(\boldsymbol{x}, \boldsymbol{x}) = 1$ para todo $\boldsymbol{x} \in \mathcal{X}$, lo que implica que $\|\phi(\boldsymbol{x})\| = 1$. En estas circunstancias ocurre que el producto escalar de dos imágenes en $\mathcal{F}$ coincide con el coseno del ángulo que forman, ya que:

$$k(\boldsymbol{x}, \boldsymbol{y}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{y}) \rangle = \underbrace{\|\phi(\boldsymbol{x})\|}_{1} \cdot \underbrace{\|\phi(\boldsymbol{y})\|}_{1} \cdot \cos(\angle(\phi(\boldsymbol{x}), \phi(\boldsymbol{y})))$$

Como además $k(\boldsymbol{x}, \boldsymbol{y}) > 0$ para todo $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$ esto implica que el ángulo entre cualquier par de imágenes es menor que $\frac{\pi}{2}$, lo que indica que las imágenes de los vectores del espacio de entrada caen dentro de una región bastante restringida del espacio de características.

Sin embargo, y en otro sentido, estas imágenes ocupan un espacio que es lo más grande posible, ya que se ha demostrado que los vectores $\phi(\boldsymbol{x}_1), \ldots, \phi(\boldsymbol{x}_n)$ son linealmente independientes (siempre que los x_i sean distintos entre sí) y, por tanto, generan un espacio de dimensión n. Así, si el número de objetos de entrada es potencialmente infinito, la dimensión del espacio de características también lo es.

Kernels para cadenas Como ya hemos mencionado, la aplicación de métodos kernel no se restringe a espacios de entrada $\mathcal{X} \subseteq \mathbb{R}^d$, sino que los objetos pueden estar representados de diversas maneras. Una de las representaciones que merecen cierta atención es la basada en el uso de cadenas de caracteres. Ello es debido fundamentalmente al creciente interés en el desarrollo de aplicaciones bioinformáticas, en donde se utilizan cadenas de caracteres para representar proteínas (como secuencias de aminoácidos) o ADN genómico (como secuencias de nucleótidos).

Para poder aplicar métodos kernel a este tipo de datos es necesario definir kernels específicos que recojan (en algún sentido) la similitud entre dos secuencias de caracteres. La idea básica es comparar las cadenas en función de las subcadenas que contienen, de forma que cuantas más subcadenas tengan en común, más similares se consideran. Las subcadenas no tienen por qué estar formadas por caracteres contiguos por lo que cuanto más alejados estén el primer y el último carácter de la subcadena, menos debe ésta contribuir al grado de similitud.

Vamos a denotar por Σ al alfabeto que contiene un número finito de caracteres que pueden aparecer en las cadenas; el conjunto de todas las cadenas de una determinada longitud n será Σ^n y el conjunto de todas las cadenas de longitud finita será $\Sigma^* = \cup_{n=0}^{\infty}\Sigma^n$. La longitud de una cadena $s \in \Sigma^*$ la denotaremos por $|s|$, siendo sus elementos $s(1), \ldots, s(|s|)$ y su concatenación con otra cadena $t \in \Sigma^*$ la denotamos por st. Para denotar una subcadena u de s vamos a utilizar un vector $\boldsymbol{i} = (i_1, \ldots, i_{|u|})$, con $1 \leq i_1 < \ldots < i_{|u|} \leq |s|$, que indica la posición en la cadena s de cada uno de los caracteres que componen u, es decir que $u = s(\boldsymbol{i}) = s(i_1), \ldots s(i_{|u|})$. Vamos a denotar por $l(\boldsymbol{i}) = i_{|u|} - i_1 + 1$, la *longitud de la subsecuencia en* s; si u está formada por caracteres no contiguos en s, entonces $l(\boldsymbol{i})$ es mayor que $|u|$.

El espacio de características construido a partir de cadenas de longitud n será $\mathcal{F}_n = \mathbb{R}^{|\Sigma^n|}$, es decir, que el espacio de características tendrá una dimensión por cada elemento de Σ^n. Para cada posible subcadena $u \in \Sigma^n$ (de cadenas de longitud n) podemos describir la función ϕ_n como

$$[\phi_n(s)]_u = \sum_{\boldsymbol{i}:s(\boldsymbol{i})=u} \lambda^{l(\boldsymbol{i})} \tag{7}$$

donde $0 < \lambda \leq 1$ es un parámetro que se utiliza para conseguir que la contribución al valor final de $[\phi_n(s)]_u$ sea menor cuanto mayor sea la discontinuidad de u en s. Por ejemplo, considerando el espacio de características $\mathcal{F}_3$ en la dimensión correspondiente a cadena `epe`, y las cadenas `Pepe` y `separense`, tendríamos

$$\begin{aligned} [\phi_3(\texttt{Pepe})]_{\texttt{epe}} &= \lambda^3 \\ [\phi_3(\texttt{separense})]_{\texttt{epe}} &= \lambda^5 + \lambda^8 \end{aligned}$$

En la cadena `Pepe` sólo aparece una vez la subcadena `epe` (`Pepe`), mientras que en la cadena `separense` aparece dos veces dicha subcadena, y en ningún caso todos los caracteres son contiguos (`separense` y `separense`).

El kernel inducido por ϕ_n es de la forma

$$k_n(s,t) = \sum_{u \in \Sigma^n} [\phi_n(s)]_u [\phi_n(t)]_u = \sum_{u \in \Sigma^n} \sum_{\boldsymbol{i}:s(\boldsymbol{i})=u} \sum_{\boldsymbol{j}:t(\boldsymbol{j})=u} \lambda^{l(\boldsymbol{i})} \lambda^{l(\boldsymbol{j})} \tag{8}$$

2.3. Combinación de funciones kernel

A partir de la combinación de funciones kernel sencillas se pueden obtener kernels más complejos, que pueden ser de utilidad en determinadas aplicaciones. Obviamente, es necesario que las operaciones aplicadas preserven las propiedades de las funciones kernel, es decir, que la función resultante sea simétrica y semi-definida positiva.

Así, si k_1 y k_2 son kernels en $\mathcal{X} \times \mathcal{X}$, con $\mathcal{X} \subseteq \mathcal{R}^n$, las funciones definidas a continuación también son kernels:

- $k(\boldsymbol{x}, \boldsymbol{y}) = k_1(\boldsymbol{x}, \boldsymbol{y}) + k_2(\boldsymbol{x}, \boldsymbol{y})$
- $k(\boldsymbol{x}, \boldsymbol{y}) = ak_1(\boldsymbol{x}, \boldsymbol{y})$
- $k(\boldsymbol{x}, \boldsymbol{y}) = k_1(\boldsymbol{x}, \boldsymbol{y})k_2(\boldsymbol{x}, \boldsymbol{y})$
- $k(\boldsymbol{x}, \boldsymbol{y}) = f(\boldsymbol{x})f(\boldsymbol{y})$
- $k(\boldsymbol{x}, \boldsymbol{y}) = k_3(\phi(\boldsymbol{x}), \phi(\boldsymbol{y}))$
- $k(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^\mathsf{T}\boldsymbol{B}\boldsymbol{y}$

donde $a \in \mathcal{R}^+$, $f : \mathcal{X} \to \mathbb{R}$, $\phi : \mathcal{X} \to \mathbb{R}^N$ con k_3 un kernel en $\mathbb{R}^N \times \mathbb{R}^N$ y $\boldsymbol{B}$ es una matriz de $n \times n$ simétrica semi-definida positiva. La demostración puede encontrarse en [27].

3. Métodos Kernel

En esta sección vamos a centrarnos en los algoritmos que procesan la información representada mediante el uso de funciones kernel. Estos algoritmos, denominados de forma genérica *métodos kernel*, únicamente reciben la información a procesar en forma de matrices calculadas con una función kernel.

En los últimos años se han desarrollado métodos kernel para llevar a cabo diversas tareas como clasificación, regresión, agrupamiento, etc. Entre éstos métodos destacan en particular las *máquinas de vectores soporte* [3], conocidas también por las siglas SVM. A continuación vamos a presentar algunos elementos comunes a estos métodos, junto con algún método kernel sencillo, antes de entrar de lleno en los detalles de las SVM.

3.1. La *kernelización* de algoritmos

La definición de kernel como producto escalar en el espacio de características ofrece la posibilidad de transformar aquellos algoritmos que puedan expresarse en términos de productos escalares en el espacio de entrada, de una forma genérica en la que se efectúen de forma implícita productos escalares en el espacio de características: basta con reemplazar el producto escalar de la formulación original por la evaluación de una función kernel. En particular, cuando el kernel utilizado es lineal (ecuación (1)), el algoritmo *kernelizado* será idéntico al original.

Con este sencillo *"truco"* y el uso de kernels diseñados específicamente para datos no vectoriales se puede extender la aplicación de algoritmos clásicos en

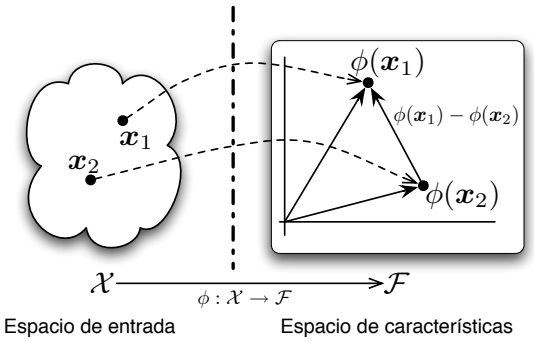


Figura 2: Definimos la distancia entre puntos del espacio de entrada $\mathcal{X}$ como la distancia en el espacio de características $\mathcal{F}$, es decir, como la norma del vector diferencia entre las imágenes de $\boldsymbol{x}_1$ y $\boldsymbol{x}_2$. Esa norma se puede calcular sin necesidad de conocer las imágenes $\phi(\boldsymbol{x}_1)$ y $\phi(\boldsymbol{x}_2)$ usando un truco con un kernel.

espacios vectoriales dotados de producto escalar a cualquier otro tipo de datos, siempre que se pueda definir un kernel. Podemos encontrar, por ejemplo, aplicaciones en el campo de la biología donde se han efectuado análisis de componentes principales sobre conjuntos de secuencias de aminoácidos gracias al uso de kernels adecuados [26].

Para ilustrar esta ventaja que proporcionan los métodos kernel vamos a generalizar un par de métodos clásicos de aprendizaje. El primero es una versión genérica de un método de clasificación basado en el principio del vecino más próximo, mientras que el segundo es el perceptrón, un clásico en el campo de las redes de neuronas artificiales.

3.1.1. Clasificación por mínima distancia

Supongamos que tenemos objetos agrupados en conjuntos, cada uno de ellos con diferentes propiedades, y tratamos de predecir a cuál de ellos pertenece un determinado objeto $\boldsymbol{z}$. Una solución razonable en un espacio vectorial sería considerar la distancia Euclídea del objeto $\boldsymbol{z}$ a cada uno de los conjuntos y suponer que pertenece al conjunto más cercano. Esa distancia puede medirse como la distancia desde el punto $\boldsymbol{z}$ al centroide de cada conjunto. Ahora bien, si el espacio de entrada no es vectorial podemos llevar los objetos a un espacio de características $\mathcal{F}$ y calcular en él la distancia entre las imágenes de los objetos originales.

