Máquinas de vector soporte para dos y más clases

Eduardo Mera Gómez

Máquinas de vector soporte para dos y más clases

Eduardo Mera Gómez

Memoria presentada como parte de los requisitos para la obtención del título de Grado en Matemáticas por la Universidad de Sevilla.

Tutorizada por

D.Justo Puerto Albandoz

Índice general

Introducción

La Máquina de Vector Soporte(SVM), nos aporta un algoritmo para, dado un conjunto de clases, predecir a qué clase pertenece una nueva observación.
La idea consiste en separar, de la mejor manera posible, las observaciones de una muestra, mediante hiperplanos en el espacio, según su clase.
Una vez realizado lo anterior, obtendremos una función clasificadora, que servirá para ver en qué lado del hiperplano se encuentra el valor de esa función al evaluar una nueva observación. La idea es asignarle a dicha observación, la clase correspondiente al lado del hiperplano en el que se encuentra.

La aplicación principal de una SVM, consiste en aportar una predicción sobre a qué clase pertenece una nueva observación, gracias a entrenar el algoritmo a partir de una muestra de entrenamiento. Podemos encontrar aplicaciones prácticas en diferentes ámbitos.
En el ámbito de la medicina, a partir de una muestra de pacientes, puede ayudar a predecir si un paciente padece o no una determinada enfermedad.
En el ámbito de las finanzas, a partir de datos de operaciones y créditos de los que ya se conocen los resultados, permite predecir si un nuevo crédito o una nueva operación es viable.

Nuestro objetivo en este trabajo, será el de definir el concepto de SVM, y abordar también los casos de separación no lineal y no separable. Trataremos de descubrir los problemas de optimización que será necesario resolver para obtener la función de clasificación.

En un primer momento, presentamos los conceptos necesarios, como puede ser el caso de cuándo unas observaciones son separables.
Una vez presentados estos conceptos, comenzaremos a desarrollar el problema deseado. Para ello abordaremos el caso en el que el hiperplano de separación es lineal y

las observaciones son separables.
Para este problema, definiremos un determinado margen, que corresponderá a la distancia, ortogonal al hiperplano de separación, de ambos conjuntos. El hecho de que un punto correspondiente a una clase deba estar en el lugar que le corresponde del margen, nos aporta las restricciones del problema. Y el problema de optimización será el de obtener el hiperplano que maximiza dicho margen.

Como todos los datos no son separables, o no conviene separarlos porque obtendríamos un margen demasiado pequeño. El siguiente paso será el de extender el problema anterior al caso no separable.
El problema de optimización a resolver, será igual que el problema anterior pero añadiéndole unos valores de error a las restricciones que permiten que alguna observación no este correctamente clasificada.

Una vez analizado este problema, trataremos extenderlo al caso en el que el separador no sea obligatoriamente lineal. Esto mejora el problema, ya que el hecho de que el separador pueda ser no lineal, relaja las restricciones.
Para llevar a cabo esta generalización, se implementarán determinadas funciones denominadas Kernels, que sustituirán a los productos escalares presentes en el problema, y de las cuales veremos algún ejemplo.

Cuando se haya construido el problema de SVM de 2 clases, no lineal y no separable, trataremos de desarrollar algunos métodos multi-clase.
Dos de ellos estarán basados en la resolución de varias SVM de dos clases, y los otros tratarán de resolver el problema mediante un único problema de optimización.

Por último desarrollaremos en el lenguaje Python, algunos ejemplos de implementación de SVM sobre algunos conjuntos de datos, de los más conocidos en el ámbito de Máquinas de Aprendizaje, para ver los resultados de la precisión en función del Kernel elegido.

Conceptos previos

En este capítulo, trataremos de definir algunos conceptos con los que trabajaremos más adelante. Para ello nos basaremos en diferentes conceptos recogidos en [1].

El "aprendizaje automático" es el algoritmo usado para la estimación de funciones, el término "entrenamiento" es el procedimiento de estimación de los parámetros que son usados en el procedimiento, "test" es el cálculo del valor de las funciones y "rendimiento" , el término usado para la precisión de la generalización.

Definición 0.1. *Una observación es un par de elementos $\{x_i, y_i\}$ donde x_i es el elemento observado e y_i el correspondiente "resultado" asociado a dicha observación.*

Ejemplo 0.1. Un caso práctico podría ser un estudio sobre pacientes de hospital. En dicho estudio, se estudia si un conjunto de pacientes tiene o no una enfermedad. En este caso, la matriz x_i serían los datos recogidos del paciente i e y_i tomaría el valor 1 si el paciente padece la enfermedad y -1 en caso contrario.

Se tratará de definir la máquina como un conjunto de posibles evaluaciones $x \mapsto f(x, \alpha)$. Para ello se asume que los puntos de la observación descrita anteriormente se distribuyen según una cierta distribución de probabilidad $P(x, y)$, dichos datos se suponen independientes e idénticamente distribuidos. Se busca que exista un y determinado para cada valor de x dado.

La máquina aprende la asignación $x_i \mapsto y_i$ de la muestra. Gracias a esta se obtiene una asignación $x \mapsto f(x, \alpha)$ que se usará para evaluar los nuevos datos en la fase "test". Se debe tener en cuenta que α es un parámetro ajustable para que la solución se acerque lo máximo posible a la realidad.

El siguiente problema sería cómo elegir el parámetro α de la mejor forma posible. Para ello se necesita definir los errores que se tratarán de minimizar.

Definición 0.2. *El error esperado de la máquina viene dado por:*

$$R(\alpha) = \int \frac{1}{2}|y - f(x,\alpha)|\partial P(x,y),$$

donde $\partial P(x,y)$ es la derivada de la función de distribución, que puede verse como la densidad $p(x,y)$ en caso de que esta exista.

Definición 0.3. *El error empírico, de una muestra, viene dado por:*

$$R_{emp}(\alpha) = \frac{1}{2l}\sum_{i=1}^{l}|y_i - f(x_i,\alpha)|,$$

donde l es el número de elementos observados.

La diferencia entre ambos errores, consiste en que el error esperado considera el error producido por toda la distribución, mientras que el error empírico solo valora el alcanzado en la muestra. Por este motivo en su fórmula no aparece la función de probabilidad de la distribución.

La cantidad $\frac{1}{2}|y_i - f(x_i,\alpha)|$ en este caso se mueve entre 0 y 1.
En ocasiones, el error esperado es difícil de calcular, por lo que se tratará minimizar una cota superior de dicho error. Para cada $0 \leq \eta \leq 1$ la cota viene dada por:

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{(\frac{h(\log(2l/h)+1) - \log(\eta/4)}{l})},$$

donde $h \in \mathbb{Z}_+ \cup \{0\}$. El valor h es conocido por el nombre de dimensión de Vapnik-Chervonenkis (VC). La dimensión VC aporta una noción de la capacidad de la máquina de vector soporte para clasificar conjuntos de datos.

Esta fórmula determina la forma de elegir el α que minimice el error esperado. Dada una familia de SVMs, familia que se obtiene con la variación de los α, se trata de tomar aquel valor de α que minimice la parte derecha de la desigualdad , una vez conocida h. Gracias a esta minimización se obtiene una cota mínima del límite superior del error esperado.

El conjunto de funciones se denotará $\{f(\alpha)\}$ ya que para cada valor de α se obtiene una función distinta. En el caso de las SVMs de dos clases, $f(x,\alpha) \in \{-1,1\}$.

Observación 0.1. Se dice que dado un conjunto de l puntos ya evaluados, es decir, puntos x_i con su correspondiente asignación $y_i \in \{-1, 1\}$, pueden ser separables por $\{f(\alpha)\}$ si existe un elemento de esta familia tal que $f(x_i, \alpha) = y_i, \forall i \in \{1..l\}$.

Definición 0.4. *La dimensión VC es el máximo número de puntos que pueden ser separados a lo sumo por* $\{f(\alpha)\}$.

0.1 Separación de puntos mediante hiperplanos orientados en $\mathbb{R}^{\mathbb{N}}$

En esto se basa principalmente el problema de SVMs: dado un cierto número de puntos de observación, separarlos lo mejor posible mediante hiperplanos en el espacio orientado hacia aquellos cuya componente y sea 1.

Más tarde esto servirá para ver si un punto dado pertenece a una clase u otra dependiendo de en qué lado del hiperplano se encuentre dicho punto.

La clave principal de este problema es saber cuando un conjunto determinado de observaciones son o no separables, para ello se usa el siguiente teorema.

Lema 0.1. Dos conjuntos de puntos en $\mathbb{R}^{\mathbb{N}}$ pueden ser separados por un hiperplano si y solo si la intersección de sus envolventes convexas es vacía.

Demostración. En esta demostración hay que tener en cuenta que las nociones de puntos en $\mathbb{R}^{\mathbb{N}}$ y la de sus vectores directores se pueden intercambiar. Sean C_A y C_B las envolventes convexas de dos conjuntos de puntos A y B de $\mathbb{R}^{\mathbb{N}}$. Entonces $A-B$ denota el conjunto de puntos cuyos vectores de posición son de la forma $a-b$ con $a \in A$, $b \in B$ (nótese que $A-B$ no contiene al origen). Se denota por $C_A - C_B$ lo mismo para las envolventes convexas. Se quiere probar que el hecho de que A y B sean linealmente separables, es decir, separables mediante hiperplano, equivale a probar que $A-B$ es linealmente separable desde el origen. Suponiendo este último, $\exists w \in \mathbb{R}^{\mathbb{N}}, b \in \mathbb{R}, b < 0$ tal que $x * w + b > 0 \ \forall x \in A-B$. Se toma $y \in B$, y se denota el conjunto de todos los puntos $a-b+y, a \in A, b \in B$ por $A-B+y$. Entonces $x * w + b > y * w \ \forall x \in A-B+y$, y claramente $y * w + b < y * w$, por lo que los conjuntos $A-B+y$ e y son linealmente separables. Repitiendo el proceso obtenemos que $A-B$ es separable del origen si y solo si A y B son linealmente separables. Ahora, se probará que si $C_A \cap C_B = \emptyset$, entonces $C_A - C_B$ es linealmente separable del origen. Claramente $C_A - C_B$ no contiene al origen. Además $C_A - C_B$ es convexo por lo que $\forall x_1 = a_1 - b_1, x_2 = a_2 - b_2, \lambda \in [0,1], a_1, a_2 \in C_A, b_1, b_2 \in C_B$, $(1-\lambda)x_1 + \lambda x_2 = ((1-\lambda)a_1 + \lambda a_2) - ((1 -$

$\lambda)b_1 + \lambda b_2) \in C_A - C_B$. Por lo tanto es suficiente con probar que cualquier conjunto convexo S, el cual no contiene al origen, es linealmente separable por el origen. Basta con utilizar el teorema de separación de conjuntos convexos. Tomando $S_1 = S$ y $S2 = \{O\}$, como S no contiene al origen, $S_1 \cap S_2 = \emptyset$. Por lo tanto, usando el teorema de separación de conjuntos convexos, $\exists p \in \mathbb{R}^{\mathbb{N}}, p \neq 0$, tal que:

$$\text{ínf}\{p'x : x \in S_1\} \geq \sup\{p'x : x \in S_2\},$$

por lo que $\exists p \in \mathbb{R}^{\mathbb{N}}, p \neq 0$ tal que $p'x \geq 0 \; \forall x \in S$. Se ha probado entonces que S es separable con respecto al origen. Por ello $C_A - C_B$ es linealmente separable del origen, y, con más razón, $A - B$ es separable del origen y, en consecuencia, por lo demostrado anteriormente, A es linealmente separable de B.

Queda probar que , si los dos conjuntos de puntos, A y B, son linealmente separables, la intersección de sus envolventes convexas es el vacío. Por hipótesis existe $w \in \mathbb{R}^{\mathbb{N}}, b \in \mathbb{R}$, tal que $\forall a_i \in A, w * a_i + b > 0$ y $\forall b_i \in B, w * b_i + b < 0$. Considerando un punto general $x \in C_A$. Este puede ser escrito como $x = \sum_i \lambda_i a_i, \sum \lambda_i = 1, 0 \leq \lambda_i \leq 1$. Entonces $w * x + b = \sum_i \lambda_i (w * a_i + b) > 0$. Análogamente, para los puntos $y \in C_B, w * y + b < 0$. Por lo que $C_A \cap C_B = \emptyset$, ya que de otra forma se podría encontrar un punto $x = y$ tal que satisficiera ambas inigualdades.

Teorema 0.1. *Sea un conjunto de m puntos en $\mathbb{R}^{\mathbb{N}}$, y se elige un punto como origen. Entonces los m puntos son separables mediante hiperplanos orientados si y solo si los vectores de posición del resto de puntos son linealmente independientes.*

Demostración. Se elige el origen, y se asume que los $m-1$ vectores de posición del resto de puntos son linealmente independientes. Se considera cualquier partición de los m puntos en dos subconjuntos, S_1 y S_2, de órdenes m_1 y m_2 respectivamente. Entonces $m_1 + m_2 = m$. Sea S_1 el subconjunto que contiene al origen. Entonces la envolvente convexa, C_1 de S_1 es el conjunto de puntos cuyo vector posición satisface:

$$x = \sum_{i=1}^{m_1} \alpha_i s_{1i}, \sum_{i=1}^{m_1} \alpha_1 = 1, \alpha_i \geq 0,$$

donde s_{1i} son los vectores de posición de los m_1 puntos de S_1(incluyendo el vector de posición nulo del origen). Similarmente, la envolvente convexa C_2 de S_2 es el conjunto de puntos cuyos vectores de posición x satisfacen:

$$x = \sum_{i=1}^{m_2} \beta_i s_{2i}, \sum_{i=1}^{m_2} \beta_i = 1, \beta_i \geq 0,$$

donde s_{2i} son los vectores de posición de los m_2 puntos de S_2. Ahora se supone que C_1 y C_2 generan intersecćon no vacía. Entonces $\exists x \in \mathbb{R}^{\mathbb{N}}$ el cual a su vez satisface ambas ecuaciones anteriores. Restando dichas ecuaciones, nos da una combinación lineal de los $m-1$ vectores de posición no nulos que es 0, lo que contradice que estos vectores sean linealmente independientes. Por el lema anterior, como C_1 y C_2 tienen intersección vacía, existe un hiperplano que separa a S_1 y S_2. Como es cierto para cualquier partición de los m puntos, estos son separables.

Queda probar que si los $m-1$ vectores de posición no nulos no son linealmente independientes, entonces los m puntos no pueden ser separados mediante hiperplanos orientados. Si los $m-1$ vectores de posición no son linealmente independientes, entonces existen $m-1$ valores, γ_i tal que:

$$\sum_{i=1}^{m-1} \gamma_i s_i = 0.$$

Si todos los γ_i son del mismo signo, entonces se pueden tomar a escala de manera que $\gamma_i \in [0,1]$ y $\sum_{i=1} \gamma_i = 1$. Luego por la ecuación anterior, el origen se encuentra en la envolvente convexa del resto de puntos, por lo que, por el lema, el origen no puede ser separado del resto de puntos por un hiperplano, y los puntos no son separables.

