

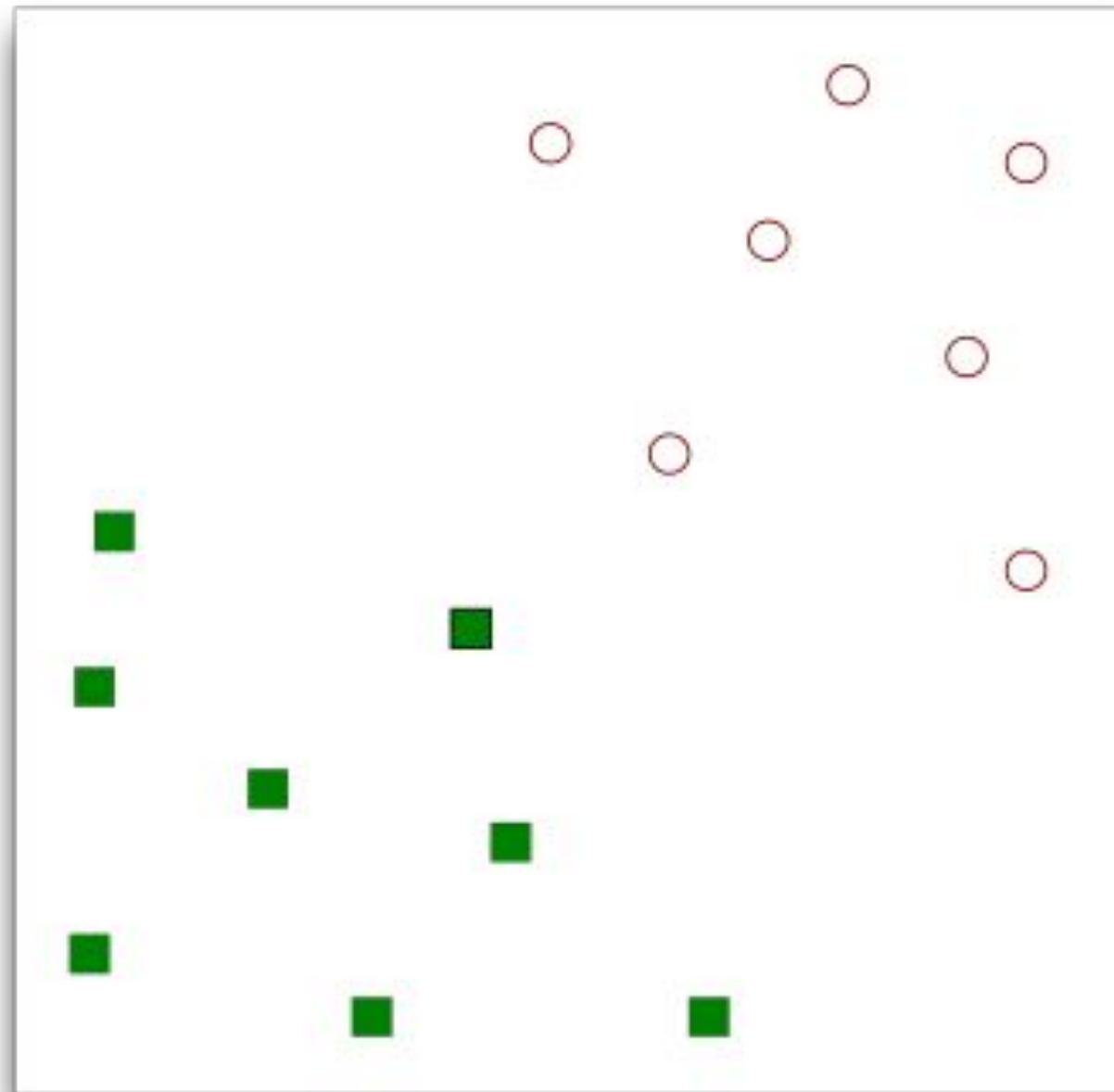


Clasificación

Support Vector Machines

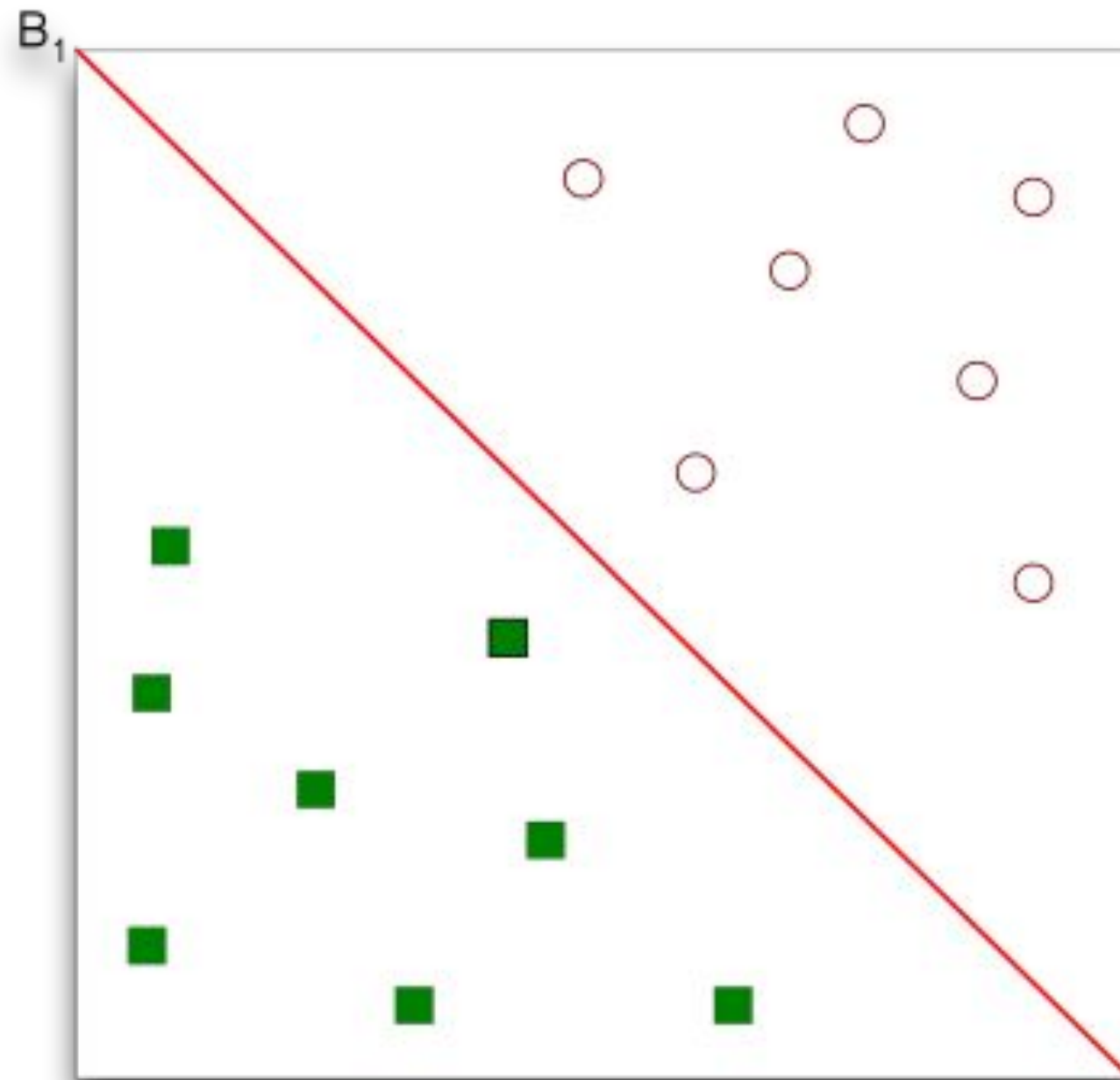
Felipe Bravo

Support Vector Machines



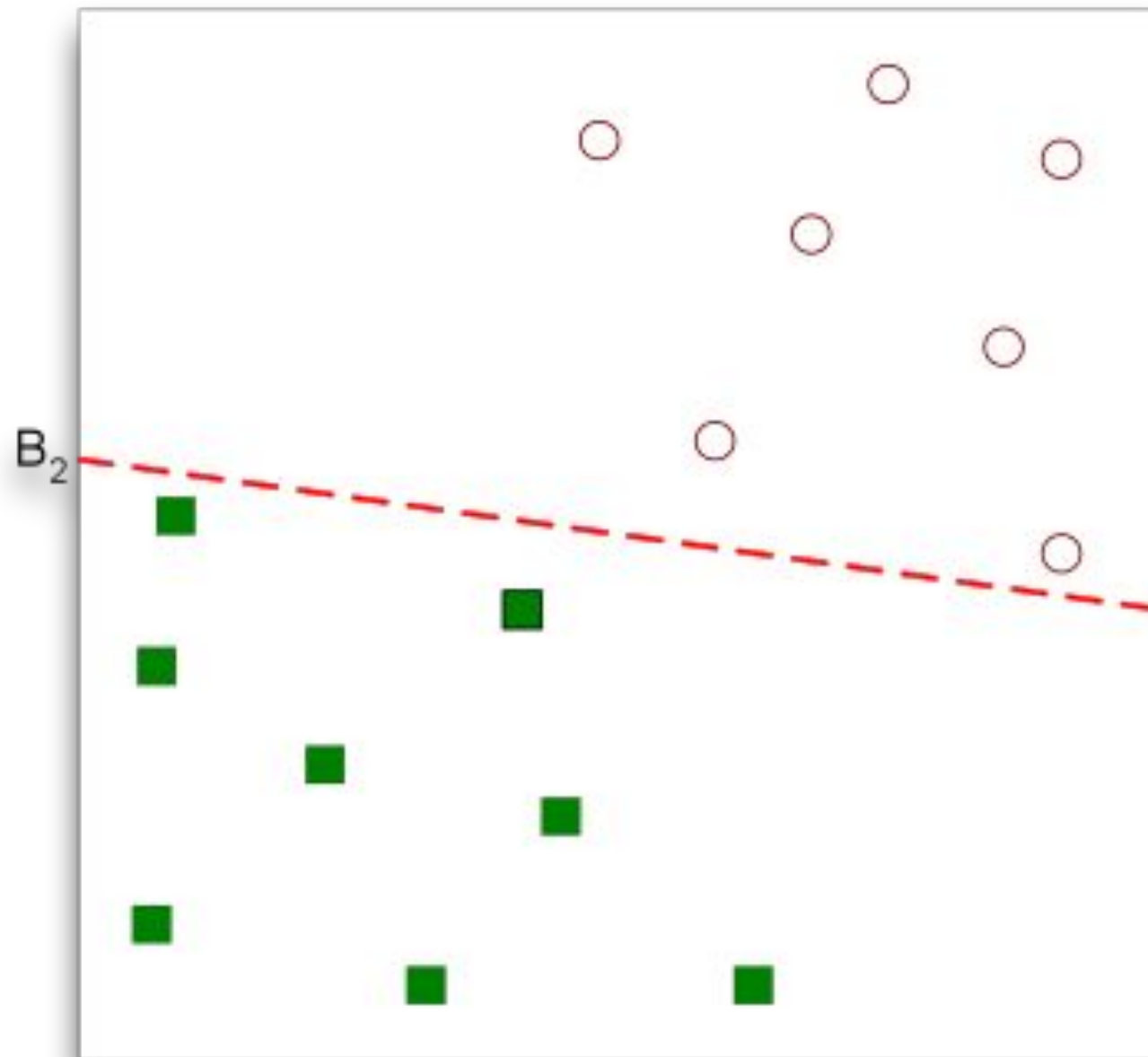
- Encontrar un hiperplano lineal (decision boundary) que separe los ejemplos positivos de los negativos.

