Entendiendo las máquinas de soporte vectorial

Una máquina de soporte vectorial (SVM, **Support vector machine**) puede imaginarse como una superficie que crea un límite entre puntos de datos representados en un espacio multidimensional que representan ejemplos y sus valores característicos.

El objetivo de una SVM es crear un límite plano llamado **hiperplano**, que divide el espacio para crear particiones bastante homogéneas en ambos lados. De esta manera, el aprendizaje de la SVM combina aspectos tanto del aprendizaje del vecino más cercano basado en instancias presentado previamente, como del modelado de regresión lineal descrito en el tema anterior. La combinación es extremadamente poderosa y permite que las SVM modelen relaciones altamente complejas.

Aunque las matemáticas básicas que impulsan las SVM existen desde hace décadas, el interés en ellas aumentó enormemente después de que fueron adoptadas por la comunidad de aprendizaje automático. Su popularidad se disparó después de historias de éxito de alto perfil sobre problemas de aprendizaje difíciles, así como del desarrollo de algoritmos SVM premiados que se implementaron en bibliotecas bien respaldadas en muchos lenguajes de programación, incluido R.

Por lo tanto, las SVM han sido adoptadas por una amplia audiencia, que de otra manera no habría podido aplicar las matemáticas algo complejas necesarias para implementar una SVM. La buena noticia es que, aunque las matemáticas pueden ser difíciles, los conceptos básicos son comprensibles.

Las SVM se pueden adaptar para su uso con casi cualquier tipo de tarea de aprendizaje, incluida la clasificación y la predicción numérica. Muchos de los éxitos clave del algoritmo han sido en el reconocimiento de patrones.

Las aplicaciones notables incluyen:

- Clasificación de datos de expresión génica de microarrays en el campo de la bioinformática para identificar el cáncer u otras enfermedades genéticas.
- Categorización de texto, como la identificación del lenguaje utilizado en un documento o la clasificación de documentos por tema.
- La detección de eventos raros pero importantes como fallas de motor, violaciones de seguridad o terremotos.

Las SVM se entienden más fácilmente cuando se utilizan para la clasificación binaria, que es como se ha aplicado tradicionalmente el método. Por lo tanto, en las secciones restantes, nos centraremos únicamente en los clasificadores SVM. Principios como los que se presentan aquí también se aplican cuando se adaptan los SVM para la predicción numérica.

Clasificación con hiperplanos

Como se señaló anteriormente, las SVM utilizan un límite llamado *hiperplano* para dividir los datos en grupos de valores de clase similares. Por ejemplo, la siguiente figura muestra hiperplanos que separan grupos de círculos y cuadrados en dos y tres dimensiones.

Debido a que los círculos y cuadrados se pueden separar perfectamente mediante una línea recta o una superficie plana, se dice que son **linealmente separables**.

Al principio, consideraremos solo un caso simple donde esto es cierto, pero las SVM también se pueden extender a problemas donde los puntos no son linealmente separables.

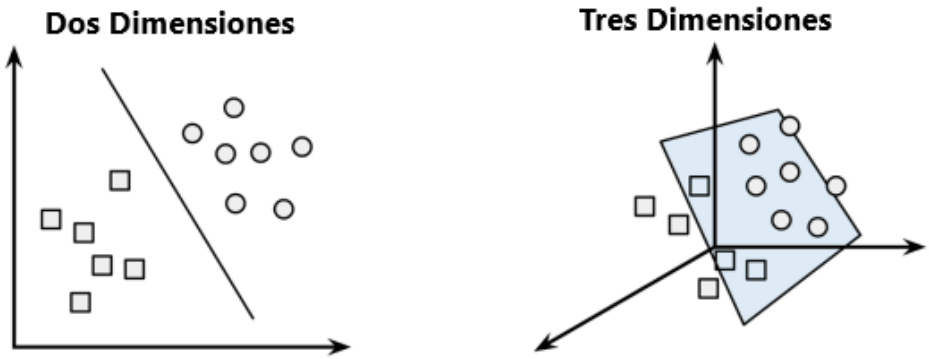

Figura 1: Los cuadrados y círculos son linealmente separables tanto en dos como en tres dimensiones.

