

Support Vector Machines (SVMs)

Paula Saavedra Nieves.
Laura Martínez Fernández.
Jacobo José Pardo Seco.

Índice.

I. Introducción	3
II. Problema de clasificación	4
III. Problema de clasificación SVMs	4 - 12
IV. Métodos Kernel & Kernel trick	13 - 15
V. Métodos de clasificación alternativos	16 - 20
A. El problema de la discriminación	
B. Aproximación clásica: el método lineal de Fisher	
C. Método k-vecinos más próximos	
VI. Aplicaciones prácticas	21 - 26
A. Diagnóstico cáncer de mama	
B. Federalist Papers	
VII. Regresión (SVR)	27 - 28
Apéndice	29 - 44
Bibliografía	45

I. Introducción.

En los últimos años ha habido un interés renovado en abordar problemas de clasificación difíciles, originados en el "mundo real" y que pueden agruparse bajo el epígrafe de "muestras extremas". Esto se debe principalmente por la presencia de desequilibrios en las muestras, al haber clases cuyo número de datos es mucho menor que el de otras, o por un alto grado de solapamiento entre patrones de distintas clases que hace difícil el establecimiento de fronteras de separación, o bien, por la necesidad de trabajar con patrones de muy alta dimensión.

Support Vector learning está basado en ideas simples procedentes de la Teoría Estadística. La simplicidad procede del hecho de que Support Vector Machines (SVMs) aplica un método lineal simple al conjunto de datos de interés pero lo hace en un espacio de dimensión alta no relacionado linealmente con el espacio de partida. Sin embargo, es posible concebir las SVMs como un algoritmo lineal en un espacio de elevada dimensión; en la práctica, no se realizan cálculos en tales espacios. Esta simplicidad permite el tratamiento de ciertos problemas (clasificación, regresión y nueva detección) a través de las SVMs.

En este trabajo nos ocuparemos de los dos primeros, presentando brevemente otros modelos de clasificación no enmarcados en el contexto anterior.

II. Problema de clasificación.

Los problemas de clasificación, parten de dos conjuntos de puntos en el espacio \mathbb{R}^n , para encontrar un hiperplano que separe los dos conjuntos adecuadamente. Ese hiperplano se utilizará para clasificar un nuevo punto; si éste está a un lado del hiperplano, lo clasificamos en el primer conjunto, mientras que si está del otro lado, lo clasificamos en el segundo conjunto.

III. Problema de clasificación SVMs.

Las SVMs buscan el hiperplano separador (al menos para problemas linealmente separables) que maximice el margen entre patrones de las dos clases a separar. En general los problemas no podrán ser tratados desde un punto de vista lineal, pero resolveremos esta situación con el truco del kernel que presentaremos posteriormente.

Comenzaremos presentando el problema de clasificación para dos clases para, a continuación, abordar el problema multiclase a partir de un enfoque binario estudiado al principio de este apartado.

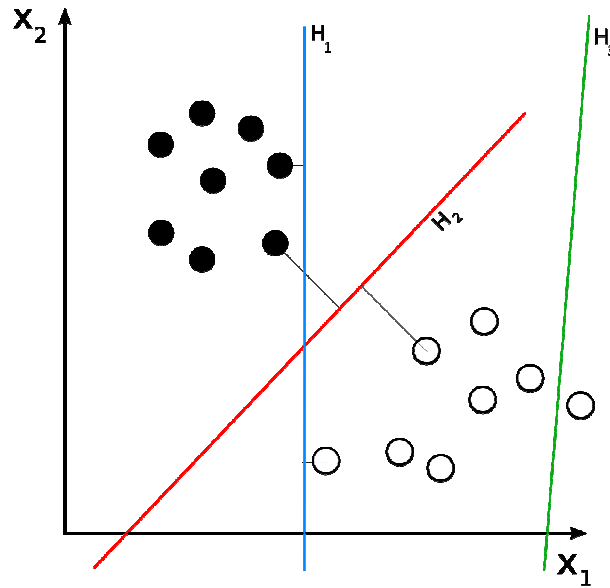
A. Clasificación (dos clases).

Dividiremos el estudio del problema de clasificación de dos clases en el caso de que nuestra muestra sea linealmente separable, o no.

A.I. Problema separable

Analicemos el caso de hallarnos frente a una muestra separable, es decir, que podemos separar las dos clases con un hiperplano sin cometer errores.

Los algoritmos SVM's buscarán un hiperplano clasificador de manera que se maximice la distancia entre el hiperplano y las dos clases.



En esta imagen observamos que la recta H_3 no nos separa las dos muestras. Las rectas H_1 y H_2 si que nos separan las dos clases, pero se aprecia que la distancia de la muestra a H_1 es menor que la distancia de la muestra a H_2 , por ello la segunda es preferible.

Este hiperplano será el resultante de plantear un problema de optimización con restricciones, que se resolverá con el método de los multiplicadores de Lagrange.

Planteemos el problema desde un punto de vista matemático:

Dado una muestra de entrenamiento de tamaño n definimos el conjunto

$$\mathcal{D} = \{(\mathbf{x}_i, c_i) | \mathbf{x}_i \in \mathbb{R}^p, c_i \in \{-1, 1\}\}_{i=1}^n$$

donde x_i es el dato y c_i nos indica la clase a la que pertenece el correspondiente x_i .

Cualquier hiperplano puede ser expresado de la siguiente forma:

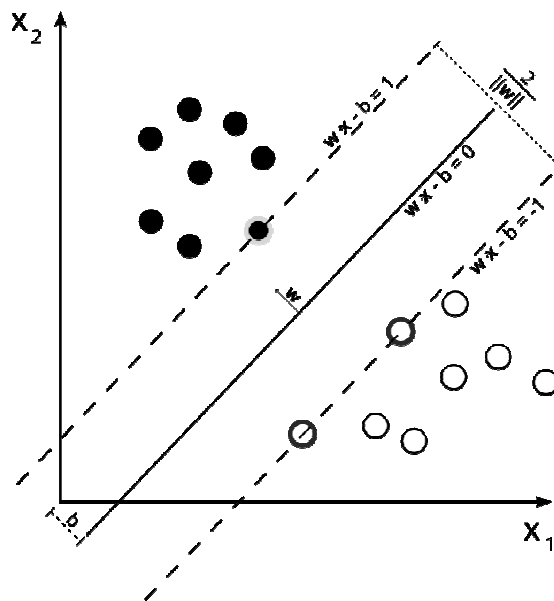
$$w \cdot x - b = 0,$$

donde w es un vector normal (perpendicular al hiperplano), b un número real y donde \cdot denota el producto escalar euclídeo.

Lo que pretendemos es elegir w y b de forma que se maximice la distancia entre los hiperplanos paralelos al original de forma que sigan clasificando los datos. Estos hiperplanos paralelos se pueden expresar mediante las siguientes ecuaciones:

$$w \cdot x - b = 1$$

$$w \cdot x - b = -1.$$



La distancia que separa a las dos nuevas hiperplanos es $\frac{2}{\|w\|}$, entonces, si queremos maximizar esto, hemos de minimizar $\|w\|$. Además tenemos las condiciones de que estos hiperplanos son clasificadores, por lo que se tiene las siguientes dos condiciones:

$w \cdot x_i - b \geq 1$ para los x_i que pertenezcan a una clase.

$w \cdot x_i - b \leq -1$ para los x_i que pertenezcan a la otra.

Estas dos inecuaciones se pueden reescribir de la siguiente manera:

$$c_i(w \cdot x_i - b) \geq 1,$$

donde c_i vale 1 o -1 dependiendo de la clase a la que pertenece x_i .

Resumiendo, tenemos un problema de optimización de la forma:

$$\begin{aligned} \min \|w\| \\ \text{sujeto a } c_i(w \cdot x_i - b) \geq 1, \text{ para } i=1, \dots, n. \end{aligned}$$

Observar que en este problema no pretendemos calcular solo el vector w , si no también el escalar b .

Para mayor comodidad trabajaremos con este otro problema:

$$\begin{aligned} \min_{w,b} \frac{1}{2} \|w\|^2 \\ \text{sujeto a } c_i(w \cdot x_i - b) \geq 1 \text{ para } i=1, \dots, n, \end{aligned}$$

ya que la función norma euclídea tiene un raíz lo que provocaría que el problema fuese más difícil de resolver.

Estamos ante un problema de programación cuadrática (ya que la función objetivo es cuadrática) que resolveremos con los multiplicadores de Lagrange. Combinamos la función a $\frac{1}{2} \|w\|^2$, con las restricciones multiplicadas por unos coeficientes no negativos (los multiplicadores de Lagrange) y minimizamos en w :

$$\Phi(w, b, \alpha_i) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^n \alpha_i [c_i(w \cdot x_i - b) - 1]$$

Anulando la derivada con respecto w tenemos que:

$$w = \sum_{i=1}^n \alpha_i c_i x_i$$

Y anulando la derivada del lagrangiano con respecto a b obtenemos que:

$$\sum_{i=1}^n \alpha_i c_i = 0$$

Este problema es convexo por lo que puede ser resuelto a través de problema dual complementario (maximizando respecto a α_i el mínimo de $-\Phi(w,b)$), que dado por las dos condiciones de arriba viene dado por

$$\max_{\alpha_i} -\frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j c_i c_j x_i \cdot x_j + \sum_{k=1}^n \alpha_k$$

Sujeto a

$$\alpha_i \geq 0$$

$$\sum_{i=1}^n \alpha_i c_i = 0 .$$

Entonces podemos calcular los coeficientes α_i y por lo tanto podemos hallar el vector w .

