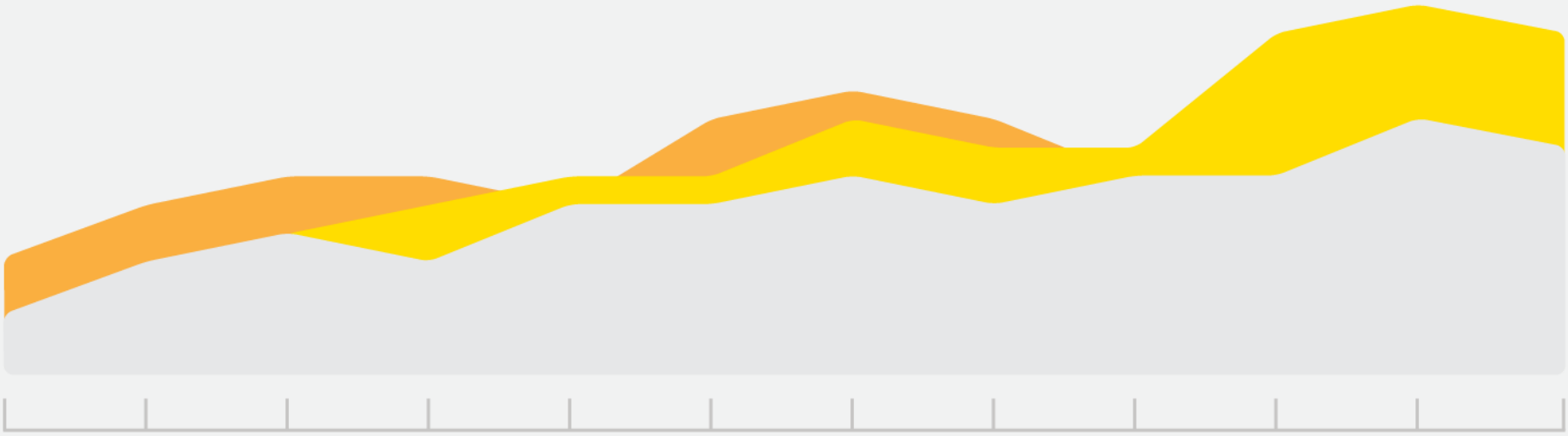
oldemar rodríguez
CONSULTOR en MINERIA DE DATOS

Aprendizaje Supervisado
Máquinas de Soporte Vectorial



SVM

- Support Vector Machines is arguably the most important & interesting recent discovery in Machine Learning.
- Support vector machines were introduced by Vapnik .

Vladimir Naumovich Vapnik

[Wikipedia] Vladimir Vapnik was born in the Soviet Union. He received his master's degree in mathematics from the Uzbek State University, Samarkand, Uzbek USSR in 1958 and Ph.D in statistics at the Institute of Control Sciences, Moscow in 1964. He worked at this institute from 1961 to 1990 and became Head of the Computer Science Research Department. At the end of 1990, Vladimir Vapnik moved to the USA and joined the Adaptive Systems Research Department at AT&T Bell Labs in Holmdel, New Jersey.

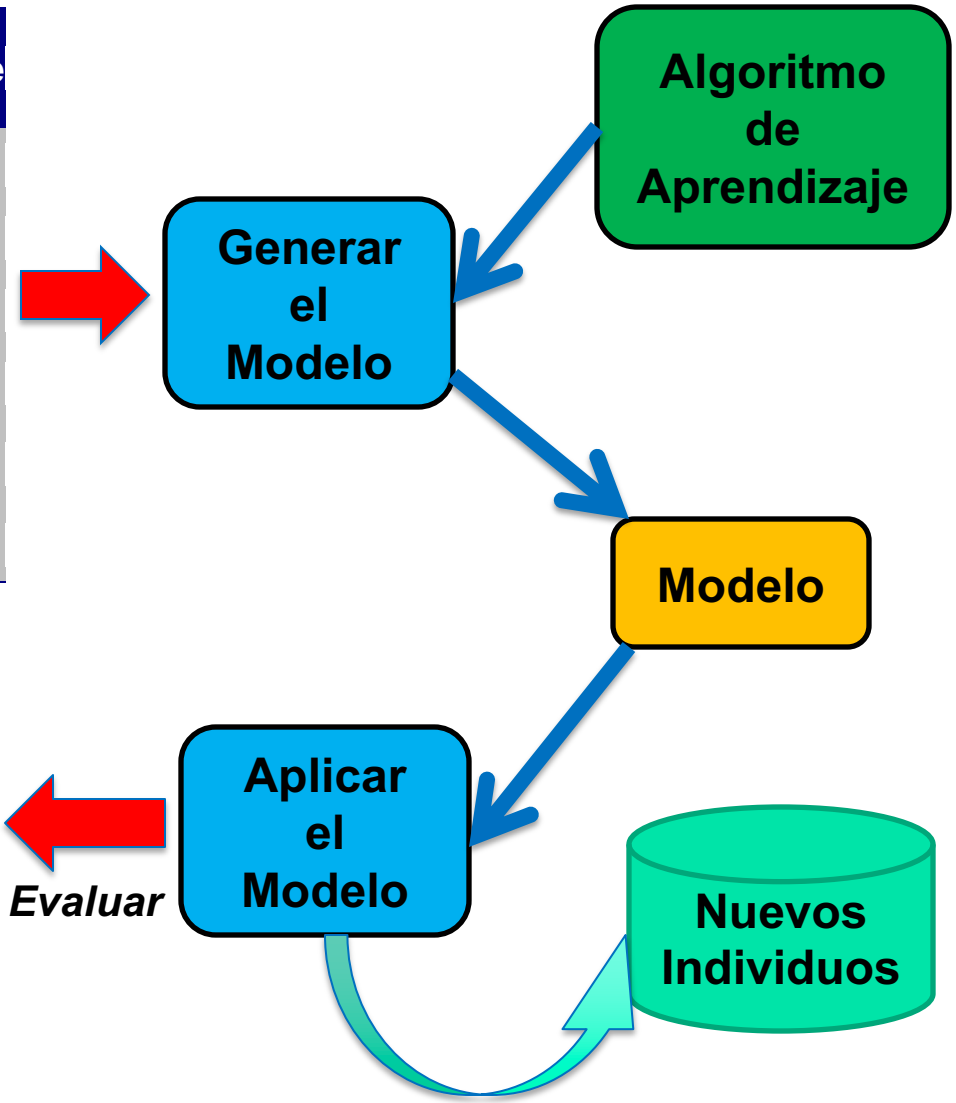
Modelo general de los métodos de Clasificación

Id	Reembolso	Estado Civil	Ingresos Anuales	Fraude
1	Sí	Soltero	125K	No
2	No	Casado	100K	No
3	No	Soltero	70K	No
4	Sí	Casado	120K	No
5	No	Divorciado	95K	Sí
6	No	Casado	60K	No

Tabla de Aprendizaje

Id	Reembolso	Estado Civil	Ingresos Anuales	Fraude
7	No	Soltero	80K	No
8	Si	Casado	100K	No
9	No	Soltero	70K	No

Tabla de Testing



Definición de Clasificación

- Dada una base de datos $D = \{t_1, t_2, \dots, t_n\}$ de tuplas o registros (individuos) y un conjunto de clases $C = \{C_1, C_2, \dots, C_m\}$, el **problema de la clasificación** es encontrar una función $f: D \rightarrow C$ tal que cada t_i es asignada una clase C_j.

- $f: D \rightarrow C$ podría ser una Red Neuronal, un Árbol de Decisión, un modelo basado en Análisis Discriminante, o una Red Beyesiana.

Ejemplo: Créditos en un Banco

Tabla de Aprendizaje

Variable Discriminante

OLDEMARRR.DMEx...ditoViviendaPeq

Id	MontoCredito	IngresoNeto	CoeficienteCre...	MontoCuota	GradoAcademico	BuenPagador
1	2	4	3	1	4	1
2	2	3	2	1	4	1
3	4	1	1	4	2	2
4	1	4	3	1	4	1
5	3	3	1	3	2	2
6	3	4	3	1	4	1
7	4	2	1	3	2	2
8	4	1	3	3	2	2
9	3	4	3	1	3	1
10	1	3	2	2	4	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Con la Tabla de Aprendizaje se entrena (aprende) el modelo matemático de predicción, es decir, a partir de esta tabla se calcula la función ***f*** de la definición anterior.

Ejemplo: Créditos en un Banco

Tabla de Testing

Variable Discriminante

OLDEMARRR.DME...iviendaPeqPRED | OLDEMARRR.DMEx...ditoViviendaPeq

	Id	MontoCredito	IngresoNeto	CoeficienteCre...	MontoCuota	GradoAcademico	BuenPagador
▶	11	3	3	3	3	1	2
	12	2	2	2	2	1	1
	13	2	2	3	2	1	1
	14	1	3	4	3	2	2
	15	1	2	4	2	1	1
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

- Con la Tabla de Testing se valida el modelo matemático de predicción, es decir, se verifica que los resultados en individuos que no participaron en la construcción del modelo es bueno o aceptable.
- Algunas veces, sobre todo cuando hay pocos datos, se utiliza la Tabla de Aprendizaje también como de Tabla Testing.

Ejemplo: Créditos en un Banco

Nuevos Individuos

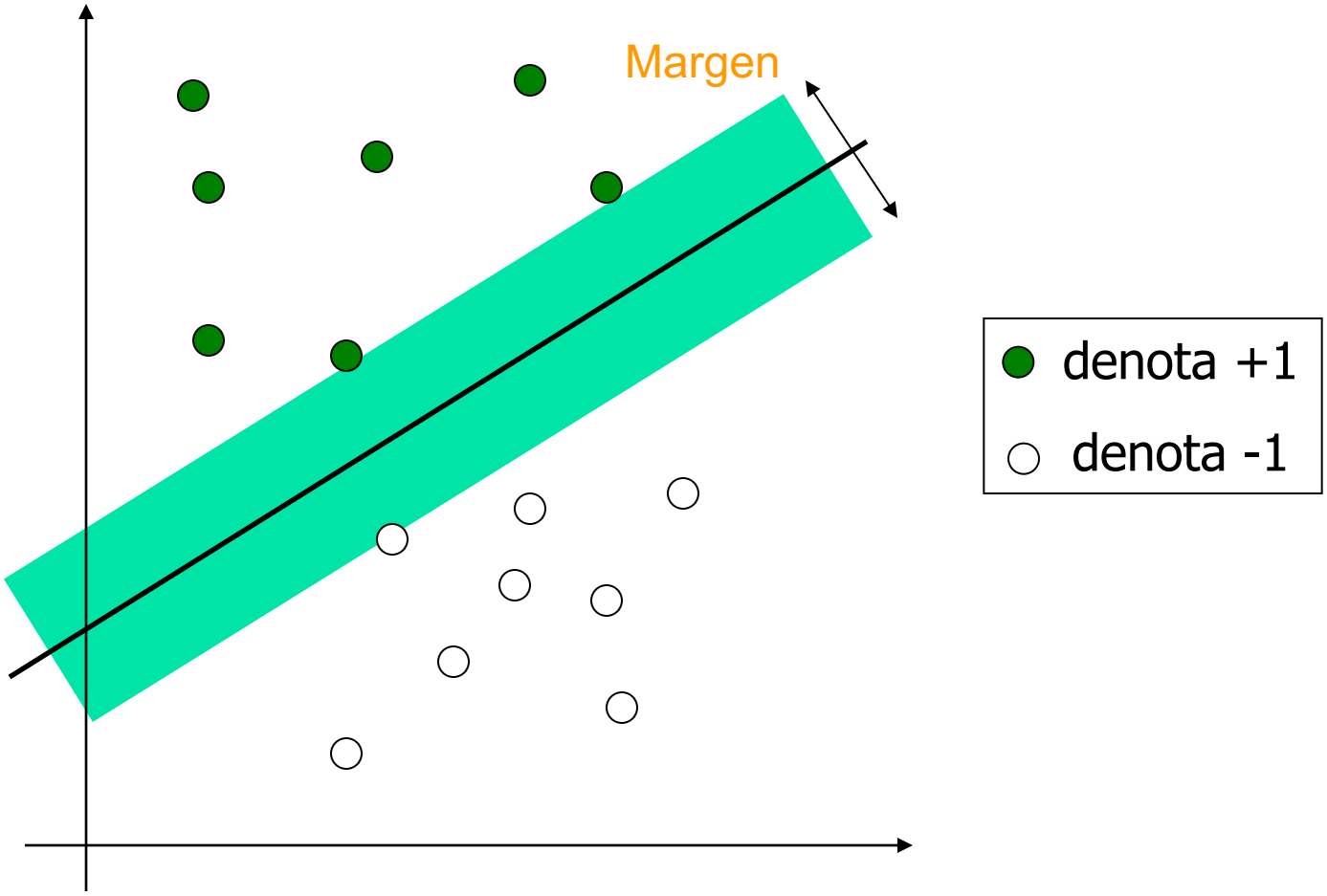
Variable Discriminante

OLDEMARRR.DMEx ...editoViviendaNI

	Id	MontoCredito	IngresoNeto	CoeficienteCre...	MontoCuota	GradoAcademico	BuenPagador
	100	4	4	2	2	3	?
	101	1	4	3	2	4	?
	102	3	2	3	4	2	?
▶*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Con la Tabla de Nuevos Individuos se predice si estos serán o no buenos pagadores.

Idea: Las Máquinas de Soporte Vectorial (Support Vector Machines) tratan de encontrar el hiperplano que separe a las clases con el mayor "margen" posible.

