Support Vector Machine

SVM

Dr. Mauricio Toledo-Acosta
mauricio.toledo@unison.mx

Diplomado Ciencia de Datos con Python

Table of Contents

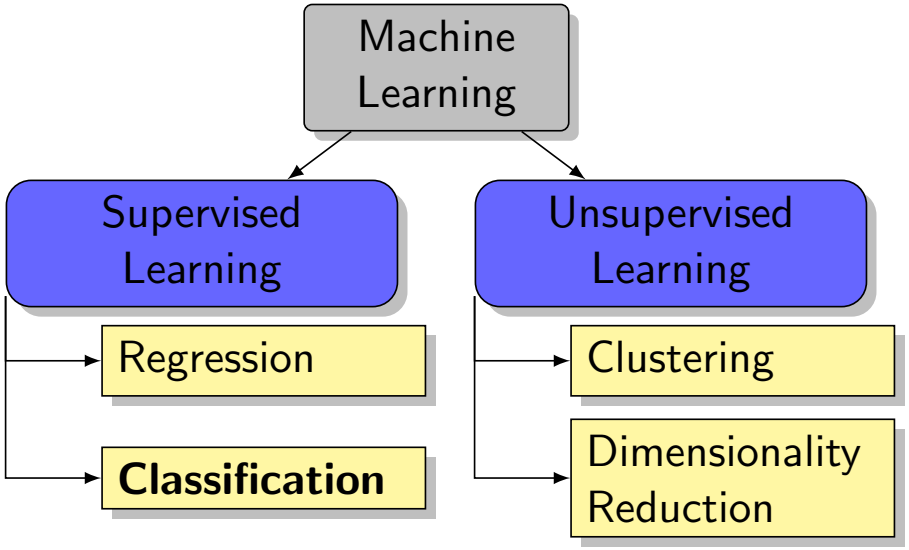

Machine Learning
Supervised Learning
Unsupervised Learning
Regression
Classification
Clustering
Dimensionality Reduction

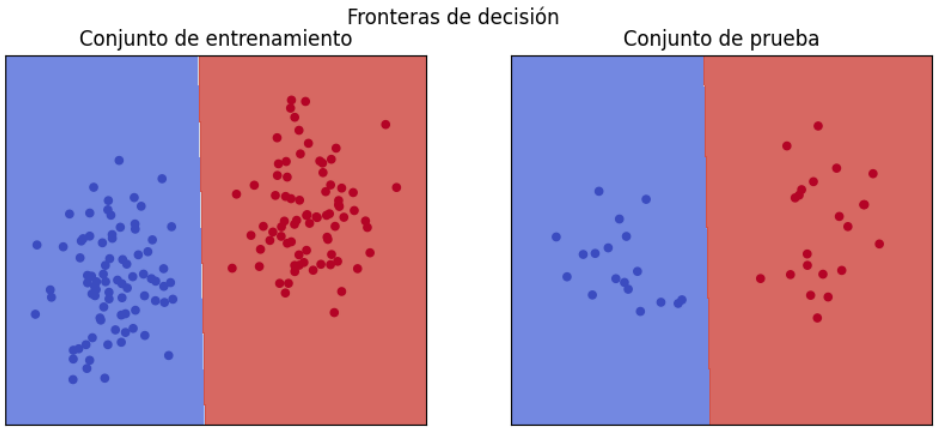
Table of Contents

Introducción

Support Vector Machine

Modelo supervisado de clasificación binaria que busca encontrar una frontera de decisión óptima que separe las clases de puntos.

Su principal objetivo es encontrar el hiperplano que mejor separa las clases en un espacio de características.

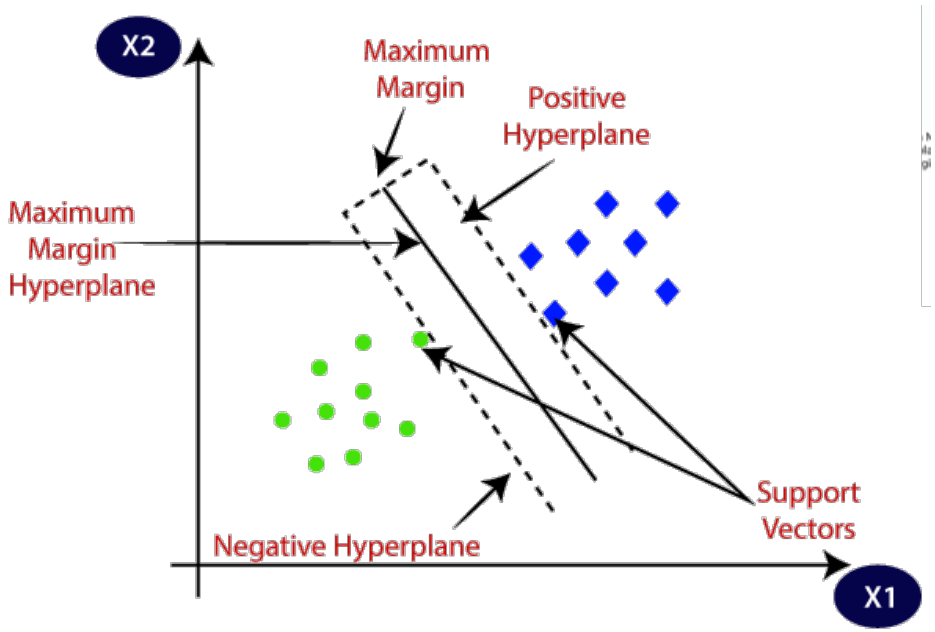


La implementación de scikit-learn soporta clasificación multiclase usando los enfoques OVR y OVO.