Los x_i cuyos coeficientes no son cero, se denominan vectores soporte, y satisfacen la condición:

$$c_i (w \cdot x_i - b) = 1$$

es decir, son puntos situados en los hiperplanos paralelos.

Los puntos cuyos coeficientes de Lagrange son cero son aquellos que estan bien clasificados.

Para el cálculo de b utilizaremos la última ecuación. Para los vectores soporte se tiene:

$$w \cdot x_i - c_i = b$$

Entonces tomemos $b = \frac{1}{N_{VS}} \sum_{i=1}^{N_{VS}} (w \cdot x_i - c_i)$ donde N_{VS} es el número de vectores soporte.

A.II.Problema no separable.

Toda esta argumentación se ha hecho bajo el supuesto de que las dos clases son linealmente separables, pero esto no tiene porque ser así, las dos clases pueden estar parcialmente mezcladas, con lo que el problema anterior no sería factible (ya que impone que los datos estén bien clasificados). Lo que haremos será relajar el problema introduciendo variables que nos permitan cometer errores en cierta medida.

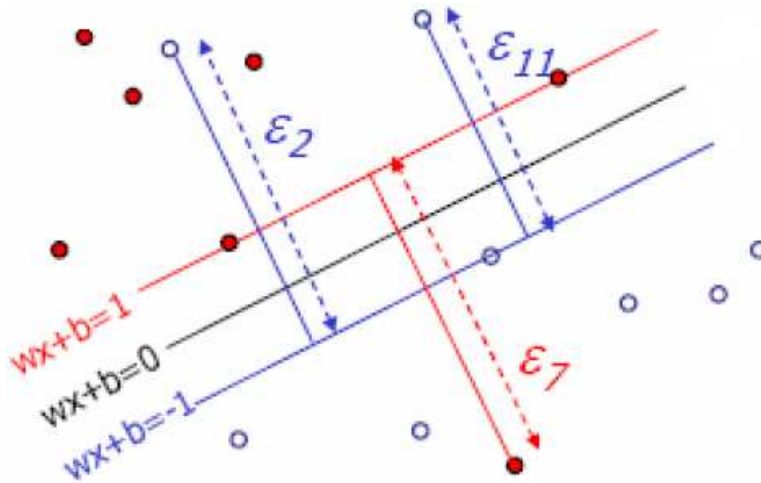
$$\min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i$$

Sujeto a

$$c_i(w \cdot x_i - b) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

Donde C es un parámetro regularizador, de tal forma que para valores grandes el hiperplano calculado se ajustará a los datos, y para valores pequeños no nos preocupamos tanto porque los datos se ajusten bien, sino porque el modelo sea correcto.



Este problema se resuelve como el problema linealmente separable, a través del método de los multiplicadores de Lagrange.

$$\Phi(w, b, \xi_i) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i - \sum_{i=1}^n \alpha_i [c_i(w \cdot x_i - b) - 1 + \xi_i] - \sum_{i=1}^n \eta_i \xi_i$$

Calculando las derivadas parciales e igualándolas a cero obtenemos:

$$w = \sum_{i=1}^n \alpha_i c_i x_i$$

$$\sum_{i=1}^n \alpha_i c_i = 0$$

$$\alpha_i + \eta_i = C.$$

Con estas condiciones planteamos el problema dual para encontrar a los multiplicadores de Lagrange, que es fácilmente resoluble.

$$\max_{\alpha_i} -\frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j c_i c_j x_i \cdot x_j + \sum_{k=1}^n \alpha_k$$

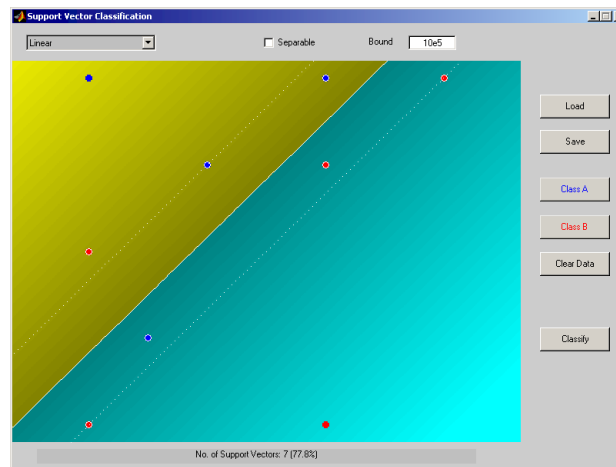
Sujeto a

$$0 \leq \alpha_i \leq C$$

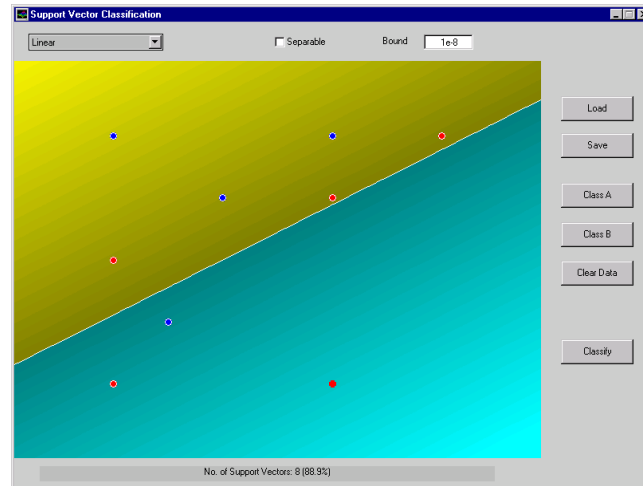
$$\sum_{i=1}^n \alpha_i c_i = 0$$

La solución a este problema es la misma que en el caso separable, con la excepción de que coeficientes de Lagrange esta acotados por C.

Ahora se mostrarán unos ejemplos donde se ve la influencia de C en el resultado final



Para $C=10^5$ vemos que la recta ajusta bastante bien los datos (solo tiene dos errores), pero podría presentar sobreajuste, ya que se esta forzando el modelo a que clasifique bien los datos.

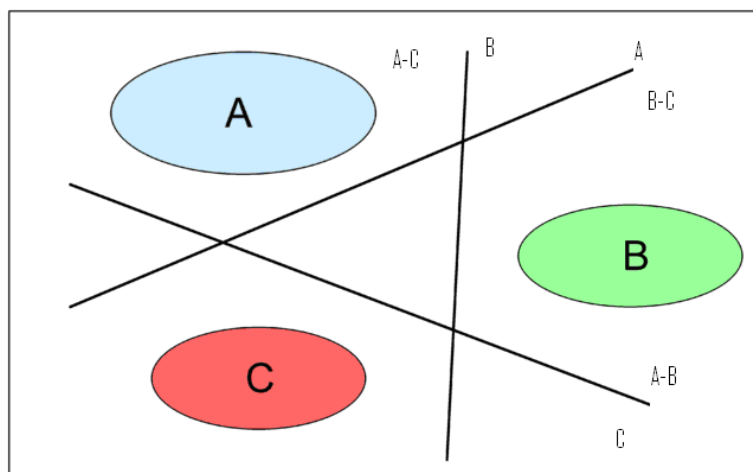


Para $C=10^{-8}$ vemos que la recta no clasifica bien, ante estos datos, por lo que no hay sobreajuste, pero se le da demasiada libertad al modelo, por lo que podría no tener validez.

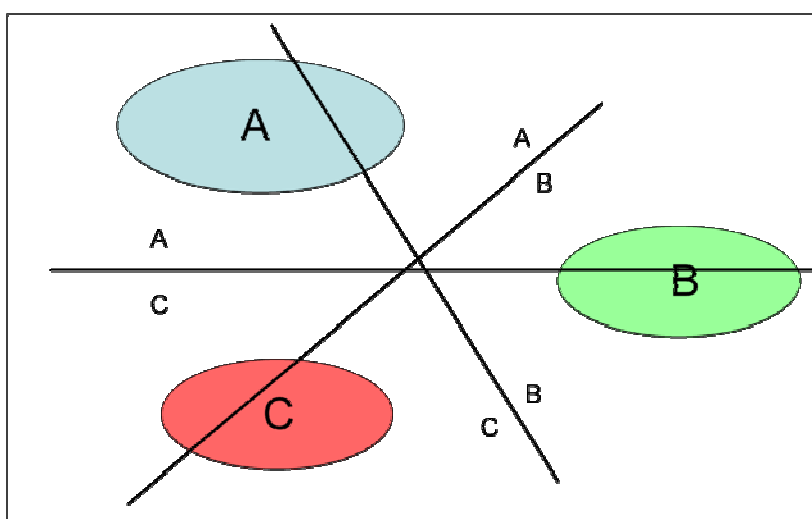
B. Problema multiclase.

