Máquinas de Soporte Vectorial
SVMs

Aprenentatge Automàtic

APA/GEI/FIB/UPC - 2025/2026 1Q

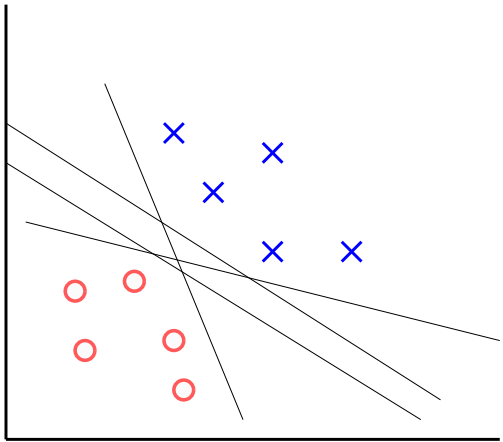
/ Javier Béjar

Introducción

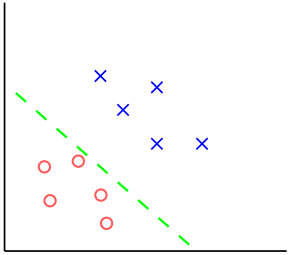
- Para modelos basados en funciones discriminativas, una forma de realizar clasificación es encontrar una superficie de decisión lineal (para problemas binarios)
- La idea es encontrar la ecuación de un plano $w^\top x + w_0$ que divide el conjunto de ejemplos en las clases objetivo así:

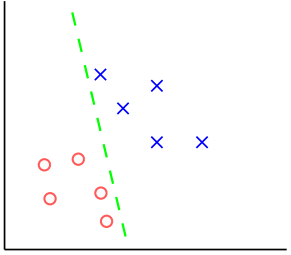
$$w^\top x + w_0 > 0 \Rightarrow clase\ positiva$$
$$w^\top x + w_0 < 0 \Rightarrow clase\ negativa$$

- En realidad cualquier plano que divida los ejemplos puede ser la respuesta al problema
- Eso es lo que pasa con la regresión logística o el perceptrón, cualquiera vale y no hay garantía sobre cuál obtendremos
- La pregunta que nos podemos hacer es ¿qué plano es el mejor?



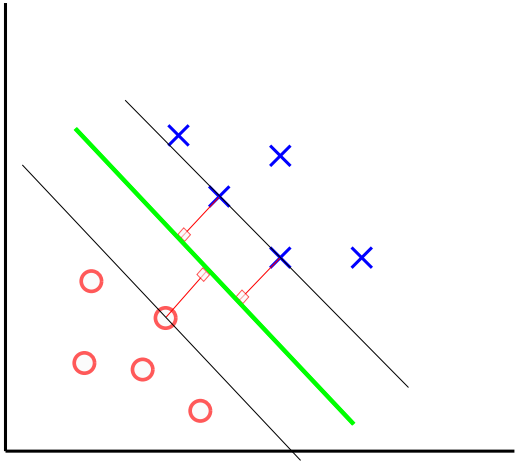
- Algunos planos parecen una mala elección debido a su pobre generalización
- Tenemos que maximizar la probabilidad de clasificar correctamente instancias no vistas
- Queremos minimizar la pérdida esperada de generalización (en lugar de la pérdida esperada empírica)





Clasificación de margen amplio

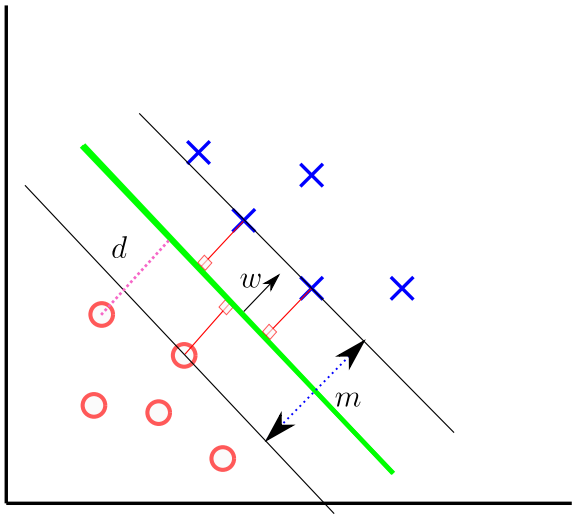
- Maximizar la distancia del separador a los ejemplos parece ser la elección correcta
- Esto significa que solo importan las instancias más cercanas al separador (el resto puede ignorarse)