Support Vector Machines



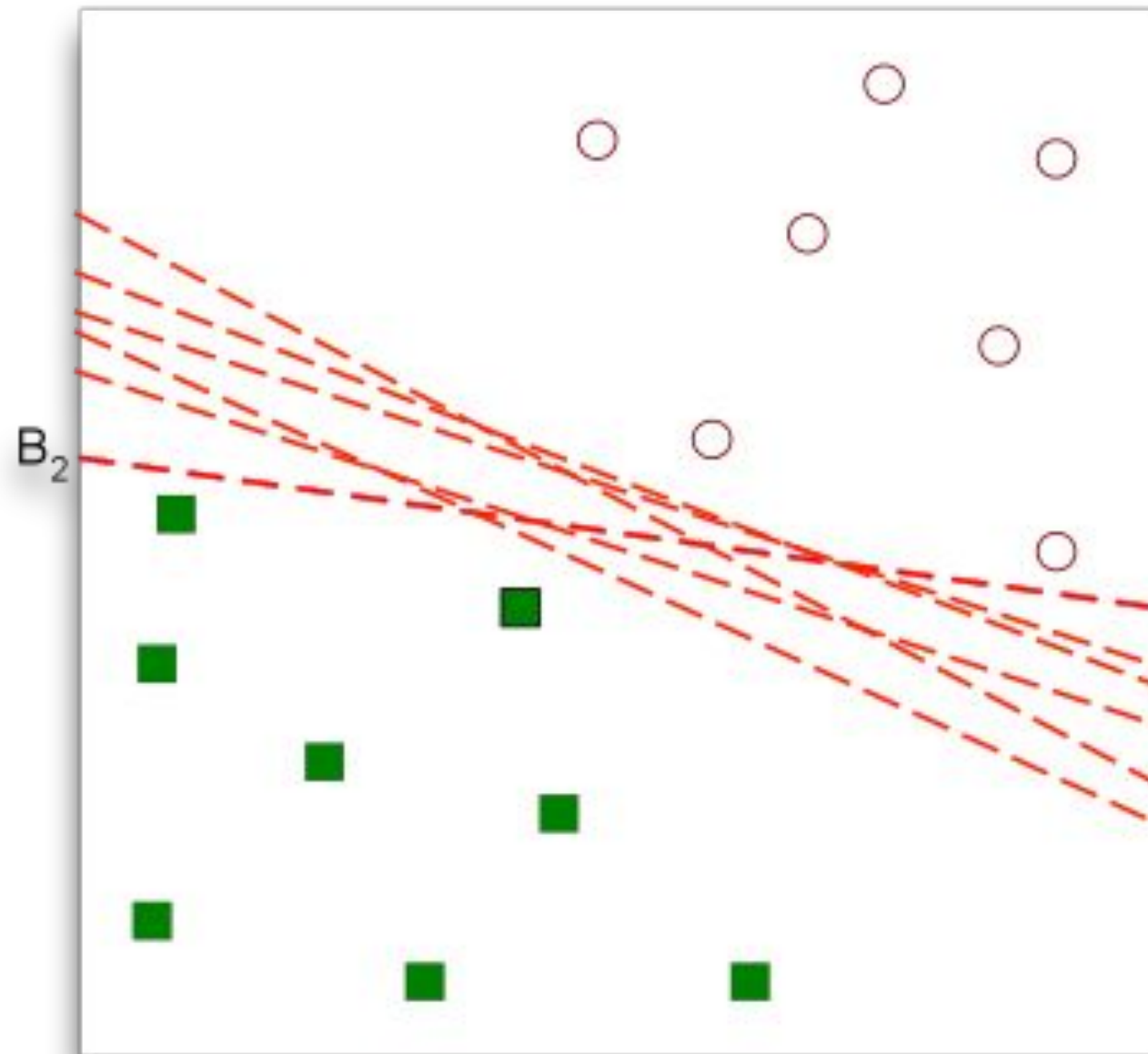
- Una posible solución

Support Vector Machines



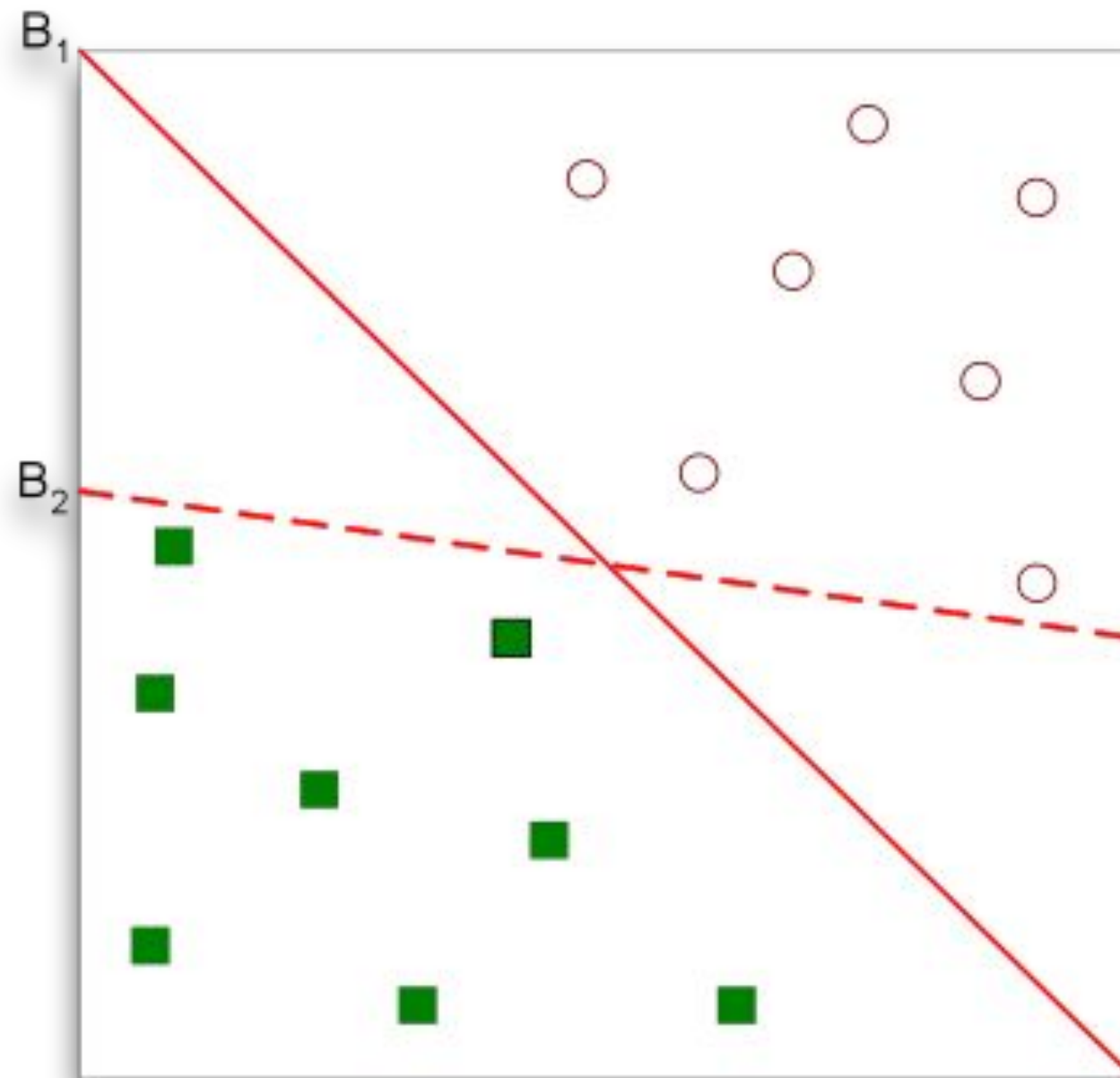
- Otra solución posible

Support Vector Machines



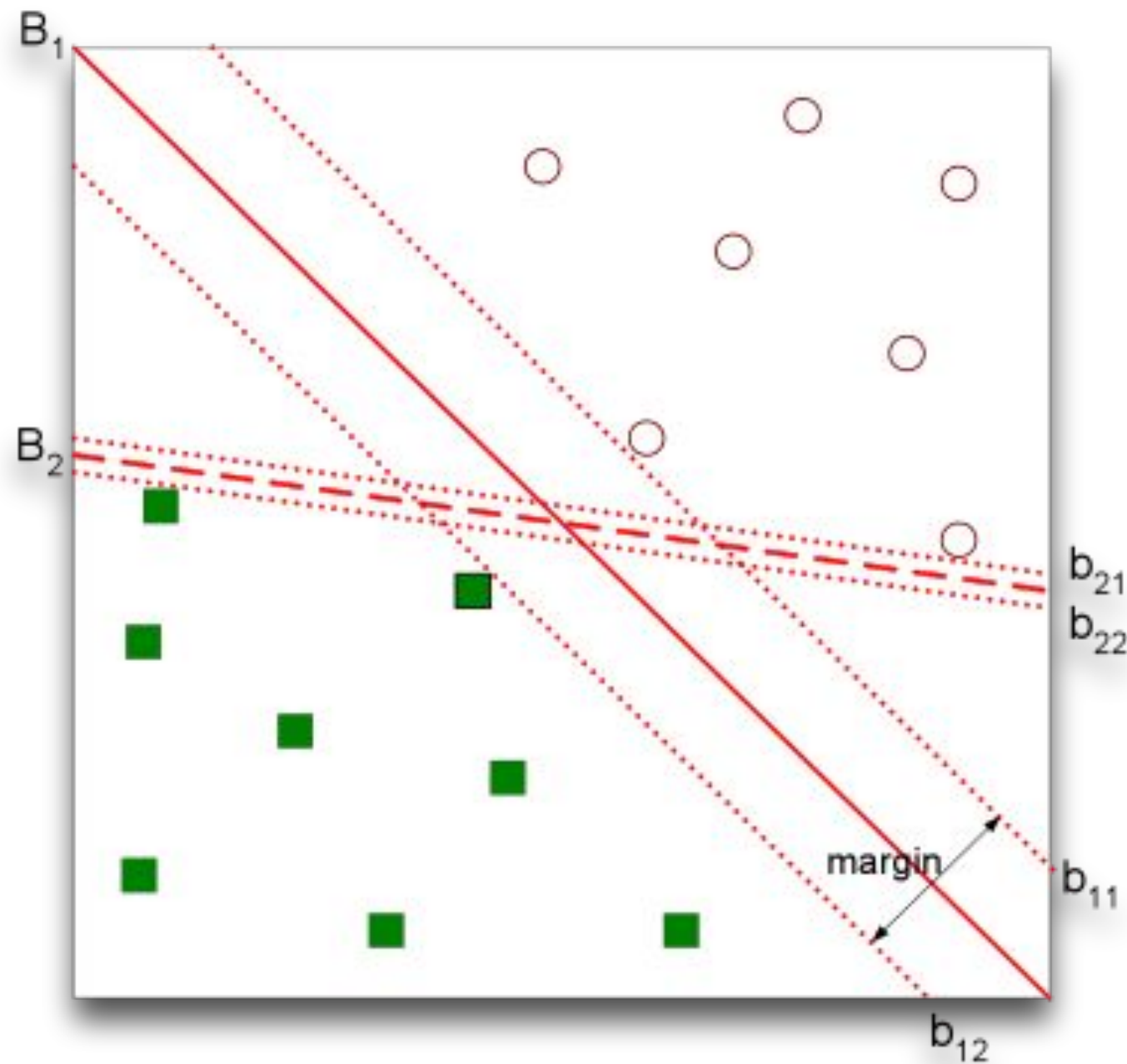
- ¡Infinitas soluciones!

Support Vector Machines



- ¿Cuál es mejor? ¿ B_1 o B_2 ?
- ¿Cómo definimos qué es mejor?

Support Vector Machines



- Encontrar un hiperplano que **maximice** el margen de entrenamiento (menores errores de generalización) \Rightarrow B_1 es mejor que B_2
- Entre más ancho el margen, mayor el poder de generalización.

Support Vector Machines

- Un SVM lineal es un clasificador que busca un hiperplano con el máximo margen.
- Sea un problema de clasificación binaria con N ejemplos, cada ejemplo se representa como la tupa $\mathbf{x}_i, \mathbf{y}_i$ ($i=1,2,\dots,N$), donde \mathbf{x}_i es un vector de \mathbf{d} dimensiones $(x_{i1}, x_{i2}, \dots, x_{id})^T$ e $y_i \in \{-1, 1\}$ (ejemplos negativos y positivos).
- El límite de decisión de un clasificador lineal se escribe de la siguiente manera:

$$\mathbf{w} \cdot \mathbf{x} + b = 0,$$

- Donde \mathbf{w} y b son parámetros del modelo.