Si tenemos un población con varias clases lo que haremos es pasar a varios problemas binarios, que han sido estudiados en el apartado anterior. En este sentido hay dos posibles enfoques:

- "Uno contra todos". Lo que haremos será, dado un dato nuevo enfrentaremos cada clase con el resto de las clases pensadas como una sola.



- "Uno contra uno". Dado un nuevo dato enfrentaremos todas las clases entre si de forma binaria, de modo que a cada ganador en cada enfrentamiento se le asigna un voto, y una vez hechas todas las pruebas la clase que más votos obtenga será a la que pertenezca el nuevo dato.



IV. Métodos Kernel & Kernel Trick.

La idea básica en los métodos Kernel es proyectar de manera adecuada los patrones iniciales en un espacio de muy alta dimensión, donde su separabilidad lineal sea más factible. Un inconveniente de las primeras técnicas al respecto, es el coste computacional que supone dicha inmersión de patrones, así como la necesidad de resolver un problema de programación cuadrática para obtener la máquina de vectores soporte (*support vector machine*, SVM). Es cierto que el *kernel trick*, el cual es una formulación dual del problema que reduce la información necesaria al cálculo de productos escalares, acota en cierta medida esta complejidad, pero aún así, ésta sigue siendo considerable.

Las SVMs asignan implícitamente mediante Φ los datos de entrada en un espacio de características definido por una función kernel, es decir, una función que devuelve el producto interior $\langle \Phi(x), \Phi(x') \rangle$ entre las imágenes de dos datos en el espacio de características, el cual será de dimensión alta.

El estudio entonces tendrá lugar en el espacio de características, donde los puntos correspondientes a los datos sólo aparecen dentro de productos escalares con otros puntos. Este procedimiento se conoce generalmente como *Kernel trick* (o Truco del Kernel).

Más concretamente, si se utiliza una proyección $\Phi: X \rightarrow H$, el producto escalar $\langle \Phi(x), \Phi(x') \rangle$ se podrá expresar como una función kernel k de la siguiente manera:

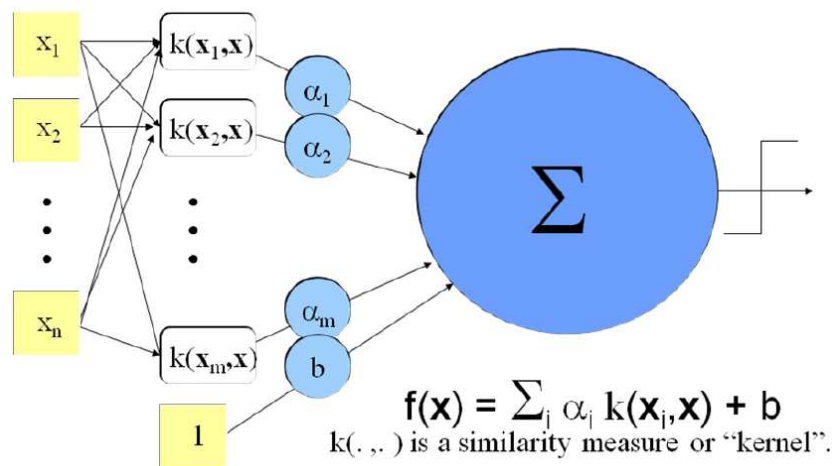
$$k(x, x') = \langle \Phi(x), \Phi(x') \rangle$$

que requiere un coste computacional más bajo que proyectar explícitamente x y x' en el espacio de características H .

Una propiedad interesante de las SVMs y otros métodos Kernel es que una vez que se ha seleccionado una función kernel apropiada, se podrá trabajar prácticamente en espacios de cualquier dimensión sin que ello suponga un coste computacional adicional significativo. De hecho, no es necesario conocer qué características se están utilizando.

Otra ventaja de las SVMs y los métodos kernel es que se puede diseñar y utilizar un determinado kernel para un problema particular que puede ser aplicado directamente sobre los datos sin necesidad de un proceso de extracción.

Gráficamente los métodos núcleo:



Funciones Kernel:

Como hemos visto anteriormente, las funciones kernel devuelven el producto interior entre dos puntos en un espacio de características adecuado, teniendo un bajo coste computacional incluso en espacios de muy alta dimensión.

Los Kernels más utilizados con métodos kernel y en particular con las SVMs incluyen los siguientes:

1.- Kernel lineal (es la función kernel más simple):

$$k(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle$$

2.- Kernel Gausiano radial base (RBF):

$$k(\mathbf{x}, \mathbf{x}') = \exp(-\sigma \|\mathbf{x} - \mathbf{x}'\|^2)$$

3.- Kernel polinómico de grado d :

$$k(\mathbf{x}, \mathbf{x}') = (\langle \mathbf{x}, \mathbf{x}' \rangle + 1)^d$$

4.- Kernel tangente hiperbólica:

$$k(x, x') = \tanh(k \langle x, x' \rangle + c)$$

para algún $k > 0$ y $c < 0$.

5.- Kernel radial de Laplace:

$$k(x, x') = \exp(-\sigma \|x - x'\|)$$

Los kernel de Laplace y Gaussiano se utilizan cuando no se tiene información a priori de los datos. El kernel lineal es muy útil cuando se trabaja con vectores grandes con datos dispersos.

V. Métodos de clasificación alternativos.

A continuación, presentaremos otras alternativas para abordar la clasificación de un nuevo individuo no enmarcadas dentro de las SVMs, desde el punto de vista clásico.

A. El problema de la discriminación.

Recordemos nuevamente el problema de clasificación, dadas n observaciones aleatorias en \mathbb{R}^d , de una población determinada, la suponemos divididas en dos subpoblaciones, P_0 y P_1 . Asumiendo que éstas están perfectamente clasificadas ¿cómo clasificar una nueva observación con origen desconocido?

Formalmente:

Sean (X_i, Y_i) , $i = 1, \dots, n$, donde X_i son valores en \mathbb{R}^d e Y_i es un indicador tomando valores 0 y 1 dependiendo de si nos ocupamos de un elemento de P_0 o de P_1 . Dado un nuevo $Z=(X,Y)$, del que solo se conoce X , deberíamos determinar el valor de Y .

Un clasificador es una sucesión de funciones

$$g_n(x) := g_n(x; (X_1, Y_1), \dots, (X_n, Y_n))$$

definidas en \mathbb{R}^d , con valores en 0,1. Dada una nueva observación X , la predicción para el correspondiente para el valor de Y es dada por $g_n(X)$.

El error del clasificador g_n está dado por:

$$L_n = P\{g_n(X) \neq Y\}$$

Pudiendo ser estimado por error aparente empírico o por error por validación cruzada:

$$\hat{L}_n = \frac{1}{n} \sum_{i=1}^n I_{\{g_n(X_i) \neq Y_i\}}, \quad \tilde{L}_n = \frac{1}{n} \sum_{i=1}^n I_{\{g_{ni}(X_i) \neq Y_i\}}$$

donde g_{ni} denota el clasificador g_n basado en la muestra original omitiendo la i -ésima observación (X_i, Y_i) .

Así, definimos el error de Bayes (o error óptimo) como:

$$L^* = \min_g P\{g(X) \neq Y\}$$

No es difícil ver que el clasificador óptimo es:

$$g^*(x) = \begin{cases} 1 & \text{if } \eta(x) > 1/2, \\ 0 & \text{if } \eta(x) \leq 1/2 \end{cases}$$

donde

$$\eta(x) = P(Y = 1|X = x) = E(Y|X = x)$$

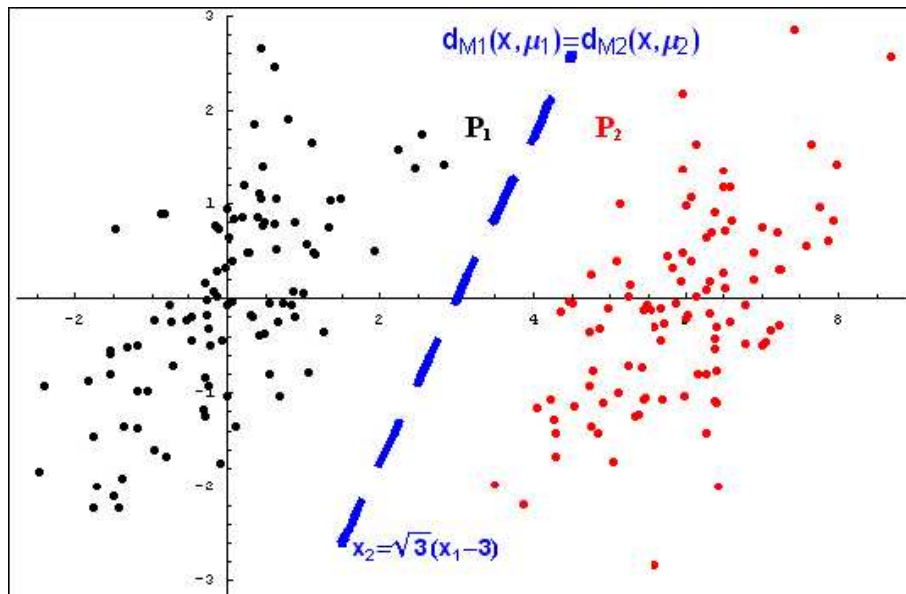
En general, la teoría estadística discriminante se ocupa de la construcción de "buenos" clasificadores, usando dos metodologías diferentes:

I) Búsqueda del clasificador óptimo (minimizando el error empírico).