- Podemos calcular la distancia desde un ejemplo x_n al separador como:

$$d = w^\top x_n + w_0$$

- Los ejemplos más cercanos al separador son vectores de soporte
- El margen (m) del separador es la distancia entre vectores de soporte de clases opuestas



- Si escogemos un punto que está en la frontera del lado negativo del margen x^- y calculamos su proyección x^+ en el margen positivo, se cumple que $x^+ = x^- + r \cdot w$
- Estos dos puntos también cumplen que $w \cdot x^+ + w_0 = +1$ y $w \cdot x^- + w_0 = -1$
- Tenemos que :

$$\begin{aligned} w \cdot (x^- + r \cdot w) + w_0 &= +1 \\ r \cdot ||w||^2 + w \cdot x^- + w_0 &= +1 \\ r \cdot ||w||^2 - 1 &= +1 \\ r = \frac{2}{||w||^2} \Rightarrow m = ||x^+ - x^-|| &= ||r \cdot w|| = \frac{2}{||w||} \end{aligned}$$

- Por lo tanto maximizar el margen es lo mismo que minimizar la norma de los pesos

- Esto permite definir el problema de aprender los pesos como un problema de optimización
- Lo definiremos como un problema de minimización sobre $||w||^2$ que es equivalente, esto se conoce como la formulación primal
- Dado un conjunto de ejemplos $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$ con etiquetas de clase $y_i \in \{-1, +1\}$ (para este problema esto es más conveniente que $\{0, 1\}$)
- La frontera de decisión que clasifica los ejemplos correctamente cumple

$$y_n(w^\top x_n + w_0) \geq 1, \quad \forall n$$

- El problema de optimización es:

  Minimizar $\frac{1}{2}||w||^2$ sujeto a $y_n(w^\top x_n + w_0) \geq 1, \quad \forall n$

- Podemos reescribir las restricciones como:

$$1 - y_n(w^\top x_n + w_0) \leq 0, \quad \forall n$$

- Este es un problema de optimización cuadrática
- Esto significa que los parámetros forman una superficie paraboloide y existe un óptimo global
- Se puede aplicar cualquier resolvedor estándar de programación cuadrática para obtener la solución

- Podemos resolver este problema de optimización restringida usando multiplicadores de Lagrange (α)

$$\ell(y_n, \hat{y}_n) = \frac{1}{2} w^\top w + \sum_{n=1}^{N} \alpha_n (1 - y_n (w^\top x_n + w_0))$$

- Si calculamos la derivada de ℓ con respecto a w y w_0 y las igualamos cero:

$$w + \sum_{n=1}^{N} \alpha_n (-y_n) x_n = 0 \Rightarrow w = \sum_{n=1}^{N} \alpha_n y_n x_n$$

y

$$\sum_{n=1}^{N} \alpha_n y_n = 0$$

Si sustituimos w en el lagrangiano ℓ

$$\begin{aligned}
\ell &= \frac{1}{2}\sum_{i=1}^{N}\alpha_i y_i x_i^\top \sum_{j=1}^{N}\alpha_j y_j x_j + \sum_{i=1}^{N}\alpha_i\left[1 - y_i(\sum_{j=1}^{N}\alpha_j y_j x_j^\top x_i + w_0)\right] \\
&= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j x_i^\top x_j + \sum_{i=1}^{N}\alpha_i - \sum_{i=1}^{N}\alpha_i y_i \sum_{j=1}^{N}\alpha_j y_j x_j^\top x_i \\
&- w_0\sum_{i=1}^{N}\alpha_i y_i \quad (y\ \ dado\ \ que\ \ \sum_{i=1}^{N}\alpha_i y_i = 0) \\
&= -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j x_i^\top x_j + \sum_{i=1}^{N}\alpha_i \quad (reorganizando\ \ terminos)
\end{aligned}$$

- Este problema se conoce como problema dual
- Esta formulación solo depende de las α_i
- Ambos problemas están vinculados, dado w podemos calcular α_i y viceversa
- Esto significa que podemos resolver uno en lugar del otro
- Ahora este problema es un problema de maximización ($\alpha_i \geq 0$)
- El usar una formulación u otra depende de la relación entre la dimensionalidad de los datos y el número de ejemplos, ya que el número de restricciones depende de ese valor en cada problema

