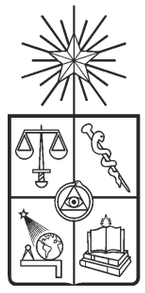


dcc
CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE CHILE

Clasificación II

Naive Bayes y Support Vector Machines

Felipe Bravo

Clasificación basada en Naïve Bayes

- Familia de modelos basados en Bayes.
- Veremos Clasificador de Naive Bayes.
- También existen las Redes Bayesianas.

Clasificación Basada en Naïve Bayes

- Modelo que busca modelar la relación **probabilística** entre atributos y clase.
- Modelo generativo, asume una distribución conjunta entre X e Y.
- Supuesto: atributos **independientes** dado la clase (naive assumption).

Clasificador Bayesiano

- Esquema probabilístico para resolver problemas de clasificación.
- Probabilidad condicional:

$$P(C \mid A) = \frac{P(A,C)}{P(A)}$$

$$P(A \mid C) = \frac{P(A,C)}{P(C)}$$

- Teorema de Bayes:

$$P(C \mid A) = \frac{P(A \mid C)P(C)}{P(A)}$$

Ejemplo Teorema de Bayes

- Dado:
  - Un doctor sabe que la meningitis produce rigidez de cuello el 50% de las veces.
  - La probabilidad previa de que cualquier paciente tenga meningitis es 1/50,000.
  - La probabilidad previa de que cualquier paciente tenga rigidez en el cuello es de 1/20.
- ¿Si un paciente tiene el cuello rígido, cuál es la probabilidad de que tenga meningitis?

$$P(M \mid S) = \frac{P(S \mid M)P(M)}{P(S)} = \frac{0.5 \times 1/50000}{1/20} = 0.0002$$

Clasificador Naïve Bayes

- Considerar cada atributo como variable condicionalmente independiente de la clase (eso es “naive”).

- Dado un record con atributos (A_1, A_2,…,A_n).
  - La meta es predecir la clase C.
  - Específicamente queremos encontrar el C que maximice P(C| A_1, A_2,…,A_n).

- ¿Podemos estimar P(C| A_1, A_2,…,A_n) directamente de los datos?

Clasificador Naïve Bayes

- Aproximación

  - Computar la probabilidad posterior P(C | A_1, A_2, …, A_n) para todos los valores de C usando el Teorema de Bayes.

$$P(C \mid A_1 A_2 \ldots A_n) = \frac{P(A_1 A_2 \ldots A_n \mid C) P(C)}{P(A_1 A_2 \ldots A_n)}$$

- Elegir un valor de C que maximice
  P(C | A_1, A_2, …, A_n).

- Equivalente a elegir un valor de C que maximice
  P(A_1, A_2, …, A_n|C) P(C).

- Esto es porque el denominador P(A_1A_2...A_n) es constante para todas las clases.

Clasificador Naïve Bayes

- Asume independencia entre los atributos A_i cuando la clase está dada (independencia condicional):
- $P(A_1, A_2, \ldots, A_n | C) = P(A_1 | C_j)\, P(A_2 | C_j) \ldots P(A_n | C_j)$.
- Se puede estimar $P(A_i | C_j)$ para todos los A_i y C_j.
- Un punto nuevo A, se clasifica como C_j si $P(C_j) \prod P(A_i | C_j)$ es máxima (en comparación con otros valores de C).

¿Cómo estimar probabilidades a partir de los datos?

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Clase: $P(C_k) = \frac{count(C_k)}{N}$
  - e.g., P(No) = 7/10,
    P(Yes) = 3/10
- Para atributos discretos:

$$P(A_i = b|C_k) = \frac{count(A_{ik} = b)}{count(C_k)}$$

- donde count(A_{ik}=b) es el número de instancias que tiene el valor b para el atributo A_i y que pertenecen a la clase C_k
- Ejemplos:
- P(Status =Married|No) = 4/7
  P(Refund= Yes |Yes) = 0

Laplace Smoothing

- P(C | A_1, A_2, …, A_n) se puede ir a cero cuando |A_{ik}=b| = 0, osea cuando para alguna clase C_k no hay ningún ejemplo con A_i=b.

- En ese caso Naive Bayes le asignaría probabilidad cero a la clase C_k a cualquier ejemplo con |A_{ik}=b| = 0, ignorando el valor de los otros atributos (acuérdense que las probabilidades se multiplican).

- Eso no es bueno para la generalización del modelo.

- Laplace Smoothing: soluciona el problema sumándole 1 a todos los conteos para que ninguna probabilidad quede en cero:

$$P(A_i = b|C_k) = \frac{count(A_{ik} = b) + 1}{count(C_K) + values(A_i)}$$

- Donde values(A_i) es la cantidad de categorías del atributo A_i.

- Con Laplace smoothing P(Status =Married|No) = (4+1)/(7+3)

Atributos Numéricos

- ¿Cómo calculamos $P(A_{ik}=b|C_k)$ cuando el atributo A_i es numérico (ej: Taxable income) ?

- Una opción es discretizar el atributo y proceder de la forma anterior.

- Otra solución es asumir que el atributo sigue una distribución Gaussiana y estimar los parámetros de la función de densidad:

$$P(A_i = b|C_k) = \frac{1}{\sqrt{2\pi}\sigma_{ik}} \exp^{-\frac{(b-\mu_{ik})^2}{2\sigma_{ik}^2}}$$

- Aquí μ_{ik} y σ_{ik} se estiman como la media muestral y la desviación estándar de los ejemplos del atributo A_i cuando la clase es C_k

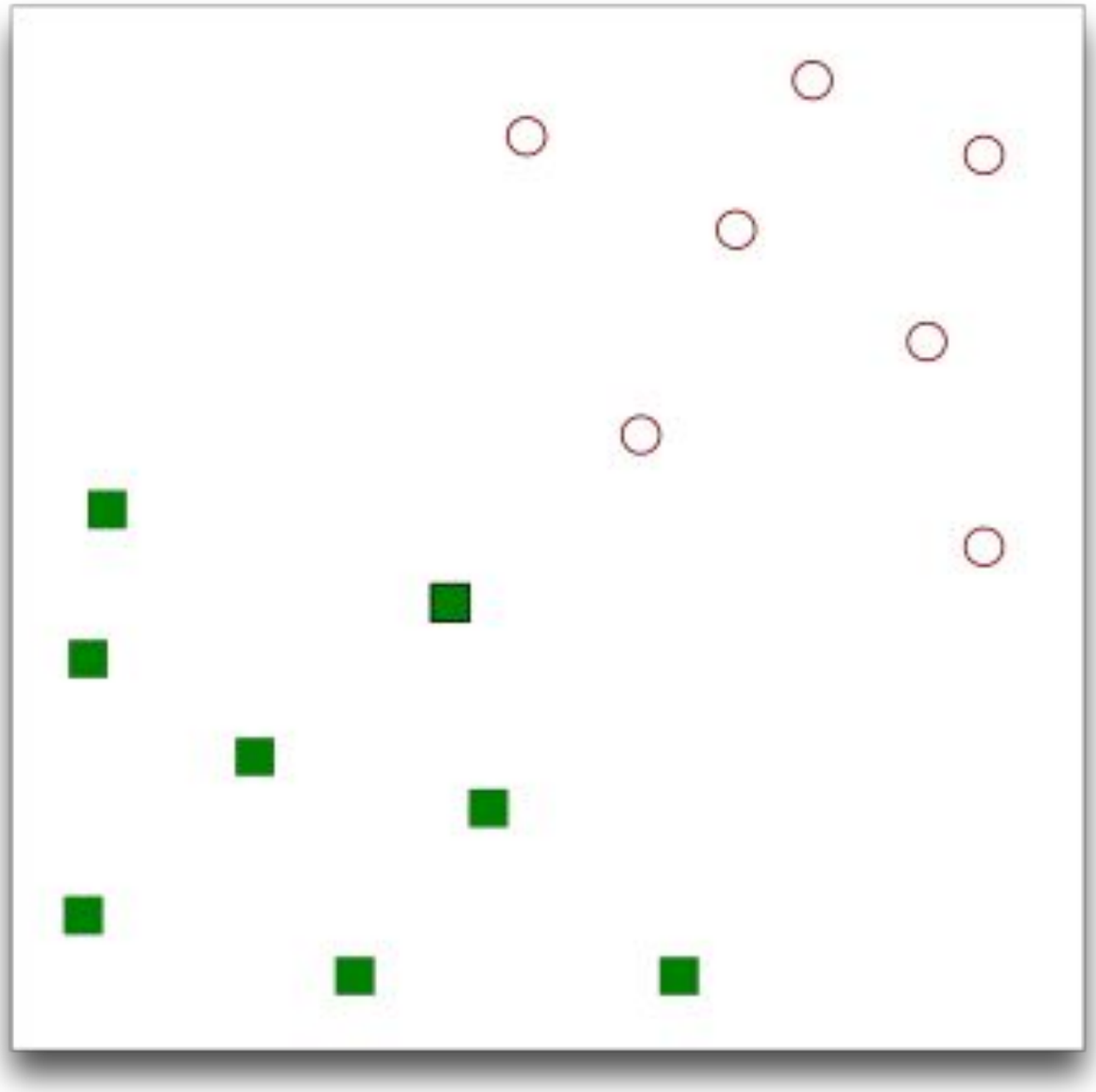
- Sea A_i = Taxable income y C_k=No, μ_{ik}= mean(125,100,70,120,60,220,75) =110 y σ_{ik}=sd(125,100,70,120,60,220,75)=54.5