Para mayor comodidad, el hiperplano se representa tradicionalmente como una línea en el espacio 2D, pero esto se debe simplemente a que es difícil ilustrar el espacio en más de 2 dimensiones. En realidad, el hiperplano es una superficie plana en un espacio de alta dimensión, un concepto que puede resultar difícil de comprender.

En dos dimensiones, la tarea del algoritmo SVM es identificar una línea que separe las dos clases. Como se muestra en la siguiente figura, hay más de una opción de línea divisoria entre los grupos de círculos y cuadrados.

Tres de esas posibilidades se denominan a, b y c. ¿Cómo elige el algoritmo?

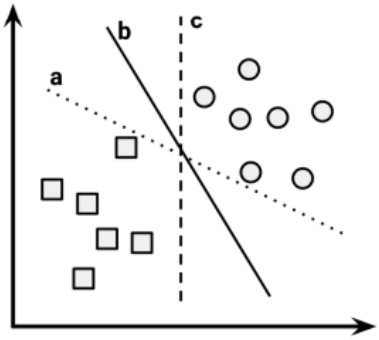

Figura 2: Tres de las muchas posibles líneas que dividen los cuadrados y los círculos.

La respuesta a esa pregunta implica una búsqueda del hiperplano de margen máximo (MMH, **Maximum margin hyperplane**) que crea la mayor separación entre las dos clases. Aunque cualquiera de las tres líneas que separan los círculos y los cuadrados clasificaría correctamente todos los puntos de datos, se espera que la línea que conduce a la mayor separación generalice mejor a los datos futuros.

El margen máximo mejorará la posibilidad de que, incluso si se agrega ruido aleatorio, cada clase permanezca en su propio lado del límite.

Los **vectores de soporte** (indicados por flechas en la siguiente figura) son los puntos de cada clase que están más cerca del MMH. Cada clase debe tener al menos un vector de soporte, pero es posible tener más de uno. Los vectores de soporte por sí solos definen el MMH. Esta es una característica clave de las SVM; los vectores de soporte proporcionan una forma muy compacta de almacenar un modelo de clasificación, incluso si la cantidad de características es extremadamente grande.

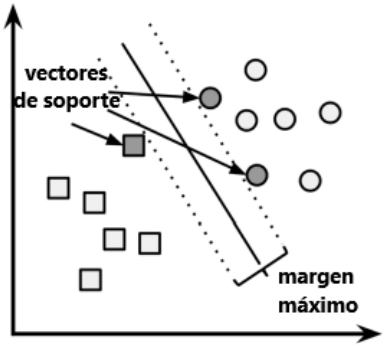

Figura 3: El MMH está definido por los vectores de soporte.

El algoritmo para identificar los vectores de soporte se basa en la geometría vectorial e implica algunas matemáticas complicadas que quedan fuera del alcance de este curso. Sin embargo, los principios básicos del proceso son sencillos.

Se puede encontrar más información sobre las matemáticas de las SVM en el artículo clásico Support-Vector Networks, Cortes, C y Vapnik, V, Machine Learning, 1995, vol. 20, págs. 273-297.

Se puede encontrar un análisis de nivel principiante en Support Vector Machines: Hype or Hallelujah?, Bennett, KP y Campbell, C, SIGKDD Explorations, 2000, vol. 2, págs. 1-13.

Se puede encontrar una mirada más profunda en Support Vector Machines, Steinwart, I y Christmann, A, Nueva York: Springer, 2008.

El caso de datos linealmente separables

Encontrar el margen máximo es más fácil bajo el supuesto de que las clases son linealmente separables. En este caso, el MMH está lo más alejado posible de los límites exteriores de los dos grupos de puntos de datos. Estos límites exteriores se conocen como la envoltura convexa. El MMH es entonces la bisectriz perpendicular de la línea más corta entre las dos envolturas convexas. Los algoritmos informáticos sofisticados que utilizan una técnica conocida como optimización cuadrática pueden encontrar el margen máximo de esta manera.

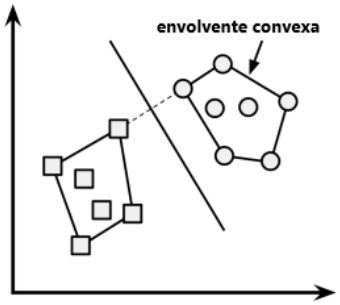

Figura 4: El MMH es la bisectriz perpendicular del camino más corto entre envolturas convexas.