- La formulación del problema dual es

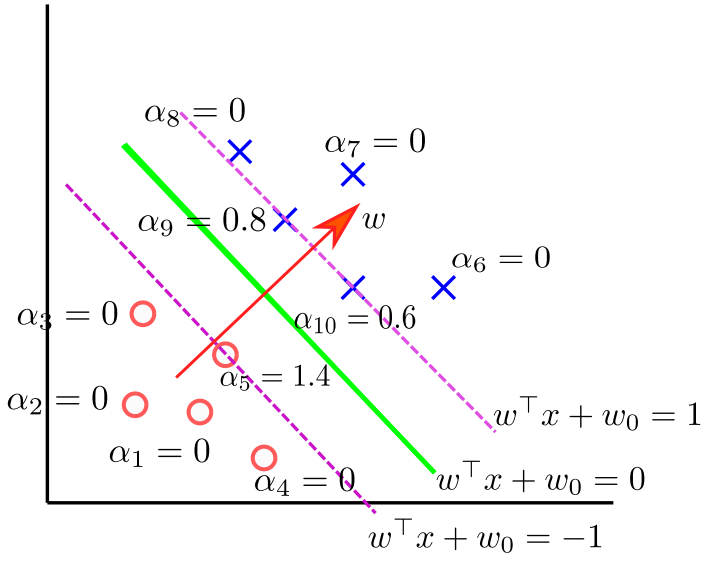
  Maximizar $\sum_{n=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^\top x_j$ sujeto a

  $\sum_{i=1}^{N} \alpha_i y_i = 0, \quad \alpha_i \geq 0 \ \ \forall i$

- También es un problema de programación cuadrática (QP)
- Podemos obtener w por

$$w = \sum_{n=1}^{N} \alpha_i y_i x_i$$

- w_0 se puede obtener de un vector de soporte positivo (x_{psv}) sabiendo que $w^\top x_{psv} + w_0 = 1$


$\alpha_8 = 0$
$\alpha_7 = 0$
$\alpha_9 = 0.8$
w
$\alpha_6 = 0$
$\alpha_3 = 0$
$\alpha_{10} = 0.6$
$\alpha_5 = 1.4$
$\alpha_2 = 0$
$\alpha_1 = 0$
$\alpha_4 = 0$
$w^\top x + w_0 = 1$
$w^\top x + w_0 = 0$
$w^\top x + w_0 = -1$

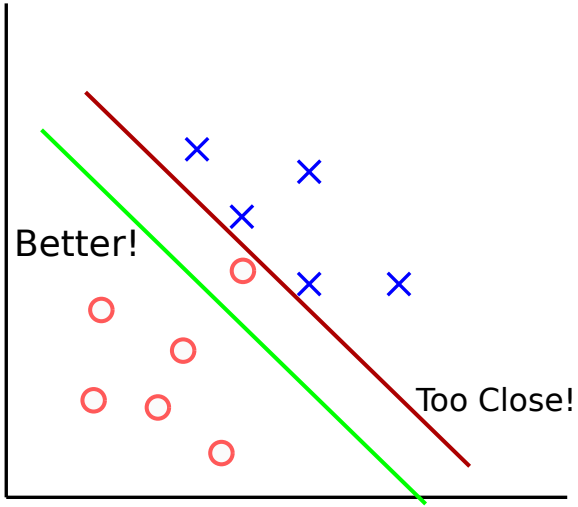
- Muchos de los α_i son cero
  - w es una combinación lineal de algunos ejemplos
  - Obtenemos una representación *dispersa* (sparse) de los datos (compresión de datos)
- Los ejemplos x_i con α_i distinto de cero son los vectores de soporte (SV)
- Para clasificar un nuevo ejemplo z solo tenemos que calcular

$$signo(w^\top z + w_0) = signo(\sum_{\forall i \in SV} \alpha_i y_i x_i^\top z + w_0)$$

- Esto significa que w no necesita calcularse explícitamente

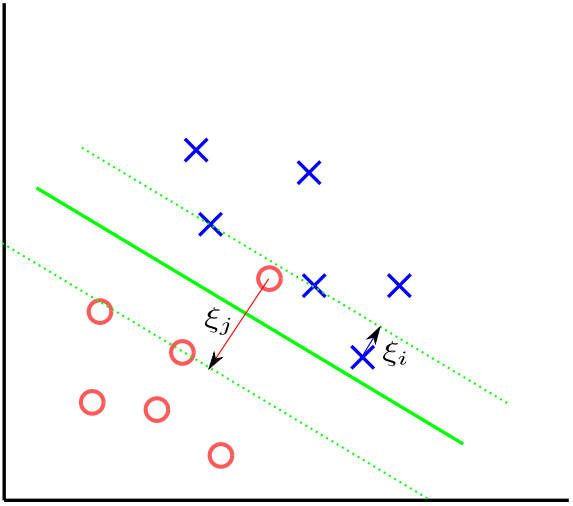
Clasificación de margen blando

- Este algoritmo funciona bien para problemas linealmente separables
- A veces los datos tienen errores, y queremos ignorarlos para obtener una mejor solución
- A veces los datos no son linealmente separables
- Podemos obtener mejores separadores lineales siendo menos estrictos