P(Taxable Income= 130 |No)= dnorm(x=130,mean=110,sd=54.5) = 0.006843379

Naïve Bayes (Resumen)

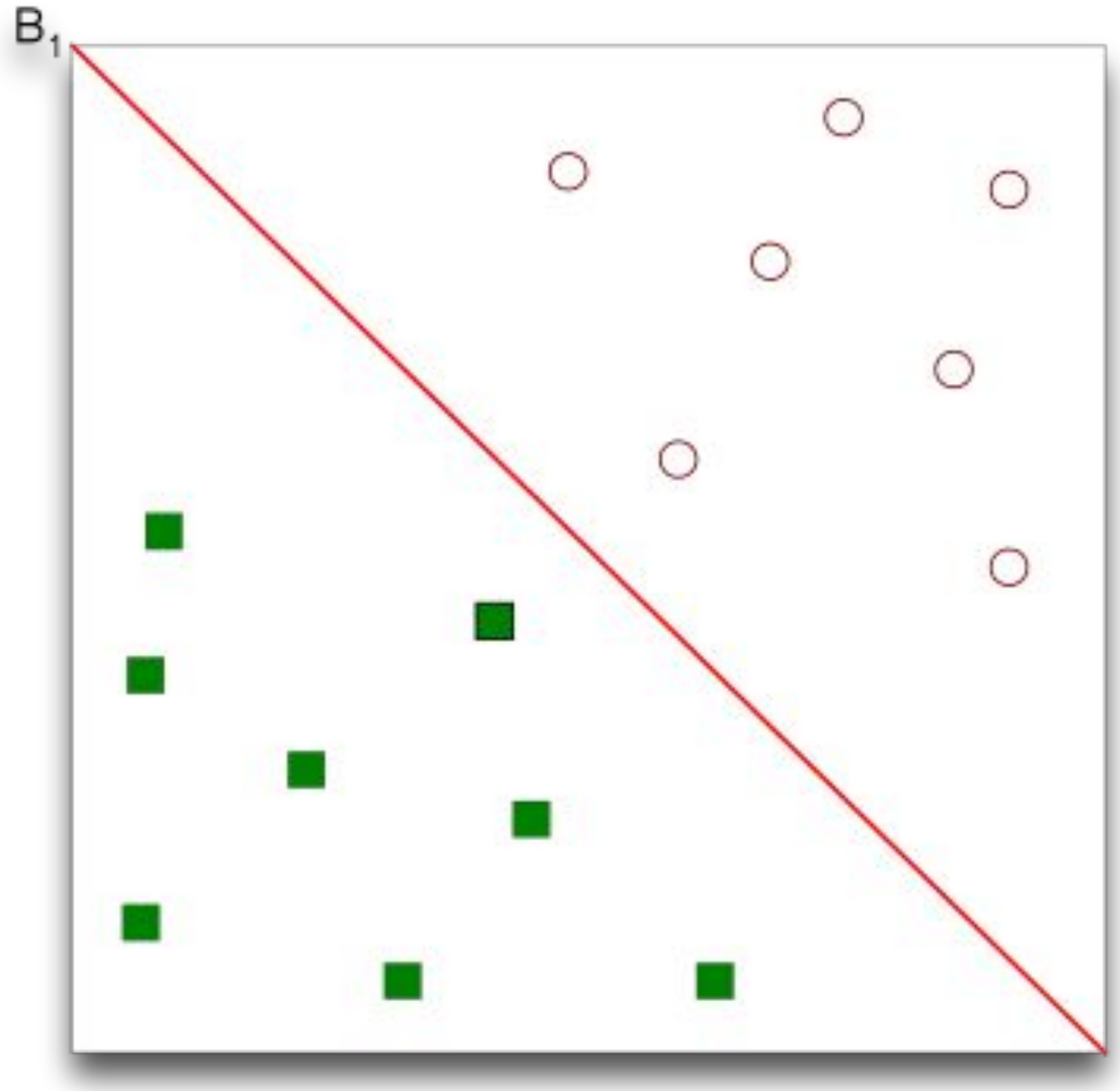
- Es robusto ante puntos de ruido aislados.
- Maneja valores faltantes ignorando la instancia durante los cálculos de estimación de probabilidades.
- Robusto a atributos irrelevantes (afectan de igual manera a todas las clases).
- El supuesto de independencia entre atributos puede no ser cierto en todos los casos.
- Las redes Bayesianas o los modelos gráficos dirigidos permiten hacer modelos probabilísticos con supuestos de independencia menos restrictivos.

Support Vector Machines



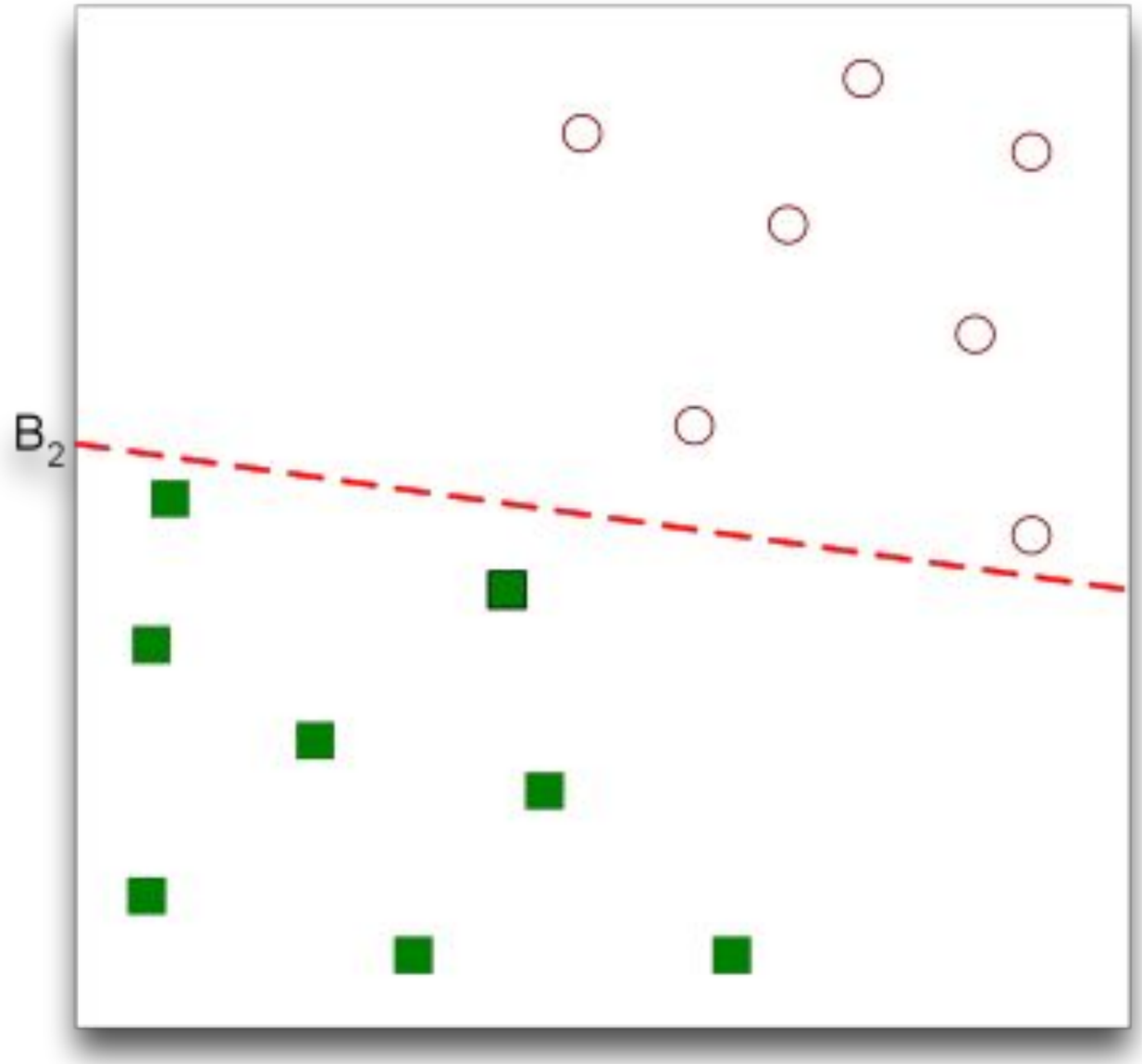
- Encontrar un hiperplano lineal (decision boundary) que separe los ejemplos positivos de los negativos.

