Máquinas de Vectores de Soporte

Dra. Graciela Meza Lovón
gmezal@ucsp.edu.pe

Ciencia de la Computación
Universidad Católica San Pablo

Basadas en Lecture Notes de Andrew Ng[4]
Tutorial de SVM de Carmona [6].

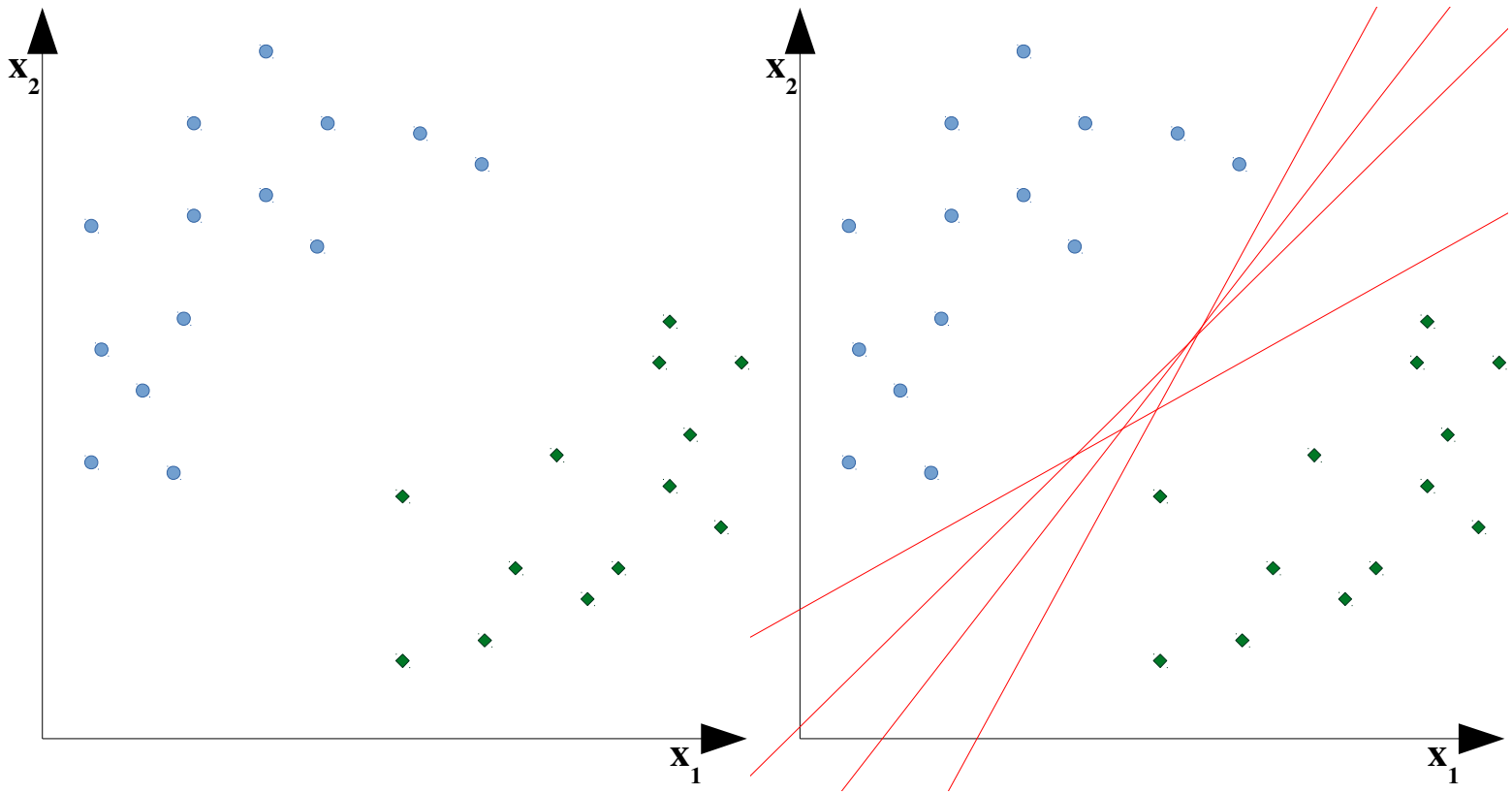
Diciembre, 2018

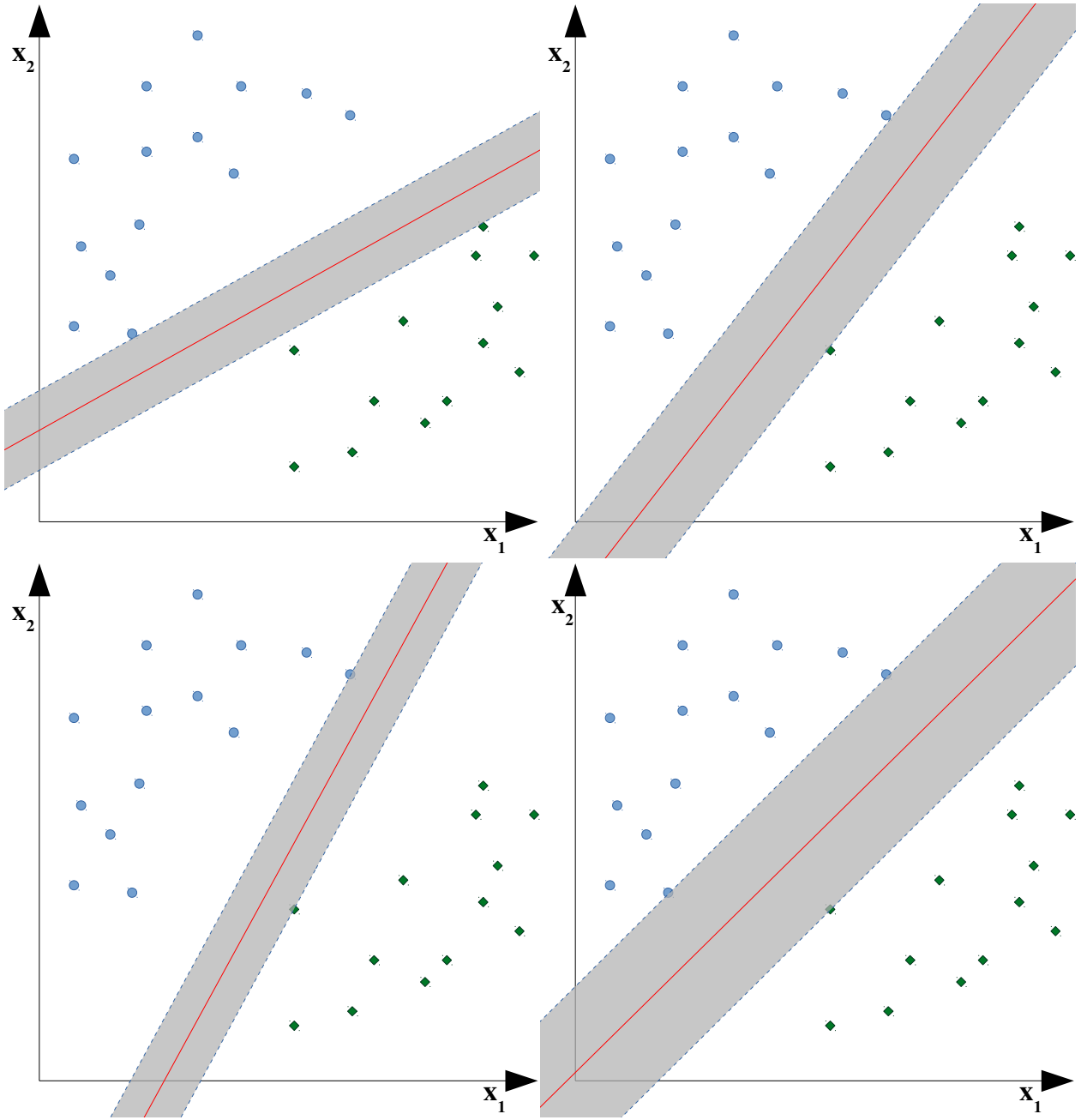
Introducción

- Las bases del algoritmo fueron dadas por Vladimir Vapnik y Alexey Chervonenkis [7] en 1964.
- En 1992, Bernhard E. Boser, Isabelle M. Guyon y Vladimir N. Vapnik [2] sugirieron una manera de crear clasificadores no lineales aplicando el truco del kernel a los hiperplanos de margen máximo.
- En 1995, el algoritmo fue propuesto por Corina Cortes y Vapnik [3].

Contenido

Introducción




x2
x1
x2
x1
x2
x1
x2
x1

- Dado un conjunto separable de ejemplos $S = \{(\boldsymbol{x}^{(1)}, y^{(1)}), \ldots, (\boldsymbol{x}^{(m)}, y^{(m)})\}$, donde $\boldsymbol{x}^{(i)} \in \mathbb{R}^d$ e $y^{(i)} \in \{+1, -1\}$, se define un hiperplano de separación (una función lineal) capaz de separar el conjunto sin errores:

$$D(\boldsymbol{x}) = (w_1 x_1) + \ldots + (w_d x_d) + b = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b$$