Si los γ_i no son del mismo signo, se pasan todos los términos con γ_i negativos a la derecha:

$$\sum_{j \in I_1} |\gamma_j| s_j = \sum_{k \in I_2} |\gamma_k| s_k$$

donde I_1, I_2 son los índices de la correspondiente partición de S/O(ó,lo que es lo mismo del conjunto S con el origen eliminado). Ahora se escala la ecuación para obtener $\sum_{j \in I_1} |\gamma_j| = 1 y \sum_{k \in I_2} |\gamma_k| \leq 1$ ó $\sum_{j \in I_1} |\gamma_j| \leq 1 y \sum_{k \in I_2} |\gamma_k| = 1$. Se puede considerar sin perdida de generalidad que se tiene el último caso. Entonces $\sum_{j \in I_1} |\gamma_j| s_j$ es el vector posición de un punto que se encuentra en la envolvente convexa de los puntos $\{\cup_{j \in I_1} s_j\} \cup O$, ó, en el caso de igualdad, de los puntos $\{\cup_{j \in I_1} s_j\}$, y $\sum_{k \in I_2} |\gamma_k| s_k$ es el vector de posición de un punto que se encuentra en la envolvente convexa de los puntos $\cup_{k \in I_2} s_k$. Entonces ambas envolventes convexas se superponen y por el lema, los dos conjuntos de puntos no pueden ser separados mediante un hiperplano. En conclusión los m puntos no pueden ser separados.

0.2 La dimensión VC y el número de parámetros

Se podría pensar, que cuantos más parámetros tenga una "máquina de aprendizaje" más capacidad de separación tendrá, y cuántos menos parámetros menos capacidad. Sin embargo, hay un claro contraejemplo encontrado por E.Levin y J.S.Denker (Vapnik,1995): una "máquina de aprendizaje" con un único parámetro que tiene dimensión VC infinita, es decir, se podría separar l puntos, sin importar lo grande que sea el número l.
Se toma como función de paso $\theta(x), x \in \mathbb{R} : \theta(x) = 1 \forall x > 0; \theta(x) = -1 \forall x \leq 0$. Y se considera la familia de funciones de un parámetro definida por:

$$f(x, \alpha) \equiv \theta(sin(\alpha x)), x, \alpha \in \mathbb{R}.$$

Se elige un valor l y se deben encontrar l puntos que sean separables. Para ello se toma:

$$x_i = 10^{-i}, i = 1, .., l.$$

Eligiendo cualquier evaluación:

$$y_1, y_2, .., y_l, y_i \in \{-1, 1\}.$$

Entonces se obtiene dicha evaluación para $f(\alpha)$ tomando :

$$\alpha = \pi(1 + \sum_{i=1}^{l} \frac{(1 - y_i)10^i}{2})$$

Por lo que la dimensión VC es infinita.

0.3 Minimizar el límite minimizando h

Queremos ver como varía la parte derecha de la desigualdad :

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{(\frac{h(\log(2l/h) + 1) - \log(\eta/4)}{l})},$$

dependiendo del valor de h, es decir, de la dimensión VC.

Definición 0.5. *Se denomina confianza VC al segundo término de la parte derecha de la ecuación anterior.*

Dado un nivel de confianza del 95 % y una muestra de entrenamiento de tamaño 10000, se puede observar como la confianza VC es monótona creciente con respecto a h. Esto quiere decir que, dadas ciertas "máquinas de aprendizaje" con error empírico 0, se busca elegir aquella que tiene menor dimensión VC, ya que de esta manera, se reduciría el valor de la parte derecha de la desigualdad descrita al comienzo de este apartado. Si el error empírico es distinto de 0, se busca aquella que minimice la parte derecha de la desigualdad.
Esto no impide que una "máquina de aprendizaje" con el mismo error empírico y una mayor dimensión VC funcione mejor, ya que la desigualdad solo da un límite superior.

0.4 Principio de minimización del error estructural(SRM)(Vapnik,1979).

El término que describe la confianza de VC varía dependiendo del conjunto de funciones, ya que tanto el error empírico como el real lo hacen dependiendo de la elección particular de α, y por lo tanto de la función $f(x,\alpha)$, en el proceso de entrenamiento. Se observa el problema que h no tiene porque variar continuamente ya que es un entero. La manera de resolver este problema consiste en introducir una "estructura" para dividir el conjunto de funciones en ciertos subconjuntos, para que así resulte más sencillo encontrar aquel subconjunto que hace mínimo el límite del error.
Por tanto, "SMR" consiste en encontrar dicho subconjunto. Esto se lleva a cabo definiendo una serie de "máquinas de entrenamiento", una para cada subconjunto, cuya misión consistirá en minimizar el error empírico. Una vez hecho esto se selecciona aquella cuya suma de error empírico y confianza de VC sea mínima.

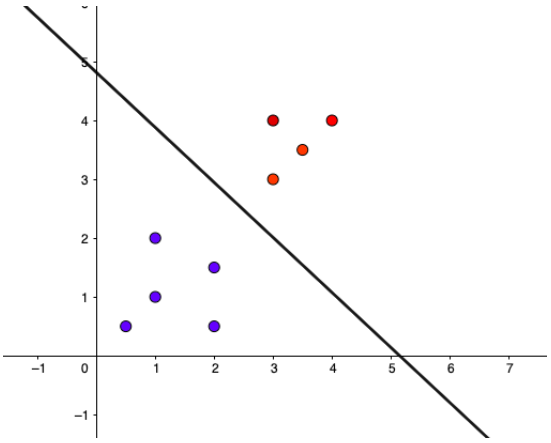
1 Máquinas de vector soporte lineales

En este capítulo, desarrollaremos el concepto de Máquinas de Vector Soporte en el caso de separador lineal. Para ello nos basaremos en los conceptos de programación lineal recogidos en [4] y en los desarrollos de las SVMs recogidos en [1] y [8].

1.1 Caso separable

En esta sección se estudiará el caso en el cual nuestros datos de entrenamiento $\{x_i, y_i\}, i = 1,..l; y_i \in \{-1,1\}, x_i \in \mathbb{R}^d$, vienen dados de forma que son separables mediante un hiperplano lineal como puede ser el siguiente ejemplo.

Ejemplo 1.1. En este ejemplo en $\mathbb{R}^2$ se puede observar como los puntos azules y los rojos, se pueden separar mediante una recta en el espacio.



El hiperplano de separación en cuestión podemos identificarlo con su ecuación general de la forma $w * x + b = 0$, donde w es un vector normal a dicho hiperplano y $|b|/\|w\|$ es la distancia perpendicular al origen de dicho hiperplano, notando $\|w\|$ como la norma euclídea. Si se toma un hiperplano que separe ambos grupos, y dado un nuevo punto, se le asigna un valor de y_i, que depende del lado del hiperplano en el que se encuentre, se realiza el método del hiperplano clasificador.

Se debe tener en cuenta que, en el caso de existir un hiperplano de separación, pueden existir infinitos, debido a que puede obtenerse otro mediante un giro, una traslación... El hiperplano que se suele utilizar es construido de la siguiente manera:

Se denota d_1 la distancia entre el conjunto de los puntos $\{x_i : y_i = 1\}$ y el hiperplano de separación. Igualmente, se denota por d_2 a la distancia entre el conjunto de puntos $\{x_i : y_i = -1\}$ y dicho hiperplano. Se define el margen como $h = d_1 + d_2$ y el problema consistirá en tomar el hiperplano de separación de tal manera que se maximice el margen.
Este método es conocido por el nombre de Clasificador de Máximo Margen (o hiperplano de separación óptima). Es el método más natural, debido a que cuanto mayor sea el margen, existe una mayor confianza de que una observación sea evaluada correctamente.

Para obtener el margen h, se construyen dos hiperplanos paralelos al hiperplano de separación, digamos $H_1 = \{x \in \mathbb{R}^d : w * x + b = 1\}$ y $H_2 = \{x \in \mathbb{R}^d : w * x + b = -1\}$ tal que:

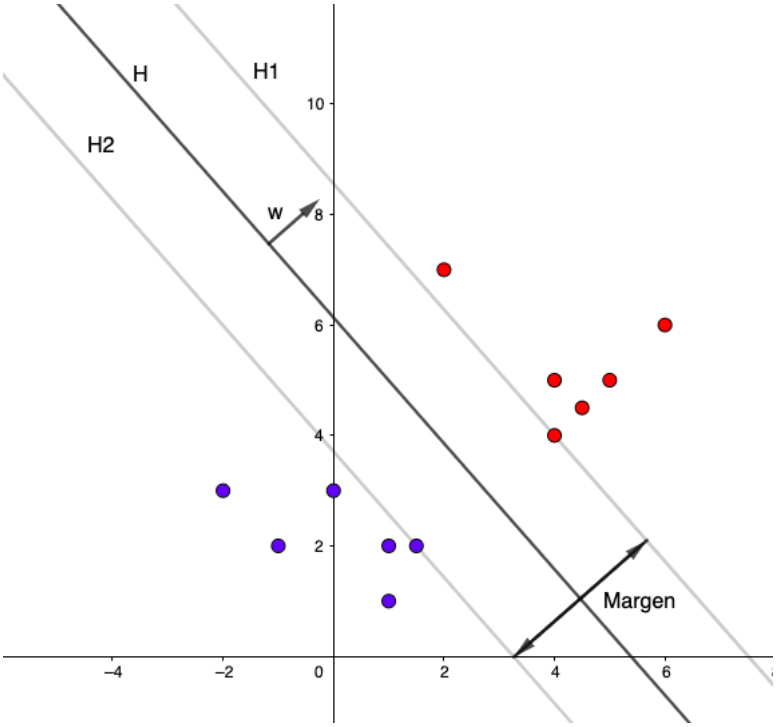
$$x_i * w + b \geq 1 \ \forall x_i | y_i = 1$$

$$x_i * w + b \leq -1 \ \forall x_i | y_i = -1.$$

Combinando ambas desigualdades obtenemos:

$$y_i(x_i * w - b) - 1 \geq 0 \ \forall i.$$

Estos dos hiperplanos son paralelos al hiperplano de separación y cumplen que no hay puntos de entrenamiento entre ambos, por ello teniendo en cuenta que H_1 tiene una distancia al origen de $\frac{|1-b|}{\|w\|}$ y H_2 de $\frac{|-1-b|}{\|w\|}$, se obtiene que si se toma $d_i, i = 1,2$ como la distancia del hiperplano H_i al hiperplano de separación, se tiene $d_1 = d_2 = 1/\|w\|$. Entonces el margen es $h = 2/\|w\|$ y por lo tanto maximizar el margen consiste en minimizar $\|w\|^2$ cumpliendo las restricciones $y_i(x_i * w + b) - 1 \geq 0 \ \forall i$.



Los puntos que se observan que caen sobre los hiperplanos H_1 o H_2 en la figura anterior, son aquellos cuya variación cambiaría el problema y por ello se les conoce como vectores soporte.
Este conjunto de puntos tiene una gran importancia, ya que el margen depende únicamente de ellos. Es decir, el movimiento de alguno de estos puntos, provocaría un cambio en el margen. Sin embargo, el movimiento del resto de puntos, siempre y cuando ese movimiento no haga que el punto cruce al otro lado del hiperplano que marca el margen, no afecta al cálculo del máximo margen.

Se tiene entonces que resolver un problema de minimizar con restricciones. Se busca su formulación lagrangiana ya que esto convierte las restricciones originales, en términos de la función original mediante multiplicadores. Además los datos de entrenamiento aparecerán simplemente como productos escalares de vectores, lo cuál será beneficioso a la hora de generalizar el problema al caso no lineal.

El problema original consiste en:

$$\min_{w} \|w\|^2$$
$$s.a: y_i(x_i * w + b) - 1 \geq 0 \ \forall i.$$

Para pasar a la forma lagrangiana, se toman los multiplicadores de Lagrange, uno por cada una de las restricciones del problema original (al no ser ecuaciones de igualdad todas ellas, se tomará la generalización de Karush-Kunh-Tucker). Si partimos de un problema de la forma:

$$\min_{x} f(x)$$
$$s.a: h_i \leq 0 \ \forall i.$$

En ese caso se obtendría como función lagrangiana:

$$L_D = f(x) + \sum_{i=1}^{l} \alpha_i * h_i$$

con $\alpha_i \geq 0 \ \forall i$. En nuestro caso se tomaría $f(w) = \frac{\|w\|^2}{2}$ y $h_i = 1 - y_i(x_i * w + b) \ \forall i \in \{1, ..., l\}$, obteniendo la siguiente función lagrangiana:

$$L_D = \frac{\|w\|^2}{2} + \sum_{i=1}^{l} \alpha_i(1 - y_i(x_i * w + b)).$$

En este caso como restricciones se obtendría la propia condición $\alpha_i \geq 0$, la fórmula de Lagrange depende de dos variables, w y b , por ello por condición de optimalidad se obtienen las siguientes ecuaciones:

$$\frac{d}{dw_k} L_D = 0$$
$$\frac{d}{db} L_D = 0.$$

Y a todo esto habría que añadirle las condiciones de holgura complementaria y las restricciones del problema original:

$$\alpha_i((x_i * w + b) - 1) = 0 \ \forall i$$
$$y_i(x_i * w + b) - 1 \geq 0 \ \forall i.$$

Por lo que se obtiene el problema siguiente:

$$\begin{array}{ll} \text{mín}_{w,b} & L_P = \dfrac{\|w\|^2}{2} + \sum_{i=1}^{l} \alpha_i (1 - y_i (x_i * w + b)) \\ \text{s.a.} & \frac{d}{dw_k} L_P = 0 \; k \in \{1..d\} \\ & \frac{d}{db} L_P = 0 \\ & \alpha_i \geq 0 \; \forall i \; in\{1..l\} \\ & \alpha_i (y_i (x_i * w + b) - 1) = 0 \; \forall i \\ & y_i (x_i * w + b) - 1 \geq 0 \; \forall i. \end{array}$$

Se toma ahora el problema dual correspondiente al problema anterior, $\text{máx}_\alpha L_D(\alpha)$. Para obtener la ecuación L_D del dual del problema de optimización lagrangiano, usamos las restricciones:

$$\begin{aligned} \nabla_w L_P = 0 \\ \nabla_b L_P = 0, \end{aligned}$$

ya que se obtiene:

$$\begin{aligned} \nabla_w L_P = \|w\| - \sum_{i=1}^{l} \alpha_i y_i x_i = 0 \Rightarrow \|w\| = \sum_{i=1}^{l} \alpha_i y_i x_i \\ \nabla_b L_P = -\sum_{i=1}^{l} \alpha_i y_i = 0 \Rightarrow \sum_{i=1}^{l} \alpha_i y_i = 0. \end{aligned}$$

Por lo que para obtener el valor de L_D basta con sustituir esos valores en L_D ya que buscamos tener el problema:

$$\text{máx}_\alpha (\text{mín}_{w,b} \quad L_P).$$

Por lo que $L_D = \text{mín}_{w,b} L_P$ y dicho mínimo se alcanza para los valores de w y de b calculados anteriormente.
Por ello:

$$\begin{aligned} L_D = \frac{1}{2} \sum_i \alpha_i y_i x_i \sum_j \alpha_j y_j x_j + \sum_i \alpha_i - \sum_i \alpha_i y_i x_i \sum_j \alpha_j y_j x_j = \\ = -\frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i x_j + \sum_i \alpha_i. \end{aligned}$$

1.1.1 Fase de Test

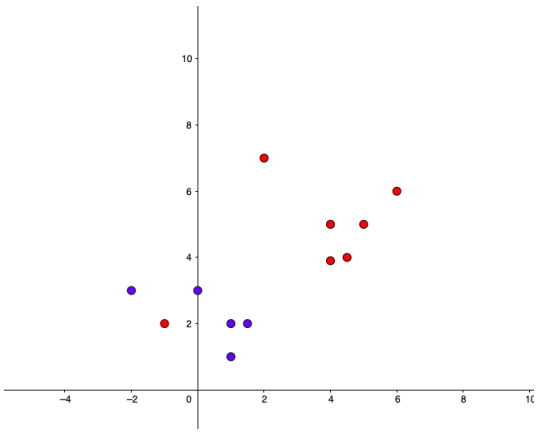
Una vez "entrenada" la máquina de vector soporte, para utilizarla lo único que debemos hacer es ver en qué lado del hiperplano de separación se encuentra el punto x, y se le asigna el valor de y correspondiente a dicho lado del hiperplano. Es decir, sabemos la asignación de x simplemente por el signo obtenido al evaluar la función $f(x) = w * x + b$.