II) Estimación de la función $\eta(x)$ que aparece en la expresión del clasificador óptimo.

B. Aproximación Clásica: El Método Lineal de Fisher.

¿Como encontrar el "mejor clasificador lineal"? En términos geométricos basta encontrar la ecuación del hiperplano que separa los subespacios para clasificar (con el menor error posible) las observaciones en $P_0 \circ P_1$.



Formalmente, debemos encontrar el clasificador lineal por dos métodos diferentes pero con resultados equivalentes:

(Asumamos para ambos $\Sigma_0 = \Sigma_1 = \Sigma$, siendo las anteriores las matrices de covarianzas de $X|Y$ e $Y|X$).

I) Método de Mahalanobis:

Para asignar la observación x a la población P_0 y P_1 podemos minimizar su distancia de Mahalanobis al correspondiente vector de medias.

II) Método de Fisher:

Debemos encontrar la dirección $\|a\|_{R^d}$ maximizando la separación entre las medias (sujeto a varianza $a' \Sigma a$ es 1).

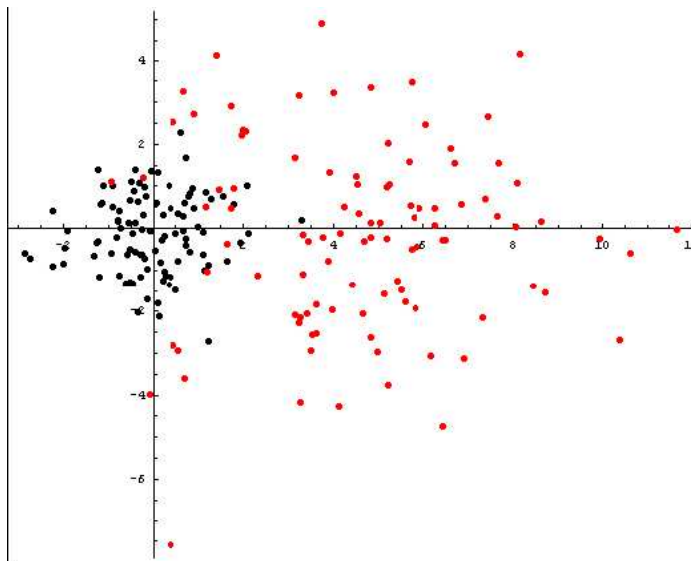
Esto es equivalente a maximizar en a :

$$\frac{(a'(\mu_0 - \mu_1))^2}{a' \Sigma a},$$

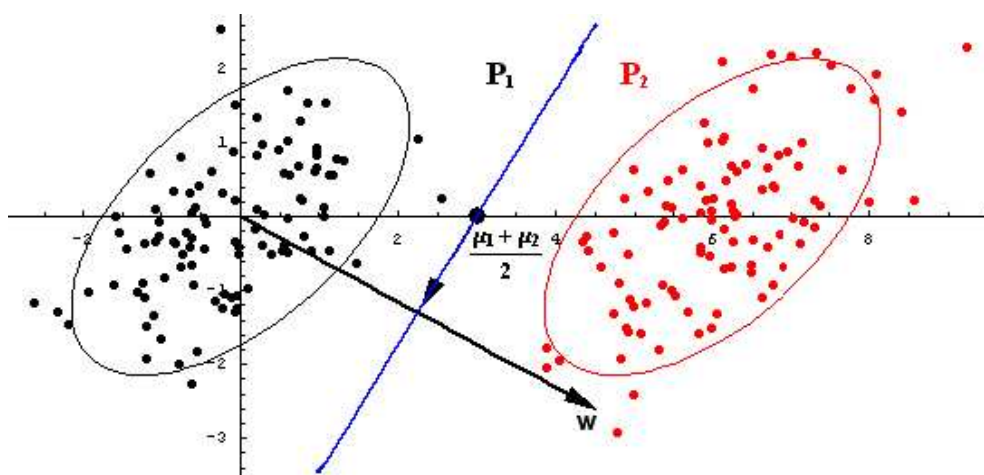
El clasificador lineal de Fisher será óptimo en el caso Gaussiano, es decir, $X|Y$ e $Y|X$ siguen una distribución normal. Un cálculo relativamente simple muestra que ambas aproximaciones proporcionan el mismo clasificador.

NOTAS:

I) La asunción de homocedasticidad es esencial. La siguiente figura muestra por que el método lineal podría ser pobre si no se da la condición anterior (no podremos separar las dos poblaciones por un hiperplano):



II) Geométricamente, podemos interpretar el clasificador lineal como sigue:



C. Método k-vecinos más próximos. Aproximación no paramétrica.

La idea que está detrás del knn-clasificador es muy simple:

Dada una observación x , decidiremos como asignar x por mayoría de voto, es decir, x será asignado a P_0 si el número de k -vecinos más próximos a x perteneciendo a P_0 es mayor que el de los pertenecientes a P_1 .

Estos clasificadores poseen una aplicabilidad que no depende de ninguna restricción realizada sobre la distribución de (X,Y) . De hecho, son universalmente consistentes.

VI. Aplicaciones prácticas. Problema clasificación.

La clasificación resulta de gran utilidad en múltiples disciplinas tales como ciencias sociales o ciencias de la vida:

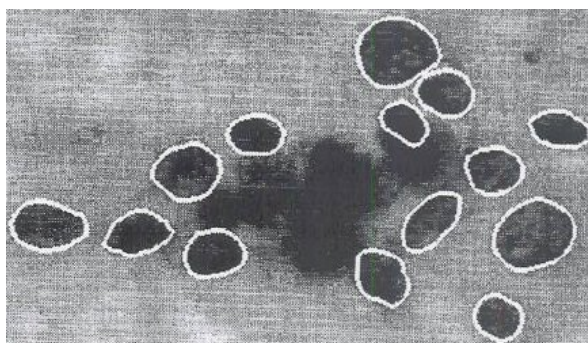
I) En individuos, según similitudes anatómicas, composición genética, relaciones evolutivas, etc. (C. Linneo, s. XVII, C. Darwin, s. XIX, entre otros).

II) En partidos políticos, para estudiar formación de coaliciones, o en actuaciones gubernamentales que determinan diversas implicaciones.

A continuación, presentaremos dos ejemplos donde, a parte de observar la utilidad de esta cuestión en la resolución de problemas reales, mostraremos éstos desde la óptica de la programación lineal:

A. Diagnóstico cáncer de mama.

Gracias al desarrollo de campos como la optimización o el Machine Learning, se ha desarrollado en la década de los noventa un sistema que permite facilitar la diagnosis de tal cáncer, a partir de la observación de masa mamaria FNA (Fine Needle Aspirates).



FNA maligno

Debido al desarrollo de una interface gráfica fue posible la obtención de medidas objetivas y precisas, a través de la digitalización de pequeñas regiones de cada FNA mamario. Para cada núcleo celular se centró la atención en diez

características: área, radio, perímetro, simetría, número y medida de concavidades, dimensión fractal (del borde), compacidad, variación local de segmentos radiales y textura; calculando para cada una de ellas el valor medio, valor extremo y error estándar, resultando así un total de 30 valores reales.

DIAGNOSIS:

Un conjunto de 569 imágenes fueron procesadas, construyendo a partir de las medidas tomadas anteriormente una base de datos con 569 puntos de 30-dimensionales. El proceso de clasificación usado para separar muestras malignas de muestras benignas es una variante del Método Multisuperficie (MSM) conocido como MSM-Tree. Este método usa un modelo de programación lineal para colocar en el espacio de las características una serie de planos separadores, asociados a cualquier problema de clasificación, de modo que si dos conjuntos de puntos son separables, el primer plano será colocado entre ellos en caso contrario, MSM-T construirá un plano que minimice el promedio de las distancias de las clases de los puntos al plano. El proceso se repite sucesivamente para cada dos nuevas regiones consideradas. Los planos resultantes pueden ser usados para clasificar nuevos puntos.

Una cuestión más difícil es la concerniente con el pronóstico de los pacientes con cáncer: ¿volverán a aparecer las células cancerígenas después de la cirugía? Existen estudios que demuestran que este hecho guarda relación con las características celulares observadas durante la diagnosis del cáncer. Éste no es un problema de clasificación usual: mientras un paciente puede ser clasificado en el grupo de los que sufrirán la reaparición de la enfermedad si ésta es observada, no hay un punto de corte real en el cual pueda ser considerado en el grupo de los que no sufren la reaparición. Una de las soluciones a este problema se conoce como técnica RSA (Recurrente Surface Approximation), la cual usa la programación lineal para determinar una combinación lineal de las características de entrada que permitan predecir el tiempo que se tarda en recaer.