Veamos cómo podemos calcular la distancia entre dos puntos cualesquiera en el espacio de características. Para empezar, consideremos dos objetos $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$, que tendrán una representación $\phi(\boldsymbol{x}_1)$ y $\phi(\boldsymbol{x}_2)$ en $\mathcal{F}$, respectivamente. Como hemos indicado previamente, vamos a definir la distancia entre los dos objetos del espacio de entrada como la distancia entre sus respectivas imágenes en $\mathcal{F}$ o, dicho de otro modo, como la norma del vector diferencia $\phi(\boldsymbol{x}_1) - \phi(\boldsymbol{x}_2)$ (véase Figura 2)

$$d(\boldsymbol{x}_1, \boldsymbol{x}_2) = \|\phi(\boldsymbol{x}_1) - \phi(\boldsymbol{x}_2)\|. \tag{9}$$

Para poder utilizar el truco del kernel es necesario reformular esta distancia

en función de productos escalares, que luego podrán ser sustituidos por alguna función kernel. Dado que la norma de un vector es $\|\boldsymbol{v}\| = \sqrt{\langle \boldsymbol{v}, \boldsymbol{v} \rangle}$ tenemos que

$$\|\phi(\boldsymbol{x}_1) - \phi(\boldsymbol{x}_2)\| = \sqrt{\langle \phi(\boldsymbol{x}_1) - \phi(\boldsymbol{x}_2), \phi(\boldsymbol{x}_1) - \phi(\boldsymbol{x}_2) \rangle}. \tag{10}$$

Pero dado que

$$\begin{aligned} &\langle \phi(\boldsymbol{x}_1) - \phi(\boldsymbol{x}_2), \phi(\boldsymbol{x}_1) - \phi(\boldsymbol{x}_2) \rangle = \\ &\quad = \langle \phi(\boldsymbol{x}_1), \phi(\boldsymbol{x}_1) \rangle + \langle \phi(\boldsymbol{x}_2), \phi(\boldsymbol{x}_2) \rangle - 2\langle \phi(\boldsymbol{x}_1), \phi(\boldsymbol{x}_2) \rangle, \end{aligned} \tag{11}$$

podemos calcular la distancia $d(\boldsymbol{x}_1, \boldsymbol{x}_2)$ en términos de operaciones con el kernel:

$$d(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sqrt{k(\boldsymbol{x}_1, \boldsymbol{x}_1) + k(\boldsymbol{x}_2, \boldsymbol{x}_2) - 2k(\boldsymbol{x}_1, \boldsymbol{x}_2)}. \tag{12}$$

Ya estamos en condiciones de plantear el cálculo en $\mathcal{F}$ de la distancia entre la imagen de un objeto y el centroide de un conjunto de imágenes. Para un conjunto $\mathcal{S} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ definimos el centroide del conjunto de sus imágenes como

$$m = \frac{1}{n} \sum_{i=1}^{n} \phi(\boldsymbol{x}_i). \tag{13}$$

Por tanto, la distancia de un objeto $\boldsymbol{z}$ al conjunto $\mathcal{S}$ se calculará como

$$\operatorname{dist}(\boldsymbol{z}, \mathcal{S}) = d(\boldsymbol{z}, m) = \|\phi(\boldsymbol{z}) - m\| = \left\| \phi(\boldsymbol{z}) - \frac{1}{n} \sum_{i=1}^{n} \phi(\boldsymbol{x}_i) \right\|. \tag{14}$$

Al igual que hicimos en (11), podemos formular la distancia en términos de productos escalares, obteniendo en este caso lo siguiente

$$\operatorname{dist}(\boldsymbol{z}, \mathcal{S}) = \sqrt{k(\boldsymbol{z}, \boldsymbol{z}) - \frac{2}{n} \sum_{i=1}^{n} k(\boldsymbol{z}, \boldsymbol{x}_i) + \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} k(\boldsymbol{x}_i, \boldsymbol{x}_j)}. \tag{15}$$

Nótese que el centroide m es un punto perfectamente definido del espacio de características $\mathcal{F}$ que *no necesariamente tiene preimagen* $\phi^{-1}(m)$ *en el espacio de entrada* $\mathcal{X}$ y, aún siendo así, podemos calcular la distancia de un objeto a un conjunto de objetos exclusivamente a partir de los valores de la función kernel.

3.2. Perceptrón: versión con kernels

Para entender mejor cómo se puede *kernelizar* un algoritmo e introducir, además, algunas de las características de las SVM, vamos a seguir un ejemplo tomado de [8] en el que se usan kernels con un algoritmo clásico: el perceptrón [23]. Es bien sabido que el perceptrón forma parte de la familia de los clasificadores lineales, algoritmos que producen una función lineal que puede utilizarse, por ejemplo, para diferenciar entre ejemplos de dos clases. Aunque muy sencillos, este tipo de algoritmos se adaptan a muchos problemas reales, como determinar si un paciente padece una enfermedad o decidir si se concede un préstamo bancario a un cliente. Veremos como el uso de kernels puede convertir este algoritmo que genera modelos simples, en un método capaz de producir funciones de decisión muy complejas.

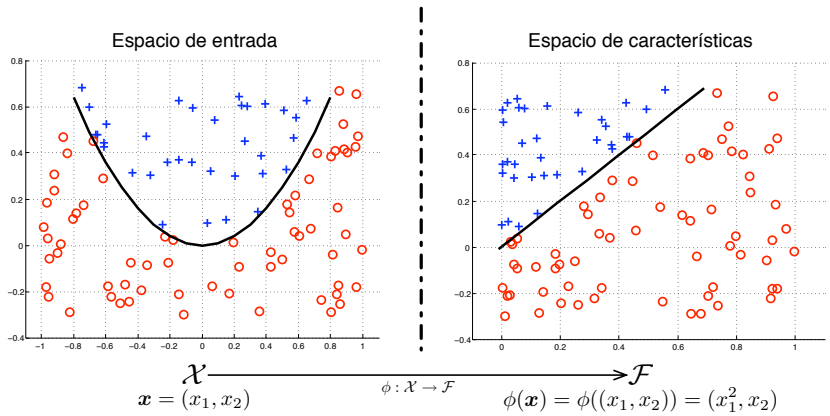


Figura 3: En la figura de la izquierda podemos ver que la función capaz de separar los objetos "o" de los objetos "+" no es lineal. Sin embargo, si representamos los mismos objetos en un determinado espacio de características, observamos que una función lineal en ese espacio sí que es capaz de separarlos. Trabajar en el espacio de características con algoritmos *kernelizados* nos permite utilizar algoritmos lineales para abordar problemas no lineales sin necesidad de conocer explícitamente ϕ (en este caso es conocido y se muestra por motivos didácticos).

La idea subyacente es que la función no lineal que representa la solución que buscamos en el espacio de entrada es una función lineal en algún espacio de características. Es en dicho espacio donde los algoritmos lineales deben operar para encontrar la solución y, si las operaciones en $\mathcal{F}$ se pueden expresar en términos de productos escalares, entonces podemos efectuar esas operaciones a través de la función kernel, sin necesidad de conocer la representación de los objetos en $\mathcal{F}$.

A modo de ejemplo, supongamos que tenemos un problema de clasificación binaria en el que los datos de entrada son vectores en $\mathbb{R}^2$, representados gráficamente como nubes de puntos en la parte izquierda de la Figura 3. Como se puede ver, en el espacio de entrada ($\mathcal{X} = \mathbb{R}^2$) necesitamos una función no lineal para discriminar correctamente las clases a las que pertenecen los objetos, así que el perceptrón no sería capaz de encontrar esa solución. Sin embargo, si representamos las imágenes de cada punto del conjunto de entrada mediante $\phi(\boldsymbol{x}) = (x_1^2, x_2)$ el problema pasa a ser linealmente separable, como se aprecia en la parte derecha de la Figura 3, y en ese espacio de características el perceptrón sí podría encontrar la solución. En lo que sigue vamos a ver cómo se puede kernelizar el perceptrón para que sea capaz de resolver este tipo de problemas.

La entrada para el perceptrón es un conjunto $\mathcal{S} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, donde cada $\boldsymbol{x}_i \in \mathbb{R}^d$, siendo la clase $y_i \in \{+1, -1\}$. El objetivo es diferenciar entre ambas clases mediante una función lineal

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = \sum_{i=1}^{n} w_i x_i + b,$$

que asigne valores mayores o iguales que cero a los objetos de la clase $+1$ y menores que cero para la clase -1. De esta forma bastaría con aplicar la función

Algoritmo 3.1 Perceptrón: versión tradicional

Función PERCEPTRON ($\mathcal{S}$) : un hiperplano
$\boldsymbol{w}_0 \leftarrow \mathbf{0}$; $t \leftarrow 0$
repetir
  para i **desde** 1 **hasta** n **hacer**
    si $y_i\langle \boldsymbol{w}_t, \boldsymbol{x}_i\rangle \leq 0$ **entonces**
      $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t + y_i\boldsymbol{x}_i$
      $t \leftarrow t + 1$
    fin si
  fin para
hasta que no haya errores en una iteración
retorna $\boldsymbol{w}$

sgn al valor devuelto por f para construir un clasificador:

$$h(\boldsymbol{x}) = \text{sgn}(f(\boldsymbol{x})).$$

Para simplificar la explicación de esta sección, prescindiremos del término independiente b y supondremos que no es necesario para separar las clases del dominio. Con esta condición, una versión muy simplificada del perceptrón puede ser la que se muestra en el Algoritmo 3.1[2]. El algoritmo converge a una solución siempre que las clases sean separables linealmente.

Este algoritmo no se puede kernelizar directamente ya que no aparecen productos escalares entre los ejemplos. Sin embargo, si lo analizamos, podemos observar como el valor de $\boldsymbol{w}$ se calcula iterativamente recorriendo el conjunto de entrada y los ejemplos que influyen en su valor son únicamente los mal clasificados, de forma que los de la clase $+1$ se van sumando a $\boldsymbol{w}$ y los de la clase -1 se van restando. Es el efecto que se produce al multiplicar por la clase y_i. El número de veces que cada ejemplo se suma o se resta dependerá de las iteraciones que tarde el hiperplano en clasificarlo bien. Asumiendo que $\boldsymbol{w}$ se inicia a $\mathbf{0}$, es claro que su valor final será una combinación lineal de los ejemplos de entrenamiento:

$$\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i.$$

Dada esta nueva representación de $\boldsymbol{w}$, podemos reescribir el algoritmo anterior para ponerlo en términos de los respectivos α_i de cada ejemplo, que nos indican las veces que ese ejemplo ha intervenido en la actualización de $\boldsymbol{w}$. El algoritmo queda como se muestra en 3.2. Aquellos ejemplos más difíciles de clasificar tendrán valores más altos de α, mientras que en los ejemplos clasificados bien en todas las iteraciones tendrá valor cero. Luego, cada α_i indica la dificultad de clasificar el ejemplo i.

El vector $\boldsymbol{\alpha}$, formado por los respectivos valores de α_i de cada ejemplo, es una representación dual de $\boldsymbol{w}$ en un espacio de coordenadas diferente. Con esta nueva representación podemos reescribir el modelo generado como

$$h(\boldsymbol{x}) = \text{sgn}\left(\langle \boldsymbol{w}, \boldsymbol{x}\rangle\right) = \text{sgn}\left(\sum_{i}^{n} \alpha_i y_i \langle \boldsymbol{x}_i, \boldsymbol{x}\rangle\right).$$

[2]Una manera sencilla de calcular el valor de b adecuado sería añadir una dimensión en todos los ejemplos de entrada y que todos ellos tomen valor 1 en esta nueva dimensión. El $\boldsymbol{w}$ devuelto tendrá una dimensión más que será el valor de b.

Algoritmo 3.2 Perceptrón: versión dual

Función PERCEPTRON ($\mathcal{S}$) : la representación dual del hiperplano
$\boldsymbol{\alpha} \leftarrow \mathbf{0}$;
repetir
  para i **desde** 1 **hasta** n **hacer**
    si $y_i \left(\sum_{j=1}^{n} \alpha_j y_j \langle \boldsymbol{x}_j, \boldsymbol{x}_i \rangle \right) \leq 0$ **entonces**
      $\alpha_i \leftarrow \alpha_i + 1$
    fin si
  fin para
hasta que no haya errores en una iteración
retorna $\boldsymbol{\alpha}$

Hemos representado tanto el algoritmo, como la hipótesis final, en una forma dual a la original en la que los ejemplos aparecen siempre a través de productos escalares entre ellos. Utilizando el truco kernel, podemos reemplazar esos productos escalares por la aplicación de una función kernel. De esa forma la condición de la sentencia `si-entonces` sería reemplazada por:

$$y_i \left(\sum_{j}^{n} \alpha_j y_j k(\boldsymbol{x}_j, \boldsymbol{x}_i) \right) \leq 0,$$

y la hipótesis también se modificaría utilizando la misma función kernel empleada en el algoritmo

$$h(\boldsymbol{x}) = \operatorname{sgn} \left(\sum_{i}^{n} \alpha_i y_i k(\boldsymbol{x}_i, \boldsymbol{x})) \right).$$

El uso de determinadas funciones kernel convertirá esta versión *kernelizada* del perceptrón en un sistema capaz de generar funciones de decisión no lineales en el espacio de entrada. Es un ejemplo claro de cómo el uso de kernels puede unir la sencillez de ciertos algoritmos con la complejidad de algunos espacios de características. Un caso análogo, aunque con un algoritmo de aprendizaje más complejo y potente, se presentará cuando introduzcamos las máquinas de vectores soporte.