- Queremos ser permisivos con ciertos ejemplos, permitiendo que su clasificación por el separador diverja de la clase real
- Estos se pueden obtener agregando al problema lo que se llama **variables de holgura** (slack variables) (ξ_i)
- Estas variables representan la desviación de los ejemplos respecto del margen
- Al hacer esto, estamos relajando el margen, estamos usando un margen **blando** (soft)

- Estamos permitiendo un error en la clasificación basado en el separador $w^\top x + w_0$
- Los valores de ξ_i aproximan el número de errores de clasificación



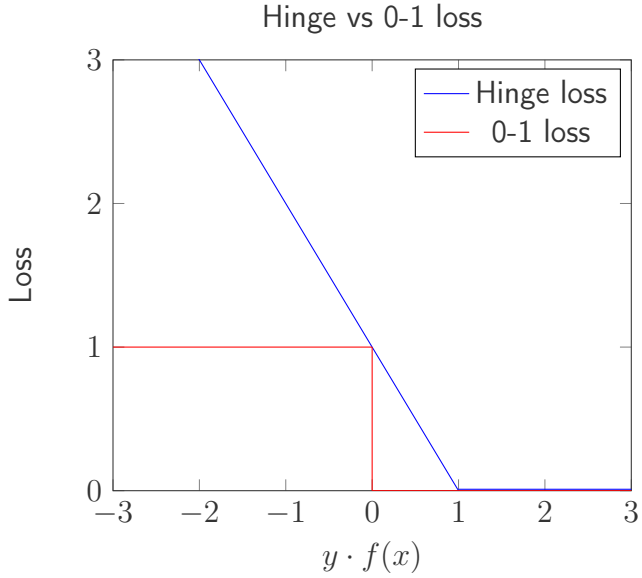
- Para minimizar el error, podemos minimizar $\sum_i \xi_i$ introduciendo variables de holgura a las restricciones del problema:

$$y_n(w^\top x_n + w_0) \geq 1 - \xi_n$$
$$\xi_n \geq 0$$

  Eso es equivalente a usar

$$\xi_n = \max(0, 1 - y_n(w^\top x_n + w_0))$$

- Esto define el error blando también conocido como **pérdida en bisagra** (Hinge loss)
- Si $\xi_n = 0$ el ejemplo x_n está correctamente clasificado
- Si $0 < \xi_n < 1$ el ejemplo x_n está correctamente clasificado pero dentro del margen
- Si $\xi_n \geq 1$ el ejemplo x_n está mal clasificado


Hinge vs 0-1 loss
Hinge loss
0-1 loss
Loss
3
2
1
0
−3
−2
−1
0
1
2
3
$y \cdot f(x)$

- Necesitamos introducir estas variables de holgura en el problema original, tenemos: Minimizar $\frac{1}{2}||w||^2 + C\sum_{n=1}^{N}\xi_n$ sujeto a $y_n(w^\top x_n + w_0) \geq 1 - \xi_n, \quad \forall n, \xi_n \geq 0$

- Observad que el primer término es un término de regularización y el segundo es el error/pérdida
- Ahora tenemos un parámetro adicional $C \geq 0$ que es la compensación entre el error y el margen
- Tendremos que ajustar este parámetro usando la selección del modelos, por ejemplo, usando la validación cruzada

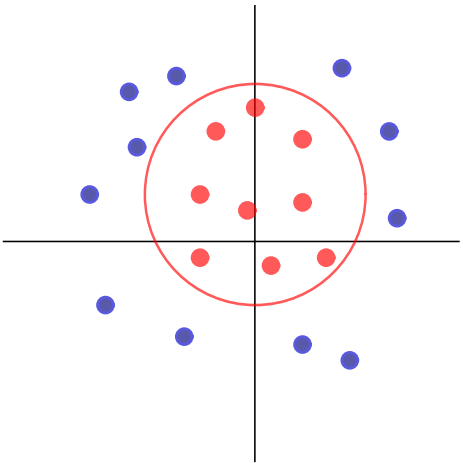
- Realizando la transformación al problema dual obtenemos lo siguiente: Maximizar $\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^\top x_j$ sujeto a