El problema se plantea del modo siguiente:

$$\begin{array}{ll}
 \underset{w, \gamma, v, y, z}{\text{minimize}} & \frac{1}{m} e^T y + \frac{1}{k} e^T z + \frac{\delta}{m} e^T v \\
 \text{subject to} & \begin{array}{ll} -v \leq Mw + \gamma e - t & \leq y \\ & -Nw - \gamma e + r \leq z \\ & v \geq 0 \\ & y \geq 0 \\ & z \geq 0 \end{array}
 \end{array}$$

El propósito de este problema lineal es el cálculo del vector w y la constante γ , que determinarán una superficie $s = wx + \gamma$ de los tiempos de reaparición observados. M es una matriz $m \times n$ de m puntos con tiempo de reaparición t . Similarmente, los k puntos donde la enfermedad no ha reaparecido son recogidos en la matriz N ; mientras que sus últimos tiempos conocidos libres de enfermedad son guardados en r . Los vectores y y z representan los errores de puntos en los que reaparece y no reaparece el cáncer respectivamente.

B. Federalist papers.

Otra aplicación interesante de la programación lineal a la clasificación se describe en Bosch y Smith (1998) que usa un hiperplano separador en tres dimensiones que cuenta la frecuencia de ciertas palabras para determinar que 12 cuestionados artículos federalistas son probablemente atribuibles a James Madison antes que a Alexander Hamilton.

Basándonos en el método propuesto por Bradley y Mangasarian, "Feature Selection via Concave Minimization and Support Vector Machines" podremos resolver el conocido problema de clasificación: *Disputed federalist papers*.

Primero, encontramos un plano separador que clasifica correctamente todos los artículos del conjunto de artículos de autor conocido, basándonos en frecuencias de sólo tres palabras. Usando el hiperplano de separación obtenido en tres

dimensiones, los 12 escritos disputados terminaron sobre el lado Madison del plano separador. Este resultado coincide con trabajos anteriores sobre este problema que usan técnicas diferentes.

Los artículos Federalistas fueron escritos en 1787-1788 por Alexander Hamilton, John Jay y James Madison para convencer a los ciudadanos del Estado Nueva York de ratificar la Constitución estadounidense. Como era común en aquel tiempo, estos 77 ensayos cortos, de aproximadamente 900 a 3500 palabras en la longitud, aparecieron en periódicos firmados con un seudónimo, en este caso, "Publius". En 1778 fueron recogidos con ocho artículos adicionales y publicados en forma de libro. Desde entonces, se ha pensado comunmente John Jay era el autor exclusivo de cinco de un total 85, que Hamilton era el autor exclusivo de 51, que Madison era el autor exclusivo de 14, y que Madison y Hamilton colaboraron sobre otros tres. La paternidad literario de los 12 restantes ha estado en la discusión; no había ningún acuerdo sobre cuales fueron escritos por Hamilton y cuales por Madison.

Mosteller y Wallace (1964) En 1964 Mosteller y Wallace en el libro " Inferencia y Disputada autoría: El Federalista " a través de las herramientas propias de la inferencia estadística se concluyendo que Madison era el autor de los 12 ensayos.

Robert A. Bosch y Jason A. Smith (1998) En 1998 Bosch y Smith usaron un método propuesto por Bennett y Mangasarian, que utiliza técnicas de programa lineales para encontrar un hiperplano separador; usando la validación cruzada, obtuvieron el hiperplano siguiente:

$$- 0.5242a_s + 0.8895o_u r + 4.9235u_p o n = 4.7368.$$

Usando este hiperplano se percataron de que los 12 escritos terminaron sobre el lado Madison del hiperplano, que es (> 4.7368).

Selección de características vía minimización cóncava.

Usando el método propuesto por Bradley y Mangasarian: "Feature Selection via Concave Minimization and Support Vector Machine", para encontrar el hiperplano que clasifique los 12 ensayos teniendo en cuenta el menor número de características posibles.

Los datos usados, en este caso, son los mismos que los empleados por Bosch y

Jason. Después de emplear el scanner, ellos emplearon el software competente para calcular frecuencias relativas de 70 palabras que Mosteller y Wallace identificaron como buenas candidatas para realizar estudios de atribución de autores.

1	<i>a</i>	15	<i>do</i>	29	<i>is</i>	43	<i>or</i>	57	<i>this</i>
2	<i>all</i>	16	<i>down</i>	30	<i>it</i>	44	<i>our</i>	58	<i>to</i>
3	<i>also</i>	17	<i>even</i>	31	<i>its</i>	45	<i>shall</i>	59	<i>up</i>
4	<i>an</i>	18	<i>every</i>	32	<i>may</i>	46	<i>should</i>	60	<i>upon</i>
5	<i>and</i>	19	<i>for</i>	33	<i>more</i>	47	<i>so</i>	61	<i>was</i>
6	<i>any</i>	20	<i>from</i>	34	<i>must</i>	48	<i>some</i>	62	<i>were</i>
7	<i>are</i>	21	<i>had</i>	35	<i>my</i>	49	<i>such</i>	63	<i>what</i>
8	<i>as</i>	22	<i>has</i>	36	<i>no</i>	50	<i>than</i>	64	<i>when</i>
9	<i>at</i>	23	<i>have</i>	37	<i>not</i>	51	<i>that</i>	65	<i>which</i>
10	<i>be</i>	24	<i>her</i>	38	<i>now</i>	52	<i>the</i>	66	<i>who</i>
11	<i>been</i>	25	<i>his</i>	39	<i>of</i>	53	<i>their</i>	67	<i>will</i>
12	<i>but</i>	26	<i>if</i>	40	<i>on</i>	54	<i>then</i>	68	<i>with</i>
13	<i>by</i>	27	<i>in</i>	41	<i>one</i>	55	<i>there</i>	69	<i>would</i>
14	<i>can</i>	28	<i>into</i>	42	<i>only</i>	56	<i>things</i>	70	<i>your</i>

Table 1: Function Words and Their Code Numbers

El planteamiento del problema es el siguiente:

$$\min_{w, \gamma, y, z, v} (1 - \lambda) \left(\frac{e^T y}{m} + \frac{e^T}{k} \right) + \lambda e^T (e - \varepsilon^{-\alpha v})$$

sujeto a:

$$\begin{aligned} -Mw + e\gamma + e &\leq y \\ Hw - e\gamma + e &\leq z, \\ y &\geq 0, z \geq 0, -v \leq w \leq v. \end{aligned}$$

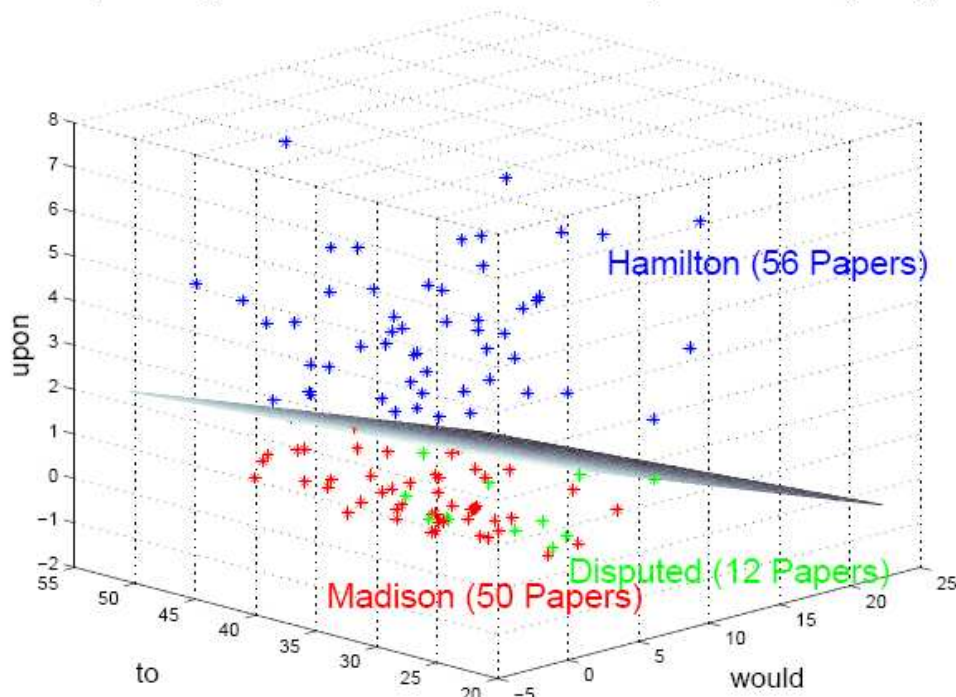
Donde, M es un elemento de $\mathbb{R}^{50 \times 70}$ y H elemento de $\mathbb{R}^{56 \times 70}$ matrices con el conteniendo las frecuencias relativas para las 70

palabras (Madison (M) y Hamilton (H)), $m = 50$, $k = 56$ y e es un vector de unos de dimensión conveniente.