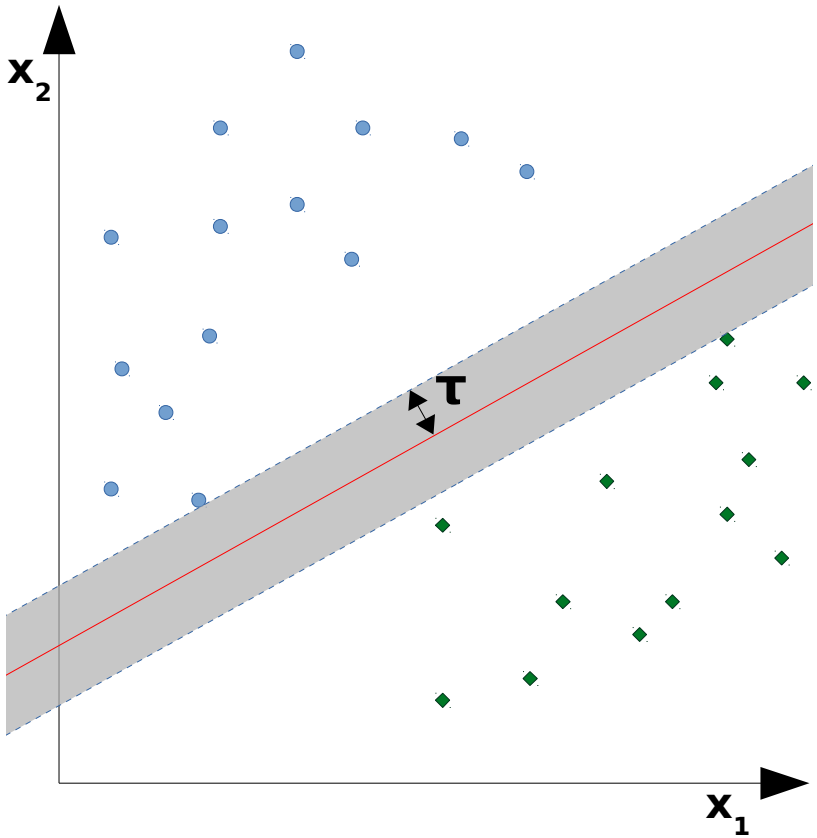
  donde $\boldsymbol{w}$ y b son coeficientes reales.
- Para todo $\boldsymbol{x}^{(i)}$ del conjunto, el hiperplano de separación cumplirá estas restricciones:
  - $\langle \boldsymbol{w}, \boldsymbol{x}^{(i)} \rangle + b \geq 0$ si $y^{(i)} = +1$,
  - $\langle \boldsymbol{w}, \boldsymbol{x}^{(i)} \rangle + b \leq 0$ si $y^{(i)} = -1$.

Margen de un Hiperplano de Separación

- De forma más compacta

$$y^{(i)}(\langle \boldsymbol{w}, \boldsymbol{x}^{(i)} \rangle + b) \geq 0, i = 1, \ldots, m$$

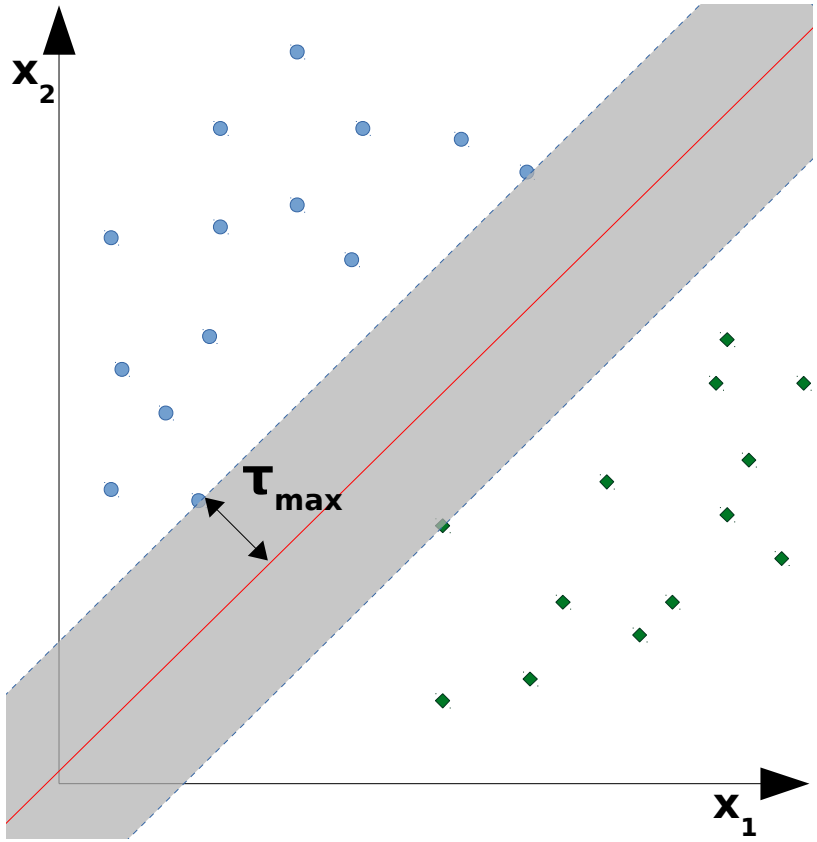
- Se define el concepto de **margen de un hiperplano de separación**, τ, como la mínima distancia entre dicho hiperplano y el ejemplo más cercano de cualquiera de las dos clases.