Introducción

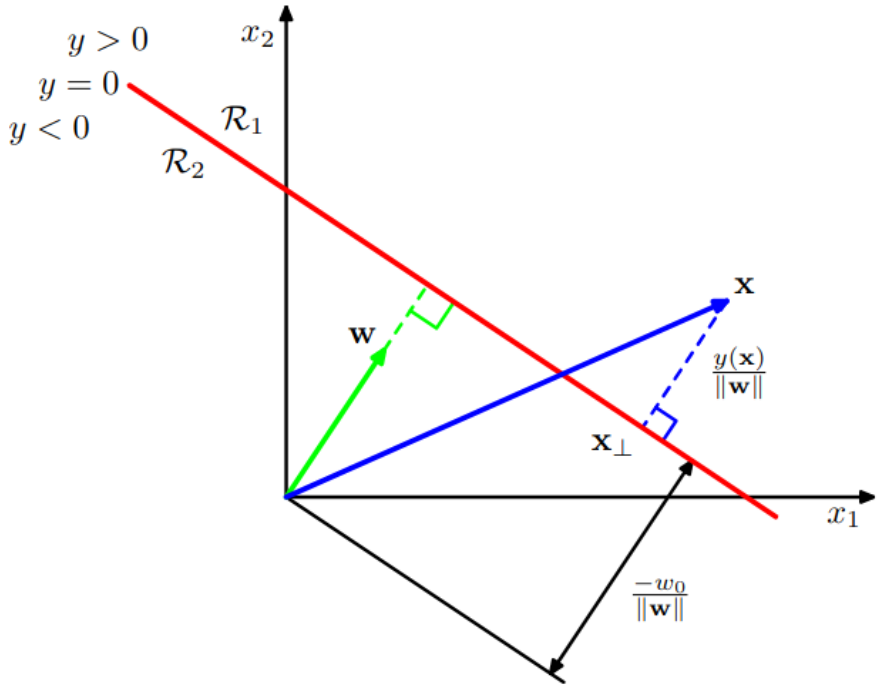
El algoritmo SVM busca los elementos de cada clase que son más similares a los de la otra clase. Estos son los *vectores de soporte*, sobre estos vectores el algoritmo busca encontrar el mejor hiperplano que los separa.

La distancia de los vectores de soporte a la frontera de decisión es el *margen*.



El modelo lineal de clasificación

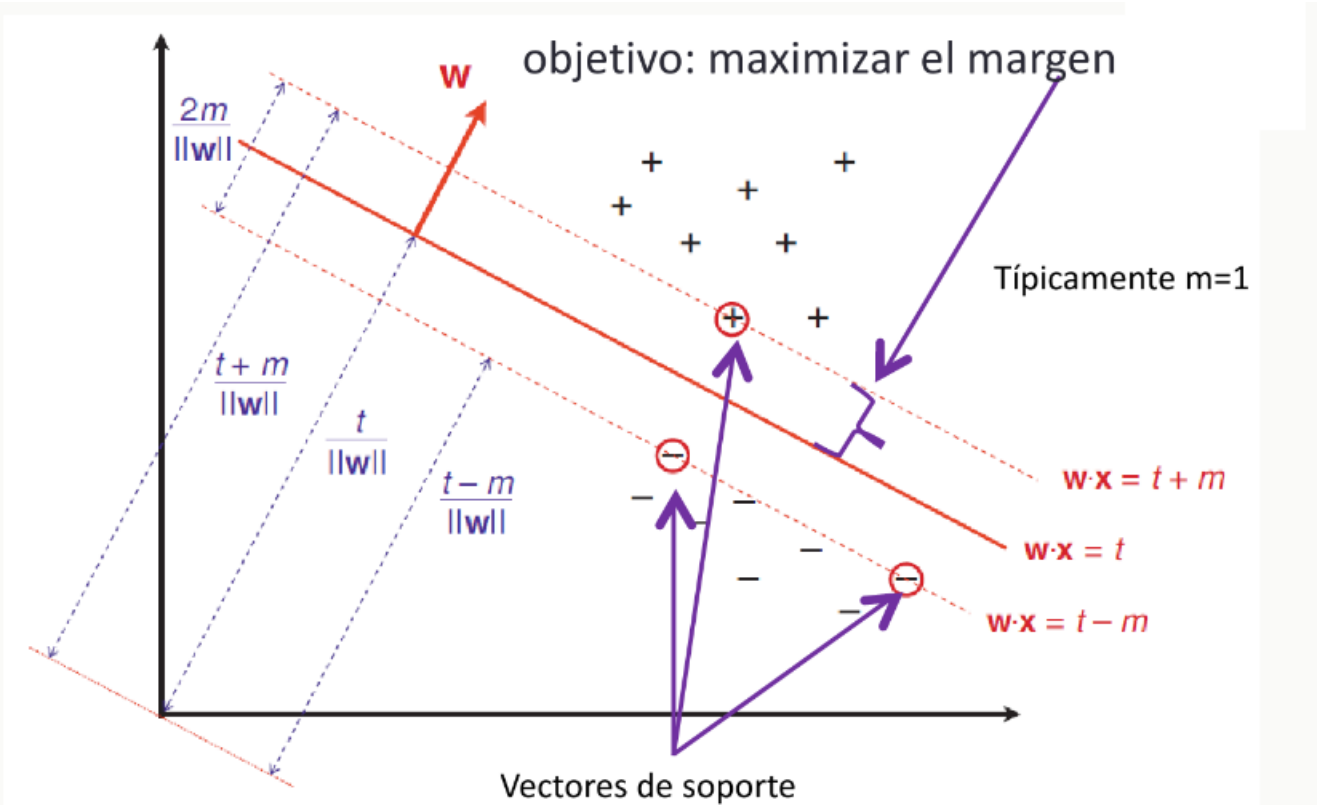
Los puntos x que satisfacen $y(x) = w^T \cdot x + w_0 = 0$ forman la frontera de decisión (FD), la cual divide al espacio de datos en dos regiones.



Buscamos determinar la FD usando la forma normal de la recta (vector normal y distancia al origen).

SVM de margen duro

Analicemos el caso con datos linealmente separables. La FD está definida por $g(x) = w^T \cdot x - t = 0$. Queremos encontrar una FD que maximice el margen $\frac{2}{\|w\|}$. Es decir, queremos minimizar $\|w\|$.