De la misma forma que hemos kernelizado el perceptrón se pueden adaptar otros mecanismos lineales, como el análisis discriminante lineal (LDA) [12] o el análisis de componentes principales (PCA) [16], para convertirlos en métodos no lineales.

4. Máquinas de Vectores Soporte

4.1. Introducción

Sin duda el método kernel más popular son las máquinas de vectores soporte, conocidas como SVM (*Support Vector Machines*). Fueron introducidas por Vapnik a partir de sus trabajos sobre las teorías del aprendizaje estadístico, en los que acotaba el error de generalización en función de la complejidad del espacio de hipótesis. Los primeros trabajos datan de principios de los noventa [3, 7]. Desde entonces las máquinas de vectores soporte han ganado un merecido reconocimiento gracias a los sólidos principios teóricos en los que se fundamenta su

diseño y al estupendo rendimiento que ofrecen en una gran variedad de aplicaciones prácticas. Parafraseando al propio Vapnik, "no existe nada más práctico que una buena teoría". Buena parte de su popularidad radica, precisamente, en el hecho de que las máquinas de vectores soporte son capaces de producir buenos modelos para múltiples tipos de aplicaciones prácticas. Y aunque en su origen se diseñaron para resolver solamente problemas de clasificación binaria, en la actualidad su aplicación se ha extendido a tareas de regresión, multiclasificación, agrupamiento, y muy recientemente se empiezan a sentar las bases teóricas para resolver problemas en los que la salida puede ser aún más compleja y estructurada, como un árbol o un grafo.

¿Qué hace diferentes las SVM de otros métodos de aprendizaje? ¿Donde reside su éxito? La mayor diferencia entre las máquinas de vectores soporte y otros métodos tradicionales de aprendizaje es que las SVM no se centran en construir hipótesis que cometan pocos errores, sino que lo que pretenden es que producir predicciones en las que se pueda tener mucha confianza, aún a costa de cometer ciertos errores. Tradicionalmente, la mayoría de los algoritmos de aprendizaje se han centrado en lo primero, reducir al mínimo los errores cometidos por el modelo generado. Se basan en lo que se denomina el principio de Minimización del Riesgo Empírico (ERM, Empirical Risk Minimization). El enfoque de las máquinas de vectores soporte es diferente, no buscarán reducir el riego empírico cometiendo pocos errores, sino que pretenderá construir modelos confiables. Ese principio recibe el nombre de Minimización del Riesgo Estructural (SRM, Structural Risk Minimization). Las SVM buscan un modelo que estructuralmente tenga poco riesgo de cometer errores ante datos futuros.

Sin embargo, en su versión original para clasificación binaria, su planteamiento no es más complejo que el del perceptrón mostrado anteriormente. En su forma de resolverlo, obviamente sí. Ambos son clasificadores lineales, es decir, producen el mismo tipo de modelos: una función lineal que podremos enriquecer (convertirla en no lineal) introduciendo funciones kernels. Esa es la primera idea de su diseño: se parte de un tipo de funciones sencillas y se busca una solución óptima. Después se amplia el tipo de funciones que pueden aprenderse usando kernels, sin aumentar apenas la complejidad del proceso.

La diferencia entre el perceptrón y las SVM está en la forma en la que cada sistema busca esa función lineal. Mientras el perceptrón trata en cada iteración de reducir el número de errores, las SVM buscan el hiperplano que nos ofrezca menos riesgo desde un punto de vista estructural. ¿Cómo lo consigue? Ahí es donde aparece uno de los elementos importantes de las máquinas de vectores soporte, como es el concepto de *margen*. El objetivo de una máquina de vectores soporte de clasificación es maximizar el margen de la solución, que en primera instancia podemos entender como producir el clasificador que separe en mayor medida las dos clases. El proceso de maximizar el margen nos llevará a un problema de optimización, que tendrá la ventaja de que nos calculará el hiperplano de mayor margen en un tiempo polinomial y sin la posibilidad de que el algoritmo se quede atrapado en un máximo local.

Para hacer la explicación más sencilla, comenzaremos con los problemas de clasificación binaria más simples, aquellos en los que las dos clases son separables linealmente, y posteriormente extenderemos la solución al caso no separable.

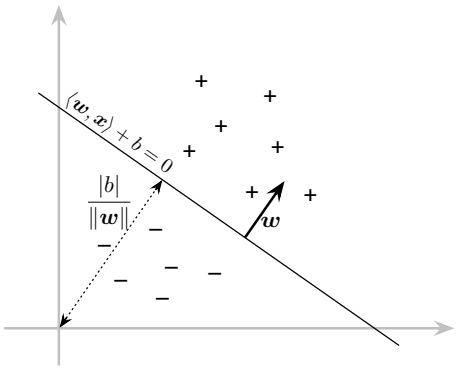


Figura 4: Hiperplano de separación. Dos clases linealmente separables se pueden clasificar mediante un hiperplano $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = 0$ que divida el espacio de entrada en dos regiones. El hiperplano es perpendicular al vector $\boldsymbol{w}$ y su distancia al origen depende de b y de $\|\boldsymbol{w}\|$. Nótese que todas las funciones $\lambda(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b)$ definen exactamente el mismo hiperplano en el espacio de entrada.

4.2. Hiperplano de margen máximo

Dado que en su planteamiento original las máquinas de vectores soporte tienen el mismo objetivo que el perceptrón estudiado en la sección 3.2, la especificación del problema es idéntica. Disponemos de una muestra de entrenamiento $\mathcal{S}$ formada por n ejemplos, es decir, $\mathcal{S} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, donde cada $\boldsymbol{x}_i$ pertenece al espacio de entrada $\mathcal{X}$ y la clase $y_i \in \{+1, -1\}$[3]. Si consideramos de momento que $\mathcal{X} \subseteq \mathbb{R}^d$, la clasificación binaria en ese dominio se puede realizar mediante una función lineal $f : \mathcal{X} \to \mathbb{R}$ que asigne valores positivos a los ejemplos de la clase $+1$ y negativos a los de la clase -1:

$$h(\boldsymbol{x}) = \operatorname{sgn}(f(\boldsymbol{x})) = \operatorname{sgn}(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b).$$

Como en el caso del perceptrón, sólo necesitaremos encontrar los valores adecuados para el vector $\boldsymbol{w}$ y el término independiente b que consigan diferenciar los objetos de ambas clases. Una interpretación geométrica de esta solución aparece en la Figura 4. El hiperplano definido por la ecuación $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = 0$ divide el espacio de entrada $\mathcal{X}$ en dos partes, correspondientes a cada una de las dos clases. Como se aprecia, el vector $\boldsymbol{w}$ define la dirección perpendicular al hiperplano y b el desplazamiento paralelo del mismo con respecto al origen.

Si ambas clases son linealmente separables, es decir, si es posible encontrar un hiperplano que separe los ejemplos de cada una de las clases sin errores, el problema planteado tiene múltiples soluciones posibles, ya que en general habrá muchos hiperplanos capaces de separar correctamente los ejemplos. En ese caso, ¿cuál escoger?, ¿cuál será capaz de generalizar mejor de acuerdo al conjunto de entrenamiento utilizado? Recordemos que el perceptrón selecciona el primer hiperplano que clasifica bien todos los ejemplos, sin tener en cuenta otros factores.

[3]El hecho de utilizar como clases $+1$ y -1 es algo habitual en SVM ya que facilita la notación de muchas expresiones, permitiendo, por ejemplo, calcular fácilmente su valor absoluto.

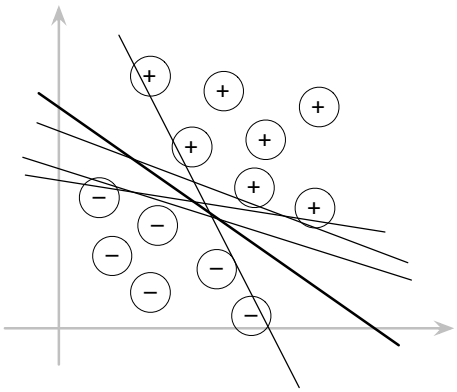


Figura 5: Dado un conjunto linealmente separable, en general, existen muchos hiperplanos capaces de separar las clases. Sin embargo, uno de ellos, el de trazo más grueso en la imagen, está más distanciado de ambas clases. Ese hiperplano presenta menos riesgo frente a un posible ruido (representado por las circunferencias que rodean a cada ejemplo) en los datos de entrenamiento. Obsérvese como algunos hiperplanos separadores están muy próximos a una de las dos clases y fallarían si los ejemplos fueran ruidosos.

Intuitivamente, parece claro que no todos los hiperplanos que consiguen separar los ejemplos sin cometer errores son igual de buenos. Obsérvese la Figura 5. Imaginemos, por ejemplo, que el conjunto de entrenamiento utilizado representa un proceso industrial que deseamos caracterizar y que los atributos que describen los ejemplos proceden de sensores electrónicos que miden las variables de interés en dicho proceso. Los sensores empleados no son totalmente precisos y producen medidas que pueden estar desviadas de las reales en un pequeño porcentaje. Eso implicaría que nuestros ejemplos de entrenamiento realmente podrían estar situados en cualquiera de los puntos de su entorno más próximo. Ese hecho lo hemos representado en la figura con una circunferencia alrededor de cada ejemplo, cuyo radio dependerá de la precisión de los sensores. Como se puede apreciar, muchos de los hiperplanos que separan sin errores las clases atraviesan algunas de esas circunferencias. Es decir, fallarían si los ejemplos estuvieran situados en esas zonas, lo cual es posible dada la inexactitud de los sensores. El ruido es un efecto muy común y en muchos casos no puede eliminarse utilizando unos sensores mejores.

Parece evidente, por tanto, que el hiperplano que esté más alejado de los ejemplos de ambas clases, es decir, que defina una frontera *más ancha* es más resistente al ruido que puedan tener los ejemplos de entrenamiento y es menos probable que cometa errores ante datos futuros. Esa separación entre el hiperplano y cada una de las dos clases es lo que se denomina *margen*. En la Figura 5, puede apreciarse cómo el hiperplano definido por la recta más gruesa está a la misma distancia de las dos clases y presenta un margen mayor que el resto, lo que estructuralmente ofrece mayores garantías. Nuestro objetivo es, por tanto, buscar entre todos los hiperplanos válidos aquel que presente un margen mayor. Veremos que esta búsqueda puede resolverse de forma óptima ya que el problema no presenta extremos locales.

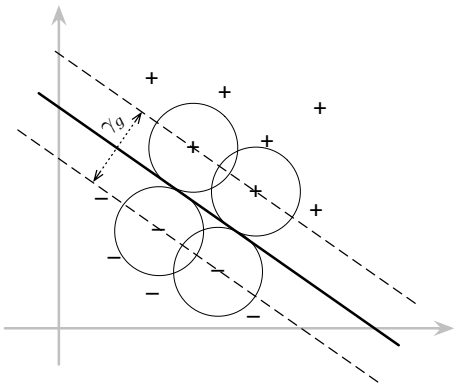


Figura 6: El margen geométrico. El hiperplano de margen geométrico máximo es el que está más distanciado de ambas clases. Esa separación γ_g permite que aún los ejemplos más cercanos al hiperplano separador, los situados justo en el margen (líneas punteadas) de cada clase, podrían ser clasificados bien aunque tuvieran mucho ruido (circunferencias).

Si consideramos la función f, el concepto de margen puede entenderse de dos formas diferentes y ambas válidas. Podríamos decir que, en realidad, hay dos tipos de margen relacionados entre sí:

- **margen funcional**: será la menor diferencia entre aplicar la función f a los ejemplos de la clase positiva y negativa,

$$\gamma_f = \min_{+}(f(\boldsymbol{x}_+)) - \max_{-}(f(\boldsymbol{x}_-)),$$

- **margen geométrico**: será la distancia entre los ejemplos de ambas clases. Esta distancia podemos calcular como la suma de la distancia del hiperplano definido por $\boldsymbol{w}$ y b al ejemplo más próximo de cada clase,

$$\gamma_g = \min_{+}(d(\boldsymbol{w}, b; \boldsymbol{x}_+)) + \min_{-}(d(\boldsymbol{w}, b; \boldsymbol{x}_-)).$$

  En la Figura 6 se puede ver una representación gráfica del margen geométrico. Como se puede observar, éste depende de la distancia a los ejemplos de cada una de las clases situados justo en la frontera.