Margen de un Hiperplano Óptimo de Separación

- Un hiperplano de separación será óptimo si su margen es de tamaño máximo.
- La distancia entre un hiperplano de separación $D(\boldsymbol{x})$ a un ejemplo $\boldsymbol{x'}$ es:

$$\frac{|D(\boldsymbol{x'})|}{||\boldsymbol{w}||}$$



- Todos los ejemplos de entrenamiento cumplirán que:

$$\frac{y^{(i)} D(\boldsymbol{x}^{(i)})}{||\boldsymbol{w}||} \geq \tau_{max}, i = 1, \ldots, m$$

Formulación del Problema

- Encontrar el hiperplano óptimo es equivalente a encontrar el valor de $\boldsymbol{w}$ que maximiza el margen.
- Ya que existen un número infinito de hiperplanos óptimos, es necesario limitar los hiperplanos separables escalando el producto de τ_{max} y la norma de $\boldsymbol{w}$ a 1, i.e.,

$$\tau_{max}||\boldsymbol{w}|| = 1.$$

- Aumentar el margen equivale a disminuir la norma de $\boldsymbol{w}$, i.e.,

$$\tau_{max} = \frac{1}{||\boldsymbol{w}||}$$

- Un hiperplano de separación para el cual se obtenga un valor mínimo de $||\boldsymbol{w}||$ y que esté restringido a $\frac{y^{(i)}D(\boldsymbol{x}^{(i)})}{||\boldsymbol{w}||} \geq \tau_{max}$ y a $\tau_{max}||\boldsymbol{w}|| = 1$ será óptimo, i.e., un hiperplano de separación óptimo cumple que:

$$y^{(i)}(\langle \boldsymbol{w}, \boldsymbol{x}^{(i)} \rangle + b) \geq 1, i = 1, \ldots, m$$

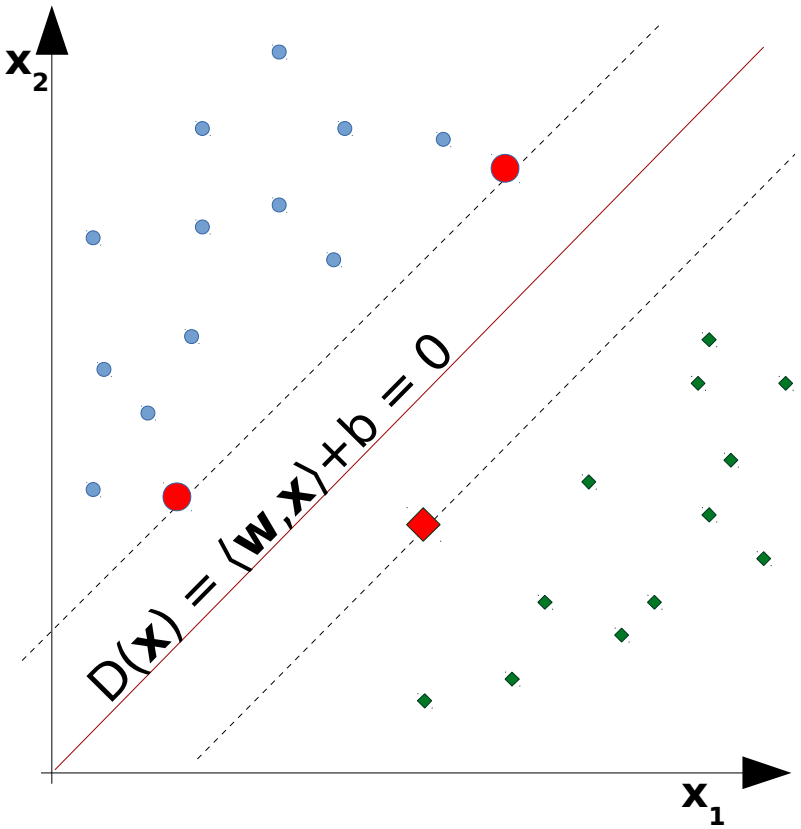
- El problema de encontrar el margen óptimo es un problema de optimización cuadrática. A este problema se le llama "Problema Primal" y puede ser resuelto usando técnicas de teoría de optimización.

$$\begin{aligned} \min_{\boldsymbol{w},b} \quad & \frac{1}{2}||\boldsymbol{w}||^2 \\ \text{s.t.} \quad & y^{(i)}(\langle \boldsymbol{w}\boldsymbol{x}^{(i)} \rangle + b) \geq 1, i = 1, \ldots, m \end{aligned}$$

- Los ejemplos $\boldsymbol{x}^{(i)}$ que dan soporte al hiperplano óptimo son los mismos que definen el margen óptimo, i.e., aquellos para los que se cumple la igualdad

$$y^{(i)}(\langle \boldsymbol{w}, \boldsymbol{x}^{(i)} \rangle + b) = 1$$

son los vectores de soporte.



Problema Dual

- Para un problema de optimización primal existe su "Problema Dual", cuya solución, bajo ciertas condiciones, (función objetivo y restricciones son estrictamente convexas), es solución del problema primal.
- El problema primal minimiza $\boldsymbol{w}, b$, pero el problema dual maximiza $\boldsymbol{\alpha}$, donde $\boldsymbol{\alpha}$ son conocidos como los multiplicadores de Lagrange obtenidos en el proceso de formulación del problema dual.
- La ventaja de resolver el problema dual en vez que el primal se debe a que la complejidad del problema primal escala con la dimensionalidad de los datos, mientras que el problema dual lo hace con el número de ejemplos. Esto quiere decir que inclusive en contextos cuyos datos tienen una alta dimensión, el costo computacional es factible.