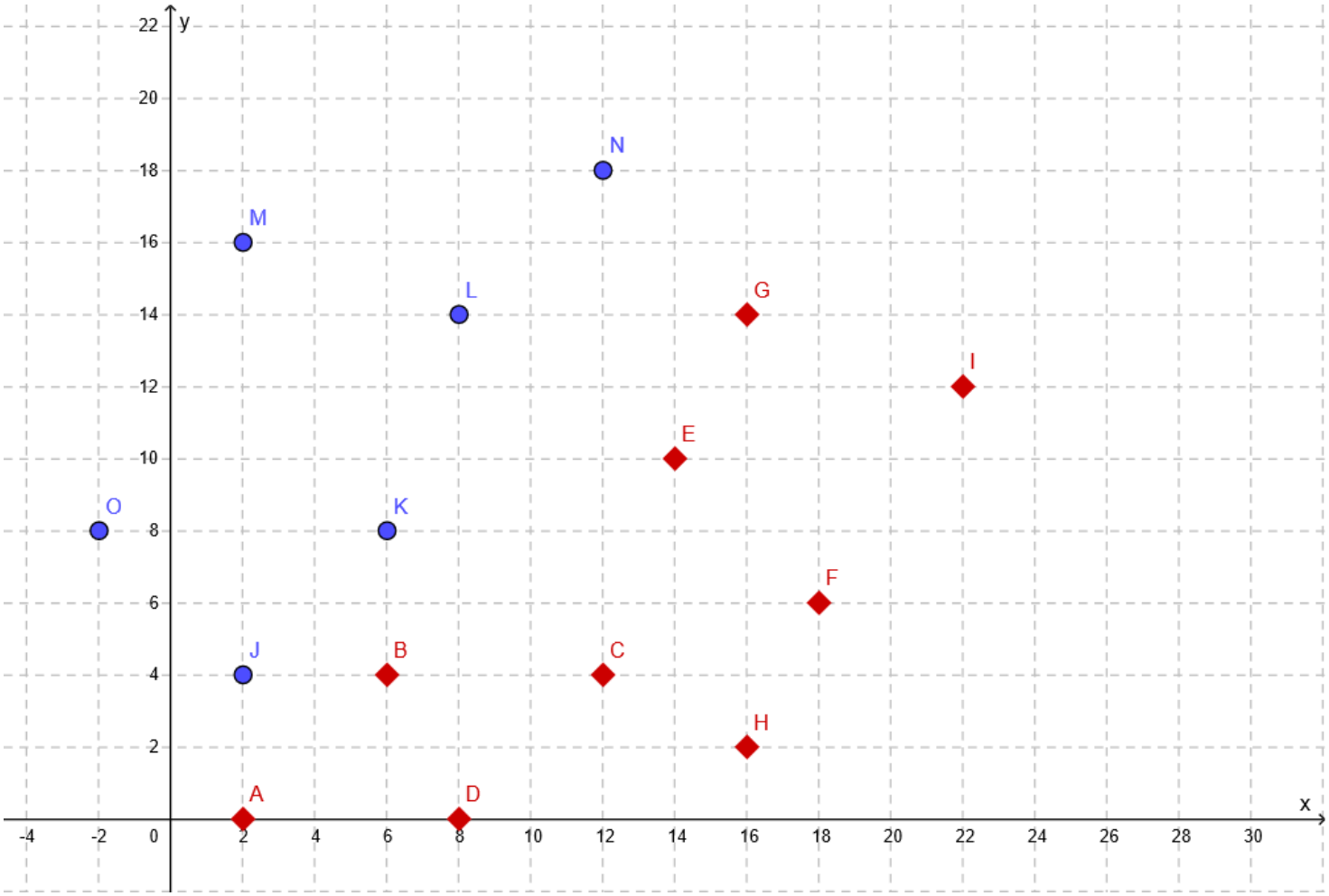


Ejemplo en dos dimensiones

Considere los siguientes datos:

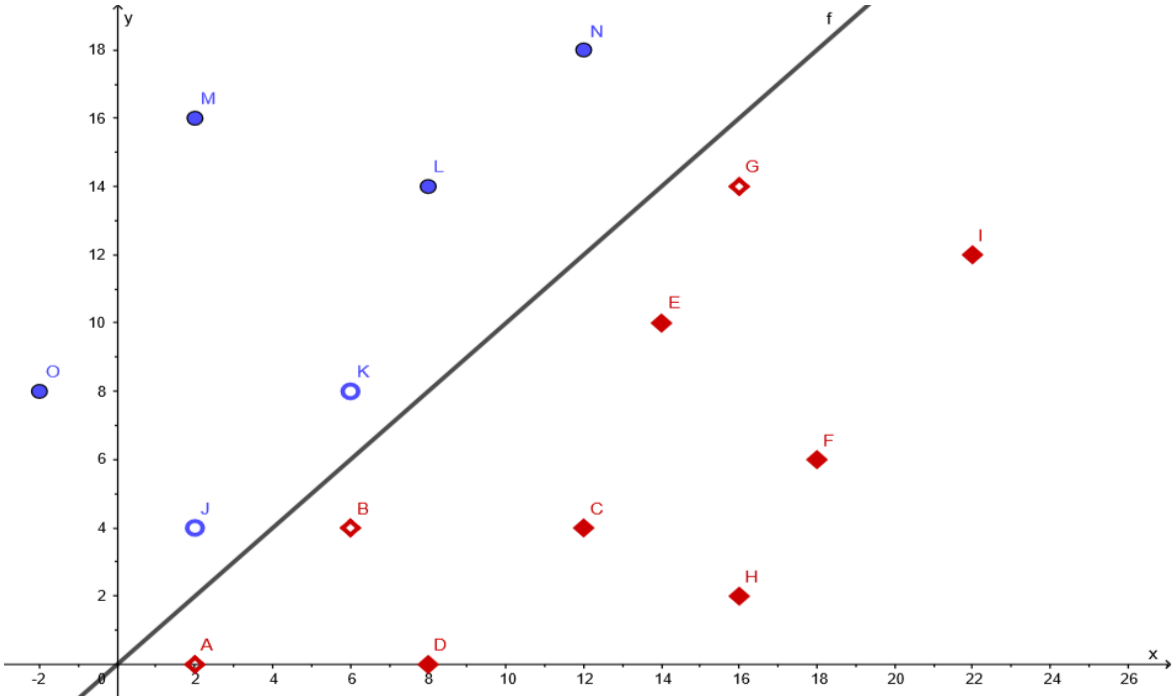
X1	*X2*	*Y*
2	0	Rojo
6	4	Rojo
12	4	Rojo
8	0	Rojo
14	10	Rojo
18	6	Rojo
16	14	Rojo
16	2	Rojo
22	12	Rojo
2	4	Azul
6	8	Azul
8	14	Azul
2	16	Azul
12	18	Azul
-2	8	Azul

Puntos en el plano



Hiperplano de separación

En este caso, haciendo uso de álgebra básica podemos encontrar el hiperplano de separación (en el caso de dos dimensiones una recta) sin necesidad de derivar, usando el punto medio entre los puntos J y A que es P1=(2,2) y el punto medio entre K y B que es P2=(6,6) con la fórmula usual de la pendiente y la recta *y=mx+b*, se puede concluir que es: *y=x o f(x)=x*.



Procedimiento para encontrar f(x): Puntos P1=(2,2) y P2=(6,6)

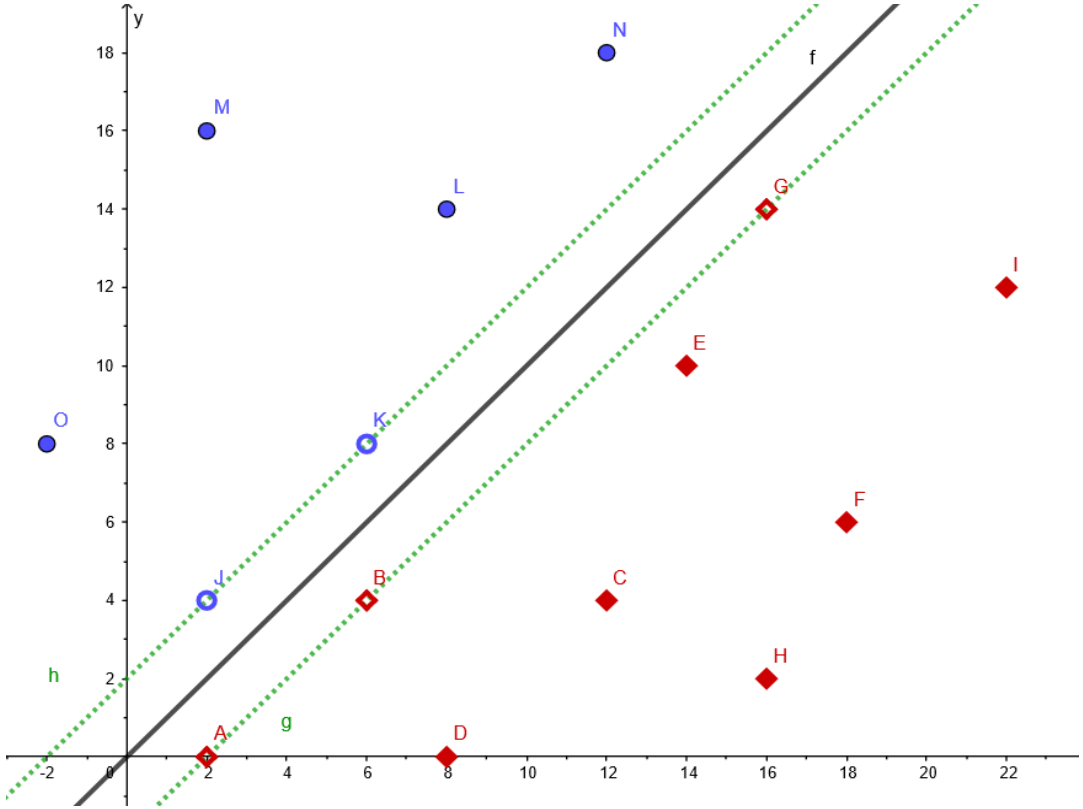
Pendiente m= (Y2-Y1)/(X2-X1) =(6-2)/(6-2)=1

*Usando P1=(2,2) b=y-mx=2-(1*2)=0*

Entonces la recta de separación es f(x)=x

Vectores de soporte y margen

Los vectores de soporte son los puntos *J, K, A, B, y G*, y los denotamos con el centro en blanco, las rectas que delimitan el margen vienen dadas por *h(x)=x+2 y g(x)=x-2.*



Procedimiento para encontrar h(x): Puntos
J=(2,4) y K=(6,8)

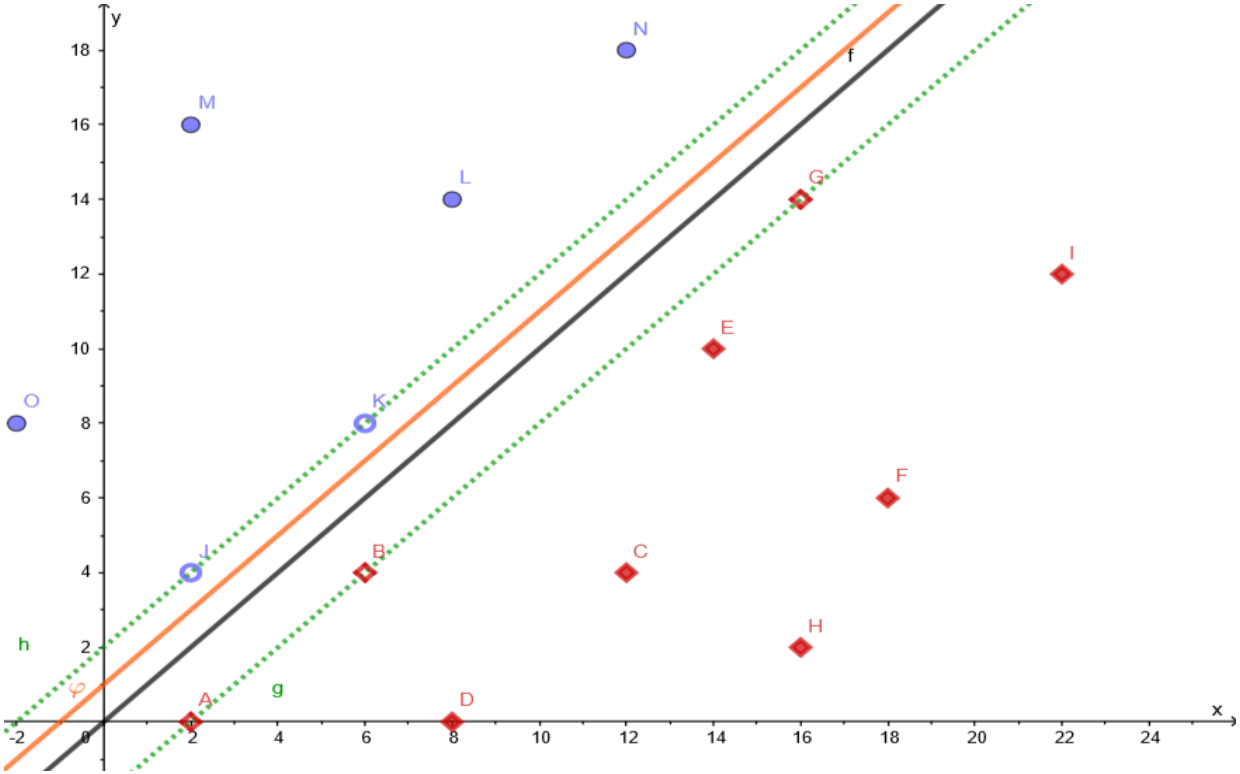
Pendiente
m= (Y2-Y1)/(X2-X1)
=(4-8)/(2-6)=1