Support Vector Machines



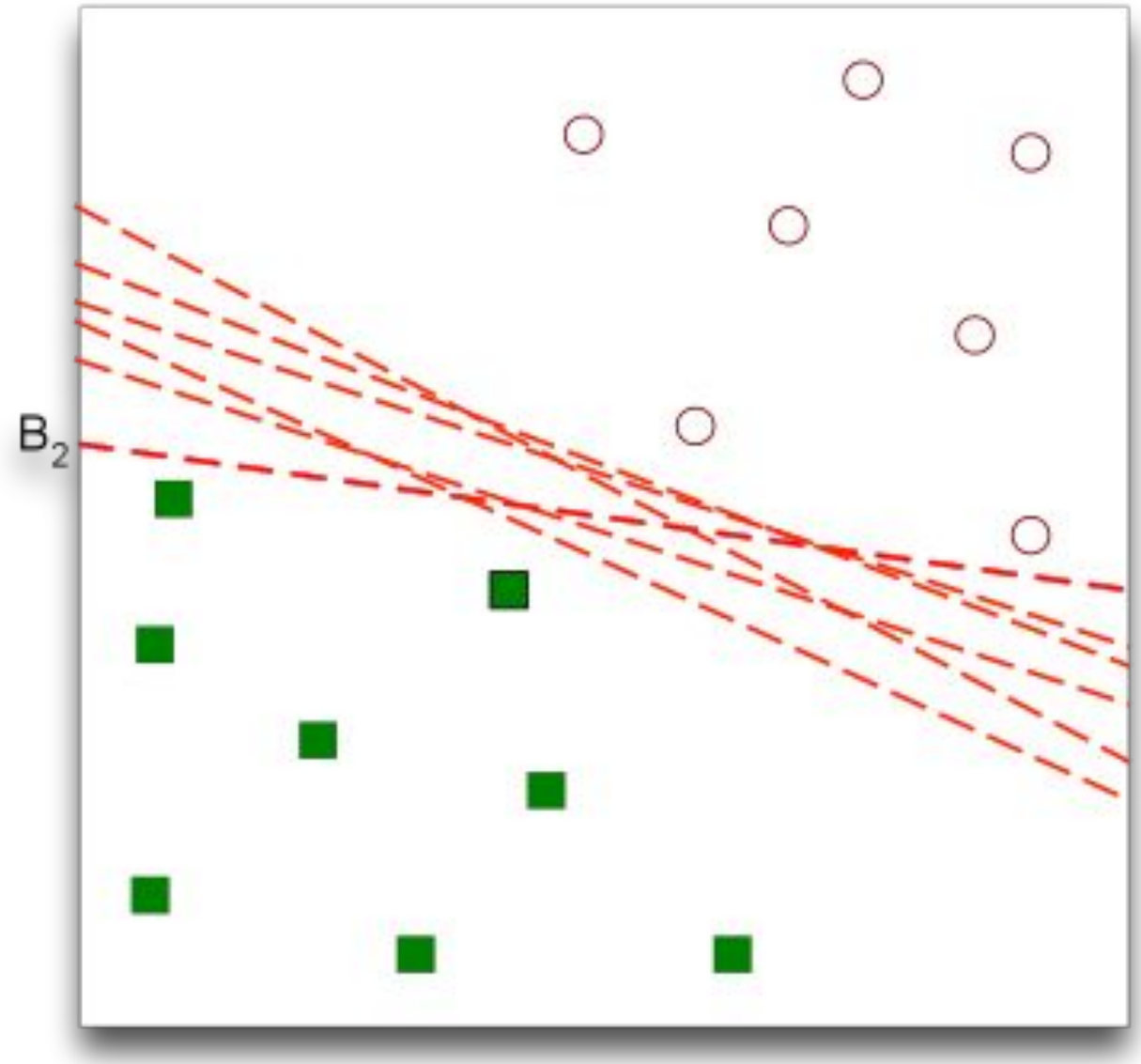
- Una posible solución

Support Vector Machines



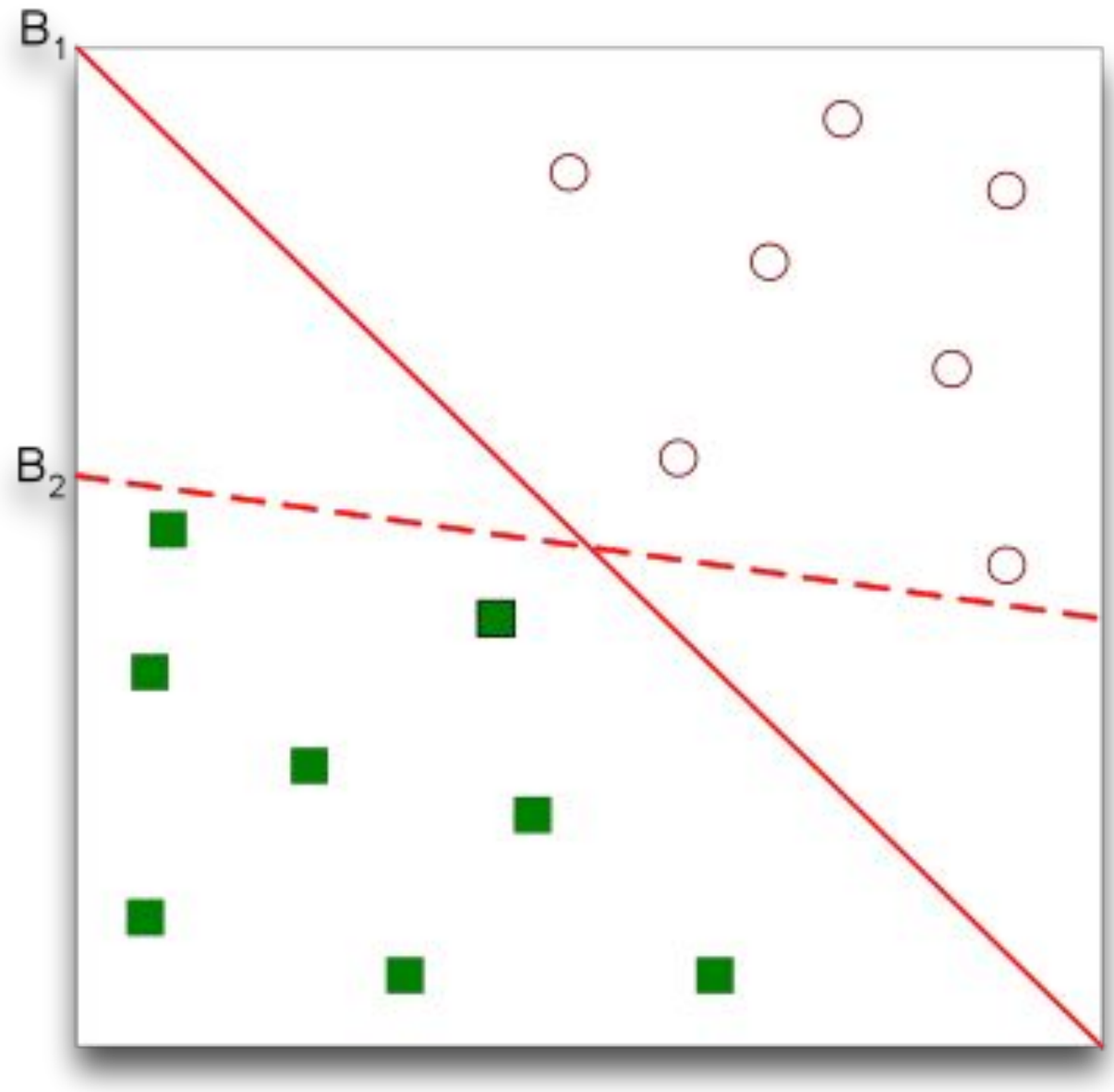
- Otra solución posible

Support Vector Machines



- ¡Infinitas soluciones!

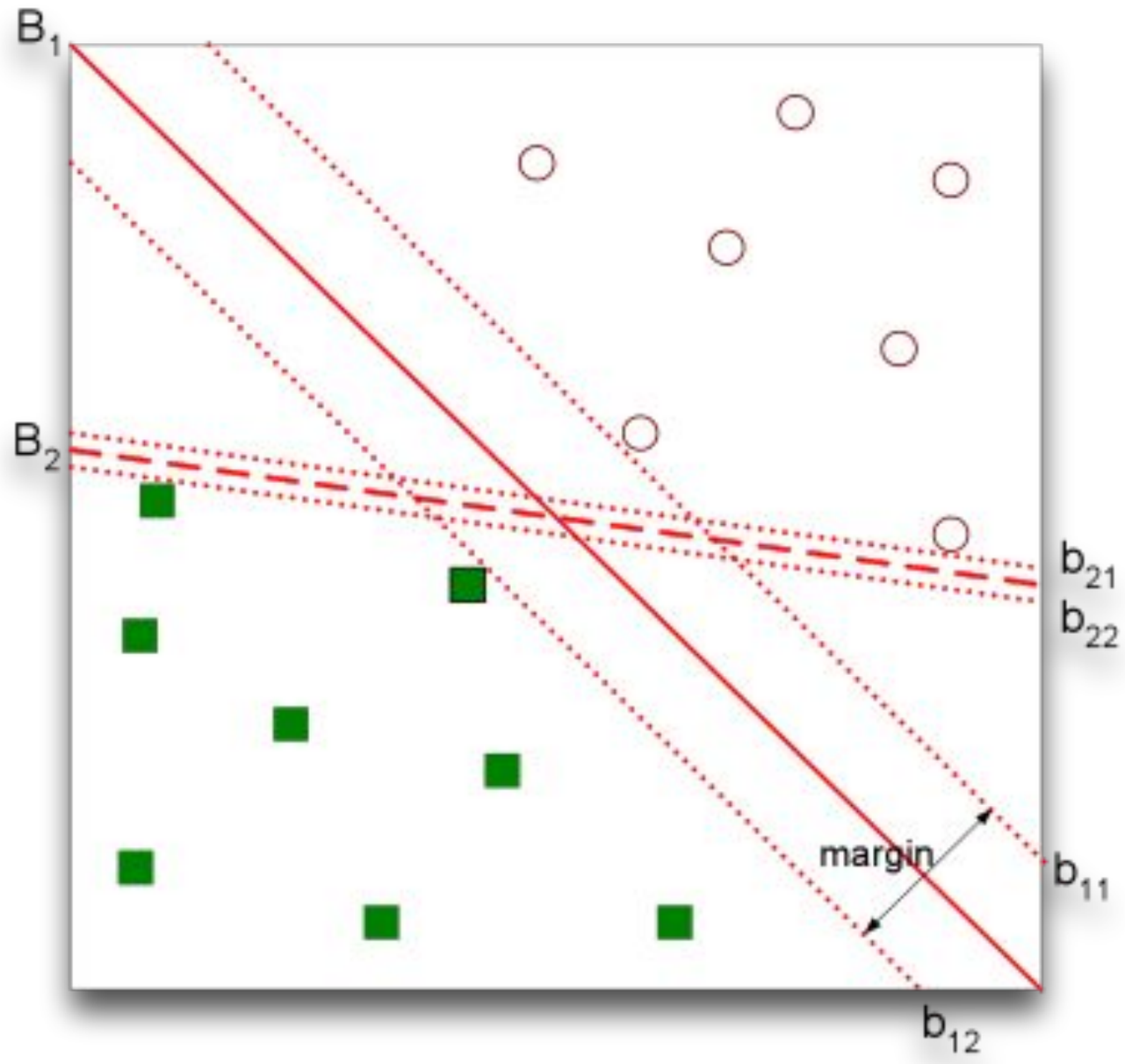
Support Vector Machines



- ¿Cuál es mejor? ¿B1 o B2?

- ¿Cómo definimos qué es mejor?

Support Vector Machines



- Encontrar un hiperplano que maximice el margen de entrenamiento (menores errores de generalización) => B1 es mejor que B2