Teniendo en cuenta que un mismo hiperplano puede definirse por muchas funciones distintas, de hecho la familia de funciones $\lambda(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b)$ define el mismo hiperplano en el espacio de entrada, para tratar de maximizar cualquier de estos márgenes debemos mantener fijo el otro. Así por ejemplo, si tratamos de maximizar el margen funcional debemos considerar solamente los hiperplanos normalizados, aquellos en los que $\|\boldsymbol{w}\| = 1$, ya que si no sería muy sencillo incrementar el margen funcional sin más que multiplicar la función por un λ cada vez mayor.

Lo más habitual es maximizar el margen geométrico manteniendo fijo el funcional. En este caso, consideramos sólo los *hiperplanos canónicos*, donde los

parámetros $\boldsymbol{w}$ y b están restringidos por las siguientes condiciones que sirven para fijar el margen funcional:

$$\begin{aligned} \min_{+}(\langle \boldsymbol{w}, \boldsymbol{x}_+ \rangle + b) &= +1, \\ \max_{-}(\langle \boldsymbol{w}, \boldsymbol{x}_- \rangle + b) &= -1. \end{aligned}$$

Teniendo en cuenta estas restricciones podemos calcular ya el margen geométrico γ_g para un hiperplano canónico cualquiera:

$$\begin{aligned} \gamma_g &= \min_{+}(d(\boldsymbol{w}, b; \boldsymbol{x}_+)) + \min_{-}(d(\boldsymbol{w}, b; \boldsymbol{x}_-)) \\ &= \min_{+}\left(\frac{|\langle \boldsymbol{w}, \boldsymbol{x}_+ \rangle + b|}{\|\boldsymbol{w}\|}\right) + \min_{-}\left(\frac{|\langle \boldsymbol{w}, \boldsymbol{x}_- \rangle + b|}{\|\boldsymbol{w}\|}\right) \\ &= \frac{|+1|}{\|\boldsymbol{w}\|} + \frac{|-1|}{\|\boldsymbol{w}\|} \\ &= \frac{2}{\|\boldsymbol{w}\|}. \end{aligned}$$

Es decir, que si deseamos maximizar el margen geométrico debemos minimizar la norma de $\boldsymbol{w}$, eso sí, sujeto a buscar únicamente hiperplanos canónicos. Con todo ello tenemos un problema de optimización típico:

$$\begin{array}{lll} \min & \|\boldsymbol{w}\| & \\ \text{s.a.} & (\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \geq +1 & \forall \boldsymbol{x}_i \in +1, \\ & (\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \leq -1 & \forall \boldsymbol{x}_i \in -1. \end{array} \tag{16}$$

Resolviendo este problema obtendremos el hiperplano de margen geométrico máximo que clasifica correctamente todos los ejemplos. Para resolverlo aplicaremos métodos conocidos de optimización de funciones. Los conceptos necesarios, para el tipo de problema planteado, los estudiaremos en la sección siguiente.

4.3. Optimización de funciones

La teoría sobre optimización de funciones estudia la resolución de problemas semejantes al planteado en (16). En esta sección daremos unas pequeñas pinceladas sobre esta teoría y sus métodos, simplemente lo necesario para comprender la resolución del problema que nos atañe. En la sección siguiente aplicaremos esta teoría en la resolución del problema de optimización planteado para maximizar el margen. El lector con conocimientos en optimización puede obviar esta sección.

El objetivo final en los problemas de optimización es obtener los valores que maximizan o minimizan una función, pero considerando que dichos valores deben cumplir una serie de restricciones. La teoría de optimización se ocupa tanto de describir las propiedades de las soluciones óptimas, como del diseño de métodos para obtener dichas soluciones. La estructura general de este tipo de problemas es la siguiente:

$$\begin{array}{lll} \min & f(\boldsymbol{w}) & \boldsymbol{w} \in \Omega, \\ \text{s.a.} & g_i(\boldsymbol{w}) \leq 0 & i = 1, \ldots, k. \end{array} \tag{17}$$

En este problema, que se denomina *problema primal*, se pretenden obtener los valores de las *variables primales*, representadas por $\boldsymbol{w}$, que minimizan la *función objetivo* $f(\boldsymbol{w})$. Las solución está sujeta a que dichos valores respeten las *restricciones de desigualdad*[4] $g_i(\boldsymbol{w})$. Si todas las funciones, f y las distintas g_i, fueran lineales estaríamos ante un problema de programación lineal; si en cambio f es cuadrática y las restricciones son lineales, el problema es de programación cuadrática. Al conjunto de todos los puntos del dominio que cumplen todas las restricciones se le denomina *conjunto admisible* o *factible*. Uno de los elementos del conjunto admisible será el *óptimo* y lo denotaremos por $\boldsymbol{w}^*$. El óptimo verifica que para cualquier otro $\boldsymbol{w}$ del conjunto admisible se cumple que $f(\boldsymbol{w}^*) \leq f(\boldsymbol{w})$.

4.3.1. Convexidad

Un aspecto clave en la resolución de estos problemas es la convexidad tanto del dominio de las posibles soluciones, como de la función objetivo. La convexidad de ambos elementos elimina la posibilidad de óptimos locales, el gran enemigo de cualquier estrategia de búsqueda.

Definición 4 *Un dominio Ω es convexo si y solo si el segmento de la recta que une cualquier par de puntos del dominio también está incluido en el dominio,*

$$\Omega \text{ es convexo } \Leftrightarrow \forall \boldsymbol{u}, \boldsymbol{v} \in \Omega, \forall \theta \in (0,1) \text{ entonces } \theta\boldsymbol{u} + (1-\theta)\boldsymbol{v} \in \Omega.$$

En un programa cuadrático, si el dominio Ω es convexo, es evidente que el conjunto admisible también lo es, ya que las restricciones lineales no pueden eliminar la convexidad del dominio Ω: el conjunto admisible que surge de la combinación de ambos elementos seguirá siendo convexo.

Definición 5 *Una función $f : \mathbb{R}^d \to \mathbb{R}$ se dice que es convexa, si $\forall \boldsymbol{v}, \boldsymbol{u} \in \mathbb{R}^d$, y para cualquier $\theta \in (0,1)$,*

$$f(\theta\boldsymbol{v} + (1-\theta)\boldsymbol{u}) \leq \theta f(\boldsymbol{v}) + (1-\theta) f(\boldsymbol{u}).$$

Es decir, que el segmento de la recta que une el valor de la función en cualquier par de puntos, $\boldsymbol{v}$ y $\boldsymbol{u}$, queda por encima del valor de la función en los puntos entre $\boldsymbol{v}$ y $\boldsymbol{u}$.

Si la anterior desigualdad fuera estricta, entonces la función sería estrictamente convexa. Una forma de verificar si una función doblemente diferenciable es convexa, es estudiar su matriz Hessiana. Si está fuese semi-definida positiva, entonces la función sería convexa.

Si tanto el dominio, como la función objetivo y las restricciones son convexas, entonces el problema se dice que es convexo. La propiedad más interesante en este tipo de problemas es la inexistencia de mínimos locales. Eso facilita notablemente la búsqueda del óptimo, ya que el primer mínimo encontrado será también el mínimo global de la función.

Proposición 1 *Si una función f es convexa, entonces cualquier mínimo local $\boldsymbol{w}^*$ es también un mínimo global.*

[4] Al no aparecer en nuestro problema de optimización restricciones de igualdad, no las consideraremos a lo largo de esta explicación.

Esta afirmación puede demostrarse fácilmente. Para cualquier $\boldsymbol{v} \neq \boldsymbol{w}^*$, por definición de mínimo local, existirá un θ suficientemente cerca de 1 tal que,

$$\begin{aligned} f(\boldsymbol{w}^*) &\leq f(\theta \boldsymbol{w}^* + (1-\theta)\boldsymbol{v}) \\ f(\boldsymbol{w}^*) &\leq \theta f(\boldsymbol{w}^*) + (1-\theta) f(\boldsymbol{v}) \\ (1-\theta) f(\boldsymbol{w}^*) &\leq (1-\theta) f(\boldsymbol{v}) \end{aligned}$$

luego $f(\boldsymbol{w}^*) \leq f(\boldsymbol{v})$ para cualquier $\boldsymbol{v}$, y por tanto $\boldsymbol{w}^*$ es mínimo global.

4.3.2. Teoría de Lagrange

Esta teoría nos permite caracterizar la solución y nos indica cómo podemos obtenerla. Originalmente fue establecida por Lagrange a finales del siglo XVIII para problemas sin restricciones, y fue completada por Kuhn y Tucker a mediados del siglo XX añadiendo el caso en el que existan restricciones de desigualdad. Los dos principales elementos de la teoría son los multiplicadores de Lagrange y la función lagrangiana, basada en incorporar a la función objetivo las restricciones del problema.

Definición 6 *Dado el problema de optimización formulado en (17), con función objetivo $f(\boldsymbol{w})$, definida sobre el dominio $\Omega \subseteq \mathbb{R}^d$ limitado por k restricciones $g_i(\boldsymbol{w}) \leq 0$, se define la función lagrangiana como*

$$L(\boldsymbol{w}, \boldsymbol{\alpha}) = f(\boldsymbol{w}) + \sum_{i=1}^{k} \alpha_i g_i(\boldsymbol{w}),$$

donde los distintos α_i se denominan multiplicadores de Lagrange y deben tener valores no negativos.

A los multiplicadores de Lagrange también se les denomina variables duales, en contraposición a las variables del problema primal. Intuitivamente, nos indicarán la importancia de cada restricción, a mayor valor, más difícil es cumplir la restricción.

A partir de la función lagrangiana, podemos definir el *problema dual* del siguiente modo:

Definición 7 *El problema dual del problema primal planteado en (17) es*

$$\begin{aligned} \text{máx} \quad & W(\boldsymbol{\alpha}) = \inf_{\boldsymbol{w} \in \Omega} L(\boldsymbol{w}, \boldsymbol{\alpha}), \\ s.a. \quad & \alpha_i \geq 0. \end{aligned} \tag{18}$$

4.3.3. Dualidad

El interés del problema dual radica en que, bajo ciertas condiciones, al resolverlo obtenemos también la solución del problema primal asociado. La ventaja de esta estrategia se basa en que puede ser más sencillo resolver el problema dual que el primal, siempre que sus restricciones sean más simples. Para comprender esta estrategia introduciremos dos teoremas que sirven para caracterizar las propiedades de esta dualidad. El primero de ellos es el teorema de la dualidad débil.

Teorema 2 *Sea $\boldsymbol{w} \in \Omega$ una solución admisible del problema primal (17) y $\boldsymbol{\alpha}$ una solución admisible del problema dual (18), entonces $W(\boldsymbol{\alpha}) \leq f(\boldsymbol{w})$.*

La demostración es bastante sencilla,

$$\begin{aligned} W(\boldsymbol{\alpha}) &= \inf_{\boldsymbol{v}\in\Omega} L(\boldsymbol{v}, \boldsymbol{\alpha}) \leq \\ &\leq L(\boldsymbol{w}, \boldsymbol{\alpha}) = f(\boldsymbol{w}) + \sum_{i=1}^{k} \alpha_i g_i(\boldsymbol{w}) \leq \\ &\leq f(\boldsymbol{w}), \end{aligned}$$

dado que la admisibilidad de $\boldsymbol{w}$ y $\boldsymbol{\alpha}$ implica que el sumatorio derivado de las restricciones será una cantidad negativa (las funciones $g_i(\boldsymbol{w})$ serán negativas y los α_i son positivos).

Este teorema nos permite sacar dos conclusiones:

- El valor del problema dual está acotado superiormente por el primal

$$\sup\{W(\boldsymbol{\alpha}) : \alpha_i \geq 0\} \leq \text{ínf}\{f(\boldsymbol{w}) : g(\boldsymbol{w}) \leq 0\}.$$

- Si $f(\boldsymbol{w}^*) = W(\boldsymbol{\alpha}^*)$, respetándose las restricciones $g_i(\boldsymbol{w}^*) \leq 0$ y $\alpha_i \geq 0$, entonces $\boldsymbol{w}^*$ y $\boldsymbol{\alpha}^*$ son, respectivamente, las soluciones del problema primal y dual.