b=y-mx
*=4-1*2=2*

g(x) se calcula análogamente

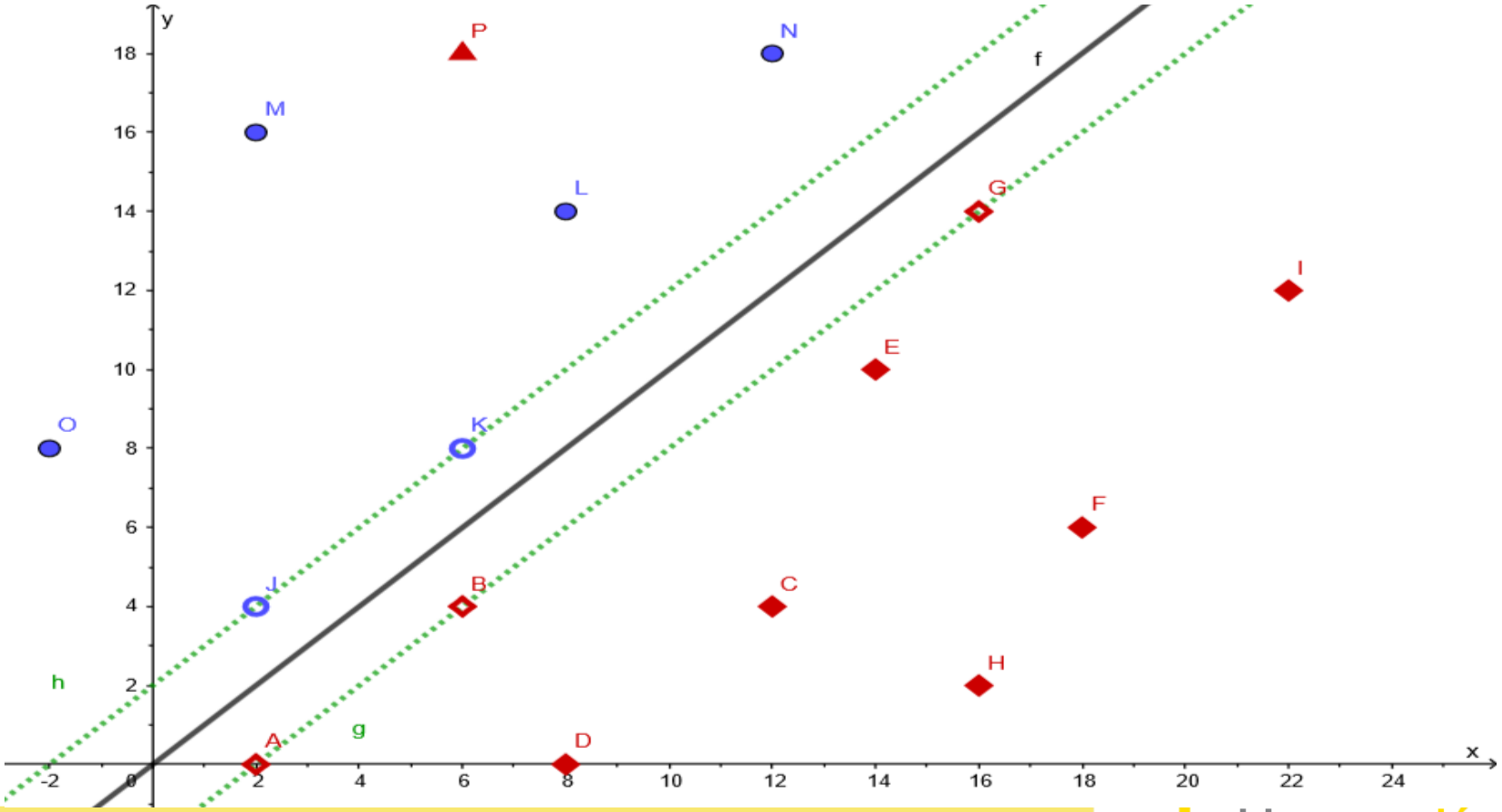
Hiperplano no óptimo

Ahora un hiperplano que separa pero no es óptimo con respecto al mejor margen es la recta naranja, con ecuación $\varphi(x)=x+1$ la cual aunque separa, no tiene margen máximo.



Problema no separable

Si se añade el punto rojo P=(6,18) este ya no es un problema linealmente separable.



Predicción de un nuevo individuo

Si tenemos un individuo nuevo, por ejemplo *Q=(a,b),* lo que hacemos para clasificarlo es seguir la siguiente regla:

- Azul si f(a) < b
- Rojo si f(a) ≥ b

Es decir que si esta por arriba del hiperplano de separación se clasifica como Azul y si esta por debajo del mismo se clasifica como Rojo.

Predicción de un nuevo individuo

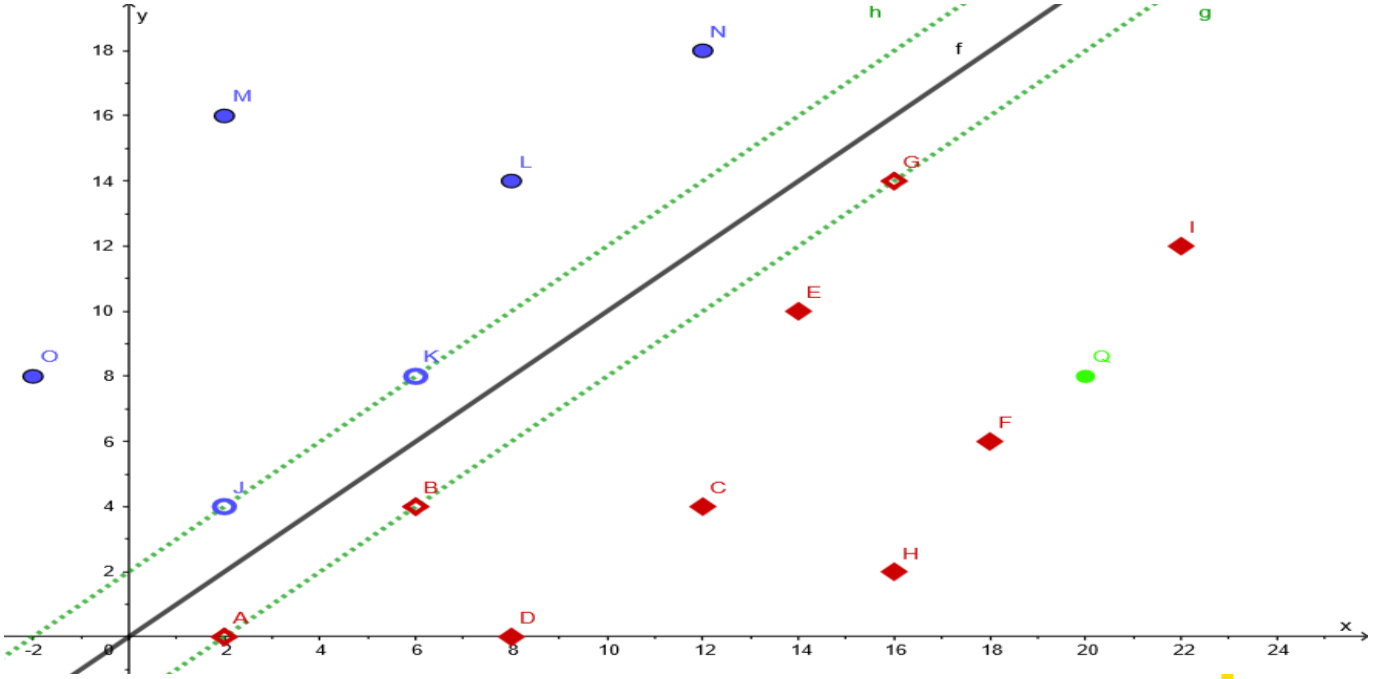
Por ejemplo si tenemos el punto *Q=(20,8)* entonces como *f(x)=x,* tenemos que:

f(20)=20, como 20 < 8 es falso

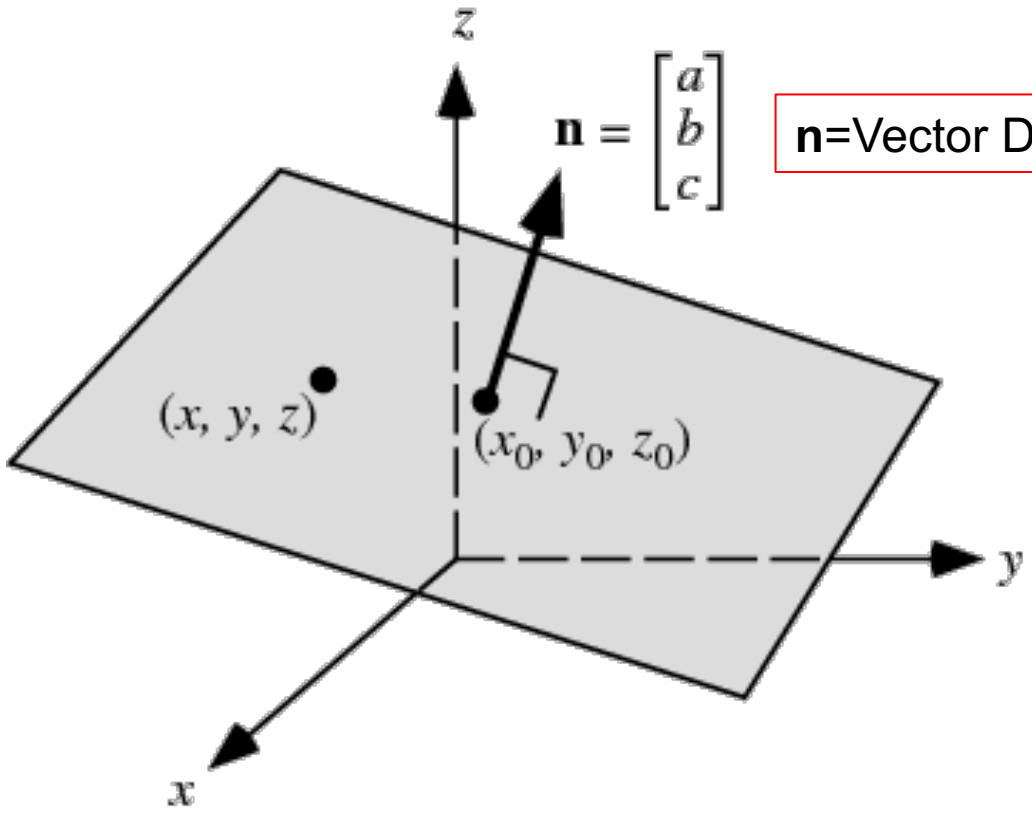
por lo tanto no es Azul.

f(20)=20, como 20 ≥ 8 es verdadero

Por lo tanto se clasifica como Rojo.

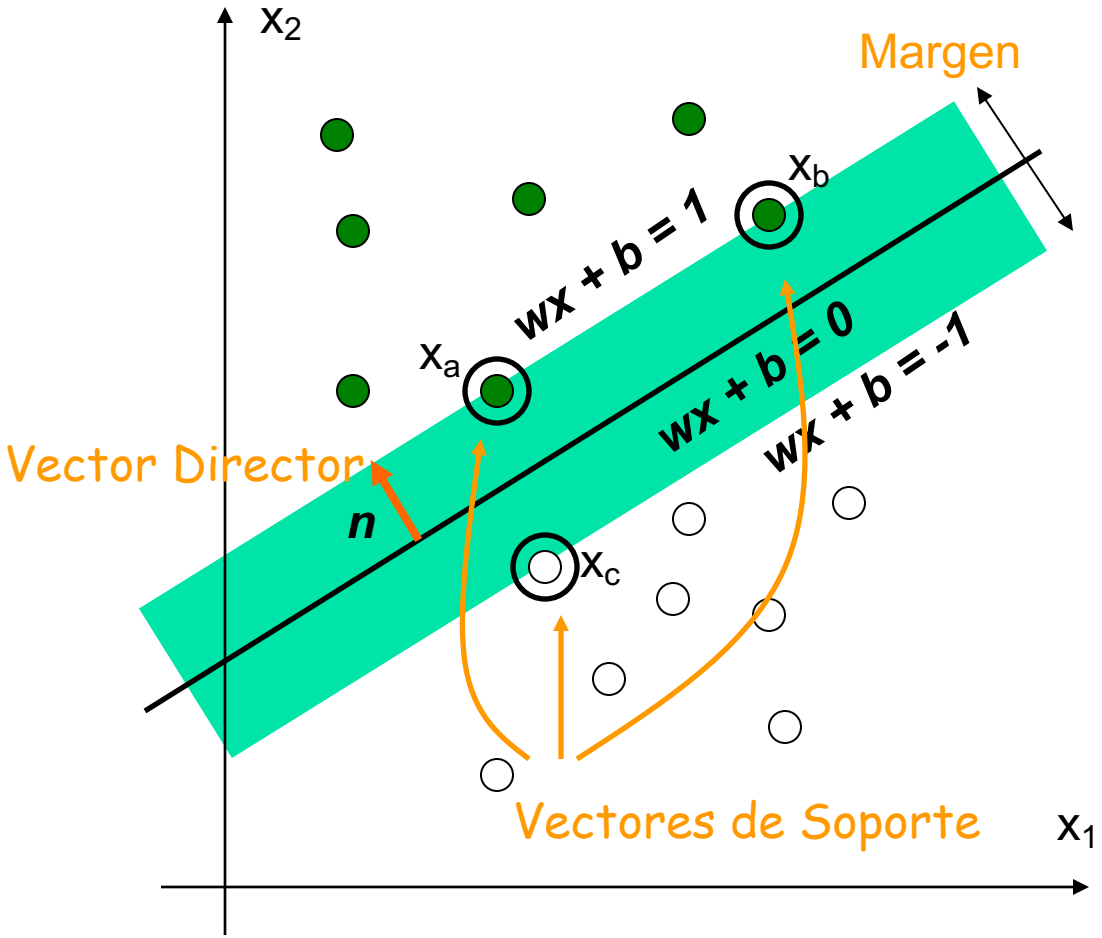


¿Por qué se denominan Máquinas de Soporte Vectorial (Support Vector Machines)?



n=Vector Director

¿Por qué se denominan Máquinas de Soporte Vectorial (Support Vector Machines)?