- En las siguientes slides vamos a ver cómo formular el margen en función de los parámetros y luego formularemos un **problema de optimización** que permita encontrar el hiperplano de máximo margen.

Support Vector Machines

- El hiperplano $\mathbf{w} \cdot \mathbf{x} + b = 0$ separa los ejemplos positivos (cuadrados) de los negativos (círculos)
- Cualquier ejemplo que se encuentre en el límite de decisión debe satisfacer la ecuación del hiperplano.
- Por ejemplo, sean \mathbf{x}_a y \mathbf{x}_b dos puntos localizados en el límite de decisión:

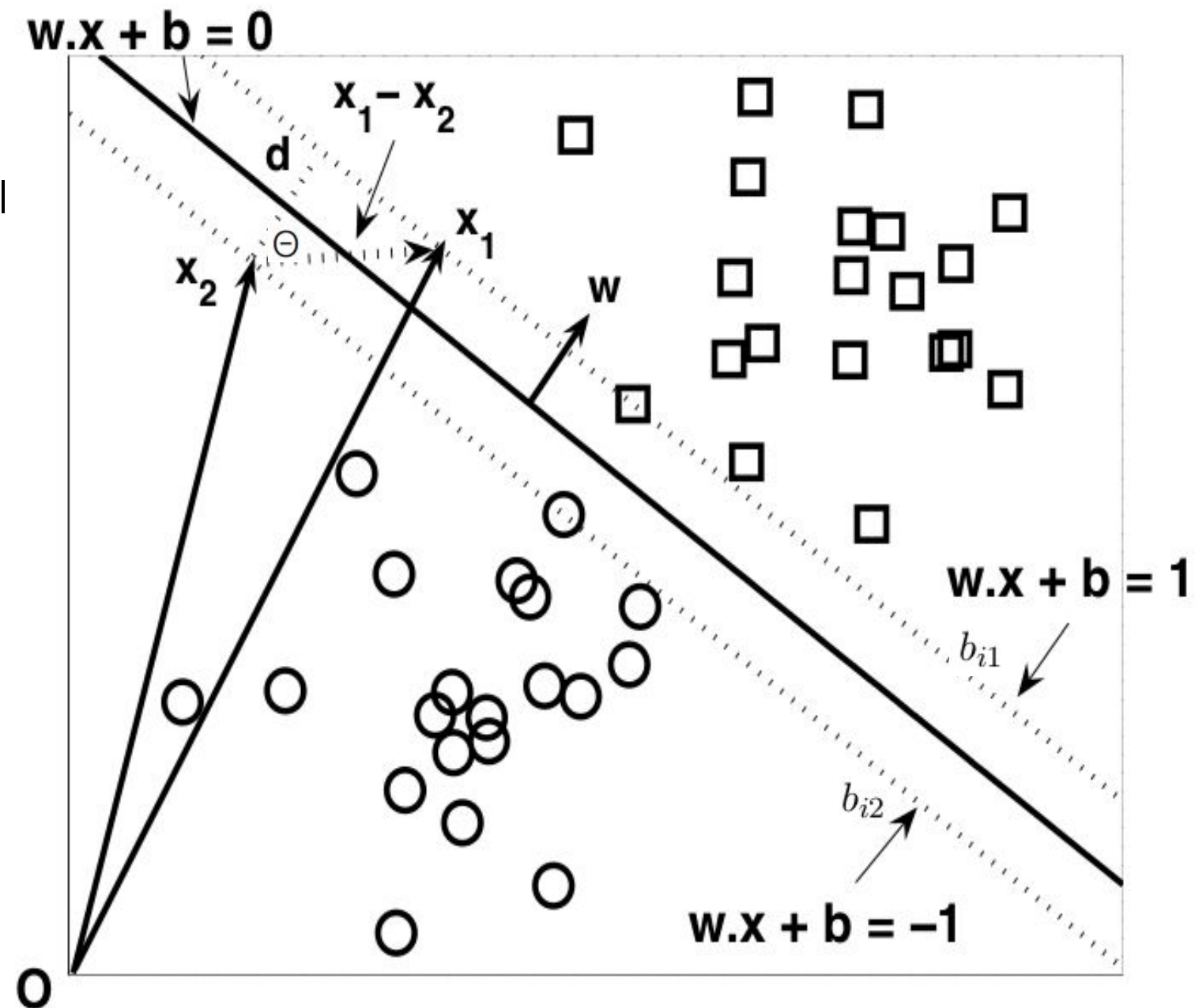
$$\mathbf{w} \cdot \mathbf{x}_a + b = 0,$$

$$\mathbf{w} \cdot \mathbf{x}_b + b = 0.$$

- Si restamos las dos ecuaciones obtenemos:

$$\mathbf{w} \cdot (\mathbf{x}_b - \mathbf{x}_a) = 0,$$

- Cómo $\mathbf{x}_b - \mathbf{x}_a$ es paralelo al límite de decisión, entonces \mathbf{w} es perpendicular al hiperplano (el producto punto de dos vectores ortogonales es cero).



Support Vector Machines

- Para cualquier cuadrado \mathbf{x}_s localizado sobre el límite de decisión, se cumple que:

$$\mathbf{w} \cdot \mathbf{x}_s + b = k,$$

donde $k > 0$.

- De manera análoga, para cualquier círculo \mathbf{x}_c localizado bajo el límite de decisión, se cumple que:

$$\mathbf{w} \cdot \mathbf{x}_c + b = k',$$

done $k' < 0$.