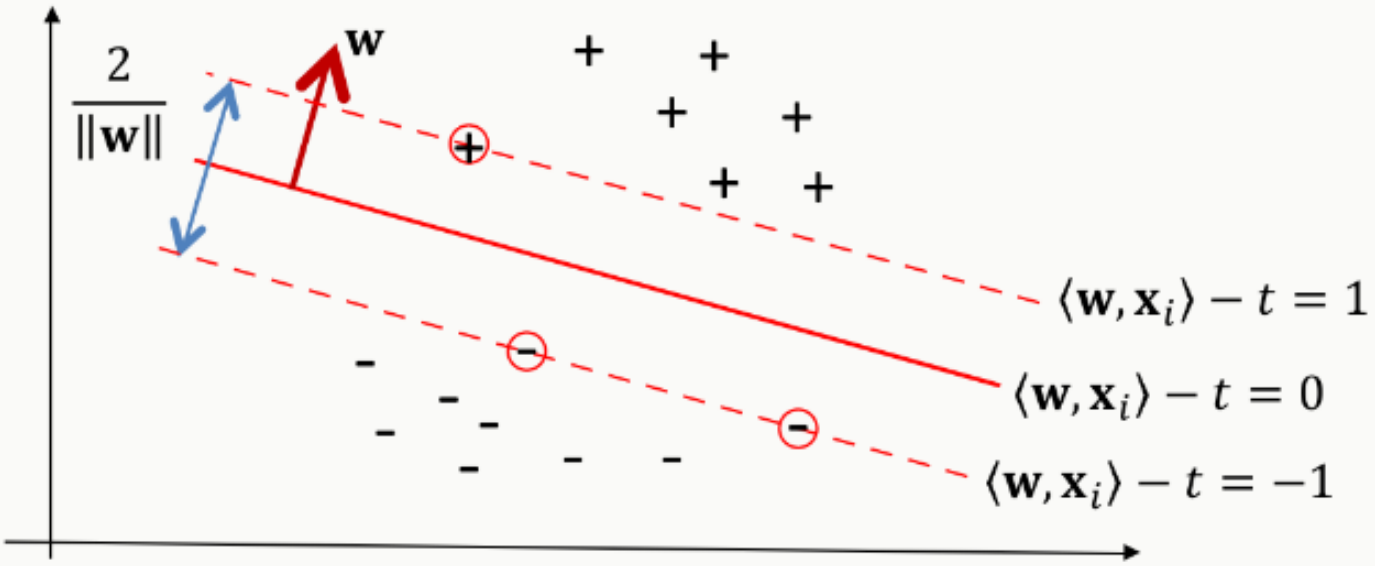
Planteamiento del problema

- Nuestro objetivo es:

$$\mathbf{w}^*, t^* = \underset{\mathbf{w},t}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{w}\|^2$$

- Sujeto a las siguientes N restricciones:

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle - t) \geq 1, \qquad 1 \leq i \leq N$$



Planteamiento del problema

Por lo anterior, usaremos el método de los multiplicadores de Lagrange.

- Definimos el lagrangiano

Minimizar **Maximizar**

$$\mathcal{L}_P(\mathbf{w}, t, \alpha_1, \cdots, \alpha_N) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{N} \alpha_i (yi\ (\langle \mathbf{w}, \mathbf{x}_i \rangle - t\ ) - 1)$$

$$= \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{N} \alpha_i y_i \langle \mathbf{w}, \mathbf{x}_i \rangle + \sum_{i=1}^{N} \alpha_i y_i t + \sum_{i=1}^{N} \alpha_i$$

$$= \frac{1}{2}\langle \mathbf{w}, \mathbf{w} \rangle - \left\langle \mathbf{w}, \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i \right\rangle + t\left(\sum_{i=1}^{N} \alpha_i y_i\right) + \sum_{i=1}^{N} \alpha_i$$

- Para un t óptimo $\partial_t \mathcal{L}_P = 0 \Longrightarrow \sum_{i=1}^{N} \alpha_i y_i = 0$
- Para pesos óptimos $\partial_{\mathbf{w}} \mathcal{L}_P = 0 \Longrightarrow \mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$

Planteamiento del problema

- *Reinsertando estas expresiones en $\mathcal{L}_P$ obtenemos $\mathcal{L}_D$ el lagrangiano del problema dual:*

$$\mathcal{L}_D(\alpha_1, \cdots, \alpha_N) = -\frac{1}{2}\left\langle \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i \, , \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i \right\rangle + \sum_{i=1}^{N} \alpha_i$$

$$= -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \, \alpha_j y_i y_j \left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle + \sum_{i=1}^{N} \alpha_i$$

Planteamiento del problema

- *El problema de optimización dual es el siguiente:*

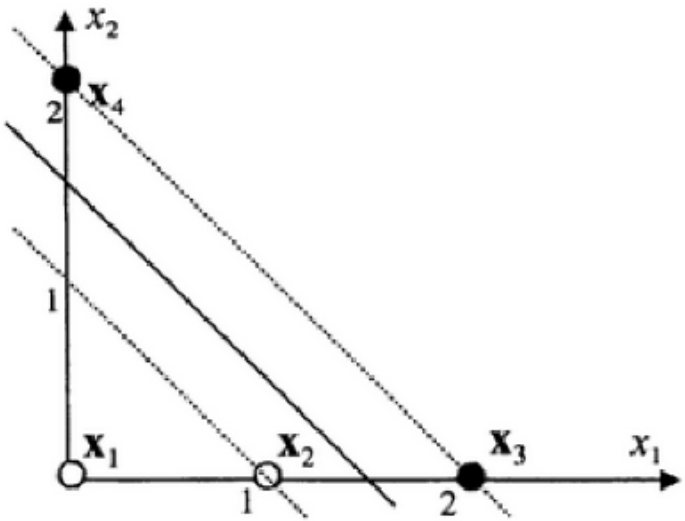
$$\alpha_1^*, \cdots, \alpha_N^* = \underset{\alpha_1, \cdots, \alpha_N}{\operatorname{argmax}} -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle + \sum_{i=1}^{N} \alpha_i$$