Problema Dual

- El problema de optimización dual es:

$$\max_{\boldsymbol{\alpha}} \quad W(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle \boldsymbol{x}^{(i)}, \boldsymbol{x}^{j)} \rangle.$$

$$\text{s.t.} \qquad \alpha_i \geq 0, i = 1, \ldots, m$$

$$\sum_{i=1}^{m} \alpha_i y^{(i)} = 0$$

- Al solucionar el problema dual se encuentra $\boldsymbol{\alpha}^*$. Luego, se soluciona el problema primal que consiste en encontrar encontrar $\boldsymbol{w}^*$ y por ende $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b$.
- Expresando $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b$ en términos de $\boldsymbol{\alpha}$ tenemos:

$$\boldsymbol{w}^T \boldsymbol{x} + b = \sum_{i=1}^{m} \alpha_i y^{(i)} \langle \boldsymbol{x}^{(i)}, \boldsymbol{x} \rangle + b.$$

- Expresando b en términos de $\boldsymbol{\alpha}$ tenemos:

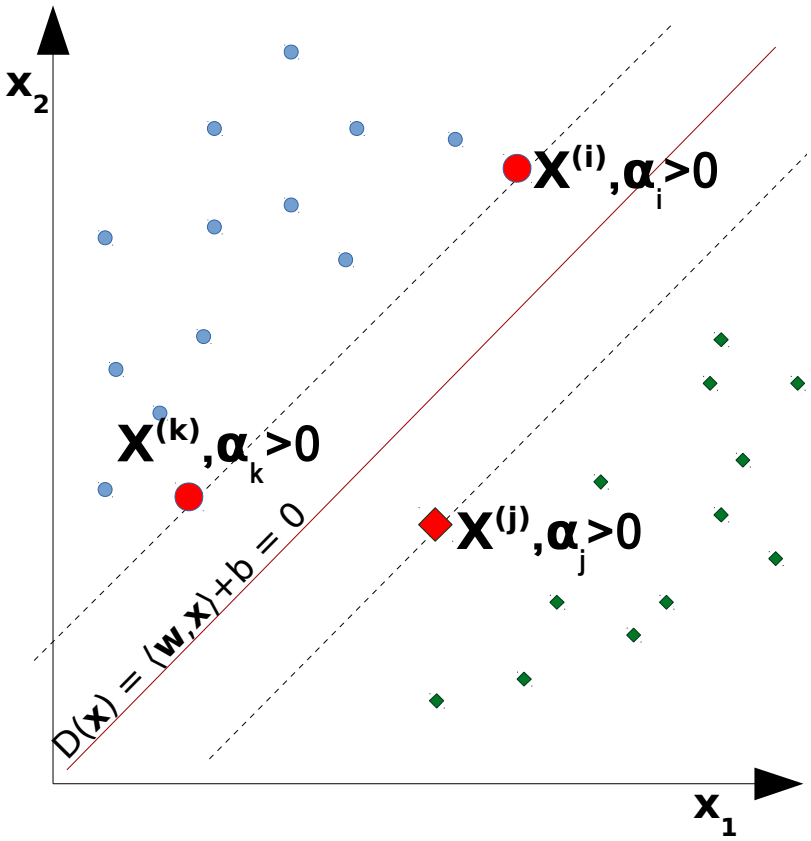
$$b = y^{(s)} - \sum_{j=1}^{m} \alpha_j y^{(j)} \langle \boldsymbol{x}^{(j)}, \boldsymbol{x}^{(s)} \rangle,$$

donde $\boldsymbol{x}^{(s)}$ es cualquier vector de soporte.

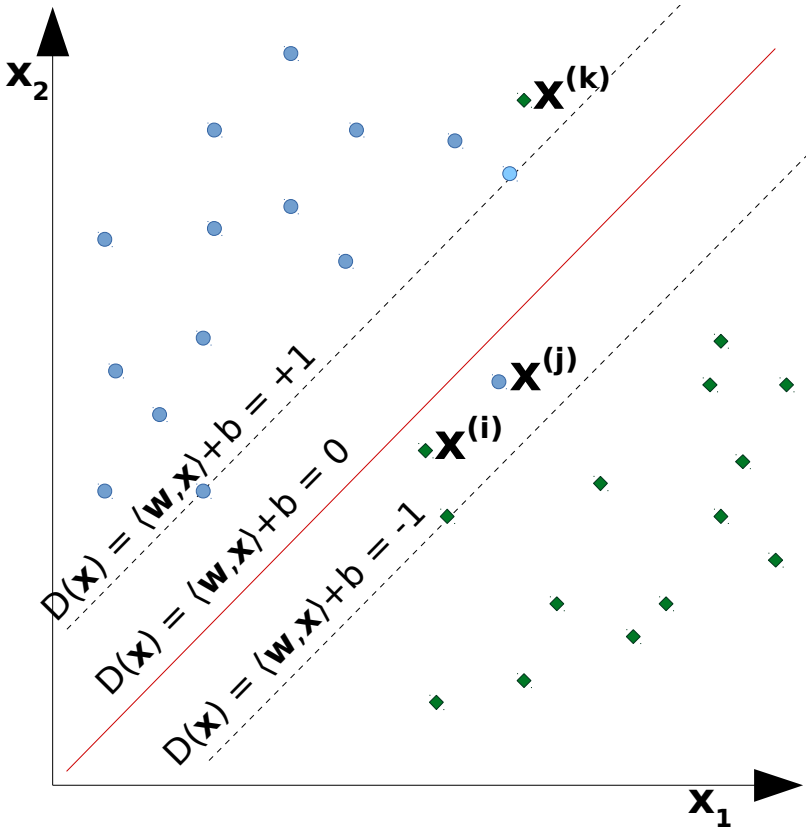
- Del proceso de formulación del problema dual, se obtiene que para un ejemplo $(\boldsymbol{x}^{(i)}, y^{(i)})$, cuyo $\alpha_i > 0$ se tiene que