- 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 $\mathbf{x_i,y_i}$ (i=1,2,...,N), donde x_i es un vector de **d** dimensiones $(x_{i1},x_{i2},...,x_{id})^T$ 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 **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 **w·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.

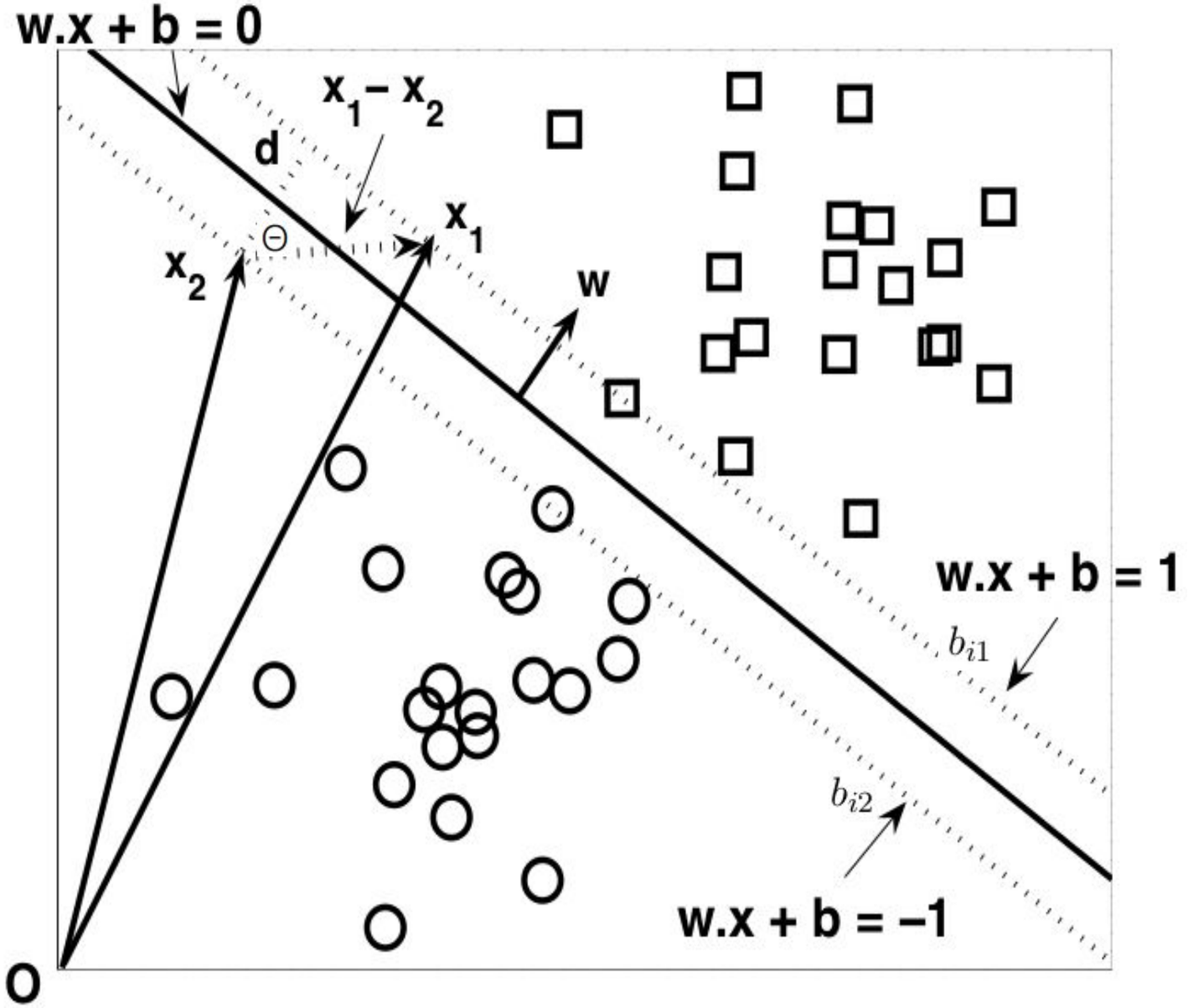
- Por ejemplo, sean x_a y x_b dos puntos localizados en el límite de decisión:

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

- Si restamos las dos ecuaciones obtenemos:

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

- Cómo x_b-x_a es paralelo al límite de decisión, entonces **w** es perpendicular al hiperplano (el producto punto de dos vectores ortogonales es cero).



Support Vector Machines

- Para cualquier cuadrado 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 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 **y** para cualquier ejemplo de test **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).

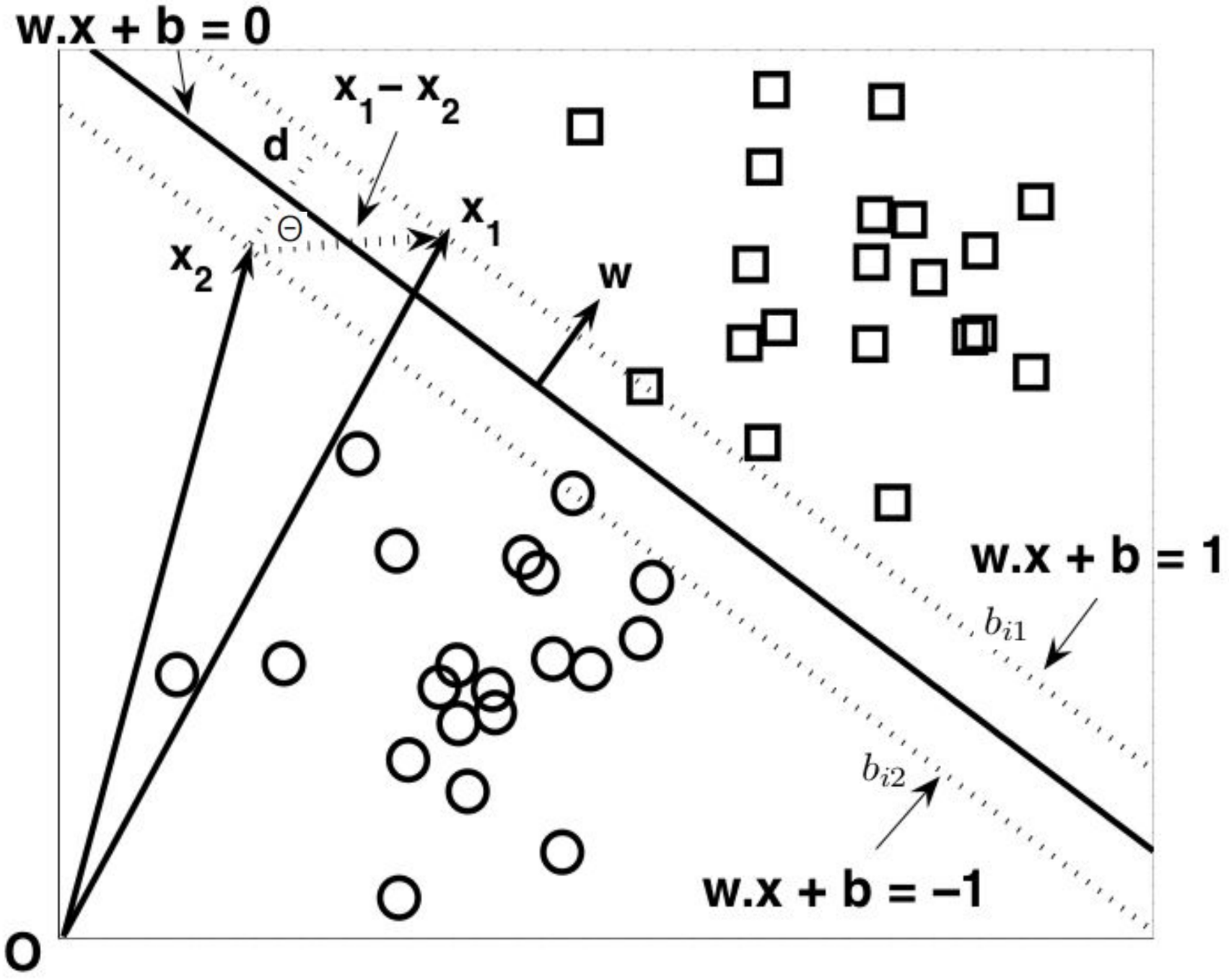
- Sobre x_1 pasa un hiperplano paralelo al límite de decisión llamado b_{i1} y sobre x_2 pasa otro hiperplano paralelo al límite de decisión llamado b_{i2}.

- 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·b) se puede representar como ||a||*||b||*cos(Θ).
- Tenemos entonces que $||w||*||x_1-x_2||*\cos(\Theta)=2$
- Si miramos la figura anterior podemos ver que $||x_1-x_2||*\cos(\Theta) = d$.
- Entonces

$$\|\mathbf{w}\| \times d = 2$$

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

¡ Encontramos una expresión del margen que depende de w !

Aprendiendo 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 :

$$\mathbf{w} \cdot \mathbf{x_i} + b \geq \ 1 \text{ if } y_i = 1,$$
$$\mathbf{w} \cdot \mathbf{x_i} + b \leq -1 \text{ if } y_i = -1.$$

- 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·x+b = 1.

- Mientras que las instancias de la clase y = -1 (los círculos) deben estar situadas en o debajo del hiperplano w·x+b = -1.

Aprendiendo una SVM Lineal

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

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

- 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:

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

$$\text{subject to} \qquad y_i(\mathbf{w} \cdot \mathbf{x_i} + b) \geq 1, \quad i = 1, 2, \ldots, N.$$

- Dado que la función objetivo es cuadrática y las restricciones son lineales para los parámetros 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:

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

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.

Aprendiendo una SVM Lineal

- Si sólo minimizamos el primer término encontraríamos como óptimo **w=0**, un vector nulo.

- Esta solución viola las restricciones del problema porque no existe solución válida para **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 **w** y **b** e igualarlas a cero:

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

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

- 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.

Aprendiendo 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).**

$$\lambda_i \geq 0, \tag{4}$$

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

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

Aprendiendo 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): $\lambda_i \left[y_i (\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \right] = 0.$

- La restricción (5) establece que el multiplicador de Lagrange λ_i debe ser cero a menos que la instancia de entrenamiento x_i satisfaga la ecuación $y_i(w \cdot x_i + b) = 1$.

- 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 w y b, que definen el límite de decisión, dependen sólo de los **vectores de soporte**.

Aprendiendo 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):

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

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

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

Aprendiendo una SVM Lineal

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

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

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

reordenando:

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

vale cero por (3)

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

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).