1.2 Caso no separable

En la realidad, podemos encontrarnos con que los puntos de la muestra no son separables, o bien, no es recomendable separarlos completamente. Esto ocurre cuando el hecho de separar todas las observaciones, de manera correcta, provoca que el margen sea muy pequeño.
Como ya se ha comentado anteriormente, el hecho de que el margen sea muy reducido, produce una menor confianza de que las observaciones estén bien asignadas. Por este motivo, en ocasiones, merece la pena dejar que alguna de las observaciones no se encuentre en el lado correcto del margen, o incluso, en el otro lado del hiperplano de separación. Esto ofrece una mayor robustez en las nuevas observaciones individuales.

La extensión del método del Hiperplano de Margen Máximo al caso no separable, es conocida como Clasificador de Vector Soporte.

Ejemplo 1.2. Veamos un caso en el cual los datos de entrenamiento no son separables mediante un hiperplano.



En este caso tenemos un conjunto de valores de entrenamiento que no son completamente separables mediante un hiperplano en el espacio.

El problema es que usando el mismo algoritmo que en el caso separable, en ocasiones el problema es infactible. Por ello como solución a este problema se pueden relajar las restricciones :

$$x_i * w + b \geq 1 \ \forall x_i : y_i = 1$$

$$x_i * w + b \leq -1 \ \forall x_i : y_i = -1.$$

Para ello se introducen unas nuevas variables ξ_i, que serán los errores a cometer, pero solo serán distintas de 0 en el caso de que sea realmente necesario introducir ese error. Luego el siguiente paso es añadir algo que impida que sea distinto de 0 en casos en los que no sea necesario introducir la variable. La forma de resolver este problema es añadir un coste adicional por cada error que se utilice, es decir, cambiar la función a minimizar a $\frac{\|w\|^2}{2} + C(\sum_i \xi_i)^k$ donde C es un parámetro elegido por el usuario que representa si le quiere añadir un coste mayor o menor a usar los errores.
Para todo valor de k el problema es un problema de programación convexo, si $k = 1$ o $k = 2$ entonces es un problema de programación cuadrática y el valor $k = 1$ tiene la ventaja extra de que ni los errores ni sus correspondientes multiplicadores de Lagrange aparezcan en el problema dual.
Introduciendo estas nuevas variables se obtiene el siguiente problema:

$$\begin{aligned} \text{mín}_{w,\xi_i} \quad & \frac{\|w\|^2}{2} + C(\sum_{i=1}^{l} \xi_i)^k \\ \text{s.a.} \quad & y_i(x_i * w + b) - 1 + \xi_i \geq 0 \ \forall i \\ & \xi_i \geq 0 \ \forall i. \end{aligned}$$

Por lo que volviendo a hacer el mismo razonamiento para pasar a su forma Lagrangiana, como se ha hecho en el caso separable, se obtiene el problema:

$$L_P = \text{mín}_{w,b,\xi_i} \quad \frac{\|w\|^2}{2} + C(\sum_{i=1}^{l} \xi_i) - \sum_{i=1}^{l} \alpha_i \{y_i(x_i * w + b) - 1 + \xi_i\} - \sum_{i=1}^{l} \mu_i \xi_i$$

s.a.
$$\frac{d}{dw_v} L_P = \frac{d}{db} L_P = \frac{d}{d\xi_i} = 0 \ \forall i \ \forall v \tag{1.1}$$

$$y_i(x_i * w + b) - 1 + \xi_i \geq 0 \ \forall i \tag{1.2}$$

$$\xi_i, \alpha_i, \mu_i \geq 0 \ \forall i \tag{1.3}$$

$$\alpha_i \{y_i(x_i * w + b) - 1 + \xi_i\} = 0 \ \forall i \tag{1.4}$$

$$\mu_i \xi_i = 0 \ \forall i \tag{1.5}$$

Para ver el valor de L_D como se ha hecho en el caso separable, se buscan los valores de w, b, ξ que hacen mínimo el problema anterior para maximizar en $\alpha y \mu$. Para ello se utiliza (1.1) :

$$\frac{d}{dw_v} L_P = w_v - \sum_i \alpha_i y_i x_i = 0$$

$$\frac{d}{db} L_P = -\sum_i \alpha_i y_i = 0$$

$$\frac{d}{d\xi_i} L_P = C - \alpha_i - \mu_i = 0.$$

Sustituyendo estos valores que son para los que se alcanza el mínimo se obtiene:

$$L_D = \tfrac{1}{2}\sum_{i,j} \alpha_i y_i x_i \alpha_j y_j x_j + \sum_i (\alpha_i + \mu_i)\xi_i - \sum_{i,j} \alpha_i y_i x_i \alpha_j y_j x_j + \sum_i \alpha_i - \sum_i (\alpha_i + \mu_i)\xi_i =$$
$$\tfrac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j x_i x_j + \sum_i \alpha_i - \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i x_j =$$
$$-\tfrac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j x_i x_j.$$

Se tiene que $w = \sum_{i=1}^{NS} \alpha_i y_i x_i$ donde NS es el número de vectores soporte. También se necesita el valor de b. Usando las ecuaciones (1.4) y (1.5) se puede obtener b , ya que usando esta última y la última ecuación obtenida a partir de (1.1) se obtiene que $\xi_i = 0 \ \forall i : \alpha_i \leq C$. Tomando aquellos puntos de entrenamiento tales que $0 \leq \alpha_i \leq C$ y usando la ecuación (1.4) con $\xi_i = 0$ se puede calcular b.

Se podría añadir un valor D, que aportase una restricción sobre la suma de ξ_i, es decir, $\sum_{i=1}^{l} \xi_i \leq D$. En este caso, el valor de D tiene una gran importancia, es elegido mediante el método de validación cruzada.
Se tiene que si el valor $\xi_i > 0$, quiere decir que el punto x_i, se encuentra al otro lado del margen. Si $\xi_i > 1$, entonces el punto x_i se encuentra en el otro lado del hiperplano de separación.
Por este motivo, como $\sum_{i=1}^{l} \xi_i \leq D$, quiere decir que si $D = 0$, todas las observaciones son separadas de manera correcta. En el caso en el que $D \geq 0$, a lo sumo D observaciones se encuentran en el lado incorrecto del hiperplano. Esto es debido a que si el punto x_i, se encuentra en el lado correcto del hiperplano, entonces $\xi_i > 1$ y $\sum_{i=1}^{l} \xi_i \leq D$.

El valor de D controla el equilibrio sesgo-varianza de la técnica de aprendizaje. Si D es pequeño, pocos puntos pueden incumplir la restricción del margen, lo cual

produce un sesgo menor, pero a su vez, una mayor varianza al reducirse la confianza de que un nuevo valor sea asignado correctamente, al reducirse el margen. Por el contrario, si D es grande, un mayor número de puntos puede encontrarse al otro lado del margen, lo cual aumenta el sesgo, pero disminuye la varianza, al conseguir un mayor margen.

2 Máquinas de Vector Soporte no lineales

En este capítulo, extenderemos el concepto de SVM al caso en el que el separador sea no lineal. Para ello, nos basaremos en los conceptos de SVM recogidos en [1] y [8], además de los conceptos de programación no lineal recogidos en [2].

La idea de una máquina de vector soporte no lineal, es la de generalizar el procedimiento de entrenamineto descrito en el capítulo anterior en el caso lineal, para casos no lineales, es decir, de manera que la función decisión (la función $f(x)$ que a cada dato x le da una asignación) no sea obligatoriamente lineal.
La ampliación del caso no separable, al no imponer la linealidad del hiperplano de separación, es lo que se conoce por Máquina de Vector Soporte.
Para ello se implantan una nueva función, la función Kernel, consistente en, una vez observado que los datos de entrenamiento x_i solo aparecen en el algoritmo de entrenamiento como producto escalares de la forma $x_i \cdot x_j$, se realiza una reparametrización de los mismos hacia otro espacio $\mathcal{H}$, que puede ser de dimensión infinita, usando la parametrización $\Phi : \mathbb{R}^d \to \mathcal{H}$, con lo que el problema pasará a depender de $\Phi(x_i) \cdot \Phi(x_j)$. Entonces la función Kernel corresponde a $K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$ y así en el algoritmo de entrenamiento solo aparecerá $K(x_i, x_j)$ sin ser necesario conocer el valor explícito de la función Φ.

Si se toma un Kernel de manera que la dimensión de $\mathcal{H}$ sea infinita, puede no ser sencillo trabajar con Φ de manera explícita. Sin embargo, reemplazando cualquier producto escalar $x_i \cdot x_j$ por su correspondiente $K(x_i, x_j)$ en el algoritmo de entrenamiento, se produce una máquina de vector soporte que "vive" en un espacio infinito, y más aún , emplea el mismo tiempo que lo que se tardaba en entrenar los datos sin reparametrizar.
Hay que tener en cuenta que se sigue haciendo una separación lineal solo que en

un espacio diferente al espacio original.
Ahora la pregunta viene dada por, cómo se utiliza esta máquina. Después de todo, el objetivo sigue siendo el mismo que anteriormente, encontrar w que, en esta ocasión, se encuentra en $\mathscr{H}$. Pero en la fase de evaluación , se trata de calcular el signo de

$$f(x) = \sum_{i=1}^{N_S} \alpha_i y_i \Phi(s_i)\Phi(x) + b = \sum_{i=1}^{N_S} \alpha_i y_i K(s_i, x) + b,$$

donde s_i son los vectores soporte del problema, y se evita calcular $\Phi(x)$ explícitamente y usamos $K(s_i, x) = \Phi(s_i) \cdot \Phi(x)$.
Cuando se escribe $\mathscr{H}$, generalmente se refiere a "High dimensional" por el espacio de mayor dimensión, y se denota $\mathscr{L}$ al espacio donde viven los datos, refiriéndose a "Low dimensional". En este caso, w debe pertenecer al espacio $\mathscr{H}$ pero no por ello debe existir un vector v en $\mathscr{L}$ tal que $\Phi(v) = w$, en el caso de existir, lo que haría sería facilitar el cálculo de $f(x)$ que pasaría a poder ejecutarse en tan solo un paso, evitando la suma y haciendo que la máquina de vector soporte sea N_S veces más rápida.
El objetivo sería encontrar Kernels de manera que la solución encontrada no dependa de la dimensión de ninguno de los dos espacios.

Vamos a introducir algunos ejemplos de Kernels que han sido muy utilizados en la literatura. Se debe tener en cuenta que el Kernel también puede servir para una separación lineal. Basta con tomar $K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}$.

En lugar de tomar este tipo de Kernels, en ocasiones se puede sustituir simplemente por el kernel polinómico de grado d. Esto es $K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d$.
Este Kernel, aporta una mayor flexibilidad a la hora de construir el hiperplano de separación.

Un Kernel no lineal, muy utilizado, es conocido como Kernel radial.
Su forma sería la siguiente: $K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2)$, con γ constante positiva.
La manera de actuar del Kernel radial es muy práctica. Dada una observación nueva x^*, el término que viene dado por $\sum_{j=1}^{p} (x_j^* - x_{ij})^2$, lo que aporta es el cuadrado de la distancia euclídea entre ambos puntos.
Al ser $K(x^*, x_i)$ una exponencial negativa, el hecho de que el nuevo punto x^* esté muy lejos del punto x_i, provoca que el valor del Kernel sea muy pequeño y, por lo tanto, que el punto x_i, no sea muy relevante a la hora de calcular $f(x^*)$.
Esto hace que los puntos más relevantes en dicha evaluación, sean los más cercanos a x^*, lo que hace intuir que la asignación será la correcta.

Normalmente, cuando se habla de máquinas de vector soporte, el espacio $\mathscr{H}$ se refiere a un espacio de Hilbert.

Definición 2.1. *Un espacio de Hilbert es un espacio con un producto escalar definido , el cual a su vez define su correspondiente norma.*

Para ver para que Kernels existen pares $\{\mathscr{H}, \Phi\}$ que cumplen las condiciones deseadas se usará la condición de Mercer (Vapnik,1995;Courant and Hilbert, 1953). Esta condición dice que existe una parametrización Φ y una expansión $K(x,y) = \sum_i \Phi(x)_i \Phi(y)_i$, sí y solo si, para cualquier $g(x)$ tal que :

$$\int g(x)^2 dx$$

es finito, entonces se tiene:

$$\int K(x,y)g(x)g(y)dxdy \geq 0.$$

Pero, ¿qué pasaría si el Kernel que utilizamos no cumple esta condición?. Puede ocurrir entonces que haya datos para los cuales el Hessiano no está definido y por lo tanto no habría solución del problema de optimización. Hay que ser cuidadoso, y no pensar que si el proceso de entrenamiento converge, entonces el Kernel que se ha tomado, funciona sin ningún tipo de problema, ya que ha podido darse el caso de que la muestra de datos de entrenamiento que se tenía, dé como resultado un Hessiano semidefinido positivo, lo cual no garantiza la interpretación geométrica que si se obtendría si cumpliera la condición de Mercer.
En ocasiones, puede ser complicado comprobar la condición de Mercer. Sin embargo para los Kernels del tipo $K(x,y) = (x \cdot y)^p$, las potencias de enteros positivos, es fácil probar que la cumplen. Para ello hay que probar que:

$$\int (\sum_{i=1}^{d} x_i y_i)^p g(x)g(y)dxdy \geq 0.$$

El típico termino en la expansión multinomial de $\sum_{i=1}^{d} x_i y_i$ contribuye a un término de la forma:

$$\frac{p!}{r_1!r_2!\cdots(p-r_1-r_2\cdots)!}\int x_1^{r_1}x_2^{r_2}\cdots y_1^{r_1}y_2^{r_2}\cdots g(x)g(y)dxdy,$$

por lo que se obtiene:

$$\frac{p!}{r_1!r_2!\cdots(p-r_1-r_2\cdots)!}(\int x_1^{r_1}x_2^{r_2}\cdots g(x)dx)^2 \geq 0.$$

Análogamente, todos los Kernels que puedan ser expresados de la forma $K(x,y) = \sum_{p=0}^{\infty} c_p (x \cdot y)^p$, donde c_p son coeficientes reales y positivos y la serie converge uniformemente, satisfacen la condición de Mercer, un hecho notado en (Smola, Schölkopf and Müller,1998b).