Un enfoque alternativo (pero equivalente) implica una búsqueda a través del espacio de cada hiperplano posible para encontrar un conjunto de dos planos paralelos que dividan los puntos en grupos homogéneos pero que estén lo más alejados posible entre sí. Para utilizar una metáfora, uno puede imaginar este proceso como tratar de encontrar el colchón más grueso que pueda subir por una escalera hasta tu dormitorio.

Para entender este proceso de búsqueda, necesitaremos definir exactamente qué queremos decir con un hiperplano. En un espacio *n*-dimensional, se utiliza la siguiente ecuación:

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

Si no estás familiarizado con esta notación, las flechas sobre las letras indican que son vectores en lugar de números individuales. En particular, *w* es un vector de *n* pesos, es decir, $\{w_1, w_2, ..., w_n\}$, y *b* es un número único conocido como sesgo. El sesgo es conceptualmente equivalente al término de intersección en la forma pendiente-intersección que se analiza en el tema de regresión.

Si tienes problemas para imaginar el plano en un espacio multidimensional, no te preocupes por los detalles. Simplemente piensa en la ecuación como una forma de especificar una superficie, de forma muy similar a cómo se utiliza la forma pendiente-intersección ($y = mx + b$) para especificar líneas en un espacio 2D.

Con esta fórmula, el objetivo del proceso es encontrar un conjunto de pesos que especifiquen dos hiperplanos, de la siguiente manera:

$$\vec{w} \cdot \vec{x} + b \geq +1$$
$$\vec{w} \cdot \vec{x} + b \leq -1$$

También necesitaremos que estos hiperplanos se especifiquen de manera que todos los puntos de una clase se encuentren por encima del primer hiperplano y todos los puntos de la otra clase se encuentren por debajo del segundo hiperplano.

Los dos planos deben crear un espacio de manera que no haya puntos de ninguna de las dos clases en el espacio entre ellos. Esto es posible siempre que los datos sean linealmente separables. La geometría vectorial define la distancia entre estos dos planos - la distancia que queremos que sea lo más grande posible - como:

$$\frac{2}{\|\vec{w}\|}$$

Inversamente proporcional

Aquí, ||*w*|| indica la norma euclidiana (la distancia desde el origen hasta el vector *w*). Como ||*w*|| es el denominador, para maximizar la distancia, necesitamos minimizar ||*w*||. La tarea se suele re expresar como un conjunto de restricciones, de la siguiente manera:

$$\min \frac{1}{2} \|\vec{w}\|^2$$
$$s.t. \, y_i(\vec{w} \cdot \vec{x_l} - b) \geq 1, \forall \vec{x}_l$$

Aunque esto parece complicado, en realidad no es demasiado complicado de entender conceptualmente. Básicamente, la primera línea implica que necesitamos minimizar la norma euclidiana (elevada al cuadrado y dividida por dos para facilitar el cálculo). La segunda línea señala que esto está sujeto a (s.t.) la condición de que cada uno de los puntos de datos *yi* se

clasifique correctamente. Ten en cuenta que *y* indica el valor de la clase (transformado en +1 o -1) y la "A" invertida es la abreviatura de "para todos".

Al igual que con el otro método para encontrar el margen máximo, encontrar una solución a este problema es una tarea que es mejor dejar para el software de optimización cuadrática. Aunque puede requerir un uso intensivo del procesador, los algoritmos especializados pueden resolver estos problemas rápidamente incluso en grandes conjuntos de datos.

El caso de los datos no linealmente separables

A medida que trabajamos con la teoría detrás de las SVM, es posible que te preguntes sobre el elefante en la habitación: ¿qué sucede en un caso en el que los datos no son linealmente separables? La solución a este problema es el uso de una **variable de holgura**, que crea un margen suave que permite que algunos puntos caigan en el lado incorrecto del margen. La figura que sigue ilustra dos puntos que caen en el lado incorrecto de la línea con los términos de holgura correspondientes (indicados por la letra griega Xi):

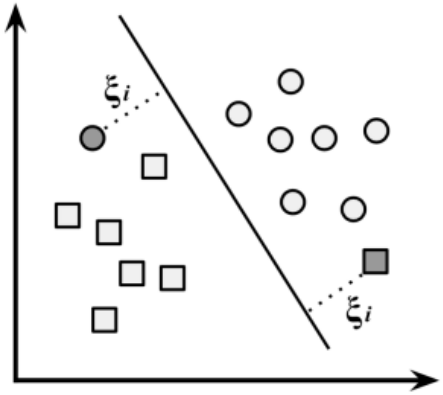

Figura 5: Los puntos que caen en el lado incorrecto del límite tienen una penalización de costo.