$$y^{(i)}(\langle \boldsymbol{w}^*, \boldsymbol{x}^{(i)} \rangle + b^*) = 1$$

- A dichos ejemplos se les llama vectores de soporte.



Caso: Ejemplos Casi Separables Linealmente

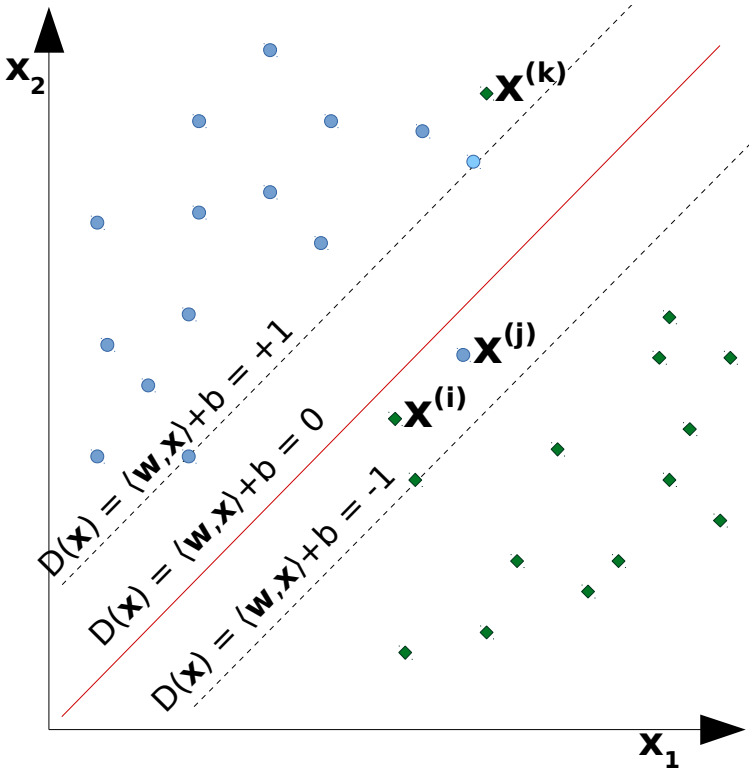


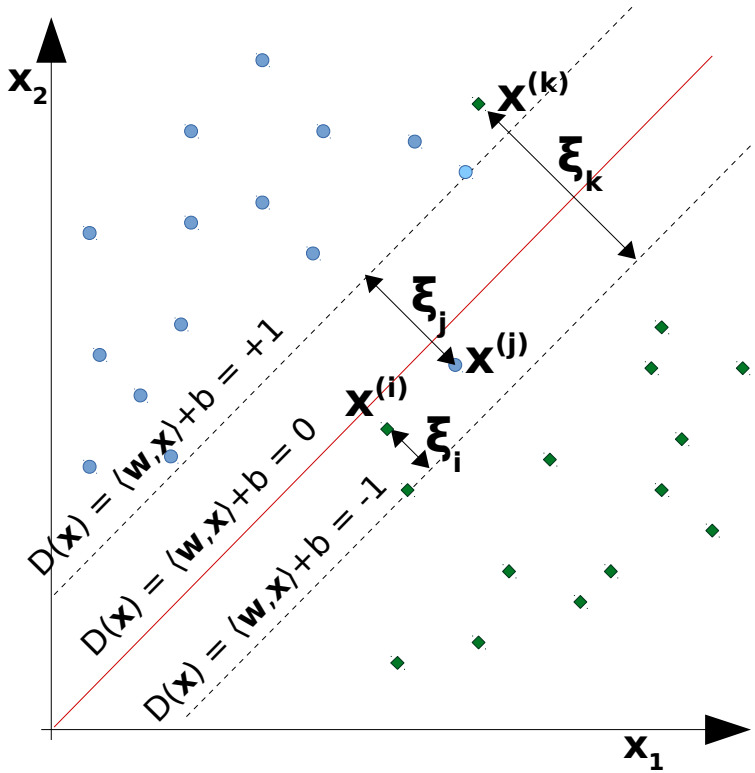
- Los problemas reales se caracterizan normalmente por poseer ejemplos ruidosos, i.e., el caso de ejemplos linealmente separables es un ideal que muy difícilmente se cumplirá.
- Un ejemplo $(y^{(i)}, \boldsymbol{x}^{(i)})$ es no separable si no cumple:

$$y^{(i)}(\langle \boldsymbol{w}, \boldsymbol{x}^{(i)} \rangle + b) \geq 1$$

- Dos casos de ejemplos no separables:
  - El ejemplo cae dentro del margen asociado a la clase correcta, e.g., $\boldsymbol{x}^{(i)}$.
  - El ejemplo cae al otro lado de dicho hiperplano. No es clasificado correctamente. e.g., $\boldsymbol{x}^{(j)}$ y $\boldsymbol{x}^{(k)}$.