Utilizando el método de selección de características vía minimización cóncava, se obtiene el hiperplano buscado en tres dimensiones ($\alpha=18$, $\lambda=0.03$), que clasifica el conjunto de datos dejando los 12 artículos a un lado del mismo (lado de Madison):

$$-0.5368to - 24.6634upon - 2.9532would = -66.6159,$$

Separating Plane for the Federalists Papers – 1788 (Fung)



VII. Regresión (SVR).

La regresión es el estudio de la dependencia de una variable continua en función de otras (continuas o discretas). A parte de lo de ya por si interesante de este hecho, también sirve para predecir valores de la susodicha variable en función de las variables que la condicionan.

La forma en la que una variable depende de otras es muy diversa, puede una dependencia lineal, polinómica, exponencial, etc.

Empezaremos analizando la regresión lineal de forma que tendremos un problema muy similar al de clasificación, y podremos generalizar a cualquier tipo de regresión con el truco de kernel.

El método SVR consistirá en un problema de optimización, donde buscaremos que el vector normal al hiperplano sea tan pequeño en norma como sea posible (a menor norma la variable respuesta dependerá menos de las variables explicativas y más de la constante) imponiendo cotas a los errores cometidos al aproximar los datos por el hiperplano de regresión.

Sea $D = \{(x_1, y_1), \dots, (x_n, y_n)\}$ con $x \in \mathbb{R}^d, y \in \mathbb{R}$, donde x va a ser el vector de variables explicativas, e y la variable respuesta sobre la que queremos hacer regresión.

Queremos encontrar el hiperplano de regresión de x sobre y , que expresaremos de la siguiente forma:

$$w \cdot x + b = y,$$

donde $w \in \mathbb{R}^d$ es el vector normal al hiperplano, y $b \in \mathbb{R}$.

Por lo que se dijo antes interesa que el vector w sea pequeño en norma, para ello minimizaremos $\frac{1}{2} \|w\|^2$ (ya que es más cómodo). Pero tenemos que tener en cuenta que queremos que el modelo sea fiable, por lo que tenemos que imponer restricciones sobre los errores cometidos al aproximar la variable y por este modelo. Entonces resulta el siguiente problema de optimización:

$$\text{Minimizar } \frac{1}{2} \|w\|^2$$

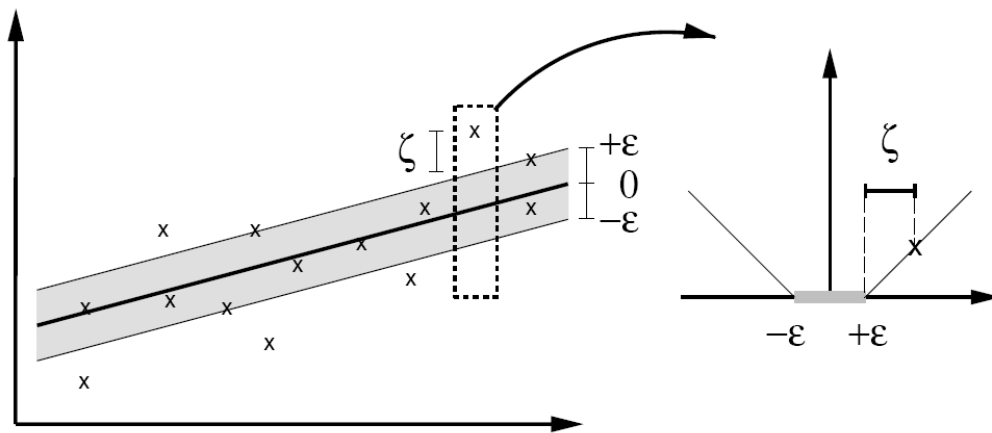
Sujeto a

$$wx_i + b - y_i \leq \varepsilon \quad (\text{para controlar los errores por exceso})$$

$$y_i - wx_i - b \leq \varepsilon \quad (\text{para controlar los errores por defecto})$$

Donde $\varepsilon \in \mathbb{R}$ es una constante fijada de antemano, que sirve de cota para limitar los errores.

En realidad, lo que estamos haciendo es buscar un *tubo de regresión*, ya que las desviaciones de un dato respecto a la recta que sean menores que ε no son considerados.



Recta de regresión

Error al aproximar un punto por la recta de regresión

A la hora de realizar el modelo, interesa que este no se vea afectado por observaciones atípicas que nos modifiquen la recta de regresión, por ello introduciremos en el problema de optimización variables de holgura que flexibilicen las restricciones de los errores, y que permitan que el problema sea factible

$$\text{Minimizar } \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n (\xi_i + \xi_i^*)$$

Sujeto a

$$wx_i + b - y_i \leq \varepsilon + \xi_i \quad (\text{para controlar los errores por exceso})$$

$$y_i - wx_i - b \leq \varepsilon + \xi_i^* \quad (\text{para controlar los errores por defecto})$$

$$\xi_i, \xi_i^* \geq 0$$

Donde C es un índice de equilibrio entre la fidelidad a la muestra y la bondad del modelo. Cuanto más grande sea C más penalizadas estarán ξ_i y ξ_i^* , por lo tanto más pequeñas resultarán y más ajustado estará al hiperplano los datos. En cambio, si tenemos un C pequeños lo que haremos será evitar el sobreajuste.

Este problema se resolverá utilizando el método de los multiplicadores de Lagrange. Planteemos la función lagrangiana.

$$\Phi(w, b, \xi_i, \xi_i^*, \alpha_i, \alpha_i^*, \eta_i, \eta_i^*) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n (\xi_i + \xi_i^*) - \sum_{i=1}^n \alpha_i (\varepsilon + \xi_i - y_i + wx_i + b) - \sum_{i=1}^n \alpha_i^* (\varepsilon + \xi_i^* + y_i - wx_i - b) - \sum_{i=1}^n (\eta_i \xi_i + \eta_i^* \xi_i^*)$$

Haciendo las derivadas parciales obtenemos:

- Respecto a w , que $w = \sum_{i=1}^n (\alpha_i - \alpha_i^*) x_i$.
- Respecto a b , que $\sum_{i=1}^n (\alpha_i - \alpha_i^*) = 0$.
- Respecto a α_i , que $w x_i + b - y_i = \varepsilon + \xi_i$.
- Respecto a α_i^* , que $y_i - w x_i - b = \varepsilon + \xi_i^*$.
- Respecto a ξ_i , que $\eta_i + \alpha_i = C$.
- Respecto a ξ_i^* , que $\eta_i^* + \alpha_i^* = C$.

Entonces planteemos el problema dual (simplificado gracias a las condiciones anteriores) para hallar los multiplicadores de Lagrange α_i y α_i^* .

$$\max \left\{ -\frac{1}{2} \sum_{i,j=1}^n (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) x_i \cdot x_j + \sum_{k=1}^n [\alpha_k (y_k - \varepsilon) - \alpha_k^* (y_k + \varepsilon)] \right\}$$

Sujeto a

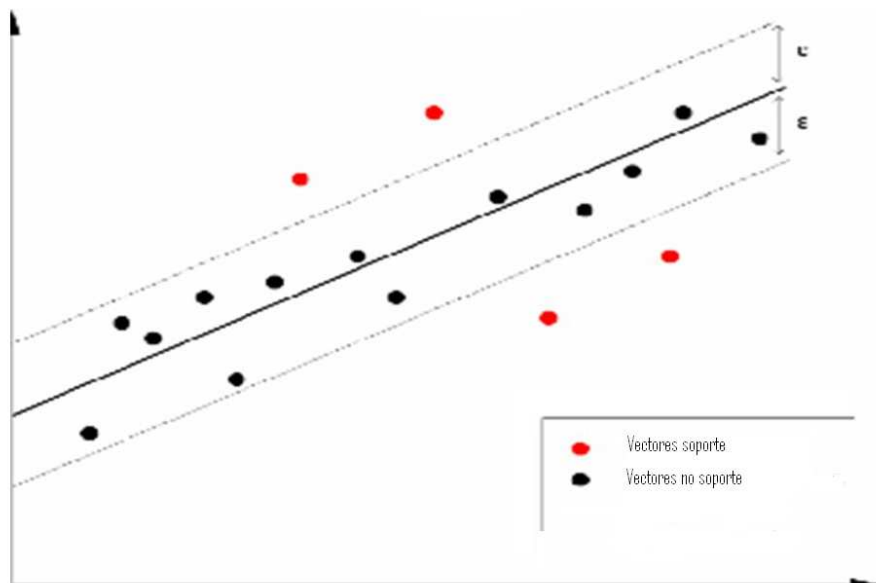
$$\sum_{i=1}^n (\alpha_i - \alpha_i^*) = 0$$

$$0 \leq \alpha_i \leq C$$

$$0 \leq \alpha_i^* \leq C$$

Resolviendo este problema obtendremos los multiplicadores, y por lo tanto la expresión de w .