Se aplica un valor de costo (indicado como *C*) a todos los puntos que violan las restricciones y, en lugar de encontrar el margen máximo, el algoritmo intenta minimizar el costo total. Por lo tanto, podemos revisar el problema de optimización como:

$$\min \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$
$$s.t.\ y_i(\vec{w}\cdot\vec{x_i} - b) \geq 1, \forall \vec{x}_l, \xi_i \geq 0$$

Si a esta altura estás confundido, no te preocupes, no estás solo. Afortunadamente, los paquetes SVM optimizarán esto para ti sin que tengas que comprender los detalles técnicos. La parte importante que debes comprender es la adición del parámetro de costo, *C*. Modificar este valor ajustará la penalización para los puntos que caen en el lado incorrecto del hiperplano. Cuanto mayor sea el parámetro de costo, más se esforzará la optimización por lograr una separación del 100 por ciento. Por otro lado, un parámetro de costo menor pondrá el énfasis en un margen general más amplio. Es importante lograr un equilibrio entre ambos para crear un modelo que se generalice bien a datos futuros.

Uso de kernels para espacios no lineales

En muchos conjuntos de datos del mundo real, las relaciones entre las variables no son lineales. Como acabamos de descubrir, una SVM puede entrenarse con dichos datos mediante la adición de una variable de holgura, lo que permite clasificar erróneamente algunos ejemplos. Sin embargo, esta no es la única forma de abordar el problema de la no linealidad.

Una característica clave de las SVM es su capacidad de mapear el problema en un espacio de mayor dimensión mediante un proceso conocido como el **truco del kernel**. Al hacerlo, una relación no lineal puede parecer de repente bastante lineal.
Aunque esto parezca una tontería, en realidad es bastante fácil de ilustrar con un ejemplo. En la siguiente figura, el diagrama de dispersión de la izquierda representa una relación no lineal entre una clase meteorológica (soleado o nevado) y dos características: latitud y longitud.

Los puntos en el centro del gráfico son miembros de la clase nevado, mientras que los puntos en los márgenes son todos soleados. Estos datos podrían haberse generado a partir de un conjunto de informes meteorológicos, algunos de los cuales se obtuvieron de estaciones cercanas a la cima de una montaña, mientras que otros se obtuvieron de estaciones alrededor de la base de la montaña.

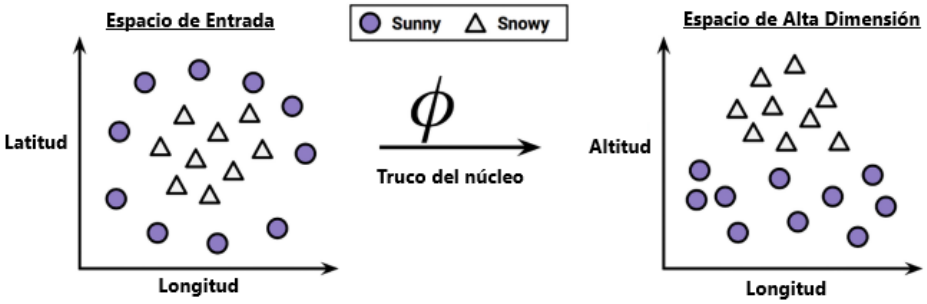

Figura 6: El truco del núcleo puede ayudar a transformar un problema no lineal en uno lineal.

En el lado derecho de la figura, después de aplicar el truco del núcleo, observamos los datos a través de la lente de una nueva dimensión: la altitud. Con la incorporación de esta

característica, las clases ahora son perfectamente separables linealmente. Esto es posible porque hemos obtenido una nueva perspectiva de los datos. En la figura de la izquierda, estamos viendo la montaña desde una vista aérea, mientras que en la de la derecha, estamos viendo la montaña desde una distancia a nivel del suelo. Aquí, la tendencia es obvia: el clima nevoso se encuentra en altitudes mayores.