Función discriminante lineal

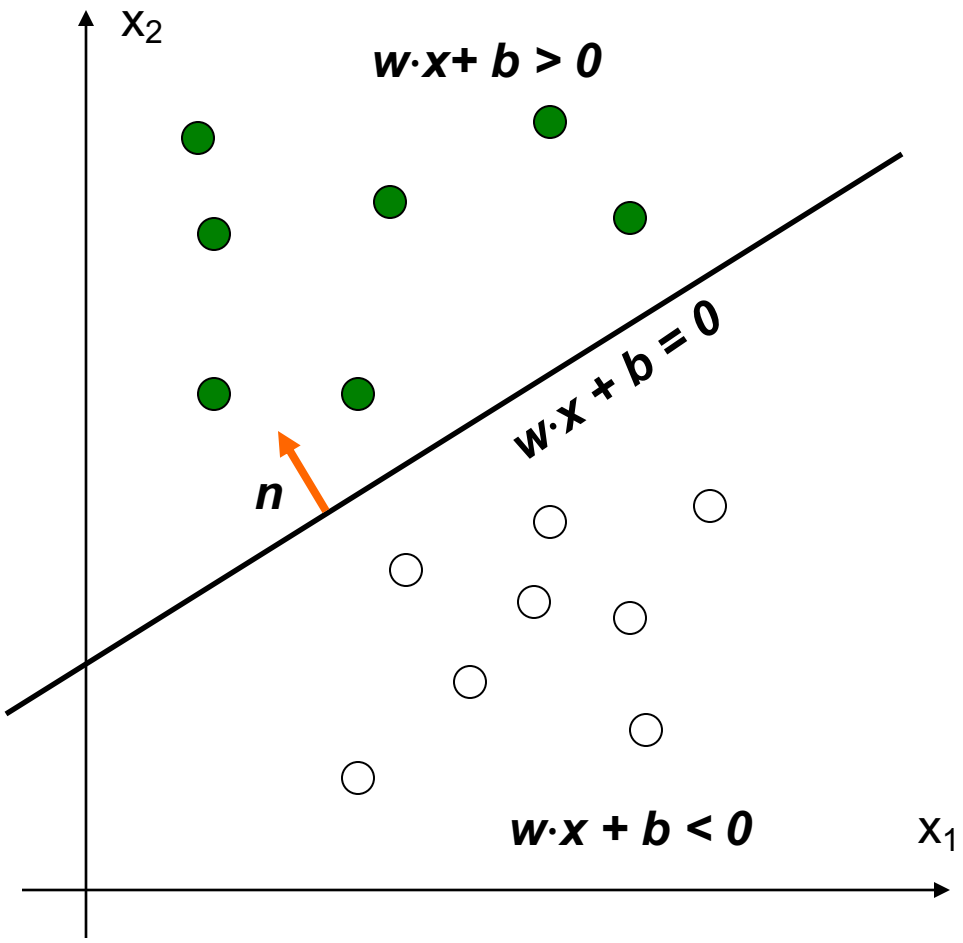
- g(x) es una función lineal:

$$g(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$$

- Se busca un hiperplano en el espacio de las variables.

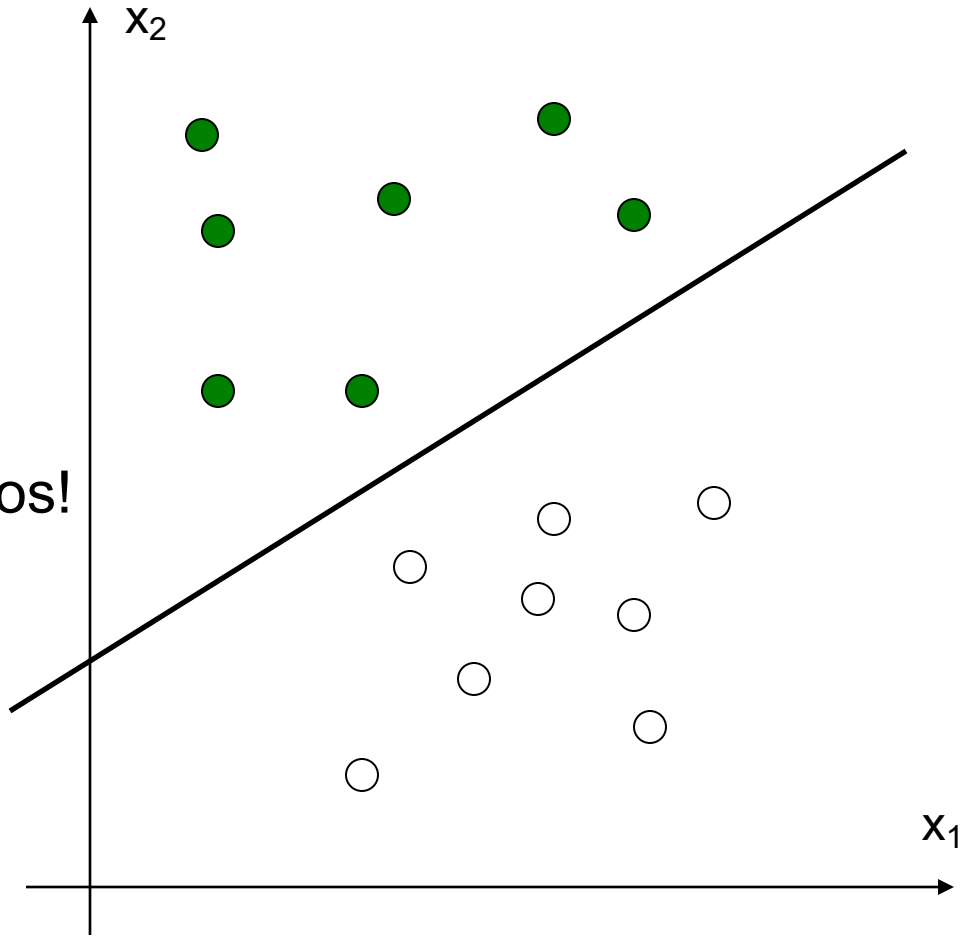
- ***n*** es el vector normal del hiperplano:

$$\mathbf{n} = \frac{\mathbf{w}}{\|\mathbf{w}\|}$$



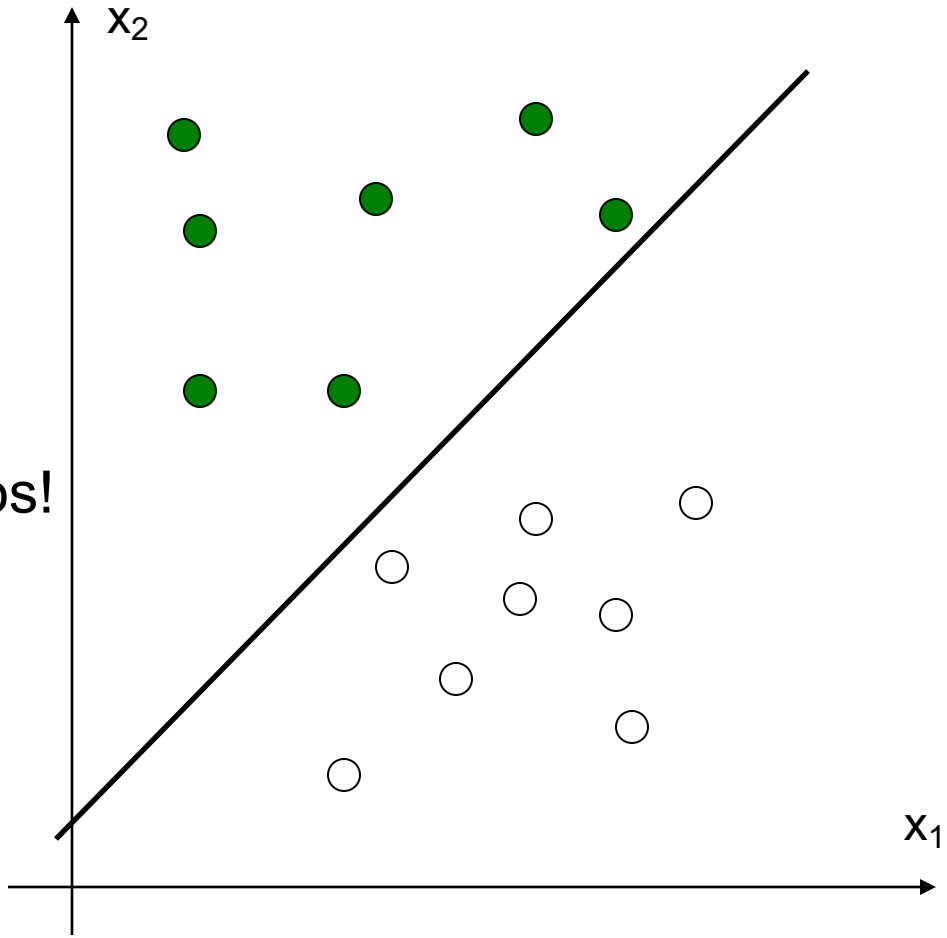
Función discriminante lineal

- ¿Cómo clasificar estos puntos mediante una función discriminante lineal reduciendo al mínimo el error?
- Podrían existir una cantidad infinita de posibles hiperplanos!



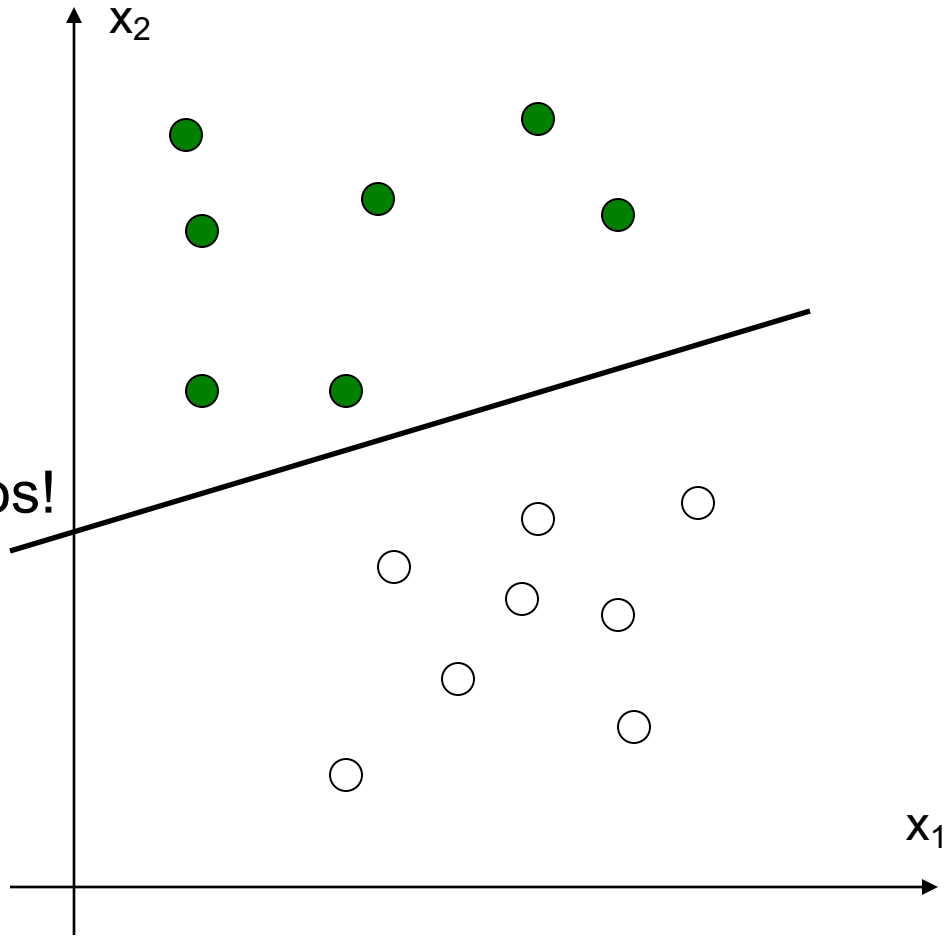
Función discriminante lineal

- ¿Cómo clasificar estos puntos mediante una función discriminante lineal reduciendo al mínimo el error?
- Podrían existir una cantidad infinita de posibles hiperplanos!