Variables de Holgura





- Solución: Relajar el grado de separabilidad del conjunto de ejemplos, permitiendo que haya errores de clasificación. Para ello se usan variables de holgura, $\xi^{(i)}, i = 1, \ldots m$, $\xi^{(i)} \geq 0, \xi \in \mathbb{R}$.

- Para un ejemplo $(\boldsymbol{x}^{(i)}, y^{(i)})$, su variable de holgura, $\xi^{(i)}$, representa la desviación del caso separable, medida desde $\boldsymbol{x}^{(i)}$ hasta borde del margen correspondiente a la clase $y^{(i)}$.
  - Si $\xi^{(i)} = 0$: Ejemplos separables
  - Si $0 < \xi^{(i)} < 1$: Ejemplos no separables bien clasificados
  - Si $\xi^{(i)} \geq 1$: Ejemplos no separables y mal clasificados.
- La suma de todas las variables de holgura, $\sum_{i=1}^{m} \xi^{(i)}$, mide el costo asociado al número de ejemplos no separables.
- Una variable de holgura mayor que cero $(\xi^{(i)} > 0)$ permite que el margen del ejemplo $\boldsymbol{x}^{(i)}$ sea menor que 1.

$$y^{(i)}(\langle \boldsymbol{w}, \boldsymbol{x}^{(i)} \rangle + b) \geq 1 - \xi^{(i)}, \xi^{(i)} \geq 0, \xi \in \mathbb{R}, i = 1, \ldots, m$$

- Sin embargo, es necesario modificar la función objetivo a fin de que el hecho de incrementar el valor de $\xi^{(i)}$ tenga asociado un costo proporcional al valor de $\xi^{(i)}$.
- Para incluir dicho costo, se define una nueva función objetivo a optimizar:

$$f(\boldsymbol{w}, \boldsymbol{\xi}) = \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{m}\xi^{(i)}$$

donde C es una constante que controla el grado en que el costo de ejemplos no separables incluye en la minimización de la norma, i.e., en la maximización del margen.

- El parámetro C controla el compromiso entre el grado de sobreajuste del clasificador y la proporción del número de ejemplos no separables.
  - Si C es muy grande, entonces los valores de $\xi^{(i)}$ serán muy pequeños. Eso quiere decir que el margen también será más pequeño y por ende habrá sobreajuste. En particular si, $(C \to \infty)$, estamos en el caso perfectamente separable $(\xi^{(i)} \to 0)$.
  - Si C es muy pequeño, entonces los valores de $\xi^{(i)}$ serán muy grandes, i.e., el número de ejemplos mal clasificados será elevado. Por ende, el margen será más grande, lo que permite disminuir el sobreajuste. En el caso límite $(C \to 0)$, todos los ejemplos estarán mal clasificados $(\xi^{(i)} \to \infty)$.

Problema Primal

- El problema de optimización incluyendo las variables de holgura es:

$$\begin{aligned} \min_{\boldsymbol{w}} \quad & \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{m}\xi^{(i)} \\ \text{s.t.} \quad & y^{(i)}(\langle \boldsymbol{w}, \boldsymbol{x}^{(i)}\rangle + b) + \xi^{(i)} - 1 \geq 0 \\ & \xi^{(i)} \geq 0, i = 1, \ldots, m \end{aligned}$$

- Al hiperplano obtenido incluyendo variables de holgura se le llama **hiperplano de separación de margen blando**. Al hiperplano obtenido en el caso separable, se le llama **hiperplano de separación de margen duro**.
- Como en el caso separable, el problema de optimización puede ser transformado a su forma dual.

- La formalización del problema dual es:

$$\max_{\boldsymbol{\alpha}} \quad \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle \boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)} \rangle.$$