- Si etiquetamos todos los cuadrados como la clase positiva +1 y todos los círculos como la clase negativa -1, podemos predecir la etiqueta \mathbf{y} para cualquier ejemplo de test \mathbf{z} de la siguiente manera:

$$y = \begin{cases} 1, & \text{if } \mathbf{w} \cdot \mathbf{z} + b > 0; \\ -1, & \text{if } \mathbf{w} \cdot \mathbf{z} + b < 0. \end{cases}$$

El Margen de un Clasificador Lineal

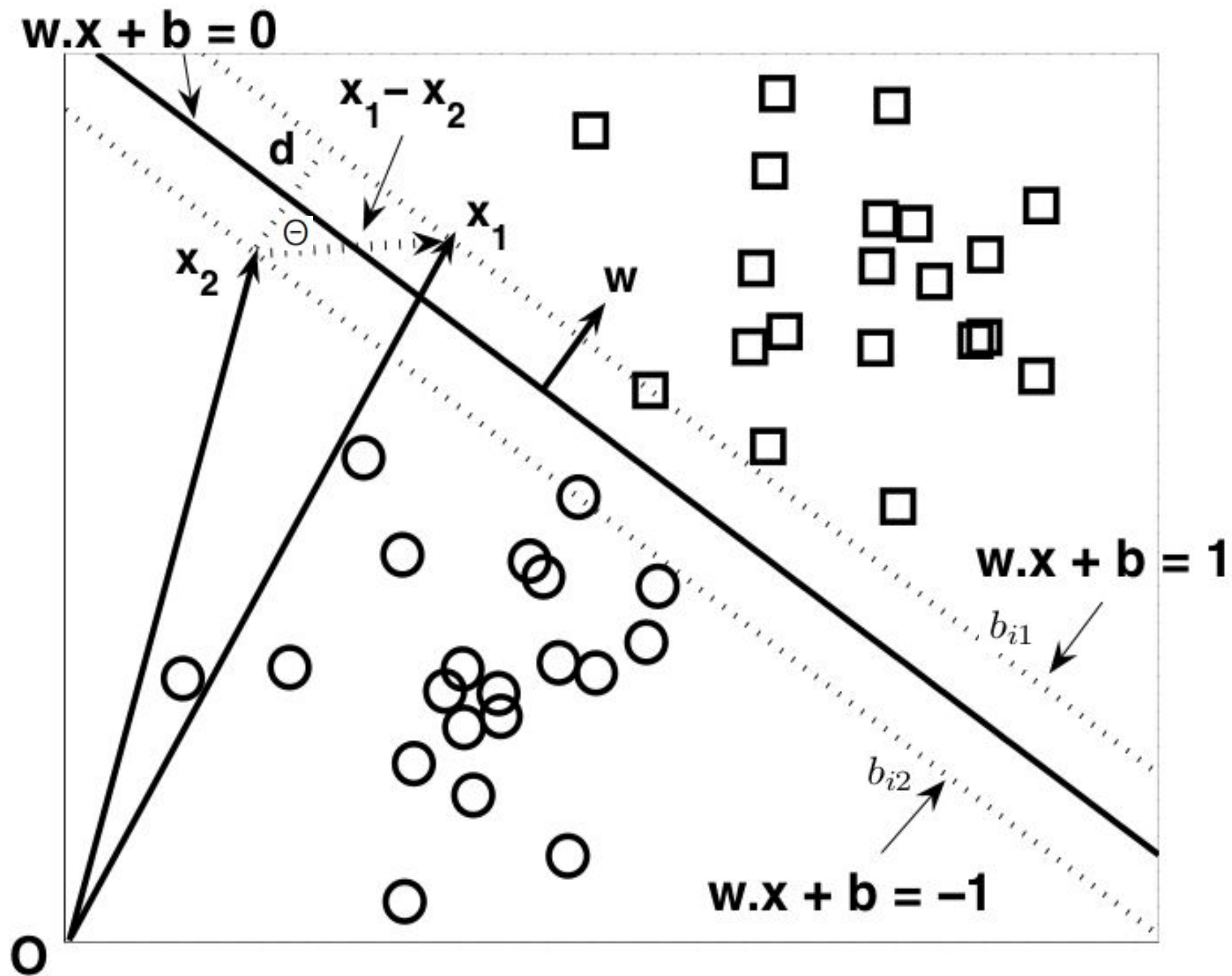
- Consideremos el cuadrado y el círculo que se encuentran más cercanos al límite de decisión como los puntos x_1 y x_2 .
- El cuadrado debe satisfacer $w \cdot x_1 + b = k$ (para algún k positivo) y el círculo debe satisfacer $w \cdot x_2 + b = -k$ (para algún k negativo).
- Sobre x_1 pasa un hiperplano paralelo al límite de decisión llamado b_{i1} y sobre x_2 pasa otro hiperplano paralelo al límite de decisión llamado b_{i2} .
- Podemos re-escalar los parámetros w y b del límite de decisión de tal manera que los hiperplanos paralelos b_{i1} y b_{i2} puedan expresarse de la siguiente manera:

$$b_{i1} : \mathbf{w} \cdot \mathbf{x} + b = 1,$$

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

- El margen del límite de decisión se calcula como la distancia entre estos dos hiperplanos.

El Margen de un Clasificador Lineal



El Margen de un Clasificador Lineal

- Para calcular el margen calculamos la distancia del círculo x_1 y el cuadrado x_2 .
- Esto se hace sustituyendo x_1 y x_2 en las ecuaciones de b_{i1} y b_{i2} respectivamente.
- Lo que nos da: 1) $w \cdot x_1 + b = 1$ y 2) $w \cdot x_2 + b = -1$.
- Si sustraemos la segunda ecuación de la primera obtenemos $w \cdot (x_1 - x_2) = 2$ (el producto punto es distributivo).

El Margen de un Clasificador Lineal

- El producto punto de dos vectores ($a \cdot b$) se puede representar como $\|a\| \cdot \|b\| \cos(\Theta)$.
- Tenemos entonces que $\|w\| \cdot \|x_1 - x_2\| \cos(\Theta) = 2$
- Si miramos la figura anterior podemos ver que $\|x_1 - x_2\| \cos(\Theta) = d$.
- Entonces

$$\|w\| \times d = 2$$
$$\therefore d = \frac{2}{\|w\|}.$$

¡ Encontramos una expresión del margen que depende de w !

Apreniendo una SVM Lineal

- La fase de entrenamiento de una SVM lineal implica la estimación de los parámetros w y b del límite de decisión a partir de los datos de entrenamiento.
- Los parámetros deben elegirse de manera que se cumplan las dos siguientes condiciones :

$$\begin{aligned}w \cdot x_i + b &\geq 1 \text{ if } y_i = 1, \\w \cdot x_i + b &\leq -1 \text{ if } y_i = -1.\end{aligned}$$

- Estas condiciones imponen el requisito de que todas las instancias de entrenamiento de la clase $y = 1$ (los cuadrados) deben estar situadas en o sobre el hiperplano $w \cdot x + b = 1$.
- Mientras que las instancias de la clase $y = -1$ (los círculos) deben estar situadas en o debajo del hiperplano $w \cdot x + b = -1$.

Aprendiendo una SVM Lineal

- Ambas desigualdades pueden resumirse en una forma más compacta de la siguiente manera:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad i = 1, 2, \dots, N.$$

- La SVM impone el requisito adicional de que el margen del límite de decisión sea maximal.
- Maximizar el margen equivale a minimizar la siguiente función objetivo:

$$f(\mathbf{w}) = \frac{\|\mathbf{w}\|^2}{2}.$$

$$\max_w \frac{2}{\|w\|} \Leftrightarrow \min_w \frac{\|w\|^2}{2}$$

Aprendiendo una SVM Lineal

- La tarea de aprendizaje en SVM puede formalizarse como el siguiente problema de optimización con restricciones:

$$\begin{aligned} & \min_{\mathbf{w}} \frac{\|\mathbf{w}\|^2}{2} \\ & \text{subject to} \quad y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad i = 1, 2, \dots, N. \end{aligned}$$

- Dado que la función objetivo es cuadrática y las restricciones son lineales para los parámetros \mathbf{w} y b , esto se conoce como un problema de optimización **convexa**.
- Este problema puede resolverse mediante el método de **multiplicadores de Lagrange**.

Aprendiendo una SVM Lineal

- Debemos reescribir la función objetivo de una forma que tenga en cuenta las restricciones impuestas a sus soluciones.
- La nueva función objetivo se conoce como el **Lagrangiano** para el problema de optimización:

$$L_P = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^N \lambda_i \left(y_i (\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \right), \quad (1)$$

donde los parámetros λ se llaman los multiplicadores de Lagrange.

- El primer término en el Lagrangiano es el mismo que la función objetivo original, mientras que el segundo término captura las restricciones de desigualdad.

Apreniendo una SVM Lineal

- Si sólo minimizamos el primer término encontraríamos como óptimo $\mathbf{w}=\mathbf{0}$, un vector nulo.
- Esta solución viola las restricciones del problema porque no existe solución válida para \mathbf{b} en ese caso.
- Las soluciones para w y b son inviables si violan las restricciones de desigualdad; es decir, si $y_i(w \cdot x_i + b) - 1 < 0$.
- El Lagrangiano incorpora esta restricción sustrayendo el término de las restricciones de su función objetivo original.
- Asumiendo que $\lambda_i \geq 0$, cualquier solución inviable sólo puede aumentar el valor de la Lagrangiano.

Aprendiendo una SVM Lineal

- Para minimizar el Lagrangiano, debemos calcular la derivada de L_p con respecto a \mathbf{w} y b e igualarlas a cero:

$$\frac{\partial L_p}{\partial \mathbf{w}} = 0 \implies \mathbf{w} = \sum_{i=1}^N \lambda_i y_i \mathbf{x}_i, \quad (2)$$

$$\frac{\partial L_p}{\partial b} = 0 \implies \sum_{i=1}^N \lambda_i y_i = 0. \quad (3)$$

- Como w es un vector, realmente hacemos $\frac{\partial L_p}{\partial w_i}$ para todo i .
- Usamos una notación vectorial para simplificar los cálculos.
- Debido a que los multiplicadores de Lagrange son desconocidos, aún no podemos encontrar los valores para w y b .