Esta condición nos sirve para dado un Kernel, ver si este fuese o no de utilidad, pero, ¿cómo se construye Φ o se consigue saber la forma que tendrá $\mathscr{H}$? En ocasiones puede resultar sencillo construir algunas parametrizaciones una vez conocido el Kernel, más adelante veremos algunos casos. Al final lo que ocurre es que se reparametrizan los datos en un espacio de dimensión bastante grande, es más , el conjunto de los hiperplanos $\{w,b\}$ son parametrizados por $\dim(\mathscr{H})+1$ números. La pregunta es, ¿cómo funciona tan bien la máquina de vector soporte teniendo en cuenta el número de parámetros que puede tener? Un posible motivo es que una vez obtenida la forma de la solución, si hay l muestras de entrenamiento, entonces la solución depende de al menos $l+1$ parámetros ajustables, o lo que es lo mismo, una vez dada la solución, w debe pertenecer a un subespacio de dimensión l.

Obviamente, la parametrización no tiene porque ser sobreyectiva, es más, ni siquiera puede serlo salvo que $\dim(\mathscr{L}) = \dim(\mathscr{H})$. La imagen de Φ no tiene porque ser un espacio vectorial.

Veamos un ejemplo donde, en lugar de buscar Φ, partimos con el valor de Φ (Vapnik.1996).

Ejemplo 2.1. Sea $\mathscr{L} = \mathbb{R}$, un desarrollo de Fourier en los datos x hasta N términos tiene la forma:

$$f(x) = \frac{a_0}{2} + \sum_{r=1}^{N} (a_{1_r} \cos r\,x + a_{2_r} \sin r\,x)$$

que puede verse como producto escalar de dos vectores en $\mathbb{R}^{2N+1}$; $a = (\frac{a_0}{2}, a_{11}, ..., a_{21}, ...)$, y la parametrización $\Phi(x) = (\frac{1}{\sqrt{2}}, \cos x, \cos 2x, ..., \sin x, \sin 2x, ...)$. Entonces el Kernel puede calcularse como:

$$\Phi(x_i) \cdot \Phi(x_j) = K(x_i, x_j) = \frac{\sin((\frac{N+1}{2})(x_i - x_j))}{2\sin(\frac{x_i - x_j}{2})}.$$

Para verlo, se toma $\delta \equiv x_i - x_j$. Entonces:

$$\begin{aligned}\Phi(x_i)\cdot\Phi(x_j) = \quad & \tfrac{1}{2} + \sum_{r=1}^{N} \cos(r x_i)\cos(r x_j) + \sin(r x_i)\sin(r x_j) \\ & -\tfrac{1}{2} + \sum_{r=0}^{N} \cos(r\delta) = -\tfrac{1}{2} + Re\{\sum_{r=0}^{N} \\ & = -\tfrac{1}{2} + Re\{(1 - e^{i(N+1)\delta})/(1 - e^{i\delta})\} \\ & = (\sin((N+1)/2)\delta))/2sin(\delta/2).\end{aligned}$$

2.0.1 Soluciones globales y unicidad

A continuación abordamos el problema de ver cuándo la solución que se obtiene es global y única. Hay una clara forma de que la solución no sea única, que existan dos soluciones donde $\{w, b\}$ sean distintos. Pero también existe otra no tan evidente y es que aunque se tenga que $\{w, b\}$ sea única, puede que ocurrir que el desarrollo de w en la forma $w = \sum_i \alpha_i x_i y_i$ no sea única al no serlo los α_i, veamos un ejemplo:

Ejemplo 2.2. Consideramos el problema de 4 puntos separables en $\mathbb{R}^2$: $x_1 = [1,1], x_2 = [-1,1], x_3 = [-1,-1], x_4 = [1,-1]$ con la asignación de signos $[+,-,-,+]$ respectivamente. Una solución es $w = [1,0], b = 0, \alpha = [0.25, 0.25, 0.25, 0.25]$, pero sin embargo hay otra con w, b iguales, pero $\alpha = [0.5, 0.5, 0, 0]$.

Esto es importante, porque se podría encontrar el desarrollo de w de manera que la fase de test sea más rápida.

No hay que preocuparse por el carácter global de las soluciones, ya que una propiedad conocida de los problemas de programación convexa es que toda solución local es global. En cuanto a la unicidad, en el caso que estamos tratando, si el Hessiano es definido positivo, entonces la función es estrictamente convexa, y tendríamos la unicidad. Esto solo ocurre en los casos donde la función objetivo es estrictamente convexa, si no lo fuese,un Hessiano definido positivo en un punto, implica función estrictamente convexa en ese punto, pero no la otra implicación. Si el Hessiano es semi-definido positivo, puede seguir siendo única la solución pero podría ocurrir que una solución sea distinta por los α_i.

En estos casos hay que tener en cuenta que dada una solución definida por un α, elegimos α' que se encuentre en el subespacio nulo del Hessiano $H_{ij} = y_i y_j x_i \cdot x_j$,

que cumpla también que es ortogonal al vector de unos. Entonces tomando $\alpha_i + \alpha_i^{'}$, si $0 \leq \alpha_i + \alpha_i^{'} \leq C$ y $\alpha^{'}$ satisface : $\sum_i \alpha_i y_i = 0$. Se tiene un valor $\widehat{\alpha} = \alpha + \alpha^{'}$, para el cual, al sustituir α por $\widehat{\alpha}$ en la solución, se convierte también en solución.

Finalmente incluimos un resultado que demuestra que en el caso de que la solución no sea única entonces un punto que es solución puede ser transformado de manera continua en otro que es solución y todos los puntos intermedios son solución.

Teorema 2.1. *Sea la variable X representando al par de variables $\{w, b\}$. Sea el Hessiano para el problema semi-definido positivo y la función objetivo convexa. Sean X_0 y X_1 dos puntos para los cuáles la función objetivo alcanza el mínimo . Entonces existe un camino $X = X(\tau) = (1-\tau)X_0 + \tau X_1, \tau \in [0,1]$, tal que , $X(\tau)$ es solución para todo τ.*

La demostración de este teorema es directa por la condición de convexidad.

3 Métodos de solución

En este capítulo, expondremos algunos métodos de solución, ya que en la mayoría de casos reales es necesaria una resolución numérica del problema. Esto solo puede ser resuelto de manera analítica en el caso en que el número de datos de entrenamiento sea pequeño. Por ello expondremos algunos métodos recogidos en [7].

Los pasos a seguir, son: ver las condiciones de optimalidad de Karush-Kunh-Tucker, definir una estrategia de aproximación al dato óptimo mediante un incremento uniforme del valor de la función objetivo del dual y usar un algoritmo de descomposición para que sólo partes de los datos de entrenamiento sean manejadas en cada momento.

El método que se usará para este fin, es un método de conjunto activo que combina el método de Newton, y el método de ascenso del gradiente conjugado. Esto se realiza, porque ambos métodos por si solos tienen algún inconveniente para este fin. El método de Newton puede resolver problemas con restricciones de igualdad en tan solo un paso, pero hay que tener en cuenta el coste de almacenamiento extra de una factorización del Hessiano proyectado. El método de ascenso del gradiente conjugado, puede resolver problemas cuadráticos en a lo sumo l pasos, con l el número de datos de entrenamiento, pero tiene la desventaja de que solo una nueva restricción es añadida cada vez. Para solventar este problema, se entra en los llamados "Métodos de proyección" donde una vez seleccionado un punto de la región factible, se realizan búsquedas unidimensionales y proyecciones para que el movimiento permanezca dentro de la región factible, lo que permite añadir varias restricciones al mismo tiempo.

Expliquemos en qué consiste el método. Primero, hay que tener en cuenta que cuando se calcula la función objetivo, también se calcula el gradiente con un pequeño coste extra. En cuanto al funcionamiento del método, en un primer momento, las

direcciones de búsqueda, denotadas por s, se encuentran a lo largo del gradiente. Se busca el conjunto de puntos más cercano dentro de la región factible. Si el producto escalar del gradiente con s indica que el máximo a lo largo de s cae entre el punto original y el conjunto de puntos comentado anteriormente, se cálcula el punto óptimo a través de la dirección de búsqueda de manera analítica y se pasa a la segunda fase. De lo contrario se busca un nuevo conjunto de puntos y se repite el paso anterior. En la fase 2 , se realiza el método de ascenso del gradiente conjugado de Polak-Ribiere, hasta que se encuentre un nuevo conjunto de puntos cómo los descritos anteriormente, en cuyo caso se volvería a aplicar el primer paso, o se llegue al criterio de parada. Dicho algoritmo se detiene cuando la razón fracción de aumento de la función objetivo cae por debajo de una cierta tolerancia(normalmente se toma $1.e-10$, para una precisión doble). Las direcciones de búsqueda se toman de manera que los α_i continúen satisfaciendo la restricción $\sum_i \alpha_i y_i = 0$. Es importante que además la proyección sea implementada de tal manera que el ángulo entre la dirección de búsqueda anterior y la resultante se minimice.

Una vez acabado el algoritmo, se puede comprobar si este esta funcionando correctamente, comprobando que la solución satisfaga las condiciones de Karush-Kun-Tucker ya que son condiciones necesarias y suficientes de optimalidad.

Una cosa importante que se debe tener en cuenta en los métodos de solución es la complejidad de dicho método. Tanto en la fase de entrenamiento como en la fase de test, solo depende del Kernel, el cual incluso aunque sea un producto escalar en un espacio infinito o de dimensión muy alta, la complejidad de calcular K puede ser pequeña. Para Kernels de la forma $K = (x_i \cdot x_j)^p$, un producto escalar en $\mathscr{H}$ requiere $\binom{d_L+p-1}{p}$ operaciones, mientras que el cálculo de K requiere d_L operaciones.

En cuanto a la fase de entrenamiento, se dará la complejidad de uno de los algoritmos de entrenamiento (Bunch-Kaufman), en cuyos resultados se tendrá en cuenta que pueden ser usadas diferentes estrategias dependiendo de la situación. Se tiene en cuenta el problema de entrenamiento con solo una "chunk"(porción). Sea l el número de puntos de entrenamiento, N_S el número de vectores soporte, y d_L la dimensión de los datos de entrada. En el caso en el que muchos vectores suporte no estén en el límite superior y $N_S/l << 1$ entonces el número de operaciones es $O(N_S^3 + (N_S^2)l + N_S d_L l)$. Si $N_S/l \approx 1$ entonces es $O(N_S^3 + N_S l + N_S d_L l)$. Para el caso en el que muchos vectores soporte estén en el límite superior, si $N_S/l << 1$ entonces es $O(N_S^2 + N_S d_L l)$ y si $N_S/l \approx 1$ es $O(D_L l^2)$.

El método "chunking" empieza con un subconjunto pequeño y arbitrario de da-

tos y entrena sobre ellos, una vez realizado dicho entrenamiento, se usa el resto de puntos de entrenamiento para probarlo y se anotan los errores cometidos, de aquellos puntos que con el método actual, se le ha asignado un valor (o un lado del hiperplano) que no les corresponde. A continuacuón se genera otra "porción" de los datos, usando los N primeros de estos, combinados con los N_S vectores soporte ya conocidos. A la hora de elegir $N + N_S$ hay que cuidar ciertos detalles como que el tamaño de la nueva "porción" sea el adecuado para la convergencia. Se sigue empleando el método hasta que todas las condiciones de Karush-Kun-Tucker se cumplan.

En la fase de test se debe simplemente evaluar la ecuación:

$$f(x) = \sum_{i=1}^{N_S} \alpha_i y_i \Phi(s_i) \cdot \Phi(x) + b = \sum_{i=1}^{N_S} \alpha_i y_i K(s_i, x) + b.$$

Lo cual requiere $O(MN_S)$ operaciones, donde M es el número de operaciones requeridas para calcular el Kernel.En el caso de los Kernels que son productos escalares o funciones de base radial, M es $O(d_L)$, la dimensión de los vectores de datos.

3.1 Método SMO

Cuando se habla de métodos de solución hay que destacar el método del algoritmo de Optimización Mínima Secuencial (SMO), que es el método más utilizado, puesto que es el más competitivo y sencillo de implementar.

El método "Chunking", o de descomposición, estaba caracterizado por empezar por un subconjunto pequeño de datos y entrenar sobre ellos. El algoritmo SMO realiza algo parecido, con la excepción de que toma un subconjunto de tan solo 2 puntos y los optimiza en cada iteración. La gran ventaja que tiene sobre otros algoritmos es el hecho de que al tratar solo la optimización de dos puntos (a través de una modificación en los multiplicadores) esto puede hacerse de manera analítica.

En el proceso de modificar los multiplicadores para buscar una mejora en el valor de la función objetivo, hay que tener en cuenta que la modificación de un multiplicador, implica la modificación de al menos otro de ellos. Esto es necesario para que la condición $\sum_{i=1}^{l} \alpha_i y_i = 0$ se siga cumpliendo. En un principio, podría parecer que dará lugar a un algoritmo costoso, ya que necesita un mayor número de iteraciones. A pesar de ello como el número de operaciones es muy pequeño para cada iteración, el algoritmo es bastante rápido.

3.1.1 Solución analítica de la optimización de dos puntos

En esta sección se explicará cómo se modifican los multiplicadores y bajo qué criterios, para mejorar el valor de la función objetivo sin dejar de cumplir las restricciones.