- *Sujeto a las restricciones:*

$$\alpha_i > 0 \ , \ 1 \leq i \leq N \qquad \text{y} \qquad \sum_{i=1}^{N} \alpha_i y_i = 0$$

Ejemplo

- Encuentra W óptimo para este problema: X1=[0,0] X2=[1,0] para la clase (+1) y X3=[2,0] y X4=[0,2] para la clase (-1)



Ejemplo

$$\mathcal{L}_D(\alpha_1, \cdots, \alpha_N) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

$$\mathcal{L}_D = (\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4) - \frac{1}{2}(\alpha_2{}^2 - 4\alpha_2\alpha_3 + 4\alpha_3{}^2 + 4\alpha_4{}^2)$$

Diferenciando con respecto a los α's y utilizando la restricción $\sum_{i=1}^{N} \alpha_i y_i = 0$ obtenemos:

$$\begin{cases} \alpha_1 + \alpha_2 - \alpha_3 - \alpha_4 = 0 \\ \alpha_2 - 2\alpha_3 = 1 \\ -2\alpha_2 + 4\alpha_3 = 1 \\ 4\alpha_4 = 1 \end{cases}$$

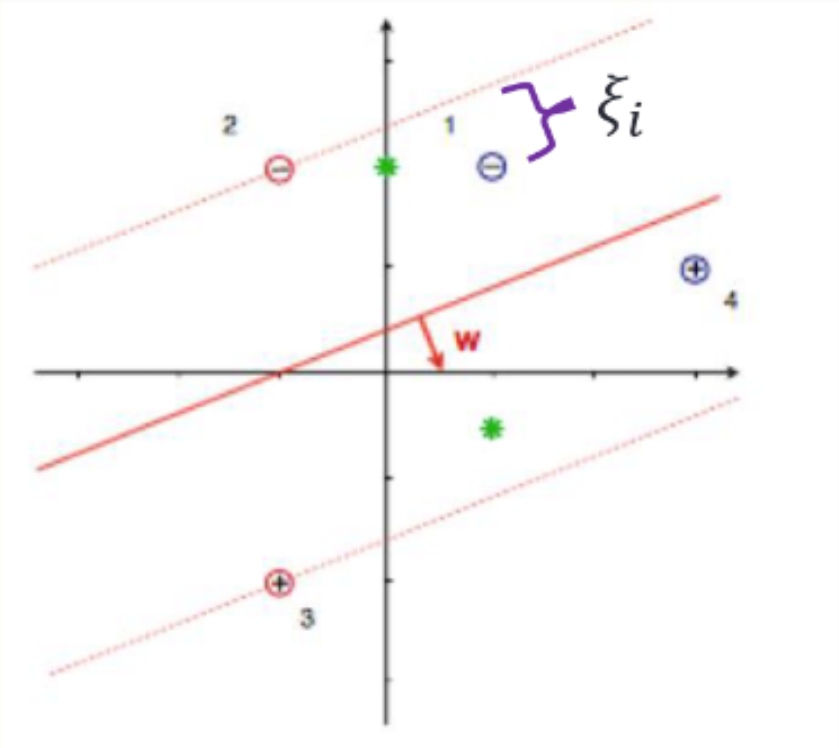
de donde: $\alpha_1 = 0,\ \alpha_2 = 1,\ \alpha_3 = \frac{3}{4},\ \alpha_4 = 1/4$

Aplicando: $\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$, finalmente obtenemos

$$\mathbf{w} = \begin{bmatrix} -1/2 \\ -1/2 \end{bmatrix}, w_0 = 3/4 \text{ y } d(x) = 3 - 2x_1 - 2x_2 = 0$$

SVM de margen suave

- La SVM anterior no funciona con datos no-separables
- Introducimos variables de holgura ξ_i para cada dato de entrada, lo que les permite a algunos de ellos estar dentro del margen, o incluso del lado equivocado de la frontera de decision.



SVM de margen suave

$$\mathbf{w}^*, t^*, \xi_i^* = \underset{\mathbf{w},t,\xi_i}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N}\xi_i$$

$$\text{sujeto a } y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle - t) \geq 1 - \xi_i \text{ y } \xi_i \geq 0, 1 \leq i \leq N$$

- C es un parámetro definido por el usuario que balancea la maximización del margen contra la minimización de las variables de holgura:
  - *un valor alto de C significa que los errores de margen son altamente costosos,*
  - *un valor pequeño de C permite más errores de margen con tal de hacer mas grande el margen.*
- Si permitimos más errores de margen necesitamos menos vectores de soporte, por lo tanto C controla la 'complejidad' de la SVM y por ello se le denomina el *parámetro de complejidad*.

SVM de margen suave

- Buscamos soluciones mediante el nuevo Lagrangiano:

$$\mathcal{L}(\mathbf{w}, t, \xi_i, \alpha_i, \beta_i) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i\big(yi\,(\langle \mathbf{w}, \mathbf{x}_i\rangle - t\,) - (1-\xi_i)\big) - \sum_{i=1}^{N}\beta_i\xi_i$$

$$= \mathcal{L}(\mathbf{w}, t, \alpha_i\,) + \sum_{i=1}^{N}(C - \alpha_i - \beta_i)\xi_i$$

- La solución óptima es tal que $\partial_{\xi_i}\mathcal{L} = 0 \Longrightarrow$ el término añadido desaparece en el problema dual.
- Además, puesto que α_i y β_i son positivos, α_i no puede ser mayor a C:

$$\alpha_1^*, \cdots, \alpha_N^* = \underset{\alpha_1,\cdots,\alpha_N}{\operatorname{argmax}} -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\,\alpha_j y_i y_j\,\langle \mathbf{x}_i, \mathbf{x}_j\rangle + \sum_{i=1}^{N}\alpha_i$$

Sujeto a las restricciones: $0 \leq \alpha_i \leq C\ ,\ 1 \leq i \leq N$ y $\sum_{i=1}^{N}\alpha_i y_i = 0$