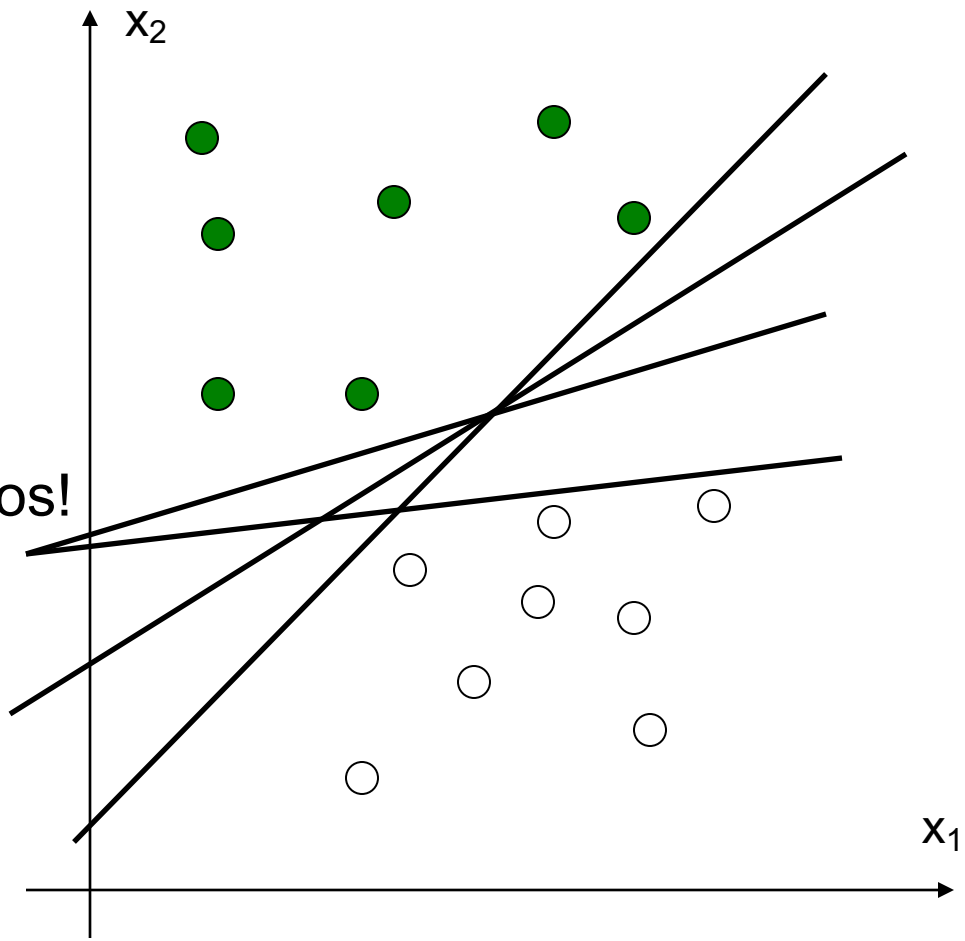
Función discriminante lineal

- ¿Cómo clasificar estos puntos mediante una función discriminante lineal reduciendo al mínimo el error?
- Podrían existir una cantidad infinita de posibles hiperplanos!



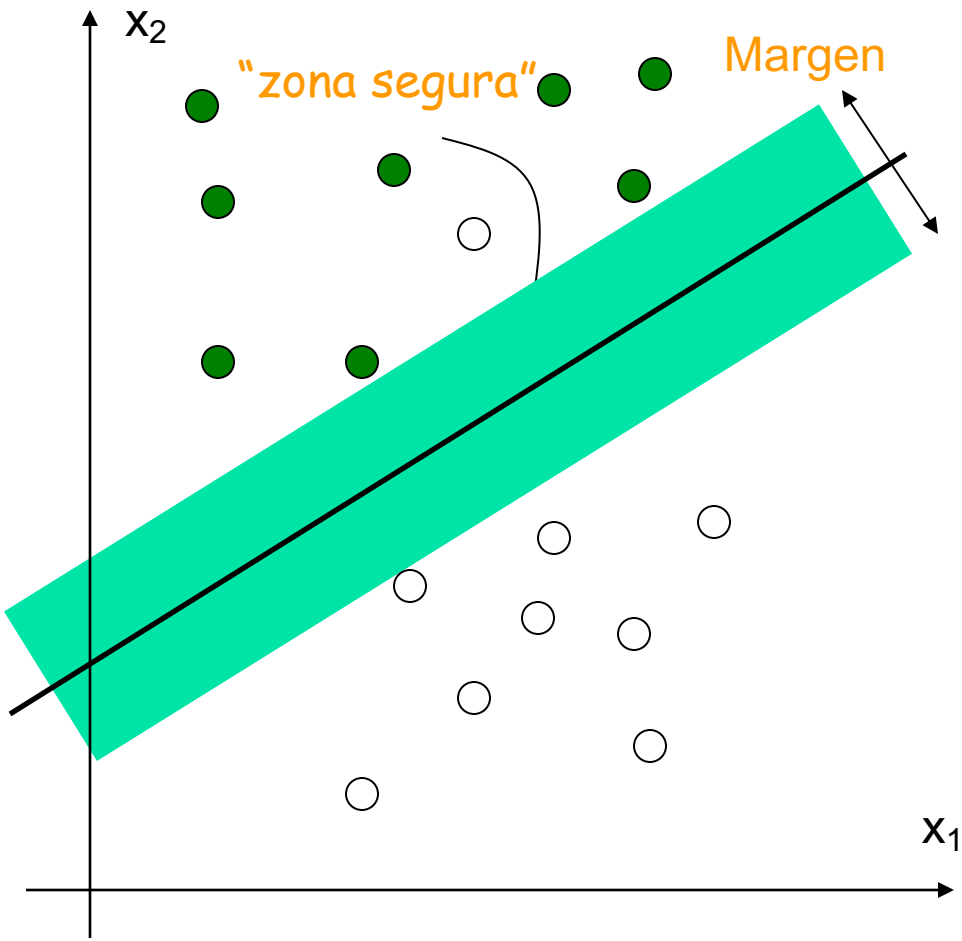
Función discriminante lineal

- ¿Cómo clasificar estos puntos mediante una función discriminante lineal reduciendo al mínimo el error?
- Podrían existir una cantidad infinita de posibles hiperplanos!
- ¿Cuál es el mejor?



Clasificador lineal con el margen más amplio

- La función discriminante lineal con el máximo margen es la mejor.
- El margen se define como la ancho que limita los datos (podría no existir).
- ¿Por qué es la mejor?
  - Generalización robusta y resistente a los valores atípicos.

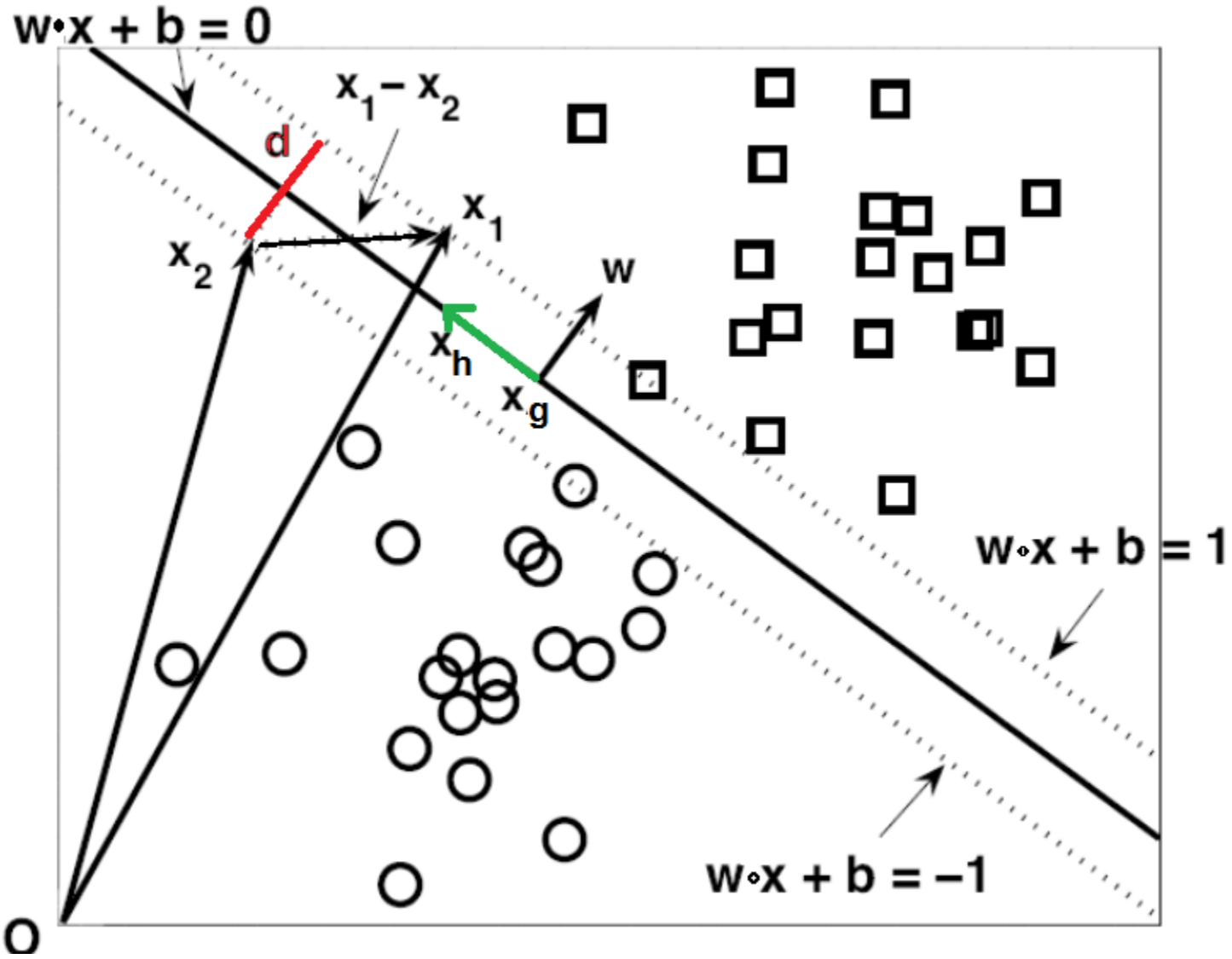


Formulación del Problema: Máquinas de Soporte Vectorial

- Supongamos que tenemos un problema de clasificación donde la variable a predecir es binaria y que tenemos *n* casos de entrenamiento (x_i, y_i) para $i = 1,2,\dots,n$ donde $x_i = (x_{i1}, x_{i2}, \dots, x_{ip})$, es decir, los x_i son los predictores y y_i es la variable a predecir.
- Asumimos que $y_i \in \{-1,1\}$ denota la etiqueta de clase.
- La frontera de decisión de puede escribir como:

$$w \cdot x + b = 0$$

- Donde w y b son los parámetros del modelo.


w•x + b = 0
x1 − x2
d
x1
x2
w
xh
xg
w•x + b = 1
w•x + b = −1
O

Formulación del Problema: Máquinas de Soporte Vectorial

Teorema: w es perpendicular a la frontera de decisión.

Prueba: Sean x_h y x_g en la frontera de decisión entonces:

$$w \cdot x_h + b = 0$$

$$w \cdot x_g + b = 0$$

Restando se tiene que:

$$w \cdot (x_h - x_g) = 0$$

Pero como $x_h - x_g$ es un vector paralelo a la frontera de decisión entonces w es perpendicular a la frontera de decisión.

Formulación del Problema: Máquinas de Soporte Vectorial

- Por otro lado se sabe que si x_s está por arriba de la frontera de decisión entonces:

$$w \cdot x_s + b = k$$

con $k > 0$.

- Análogamente se sabe que si x_c está por debajo de la frontera de decisión entonces:

$$w \cdot x_c + b = k'$$

con $k' < 0$.

Formulación del Problema: Máquinas de Soporte Vectorial

Entonces si etiquetamos los cuadrados con 1 y los círculos con -1 el problema de predecir y para cualquier valor z se puede escribir como:

$$y = \begin{cases} 1 & \text{si} \quad w \cdot z + b > 0 \\ -1 & \text{si} \quad w \cdot z + b < 0 \end{cases}$$

Teorema: El margen *d* se puede calcular como:

$$d = \frac{2}{\| w \|}$$

Prueba: Re-escalando w y b de la frontera de decisión se pueden escribir dos hiperplanos paralelos a la frontera de decisión como sigue:

$$b_{i1} : w \cdot x + b = 1$$
$$b_{i2} : w \cdot x + b = -1$$

Así el margen es igual a la distancia entre estos dos hiperplanos, sean $x_1 \in b_{i1}$ y $x_2 \in b_{i2}$ (ver gráfico) entonces es claro que:

$$b_{i1} : w \cdot x_1 + b = 1 \implies w \cdot (x_1 - x_2) = 2$$
$$b_{i2} : w \cdot x_2 + b = -1 \implies d \,||\, w \,|| = 2 \text{ (tarea)}$$
$$\implies d = \frac{2}{||\, w \,||}$$

Formulación del Problema: Máquinas de Soporte Vectorial

Entonces se deben calcular los parámetros del modelo w y b que cumplan:

$$w \cdot x_i + b \geq 1 \quad \text{si} \quad y_i = 1$$
$$w \cdot x_i + b \leq -1 \quad \text{si} \quad y_i = -1$$

Lo cual es equivalente a: $y_i(w \cdot x_i + b) \geq 1$ para $i = 1,2, \ldots, n$.

Además se debe maximizar: $d = \frac{2}{\| w \|}$

Es equivalente a minimizar: $f(w) = \frac{\| w \|^2}{2}$

Resolver un Problema Optimización

Un problema de programación cuadrática con restricciones lineales

$$\min_{w} \frac{\| w \|^2}{2}$$

Sujeto a: $y_i(w \cdot x_i + b) \geq 1$ para $i = 1,2, \ldots, n$

Minimización de Lagrange

$$L_P(w, b, \lambda_i) = \frac{\| w \|^2}{2} - \sum_{i=1}^{n} \lambda_i \left(y_i(w \cdot x_i + b) - 1\right)$$

con $\lambda_i \geq 0$ (los λ se llaman multiplicadores de Lagrange)

Derivando L_P respecto a w y b e igualando a cero se tiene:

$$\frac{\partial L_P}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{n} \lambda_i y_i x_i$$

$$\frac{\partial L_P}{\partial b} = 0 \Rightarrow \sum_{i=1}^{n} \lambda_i y_i = 0.$$

Esto aún no resuelve nada porque los λ_i son desconocidos.