La primera restricción que se debe cumplir es $\sum_{i=1}^{l} \alpha_i y_i = 0$. Para ello se debe mantener la proporción. Si se eligen por ejemplo α_1 y α_2, denotando por α^n el nuevo valor de α y por α^o el antiguo valor, se debe mantener $\alpha_1^n y_1 + \alpha_2^n y_2 = \alpha_1^o y_1 + \alpha_2^o y_2$. Añadiendo la restricción $0 \leq \alpha_1, \alpha_2 \leq C$ se puede acotar la solución de los nuevos valores. Veamos los valores en los que se mueve α_2^n Para ello hay que diferenciar dos casos: Si $y_1 \neq y_2$, tomaremos por ejemplo $y_1 = -1, y_2 = 1$. Usando la condición descrita anteriormente sustituyendo los valores de y_1, y_2 se obtiene:

$$\begin{aligned} -\alpha_1^n + \alpha_2^n &= -\alpha_1^o + \alpha_2^o \\ \alpha_2^n &= -\alpha_1^o + \alpha_2^o + \alpha_1^n. \end{aligned}$$

Usando que $\alpha_1^n \geq 0$:

$$\alpha_2^n \geq -\alpha_1^o + \alpha_2^o.$$

Por lo tanto $\alpha_2 \geq \text{máx}\{0, \alpha_2^o - \alpha_1^o\}$.

Por otro lado usando $\alpha_2^n = -\alpha_1^o + \alpha_2^o + \alpha_1^n$ y $\alpha_1^n \leq C$, se tiene $\alpha_2^n \leq C - \alpha_1^o + \alpha_2^o$. Por ello $\alpha_2 \leq \text{mín}\{C, C - \alpha_i^o + \alpha_2^o\}$.

Si $y_1 = y_2$ análogamente, utilizando las mismas ecuaciones pero tomando $y_1 = y_2 = 1$ y despejando α_2^n se obtiene:

$$\text{máx}\{0, \alpha_1^o + \alpha_2^o - C\} \leq \alpha_2^n \leq \text{mín}\{C, \alpha_1^o + \alpha_2^o\}.$$

Para facilitar la notación, tomaremos U como la acotación inferior en cada caso, y V como la cota superior.

El objetivo es ver, analíticamente, cuándo se alcanza el máximo de la función objetivo, en el caso de que solo se puedan variar, por ejemplo, α_1 y α_2. Para este fin, es necesario introducir una definición de error, que será la diferencia entre el valor y_i y el valor de $f(x_i)$. Esto es, $E_i = f(x_i) - y_i = (\sum_{j=1}^{l} \alpha_j y_j K(x_j, x_i) + b) - y_i$. Este error no se considera un error de asignación, ya que si $y_i = 1$ y $f(x_i) = 4$, la asignación es correcta, sin embargo el error es 3. Se necesitará también la segunda derivada de la función objetivo a lo largo de la línea diagonal, que toma el valor $-k$ con $k = K(x_1, x_1) + K(x_2, x_2) - 2K(x_1, x_2) = \|\Phi(x_1) - \Phi(x_2)\|^2$.

Teorema 3.1. *El máximo de la función objetivo del problema de optimización dual, en el caso en el que se pueden modificar tan solo α_1 y α_2, se obtiene para los siguientes valores de α_1 y α_2. Se calcula $\alpha_2^{n,u} = \alpha_2^o + \frac{y_2(E_1-E_2)}{k}$. Se acota para que cumpla las restricciones alcanzadas anteriormente:*

$$\alpha_2^n = \begin{cases} V & si \quad \alpha_2^{n,u} > V \\ \alpha_2^{n,u} & si \quad U \leq \alpha_2^{n,u} \leq V \\ U & si \quad \alpha_2^{n,u} < U \end{cases}$$

Por último, se toma $\alpha_1^n = \alpha_1^o + y_1 y_2(\alpha_2^o - \alpha_2^n)$.

Demostración. Se define
$v_i = \sum_{j=3}^{l} y_j \alpha_j K(x_i, x_j) = f(x_i) - \sum_{j=1}^{2} y_j \alpha_j K(x_i, x_j) - b, i = 1,2.$
y se considera la función objetivo como función de α_1 y α_2:

$$W(\alpha_1, \alpha_2) = \alpha_1 + \alpha_2 - \frac{1}{2}K_{11}\alpha_1^2 - \frac{1}{2}K_{22}\alpha_2^2 - y_1 y_2 K_{12}\alpha_1\alpha_2 - y_1\alpha_1 v_1 - y_2\alpha_2 v_2 + Const.$$

Usando lo visto anteriormente se tiene: $\alpha_1 + s\alpha_2 = \alpha_1^o + s\alpha_2^o = \gamma$, donde s toma el valor 1 en el caso $y_1 = y_2$, y -1 en caso contrario. Utilizando este valor, se busca la función objetivo en función de α_2.

$$\begin{aligned} W(\alpha_2) = \quad & \gamma - s\alpha_2 + \alpha_2 - \tfrac{1}{2}K_{11}(\gamma - s\alpha_2)^2 - \tfrac{1}{2}K_{22}\alpha_2^2 \\ & - sK_{12}(\gamma - s\alpha_2)\alpha_2 - y_1(\gamma - s\alpha_2)v_1 - y_2\alpha_2 v_2 + Const. \end{aligned}$$

Esta será la nueva función a optimizar, con respecto a α_2, por lo que usando la condición de optimalidad $\frac{d(W(\alpha_2))}{d\alpha_2} = 0$, se obtiene:

$$\begin{aligned} \tfrac{d(W(\alpha_2))}{d\alpha_2} = \quad & 1 - s + sK_{11}(\gamma - s\alpha_2) - K_{22}\alpha_2 \\ & + K_{12}\alpha_2 - sK_{12}(\gamma - s\alpha_2) + sy_1 v_1 - y_2 v_2 = 0. \end{aligned}$$

Como $s = 1$, si $y_1 = y_2$, y -1 en el caso contrario, se puede cambiar sy_1v_1 por y_2v_1, ya que si $y_1 = y_2$, entonces $s = 1$ y se puede cambiar y_1 por y_2. Por otro lado, si $y_1 \neq y_2$, entonces $s = -1$ y convierte y_1 en y_2 al cambiarle el signo.

La condición de optimalidad permite llegar a la siguiente igualdad:

$$\begin{aligned}\alpha_2^{n,u}(K_{11}+K_{22}-2K_{12}) &= 1-s+\gamma s(K_{11}-K_{12})+y_2(v_1-v_2)\\ &= y_2(y_2-y_1+\gamma y_1(K_{11}-K_{12})+v_1-v_2).\end{aligned}$$

Por lo tanto usando los valores de v_1 y v_2:

$$\begin{aligned}\alpha_2^{n,u}ky_2 &= y_2-y_1+f(x_1)-\textstyle\sum_{j=1}^{2}y_j\alpha_j K_{ij}+\gamma y_1K_{11}\\ &\quad -f(x_2)+\textstyle\sum_{j=1}^{2}y_j\alpha_j K_{ij}-\gamma y_1K_{12}\\ &= y_2-y_1+f(x_1)-f(x_2)+y_2\alpha_2K_{11}-y_2\alpha_2K_{12}+y_2\alpha_2K_{22}-y_2\alpha_2K_{12}\\ &= y_2\alpha_2k+(f(x_1)-y_1)-(f(x_2)-Y_2),\end{aligned}$$

obteniéndose:

$$\alpha_2^{n,u}=\alpha_2^0+\frac{y_2(E_1-E_2)}{k}.$$

Por último hay que acotar α_2^n, a partir de este valor, para que cumpla las condiciones.

3.1.2 Selección heurística de dos puntos

Es importante la elección de los dos puntos que se tratarán de optimizar. Para llevarlo a cabo, se busca elegir aquellos puntos que tienen una mayor influencia en el progreso general hacia la solución. Una forma de elegir puntos que tengan una gran influencia en este proceso es elegir aquellos que más incumplen las condiciones de Karish-Kunh-Tucker.

En este proceso hay que elegir dos puntos. Para elegir el primero, se busca a través de aquellos puntos que no cumplen las condiciones de KKT. Para buscarlos de una manera más sencilla, el algoritmo, en un principio, busca estos puntos a través de aquellos que cumplen la restricción $0<\alpha_i<C$. Es decir, a través de aquellos puntos que no se encuentran en el límite de la región factible. Solo cuando todos estos puntos satisfacen las condiciones KKT, se realiza el algoritmo a través de todo el espacio. Cuando encuentra un punto que incumple las condiciones, se selecciona

y pasa a buscar el segundo de la siguiente manera:
Como ya se tiene el primer punto, el segundo se busca de manera que la actualización de α_1 y α_2 produzca un mayor aumento de la función dual objetivo.
La manera de seleccionar el segundo punto, es buscar aquel que consigue maximizar $|E_1 - E_2|$.

Nótese, que todo el desarrollo del apartado anterior y de este, depende del valor de C. Este proceso puede realizarse de igual manera en el caso de que C fuese infinito. Esto relajaría las restricciones sobre α_i. La única diferencia con lo realizado anteriormente es que se alcanzan unos valores distintos de U, V y, como consecuencia, también cambian los valores de α_1^n, α_2^n calculados en el teorema del apartado anterior. Todo esto se obtiene de manera idéntica al caso anterior.

4 Métodos Multi-Clase

En ocasiones, no se pretende clasificar las observaciones en dos clases distintas, sino en más de dos clases.

Las máquinas de vector soporte de K-clases, consisten en, a través de una muestra de l observaciones, (x_i, y_i), donde x_i es un vector d-dimensional, e $y_i \in \{1, ..., k\}$, que marca a qué clase corresponde la observación i-ésima. Se debe obtener una función de decisión, que a cada valor x le asigne la clase que le corresponda.
Para llevar a cabo esta tarea, describiremos ciertos métodos eficaces para asignarle a cada observación su correspondiente clase.

La mayoría de métodos multi-clase, consisten en reducir un problema de clasificación multi-clase, en múltiples problemas de clasificación de dos clases.

En este apartado vamos a ver métodos que se obtienen de realizar esta división en problemas de dos clases, basándonos en [8]. Además, desarrollaremos algunos de los métodos que se basan en la resolución directa de un problema de optimización para varias clases. Concretamente, expondremos los recogidos en [3] y [9].

Empecemos por dos bastante conocidos, y que se obtienen directamente utilizando las máquinas de vector soporte descritas anteriormente.

4.1 Método uno contra uno (OVO)

Suponemos que se tienen K clases diferentes.
En este método, se construirá una máquina de vector soporte para cada combinación posible de las K clases. Por lo tanto habrá $\binom{K}{2}$ máquinas de vector soporte.

Una vez construidas las máquinas de vector soporte para cada par de clases, dada una nueva observación, esta es asignada de la siguiente manera:
Se ve la asignación que le corresponde para cada par de clases, dada por la máquina de vector soporte construida para estas dos clases. Una vez que se ha hecho con todas y cada una de las combinaciones posibles, se observa la frecuencia con la que es asignada a cada una de las clases, y se le asigna aquella clase que tenga una mayor frecuencia.

4.2 Método uno contra todos (OVA)

Igual que en el caso anterior, se supone que se tienen K clases diferentes.
Este método tiene una idea similar a la anterior, pero en este caso, en lugar de construir una máquina de vector soporte para cada par de clases, dada una variable i-ésima, se construirá la máquina de vector soporte que compara esta clase con todas las demás, asignando el valor $y = 1$ para la clase i-ésima y el valor $y = -1$ para el resto.

Tras construir todas las máquinas de vector soporte necesarias para comparar cada variable con el resto, dada una nueva observación, x^*, se evalúa $f_i(x^*)$, donde f_i hace referencia a la función test de la máquina de vector soporte que compara la clase i-ésima con el resto.
Una vez llevado a cabo el cálculo de $f_i(x^*)\ \forall i$, se asigna x^* a la clase para la cual ese valor es mayor que en el resto de clases.
Esto se debe a que un mayor valor de $f_i(x^*)$ indicaría una confianza mayor de que x^* pertenezca a la clase i-ésima antes que al resto de clases, al comparar esa máquina la clase i-ésima con el resto, asignándole el valor $y = 1$ a dicha clase.

Ambos métodos se basan en una combinación de diferentes, e independientes, máquinas de vector soporte, que están diseñadas para casos de 2 clases.
El siguiente objetivo consistirá en desarrollar algunos métodos para formular una máquina de vector soporte, que permita resolver problemas multi-clase, con un problema de optimización.

Primero consideremos el método de Weston-Watkins.

4.3 Método de Weston-Watkins

Una manera de llevar a cabo lo descrito anteriormente, es crear una separación de las K-clases mediante un problema de optimización.
Esto se consigue mediante una generalización del método ya conocido para 2 clases. Se obtiene el siguiente problema:

$$\begin{aligned} \min \Phi(w,\xi) \quad & = \tfrac{1}{2}\sum_{m=1}^{k}(w_m \cdot w_m) + C\sum_{i=1}^{l}\sum_{m\neq y_i}\xi_i^m \\ s.a \quad & (w_{y_i}\cdot x_i) + b_{y_i} \geq (w_m \cdot x_i) + b_m + 2 - \xi_i^m \\ & \xi_i^m \geq 0, \;\; i = 1,...,l, \;\; m \in \{1,...,k\}\setminus y_i. \end{aligned}$$

En este caso, la función de decisión posterior viene dada por:

$$f(x) = \underset{n}{\operatorname{argmax}}\,[(w_n \cdot x) + b_n], \;\; n = 1,...,k.$$

Esta manera de abordar la separación en K-clases, aporta una ventaja con respecto a otros métodos como el OVA. Dada una muestra de diferentes observaciones, mientras que este método permite buscar una regla de decisión que los clasifique perfectamente, por ejemplo en el método OVA, no es tan sencillo.

Igual que se hizo en el caso de dos clases, para obtener la solución del problema de optimización del caso multi-clase, se busca el punto silla del lagrangiano:

$$\begin{aligned} L(w,b,\xi,\alpha,\beta) \quad & = \tfrac{1}{2}\sum_{m=1}^{k}(w_m \cdot w_m) + C\sum_{i=1}^{l}\sum_{m=1}^{k}\xi_i^m \\ & -\sum_{i=1}^{l}\sum_{m=1}^{k}\alpha_i^m[((w_{y_i} - w_m)\cdot x_i) + b_{y_i} - b_m - 2 + \xi_i^m] - \sum_{i=1}^{l}\sum_{m=1}^{k}\beta_i^m\xi_i^m, \end{aligned}$$

con las variables ficticias $\alpha_i^{y_i} = 0,\ \xi_i^{y_i} = 2,\ \beta_i^{y_i} = 0\ \ i = 1,...,l$. Añadiéndole las restricciones: $\alpha_i^m \geq 0,\ \beta_i^m \geq 0,\ \xi_i^m \geq 0,\ i = 1,...,l;\ m \in \{1,...,k\} \setminus y_i$.

Este lagrangiano, se trata de maximizar con respecto a α y β, y minimizar con respecto a w y ξ.