Apreniendo una SVM Lineal

- Si el problema de optimización tuviese restricciones de **igualdad** en vez de desigualdad, podríamos usar las N restricciones para encontrar soluciones válidas de w, b , y λ_i haciendo lo siguiente:

$$\frac{\partial L_p}{\partial w} = 0 \quad \frac{\partial L_p}{\partial b} = 0 \quad \frac{\partial L_p}{\partial \lambda_i} = 0 \quad \forall i$$

- Una forma de manejar problemas con restricciones de **desigualdad** es transformar estas restricciones en restricciones de **igualdad** y luego verificar que las soluciones encontradas satisfagan las condiciones de **Karush-Kuhn-Tucker (KKT)**.

$$\lambda_i \geq 0, \quad (4)$$

$$\lambda_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] = 0. \quad (5)$$

- La primera restricción nos exige que los multiplicadores de Lagrange sean no negativos.
- La segunda restricción se conoce como “holgura complementaria” y es crucial para encontrar los **vectores de soporte**.

Apreniendo una SVM Lineal

- A primera vista, puede parecer que existen tantos multiplicadores de Lagrange como instancias de entrenamiento.
- Sin embargo, muchos de los multiplicadores de Lagrange quedan en cero después de aplicar la restricción de la ecuación (5): $\lambda_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] = 0$.
- La restricción (5) establece que el multiplicador de Lagrange λ_i debe ser cero a menos que la instancia de entrenamiento \mathbf{x}_i satisfaga la ecuación $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) = 1$.
- Un ejemplo de entrenamiento con $\lambda_i > 0$, se encuentra dentro de los hiperplanos \mathbf{b}_{i1} o \mathbf{b}_{i2} y se conoce como **vector de soporte**.
- Los ejemplos de entrenamiento que no residen en \mathbf{b}_{i1} o \mathbf{b}_{i2} tienen $\lambda_i = 0$.
- Las ecuaciones (2) y (3) también indican que los parámetros \mathbf{w} y b , que definen el límite de decisión, dependen sólo de los **vectores de soporte**.

Apreniendo una SVM Lineal

- Resolver este problema de optimización es aún computacionalmente caro porque involucra un gran número de parámetros: w , b , y λ_i .
- El problema puede simplificarse transformando el Lagrangiano en una función que depende solamente de los multiplicadores de Lagrange conocida como el **problema dual**.
- El dual de un problema de optimización es una visión alternativa del problema **primal** que para problemas convexos (como este) tienen resultados equivalentes.

- Para encontrar el dual, sustituimos las ecuaciones (2) y (3) en la ecuación (1):

$$\mathbf{w} = \sum_{i=1}^N \lambda_i y_i \mathbf{x}_i, \quad \sum_{i=1}^N \lambda_i y_i = 0. \quad \implies \quad L_P = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^N \lambda_i \left(y_i (\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \right)$$

- Esto nos llevará a la siguiente formulación dual del problema de optimización:

$$L_D = \sum_{i=1}^N \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j. \quad (6)$$

Aprendiendo una SVM Lineal

- Resolución:
$$L_p = \frac{1}{2} w^T w - \sum_{i=1}^N \lambda_i (y_i (w \cdot x_i + b) - 1)$$

$\|w\|^2 = w^T \cdot w$, luego reemplazando (2) $w = \sum_{i=1}^N \lambda_i y_i x_i$,

$$= \frac{1}{2} \left(\sum_i^N \lambda_i y_i x_i \right)^T \left(\sum_j^N \lambda_j y_j x_j \right) - \sum_{i=1}^N \lambda_i \left(y_i \left(\left(\sum_j^N \lambda_j y_j x_j \right) \cdot x_i + b \right) - 1 \right)$$

reordenando:

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

vale cero por (3)

$$L_D = \sum_{i=1}^N \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j x_i \cdot x_j. \quad (6)$$

Lagrangiano Primal y Dual

- El Lagrangiano dual involucra sólo los multiplicadores de Lagrange y los datos de entrenamiento (depende de menos parámetros).
- Mientras que el Lagrangiano primal involucra los multiplicadores de Lagrange así como los parámetros del hiperplano separador (w,b) .
- El problema de **minimización** original para el Lagrangiano primal L_p se convierte en un problema de **maximización** para el Lagrangiano dual L_d .
- Como este es un problema de optimización convexo las soluciones para ambos problemas de optimización son **equivalentes** (siempre y cuando nuestra solución satisfaga las condiciones KKT).

Apreniendo una SVM Lineal

- El problema de la optimización dual puede resolverse mediante técnicas numéricas de **programación cuadrática**.
- Existe un algoritmo muy eficiente para resolver ese problema llamado **Sequential Minimal Optimization (SMO)**.
- Una vez encontrados los valores de λ_i , podemos utilizar las ecuaciones (2) y (5),
$$\mathbf{w} = \sum_{i=1}^N \lambda_i y_i \mathbf{x}_i, \quad \lambda_i [y_i (\mathbf{w} \cdot \mathbf{x}_i + b) - 1] = 0,$$
 para obtener soluciones válidas para \mathbf{w} y b .
- El límite de decisión queda expresado de la siguiente forma, dependiendo solo de los **vectores de soporte**:

$$\left(\sum_{i=1}^N \lambda_i y_i \mathbf{x}_i \cdot \mathbf{x} \right) + b = 0. \quad (7)$$

- El valor de b se obtiene resolviendo la ecuación (5) para los vectores de soporte.

Aprendiendo una SVM Lineal

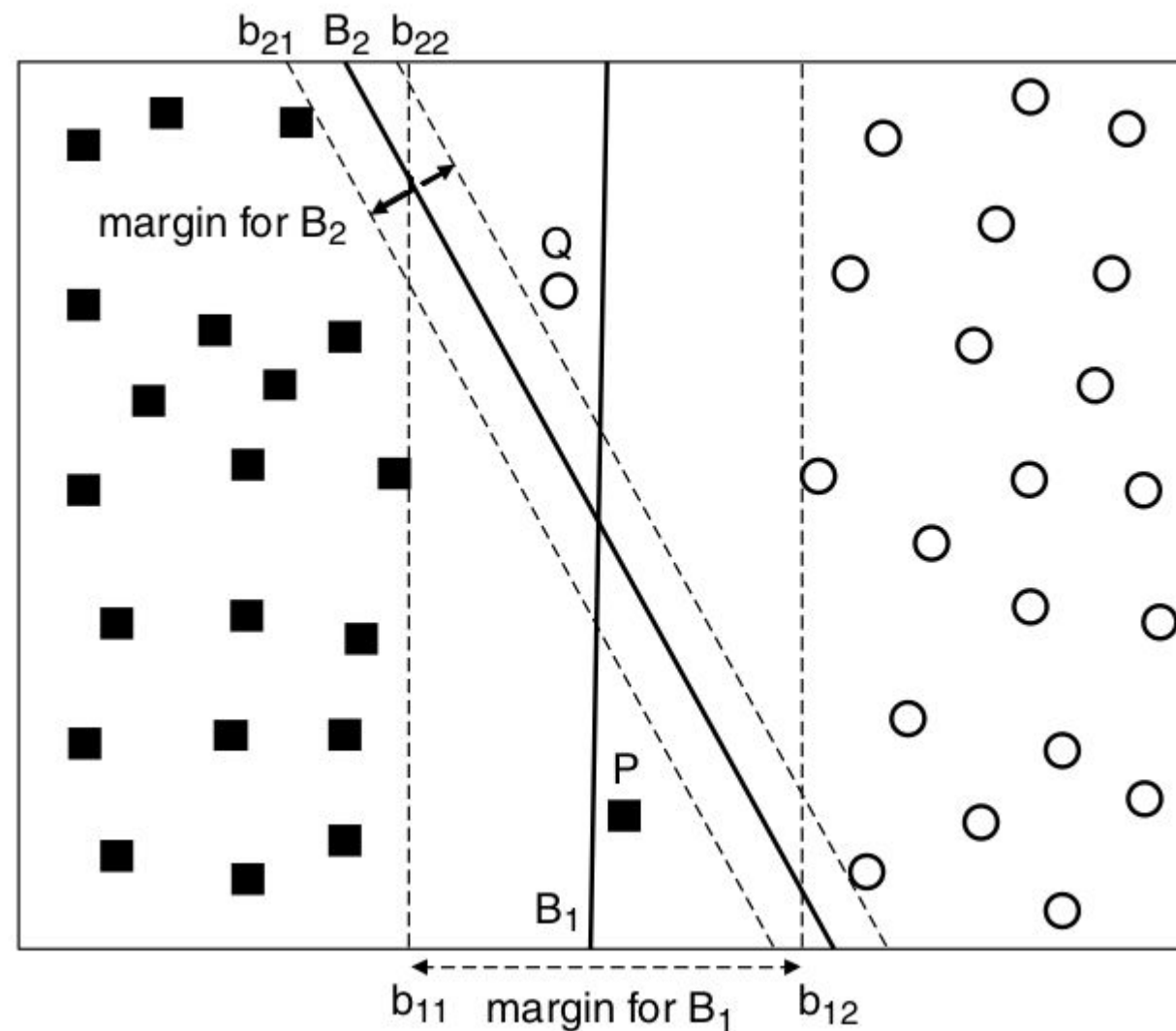
- Debido a que los λ_i se calculan numéricamente, éstos pueden tener errores numéricos y el valor calculado para b puede no ser único.
- De hecho, depende del **vector de soporte** utilizado en la Ecuación (5).
- En la práctica, se elige el valor promedio de b para todos los vectores de soporte.
- Una vez que se encuentran los parámetros del límite de decisión, clasificamos un ejemplo de testing \mathbf{z} de la siguiente manera:

$$f(\mathbf{z}) = \text{sign}(\mathbf{w} \cdot \mathbf{z} + b) = \text{sign}\left(\sum_{i=1}^N \lambda_i y_i \mathbf{x}_i \cdot \mathbf{z} + b\right).$$

- Si $f(\mathbf{z}) = 1$, entonces el ejemplo se clasifica con la clase positiva; de lo contrario, se clasifica con la clase negativa.
- Para clasificar solo necesitamos iterar sobre los **vectores de soporte** (donde λ_i distinto de 0) que son muchos menos que los datos entrenamiento.