El hiperparámetro C

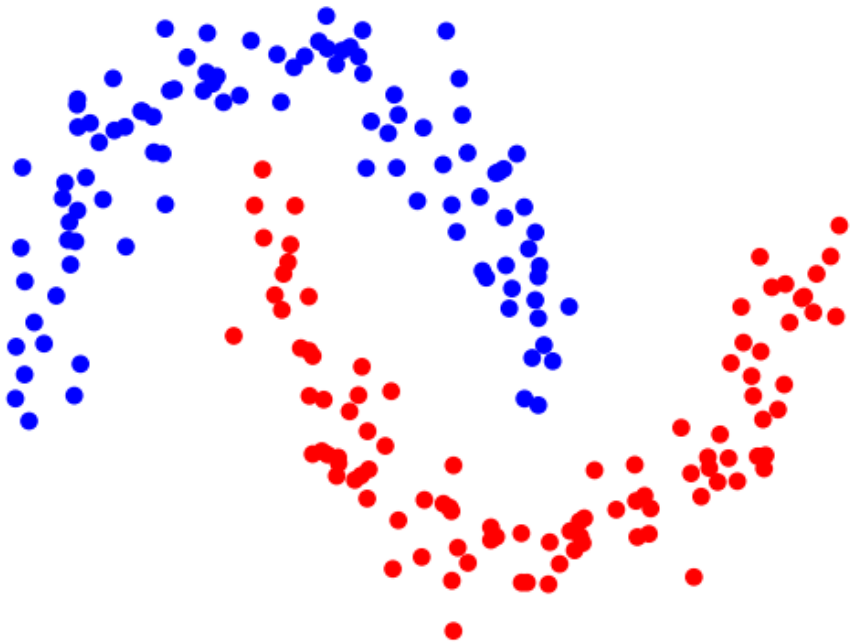
El valor C depende del problema y es necesario experimentar con varios valores.

- **Valor alto de** C: El modelo se enfoca en clasificar correctamente todos los ejemplos del conjunto de entrenamiento, incluso si eso significa tener un margen más pequeño. El modelo penaliza más fuertemente los errores de clasificación, lo que puede llevar a un margen más estrecho entre las clases. Esto puede resultar en un modelo que se ajuste demasiado a los datos de entrenamiento (overfitting).
- **Valor bajo de** C: El modelo permite más errores de clasificación en el conjunto de entrenamiento para lograr un margen de separación más amplio. Esto implica una regularización más fuerte, lo que puede resultar en un modelo más generalizado y menos propenso a overfitting.

Table of Contents

El truco del Kernel

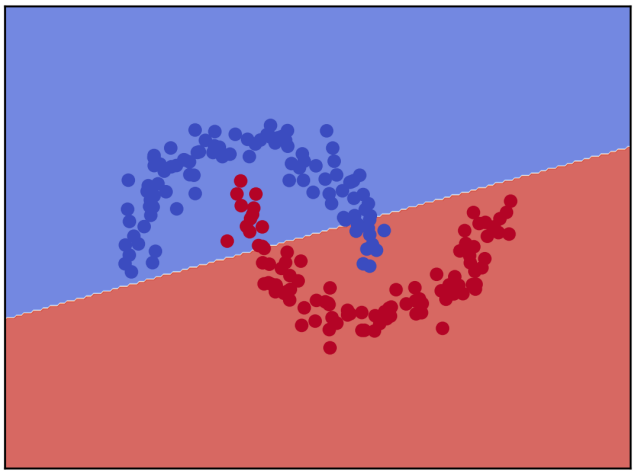
El truco del kernel consiste en sustituir el producto interior usual en la formulación del algoritmo SVM por un nueva función, llamada kernel. Esto nos permite encontrar fronteras de decisión que no necesariamente son hiperplanos.



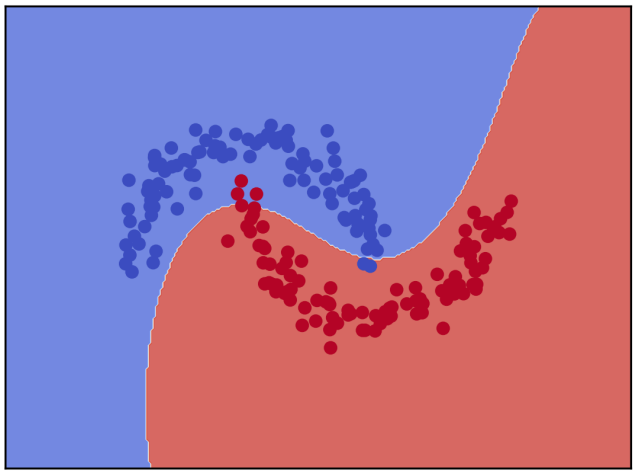
El truco del Kernel

El truco del kernel consiste en sustituir el producto interior usual en la formulación del algoritmo SVM por un nueva función, llamada kernel. Esto nos permite encontrar fronteras de decisión que no necesariamente son hiperplanos.