De esta manera, las SVM con núcleos no lineales agregan dimensiones adicionales a los datos para crear separación. Esencialmente, el truco del núcleo implica un proceso de construcción de nuevas características que expresan relaciones matemáticas entre las características medidas. Por ejemplo, la característica Altitud se puede expresar matemáticamente como una interacción entre Latitud y Longitud: cuanto más cerca esté el punto del centro de cada una de estas escalas, mayor será la Altitud.

Esto permite que la SVM aprenda conceptos que no se midieron explícitamente en los datos originales. Las SVM con núcleos no lineales son clasificadores extremadamente potentes, aunque tienen algunas desventajas, como se muestra en la siguiente tabla:

Fortalezas	**Debilidades**
• Se pueden usar para problemas de clasificación o predicción numérica • No se ven demasiado influenciados por datos ruidosos y no son muy propensos al sobreajuste • Pueden ser más fáciles de usar que las redes neuronales, en particular debido a la existencia de varios algoritmos SVM bien respaldados • Ganaron popularidad debido a su alta precisión y sus victorias destacadas en competencias de minería de datos	• Encontrar el mejor modelo requiere la prueba de varias combinaciones de núcleos y parámetros del modelo • Pueden ser lentos de entrenar, en particular si el conjunto de datos de entrada tiene una gran cantidad de características o ejemplos • Da como resultado un modelo complejo de caja negra que es difícil, si no imposible, de interpretar

Las funciones de núcleo, en general, tienen la siguiente forma. La función denotada por la letra griega phi, es decir, $\phi(x)$, es un mapeo de los datos en otro espacio. Por lo tanto, la función kernel general aplica alguna transformación a los vectores de características x_i y x_j, y los combina utilizando el **producto escalar**, que toma dos vectores y devuelve un solo número:

$$\mathrm{K}(\vec{x}_i, \vec{x}_j) = \phi(\vec{x}_i) \cdot \phi(\vec{x}_j)$$

Utilizando esta forma, se han desarrollado funciones kernel para muchos dominios diferentes. Algunas de las funciones kernel más utilizadas se enumeran a continuación. Casi todos los paquetes de software SVM incluirán estos kernels, entre muchos otros.

El *kernel lineal* no transforma los datos en absoluto. Por lo tanto, se puede expresar simplemente como el producto escalar de las características:

$$K(\vec{x}_i, \vec{x}_j) = \vec{x}_i \cdot \vec{x}_j$$

El **kernel polinomial** de grado *d* agrega una transformación no lineal simple de los datos:

$$K(\vec{x}_i, \vec{x}_j) = (\vec{x}_i \cdot \vec{x}_j + 1)^d$$

El **kernel sigmoidal** da como resultado un modelo SVM algo análogo a una red neuronal que utiliza una función de activación sigmoidal. Las letras griegas kappa y delta se utilizan como parámetros del kernel:

$$K(\vec{x}_i, \vec{x}_j) = \tanh(\kappa\, \vec{x}_i \cdot \vec{x}_j - \delta)$$

El **kernel RBF gaussiano** es similar a una red neuronal RBF. El kernel RBF funciona bien en muchos tipos de datos y se cree que es un punto de partida razonable para muchas tareas de aprendizaje:

$$K(\vec{x}_i, \vec{x}_j) = e^{\frac{-\|\vec{x}_i - \vec{x}_j\|^2}{2\sigma^2}}$$

No existe una regla confiable para hacer coincidir un kernel con una tarea de aprendizaje en particular. El ajuste depende en gran medida del concepto que se va a aprender, así como de la cantidad de datos de entrenamiento y las relaciones entre las características. A menudo, se requiere un poco de prueba y error al entrenar y evaluar varias SVM en un conjunto de datos de validación.

Dicho esto, en muchos casos, la elección del núcleo es arbitraria, ya que el rendimiento puede variar solo levemente. Para ver cómo funciona esto en la práctica, apliquemos nuestro conocimiento de la clasificación SVM a un problema del mundo real.