Usando la notación $A_i = \sum_{m=1}^{k} \alpha_i^m$, $C_i^n = \begin{cases} 1 & si \quad y_i = n \\ 0 & si \quad y_i \neq n \end{cases}$ y realizando algunos cambios. Se llega al problema:

$$\begin{aligned} \text{máx}\, W(\alpha) \quad & = 2\sum_{i,m} \alpha_i^m + \sum_{i,j,m} [-\tfrac{1}{2} C_j^{y_i} A_i A_j + \alpha_i^m \alpha_j^{y_i} - \tfrac{1}{2} \alpha_i^m \alpha_j^m](x_i \cdot x_j). \\ s.a: \quad & \sum_{i=1}^{l} \alpha_i^n = \sum_{i=1}^{l} C_i^n A_i \ \ n = 1,...,k \\ & 0 \leq \alpha_i^m \leq C,\ \alpha_i^{y_i} = 0\ \ i = 1,...,l;\ m \in \{1...k\} \setminus y_i. \end{aligned}$$

Se pueden encontrar valores de b_n, para resolver el conjunto de ecuaciones simultáneas de las condiciones KKT, u obtenerlos como una variable dual cuando se utiliza un método de optimización de puntos interiores (notado por A.Smola y B.Schölkopf). Obteniéndose la función de decisión:

$$f(x) = \underset{n}{\text{argmax}}[\sum_{i=1}^{l} (c_i^n A_i - \alpha_i^n)(x_i \cdot x) + b_n].$$

Análogamente al caso de dos clases, se puede sustituir los productos escalares $x_i \cdot x$, por Kernels, $K(x_i, x)$.

También se puede generalizar al caso multiclase las máquinas de vector soporte lineales mediante el empleo de Kernels. La generalización consiste en tomar el siguiente problema:

$$\begin{aligned} \text{mín} \quad & \sum_{i=1}^{l} \alpha_i + C\sum_{i=1}^{l} \sum_{j \neq y_i} \xi_i^j \\ s.a: \quad & \sum_{m:y_m=y_i} \alpha_m K(x_i, x_m) + b_{y_i} \geq \sum_{n:y_n=y_j} \alpha_n K(x_i, x_n) + b_{y_j} + 2 - \xi_i^j \\ & \alpha_i \geq 0,\ \xi_i^j \geq 0,\ i = 1,...,l;\ j \in \{1,...,k\} \setminus y_i. \end{aligned}$$

Y la función de decisión es:

$$f(x) = \underset{n}{\text{argmax}}(\sum_{i:y_i=n} \alpha_i K(x, x_i) + b_n).$$

En este caso, cabe observar que el número de clases no influye en el número de coeficientes (que en este caso es l). Sin embargo, en otros métodos, el número de clases si influye, y el número de coeficientes se convierte en $l * k$, para el caso de k clases.

Es importante conocer, la esperanza de la probabilidad de cometer un error a la hora de clasificar en su correspondiente clase, un dato test. Esta esperanza, viene dada por el cociente entre la esperanza del número de vectores soporte, y el número de observaciones de la muestra de entrenamiento, menos 1.
El hecho de que esta esperanza dependa de los vectores soporte, no debe hacer pensar que no tiene importancia la presencia del resto de observaciones en la muestra. En el caso en el que una observación no estuviera en la muestra, ya no aseguraría que esta fuese clasificada correctamente por la máquina de vector soporte creada por el resto de observaciones.
El valor de la esperanza de la probabilidad de cometer un error, significa que estamos interesados, en el tamaño de la unión del conjunto de todos los vectores soporte que definen cada hiperplano en la clasificación, lo que equivale a la cantidad de cálculos de Kernel necesarios.

Otra cosa que cabe destacar es, como se intuye al ser este método más preciso (como ya se comentó anteriormente) a la hora de encontrar una función de decisión que clasifique correctamente a todas las observaciones de la muestra, que la solución del método OVA es una solución factible de este método, pero no necesariamente la óptima.

4.4 Método de Cramer-Singer

Al igual que el método anterior, este busca una solución a los problemas multiclase, mediante la resolución de un único problema de optimización, en lugar de separarlo en una combinación de problemas de dos clases.
Esto se lleva a cabo mediante una generalización del concepto de margen separador.

La dificultad del cálculo de un problema de optimización para varias clases, es que puede existir una gran complejidad de cálculo. Por ello, en este método, para resolver el problema de optimización, se aplica una división del problema en múltiples problemas de optimización más simples.
Para un método de K-clases, se contará con una función clasificadora del tipo :

$$H_M(\bar{x}) = \underset{r=1}{\overset{k}{\operatorname{argmax}}}\{\bar{M_r} \cdot \bar{x}\}.$$

Donde M es una matriz de dimensión kxn. $\bar{M_r}$ hace referencia a la r-ésima fila de la matriz y $\bar{M_r} \cdot \bar{x}$ se conoce como similitud de $\bar{x}$ con la clase r-ésima. Entonces la predicción elegida es la de mayor similitud.

En todo método de clasificación, hay que valorar el error cometido en la predicción. En este caso se denota: $\epsilon_S(M) = \frac{1}{m}\sum_{i=1}^{m}[[H_M(x_i) \neq y_i]]$.
Donde S es el conjunto de observaciones de la muestra, y $[[p]] = \begin{cases} 1 & si \quad p\ es\ cierto. \\ 0 & si \quad cc. \end{cases}$
Se busca una matriz M que minimice el error cometido. Por lo tanto, se tratará de transformar la minimización del error en un problema de optimización cuadrática.
Se toma $\delta_{ij} = \begin{cases} 1 & si \quad i = j \\ 0 & si \quad i \neq j \end{cases}$ y se cambia el error por:

$$\underset{r}{\text{máx}}\{\bar{M_r} \cdot \bar{x} + 1 - \delta_{y,r}\} - \bar{M_y} \cdot \bar{x}.$$

Ese error será igual a 0 cuando $\text{máx}_r\{\bar{M_r} \cdot \bar{x} + 1 - \delta_{y,r}\} = \bar{M_y} \cdot \bar{x}$. Esto ocurre en el caso de que $\bar{M_y} \cdot \bar{x} \geq \bar{M_r} \cdot \bar{x} + 1 \ \forall r, r \neq y$, es decir, la confianza o similitud de y supera al resto en al menos 1.

Una vez definido ese error, se tratará de minimizar su suma:

$$\epsilon_S(M) \leq \frac{1}{m}\sum_{i=1}^{m}[\underset{r}{\text{máx}}\{\bar{M_r} \cdot \bar{x_i} + 1 - \delta_{y_i,r}\} - \bar{M_{y_i}} \cdot \bar{x_i}].$$

Tras llegar a este punto, el concepto de que los datos sean separables en este modelo, consiste en que todos esos errores sean iguales a 0.

Al igual que en el caso de dos clases, se puede desarrollar el problema de optimización, tanto en los casos separables, como en los no separables. Así, en el caso de datos separables, se tiene el siguiente problema:

$$\begin{aligned} &\text{mín}_M \quad \tfrac{1}{2}\|M\|_2^2 \\ &s.a: \quad \bar{M}_{y_i}\cdot\bar{x}_i + \delta_{y_i,r} - \bar{M}_r\cdot\bar{x}_i \geq 1 \ \forall i,r. \end{aligned}$$

Donde $\|M\|_2^2 = \|(\bar{M}_1,...,\bar{M}_k)\|_2^2 = \sum_{i,j} M_{ij}^2$.
Mientras que en el caso no separable, se añaden las variables $\xi_i \geq 0$, para obtener el siguiente problema de optimización:

$$\begin{aligned} &\text{mín}_{M,\xi} \quad \tfrac{1}{2}\beta\|M\|_2^2 + \textstyle\sum_{i=1}^m \xi_i \\ &s.a: \quad \bar{M}_{y_i}\cdot\bar{x}_i + \delta_{y_i,r} - \bar{M}_r\cdot\bar{x}_i \geq 1-\xi_i \ \forall i,r. \end{aligned}$$

Pasamos al problema de optimización lagrangiano, con función lagrangiana:

$$L(M,\xi,\eta) = \frac{1}{2}\beta\sum_r\|\bar{M}_r\|_2^2 + \sum_{i=1}^m \xi_i + \sum_{i,r}\eta_{i,r}[\bar{M}_r\cdot\bar{x}_i - \bar{M}_{y_i}\cdot\bar{x}_i - \delta_{y_i,r} + 1 - \xi_i],$$

y $\eta_{i,r} \geq 0 \forall i,r$.
Este problema consistiría en minimizar sobre M y ξ y maximizar sobre η. Para el problema de minimizar, las condiciones de mínimo son:

$$\frac{d}{d\xi_i}L = 1 - \sum_r \eta_{i,r} = 0 \Rightarrow \sum_r \eta_{i,r} = 1. \tag{4.1}$$

$$\frac{d}{dM_r}L = \sum_i \eta_{i,r}\bar{x}_i - \sum_{i,y_i=r}(\sum_q \eta_{i,q})\bar{x}_i + \beta\bar{M}_r = \sum_i \eta_{i,r}\bar{x}_i - \sum_i \delta_{y_i,t}\bar{x}_i + \beta\bar{M}_r = 0 \tag{4.2}$$

De (4.2) se obtiene $\bar{M}_r = \beta^{-1}[\sum_i(\delta_{y_i,r} - \eta_{i,r})\bar{x}_i]$. Que se escribe como:
$\bar{M}_r = \beta^{-1}[\sum_{i:y_i=r}(1-\eta_{i,r})\bar{x}_i + \sum_{i:y_i\neq r}(-\eta_{i,r})\bar{x}_i]$.

Se dirá que $\bar{x}_i$ es un vector soporte si el coeficiente que multiplica a $\bar{x}_i$ es distinto de 0 para algún r. Es decir, $\eta_{i,y_i} < 1$ o, $\eta_{i,r} > 0$ para algún r distinto de y_i.

Introduciendo estos valores en la formulación lagrangiana, se obtiene un problema de máximo que solo depende de η. Esto es:

$$Q(\eta) = -\frac{1}{2}\beta^{-1}\sum_{i,j}(\bar{x}_i \cdot \bar{x}_j)[\sum_r(\delta_{y_i,r} - \eta_{i,r})(\delta_{y_j,r} - \eta_{j,r})] - \sum_{i,r}\eta_{i,r}\delta_{y_i,r}.$$

Usando $\bar{1}_{y_i}$ como el vector que tiene el valor 1 en la componente y_i y 0 en el resto, se llega al problema:

$$\begin{aligned} \text{máx}_\eta\, Q(\eta) = \quad & -\tfrac{1}{2}\beta^{-1}\textstyle\sum_{i,j}(\bar{x}_i \cdot \bar{x}_j)[(\bar{1}_{y_i} - \bar{\eta}_i)\cdot(\bar{1}_{y_j} - \bar{\eta}_j)] - \textstyle\sum_i \bar{\eta}_i \cdot \bar{1}_{y_i} \\ s.a: \quad & \bar{\eta}_i \geq 0, \bar{\eta}_i \cdot \bar{1} = 1\ \forall i. \end{aligned}$$

Para simplificar la notación se toma $\bar{\tau}_i = \bar{1}_{y_i} - \bar{\eta}_i$, notación con la que se obtiene:

$$\bar{M}_r = \beta^{-1}\sum_i \tau_{i,r}\bar{x}_i. \tag{4.3}$$

Además, sustituyendo los productos escalares por Kernels y omitiendo cualquier constante aditiva o multiplicativa, lo que se puede hacer ya que buscamos los valores de las variables que maximizan la función y no el valor de la función en sí, se llega finalmente al problema de optimización:

$$\begin{aligned} \text{máx}_\tau\, Q(\tau) = \quad & -\tfrac{1}{2}\textstyle\sum_{i,j} K(\bar{x}_i \cdot \bar{x}_j)(\bar{\tau}_i \cdot \bar{\tau}_j) + \beta\textstyle\sum_i \bar{\tau}_i \cdot \bar{1}_{y_i} \\ s.a: \quad & \bar{\tau}_i \leq \bar{1}_{y_i}, \bar{\tau}_i \cdot \bar{1} = 0\ \forall i. \end{aligned}$$

La función de decisión toma el valor:

$$H(\bar{x}) = \underset{r=1}{\overset{k}{\text{argmax}}}\{\sum_i \tau_{i,r}K(\bar{x}, \bar{x}_i)\}.$$

En el caso en el que quisiésemos resolver este problema dual de programación cuadrática, al tener $m*k$ variables, una técnica normal de programación cuadrática, emplea una matriz de dimensión $(mk)x(mk)$. Esto para problemas de dimensiones elevadas, es muy difícil de manejar.
Debido a todo esto, se tratará de descomponer el problema, en problemas más pequeños.
La idea es tomar las restricciones del problema original, y dividirlas en m conjuntos

de la forma: $\{\bar{\tau}_i, \bar{\tau}_i \leq \bar{1}_{y_i}, \bar{\tau}_i \cdot \bar{1} = 0\}_{i=1}^{m}$.
El algoritmo usado para resolver el problema de optimización se basa en tomar p, y mejorar el valor de la función objetivo, actualizando el valor de $\bar{\tau}_p$ bajo las restricciones correspondientes a ese valor.

Veamos el problema de optimización a realizar en cada elección de p, aislando la contribución de p al valor de la función objetivo.

$$\begin{aligned} Q_p(\bar{\tau}_p) &= -\tfrac{1}{2}\textstyle\sum_{i,j} K_{ij}(\bar{\tau}_i \cdot \bar{\tau}_j) + \beta \sum_i \bar{\tau}_i \cdot \bar{1}_{y_i} \\ &= -\tfrac{1}{2} K_{p,p}(\bar{\tau}_p \cdot \bar{\tau}_p) - \textstyle\sum_{i \neq p} K_{i,p}(\bar{\tau}_p \cdot \bar{\tau}_i) \\ &\quad -\tfrac{1}{2}\textstyle\sum_{i \neq p, j \neq p} K_{i,j}(\bar{\tau}_i \cdot \bar{\tau}_j) + \beta \bar{\tau}_p \cdot \bar{1}_{y_p} + \beta \textstyle\sum_{i \neq p} \bar{\tau}_i \bar{1}_{y_i} \\ &= -\tfrac{1}{2} K_{p,p}(\bar{\tau}_p \cdot \bar{\tau}_p) - \bar{\tau}_p \cdot [-\beta \bar{1}_{y_p} + \textstyle\sum_{i \neq p} K_{i,p} \bar{\tau}_i] \\ &\quad + [-\tfrac{1}{2}\textstyle\sum_{i \neq p, j \neq p} K_{i,j}(\bar{\tau}_i \cdot \bar{\tau}_j) + \beta \sum_{i \neq p} \bar{\tau}_i \bar{1}_{y_i}]. \end{aligned}$$

Tomando los siguientes valores:

$$A_p = K_{p,p} > 0 \tag{4.4}$$

$$\bar{B}_p = -\beta \bar{1}_{y_p} + \sum_{i \neq p} K_{i,p} \bar{\tau}_i \tag{4.5}$$

$$C_p = -\frac{1}{2} \sum_{i,j \neq p} K_{i,j}(\bar{\tau}_i \cdot \bar{\tau}_j) + \beta \sum_{i \neq p} \bar{\tau}_i \cdot \bar{1}_{y_i} \tag{4.6}$$

Como C_p no influye en la solución de esta etapa al no depender de p, se puede omitir. Se llega al problema de optimización :

$$\begin{aligned} \text{mín}_{\bar{\tau}_p}\, Q(\bar{\tau}_p) = \quad & \tfrac{1}{2} A_p(\bar{\tau}_p \cdot \bar{\tau}_p) + \bar{B}_p \cdot \bar{\tau}_p \\ s.a: \quad & \bar{\tau}_p \leq \bar{1}_{y_p}, \bar{\tau}_p \cdot \bar{1} = 0. \end{aligned}$$

En la aplicación de esta estrategia, los tres problemas a resolver son: cómo seleccionar el criterio de parada, cómo seleccionar el valor p en cada etapa, y cómo resolver el problema reducido.