Aprendiendo 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), $\mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i$, $\lambda_i \left[y_i (\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \right] = 0$, para obtener soluciones válidas para 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 **z** de la siguiente manera:

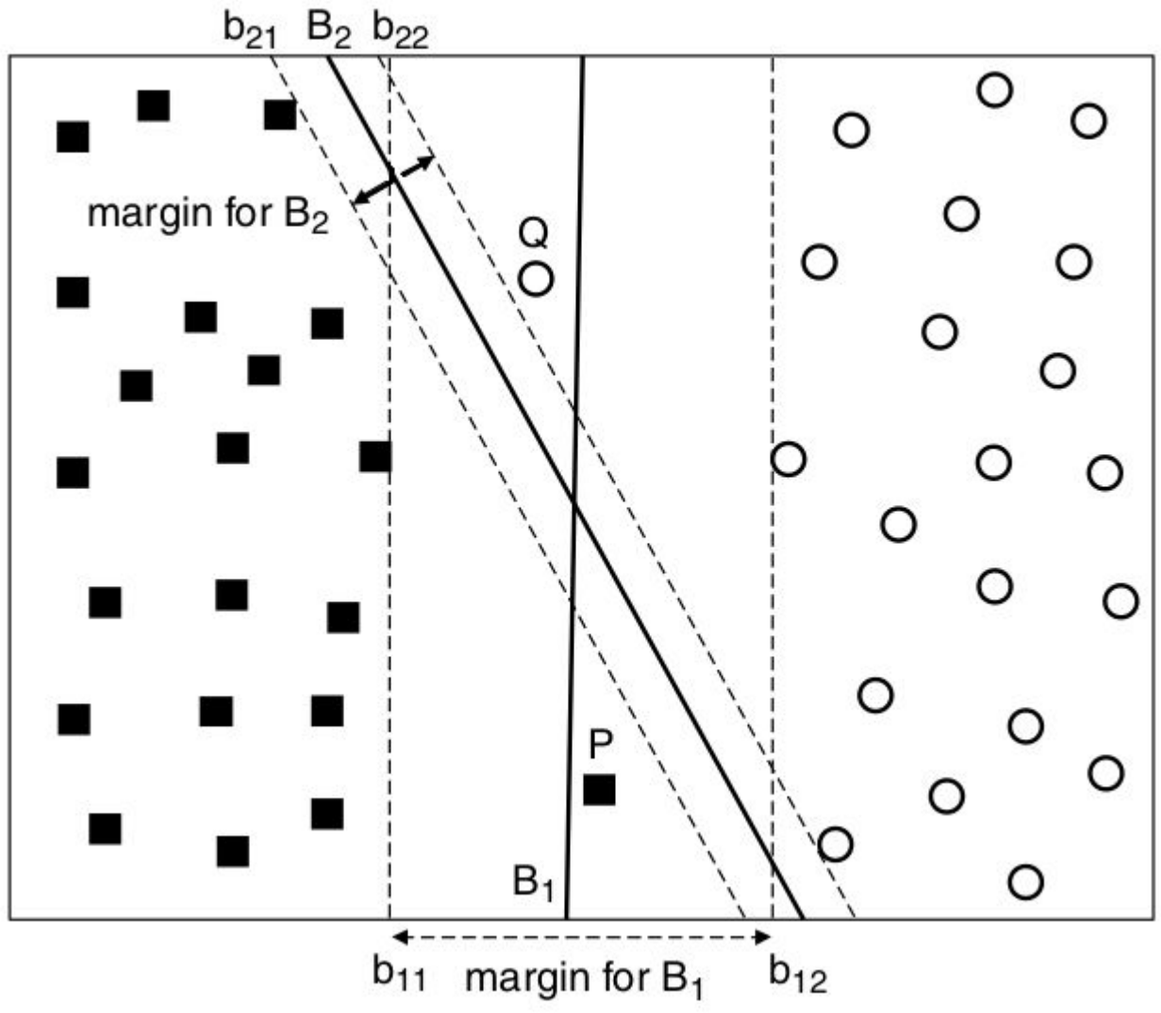
$$f(\mathbf{z}) = sign\big(\mathbf{w} \cdot \mathbf{z} + b\big) = sign\Big(\sum_{i=1}^{N} \lambda_i y_i \mathbf{x_i} \cdot \mathbf{z} + b\Big).$$

- Si f (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

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

- 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:

$$\mathbf{w} \cdot \mathbf{x_i} + b \geq 1 - \xi_i \quad \text{if } y_i = 1,$$
$$\mathbf{w} \cdot \mathbf{x_i} + b \leq -1 + \xi_i \quad \text{if } y_i = -1,$$

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

SVM de margen suave

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

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

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

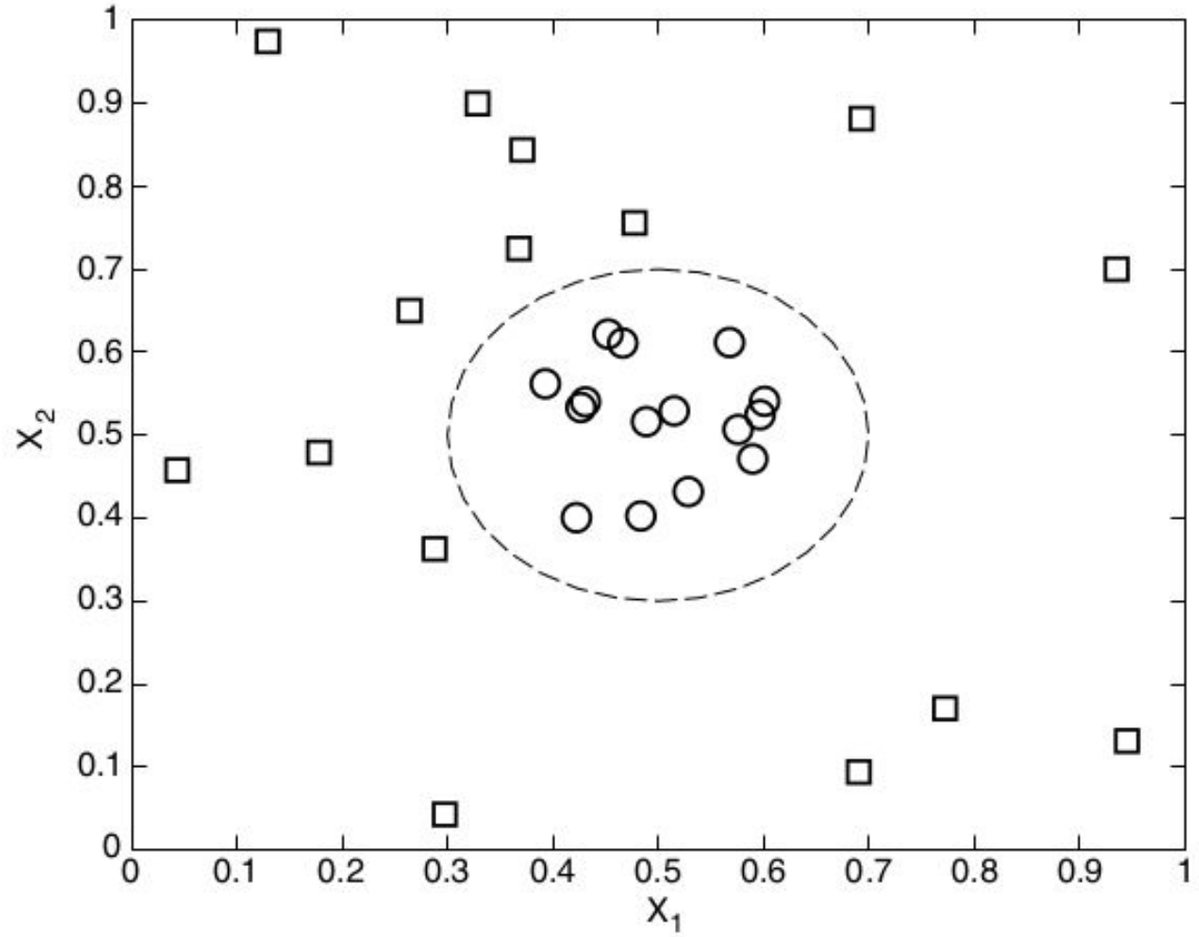
$$\xi_i \geq 0, \quad i = 1, \ldots, m.$$

- Donde $\boldsymbol{\xi_i}$ representa la magnitud del error para 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.