SVM de margen suave

- La figura de más abajo muestra el dataset del ejemplo anterior con dos ejemplos nuevos (**ruidosos**): un círculo Q y un cuadrado P.



- También muestra dos hiperplanos separadores posibles B_1 y B_2 .

SVM de margen suave

- El límite de decisión B_1 clasifica erróneamente los dos nuevos ejemplos, mientras que B_2 los clasifica correctamente.
- Pero esto no significa que B_2 sea un mejor límite de decisión que B_1 .
- Los nuevos ejemplos pueden corresponder a ruido en los datos de entrenamiento.
- B_1 debería seguir siendo preferible a B_2 porque tiene un margen más amplio lo que lo hace menos susceptible a overfitting.
- La formulación de la SVM presentada en la anteriormente no contempla errores en los datos de entrenamiento.

SVM de margen suave

- Ahora veremos cómo modificar el problema de optimización para aprender un límite de decisión que sea tolerable al ruido usando un método conocido como el **margen suave** (soft margin).
- Esto permitirá a la SVM construir una límite de decisión incluso en situaciones en las que las clases no sean linealmente separables.
- Para esto, el algoritmo de aprendizaje de SVM debe encontrar un balance entre el **ancho del margen** y el número de **errores** hechos por la SVM sobre los datos de **entrenamiento**.

SVM de margen suave

- El límite de decisión de B_1 no satisface las restricciones del problema original

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad i = 1, 2, \dots, N.$$

- Debemos relajar las restricciones para poder suportar datos no linealmente separables.
- Esto se puede hacer introduciendo **variables de holgura** o slack variables (ξ) a las restricciones del problema:

$$\begin{aligned} \mathbf{w} \cdot \mathbf{x}_i + b &\geq 1 - \xi_i \quad \text{if } y_i = 1, \\ \mathbf{w} \cdot \mathbf{x}_i + b &\leq -1 + \xi_i \quad \text{if } y_i = -1, \end{aligned}$$

where $\forall i : \xi_i > 0$.

SVM de margen suave

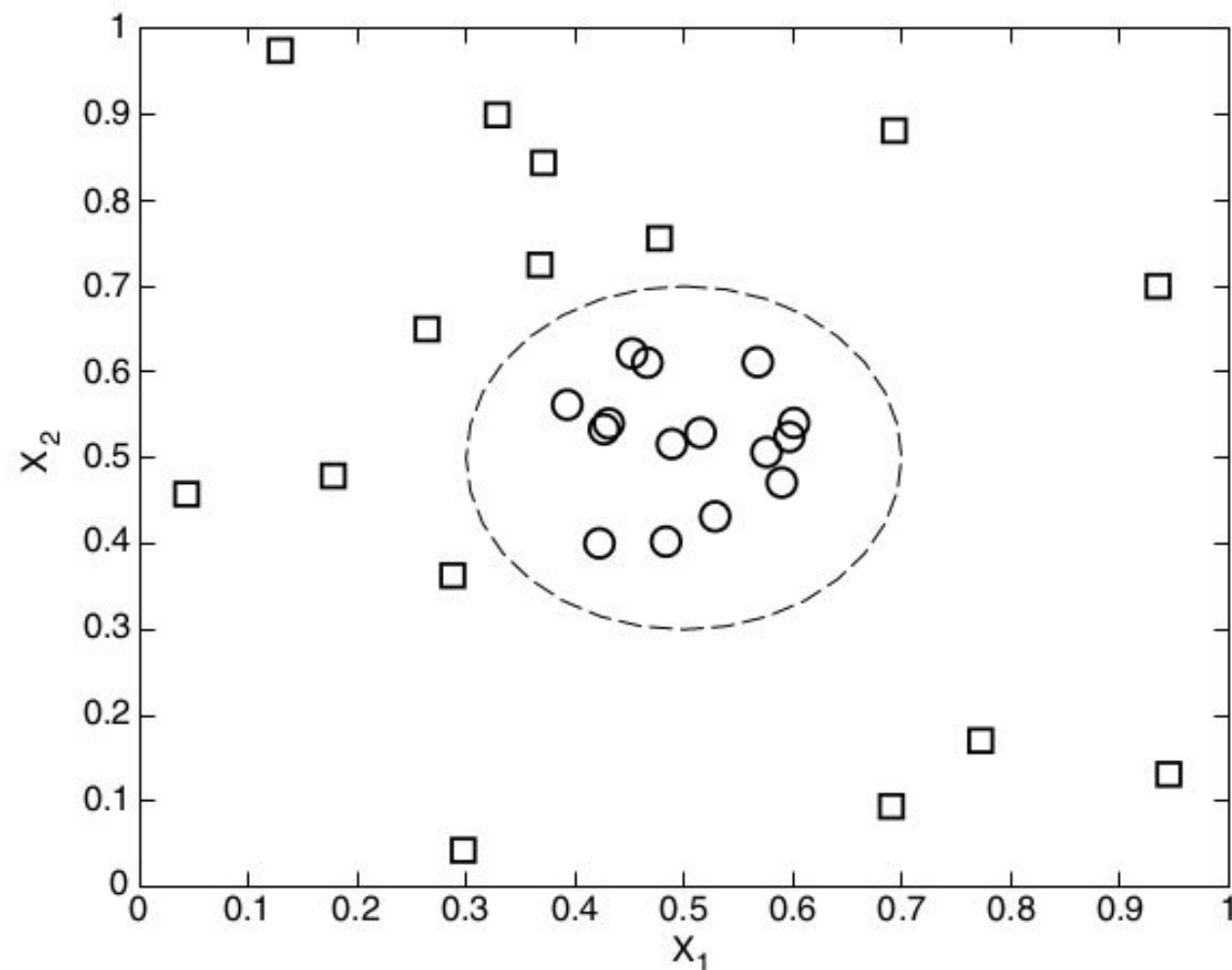
- El problema de optimización primal quedaría formulado como minimizar la siguiente función objetivo:

$$\frac{1}{2}||w||^2 + C \sum_{i=1}^m \xi_i$$

$$\text{s.t. } y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, \quad i = 1, \dots, m$$
$$\xi_i \geq 0, \quad i = 1, \dots, m.$$

- Donde ξ_i representa la magnitud del error para el ejemplo i y C es un costo que pagamos por equivocarnos.
- El valor del parámetro C se debe calibrar sobre un conjunto de datos independientes llamado de **validación** o haciendo cross-validation.
- Una vez formulado el problema procedemos de manera análoga: planteamos el nuevo Lagrangiano, el dual y verificamos que las soluciones respeten las condiciones KKT.

Support Vector Machines No-Lineales



- El diagrama muestra un ejemplo de un dataset bidimensional compuesto por cuadrados ($y = 1$) y círculos ($y = -1$).
- Todos los círculos están agrupados cerca del centro del diagrama y todos los cuadrados se distribuyen más lejos del centro.
- Este problema no se puede resolver con una SVM lineal.

Support Vector Machines No-Lineales

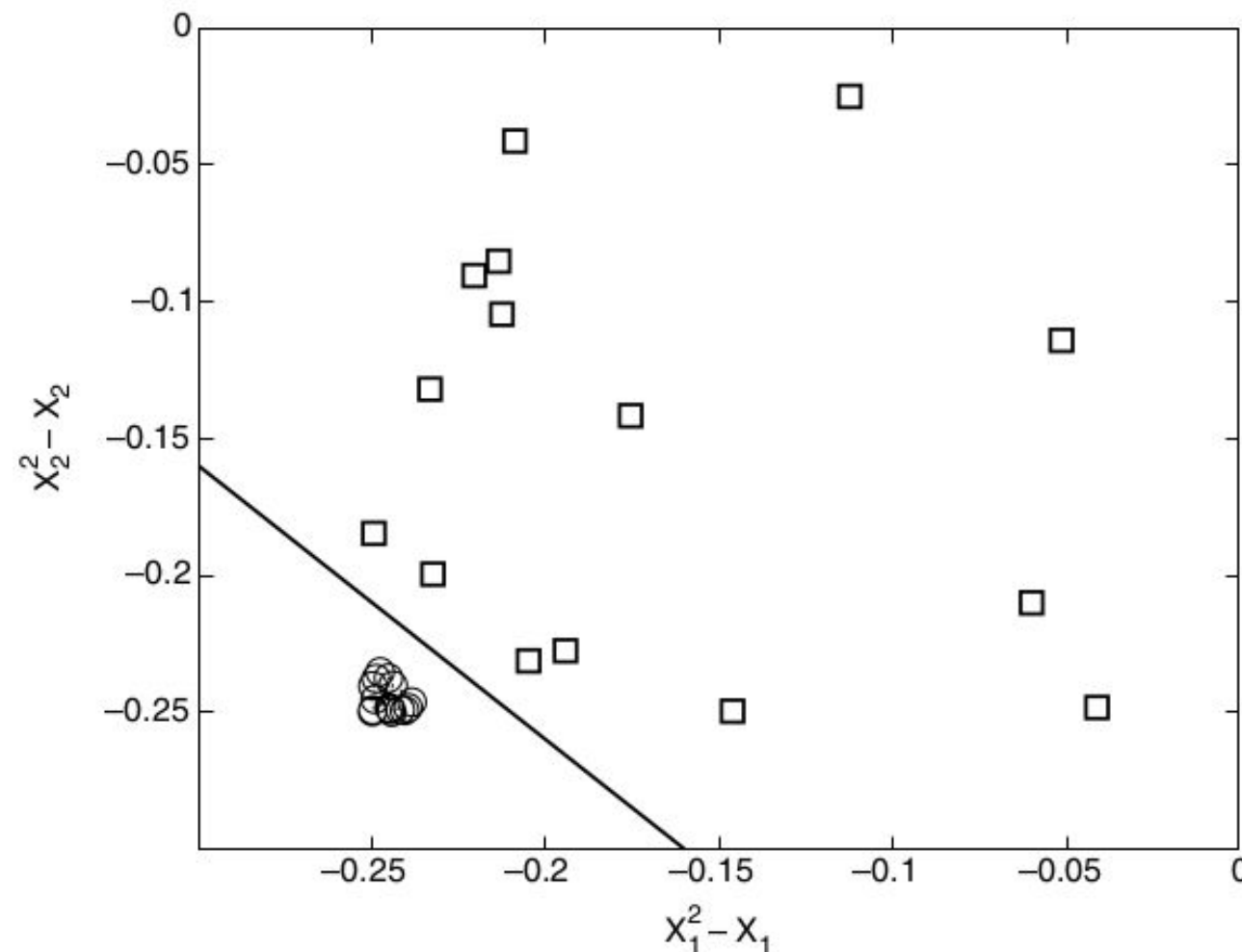
- Podríamos aplicar una transformación no lineal Φ para mapear los datos de su espacio de original a un nuevo espacio (de más dimensiones) donde el límite de decisión se vuelva lineal.
- Supongamos que elegimos la siguiente transformación que transforma de 2 dimensiones a 5 dimensiones:

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

- En este espacio transformado si es posible encontrar parámetros $\mathbf{w} = (w_0, w_1, \dots, w_4)$ que separen linealmente los datos:

$$w_4x_1^2 + w_3x_2^2 + w_2\sqrt{2}x_1 + w_1\sqrt{2}x_2 + w_0 = 0.$$

Support Vector Machines No-Lineales



- La figura muestra que en el espacio transformado se puede construir un límite de decisión lineal para separar las clases.
- Un problema potencial de este enfoque es que puede sufrir de la **maldición la dimensionalidad**: para datos de alta dimensión muchas técnicas de data mining no escalan o no funcionan bien.
- Mostraremos cómo una SVM no lineal evita este problema usando un truco llamado **Kernel Trick**.

Support Vector Machines No-Lineales

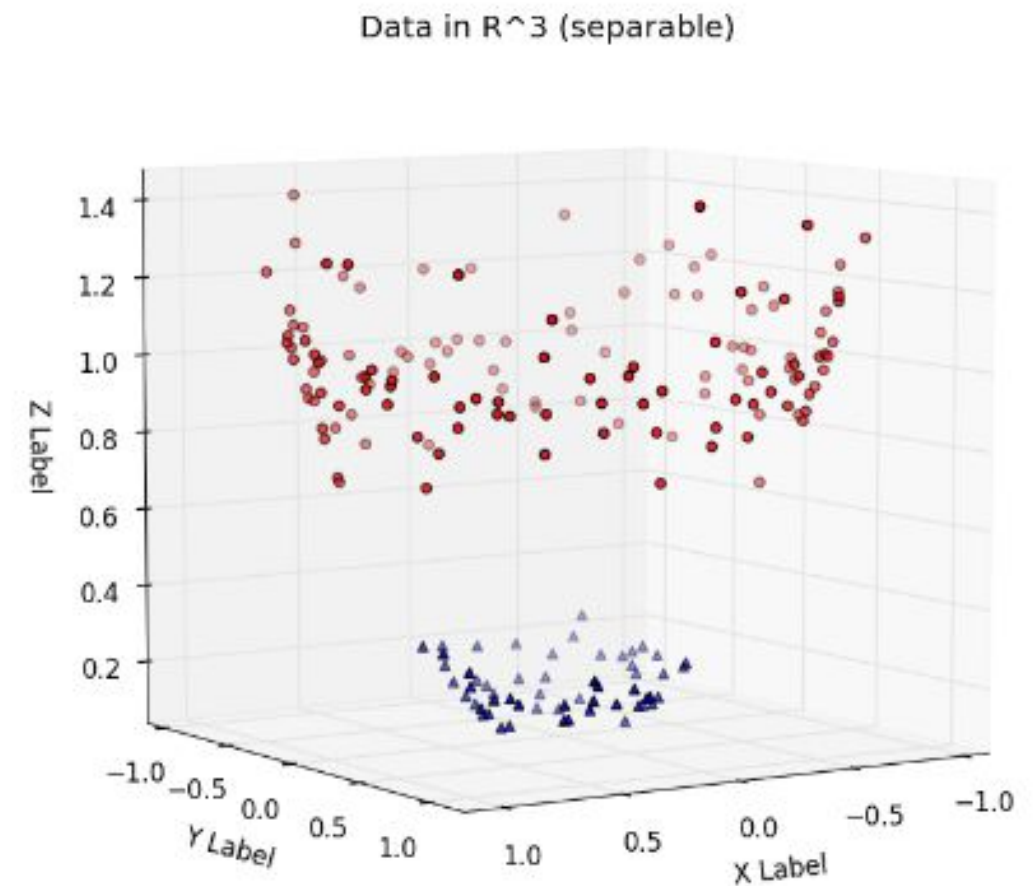
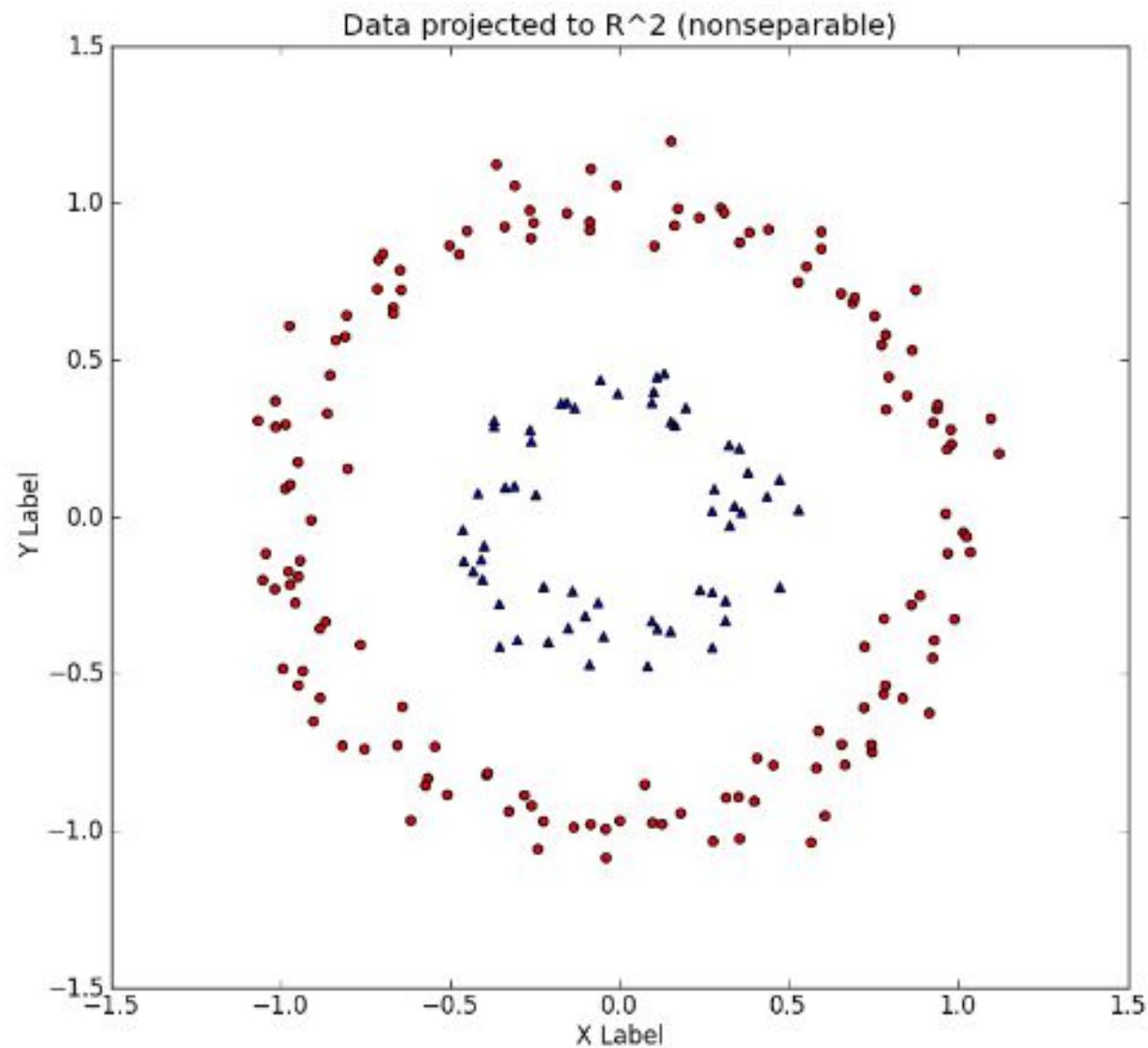