Kernel Lineal

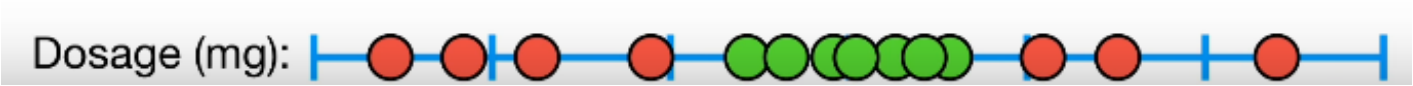


Kernel Polinomial

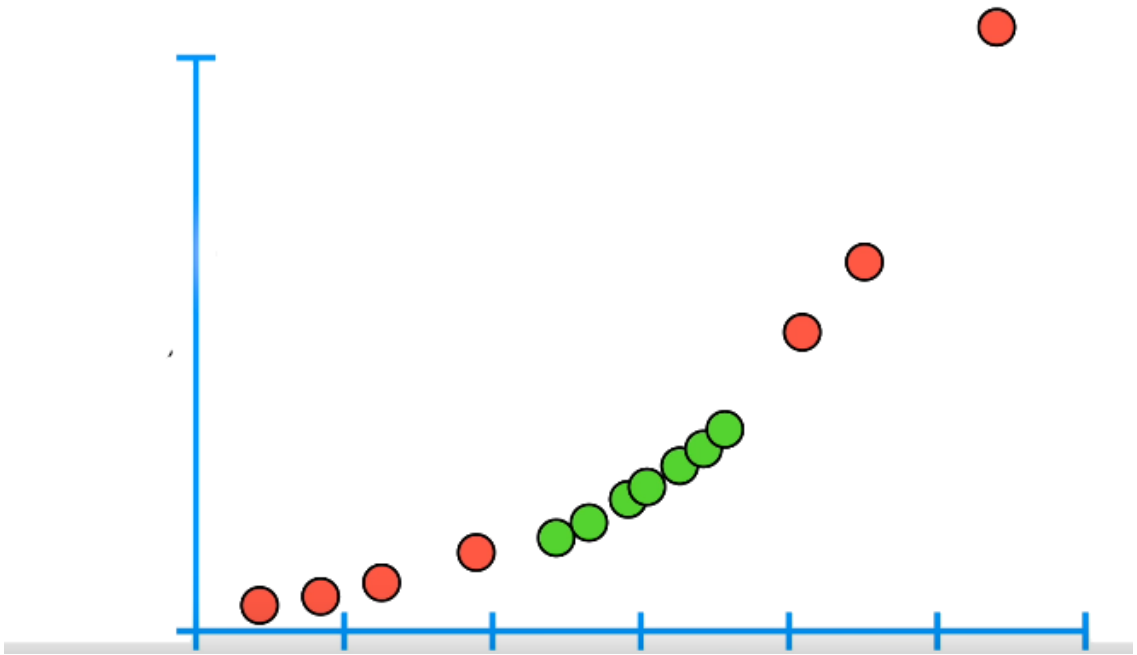


El truco del Kernel

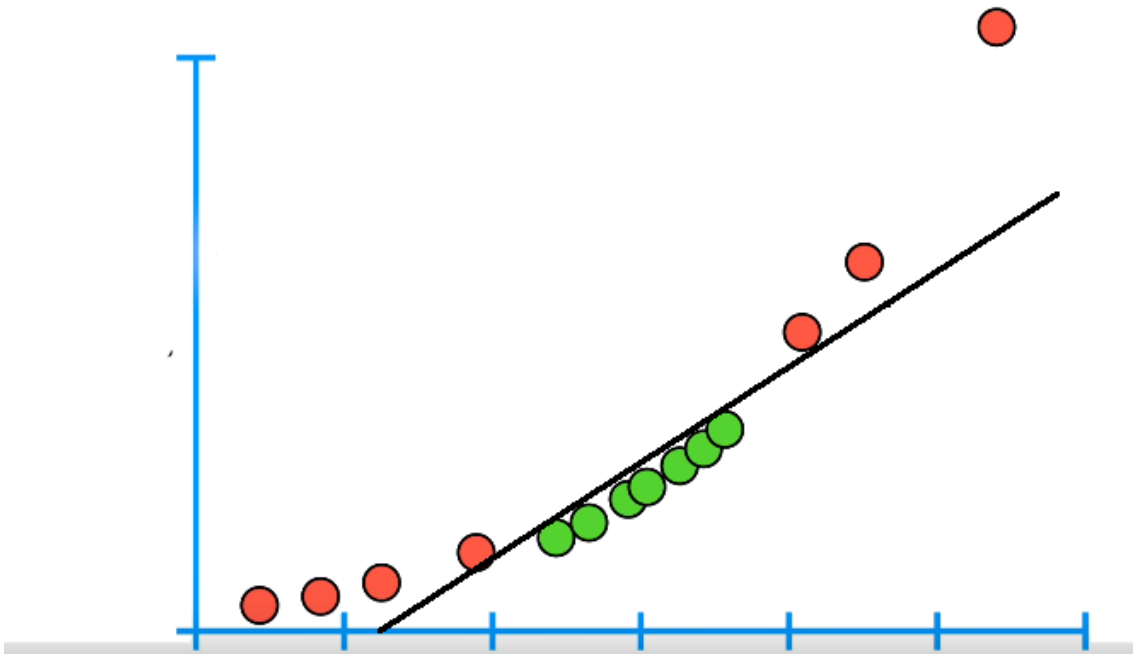
Supongamos que queremos determinar qué dosis de un medicamento es la adecuada para tratar alguna condición. Coloreamos en verde las dosis que lograron una mejoría y en rojo las que no. ¿Cómo logramos una separación lineal de los datos?



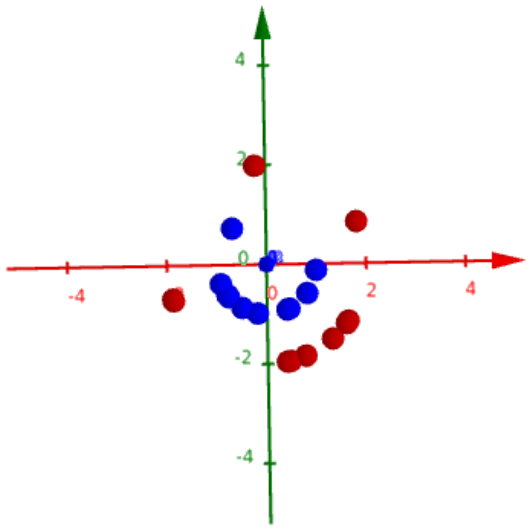
El truco del Kernel



El truco del Kernel



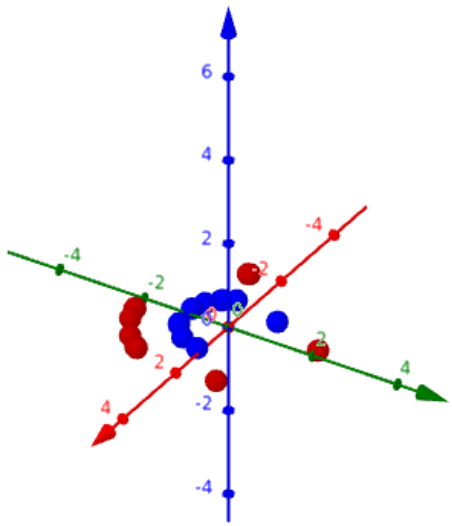
Un ejemplo con “kernel polinomial”