Sustituyendo los anteriores valores en (tarea):

$$L_P(w, b, \lambda_i) = \frac{\| w \|^2}{2} - \sum_{i=1}^{n} \lambda_i \left(y_i(w \cdot x_i + b) - 1\right)$$

- Obtenemos el siguiente problema de optimización (conocido como problema dual) L_D que es el que finalmente se resuelve usando métodos numéricos:

$$L_D = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j x_i x_j$$

- Una vez calculados los λ_j y usando que $w = \sum_{i=1}^{n} \lambda_i y_i x_i$
- La frontera de decisión:

$$w \cdot x + b = 0$$

- Se puede escribir como: $\sum_{i=1}^{n} \lambda_i y_i x_i \cdot x + b = 0$

- Para calcular b sabemos que:

$$y_i(w \cdot x_i + b) - 1 \geq 0 \text{ y que } \lambda_i \geq 0 \text{ para } i = 1,2, \dots, n$$

- Esto es equivalente a (Karush-Kuhn-Tuker) (tarea):

$$\lambda_i[y_i(w \cdot x_i + b) - 1] = 0 \text{ y que } \lambda_i \geq 0 \text{ para } i = 1,2, \dots, n$$

- Si λ_i no es cero entonces $y_i(w \cdot x_i + b) - 1 = 0$ lo que implica que:

$$b = \frac{1}{y_i} - w \cdot x_i \text{ para } i = 1,2, \dots, n$$

- En la práctica b se calcula como el promedio de los b's anteriores.

Ejemplo

Dada la siguiente tabla de datos calcule los parámetros del modelo w y b.

x_1	x_2	y	Lagrange Multiplier
0.3858	0.4687	1	65.5261
0.4871	0.611	−1	65.5261
0.9218	0.4103	−1	0
0.7382	0.8936	−1	0
0.1763	0.0579	1	0
0.4057	0.3529	1	0
0.9355	0.8132	−1	0
0.2146	0.0099	1	0

$$w_1 = \sum_i \lambda_i y_i x_{i1} = 65.5621 \times 1 \times 0.3858 + 65.5621 \times -1 \times 0.4871 = -6.64$$

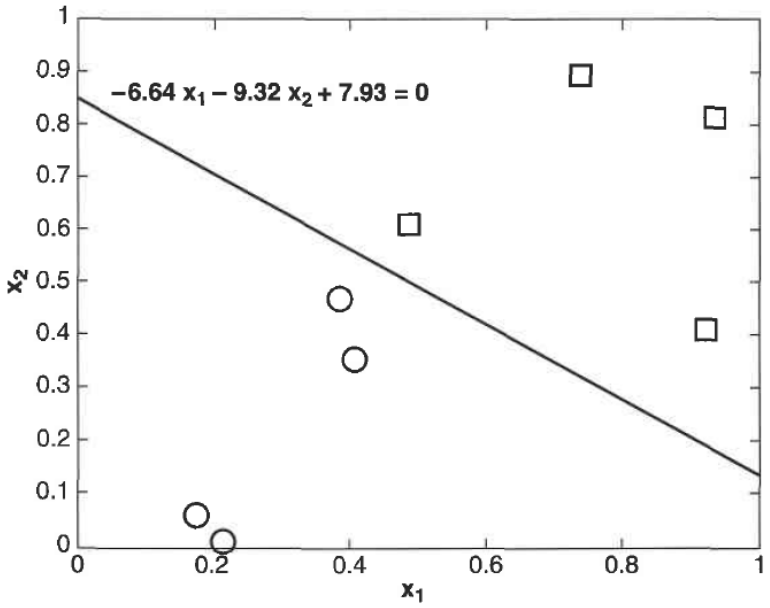
$$w_2 = \sum_i \lambda_i y_i x_{i2} = 65.5621 \times 1 \times 0.4687 + 65.5621 \times -1 \times 0.611 = -9.32$$

Ejemplo

$$b^{(1)} = 1 - \mathbf{w} \cdot \mathbf{x}_1 = 1 - (-6.64)(0.3858) - (-9.32)(0.4687) = 7.9300$$

$$b^{(2)} = -1 - \mathbf{w} \cdot \mathbf{x}_2 = -1 - (-6.64)(0.4871) - (-9.32)(0.611) = 7.9289$$

Promediando los $b's$ se tiene que $b = 7.92945$ luego redondeando $b = 7.93$.



Ejemplo

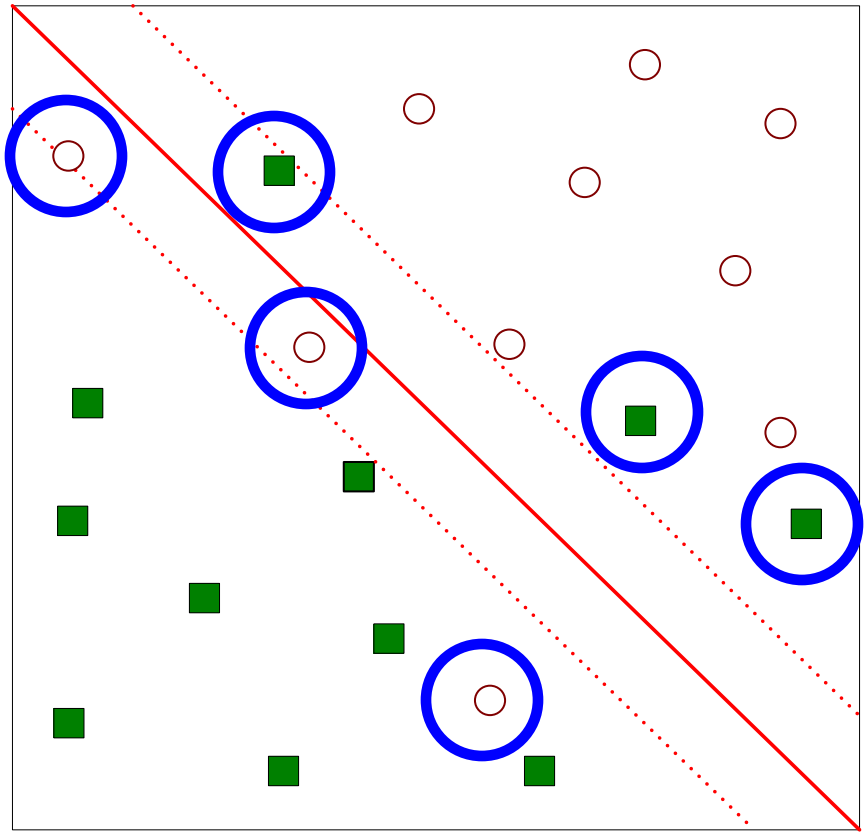
Para este ejemplo dado un nuevo individuo z se clasifica según la siguiente función de decisión:

$$f(z) = sign(w \cdot z + b) = sign\left(\sum_{i=1}^{n} \lambda_i y_i x_i \cdot z + b\right)$$

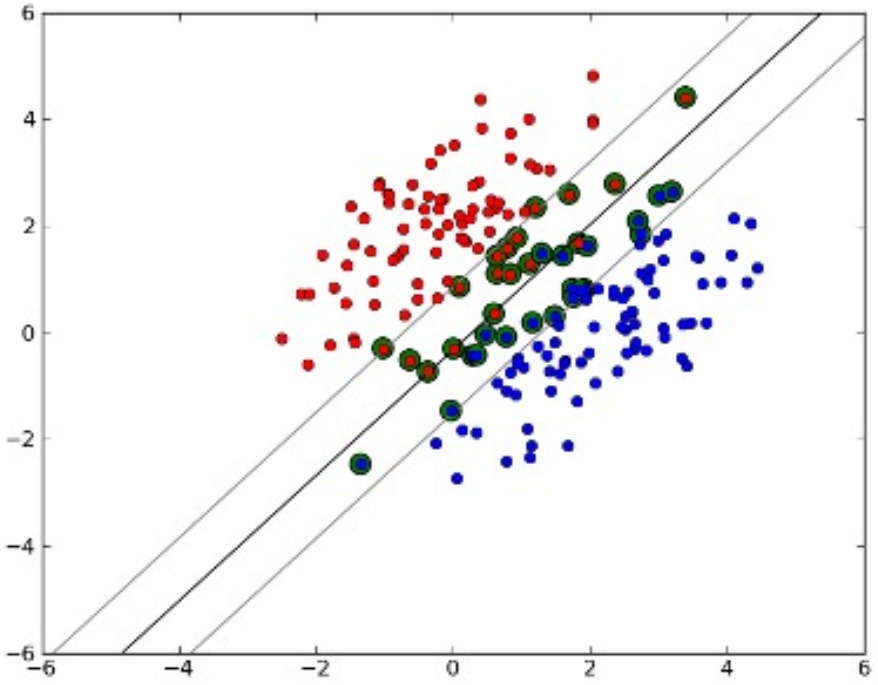
Si el $f(z) = +$ entonces w se clasifica como rectángulo y en caso contrario si $f(z) = -$ entonces se clasifica como círculo.

Máquinas de Soporte Vectorial

- ¿Qué pasa si el problema no es linealmente separable?



Margen de soporte débil (más vectores de soporte)



- El número de vectores de soporte entra a jugar cuando los datos no son linealmente separable, o sea, no existe un hiperplano que separe las 2 clases en los datos.
- En este caso el método trata entonces de encontrar un ***margen débil***, lo cual quiere decir que permite que algunos puntos de ambas clases queden dentro del margen, esto se logra permitiendo más vectores de soporte lo cual hace que el error aumente.
- Esto se hace porque de contrario entonces no existiría el plano de separación, claro el precio es un mayor error.

Máquinas Vectoriales de Soporte

- ¿Qué pasa si el problema no es linealmente separable?
  - Se introducen variables de holgura (slack variables)
    - El problema a minimizar es (donde *C* y *k* son parámetros definidos por el usuario para la penalización):

$$L(w) = \frac{\| w \|^2}{2} + C\left(\sum_{i=1}^{n} \xi_i \right)^k$$

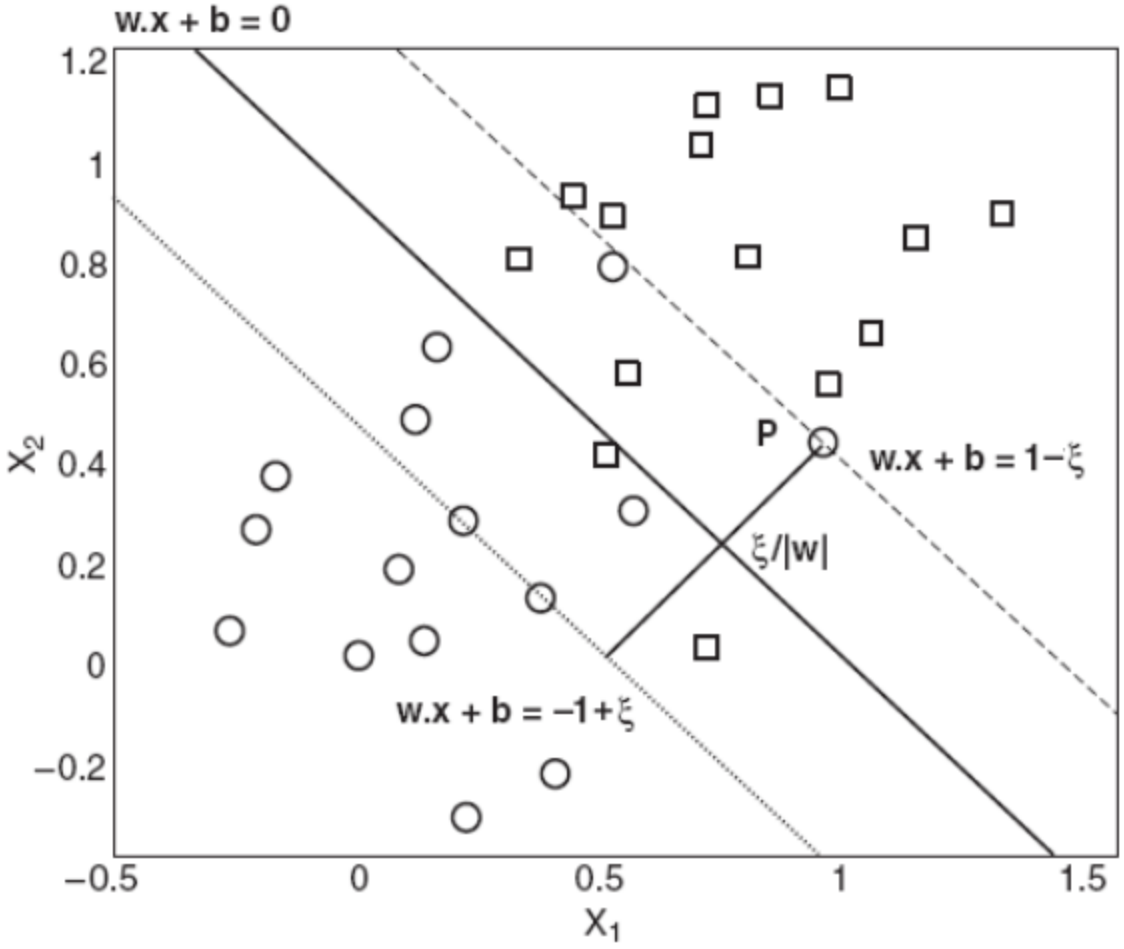
    - Sujeto a:

$$f(x_i) = \begin{cases} 1 & \text{if } w \cdot x_i + b \geq 1 - \xi_i \\ -1 & \text{if } w \cdot x_i + b \leq -1 + \xi_i \end{cases}$$

Parámetro de regularización *C*

- El parámetro de Regularización (a menudo denominado como parámetro *C* en la biblioteca sklearn de Python) le dice a la optimización de SVM cuánto quiere evitar clasificar erróneamente cada caso de entrenamiento.
- Para valores grandes de *C*, la optimización elegirá un hiperplano de margen más pequeño si ese hiperplano hace un mejor trabajo al clasificar correctamente todos los puntos de entrenamiento.
- Por el contrario, un valor muy pequeño de *C* hará que el optimizador busque un hiperplano de separación de mayor margen, incluso si ese hiperplano clasifica incorrectamente más puntos.