El interés de este teorema radica en que nos da una pista para resolver simultáneamente los problemas primal y dual. Detectaremos que hemos alcanzado la solución comparando los valores en las funciones f y W, cuando la diferencia sea muy cercana a cero habremos resuelto ambos problemas. La no existencia de diferencia entre ambos valores indica la presencia de un punto de silla en la función lagrangiana. Esta idea es la que utilizan muchos de los algoritmos que resuelven programas cuadráticos, entre ellos algunas implementaciones de las SVM.

El teorema de Kuhn-Tucker nos indica las condiciones que deben cumplirse para que podamos resolver el problema primal gracias al dual.

Teorema 3 *Sea el problema de optimización (17), donde tanto el dominio Ω como f son convexas y las restricciones g_i son funciones lineales, las condiciones necesarias y suficientes para que un punto $\boldsymbol{w}^*$ sea óptimo, es que exista $\boldsymbol{\alpha}^*$ tal que*

$$\begin{aligned} \frac{\partial L(\boldsymbol{w}^*, \boldsymbol{\alpha}^*)}{\partial \boldsymbol{w}} &= 0 \\ \alpha_i^* g_i(\boldsymbol{w}^*) &= 0, \quad i = 1, \ldots, k, \\ g_i(\boldsymbol{w}^*) &\leq 0, \quad i = 1, \ldots, k, \\ \alpha_i^* &\geq 0, \quad i = 1, \ldots, k. \end{aligned}$$

Al conjunto de todas estas condiciones se les denomina condiciones de Karush-Kuhn-Tucker o condiciones KKT. La primera de ellas surge de forma natural considerando la definición de la función W como el ínfimo de la función lagrangiana, punto en el que la parcial respecto a $\boldsymbol{w}$ debe ser cero. Las dos últimas

son las restricciones de ambos problemas, el primal y el dual. Por tanto, la más interesante y la única novedosa es la segunda condición, que se denomina condición complementaria. En caso de cumplirse, estamos garantizando que el valor en los óptimos del problema primal y el dual coinciden ($f(\boldsymbol{w}^*) = W(\boldsymbol{\alpha}^*)$), ya que todos los sumandos añadidos a la función lagrangiana serían iguales a cero.

Una lectura, desde un punto de vista más práctico, de la condición complementaria nos indica que para las restricciones g_i activas, aquellas que valen exactamente cero, su multiplicador de Lagrange podrá ser mayor o igual que cero. Sin embargo, para las condiciones inactivas, las que valgan estrictamente menos que cero, el multiplicador asociado debe ser cero. Como veremos más adelante, este aspecto es el que determina una propiedad muy importante de las SVM, como es la dispersión de la solución.

El método de resolución, aunque de cierta complejidad teórica, es sencillo de llevar a la práctica. Partiendo del problema primal, lo transformaremos en el dual sin más que igualar a cero las derivadas parciales de la función lagrangiana respecto a las variables primales, y sustituyendo las relaciones obtenidas de nuevo en la función lagrangiana. Esto eliminará la dependencia de las variables primales. La función dual contendrá como variables los multiplicadores de Lagrange y deberemos maximizarla teniendo en cuenta que todos ellos deben ser no negativos. Si se cumplen todas las condiciones KKT descritas anteriormente, entonces al resolver el problema dual tendremos también resuelto el primal aplicando las relaciones obtenidas al igualar a cero las derivadas parciales de la función lagrangiana. El método quedará más claro cuando en la siguiente sección lo apliquemos sobre nuestro problema original, en el que pretendemos maximizar el margen del hiperplano de clasificación.

4.4. Hiperplano de margen máximo (continuación)

Habíamos dejado planteado el problema de optimización convexa (16) que nos debe permitir obtener el hiperplano de margen máximo. Cambiaremos la formulación de ese problema de optimización por uno equivalente, más compacto y simple. Modificaremos la función a minimizar y unificaremos las restricciones. La función objetivo ($\|\boldsymbol{w}\|$) la sustituiremos por su equivalente ($\langle \boldsymbol{w}, \boldsymbol{w} \rangle$), más sencillo, e incluiremos una constante multiplicativa positiva ($\frac{1}{2}$) que facilitará la resolución posterior del problema. Por su parte, las expresiones de las restricciones las unificaremos haciendo uso de la clase de los ejemplos. Con todo ello, tendremos el siguiente problema a minimizar, equivalente al anteriormente formulado:

$$\begin{aligned} \text{mín} \quad & \frac{1}{2}\langle \boldsymbol{w}, \boldsymbol{w} \rangle \\ \text{s.a.} \quad & y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \geq 1, \quad i = 1, \ldots, n. \end{aligned} \tag{19}$$

Aplicando la teoría de optimización expuesta en la sección 4.3, la función lagrangiana asociada al problema anterior es:

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\langle \boldsymbol{w}, \boldsymbol{w} \rangle - \sum_{i=1}^{n} \alpha_i \left[y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) - 1 \right],$$

siendo los $\alpha_i \geq 0$ las variables duales también llamados multiplicadores de Lagrange. El signo menos que precede al sumatorio surge al ser las restricciones $\geq$

en lugar de $\leq$. El problema dual se determina diferenciando la función lagrangiana respecto a las variables primales, en este caso $\boldsymbol{w}$ y b,

$$\begin{aligned}\frac{\partial L(\boldsymbol{w}, b, \boldsymbol{\alpha})}{\partial \boldsymbol{w}} &= \boldsymbol{w} - \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i = 0, \\ \frac{\partial L(\boldsymbol{w}, b, \boldsymbol{\alpha})}{\partial b} &= \sum_{i=1}^{n} \alpha_i y_i = 0.\end{aligned}$$

A partir de la derivada respecto a $\boldsymbol{w}$ obtenemos la relación entre las variables primales y las duales,

$$\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i. \tag{20}$$

Obsérvese que es la misma relación que obtuvimos al kernelizar el perceptrón (véase sección 3.2). Dada esta relación podemos pasar al problema dual sustituyéndola en la función lagrangiana,

$$\begin{aligned}L(\boldsymbol{w}, b, \boldsymbol{\alpha}) &= \frac{1}{2}\langle \boldsymbol{w}, \boldsymbol{w} \rangle - \sum_{i=1}^{n} \alpha_i \left[y_i (\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) - 1 \right] \\ &= \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle - \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \\ &\quad - \sum_{i=1}^{n} \alpha_i y_i b + \sum_{i=1}^{n} \alpha_i \\ &= -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + \sum_{i=1}^{n} \alpha_i\end{aligned}$$

El problema dual queda de la siguiente forma:

$$\begin{aligned}\text{máx} \quad & -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + \sum_{i=1}^{n} \alpha_i \\ \text{s.a..} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0, \\ & \alpha_i \geq 0 \quad i = 1, \ldots, n.\end{aligned} \tag{21}$$

Como puede apreciarse, en el problema dual aparecen productos escalares entre los ejemplos, es decir, que se adapta perfectamente a la aplicación del truco kernel indicado en las secciones anteriores: podemos sustituir los productos escalares entre los ejemplos de entrenamiento por la aplicación de una función kernel. Además, las características que tiene cualquier función kernel nos garantizan que el problema seguirá siendo convexo: la función kernel debe ser simétrica y semidefinida positiva; eso hace que la función a maximizar tenga una Hessiana semidefinida positiva, luego será convexa con todo lo que ello implica.

Nótese que la variable primal b no aparece directamente en el problema dual y deberemos calcular su valor óptimo b^* una vez determinado $\boldsymbol{w}^*$. La forma de hacerlo es bastante simple y viene dada por las restricciones que establecemos al fijar el margen funcional. Dado que sabemos exactamente el valor funcional

mínimo y máximo que deben tener los ejemplos de las clases $+1$ y -1, debe ser $+1$ y -1 respectivamente, calcularemos el b óptimo para que así sea:

$$b^* = -\frac{\text{máx}_-(\langle \boldsymbol{w}^*, \boldsymbol{x}_- \rangle) + \text{mín}_+(\langle \boldsymbol{w}^*, \boldsymbol{x}_+ \rangle)}{2}. \tag{22}$$

Determinados $\boldsymbol{w}^*$ y b^* a partir de la solución del problema dual, podemos construir nuestro clasificador sin más que utilizar el signo al aplicar la función f sobre cualquier $\boldsymbol{x}$:

$$h(\boldsymbol{x}) = \text{sgn}(f(\boldsymbol{x})) = \text{sgn}\left(\sum_{i=1}^{n} \alpha_i^* y_i \langle \boldsymbol{x}_i, \boldsymbol{x} \rangle + b^*\right). \tag{23}$$

Se puede apreciar como la solución es una combinación lineal de los datos de entrenamiento, no necesariamente de todos, sino simplemente de aquellos que tengan un multiplicador de Lagrange distinto de cero. Esos ejemplos son los que se denominan *vectores soporte*. En el caso separable, los vectores soporte serán los ejemplos que estén justo en el margen de cada clase, como puede apreciarse en la Figura 6. El resto de ejemplos tendrá multiplicador cero, es decir, podríamos eliminarlos antes del aprendizaje y se generaría el mismo hiperplano separador.

Como puede apreciarse es una solución similar a la del perceptrón: de nuevo el hiperplano es una combinación de los ejemplos de entrenamiento y la interpretación de los valores α_i es la misma, indican la dificultad del modelo para clasificar los ejemplos, aquellos que sean fáciles de clasificar tendrán un multiplicador cero y los que estén cerca del margen tendrán un valor positivo.

4.5. Propiedades de la solución

Es muy importante estudiar las características de la solución obtenida al buscar el hiperplano de margen máximo:

- **Margen**. Obtenemos la solución que, desde un punto de vista estructural, tiene menor posibilidad de cometer errores futuros y es más robusta ante un posible ruido en el conjunto de entrenamiento.
- **Convexidad**. La solución se obtiene resolviendo un programa de optimización cuadrática, convexo, sin mínimos locales y resoluble en tiempo polinomial.
- **Dualidad y kernels**. Al pasar al dual logramos que el problema sólo dependa de productos escalares entre los ejemplos de entrada, véase problema en la ecuación (21). Esto nos permite realizar el truco kernel, sustituyendo el producto escalar en el espacio de entrada, por el producto escalar en un espacio de características diferente mediante una función kernel:

$$\begin{aligned} \text{máx} \quad & -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) + \sum_{i=1}^{n} \alpha_i \\ \text{s.a.} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0, \\ & \alpha_i \geq 0 \quad i = 1, \ldots, n. \end{aligned} \tag{24}$$

La hipótesis que nos permitirá clasificar los ejemplos utilizará la misma función kernel:

$$h(\boldsymbol{x}) = \operatorname{sgn}(f(\boldsymbol{x})) = \operatorname{sgn}\left(\sum_{i=1}^{n} \alpha_i^* y_i k(\boldsymbol{x}_i, \boldsymbol{x}) + b^*\right).$$

Un cierto conocimiento del problema nos llevará a escoger una función kernel adecuada, es decir, que genere un espacio de características en el que los ejemplos sean más fáciles de separar linealmente. Como siempre, la solución lineal obtenida en el espacio características, se convertirá en no lineal al trasladarla al espacio de entrada.

- **Dispersión**. La solución depende de una parte, posiblemente pequeña, de los ejemplos del conjunto de entrenamiento y se expresa como una combinación lineal de ellos. La condición complementaria de Karush-Kuhn-Tucker nos indica que las soluciones óptimas $\boldsymbol{\alpha}^*$ y $(\boldsymbol{w}^*, b^*)$ deben satisfacer que:

$$\alpha_i^* \left[y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i^* \rangle + b^*) - 1\right] = 0, \quad i = 1, \ldots, n.$$

Esto significa que solamente los ejemplos más cercanos al hiperplano tiene un multiplicador de Lagrange distinto de cero. El resto de ejemplos tendrá un valor de α igual a cero y por tanto no intervendrán en la combinación lineal que determina el hiperplano de separación de las clases. Es decir, si esos ejemplos se eliminasen la solución seguiría siendo exactamente la misma. Los ejemplos que sí intervienen, tienen un valor de α distinto de cero, son los *vectores soporte* que dan nombre al método. En algunas ocasiones, los vectores soporte pueden ser un subconjunto muy pequeño del conjunto de entrenamiento. La dispersión de la solución es un aspecto fundamental desde un punto de vista computacional, a la hora de diseñar algoritmos para resolver el problema de optimización descrito.