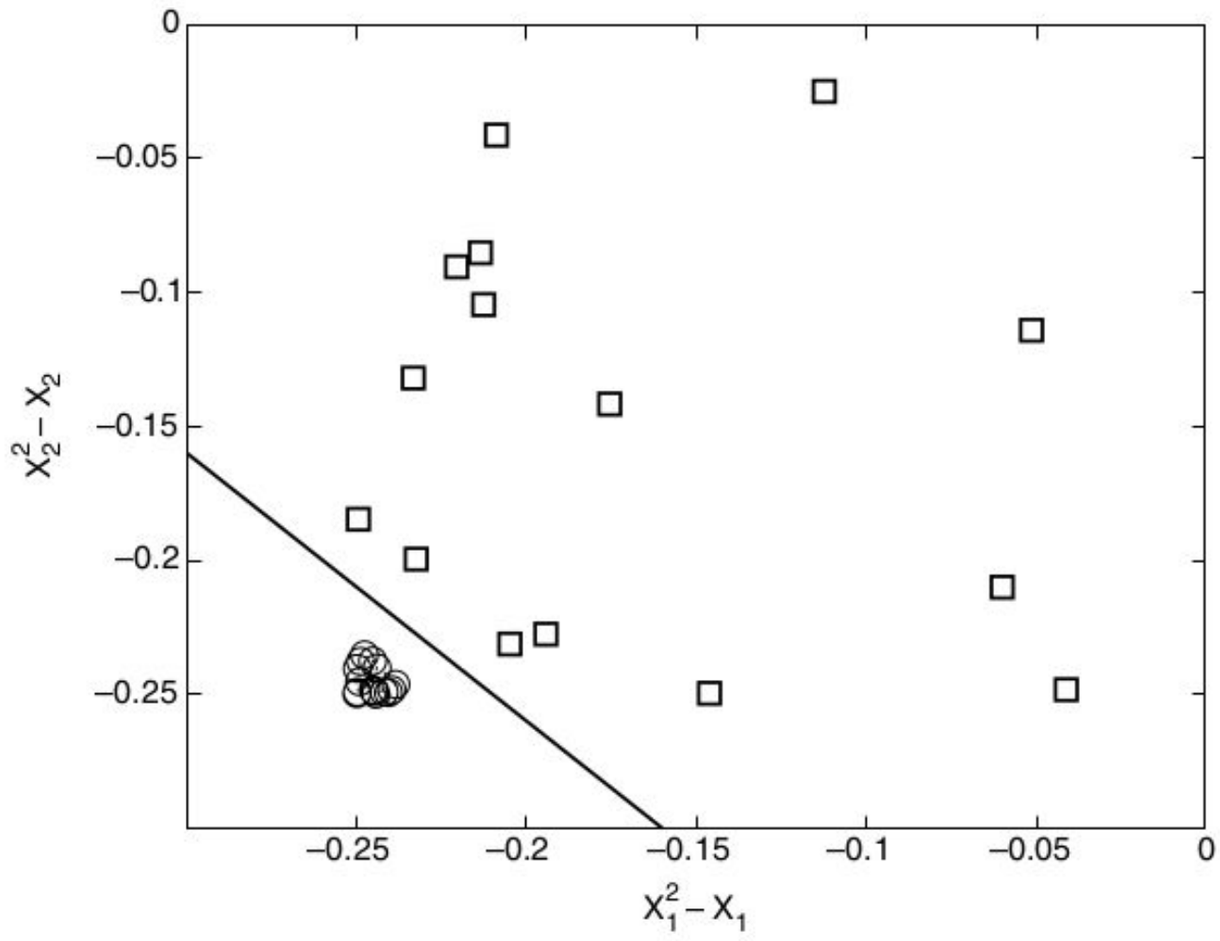
- Supongamos que elegimos la siguiente transformación que transforma de 2 dimensiones a 5 dimensiones:

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

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

$$w_4 x_1^2 + w_3 x_2^2 + w_2\sqrt{2}x_1 + w_1\sqrt{2}x_2 + w_0 = 0.$$

Support Vector Machines No-Lineales

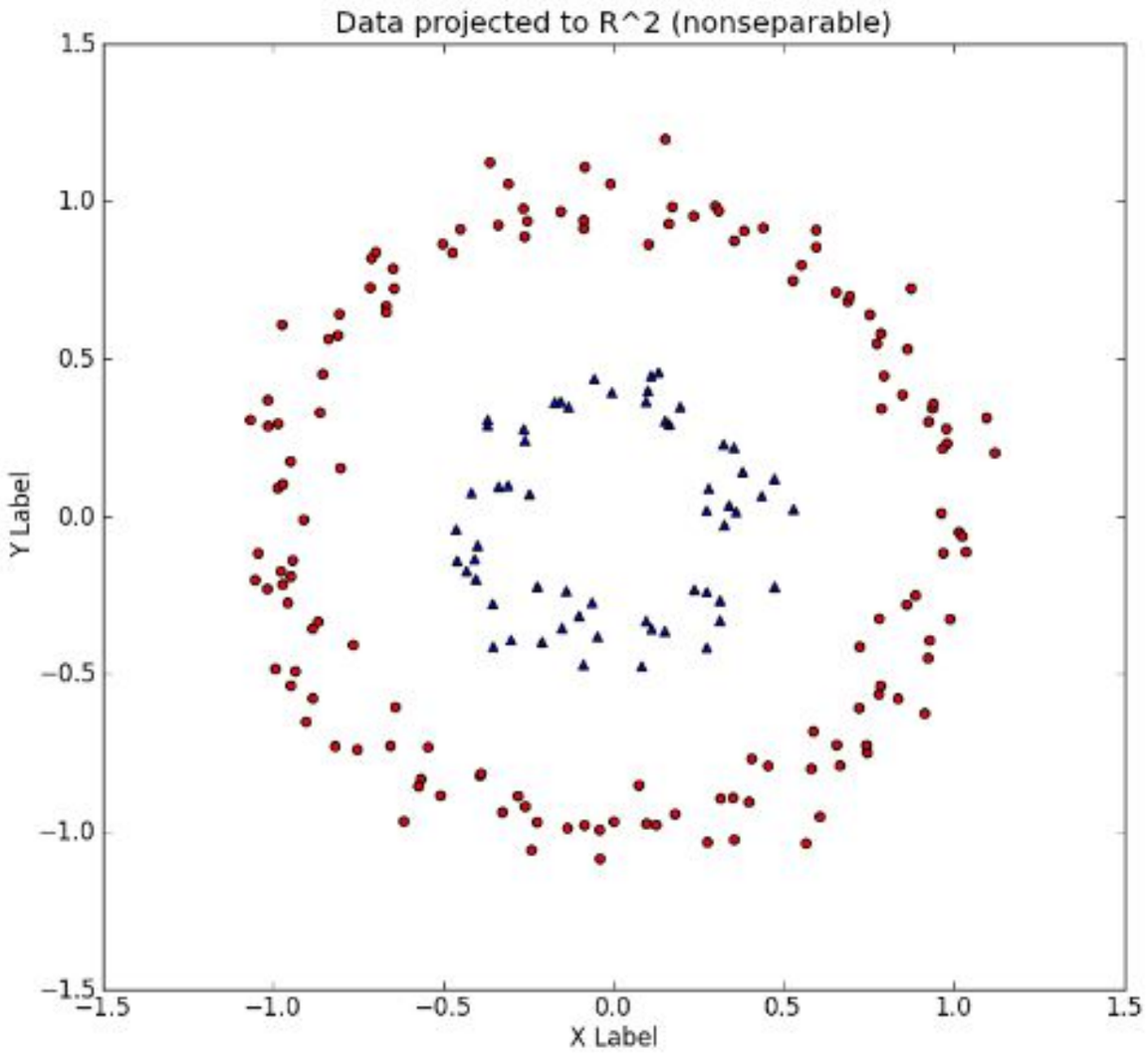


- La figura muestra que en el espacio transformado se puede construir un límite de decisión lineal para separar las clases.