Comencemos buscando el criterio de parada.
El problema que se quiere resolver es:

$$\begin{aligned} \text{mín}_\tau\, Q(\tau) = \quad & \tfrac{1}{2}\sum_{i,j} K(\bar{x}_i \cdot \bar{x}_j)(\bar{\tau}_i \cdot \bar{\tau}_j) - \beta \sum_i \bar{\tau}_i \cdot \bar{1}_{y_i} \\ s.a: \quad & \bar{\tau}_i \leq \bar{1}_{y_i}, \bar{\tau}_i \cdot \bar{1} = 0 \ \forall i. \end{aligned}$$

Se utilizan las condiciones de KKT para determinar las condiciones de optimalidad que debe cumplir una solución $\bar{\tau}$. Pasando al lagrangiano se obtiene:

$$\begin{aligned} L(\tau, u, v) \quad & = \tfrac{1}{2}\sum_{i,j} K_{i,j} \sum_r (\tau_{i,r}\tau_j) - \beta \sum_{i,r} \tau_{i,r}\delta_{y_i,r} \\ & \quad + \sum_{i,r} u_{i,r}(\tau_{i,r} - \delta_{y_i,r}) - \sum_i v_i \sum_r \tau_{i,r} \\ s.a: \quad & u_{i,r} \geq 0 \ \forall i, r. \end{aligned}$$

Las condiciones de optimalidad dan la condición:

$$\frac{d}{d\tau_{i,r}} L = \sum_j K_{i,j}\tau_{j,r} - \beta\delta_{y_i,r} + u_{i,r} - v_i = 0. \tag{4.7}$$

Se define:

$$F_{i,r} = \sum_j K_{i,j}\tau_{j,r} - \beta\delta_{y_i,r}. \tag{4.8}$$

Este valor marca la confianza en asignar la clase r a $\bar{x}_i$.

Nótese que:

$$F_{p,r} = B_{p,r} + k_{p,p}\tau_{p,r}, \tag{4.9}$$

lo que será necesario más adelante.

Derivando $L(\tau, u, v)$ con respecto a las variables duales, u y v, y utilizando (4.8), se llega a:

$$F_{i,r} + u_{i,r} = v_i \ \forall i, r. \tag{4.10}$$

$$u_{i,r}(\tau_{i,r} - \delta_{y_i,r}) \tag{4.11}$$

$$u_{i,r} \geq 0 \tag{4.12}$$

A la hora de interpretar estas restricciones, nos encontramos con dos casos claramente diferenciados:
El primero: $\tau_{i,r} = \delta_{y_i,r}$. Entonces como $u_{i,r} \geq 0$, se tiene por (4.10)

$$F_{i,r} \leq v_i \tag{4.13}$$

Por otro lado, si $\tau_{i,r} < \delta_{y_i,r}$. Entonces para que se cumpla (4.11), se tiene $u_{i,r} = 0$, por lo que, por (4.10), $F_{i,r} = v_i \ \forall i, r$. Esto último equivale a las dos restricciones:

$$F_{i,r} \leq v_i \tag{4.14}$$

$$F_{i,r} \geq v_i \tag{4.15}$$

Utilizando las restricciones del problema original, $\bar{\tau}_i \leq \bar{1}_{y_i}$ y $\bar{\tau}_i = 0 \ \forall i$, obtenemos: $\exists r : \tau_{i,r} < \delta_{y_i,r}$.
Se tiene $v_i = \text{máx}_r F_{i,r}$. Combinando (4.13), (4.14) y (4.15).

$$max_r F_{i,r} \leq v_i \leq \min_{r:\tau_{i,r}<\delta_{y_i,r}} F_{i,r} \tag{4.16}$$

Por lo que se obtiene:

$$\max_r F_{i,r} \leq \min_{r:\tau_{i,r}<\delta_{y_i,r}} F_{i,r} \tag{4.17}$$

Si se define $\Psi_i = \text{máx}_r F_{i,r} - \text{mín}_{r:\tau_{i,r}<\delta_{y_i,r}} F_{i,r}$, entonces la condición de optimalidad necesaria y suficiente es $\Psi_i = 0$. El criterio de parada buscado será: dado $\epsilon > 0$, $\Psi_i < \epsilon$.

Una vez decidido el criterio de parada del algoritmo, queda por determinar el valor inicial y cómo resolver el problema de optimización reducido.
Comenzamos reescribiendo $Q(\tau)$ usando una expresión como forma cuadrática y sin el índice p

$$\begin{aligned} Q(\tau) &= -\tfrac{1}{2} A(\bar{\tau} \cdot \bar{\tau}) - \bar{B} \cdot \bar{\tau} \\ &= -\tfrac{1}{2} A[(\bar{\tau} + \tfrac{\bar{B}}{A}) \cdot (\bar{\tau} + \tfrac{\bar{B}}{A})] + \tfrac{\bar{B} \cdot \bar{B}}{2A}. \end{aligned}$$

Es necesario introducir ahora las siguientes variables:

$$\bar{v} = \bar{\tau} + \frac{\bar{B}}{A} \quad \bar{D} = \frac{\bar{B}}{A} + \bar{1}_y. \tag{4.18}$$

Al igual que ocurrió antes, no es necesario tener en cuenta las constantes aditivas o multiplicativas ya que no influyen sobre las variables en el óptimo. Así, se llega al problema:

$$\begin{aligned} \text{mín}_{\nu} \quad & = Q(\bar{\nu}) = \|\bar{\nu}\|^2 \\ s.a: \quad & \bar{\nu} \leq \bar{D}, \bar{\nu} \cdot \bar{1} = \bar{D} \cdot \bar{1} - 1. \end{aligned}$$

Denotando las variables duales del problema como θ y α_i, se obtiene por las condiciones de KKT las siguientes restricciones:

$$\nu_r \leq D_r; \alpha_r(\nu_r - D_r) = 0; \nu_r + \alpha_r - \theta = 0 \; \forall r.$$

Como $\alpha_r \geq 0$, se tiene $\nu_r \leq \theta \; \forall r$. Uniéndolo a la primera restricción, se tiene:

$$\nu_r \leq \text{mín}\{\theta, D_r\}. \tag{4.19}$$

En realidad, se espera la igualdad ya que en el caso en el que $\alpha_r = 0$, $\nu_r = \theta$, y en el caso en el que $\alpha_r > 0$, se tiene $\nu_r = D_r$.
Usando $\bar{\nu} \cdot \bar{1} = \bar{D} \cdot \bar{1} - 1$, se llega a:

$$\sum_{r=1}^{k} \min_r\{\theta, D_r\} = \sum_{r=1}^{k} D_r - 1. \tag{4.20}$$

La suma $\sum_{r=1}^{k} \text{mín}_r\{\theta, D_r\}$ es una función estrictamente monótona y continua en θ. Por ello existe un único θ^* que satisface (4.20), veamos que corresponde a la solución óptima.

Teorema 4.1. *Sea* $\nu_r^* = \text{mín}\{\theta^*, D_r\}$ *donde* θ^* *es solución de* $\sum_{r=1}^{k} \text{mín}_r\{\theta, D_r\} = \sum_{r=1}^{k} D_r - 1$. *Entonces,* $\forall \bar{\nu}$, *se tiene:* $\|\bar{\nu}\|^2 > \|\bar{\nu}^*\|$.

Demostración. Se llevará acabo mediante reducción al absurdo.
Supongamos que existe otro punto factible $\bar{\nu} = \bar{\nu}^* + \bar{\Delta}$ que minimiza la función objetivo y $\bar{\nu} \neq \bar{\nu}^*$, por lo que $\bar{\Delta} \neq 0$. Como tanto $\bar{\nu}$ y $\bar{\nu}^*$ deben cumplir las restricciones del problema original, $\sum_r \Delta_r = 0$, por el mismo motivo, también se tiene: $\Delta_r \leq 0$ cuando $\nu_r^* = D_r$. Combinando ambas con la restricción $\bar{\Delta} \neq 0$, se tiene $\Delta_s > 0$ para algún s con $\nu_s^* = \theta$. Usando de nuevo $\sum_r \Delta_r = 0$, se tiene que $\exists u$ tal que, $\Delta_u < 0$.
Sea $\epsilon = \text{mín}\{|\Delta_s|, |\Delta_u|\}$. Se define otro punto factible $\bar{\nu}'$ como: $\nu_s' = \nu_s - \epsilon, \nu_u' = \nu_u + \epsilon$ y $\nu_r' = \nu_r$ en otro caso.
Como $\bar{\nu}$ y $\bar{\nu}'$ solo difieren en las coordenadas s y u, se tiene:

$$\|\bar{\nu}'\|^2 - \|\bar{\nu}\|^2 = (\nu_s')^2 + (\nu_u')^2 - (\nu_s)^2 - (\nu_u)^2.$$

Al escribir $\bar{v}'$ en función de $\bar{v}$ y ϵ, se tiene :

$$\|\bar{v}'\|^2 - \|\bar{v}\|^2 = 2\epsilon(\epsilon - v_s + v_u).$$

Por la construcción de $\bar{v}'$, se tiene $v_u - \epsilon = \theta > v_s$ o $v_u - \epsilon > \theta \geq v_s$, y por lo tanto, $v_u - \epsilon > v_s$. Esto implica:

$$\|\bar{v}'\|^2 - \|\bar{v}\|^2 < 0.$$

lo que es una contradicción.

Una vez probada esta propiedad, se busca un algoritmo de punto fijo para aproximar el valor de θ^*.
Se utiliza que $\min\{\theta, D_r\} + \max\{\theta, D_r\} = \theta + D_r$, que con (4.20) implica que :

$$\sum_{r=1}^{k} [\theta + D_r - \max\{\theta, D_r\}] = \sum_{r=1}^{k} D_r - 1,$$

lo que nos lleva a

$$F(\theta^*) = \theta^* = \frac{1}{k}[\sum_{r=1}^{k} \max\{\theta^*, D_r\}] - \frac{1}{k}. \tag{4.21}$$

El algoritmo consistirá en, una vez elegido un punto θ, obtener el siguiente valor sustituyendo θ por $F(\theta)$. El valor óptimo cumple $F(\theta^*) = \theta^*$.
El algoritmo termina cuando dos valores consecutivos de θ se encuentran lo suficientemente cerca.

En cuanto al valor inicial, se demuestra que si $\theta_1 \leq \max_r D_r$, el algoritmo converge hacia θ^*

Teorema 4.2. *Sea θ^* el punto fijo de la ecuación $F(\theta)$. Si $\theta_1 \leq \max_r D_r$ y sea $\theta_{l+1} = F(\theta_l)$. Para $l \geq 1$ se tiene:*

$$\frac{\|\theta_{l+1} - \theta^*\|}{\|\theta_l - \theta^*\|} \leq 1 - \frac{1}{k},$$

donde k es el número de clases.

Demostración. Asumimos sin pérdida de generalidad: $\max_r D_r = D_1 \geq D_2 \geq ... \geq D_k \geq D_{k+1} = -\infty$. Asumimos $\theta^* \in (D_{s+1}, D_s)$ y $\theta_l \in (D_{u+1}, D_u)$ donde $u, s \in \{1, ..., k\}$. Se tiene:

$$\begin{aligned} \theta_{l+1} &= F(\theta_l) = \tfrac{1}{k}[\textstyle\sum_{r=1}^{k} \max\{\theta_l, D_r\}] - \tfrac{1}{k} \\ &= \tfrac{1}{k}(\textstyle\sum_{r=u+1}^{k} \theta_l) + \tfrac{1}{k}(\textstyle\sum_{r=1}^{u} D_r) - \tfrac{1}{k} = (1 - \tfrac{u}{k})\theta_l + \tfrac{1}{k}(\textstyle\sum_{r=1}^{u} D_r - 1). \end{aligned}$$

Si $\theta_l \leq \text{máx}_r\, D_r \Rightarrow \theta_{l+1} \leq \text{máx}_r\, D_r$. Análogamente, se obtiene:

$$\theta^* = F(\theta^*) = (1 - \frac{s}{k})\theta^* + \frac{1}{k}(\sum_{r=1}^{s} D_r - 1) \Rightarrow \theta^* = \frac{1}{s}(\sum_{r=1}^{s} D_r - 1).$$

Consideramos 3 casos:

Primer caso, $u = s$. En ese caso :

$$\begin{aligned}
\frac{|\theta_{l+1} - \theta^*|}{|\theta_l - \theta^*|} &= \frac{|(1-\frac{s}{k})\theta_l + \frac{1}{k}(\sum_{r=1}^{s} D_r - 1) - \theta^*|}{|\theta_l - \theta^*|} \\
&= \frac{|(1-\frac{s}{k})\theta_l + \frac{s}{k}\theta^* - \theta^*|}{|\theta_l - \theta^*|} \\
&= 1 - \frac{s}{k} \leq 1 - \frac{1}{k}.
\end{aligned}$$

Segundo caso $u > s$, entonces $\forall r = s+1, ..., u$ se tiene:

$$\theta_l \leq D_r \leq \theta^*. \tag{4.22}$$

Usando los valores para θ_{l+1} y θ^* anteriores, se llega a

$$\begin{aligned}
\theta_{l+1} &= (1 - \frac{u}{k})\theta_l + \frac{1}{k}(\sum_{r=1}^{u} D_r - 1) \\
&= (1 - \frac{u}{k})\theta_l + \frac{s}{k}\frac{1}{s}(\sum_{r=1}^{s} D_r - 1) + \frac{1}{k}(\sum_{r=s+1}^{u} D_r) \\
&= (1 - \frac{u}{k})\theta_l + \frac{s}{k}\theta^* + \frac{1}{k}(\sum_{r=s+1}^{u} D_r).
\end{aligned}$$

Aplicando (4.22), se obtiene :

$$\begin{aligned}
\theta_{l+1} &\leq (1 - \frac{u}{k})\theta_l + \frac{s}{k}\theta^* + \frac{1}{k}(u - s)\theta^* \\
&= (1 - \frac{u}{k})\theta_l + \frac{u}{k}\theta^*.
\end{aligned}$$

Entonces se tiene:

$$\begin{aligned}
\frac{|\theta_{l+1} - \theta^*|}{|\theta_l - \theta^*|} &= \frac{\theta^* - \theta_{l+1}}{\theta^* - \theta_l} \\
&\leq \frac{\theta^* - (1-\frac{u}{k})\theta_l - \frac{u}{k}\theta^*}{\theta^* - \theta_l} \\
&= 1 - \frac{u}{k} \leq 1 - \frac{1}{k}.
\end{aligned}$$

Caso 3 $u < s$ es análogo al anterior intercambiando los roles de u y s.