4.6. Margen blando

Indudablemente, el problema planteado hasta ahora tiene un escaso interés desde un punto de vista meramente práctico. Es muy difícil encontrar aplicaciones reales en las que los ejemplos sean linealmente separables, la presencia de ruido y la habitual dificultad de las propias aplicaciones lo suele impedir. La única posibilidad pasaría por generar, a partir de ciertos kernels, espacios de características de muy alta dimensión donde los datos fueran más fácilmente separables. Pero en ese caso estamos asumiendo un riesgo adicional, es muy posible que nos sobreajustemos a los datos de entrenamiento al emplear un espacio de hipótesis excesivamente rico. Por todo ello, es necesario generalizar el problema planteado, permitiendo que determinados ejemplos puedan estar mal clasificados por el hiperplano elegido. Debe tenerse en cuenta que cuando el conjunto de entrenamiento no es linealmente separable, encontrar el hiperplano que clasifique correctamente el mayor número de ejemplos es un problema NP-completo.

Desde el punto de vista de nuestro problema de optimización, permitir errores de clasificación en algunos ejemplos equivale a consentir que las restricciones de esos ejemplos puedan ser violadas. Para contemplar estas situaciones, se introduce en cada restricción original una variable de holgura, ξ_i, que indicará

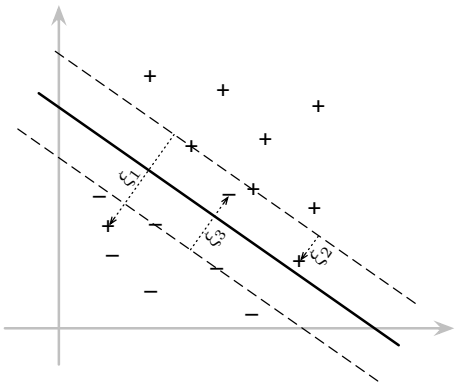


Figura 7: Interpretación de las variables de holgura. Las variables de holgura ξ_i tomarán valor distinto de cero para los ejemplos que no pueden clasificarse correctamente con el suficiente nivel de confianza. En la imagen, los ejemplos 1 y 3 están mal clasificados, mientras el ejemplo 2, aún estando bien clasificado, no lo está en la zona de seguridad determinada por el margen. El valor de las respectivas ξ_i la diferencia funcional entre el valor que toma el ejemplo y el que debería tomar para estar en la región de su clase, es decir, $+1$ ó -1, dependiendo de su clase.

en qué cuantía se viola la condición de clasificar bien ese ejemplo. La expresión de estas nuevas restricciones será

$$y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \geq 1 - \xi_i, \quad i = 1, \ldots, n. \tag{25}$$

Las variables ξ_i tomarán valor cero en el caso de que la restricción original se mantenga, es decir, cuando el ejemplo esté en la zona definida por el margen de su clase, y un valor mayor que cero cuando no lo esté. Esa cantidad reflejará la diferencia funcional entre el valor que toma el ejemplo y el que debería tomar para estar en la región de su clase determinada por el margen. Es decir, no es suficiente con el que el ejemplo esté bien clasificado, debe estar bien clasificado y más allá del margen. El significado de las variables de holgura puede observarse gráficamente en la Figura 7.

Una vez definidas las nuevas restricciones que impondremos para permitir la existencia de ciertos errores de clasificación, nos queda definir la función que pretendemos optimizar. En este caso, es evidente que no basta con maximizar el margen, ya que podríamos lograrlo aún fallando muchísimos ejemplos. La función debe incluir de alguna forma los errores que está cometiendo el hiperplano. Este aspecto se logra añadiendo un término que indique el coste de la solución[5]:

$$\frac{1}{2}\langle \boldsymbol{w}, \boldsymbol{w} \rangle + C \sum_{i=1}^{n} \xi_i. \tag{26}$$

La constante C, que multiplica al término relativo al coste, nos permite controlar en qué grado influye dicho término en relación con la minimización de la

[5]En este texto hemos usado como factor de regularización la $\|\cdot\|_1$ de las variables de holgura. Una formulación también típica sería emplear en su lugar la $\|\cdot\|_2$, quedando la función a minimizar como $\frac{1}{2}\langle \boldsymbol{w}, \boldsymbol{w} \rangle + C \sum_{i=1}^{n} \xi_i^2$

norma, es decir, nos servirá para regular el grado de sobreajuste que permitimos a la hipótesis generada. Esta constante será, junto con el kernel elegido para representar los datos, uno de los dos parámetros clave al aplicar una SVM. Es habitual que el valor de C a emplear se determine empíricamente observando el resultado que producen distintos valores en pruebas, generalmente, de validación cruzada.

Con todo ello, el problema de optimización para el caso general no separable linealmente queda definido de la siguiente forma:

$$\begin{array}{ll} \text{mín} & \frac{1}{2}\langle \boldsymbol{w}, \boldsymbol{w}\rangle + C\sum_{i=1}^{n}\xi_i, \\ \text{s.a.} & y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle + b) \geq 1 - \xi_i, \quad i = 1, \ldots, n, \\ & \xi_i \geq 0, \qquad i = 1, \ldots, n. \end{array} \tag{27}$$

A partir de la definición del problema de optimización, podemos aplicar los métodos de resolución ya conocidos. La función lagrangiana asociada en este caso tiene dos grupos de multiplicadores de Lagrange, uno para cada conjunto de restricciones:

$$L(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\langle \boldsymbol{w}, \boldsymbol{w}\rangle + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\left[y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle + b) - 1 + \xi_i\right] - \sum_{i=1}^{n}\beta_i\xi_i.$$

Las derivadas parciales respecto a las variables del problema primal, en este caso $\boldsymbol{w}$, b y $\boldsymbol{\xi}$, debemos igualarlas a cero para obtener las relaciones entre las variables primales y duales:

$$\begin{aligned} \frac{\partial L(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})}{\partial \boldsymbol{w}} &= \boldsymbol{w} - \sum_{i=1}^{n}\alpha_i y_i x_i = 0, \\ \frac{\partial L(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})}{\partial b} &= \sum_{i=1}^{n}\alpha_i y_i = 0, \\ \frac{\partial L(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})}{\partial \xi_i} &= C - \alpha_i - \beta_i = 0. \end{aligned}$$

Por tanto, las relaciones definitivas entre ambos grupos de variables quedan del siguiente modo:

$$\begin{aligned} \boldsymbol{w} &= \sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i, \\ C &= \alpha_i + \beta_i. \end{aligned}$$

Obsérvese cómo la relación de $\boldsymbol{w}$ con las variables duales sigue siendo exactamente la misma que en el caso separable, luego la hipótesis generada tendrá la misma expresión (ecuación (23)).

Introduciendo estas relaciones de nuevo en la función lagrangiana obtenemos el problema dual que debemos maximizar:

$$L(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j\rangle + \sum_{i=1}^{n}(\alpha_i + \beta_i)\xi_i$$

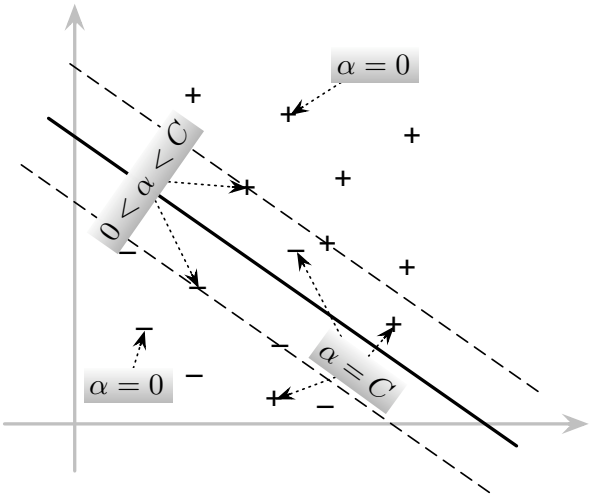


Figura 8: Valores de los multiplicadores de Lagrange. El valor de α de cada ejemplo depende de la posición en la que se sitúe con respecto al hiperplano de separación y a la frontera del margen de cada clase. Los ejemplos en las zonas definidas por el margen de cada clase tienen una valor de α igual a cero, los situados justo en el margen, un valor entre cero y C y los que o bien están mal clasificados o están dentro del margen, un valor de exactamente C.

$$\begin{aligned} & -\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j\rangle - \sum_{i=1}^{n}\alpha_i y_i b \\ & +\sum_{i=1}^{n}\alpha_i - \sum_{i=1}^{n}\alpha_i\xi_i - \sum_{i=1}^{n}\beta_i\xi_i \\ = \quad & -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j\rangle + \sum_{i=1}^{n}\alpha_i \end{aligned}$$

El problema dual definitivo queda de la siguiente forma:

$$\begin{aligned} \text{máx} \quad & -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j\rangle + \sum_{i=1}^{n}\alpha_i \\ \text{s.a..} \quad & \sum_{i=1}^{n}\alpha_i y_i = 0, \\ & 0 \leq \alpha_i \leq C \quad i = 1, \ldots, n. \end{aligned} \tag{28}$$

Como se puede apreciar es prácticamente idéntico al obtenido para el caso separable. La única diferencia es que los valores de los multiplicadores de Lagrange asociados a las restricciones relativas a la clasificación de cada ejemplo, α_i, están acotados superiormente por la constante C. Los valores de α en cada caso se muestran en la Figura 8. El resto de la solución es exactamente igual, incluyendo la expresión de la hipótesis finalmente devuelta (véase ecuación (23)) y, por supuesto, las propiedades comentadas en la sección 4.5 se mantienen íntegramente.

4.7. Regularización

Las máquinas de vectores soporte, como ocurre con cualquier método de aprendizaje, deben ser caracterizadas por el sesgo de aprendizaje que presentan a la hora de seleccionar la hipótesis más adecuada dada una muestra de entrenamiento $\mathcal{S}$. En primer lugar, debe quedar claro que la elección de la función kernel para representar los ejemplos determina el espacio de hipótesis $\mathcal{H}_k$ que el algoritmo explorará. Las SVM siempre buscan un clasificador lineal, la diferencia es que en unos casos lo harán en un espacio de características y en otros casos en otro. El espacio de características depende de la función kernel empleada y del valor de sus parámetros, caso de tenerlos (por ejemplo, el grado en un kernel polinómico).

Una vez elegido el espacio de características, las máquinas de vectores soporte seleccionan la función que cumple la siguiente condición:

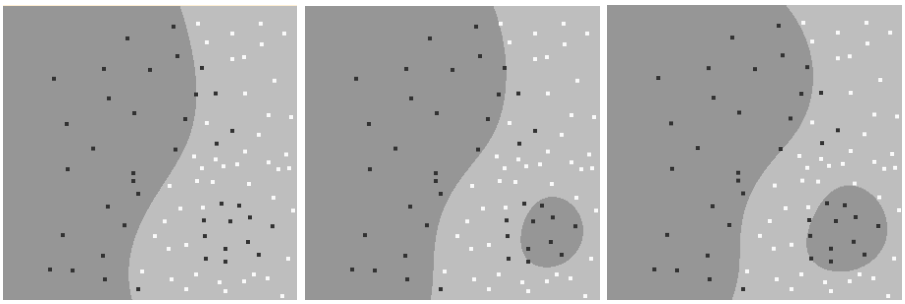
$$\min_{f \in \mathcal{H}_k} \|f\|_{\mathcal{H}_k} + CL(f, S).$$

Es importante interpretar los dos sumandos de esta expresión. El primero de ellos representa la complejidad de la hipótesis elegida. Este factor tiene una larga historia en el campo del aprendizaje y está motivado por el principio de la navaja de Ockham. Muchos sistemas de aprendizaje han tenido en cuenta este aspecto en sus métodos de aprendizaje, incorporando mecanismos para reducir la complejidad de los modelos, por ejemplo, los métodos de poda en árboles de decisión o la medida MDL (*Minimum Description Length*), que mide la complejidad de una hipótesis dados unos datos concretos. En las máquinas de vectores soporte es un aspecto esencial.