Máquinas de Soporte Vectorial



Máquinas de Soporte Vectorial

- En este caso, con $k = 1$, se tiene el siguiente problema de optimización L_P:

$$L_P = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\lambda_i\left\{y_i\left(\mathbf{w}\cdot\mathbf{x_i}+b\right)-1+\xi_i\right\} - \sum_{i=1}^{N}\mu_i\xi_i$$

- Las derivadas parciales son (de aquí se puede calcular w):

$$\frac{\partial L}{\partial w_j} = w_j - \sum_{i=1}^{N}\lambda_i y_i x_{ij} = 0 \Longrightarrow w_j = \sum_{i=1}^{N}\lambda_i y_i x_{ij}$$

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^{N}\lambda_i y_i = 0 \Longrightarrow \sum_{i=1}^{N}\lambda_i y_i = 0$$

$$\frac{\partial L}{\partial \xi_i} = C - \lambda_i - \mu_i = 0 \Longrightarrow \lambda_i + \mu_i = C$$

Máquinas Vectoriales de Soporte

- Ecuaciones de Karush-Kuhn-Tuker para calcular b:

$$\xi_i \geq 0, \quad \lambda_i \geq 0, \quad \mu_i \geq 0$$

$$\lambda_i \left\{ y_i \left(\mathbf{w} \cdot \mathbf{x}_i + b \right) - 1 + \xi_i \right\} = 0$$

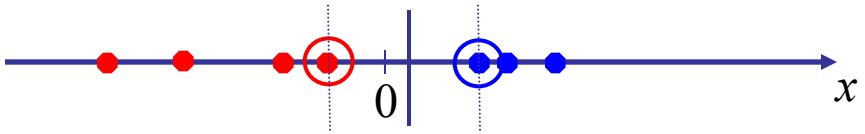
$$\mu_i \xi_i = 0$$

- En este caso problema de optimización dual es L_D (de aquí se pueden calcular los $\lambda's$):

$$\begin{aligned} L_D = & \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j + C \sum_i \xi_i \\ & - \sum_i \lambda_i \left\{ y_i \left(\sum_j \lambda_j y_j \mathbf{x}_i \cdot \mathbf{x}_j + b \right) - 1 + \xi_i \right\} \\ & - \sum_i \left(C - \lambda_i \right) \xi_i \\ = & \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \mathbf{x_i} \cdot \mathbf{x_j} \end{aligned}$$

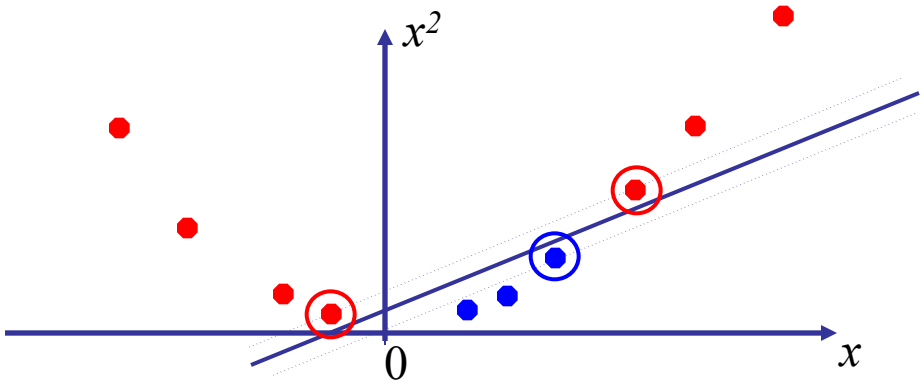
MVS no linealmente separables

- Datos linealmente separables:



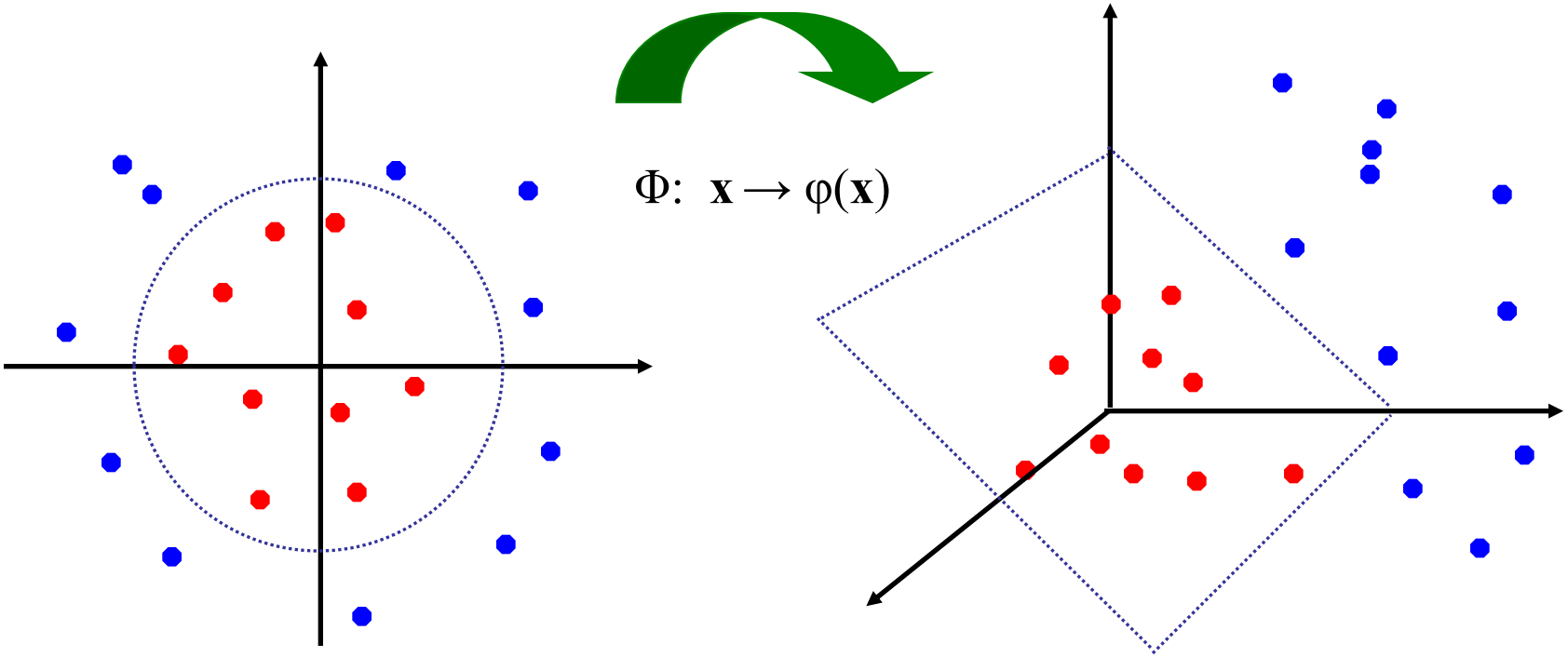
- Datos no linealmente separables:

- La idea es… Encontrar una función para trasladar los datos a un espacio de mayor dimensión:



MVS no linealmente separables

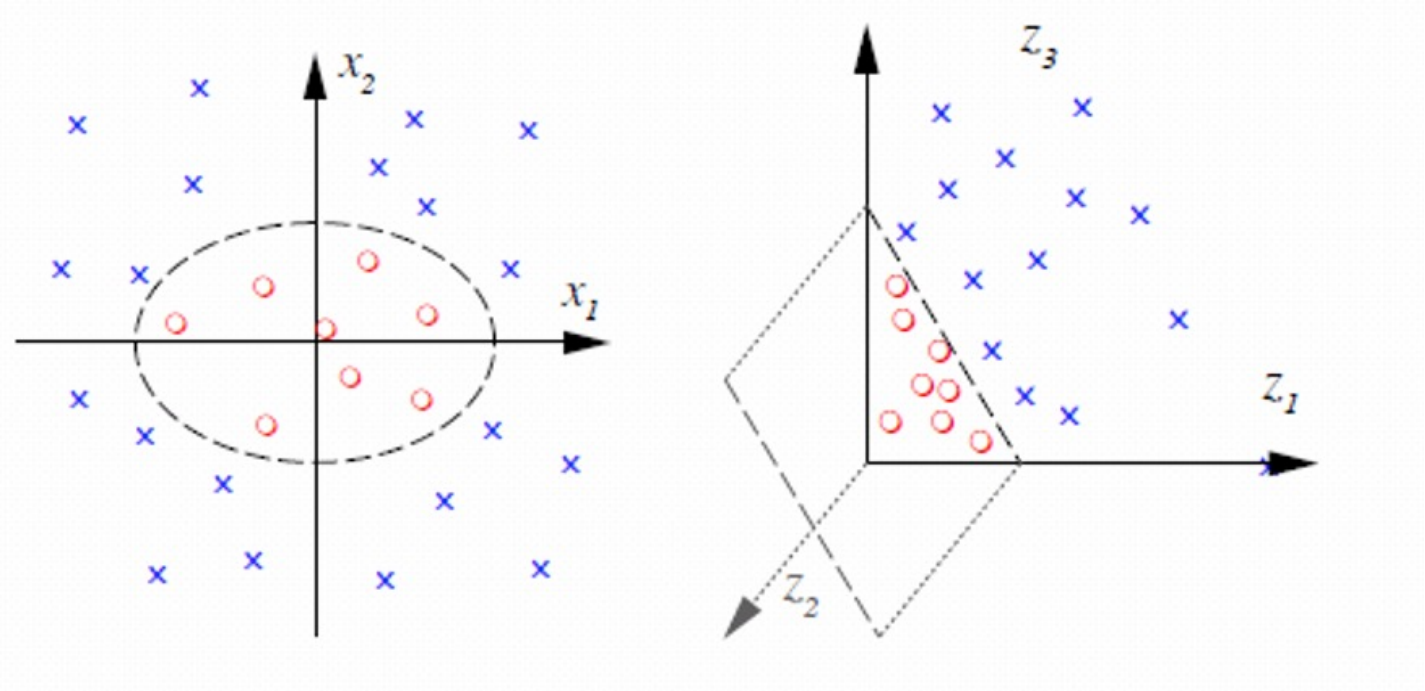
- Idea general: Los datos de entrada se puede trasladar a algún espacio de mayor dimensión en el que la Tabla de Entrenamiento sí sea separable:



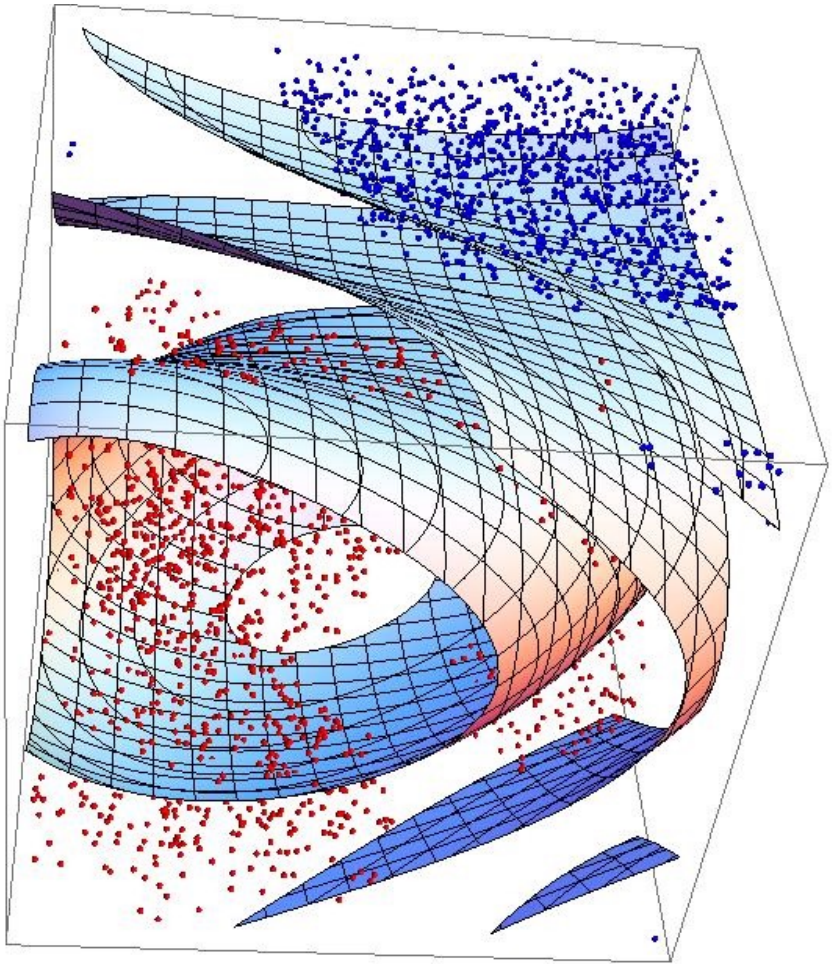
MVS no linealmente separables

$$\Phi : I\!R^2 \to I\!R^3$$

$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right)$$



El Truco del Núcleo (Kernel Trick)

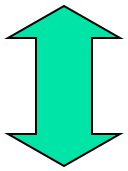


Resolver un Problema Optimización

Un problema de programación cuadrática con restricciones lineales

$$\min_{w} \frac{\| w \|^2}{2}$$

Sujeto a: $y_i(w \cdot \Phi(x_i) + b) \geq 1$ para $i = 1,2, \ldots, n$



Minimización de Lagrange

$$L_P(w, b, \lambda_i) = \frac{\| w \|^2}{2} - \sum_{i=1}^{n} \lambda_i \left(y_i(w \cdot \Phi(x_i) + b) - 1\right)$$

con $\lambda_i \geq 0$ (los λ se llaman multiplicadores de Lagrange)

- En este caso se tiene el siguiente problema de optimización L_D:

$$L_D = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \Phi(x_i) \cdot \Phi(x_j)$$

- w se calcula como sigue: $w = \sum_{i=1}^{n} \lambda_i y_i \Phi(x_i)$

- b se calcula de la siguiente igualdad:

$$\lambda_i [y_i \left(\sum_j \lambda_j y_j \Phi(x_j) \cdot \Phi(x_i) + b \right) - 1] = 0$$

- La función de clasificación es:

$$f(z) = sign(w \cdot \Phi(z) + b) = sign\left(\sum_{i=1}^{n} \lambda_i y_i \Phi(x_i) \cdot \Phi(z) + b \right)$$

El Truco del Núcleo (Kernel Trick)

- El producto punto (producto interno) entre dos vectores es a menudo considerado como una medida de similitud.