4.5 Conclusiones

En este apartado, se han desarrollado diferentes métodos multi-clase.
Los dos primeros métodos, se basan en realizar varias máquinas de vector soporte de 2 clases y combinarlas para clasificar nuevos datos.

El método OVO, crea una máquina de vector soporte para cada combinación de clases, lo que, para el caso de K-clases, precisa de $\binom{K}{2}$ máquinas de vector soporte.
El método OVA, genera una máquina por cada clase, enfrentando a esta clase con todas las demás. Esto reduce el número de máquinas de vector soporte necesarias.

El hecho de utilizar una combinación de varias máquinas de vector soporte, puede ser muy costoso en el caso en el que la muestra sea muy amplia, ya que el número de máquinas de vector soporte necesarias puede ser muy grande.

Los otros dos métodos, están planteados para resolver el problema mediante una única máquina de vector soporte.
Las principales ventajas que conlleva, son la mejora de la interpretación del método, y el hecho de que es más directa la clasificación de un nuevo elemento. Esto se debe a que a la hora de clasificarlo, solo se debe evaluar una máquina de vector soporte.
Sin embargo, presenta el siguiente inconveniente.
Un problema de optimización de varias clases, puede provocar cálculos muy costosos. Para solucionar estas dificultades, el método C-S, plantea una división del problema de optimización, en múltiples problemas de optimización más simples, esto rebaja la complejidad de cálculo pero lleva a tener que resolver varios problemas distintos, lo que aumenta la cantidad de cálculos.

5 Experiencia computacional

En este capítulo, mostraremos la implementación en Python de algunas SVMs, basadas en algunos de los conjuntos de datos más usados en el ámbito de máquinas de aprendizaje. Dichos datos serán extraídos de [5].
El objetivo consiste en, para cada conjunto de datos, observar la precisión de la SVM construida en función del Kernel utilizado. Para ello, se dividirá el conjunto de datos en dos subconjuntos, uno destinado al entrenamiento de la SVM y otro destinado a la fase test. Una vez construida la SVM, se realizará una predicción de en qué clase debería estar cada observación del subconjunto destinado a la fase test. Por último, observaremos la precisión de la SVM comparando la predicción con los datos reales. Para ello nos fijaremos en el ejemplo que aparece en [6].

5.1 Conjunto de datos IRIS

En primer lugar, trabajamos con el conjunto de datos IRIS. Este conjunto recoge 4 variables: sepal-length, sepal-width, petal.length y petal.width. Además, también recoge el tipo de planta: Iris-setosa, Iris-versicolor, Iris-virginica.
El conjunto de datos, recoge las 4 variables y el tipo de planta de 150 observaciones. El objetivo será construir una SVM capaz de predecir el tipo de planta de una nueva observación a partir de las 4 variables.

En primer lugar, recogeremos una matriz X que recoge el valor de cada variable de las 150 observaciones, y un vector Y con el tipo de planta a la que corresponde cada observación.
Se dividirán tanto la matriz X como el vector Y en entrenamiento y test aplicando la siguiente orden:

```
X_train, X_test, Y_train, Y_test =
train_test_split(X, Y, test_size = 0.20).
```

Lo cual nos aporta el 80 % de las observaciones para la fase de entrenamiento, y, el resto, para la fase test.

Con estos datos, ya podemos construir la SVM, en función del Kernel que deseemos utilizar, entrenándola con los datos de entrenamiento. Por ejemplo, en primer lugar, utilizaremos un Kernel polinómico. Esto se realiza mediante:

```
svclassifier = SVC(kernel='poly')
svclassifier.fit(X_train, Y_train).
```

Una vez entrenada la SVM, podemos usarla para la predicción de los datos test y compararlo con la clase real de dichas observaciones. Se realiza mediante el siguiente proceso:

```
Y_pred = svclassifier.predict(X_test)
print(confusion_matrix(Y_test,Y_pred))
print(classification_report(Y_test,Y_pred))
```

El primer "print", aporta una matriz que muestra en su diagonal, el numero de observaciones en las que coinciden su predicción con su valor real.
El segundo "print" aporta una tabla con los valores de la precisión en cada clase y el término "accuracy", que se refiere a la precisión total de la SVM.

```
                precision  recall  f1-score  support
    Iris-setosa       1.00    1.00      1.00       12
Iris-versicolor       1.00    1.00      1.00       10
 Iris-virginica       1.00    1.00      1.00        8

       acuraccy                         1.00       30
      macro avg       1.00    1.00      1.00       30
   weighted avg       1.00    1.00      1.00       30
```

En este caso es de 1.00, por lo que esta SVM en particular tiene un buen funcionamiento.

El término "precision" hace referencia, a la probabilidad de que una observación que ha sido clasificada en esa clase, pertenezca realmente a dicha clase. Para calcularla, se toma el número de observaciones que han sido bien clasificadas en esa clase, y se divide entre el número total de observaciones que han sido clasificadas en esa clase.

El término "recall", hace referencia a la exhaustividad, es decir, a la probabilidad que de si una observación pertenece realmente a esa clase, sea clasificada en esa clase. Para calcular este término, se divide el número de observaciones que han sido bien clasificadas en esta clase, entre el número real de observaciones que pertenecen a esa clase.

El término "f1-score" aporta una relación entre "precision" y "recall" con la fórmula:

$$f_1 = 2 * \frac{precision * recall}{precision + recall}.$$

El valorar este término, asume que le damos una relevancia similar a ambos términos.

El término "support" hace referencia al número de observaciones que clasifica en cada clase, es decir, en este caso, el valor "support" de la clase "Iris-setosa", hace referencia al número de observaciones que la SVM clasifica como "Iris-setosa", independientemente de si la clasifición es acertada.

Otro Kernel que utilizaremos será un Kernel Gaussiano. Para ello, se realiza el mismo procedimiento anterior, con la salvedad de que a la hora de construir la SVM, se sustituirá "kernel='poly'" por "kernel='rbf'".

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	12
Iris-versicolor	1.00	0.90	0.95	10
Iris-virginica	0.89	1.00	0.94	8
acuraccy			0.97	30
macro avg	0.96	0.97	0.96	30
weighted avg	0.97	0.97	0.97	30

En este caso, al realizar los "print", se obtiene que la precisión total, dada por el término "accuracy", es 0.97. Por lo que esta SVM también funciona correctamente.

Sin embargo, usando el Kernel Sigmoid, al igual que el caso anterior basta con sustituir "kernel='sigmoid'", se obtiene una precisión de 0 en las primeras dos clases, y de 0.27 en la tercera, obteniendose una precisión total de 0.27. Por ello, la SVM no funciona correctamente con este tipo de Kernel.

	precision	recall	f1-score	support
Iris-setosa	0.00	0.00	0.00	12
Iris-versicolor	0.00	0.00	0.00	10
Iris-virginica	0.27	1.00	0.42	8
acuraccy			0.27	30
macro avg	0.09	0.33	0.14	30
weighted avg	0.07	0.27	0.11	30

5.2 Conjunto de datos ADULT

Este conjunto de datos, recoge 48842 observaciones, correspondientes a personas de las cuáles se muestran 14 variables: edad, tipo de trabajo (Privado, Fedgov,...), fnwlgt (Variable continua), educación (Nivel de educación), educación (Variable continua), estado marital, ocupación profesional, relación sentimental, raza, sexo, ganancias de capital, perdidas de capital, horas de trabajo a la semana y país de nacimiento. Y para cada observación, también se recoge si esa persona gana al año más de $50K$, que es lo que se tratará de predecir con la SVM.

En este caso, lo primero es convertir todas las variables en numéricas, lo que es necesario, ya que el método que estamos utilizando para construir la SVM, solo trabaja con variables numéricas. Esto se lleva a cabo asignando un número a cada valor de la variable, por ejemplo, en la variable sexo, cambiamos el valor hombre, por 0, y el valor mujer por 1.

El procedimiento es análogo al realizado sobre el conjunto de datos anterior, primero se toma una matriz que recoja las variables de las observaciones, y un vector con la clase correspondiente a cada observación. Se realiza la división en subconjuntos de entrenamiento y de test, tomando el 80 % de las observaciones para el subconjunto de entrenamiento de manera aleatoria. Por último entrenamos la SVM y comprobamos su eficacia, prediciendo la clase a la que pertenecerán las observaciones del subconjunto test y comparándola con las clases reales a las que pertenecen.

Todo esto se realizará de manera análoga al caso anterior, para los 3 mismos Kernels.
En este caso se obtiene una mayor precisión en los Kernels: Gaussiano y Sigmoid, en los que se obtiene un rendimiento de 0.76. Observemos, por ejemplo, la tabla de precisiones obtenida para el Kernel Gaussiano.

	precision	recall	f1-score	support
≤ 50	0.76	1.00	0.86	4928
> 50	0.40	0.00	0.00	1585
acuraccy			0.76	6513
macro avg	0.58	0.50	0.43	6513
weighted avg	0.67	0.76	0.65	6513

5.3 Conjunto de datos WINE

En esta sección, trabajaremos sobre el conjunto de datos WINE. En este conjunto de datos se recogen 178 observaciones. Cada una de ellas recoge 13 variables correspondientes al estudio de las características de un determinado vino. Cada uno de los vinos, pertenece a la misma región de Italia, pero de 3 proveedores diferentes. El objetivo será construir una SVM capaz de predecir a partir de dichas variables a cual de los proveedores pertenece un vino dado.

Las variables a estudio son las siguientes: nivel de alcohol del vino, nivel de ácido málico, nivel de ceniza, alcalinidad de cenizas, nivel de magnesio, fenoles totales, flavanoides, fenoles no flavanoides, proantocianidinas, intensidad de color, matiz, OD280/OD315 de vinos diluidos y prolina.

Se realiza un estudio idéntico al de los datos IRIS. Se toma la matriz X con las variables de las observaciones y el vector y, en el que aparece a qué proveedor corresponde cada vino. Se dividen aleatoriamente la matriz X y el vector y, seleccionando un 80 % de las observaciones para la fase de entrenamiento y el resto para la fase test.
Se construye la SVM a partir de los datos de entrenamiento y se comprueba su rendimiento comparando la predicción del conjunto test con su valor real.

En este caso, utilizando los mismos 3 Kernels, el mejor resultado se obtiene para el Kernel polinómico, obteniéndose un rendimiento de 1.00. Mientras que para el Kernel Gaussiano, se obtiene un rendimiento de 0.44, y para el Sigmoid de 0.39. Veamos la tabla obtenida para el caso del Kernel polinómico, el que mayor rendimiento tiene:

	precision	recall	f1-score	support
1	1.00	1.00	1.00	12
2	1.00	1.00	1.00	14
3	1.00	1.00	1.00	10
acuraccy			1.00	36
macro avg	1.00	1.00	1.00	36
weighted avg	1.00	1.00	1.00	36

6 Conclusión

En este trabajo, hemos desarrollado el concepto de SVM para dos y más clases. Esto nos ha proporcionado un algoritmo para predecir la clase a la que pertenece una observación dada. Para ello hemos visto la manera de entrenar la SVM a partir de un conjunto de datos de entrenamiento.

Hemos comenzado desarrollando el caso lineal con datos separables. Para ello hemos visto conceptos de programación lineal, ya que el objetivo consistía en maximizar el margen, que es una función lineal, bajo ciertas restricciones. Además, para simplificar el problema de optimización, trabajaremos con el problema dual, concepto también relacionado con la programación lineal.
Tras esto, hemos ampliado el problema al caso no separable. Para ello se introducen variables de error en las restricciones, para que algunas variables puedan encontrarse en el lado incorrecto del margen. Para que el problema no abuse de estos errores, se añade una penalización, por el uso de estos, en la función objetivo dual, ya que, en ese caso, se tratarán de minimizar dichos errores. Este concepto es utilizado en numerosas técnicas de investigación operativa.

Para terminar con las SVMs de dos clases, hemos realizado la ampliación del problema anterior al caso no lineal. En este caso, el problema de optimización a resolver es un problema de programación no lineal. Para simplificar la complejidad del problema, introducimos el concepto de Kernel.

A continuación, se desarrollan algunos métodos de solución, para llegar al método más utilizado, el SMO, un método iterativo para buscar la mejor solución del problema de optimización

Más tarde, hemos resuelto el problema de SVMs de más de dos clases. Para ello hemos construido diferentes métodos, algunos que utilizan varias SVMs de dos clases,

y otros destinados a buscar la resolución de un único problema de optimización, es decir, construir una SVM multi-clase.

Por último, se ha realizado la implementación en Python de las SVMs sobre diversos conjuntos de datos, muy utilizados en el ámbito de máquinas de aprendizaje. Lo cual aporta una idea más clara de la utilidad de las SVM en el ámbito de máquinas de aprendizaje, observando su precisión a la hora de predecir determinados resultados.

En definitiva, este trabajo me ha enseñado la manera de tratar problemas de clasificación en dos y más clases, mediante el planteamiento de diversos problemas de optimización vistos a lo largo de mis estudios en el grado. Además he aprendido la implementación en Python, y la forma de trabajar con máquinas de aprendizaje, separando las observaciones en dos subconjuntos, entrenamiento y test, para construir el algoritmo y comprobar su eficacia o rendimiento.

Bibliografía

[1] Christopher J.C. Burges. "A Tutorial on Support Vector Machines for Pattern Recognition". En: *Data Mining and Knowledge Discovery 2, 121-167* (1998).

[2] Bazaara M.S;Sherall Hanif D.;Shetty C.M. *Nonlinear Programming. Theory and Algorithms*. Wiley Sons John Wiley, 1993.

[3] J.Weston; C.Watkins. "Support Vector Machines for Multi-Class Pattern Recognition". En: *European Symposium on Artificial Neural Networks* 99 (abr. de 1999), págs. 219-224.

[4] Bazaara M.; Jarwis J. *Linear Programming and Network*. Wiley, 1977.

[5] *Machine Learning Repository*. URL: `https://archive.ics.uci.edu/ml/index.php`.

[6] Usman Malik. *Implementing SVM and Kernel SVM with Python's Scikit-Learn*. URL: `https://stackabuse.com/implementing-svm-and-kernel-svm-with-pythons-scikit-learn/`.

[7] John Shawe-Taylor Nello Cristianini. "Implementation Techniques". En: *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2013. Cap. 7, págs. 125-144.

[8] James G.; Witter D.; Hastie T.; Tibshirani R. "Support Vector Machines". En: *An Introduction to Statistical Learning*. Springer, 2013. Cap. 9, págs. 337-356.

[9] Crammer K.; Singer Y. "On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines". En: *Journal of Machine Learning Research 2* (ene. de 2001), págs. 265-292.