  $\sum_{i=1}^{N} \alpha_i y_i = 0, \quad C \geq \alpha_i \geq 0 \ \ \forall i$

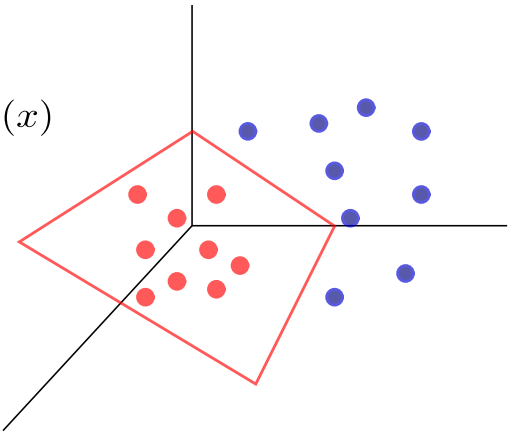
- Podemos recuperar la solución como $w = \sum_{\forall i \in SV} \alpha_i y_i x_i$
- Este problema es muy similar al caso linealmente separable, excepto que hay un límite superior C en los valores de α_i
- Este también es un problema de QP

Clasificación no lineal - Kernels

- Hasta ahora solo hemos considerado clasificadores de margen amplio que usan una frontera lineal
- Para tener un mejor rendimiento, tenemos que ser capaces de obtener fronteras no lineales
- La idea es transformar el espacio de los atributos de entrada a un espacio de más dimensiones usando una función $\phi(x)$
- Este nuevo espacio se llama **espacio de características** (feature space)
- Esta transformación hará que las operaciones lineales en el espacio de características sean equivalentes a operaciones no lineales en el espacio de entrada



$$\phi : x \Rightarrow \varphi(x)$$



- El problema XOR no es linealmente separable en su definición original, pero podemos hacerlo linealmente separable si agregamos una nueva característica $x_1 \cdot x_2$

x_1	x_2	$x_1 \cdot x_2$	x_1 XOR x_2
0	0	0	1
0	1	0	0
1	0	0	0
1	1	1	1

- La función lineal $f(x) = 2x_1x_2 - x_1 - x_2 + 1$ clasifica correctamente todos los ejemplos

- Trabajar directamente en el espacio de características puede ser costoso
- Tenemos que crear explícitamente este espacio y operar en él
- Podemos necesitar infinitas características para hacer que un problema sea linealmente separable
- Podemos usar lo que se llama el **truco del kernel** (kernel trick)

- El problema definido por una SVM solo necesita el producto interno de los ejemplos ($x_i^\top x_j$)
- Si podemos definir cómo calcular este producto en el espacio de características, entonces no es necesario construirlo explícitamente
- Hay muchas operaciones geométricas que se pueden expresar como productos internos, por ejemplo ángulos o distancias
- Podemos definir una **función de kernel** como:

$$K(x_i, x_j) = \phi(x_i)^\top \phi(x_j)$$

- Supongamos un espacio de características definido como:

$$\phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2 x_2^2, \sqrt{2}x_1 x_2)$$

- Un producto interno en este espacio de funciones es:

$$\left\langle \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right), \phi\left(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}\right) \right\rangle = (1 + x_1 y_1 + x_2 y_2)^2$$

- Entonces, podemos definir una función kernel para calcular productos internos en este espacio como:

$$K(x, y) = (1 + x_1 y_1 + x_2 y_2)^2 = (1 + x^\top y)^2$$

y estamos usando solo las características del espacio de entrada

- Dado un conjunto de ejemplos $X = \{x_1, x_2, \dots x_n\}$, y una función simétrica $k(x, z)$ podemos definir la matriz del kernel K como:

$$K = \phi^\top \phi = \begin{pmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) \\ \cdots & \cdots & \cdots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{pmatrix}$$