Un menor valor de la norma indica una hipótesis más "suave", o dicho de otro modo, más simple. Es decir, minimizar la norma, o maximizar el margen, equivale a ajustarse menos a los datos: podemos obtener un hiperplano de un grandísimo margen si hacemos que los objetos justo en el margen sean los ejemplos más alejados entre sí de las dos clases. Evidentemente, el resto de ejemplos quedarán en el interior del margen (mal clasificados o no clasificados con seguridad, en términos de margen). Tendremos una hipótesis muy simple que seguramente se subajusta a los datos. A medida que el margen va decreciendo, un mayor número de ejemplos estará en las zonas seguras definidas por el margen y menos ejemplos en la zona interior. La hipótesis debe ser más compleja para lograrlo. Si los ejemplos son muy difíciles de separar, el margen se ira reduciendo a medida que la hipótesis sea cada vez más compleja. En este sentido, el valor de la norma de la función nos indica la complejidad de la solución.

El segundo sumando sirve para controlar lo bien que la hipótesis se adapta a los datos de entrenamiento utilizados. Es necesario incluirlo ya que no basta con saber lo grande o pequeña que es la norma de la función, hay que saber si con ese valor de la norma se fallan más o menos ejemplos, o de una forma más general, se tiene un mayor o menor coste. Esa es precisamente la interpretación de ese sumando, el coste que implica una función en los términos del tipo de problema que pretendamos resolver. Dado que queremos minimizar la suma de ambos términos, la función de coste L siempre debe crecer con los errores empíricos cometidos. Es el caso de la función planteada en la sección anterior o la que utilizaremos para resolver la regresión.

La constante C es la que nos permite regular la solución de compromiso entre ambos términos, complejidad y coste. Desde un aspecto puramente práctico es



(a) (b) (c)

Figura 9: Las imágenes representan tres soluciones diferentes para un mismo problema. En cada una de ellas se ha variado mediante el valor de C la importancia de los dos términos de la función a minimizar: la complejidad y el coste. La solución de la imagen (a) es la menos compleja y de más coste (valor más bajo de C), mientras que la de (c) es la más compleja y de menor coste (valor más alto en C). La solución (b) se obtuvo con un valor intermedio de C, y por tanto, es una solución intermedia.

conveniente interpretar el significado de esta constante. A medida que utilizamos valores más altos para C estamos dando más importancia al término que nos mide el coste, con lo que lograremos que el hiperplano se sobreajuste más a los datos, es decir, que considere en mayor medida los *outliers* de cada clase presentes en el conjunto de entrenamiento, ya que el valor de sus α_i crecerá hasta el valor de C e influirá más en el hiperplano final. Por el contrario, valores bajos en la constante C harán que el sistema busque una hipótesis más simple y que, sin duda se sobreajustará menos a los datos, hasta el punto de poder subajustarse. En la Figura 9 se muestra un ejemplo donde se ha variado el valor de la constante C para obtener distintas soluciones en las que se pondera de distinta forma la complejidad y el coste.

La determinación del valor adecuado para C en una aplicación real es quizás más difícil que decidir el kernel a emplear, ya que éste en muchos casos puede venir dado por la naturaleza de los datos y la forma en la que puede medirse la similitud entre ellos.

4.8. Vectores soporte para regresión

Como se ha comentado anteriormente, las máquinas de vectores soporte surgieron para problemas de clasificación binaria. Sin embargo, enseguida se percibió que los principios que las sustentan podrían ser aplicados para resolver otros tipos de problemas. Este es el caso de la regresión lineal. En estos problemas partimos de una muestra de entrenamiento $\mathcal{S} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, donde cada $\boldsymbol{x}_i \in \mathcal{X}$, $y_i \in \mathbb{R}$, y el objetivo es encontrar la función lineal que mejor interpole los valores y_i de los ejemplos del conjunto $\mathcal{S}$. Sin pérdida de generalidad, podemos seguir manteniendo que $\mathcal{X} \subseteq \mathbb{R}^d$, y nuestra hipótesis será directamente

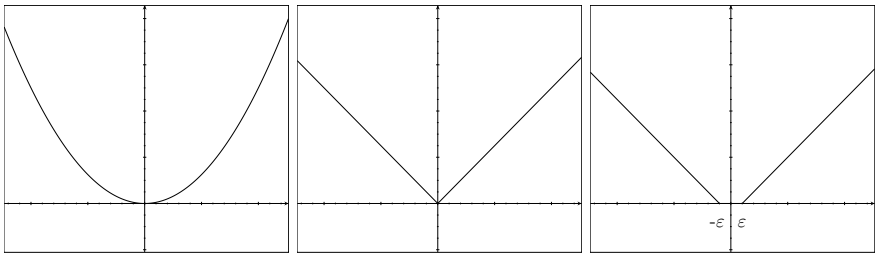


Figura 10: Funciones de pérdida para regresión. En (a) tenemos la función cuadrática empleada en el método de los mínimos cuadrados, en (b) la función de Laplace, menos sensible a los valores extremos y en (c) la función ϵ-insensible, la más empleada con las SVM para regresión.

la función lineal $f(\boldsymbol{x})$:

$$h(\boldsymbol{x}) = f(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = \sum_{i=1}^{n} w_i x_i + b.$$

De la misma forma que hicimos con el margen blando, debemos añadir a la función a optimizar un término regularizador que contemple el coste de los errores que la función comete en la muestra de entrenamiento. En los problemas de regresión tenemos aún más opciones para definir esta función de coste, varias de ellas aparecen en la Figura 10. Alguna de estas funciones está relacionada con algoritmos de regresión conocidos, como es el caso de la función cuadrática empleada en el conocido método de los mínimos cuadrados.

Aunque cualquiera de estas funciones es válida para construir un sistema de regresión basado en vectores soporte, la que más se emplea de todas ellas es la función $\epsilon-$insensible. Básicamente por dos motivos: i) no tiene en cuenta los pequeños errores, todos aquellos menores que el ϵ fijado, y ii) permite mantener una cualidad esencial de las SVM: la dispersión de la solución, con todas las ventajas que eso supone desde un punto de vista computacional. Todos los puntos dentro del tubo definido por ϵ no serán vectores soporte, lo que reducirá significativamente su número.

La expresión de la función $\epsilon-$insensible es:

$$L_\epsilon(y) = \begin{cases} 0 & \text{si } |f(\boldsymbol{x}) - y| < \epsilon \\ |f(\boldsymbol{x}) - y| - \epsilon & \text{en otro caso} \end{cases} \tag{29}$$

En los problemas de regresión necesitamos introducir dos variables de holgura asociadas a cada ejemplo, ξ_i^+ y ξ_i^-, para reflejar la posibilidad de que el valor predicho para cada ejemplo esté por encima o por debajo del que realmente tiene asociado. Nótese que con este significado, $\xi_i^+ \xi_i^- = 0$ ya que sólo una de las dos podrá ser distinta de cero. La interpretación geométrica de estas variables puede apreciarse en la Figura 11.

Con todo ello, podemos seguir la metodología habitual y definir el problema primal a través de la función que deseamos minimizar y de las restricciones que

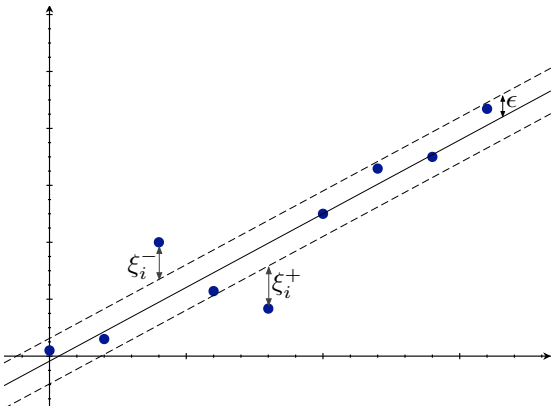


Figura 11: Interpretación de las variables de holgura en problemas de regresión. En las máquinas de vectores soporte para regresión se definen dos variables de holgura para cada ejemplo. La variable ξ^+ toma un valor positivo cuando la predicción del ejemplo es mayor que su valor real en una cantidad superior a ϵ. Por el contrario, la variable ξ^- es mayor que cero si la predicción es menor que la real en una cuantía superior a ϵ. Con esta definición, como mucho una de las dos podrá ser distinta de cero. Para los ejemplos en el interior del tubo-ϵ ambas valdrán cero. De forma general, su producto será igual a cero en cualquier ejemplo.

deben cumplirse:

$$\begin{aligned} \text{mín} \quad & \frac{1}{2}\langle \boldsymbol{w}, \boldsymbol{w}\rangle + C\sum_{i=1}^{n}(\xi_i^+ + \xi_i^-), \\ \text{s.a.} \quad & (\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle + b) - y_i \leq \epsilon + \xi_i^+, \quad i = 1, \ldots, n, \\ & y_i - (\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle + b) \leq \epsilon + \xi_i^-, \quad i = 1, \ldots, n, \\ & \xi_i^+, \xi_i^- \geq 0, \qquad i = 1, \ldots, n. \end{aligned} \tag{30}$$

Siguiendo la metodología conocida obtenemos la función lagrangiana:

$$\begin{aligned} L(\boldsymbol{w}, b, \boldsymbol{\xi}^+, \boldsymbol{\xi}^-, \boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-, \boldsymbol{\beta}^+, \boldsymbol{\beta}^-) \quad = \quad & \frac{1}{2}\langle \boldsymbol{w}, \boldsymbol{w}\rangle + C\sum_{i=1}^{n}(\xi_i^+ + \xi_i^-) \\ & + \sum_{i=1}^{n}\alpha_i^+ \left[(\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle + b) - y_i - \epsilon - \xi_i^+\right] \\ & + \sum_{i=1}^{n}\alpha_i^- \left[y_i - (\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle + b) - \epsilon - \xi_i^-\right] \\ & - \sum_{i=1}^{n}\beta_i^+\xi_i^+ - \sum_{i=1}^{n}\beta_i^-\xi_i^-. \end{aligned}$$

Como siempre, las derivadas parciales respecto a las variables del problema primal, $\boldsymbol{w}$, b, $\boldsymbol{\xi}^+$ y $\boldsymbol{\xi}^-$, debemos igualarlas a cero para obtener las relaciones

entre las variables primales y duales:

$$\begin{aligned}
\frac{\partial L(\boldsymbol{w}, b, \boldsymbol{\xi}^+, \boldsymbol{\xi}^-, \boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-, \boldsymbol{\beta}^+, \boldsymbol{\beta}^-)}{\partial \boldsymbol{w}} &= \boldsymbol{w} + \sum_{i=1}^{n} \alpha_i^+ \boldsymbol{x}_i - \sum_{i=1}^{n} \alpha_i^- \boldsymbol{x}_i = 0, \\
\frac{\partial L(\boldsymbol{w}, b, \boldsymbol{\xi}^+, \boldsymbol{\xi}^-, \boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-, \boldsymbol{\beta}^+, \boldsymbol{\beta}^-)}{\partial b} &= \sum_{i=1}^{n} \alpha_i^+ - \sum_{i=1}^{n} \alpha_i^- = 0, \\
\frac{\partial L(\boldsymbol{w}, b, \boldsymbol{\xi}^+, \boldsymbol{\xi}^-, \boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-, \boldsymbol{\beta}^+, \boldsymbol{\beta}^-)}{\partial \xi_i^+} &= C - \alpha_i^+ - \beta_i^+ = 0, \\
\frac{\partial L(\boldsymbol{w}, b, \boldsymbol{\xi}^+, \boldsymbol{\xi}^-, \boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-, \boldsymbol{\beta}^+, \boldsymbol{\beta}^-)}{\partial \xi_i^-} &= C - \alpha_i^- - \beta_i^- = 0,
\end{aligned}$$

con lo que las relaciones entre el grupo de variables primal y dual es

$$\begin{aligned}
\boldsymbol{w} &= \sum_{i=1}^{n} (\alpha_i^- - \alpha_i^+) \boldsymbol{x}_i, \\
0 &= \sum_{i=1}^{n} (\alpha_i^- - \alpha_i^+), \\
\beta_i^+ &= C - \alpha_i^+, \\
\beta_i^- &= C - \alpha_i^-.
\end{aligned}$$