Notar que por las condiciones de optimalidad de Karush-Kuhn-Tucker se tiene $\alpha_i \alpha_i^* = 0$ para todo i , es decir, que nunca son ambos no nulos; pues bien, los x_i tales que o bien α_i , o bien α_i^* es distinto de cero, se denominan vectores soporte, y estos son los que se encuentran fuera del tubo de regresión.



Apéndice.

En esta sección se recoge el tratamiento del problema de clasificación y regresión utilizando R a partir de ficheros de datos concretos. Como norma se ha considerado el conjunto de entrenamiento como el 50% de los datos de cada clase y como conjunto de testeo el restante 50%. En general, basta modificar las primeras secuencias de los scripts siguientes para considerar otro porcentaje (70%-30%, por ejemplo). Cada programa proporciona la salida en forma de tabla, siendo suficiente observar la diagonal de la misma para contabilizar los datos correctamente clasificados.

Apéndice I. Clasificación dos clases.

A. Problema Accidentes/incidentes.

A.1 SVM.

```
#####
#                               CLASIFICACIÓN SVM                               #
#####

datos=read.csv2("accidentes1.csv",header=FALSE,sep=" ",dec=".")

clases<-factor(c(rep("A",18),rep("I",44)))

library(class)

library(e1071)

#####
#                               SVM con Kernel=RADIAL                               #
#####

train1<-sample(1:18,9)
train2<-sample(19:62,22)      #Tomando mitad (A) y mitad (I).
train<-sort(c(train1,train2)) #Generacion 31 (62/2) n° aleatorios de 1 a 62.

m=svm(datos[train,],clases[train], scale = TRUE, type = NULL, kernel ="radial",
+ gamma = if (is.vector(datos[train,])) 1 else 1 / ncol(datos[train,]))

k=predict(m,datos[-train,])      #Hago la prediccion para los datos restantes.

clases[-train]                  #Valores reales clases para los datos testeados.

table(k,clases[-train])
```

```
#####
#           SVM con Kernel=<POLINOMIAL GRADO 2           #
#####

train1<-sample(1:18,9)
train2<-sample(19:62,22)      #Tomando mitad (A) y mitad (I).
train<-sort(c(train1,train2))  #Generacion 31 (62/2) n° aleatorios de 1 a 62.

m=svm(datos[train,],clases[train], kernel ="polynomial",degree=2)

k=predict(m,datos[-train,])    #Hago la prediccion para los datos restantes.

clases[-train]                 #Valores reales clases para los datos testeados.

table(k,clases[-train])

#####
#           SVM con Kernel=<POLINOMIAL GRADO 3           #
#####

train1<-sample(1:18,9)
train2<-sample(19:62,22)      #Tomando mitad (A) y mitad (I).
train<-sort(c(train1,train2))  #Generacion 31 (62/2) n° aleatorios de 1 a 62.

m=svm(datos[train,],clases[train], kernel ="polynomial",degree=3)

k=predict(m,datos[-train,])    #Hago la prediccion para los datos restantes.

clases[-train]                 #Valores reales clases para los datos testeados.
|
table(k,clases[-train])

#####
#           SVM con Kernel=<POLINOMIAL GRADO 4           #
#####

train1<-sample(1:18,9)
train2<-sample(19:62,22)      #Tomando mitad (A) y mitad (I).
train<-sort(c(train1,train2))  #Generacion 31 (62/2) n° aleatorios de 1 a 62.

m=svm(datos[train,],clases[train], kernel ="polynomial",degree=4)

k=predict(m,datos[-train,])    #Hago la prediccion para los datos restantes.

clases[-train]                 #Valores reales clases para los datos testeados.

table(k,clases[-train])
```


A.2 KSVM.

```
#####
#                               CLASIFICACIÓN KSVM                               #
#####

datos1=read.csv2("accidentes1.csv",header=FALSE,sep="," ,dec=".")

clases<-factor(c(rep("A",18),rep("I",44)))

library(class)

library(kernlab)

datos=data.frame(datos1,clases)

#####
#   C-svc C classification, Kernel=Radial Basis kernel "Gaussian"               #
#####

train1<-sample(1:18,9)
train2<-sample(19:62,22)      #Tomando mitad (A) y mitad (I).
train<-sort(c(train1,train2))  #Generacion 31 (62/2) n° aleatorios de 1 a 62.

m=ksvm(clases[train]~., data = datos[train,],type = "C-bsvc", kernel = "rbfdot",
+kpar = list(sigma = 0.1), prob.model = TRUE)

k=predict(m,datos1[-train,])    #Predicción para el conjunto a testear.

clases[-train]                 #Clases verdaderas.

table(k,clases[-train])

#####
#   Nu-svc nu classification , Kernel=Radial Basis kernel "Gaussian"           #
#####

train1<-sample(1:18,9)
train2<-sample(19:62,22)      #Tomando mitad (A) y mitad (I).
train<-sort(c(train1,train2))  #Generacion 31 (62/2) n° aleatorios de 1 a 62.

m=ksvm(clases[train] ~ ., data = datos[train,],type = "nu-svc", kernel = "rbfdot",
+kpar = list(sigma = 0.1), nu = 0.2, prob.model = TRUE)

k=predict(m,datos1[-train,])    #Predicción para el conjunto a testear.

clases[-train]                 #Clases verdaderas.

table(k,clases[-train])
```

A.3 Clasificación lineal de Fisher.

```
#####
#                               CLASIFICACION LINEAL                               #
#####

datos=read.csv2("accidentes1.csv",header=FALSE,sep="," ,dec=".")

clases<-factor(c(rep("A",18),rep("I",44)))

pairs(datos,pch=c(rep(16,18),rep(1,44)),col=c(rep("red",18),rep("blue",44)))

#####
#                               Modelo lineal:Análisis discriminante lineal de Fisher I                               #
#####

library(MASS)

df<- data.frame(datos, clases) #Data.frame que incluye columna de clase I/A.

train<-sample(1:62,31)          #Generacion 31 (62/2) n° aleatorios de 1 a 62.

table(df$clases[train])        #Recuento elementos cada clase ind aleatorios.

z<-lda(clases ~ .,df,prior=c(1,1)/2,subset=train) #Discriminante lineal.

k=predict(z, df[-train, ])$class #Prediccion para datos restantes.

clases[-train]                 #Para comparar lo real con prediccion anterior.

table(k,clases[-train])

#####
#                               Modelo lineal:Análisis discriminante lineal de Fisher II                               #
#####

library(MASS)

df<- data.frame(datos, clases) #Data.frame que incluye columna de clase I/A.

train1<-sample(1:18,9)
train2<-sample(19:62,22)        #Tomando mitad (A) y mitad (I).
train<-sort(c(train1,train2))    #Generacion 31 (62/2) n° aleatorios de 1 a 62.

table(df$clases[train])        #Recuento elementos cada clase ind aleatorios.

z<-lda(clases ~ .,df,prior=c(1,1)/2,subset=train) #Discriminante lineal.

k=predict(z, df[-train, ])$class #Prediccion para datos restantes.

clases[-train]                 #Para comparar lo real con prediccion anterior.

table(k,clases[-train])
```

```
#####
#                               Tasa de error aparente:                               #
#####

library(MASS)

datos.lda<-lda(datos,clases)           #Informacion relevante para el análisis.

datos.lda$scaling                       #Coef funcion discriminante lineal Fisher.

predclas <- predict(datos.lda)$class   #Clasificación datos.

tea <- 1 - sum(predclas == clases)/62

#####
#                               CLASIFICACIÓN VALIDACIÓN CRUZADA                               #
#####

predclas.cv <- lda(datos,clases,CV=T)$class

#####
#                               Tasa de error aparente validacion cruzada                               #
#####

tevc <- 1 - sum(predclas.cv == clases)/62

#####
#                               Comparación tasas de errores: CV y lineal                               #
#####

tea

tevc                                     # Es mayor la tasa de error aparente en
                                         # validacion cruzada.
```