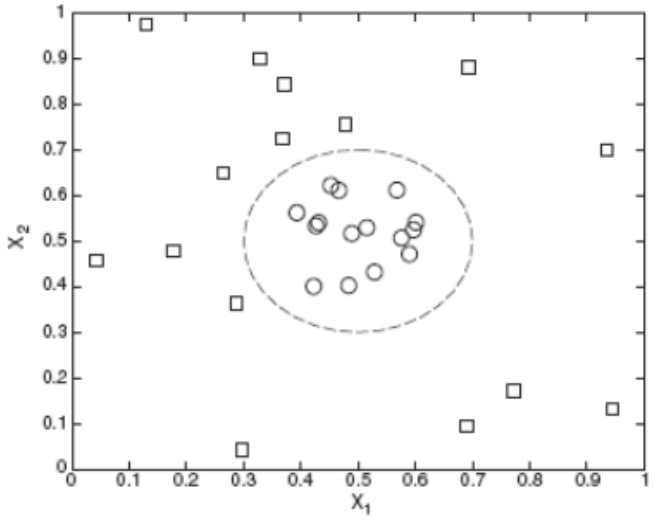
- Por ejemplo $u = (x_1, x_2, x_3)$ y $v = (y_1, y_2, y_3)$:

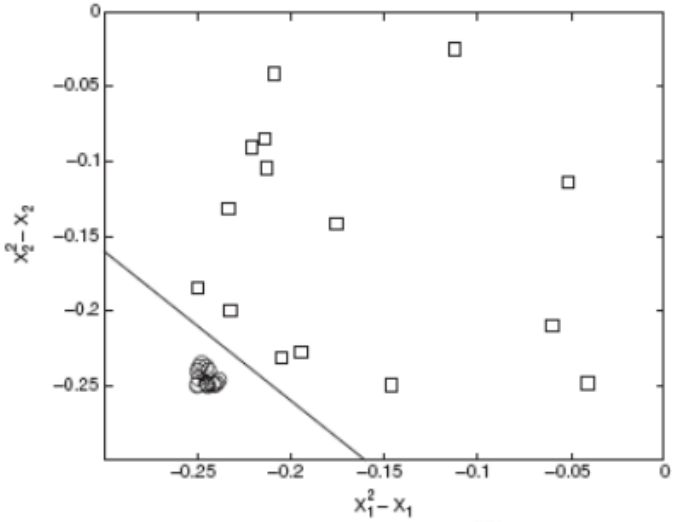
$$d(u,v) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_3 - y_3)^2}$$
$$= \sqrt{(x_1 - y_1, x_2 - y_2, x_3 - y_3) \cdot (x_1 - y_1, x_2 - y_2, x_3 - y_3)}$$
$$= \sqrt{u \cdot v}$$

El Truco del Núcleo (Kernel Trick)

- La funciones de transformación del espacio vectorial pueden ser vistas como un producto punto.

- Ejemplo:





$$\Phi : (x_1, x_2) \longrightarrow \left(x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, 1\right)$$

El Truco del Núcleo (Kernel Trick)

$$\begin{aligned}\Phi(\mathbf{u}) \cdot \Phi(\mathbf{v}) &= \left(u_1^2, u_2^2, \sqrt{2}u_1, \sqrt{2}u_2, 1\right) \cdot \left(v_1^2, v_2^2, \sqrt{2}v_1, \sqrt{2}v_2, 1\right) \\ &= u_1^2 v_1^2 + u_2^2 v_2^2 + 2u_1 v_1 + 2u_2 v_2 + 1 \\ &= (\mathbf{u} \cdot \mathbf{v} + 1)^2\end{aligned}$$

- Esto permite que el cálculo del producto punto en el espacio vectorial transformado pueda ser calculado en el espacio vectorial original.
- Esta función de similaridad, K, que puede ser calcula en el espacio vectorial original, se llama la **Función Núcleo** (Kernel Function)

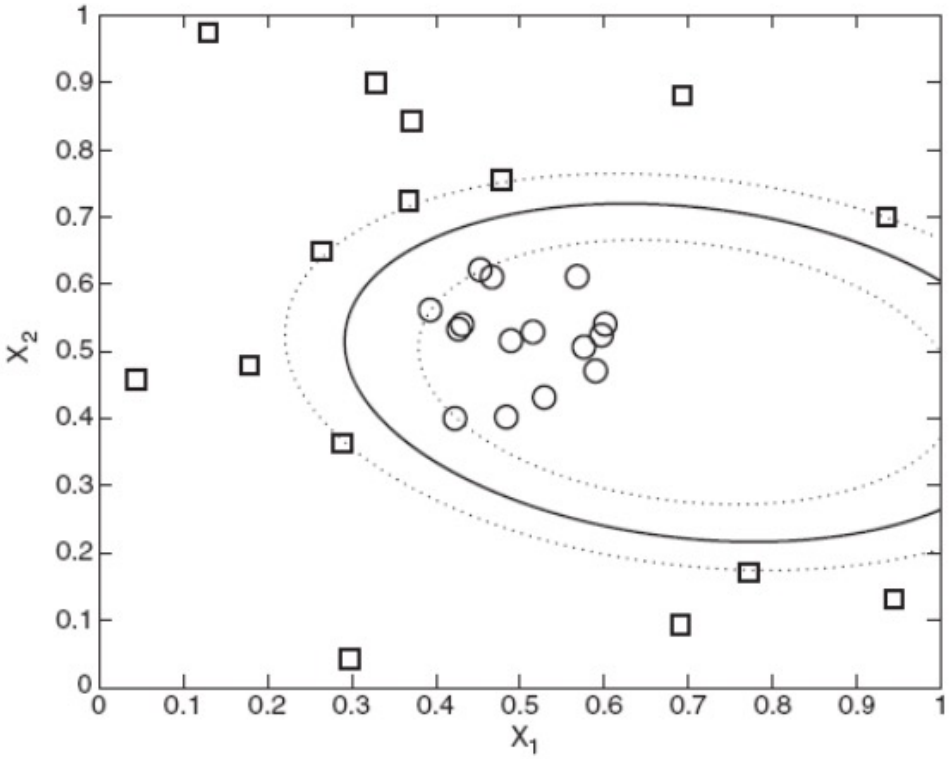
$$K(\mathbf{u}, \mathbf{v}) = \Phi(\mathbf{u}) \cdot \Phi(\mathbf{v}) = (\mathbf{u} \cdot \mathbf{v} + 1)^2$$

El Truco del Núcleo (Kernel Trick)

- El truco kernel ayuda a abordar algunos problemas sobre cómo implementar SVM en el caso no linealmente separable.
  1. No tenemos que saber la forma exacta de la función de transformación Φ debido a que las funciones de núcleo utilizadas deben satisfacer un principio matemático conocido como ***Teorema de Mercer***. Este principio asegura que las funciones de núcleo siempre se pueden expresar como el producto escalar entre dos vectores de entrada.
  2. El cálculo de los productos escalares utilizando las funciones del núcleo es considerablemente más rápido que utilizar el atributo transformado $\Phi(x)$.
  3. Puesto que los cálculos se realizan en el espacio vectorial original, los problemas asociados al aumento de la dimensión se pueden evitar.

El Truco del Núcleo (Kernel Trick)

Para el ejemplo anterior usando el Truco del Núcleo la frontera de decisión se puede “ver” en el espacio original:



El Truco del Núcleo (Kernel Trick)

Para el ejemplo anterior para clasificar un nuevo individuo o caso se puede hacer con las siguientes ecuaciones:

$$
\begin{aligned}
f(\mathbf{z}) &= \operatorname{sign}\left(\sum_{i=1}^{n} \lambda_i y_i \Phi\left(\mathbf{x}_i\right) \cdot \Phi(\mathbf{z}) + b\right) \\
&= \operatorname{sign}\left(\sum_{i=1}^{n} \lambda_i y_i K\left(\mathbf{x}_i, \mathbf{z}\right) + b\right) \\
&= \operatorname{sign}\left(\sum_{i=1}^{n} \lambda_i y_i \left(\mathbf{x}_i \cdot \mathbf{z} + 1\right)^2 + b\right)
\end{aligned}
$$

El Truco del Núcleo (Kernel Trick)

Teorema de Mercer: Una función núcleo K puede ser expresada como:

$$K(u, v) = \Phi(u) \cdot \Phi(v)$$

si y solamente si, para toda función $g(x)$ tal que $\int g(x)^2 dx$ es finito, se tiene que:

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

El Truco del Núcleo (Kernel Trick)

Ejemplos: Algunas funciones núcleo K usadas son:

$$K(x, y) = (x \cdot y + 1)^p$$

$$K(x, y) = e^{-\|x-y\|^2/(2\sigma^2)}$$

$$K(x, y) = \tanh(kx \cdot y - \delta)$$

Máquinas Vectoriales de Soporte en Python

sklearn.svm.SVC

class `sklearn.svm.`**SVC**(**, C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', break_ties=False, random_state=None*) [source]

C-Support Vector Classification.

Ejemplo 1: IRIS.CSV

Ejemplo con la tabla de datos IRIS
IRIS Información de variables:
1.sepal largo en cm
2.sepal ancho en cm
3.petal largo en cm
4.petal ancho en cm
5.clase:

- Iris Setosa
- Iris Versicolor
- Iris Virginica

	A	B	C	D	E
1	s.largo	s.ancho	p.largo	p.ancho	tipo
2	5.1	3.5	1.4	0.2	setosa
3	4.9	3.0	1.4	0.2	setosa
4	4.7	3.2	1.3	0.2	setosa
5	4.6	3.1	1.5	0.2	setosa
6	5.0	3.6	1.4	0.2	setosa
7	5.4	3.9	1.7	0.4	setosa
8	4.6	3.4	1.4	0.3	setosa
9	5.0	3.4	1.5	0.2	setosa
10	4.4	2.9	1.4	0.2	setosa
11	4.9	3.1	1.5	0.1	setosa
12	5.4	3.7	1.5	0.2	setosa
13	4.8	3.4	1.6	0.2	setosa
14	4.8	3.0	1.4	0.1	setosa
15	4.3	3.0	1.1	0.1	setosa
16	5.8	4.0	1.2	0.2	setosa
17	5.7	4.4	1.5	0.4	setosa
18	5.4	3.9	1.3	0.4	setosa
19	5.1	3.5	1.4	0.3	setosa
20	5.7	3.8	1.7	0.3	setosa
21	5.1	3.8	1.5	0.3	setosa
22	5.4	3.4	1.7	0.2	setosa
23	5.1	3.7	1.5	0.4	setosa
24	4.6	3.6	1.0	0.2	setosa
25	....	...	...	...	....

Ejemplo 2: Credit-Scoring

MuestraAprendizajeCredito2500.csv
MuestraTestCredito2500.csv

```
> setwd("C:/Users/Oldemar/Google Drive/Curso Mineria Datos II - Optativo/Datos")
> taprendizaje<-read.csv("MuestraAprendizajeCredito2500.csv",sep = ";",header=T)
> taprendizaje
```

	MontoCredito	IngresoNeto	CoefCreditoAvaluo	MontoCuota	GradoAcademico	BuenPagador
1	1	1	1	1	1	Si
2	3	1	1	1	1	Si
3	2	1	1	1	1	Si
4	1	2	1	1	1	Si
5	1	1	1	1	1	Si
6	2	1	1	1	1	Si
7	4	1	1	1	1	Si
8	1	2	1	1	1	Si
9	1	2	1	1	1	Si
10	3	2	1	1	1	Si
11	1	1	1	1	1	Si
12	1	2	1	1	1	Si
13	3	1	1	1	1	Si
14	3	1	1	1	1	Si
15	2	1	1	1	1	Si
16	3	1	1	1	1	Si
17	3	1	1	1	1	Si

Descripción de Variables

MontoCredito

1=Muy Bajo
2=Bajo
3=Medio
4=Alto

MontoCuota

1=Muy Bajo
2=Bajo
3=Medio
4=Alto

IngresoNeto

1=Muy Bajo
2=Bajo
3=Medio
4=Alto

GradoAcademico

1=Bachiller
2=Licenciatura
3=Maestría
4=Doctorado

CoeficienteCreditoAvaluo

1=Muy Bajo
2=Bajo
3=Medio
4=Alto

BuenPagador

1=NO
2=Si

oldemar rodríguez

CONSULTOR en M1NƐR14 DƐ D4T0S

Gracias….