Reemplazando las variables necesarias en la función lagrangiana obtendremos la expresión final del problema dual:

$$\begin{aligned}
L(\boldsymbol{w}, b, \ldots) &= \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} (\alpha_i^- - \alpha_i^+)(\alpha_j^- - \alpha_j^+) \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + C \sum_{i=1}^{n} (\xi_i^+ + \xi_i^-) \\
&\quad + \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i^+ (\alpha_j^- - \alpha_j^+) \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + \sum_{i=1}^{n} \alpha_i^+ b \\
&\quad - \sum_{i=1}^{n} \alpha_i^+ y_i - \sum_{i=1}^{n} \alpha_i^+ \epsilon - \sum_{i=1}^{n} \alpha_i^+ \xi_i^+ \\
&\quad - \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i^- (\alpha_j^- - \alpha_j^+) \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle - \sum_{i=1}^{n} \alpha_i^- b \\
&\quad + \sum_{i=1}^{n} \alpha_i^- y_i - \sum_{i=1}^{n} \alpha_i^- \epsilon - \sum_{i=1}^{n} \alpha_i^- \xi_i^- \\
&\quad - \sum_{i=1}^{n} (C - \alpha_i^+) \xi_i^+ - \sum_{i=1}^{n} (C - \alpha_i^-) \xi_i^- \\
&= -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} (\alpha_i^- - \alpha_i^+)(\alpha_j^- - \alpha_j^+) \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \\
&\quad + \sum_{i=1}^{n} (\alpha_i^- - \alpha_i^+) y_i - \epsilon \sum_{i=1}^{n} (\alpha_i^- + \alpha_i^+).
\end{aligned}$$

Tras estas largas operaciones, podemos definir el problema dual como

$$\begin{aligned} \text{máx} \quad & -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\alpha_i^- - \alpha_i^+)(\alpha_j^- - \alpha_j^+)\langle \boldsymbol{x}_i, \boldsymbol{x}_j\rangle \\ & +\sum_{i=1}^{n}(\alpha_i^- - \alpha_i^+)y_i - \epsilon\sum_{i=1}^{n}(\alpha_i^- + \alpha_i^+), \\ \text{s.a.} \quad & \sum_{i=1}^{n}(\alpha_i^+ - \alpha_i^-) = 0, \quad i = 1, \ldots, n, \\ & 0 \leq \alpha_i^+, \alpha_i^- \leq C, \qquad i = 1, \ldots, n, \end{aligned} \tag{31}$$

teniendo en cuenta que para resolver simultáneamente el problema primal deben cumplirse además las condiciones KKT:

$$\begin{aligned} \alpha_i^+[(\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle + b) - y_i - \epsilon - \xi_i^+] = 0, \qquad & i = 1, \ldots, n, \\ \alpha_i^-[y_i - (\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle + b) - \epsilon - \xi_i^-] = 0, \qquad & i = 1, \ldots, n, \\ \xi_i^+\xi_i^- = 0, \quad \alpha_i^+\alpha_i^- = 0, \qquad & i = 1, \ldots, n, \\ (C - \alpha_i^+)\xi_i^+ = 0, \quad (C - \alpha_i^-)\xi_i^- = 0, \qquad & i = 1, \ldots, n. \end{aligned}$$

La función final que obtendremos será de nuevo una combinación lineal de los vectores soporte y estará expresada en función de productos escalares entre los ejemplos:

$$f(\boldsymbol{x}) = \sum_{i=1}^{n}(\alpha_i^- - \alpha_i^+)\langle \boldsymbol{x}_i, \boldsymbol{x}\rangle + b^*, \tag{32}$$

donde b^* se calcula de forma análoga a como hicimos en los problemas de clasificación, escogiendo un ejemplo justo en el margen ($0 < \alpha_i < C$) ya que sabemos que la diferencia en valor absoluto entre el valor devuelto por la función y el real será exactamente ϵ:

$$|y_i - f(\boldsymbol{x}_i)| = \epsilon.$$

Como ha ocurrido con los problemas de clasificación, el hecho de que el programa a maximizar dependa de productos escalares en el espacio de entrada, permite sustituirlos por funciones kernel. Ese cambio provoca de una manera elegante que nuestra maquina de vectores, en principio diseñada para realizar regresión lineal en el espacio de entrada, realice regresión no lineal al trasladar la regresión lineal realizada en el espacio de características al espacio de entrada original. La función de regresión obtenida será:

$$f(\boldsymbol{x}) = \sum_{i=1}^{n}(\alpha_i^- - \alpha_i^+)k(\boldsymbol{x}_i, \boldsymbol{x}) + b^*. \tag{33}$$

4.9. Algunas implementaciones

En la actualidad podemos encontrar un importante número de implementaciones de los conceptos teóricos que se han expuesto en este capítulo, así como de herramientas adicionales relacionadas o variantes específicas de algoritmos para su aplicación en contextos determinados. Con respecto a los métodos kernel en general (y SVM en particular) se puede encontrar mucha información en

el portal `http://www.kernel-machines.org/`, donde también hay un apartado reservado para el software desarrollado. En esta sección sólo vamos a reseñar tres herramientas que, probablemente, son las más conocidas en lo respectivo a las SVM.

The Spider (`http://www.kyb.tuebingen.mpg.de/bs/people/spider/`)

Es una librería de objetos para utilizar desde Matlab® en la que se implementan diversos métodos kernel, entre ellos, SVM, con diferentes optimizadores. La programación con esta librería es muy simple y permite desarrollar prototipos de aplicaciones muy rápido. También facilita la experimentación ya que incorpora mecanismos para efectuar entrenamiento, test, selección de modelos, tests estadísticos, representaciones gráficas sencillas, etc. sobre los algoritmos que viene ya implementados o sobre los que el usuario desarrolle.

La implementación de SVM que incorpora puede ser suficiente para comenzar a experimentar con este tipo de algoritmos, aunque para aplicaciones en problemas reales puede ser recomendable utilizar (incluso desde Spider) alguna implementación alternativa, como las que citamos a continuación.

SVM^{light} (`http://svmlight.joachims.org/`)

SVM^{light} es una implementación debida a Thorsten Joachims aplicable a problemas de clasificación, regresión y para aprendizaje de rankings. El algoritmo de optimización empleado se describe en [15]. Esta implementación puede manipular eficientemente problemas con miles de vectores soporte, e incorpora métodos para evaluar el rendimiento de los modelos inducidos, entre los que se incluyen métodos de estimación para algunas medidas habituales en el campo de la recuperación de información.

Una de las últimas ampliaciones de esta implementación ha sido la capacidad de aprender funciones de ranking a partir de conjuntos de preferencias, con el objeto de tratar de ordenar nuevos conjuntos lo mejor posible.

LIBSVM (`http://www.csie.ntu.edu.tw/$\sim$cjlin/libsvm/`)

LIBSVM [5] es una librería que implementa de forma muy eficiente métodos para clasificación, regresión y estimación de probabilidades basados en SVM. También es capaz de abordar problemas de multiclasificación. En la versión actual el algoritmo de optimización utilizado es una adaptación del algoritmo SMO [20] y el mecanismo para convertir los modelos obtenidos por SVM en probabilidades es también una adaptación del propuesto en [21].

La librería tiene un interfaz que permite que un usuario enlace sus aplicaciones con los métodos implementados. Además se han desarrollado interfaces para distintos entornos y lenguajes de programación, tales como Python, R, Matlab, Perl y otros.

5. Lecturas adicionales

Son muchas las referencias bibliográficas que pueden servir para profundizar en el conocimiento sobre los métodos kernel y las máquinas de vectores soporte. El portal `http://www.kernel-machines.org/` recopila y mantiene actualizadas las más importantes. Además, cuenta con una sección dedicada a tutoriales,

generalmente impartidos en congresos internacionales, que pueden ser un buen punto de partida para el lector novel. Entre ellos podemos destacar [4, 29].

En la elaboración de este capítulo se han utilizado muchas textos sobre la materia, aunque dos de ellos de una manera más especial. Por una parte, el libro de Nello Cristianini y John Shawe-Taylor [8] que ofrece una visión bastante sencilla, ideal para principiantes, de las máquinas de vectores soporte y del que se ha tomado el ejemplo del perceptrón y la estructura general de la sección dedicada a las SVM. Por otro lado, se ha seguido el enfoque adoptado en uno de los capítulos de [26] para presentar las funciones kernel como un mecanismo genérico de representación de los datos.

Si se desean conocer con más detalle los aspectos teóricos, recomendamos leer las obras del propio Vapnik [30, 31]. En ellas se ofrecen todas las justificaciones teóricas que sustentan a las máquinas de vectores soporte y que garantizan su capacidad de generalización. También se pueden utilizar algunos libros muy exhaustivos, como [13, 25, 27] y recopilaciones de trabajos como [24, 28].

Además, hay muchas referencias que pueden servir para profundizar en aspectos más concretos introducidos en este capítulo. Para estudiar las condiciones que debe cumplir una función para ser un kernel pueden leerse los trabajos de Mercer [19] y Aronszajn [1]; para revisar conceptos sobre las máquinas de aprendizaje lineal [9] y su posible representación dual [11], para profundizar en la teoría sobre optimización de funciones [2, 10, 18] y especialmente [17], y para otras versiones de los máquinas de vectores soporte dedicadas a problemas de multiclasificación [22, 32] o regresión ordinal [6, 14].

Referencias

[1] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68:337–404, 1950.

[2] M. Bazzara, D. Sherali, and C. Shetty. Nonlinear programming: Theory and algorithms. Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley and Sons, Inc., 1992.

[3] B. E. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Computational Learing Theory*, pages 144–152, 1992.

[4] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[5] C.-C. Chang and C.-J. Lin. LIBSVM : A library for support vector machines. http://www.csie.ntu.edu.tw/∼cjlin/libsvm, 2001.

[6] W. Chu and S. S. Keerthi. New approaches to support vector ordinal regression. In *Proceedings of the ICML '05*, pages 145–152, Bonn, Germany, 2005.

[7] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:237–297, 1995.

[8] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods.* Cambridge University Press, 2000.

[9] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. Wiley and Sons, Inc., New York, 1973.

[10] R. Fletcher. *Practical Methods of Optimization*. John Wiley, 1987.

[11] I. Guyon and D. Stork. Linear discriminant and support vector classifiers. In A. Smola, P. Barlett, B. Schölkopf, and C. Schuurmans, editors, *Advances in Large Margin Classifiers*. MIT Press, 1996.

[12] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer Verlag, New York, 2001.

[13] R. Herbrich. *Learning Kernels Classifiers*. MIT Press, Cambridge, MA, 2002.

[14] R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. In A. Smola, P. Bartlett, B. Scholkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 115–132. MIT Press, Cambridge, MA, 2000.

[15] T. Joachims. *Learning To Classify Text Using Support Vector Machines*, volume 66 of *The International Series in Engineering and Computer Science*. Kluwer, 2002.

[16] I. Jolliffe. *Principal Component Analysis*. Springer Verlag, New York, 1986.

[17] H. Kuhn and A. Tucker. Nonlinear programming. In *Proceedings of 2^{nd} Berkeley Symposium on Mathematical Statistics and Probabilstics*, pages 481–492. Univesity of California Press, 1951.

[18] D. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley, 1984.

[19] J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philos. Trans. Roy. Soc. London*, A 209:415–446, 1909.

[20] J. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods - Support Vector Learning*. MIT Press, Cambridge, MA., 1998.

[21] J. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In A. Smola, P. Bartlett, B. Scholkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 2000.

[22] J. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin dags for multiclass classification. In *Proceedings of the Neural Information Processing Systems (NIPS 99)*, 1999.

[23] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1959.

[24] B. Schölkopf, C. Burges, and A. Smola, editors. *Advances in Kernel Methods – Support Vector Learning*. MIT Press, 1999.

[25] B. Schölkopf and A. Smola. *Learning with Kernels.* MIT Press, Cambridge, MA, 2002.

[26] B. Schölkopf, K. Tsuda, and J.-P. Vert, editors. *Kernel Methods in Computational Biology.* MIT Press, 2004.

[27] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis.* Cambridge University Press, 2004.

[28] A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, editors. *Advances in Large Margin Classifiers.* MIT Press, 2000.

[29] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14:199–222, 2004.

[30] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer Verlag, 1995.

[31] V. Vapnik. *Statistical Learning Theory.* John Wiley, New York, NY, 1998.

[32] J. Weston and C. Watkins. Support vector machines for multi-class pattern recognition. In *Proceedings of the 6^{th} European Symposium on Artificial Neural Networks (ESANN)*, 1999.