- Un problema potencial de este enfoque es que puede sufrir de la 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



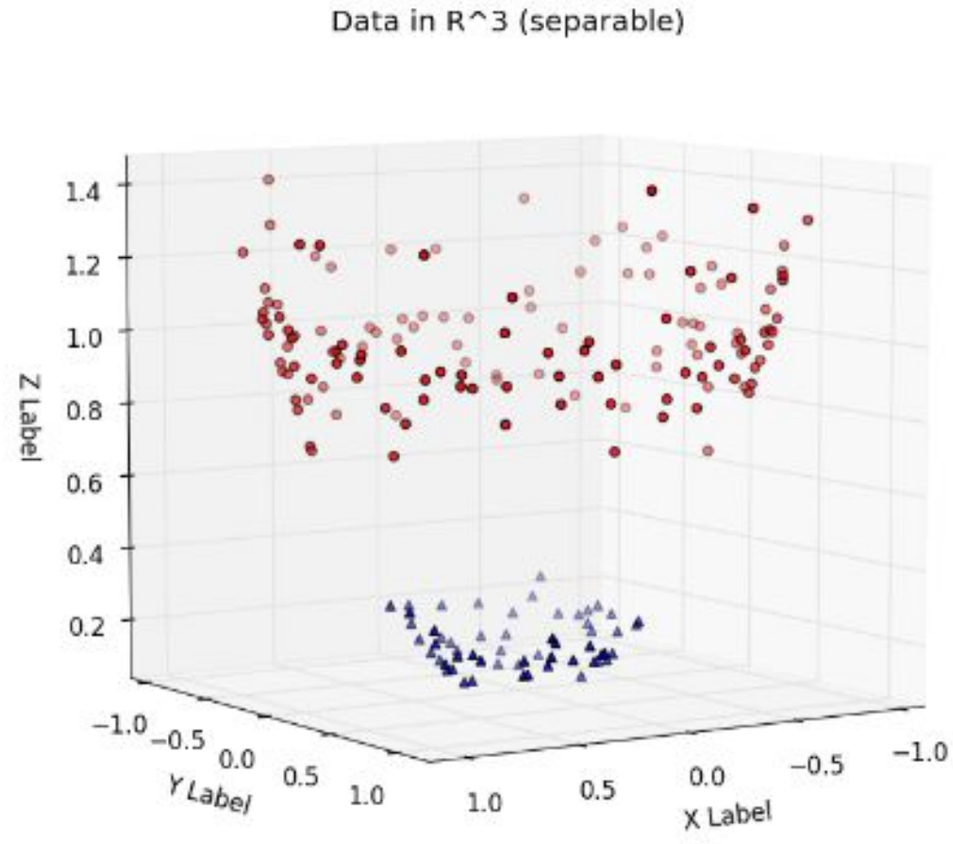


Figure 5: (Left) A dataset in $\mathbb{R}^2$, not linearly separable. (Right) The same dataset transformed by the transformation: $[x_1, x_2] = [x_1, x_2, {x_1}^2 + {x_2}^2]$.

source:

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

Support Vector Machines No-Lineales

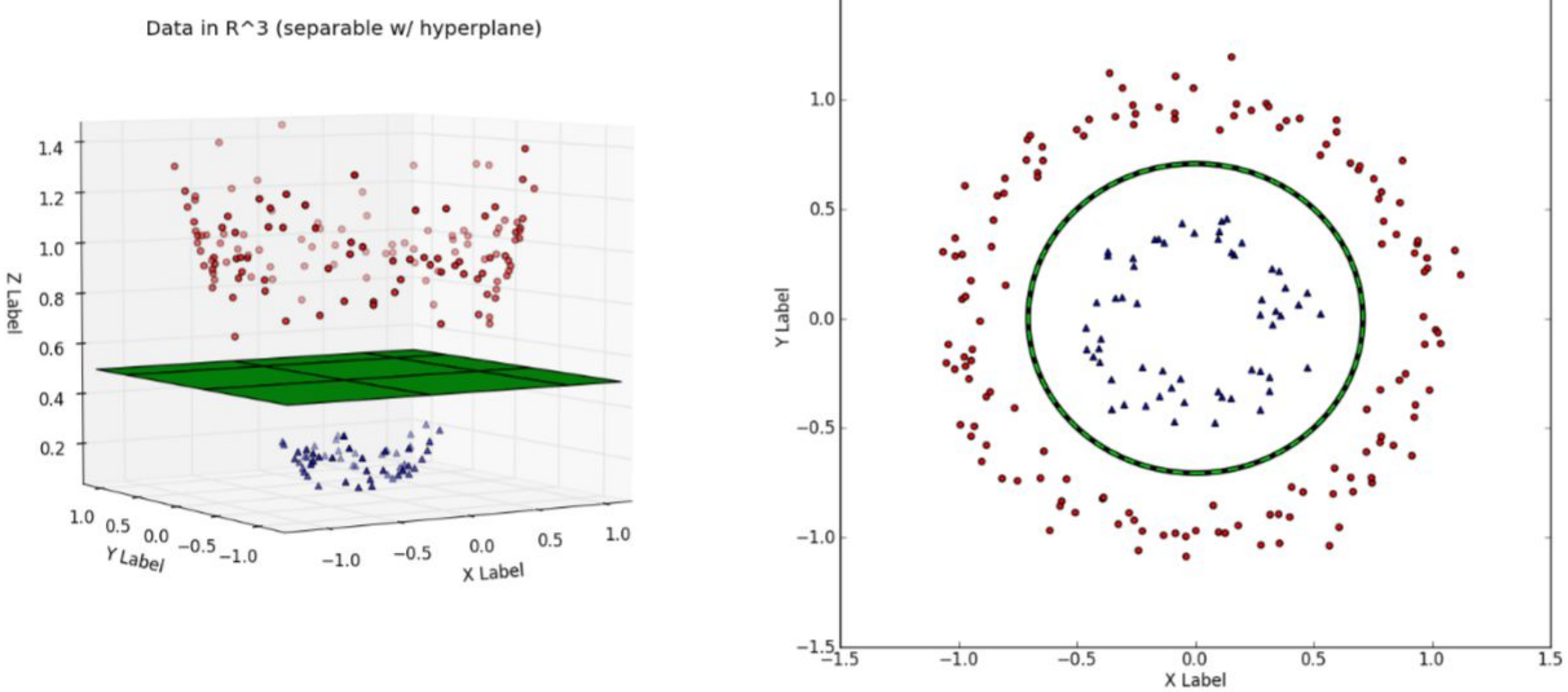


Figure 6: (Left) The decision boundary $\vec{w}$ shown to be linear in $\mathbb{R}^3$. (Right) The decision boundary $\vec{w}$, when transformed back to $\mathbb{R}^2$, is nonlinear.

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 Φ(x) que mapea los datos a una dimensión mayor, podemos redefinir el problema de optimización.

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

$$\text{subject to} \qquad y_i(\mathbf{w} \cdot \Phi(\mathbf{x}_i) + b) \geq 1, \quad i = 1, 2, \ldots, N.$$

- Lo único que hacemos es reemplazar todas las instancia del vector **x** por **Φ(x)**.

- Podemos formular nuevamente el dual y las expresiones que nos permiten encontrar los valores óptimos de w y b.

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

$$\mathbf{w} = \sum_i \lambda_i y_i \Phi(\mathbf{x}_i)$$

$$\lambda_i \{ y_i (\sum_j \lambda_j y_j \Phi(\mathbf{x}_j) \cdot \Phi(\mathbf{x}_i) + b) - 1 \} = 0,$$

Support Vector Machines No-Lineales

- Análogamente, una instancia de prueba **z** puede clasificarse utilizando la siguiente ecuación:

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

- Si nos fijamos en las ecuaciones veremos que tanto para la formulación del dual como para clasificar un ejemplo Φ(x) siempre está acompañada por la función Φ aplicada a otro vector por medio de un producto punto (ej: Φ(x) · Φ(z)).

- El producto punto de dos vectores tiene relación con su similitud (piensen en el coseno del ángulo). Entonces Φ(x) · Φ(z) representa la similitud entre x y z en el espacio transformado.

- Si las transformaciones Φ(x) son de alta dimensión, entonces calcular Φ(x) · Φ(z) será computacionalmente caro.

- El truco del Kernel trick consiste en que para algunas funciones Φ es posible calcular el resultado de Φ(x) · Φ(z) directamente en el espacio original sin necesidad de calcular Φ(x) o Φ(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 Φ(u)·Φ(v):

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

- 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 Φ(x) · Φ(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 **z** de la siguiente forma:

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

- 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

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

- 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: Φ(x) · Φ(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

/ DCCUCHILE