- Si K es una matriz semidefinida positiva y simétrica, entonces $k(x, z)$ es una función de kernel
- La justificación es que si K es una matriz semidefinida positiva entonces se puede descomponer en:

$$K = V \Lambda V^\top$$

- V se puede interpretar como una proyección en el espacio de características

- Una función de kernel satisface el Teorema de Mercer
- Dada una función **simétrica** $K(x, y)$

$$K : [a, b] \times [a, b] \to \mathbb{R}$$

K se dice que es positiva semidefinida si y solo si

$$\sum_{i=1}^{n} \sum_{j=1}^{n} K(x_i, x_j) c_i c_j = \sum_{i=1}^{n} \sum_{j=1}^{n} \phi(x_i)^\top \phi(x_j) c_i c_j \geq 0$$

para todas las secuencias finitas de puntos $x_1, \ldots, x_n$ de $[a, b]$ y todos los posibles números reales $c_1, \ldots, c_n$

Podemos definir el kernel lineal como $K(x, y) = x \cdot y$

Podemos demostrar que para cualquier secuencia de puntos $x_1, \ldots, x_n$ y todos los números reales $c_1, \ldots, c_n$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} K(x_i, x_j)c_i c_j = \sum_{i=1}^{n}\sum_{j=1}^{n} \phi(x_i)^\top \phi(x_j)c_i c_j \geq 0$$

dado que

$$\sum_{i=1}^{n}\sum_{j=1}^{n} x_i^\top x_j c_i c_j = \left(\sum_{i=1}^{n} x_i c_i\right)\left(\sum_{j=1}^{n} x_j c_j\right) = \left(\sum_{i=1}^{n} x_i c_i\right)^2 \geq 0$$

Alternativamente

$$C^\top X^\top XC = (C^\top X^\top)(XC) = (XC)^\top (XC) \geq 0$$

Dadas dos funciones del kernel K_1 y K_2, también son kernels

- $cK_1(x, y)$ (donde c es una constante positiva)
- $f(x)K_1(x, y)f(y)$
- $K_1(x, y) + K_2(x, y)$
- $K_1(x, y) \times K_2(x, y)$
- $\exp(K_1(x, y))$
- $poly(K_1(x, y))$ siempre que los coeficientes no sean negativos

- Ejemplos de funciones que cumplen esta condición son:
  - Kernel polinómico de grado d:

$$K(x, y) = (x^\top y + c)^d$$

  - Función gaussiana con ancho σ: (kernel de función de base radial, RBF)

$$K(x, y) = \exp(-||xy||^2/2\sigma^2)$$

  - Tangente hiperbólica sigmoide con parámetros k y c (solo para ciertos valores de los parámetros):

$$K(x, y) = \tanh(kx^\top y + c)$$

- Las funciones de kernel también se pueden interpretar como una función de similitud
- Una función de similitud se puede transformar en un kernel dado que cumple la condición de matriz definida positiva

Podemos probar que una función es un kernel transformándola en una composición de kernels conocidos

El kernel cuadrático se define como:

$$K(x, y) = (x^\top y + 1)^2 = (x^\top y)^2 + 2x^\top y + 1$$

Sabemos que un polinomio de un kernel es un kernel si los coeficientes son positivos, entonces $(x^\top y)^2$ es un polinomio del kernel lineal.

También una constante multiplicada por un kernel es un kernel, por lo que $2x^\top y$ es el kernel lineal multiplicado por dos El kernel constante también es un kernel (prueba para el lector)

La suma de kernels es un kernel

- Debido a la introducción de la función de kernel, el problema de optimización debe modificarse:

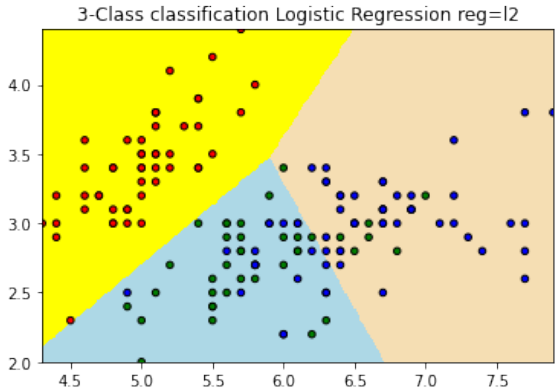
  Maximizar $\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$

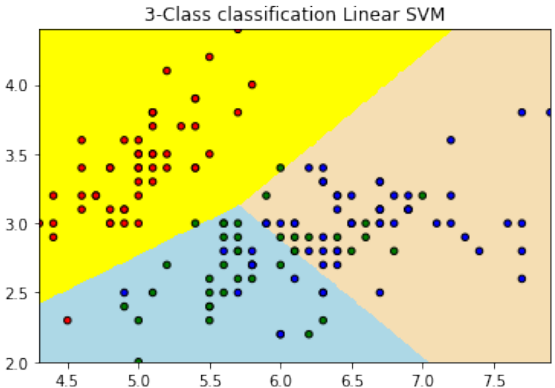
  sujeto a $\sum_{i=1}^{N} \alpha_i y_i = 0, \;\; C \geq \alpha_i \geq 0 \;\; \forall i$