Queremos separar con *linealmente* estos datos.

https://www.geogebra.org/m/xawkavxe

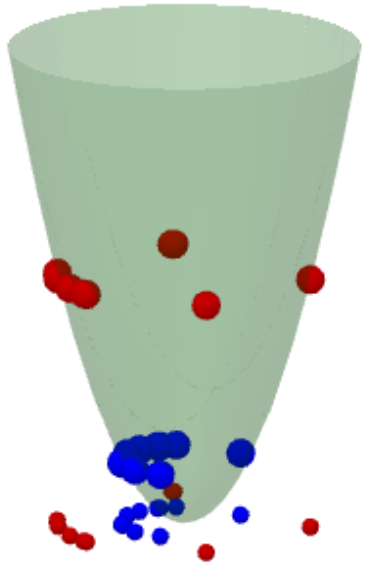
Un ejemplo con “kernel polinomial”



Consideramos los datos en un espacio de dimensión superior.

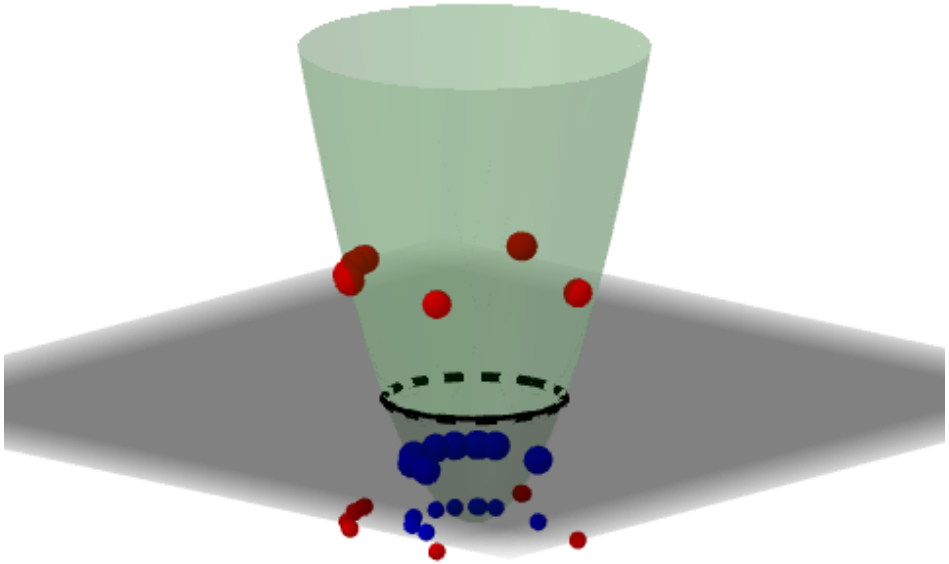
https://www.geogebra.org/m/xawkavxe

Un ejemplo con “kernel polinomial”



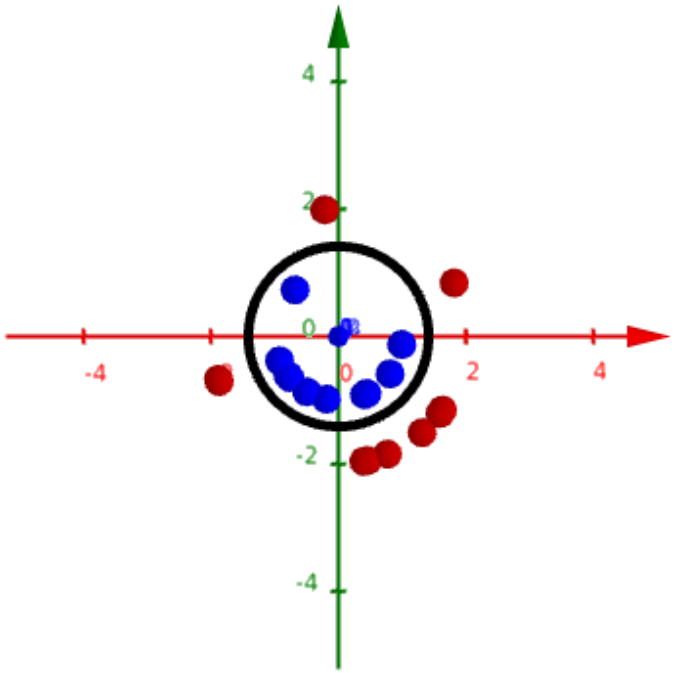
https://www.geogebra.org/m/xawkavxe

Un ejemplo con “kernel polinomial”



https://www.geogebra.org/m/xawkavxe

Un ejemplo con “kernel polinomial”



https://www.geogebra.org/m/xawkavxe

El ejemplo más sencillo: kernel polinomial de grado 2

Para puntos en $\mathbb{R}^2$, reemplazamos las 2 features originales por 4 features nuevas por medio de la función $\phi : \mathbb{R}^2 \to \mathbb{R}^4$ dada por

$$\phi(x_1, x_2) = (x_1^2, x_1 x_2, x_1 x_2, x_2^2)$$

El ejemplo más sencillo: kernel polinomial de grado 2

Para puntos en $\mathbb{R}^2$, reemplazamos las 2 features originales por 4 features nuevas por medio de la función $\phi : \mathbb{R}^2 \to \mathbb{R}^4$ dada por

$$\phi(x_1, x_2) = (x_1^2, x_1x_2, x_1x_2, x_2^2)$$

El producto interior en estas 4 features es

$$\begin{aligned}
\langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle &= \langle (x_1^2, x_1x_2, x_1x_2, x_2^2), (y_1^2, y_1y_2, y_1y_2, y_2^2) \rangle \\
&= x_1^2y_1^2 + x_1x_2y_1y_2 + x_1x_2y_1y_2 + x_2^2y_2^2 \\
&= x_1y_1x_1y_1 + x_1y_1x_2y_2 + x_1y_1x_2y_2 + x_2y_2x_2y_2 \\
&= (x_1y_1 + x_2y_2)^2 = \langle (x_1, x_2), (y_1y_2) \rangle^2 = \langle \mathbf{x}, \mathbf{y} \rangle^2
\end{aligned}$$