Figure 5: (Left) A dataset in \mathbb{R}^2 , not linearly separable. (Right) The same dataset transformed by the transformation:
$$[x_1, x_2] = [x_1, x_2, x_1^2 + x_2^2].$$

source:

http://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html

Support Vector Machines No-Lineales

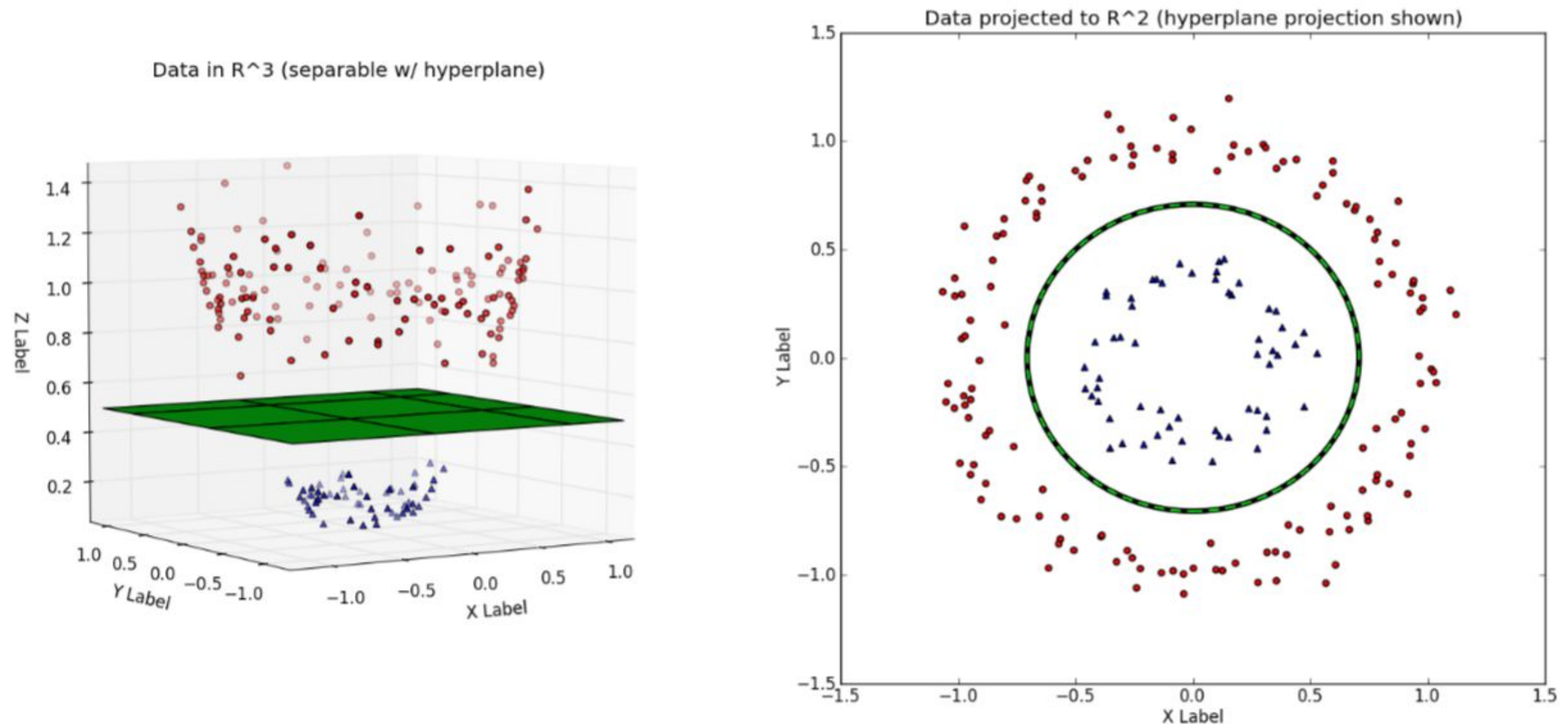


Figure 6: (Left) The decision boundary \vec{w} shown to be linear in \mathbb{R}^3 . (Right) The decision boundary \vec{w} , when transformed back to \mathbb{R}^2 , is nonlinear.

source:

http://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html

Support Vector Machines No-Lineales

- Usando la idea de tener una función $\Phi(\mathbf{x})$ que mapea los datos a una dimensión mayor, podemos redefinir el problema de optimización.

$$\begin{aligned} & \min_{\mathbf{w}} \frac{\|\mathbf{w}\|^2}{2} \\ \text{subject to} \quad & y_i(\mathbf{w} \cdot \Phi(\mathbf{x}_i) + b) \geq 1, \quad i = 1, 2, \dots, N. \end{aligned}$$

- Lo único que hacemos es reemplazar todas las instancia del vector \mathbf{x} por $\Phi(\mathbf{x})$.
- Podemos formular nuevamente el dual y las expresiones que nos permiten encontrar los valores óptimos de \mathbf{w} y b .

$$L_D = \sum_{i=1}^n \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$$
$$\begin{aligned} \mathbf{w} &= \sum_i \lambda_i y_i \Phi(\mathbf{x}_i) \\ \lambda_i \{ y_i (\sum_j \lambda_j y_j \Phi(\mathbf{x}_j) \cdot \Phi(\mathbf{x}_i) + b) - 1 \} &= 0, \end{aligned}$$

Support Vector Machines No-Lineales

- Análogamente, una instancia de prueba \mathbf{z} puede clasificarse utilizando la siguiente ecuación:

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

- Si nos fijamos en las ecuaciones veremos que tanto para la formulación del dual como para clasificar un ejemplo $\Phi(\mathbf{x})$ siempre está acompañada por la función Φ aplicada a otro vector por medio de un producto punto (ej: $\Phi(\mathbf{x}) \cdot \Phi(\mathbf{z})$).
- El producto punto de dos vectores tiene relación con su similitud (piensen en el coseno del ángulo). Entonces $\Phi(\mathbf{x}) \cdot \Phi(\mathbf{z})$ representa la similitud entre \mathbf{x} y \mathbf{z} en el espacio transformado.
- Si las transformaciones $\Phi(\mathbf{x})$ son de alta dimensión, entonces calcular $\Phi(\mathbf{x}) \cdot \Phi(\mathbf{z})$ será computacionalmente caro.
- El truco del Kernel trick consiste en que para algunas funciones Φ es posible calcular el resultado de $\Phi(\mathbf{x}) \cdot \Phi(\mathbf{z})$ directamente en el espacio original sin necesidad de calcular $\Phi(\mathbf{x})$ o $\Phi(\mathbf{z})$ por separado.

Support Vector Machines No-Lineales

- Usemos la siguiente función $\Phi : (x_1, x_2) \rightarrow (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, 1)$.
- Sean dos vectores u, v , calculemos $\Phi(u) \cdot \Phi(v)$:

$$\begin{aligned}\Phi(u) \cdot \Phi(v) &= (u_1^2, u_2^2, \sqrt{2}u_1, \sqrt{2}u_2, \sqrt{2}u_1u_2, 1) \cdot (v_1^2, v_2^2, \sqrt{2}v_1, \sqrt{2}v_2, \sqrt{2}v_1v_2, 1) \\ &= u_1^2v_1^2 + u_2^2v_2^2 + 2u_1v_1 + 2u_2v_2 + 2u_1u_2v_1v_2 + 1 \\ &= (u \cdot v + 1)^2.\end{aligned}$$

- El producto punto en el espacio transformado se puede expresar como una función de similitud en el espacio original:

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

- La función de similitud, K , que se calcula en el espacio de atributos original, se conoce como la función de **kernel**.
- El kernel trick consiste en reemplazar $\Phi(x) \cdot \Phi(z)$ por $K(x,z)$ en las ecuaciones de la SVM y así trabajar en el espacio transformado de manera implícita.

Support Vector Machines No-Lineales

- Si usamos la función de Kernel polinomial: $K(\mathbf{u}, \mathbf{v}) = \Phi(\mathbf{u}) \cdot \Phi(\mathbf{v}) = (\mathbf{u} \cdot \mathbf{v} + 1)^2$.
- Una SVM clasificaría un ejemplo nuevo \mathbf{z} de la siguiente forma:

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

- Existen varias funciones de Kernel que se pueden usar:

Polinomial: $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^p$

Radial o Gaussiano: $K(\mathbf{x}, \mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|^2/(2\sigma^2)}$

Sigmoidal: $K(\mathbf{x}, \mathbf{y}) = \tanh(k\mathbf{x} \cdot \mathbf{y} - \delta)$

Support Vector Machines No-Lineales

- Los parámetros de la función de Kernel (ej: el valor de σ para un Kernel radial) deben ajustarse experimentalmente en un dataset de validación o haciendo cross-validation.
- En la práctica uno hace una búsqueda de grilla para combinaciones de valores de hiperparametros (ej: el valor C del margen suave y el valor de σ del Kernel radial).

- Un ejemplo de búsqueda típico es

$$C \in \{2^{-5}, 2^{-3}, \dots, 2^{13}, 2^{15}\} \quad \sigma \in \{2^{-15}, 2^{-13}, \dots, 2^1, 2^3\}$$

- Uno no puede usar cualquier función como función de Kernel, el teorema de **Mercer** muestra matemáticamente que la función K debe ser positiva semi-definida.
- Esto implica que la función es representable como el producto punto de dos vectores en el espacio transformado: $\Phi(x) \cdot \Phi(z)$ por $K(x,z)$.
- En la práctica nos limitamos a usar las funciones de Kernel conocidas que sabemos que cumplen con la condición.

Support Vector Machines No-Lineales

- El espacio transformado de una función de Kernel en una SVM se llama **reproducing kernel Hilbert space (RKHS)**.
- El cálculo de los productos de puntos usando las funciones del Kernel es considerablemente más barato que operar sobre el espacio transformado $\Phi(x)$.
- Como los cálculos se realizan en el espacio original, se evitan los problemas asociados con la maldición de la dimensionalidad.

Conclusiones

- El problema de aprendizaje de una SVM se formula como un problema de **optimización convexa** en donde hay algoritmos eficientes para encontrar el óptimo global.
- Otros métodos de clasificación como como los árboles de decisión y las redes neuronales tienden a encontrar **óptimos locales**.
- La SVM optimiza explícitamente la **capacidad de generalización** al maximizar el margen del límite de decisión.
- En una SVM el usuario debe **ajustar** hiper-parámetros, como el tipo de función de Kernel y el costo C para las variables de holgura (esto puede ser caro).
- La SVM puede aplicarse a los datos categóricos creando **variables dummy** binarias por cada categoría.
- La formulación de SVM presentada en esta clase se limita a problemas de clasificación **binaria**.

Conclusiones

- Existen adaptaciones para trabajar con múltiples clases.
- Lo más simple es entrenar un clasificador por cada clase donde los ejemplos negativos se obtienen combinando todos los ejemplos que no sean de la clase correspondiente. Luego se clasifica a la clase que se prediga con mayor margen (confianza).
- Esto se llama **One-vs-all**.
- Las SVMs fueron unos de los métodos de Machine Learning más populares en los 90's y principios de los años 2000.
- La gran limitación de las SVMs es que **no escalan** bien para datasets masivos.
- Hoy en día el uso de **redes neuronales profundas** les ha quitado su popularidad.
- Sin embargo, entender cómo funcionan las SVMs sigue siendo muy valioso.



dcc

CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE CHILE

www.dcc.uchile.cl

f @ in  / DCCUCHILE