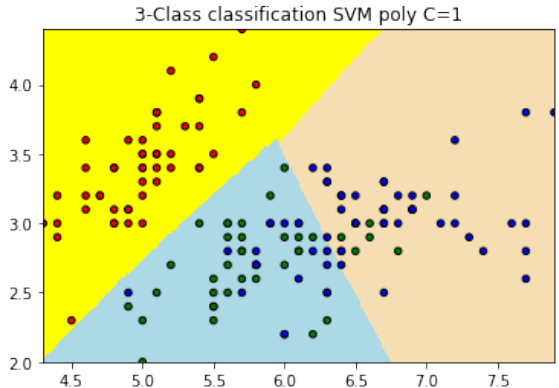
- Para clasificar nuevos ejemplos:

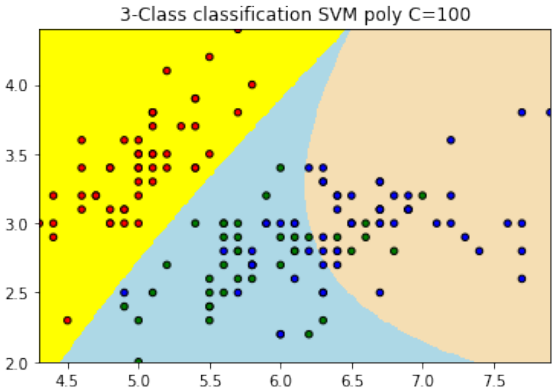
$$f(x_n) = signo \left(\sum_{\forall i \in SV} \alpha_i y_i K(x_i, x_n) + w_0 \right)$$

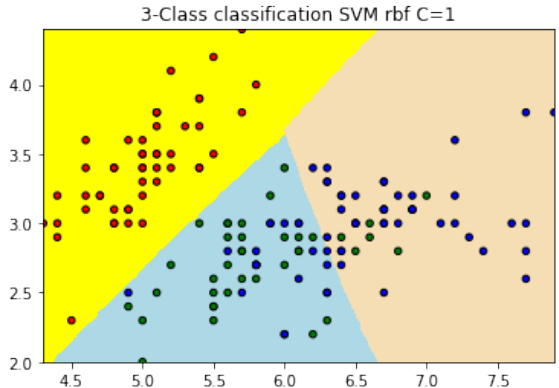
- Dado que el entrenamiento de SVM solo necesita el valor de $K(x_i, x_j)$, no hay restricciones sobre cómo se representan los ejemplos
- Solo tenemos que definir el kernel como una similitud entre ejemplos
- Podemos definir funciones de similitud para diferentes representaciones
  - Cadenas, secuencias (subsecuencias comunes)
  - Conjuntos ($\frac{A \cap B}{A \cup B}$)
  - Grafos/Árboles (subárboles/subgráfos comunes, caminos comunes)
  - Documentos (proporción entre palabras comunes y número total de palabras)
  - ...

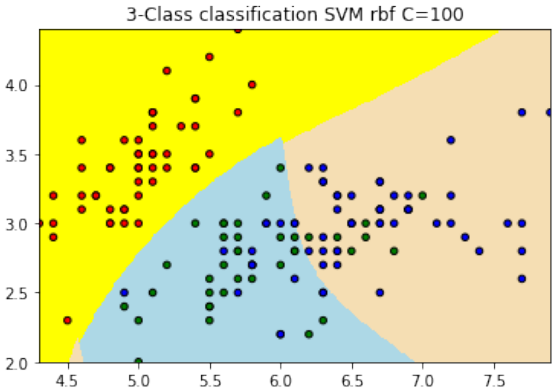

3-Class classification Logistic Regression reg=l2
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5


3-Class classification Linear SVM
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5


3-Class classification SVM poly C=1
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5


3-Class classification SVM poly C=100
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5


3-Class classification SVM rbf C=1
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5