El ejemplo más sencillo: kernel polinomial de grado 2

Para puntos en $\mathbb{R}^2$, reemplazamos las 2 features originales por 4 features nuevas por medio de la función $\phi : \mathbb{R}^2 \to \mathbb{R}^4$ dada por

$$\phi(x_1, x_2) = (x_1^2, x_1x_2, x_1x_2, x_2^2)$$

El producto interior en estas 4 features es

$$\begin{aligned}\langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle &= \langle (x_1^2, x_1x_2, x_1x_2, x_2^2), (y_1^2, y_1y_2, y_1y_2, y_2^2) \rangle \\ &= x_1^2y_1^2 + x_1x_2y_1y_2 + x_1x_2y_1y_2 + x_2^2y_2^2 \\ &= x_1y_1x_1y_1 + x_1y_1x_2y_2 + x_1y_1x_2y_2 + x_2y_2x_2y_2 \\ &= (x_1y_1 + x_2y_2)^2 = \langle (x_1, x_2), (y_1y_2) \rangle^2 = \langle \mathbf{x}, \mathbf{y} \rangle^2\end{aligned}$$

Nos podemos ahorrar la definición de la función ϕ si definimos el kernel

$$\kappa(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle^2$$

El siguiente ejemplo más sencillo

Para puntos en $\mathbb{R}^2$, reemplazamos las 2 features originales por 7 features nuevas por medio de la función $\phi : \mathbb{R}^2 \to \mathbb{R}^7$ dada por

$$\phi(x_1, x_2) = (x_1^2, x_1x_2, x_1x_2, x_2^2, \sqrt{2r}x_1, \sqrt{2r}x_2, r)$$

El siguiente ejemplo más sencillo

Para puntos en $\mathbb{R}^2$, reemplazamos las 2 features originales por 7 features nuevas por medio de la función $\phi : \mathbb{R}^2 \to \mathbb{R}^7$ dada por

$$\phi(x_1, x_2) = (x_1^2, x_1x_2, x_1x_2, x_2^2, \sqrt{2r}x_1, \sqrt{2r}x_2, r)$$

El producto interior en estas 7 features es

$$\begin{aligned}\langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle &= \ldots \\ &= (x_1y_1 + x_2y_2 + r)^2 \\ &= (\langle (x_1, x_2), (y_1y_2) \rangle + r)^2 = (\langle \mathbf{x}, \mathbf{y} \rangle + r)^2\end{aligned}$$

El siguiente ejemplo más sencillo

Para puntos en $\mathbb{R}^2$, reemplazamos las 2 features originales por 7 features nuevas por medio de la función $\phi : \mathbb{R}^2 \to \mathbb{R}^7$ dada por

$$\phi(x_1, x_2) = (x_1^2, x_1x_2, x_1x_2, x_2^2, \sqrt{2r}x_1, \sqrt{2r}x_2, r)$$

El producto interior en estas 7 features es

$$\begin{aligned}\langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle &= \ldots \\ &= (x_1y_1 + x_2y_2 + r)^2 \\ &= (\langle (x_1, x_2), (y_1y_2) \rangle + r)^2 = (\langle \mathbf{x}, \mathbf{y} \rangle + r)^2\end{aligned}$$

Nos podemos ahorrar la definición de la función ϕ si definimos el kernel

$$\kappa(\mathbf{x}, \mathbf{y}) = (\langle \mathbf{x}, \mathbf{y} \rangle + r)^2$$

Tipos de Kernel

Los cuatro principales kernels son:

Lineal	$\kappa(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle$
Polinomial	$\kappa(\mathbf{x}, \mathbf{y}) = (\langle \mathbf{x}, \mathbf{y} \rangle + r)^p,\ r \geq 0$
Gaussiano (Radial Basis Function)	$\kappa(\mathbf{x}, \mathbf{y}) = \mathrm{e}^{-\gamma \|\mathbf{x}-\mathbf{y}\|^2}$
Sigmoide	$\kappa(\mathbf{x}, \mathbf{y}) = \tanh\left(\langle \mathbf{x}, \mathbf{y} \rangle + r\right)$

Algunos criterios para escoger el kernel

- Primero veamos si los datos son linealmente separables.
- El kernel lineal suele ser bueno cuando hay muchas features. Además, es el más rápido.
- El kernel polinomial es una generalización del kernel lineal.
- El kernel RBF es un buen kernel por defecto. Suele usarse cuando no hay información adicional sobre los datos.
- El kernel sigmoide equivale a usar una red neuronal de tipo *perceptron de dos capas*.
- Se pueden probar los demás kernels usando *grid seach* y *validación cruzada*.

Algunos criterios para escoger el kernel

- Primero veamos si los datos son linealmente separables.
- El kernel lineal suele ser bueno cuando hay muchas features. Además, es el más rápido.
- El kernel polinomial es una generalización del kernel lineal.
- El kernel RBF es un buen kernel por defecto. Suele usarse cuando no hay información adicional sobre los datos.
- El kernel sigmoide equivale a usar una red neuronal de tipo *perceptron de dos capas*.
- Se pueden probar los demás kernels usando *grid seach* y *validación cruzada*.

Cawley, G. C., & Talbot, N. L. (2010). On over-fitting in model selection and subsequent selection bias in performance evaluation. The Journal of Machine Learning Research, 11, 2079-2107.