A.4 Clasificación knn.

```
#####
#      CLASIFICACIÓN MÉTODO DE LOS K VECINOS MAS PROXIMOS      #
#####

datos=read.csv2("accidentes1.csv",header=FALSE,sep=" ",dec=".")

clases<-factor(c(rep("A",18),rep("I",44)))

library(class)

#####
#      VOTACION DE LOS k=2 VECINOS MÁS PROXIMOS      #
#####

train1<-sample(1:18,9)

train2<-sample(19:62,22)      #Tomando mitad (A) y mitad (I).

train<-sort(c(train1,train2))  #Generacion 31 (62/2) n° aleat.

k=knn(datos[train,], datos[-train,], clases[train], k =2, prob=TRUE)

clases[-train]      #Clases reales de pertenencia.

table(k,clases[-train])

#####
#      VOTACION DE LOS k=3 VECINOS MÁS PROXIMOS      #
#####

train1<-sample(1:18,9)

train2<-sample(19:62,22)

train<-sort(c(train1,train2))

k=knn(datos[train,], datos[-train,], clases[train], k =3, prob=TRUE)

clases[-train]

table(k,clases[-train])
```

```
#####
# CLASIFICACIÓN MÉTODO DE LOS K VECINOS MAS PROXIMOS VALIDACION CRUZADA #
#####

datos=read.csv2("accidentes1.csv",header=FALSE,sep="," ,dec=".")

library(class)

#####
#                               k=2                               #
#####

train=datos

clases<-factor(c(rep("A",18),rep("I",44))) # Vector clases.

knn.cv(train,clases, k =2, prob=TRUE)

#####
#                               k=3                               #
#####

knn.cv(train,clases,k =3, prob=TRUE)

#####
#                               k=4                               #
#####

knn.cv(train,clases,k =4, prob=TRUE)

#####
# Calculo errores aparentes. Metodo de k vecinos más próximos: #
#####

datos=read.csv2("accidentes1.csv",header=FALSE,sep="," ,dec=".")

clases<-factor(c(rep("A",18),rep("I",44)))

library(class)

#Para k=2:

datos.knn2 <- knn(datos,datos,clases,2)
|
tea.knn2 <- 1 - sum(datos.knn2==clases)/62

#Para k=3:

datos.knn3 <- knn(datos,datos,clases,3)

tea.knn3 <- 1 - sum(datos.knn3==clases)/62
```

```
#####
#                               Con validacion cruzada:                               #
#####

#Para k=2:

datos.knn2cv <- knn.cv(datos,clases,2)

tecv.knn2 <- 1 - sum(datos.knn2cv==clases)/62

#Para k=3:

datos.knn3cv <- knn.cv(datos,clases,3)

tecv.knn3 <- 1 - sum(datos.knn3cv==clases)/62
```

B. Problema Morosidad/No morosidad.

B.1 SVM.

```
#####
#                               CLASIFICACIÓN SVM                               #
#####

datos=read.csv2("credit1.csv",header=FALSE,sep=" ",dec=".")

clases<-factor(c(rep("NM",300),rep("M",700)))

library(class)

library(e1071)

#####
#                               SVM con Kernel=RADIAL                               #
#####

train1<-sample(1:300,150)
train2<-sample(301:1000,350)      #Tomando mitad (NM) y mitad (M).
train<-sort(c(train1,train2))

m=svm(datos[train,],clases[train], scale = TRUE, type = NULL, kernel = "radial",
+gamma = if (is.vector(datos[train,])) 1 else 1 / ncol(datos[train,]))

k=predict(m,datos[-train,])      #Hago la prediccion para los datos restantes.

clases[-train]                  #Valores reales  clases en los datos testeados.

table(k,clases[-train])

#####
#                               SVM con Kernel=<POLINOMIAL GRADO 3                               #
#####

train1<-sample(1:300,150)
train2<-sample(301:1000,350)      #Tomando mitad (NM) y mitad (M).
train<-sort(c(train1,train2))

m=svm(datos[train,],clases[train], kernel = "polynomial",degree=3)

k=predict(m,datos[-train,])      #Hago la prediccion para los datos restantes.

clases[-train]                  #Valores reales  clases en los datos testeados.

table(k,clases[-train])
```

```
#####
#           SVM con Kernel=<POLINOMIAL GRADO 4           #
#####

train1<-sample(1:300,150)
train2<-sample(301:1000,350)      #Tomando mitad (NM) y mitad (M).
train<-sort(c(train1,train2))

m=svm(datos[train,],clases[train], kernel ="polynomial",degree=4)

k=predict(m,datos[-train,])      #Hago la prediccion para los datos restantes.

clases[-train]                  #Valores reales clases en los datos testeados.

table(k,clases[-train])
|
```


B.2 KSVM.

```
#####
#          CLASIFICACIÓN KSVM          #
#####

datos1=read.csv2("credit1.csv",header=FALSE,sep=",",dec=".")

clases<-factor(c(rep("NM",300),rep("M",700)))

library(class)

library(kernlab)

datos=data.frame(datos1,clases)

#####
#      C-svc C classification, Kernel=Radial Basis kernel "Gaussian"      #
#####

train1<-sample(1:300,150)
train2<-sample(301:1000,350)      #Tomando mitad (NM) y mitad (M).
train<-sort(c(train1,train2))

m=ksvm(clases[train] ~ ., data = datos[train,],type = "C-bsvc", kernel = "rbfdot",
+kpar = list(sigma = 0.1), prob.model = TRUE)

k=predict(m,datos1[-train,])      #Predicción para el conjunto a testear.

clases[-train]                    #Valores reales de las clases en los datos testeados.

table(k,clases[-train])

#####
#      Nu-svc nu classification , Kernel=Radial Basis kernel "Gaussian"      #
#####

train1<-sample(1:300,150)
train2<-sample(301:1000,350)      #Tomando mitad (NM) y mitad (M).
train<-sort(c(train1,train2))

m=ksvm(clases[train] ~ ., data = datos[train,],type = "nu-svc", kernel = "rbfdot",
+kpar = list(sigma = 0.1), nu = 0.2, prob.model = TRUE)

k=predict(m,datos1[-train,])      #Predicción para el conjunto a testear.

clases[-train]                    #Valores reales de las clases en los datos testeados.

table(k,clases[-train])

|
```

Apéndice II. Clasificación más de dos clases.

```
#####  
#           CLASIFICACIÓN MÁS DE DOS CLASES           #  
#####  
  
data(iris3)  
  
#####  
#           ANALISIS DISCRIMINANTE LINEAL           #  
#####  
  
library(MASS)  
  
#Creación data.frame tres especies (Setosa,Versicolor,Virginica) ordenadas.  
  
Iris <- data.frame(rbind(iris3[,1], iris3[,2], iris3[,3]),  
+Sp = rep(c("s","c","v"), rep(50,3)))  
  
#Generacion 75 numeros aleatorios de 1 a 150.  
  
train <- sample(1:150, 75)  
  
#Recuento especies con los indices anteriores (conjunto de entrenamiento).  
  
table(Iris$Sp[train])  
  
#Análisis discriminante lineal.  
  
z <- lda(Sp ~ ., Iris, prior = c(1,1,1)/3, subset = train)  
  
#Predicción para los datos restantes (testamos los restantes).  
  
predict(z, Iris[-train, ])$class  
  
#Valor real de la especie en los datos testeados.  
  
Iris$Sp[-train]
```

```
#####
# ANALISIS DISCRIMINANTE CUADRATICO #
#####

library(MASS)

tr <- sample(1:50, 25)

train <- rbind(iris3[tr,,1], iris3[tr,,2], iris3[tr,,3])

test <- rbind(iris3[-tr,,1], iris3[-tr,,2], iris3[-tr,,3])

cl <- factor(c(rep("s",25), rep("c",25), rep("v",25)))

z <- qda(train, cl)

predict(z,test)$class

#####
# K-VECINOS MAS PROXIMOS #
#####

library(class)

train <- rbind(iris3[1:25,,1], iris3[1:25,,2], iris3[1:25,,3])

test <- rbind(iris3[26:50,,1], iris3[26:50,,2], iris3[26:50,,3])

cl <- factor(c(rep("s",25), rep("c",25), rep("v",25)))

knn(train, test, cl, k = 3, prob=TRUE)

#####
# K-VECINOS MAS PROXIMOS VALIDACION CRUZADA #
#####

library(class)

train <- rbind(iris3[,1], iris3[,2], iris3[,3])

cl <- factor(c(rep("s",50), rep("c",50), rep("v",50)))

knn.cv(train, cl, k = 3, prob = TRUE)
```

Apéndice II. Regresión.

```
#####
#          REGRESION  DOS DIMENSIONES - eps-regression          #
#####
|
library(class)

library(e1071)

library(kernlab)

x <- seq(0.1, 5, by = 0.05)          #Creación de los datos.

y <- log(x) + rnorm(x, sd = 0.2)

m <- svm(x, y)                      #Estimacion del modelo y prediccion.

new <- predict(m, x)                #Dependiendo de si y es factor o no,
                                   #por defecto C-classification o eps-regression.

plot(x, y)                          #Visualización.

points(x, log(x), col = 2)

points(x, new, col = 4)

#####
#          REGRESION - nu-regression          #
#####

#Basta considerar en la opción type=nu-regression y dar a nu un valor.

#####
#          REGRESION,  eps-svr epsilon regression          #
#####

x <- seq(-20,20,0.1)                #Creación de los datos.

y <- sin(x)/x + rnorm(401,sd=0.03)

regm <- ksvm(x,y,epsilon=0.01,kpar=list(sigma=16),cross=3)

plot(x,y,type="l")

lines(x,predict(regm,x),col="red")
```

Bibliografía.

- Wikipedia.
- "Support Vector Machines in R"; Karatzoglou, Alexandros y Meyer, David.
- "A Tutorial on Support Vector Regression"; Smola, Alex J. y Schölkopf, Bernhard .
- "Support Vector Machines for Classification and Regression"; Gunn, Steve R..
- "Breast Cancer Diagnosis and Prognosis via Linear Programming"; Wolberg, William H..
- "The Disputed Federalist Papers: SVM Feature Selection via Concave Minimization"; Fung Glenn y Mangasarian, Olvi.