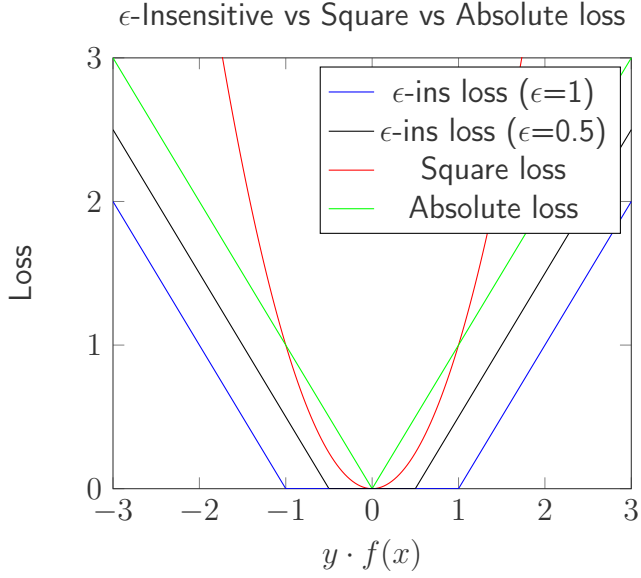

3-Class classification SVM rbf C=100
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5

Regresión SVM

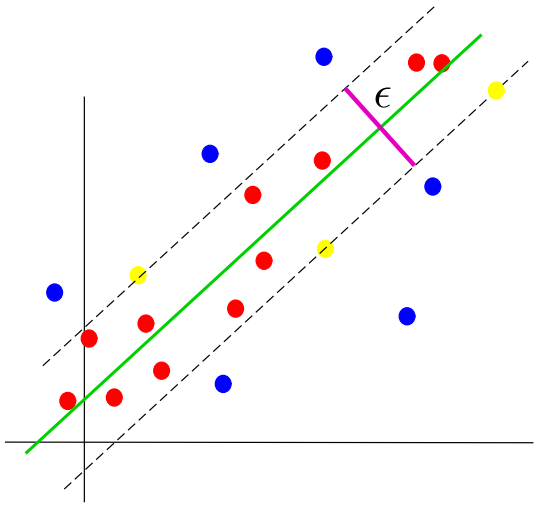
- Las SVM también se pueden usar para regresión con una formulación similar
- La idea es obtener un modelo que use solo un número limitado de ejemplos para definir la función de regresión (vectores de soporte)
- Para hacer esto, la función de pérdida se cambia del error cuadrático a la función de error/pérdida ϵ-insensible (ϵ-insensitive)

$$\ell(y_n\, \hat{y}_n) = \begin{cases} 0 & si \;\; |y_n - \hat{y}_n| < \epsilon \\ |y_n - \hat{y}_n| - \epsilon & en\ otro\ caso \end{cases}$$

- Donde ϵ es un hiper parámetro que define el margen alrededor de la función lineal que no cuenta como error


ϵ-Insensitive vs Square vs Absolute loss
ϵ-ins loss (ϵ=1)
ϵ-ins loss (ϵ=0.5)
Square loss
Absolute loss
Loss
3
2
1
0
−3
−2
−1
0
1
2
3
$y \cdot f(x)$

- Geométricamente estamos definiendo un tubo (en el espacio de características) alrededor de la función lineal en el que las instancias dentro de él no se utilizan en la regresión



- En este caso, se definen dos tipos de variables de holgura para tener en cuenta el error a ambos lados de la función lineal.

Minimizar $\frac{1}{2}||w||^2 + C\sum_{n=1}^{N}(\xi_n^+ + \xi_n^-)$ sujeto a

$$\begin{aligned} y_n - (w^\top x + w_0) &\leq \epsilon + \xi_n^+ \\ (w^\top x + w_0) - y_n &\leq \epsilon + \xi_n^- \\ \xi_n^+, \xi_n^- &\geq 0 \end{aligned}$$

- Eso se puede transformar en la formulación dual que define la regresión como una combinación ponderada de los ejemplos que son vectores de soporte
- Usando el truco del kernel, funciones no lineales se pueden ajustar en el espacio original, que serán lineales en el espacio de características

Este notebook muestra la superficie de predicción de diferentes Kernels y el efecto de los hiperparámetros de SVM para clasificación y regresión

Interpretabilidad/Explicabilidad

- Podemos calcular con los vectores de soporte los pesos del hiperplano separador, pero solo es útil para la SVM lineal
- Los pesos de la SVM lineal dan la importancia de la dimensión para determinar la clase o el valor de la regresión
- El uso de kernels convierte a las SVM en una caja negra, por lo que se deben aplicar métodos independientes del modelo para la importancia de la característica o una explicación de ejemplo

Este notebook muestra un ejemplo de SVM con una explicación de los modelos