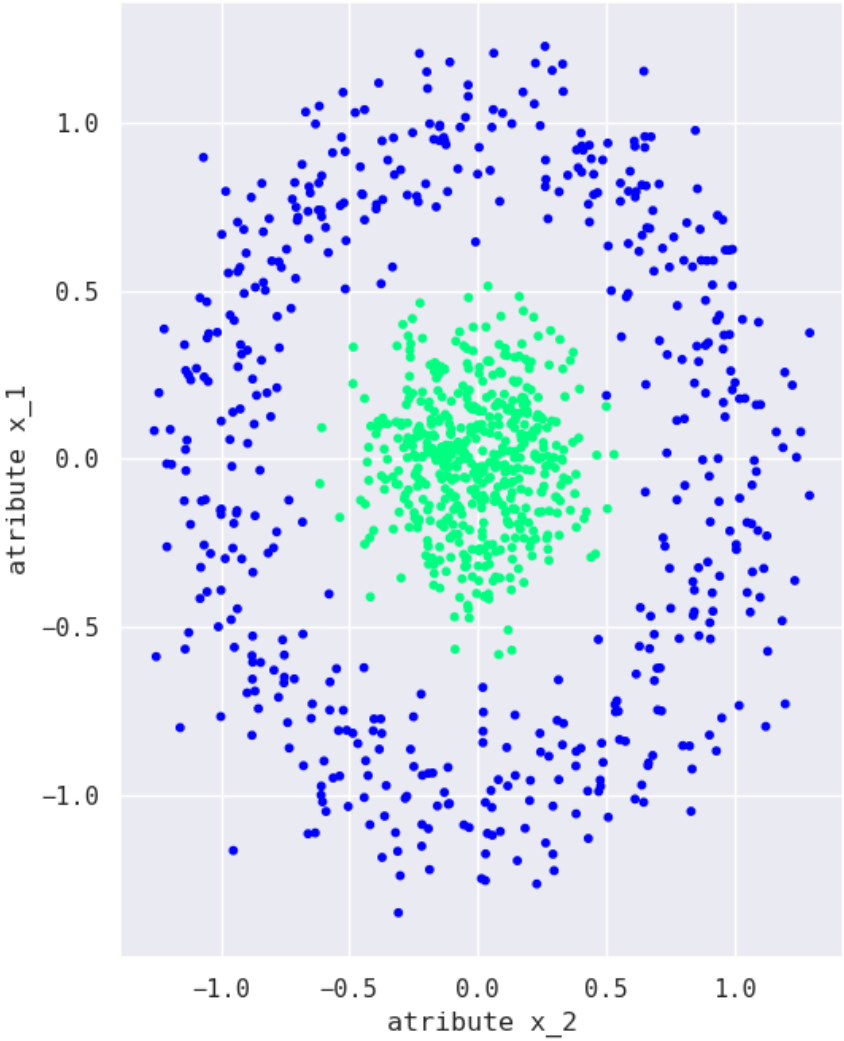
$$\text{s.t.} \quad \sum_{i=1}^{m} \alpha_i y^{(i)} = 0$$

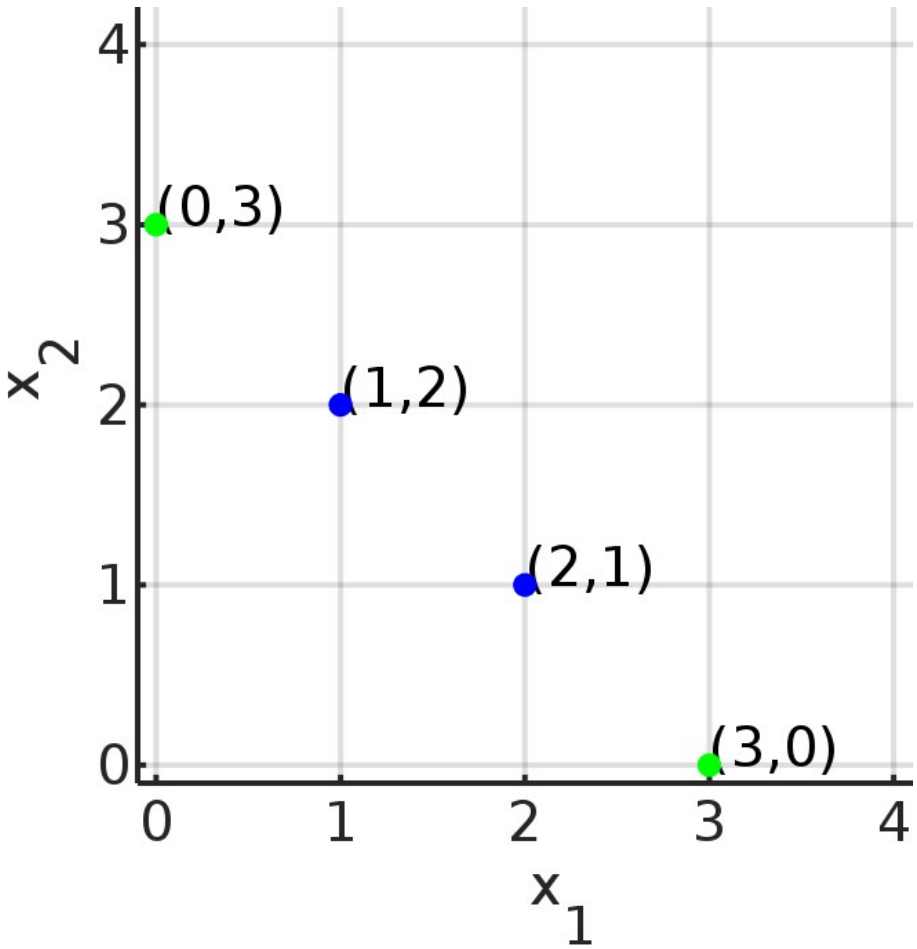
$$0 \leq \alpha_i \leq C, i = 1, \ldots, m$$

- Un ejemplo $\boldsymbol{x}^{(i)}$, es un vector soporte si y solo si $0 < \alpha_i < C$.

Kernels

- El conjunto no puede ser separado por una función lineal.





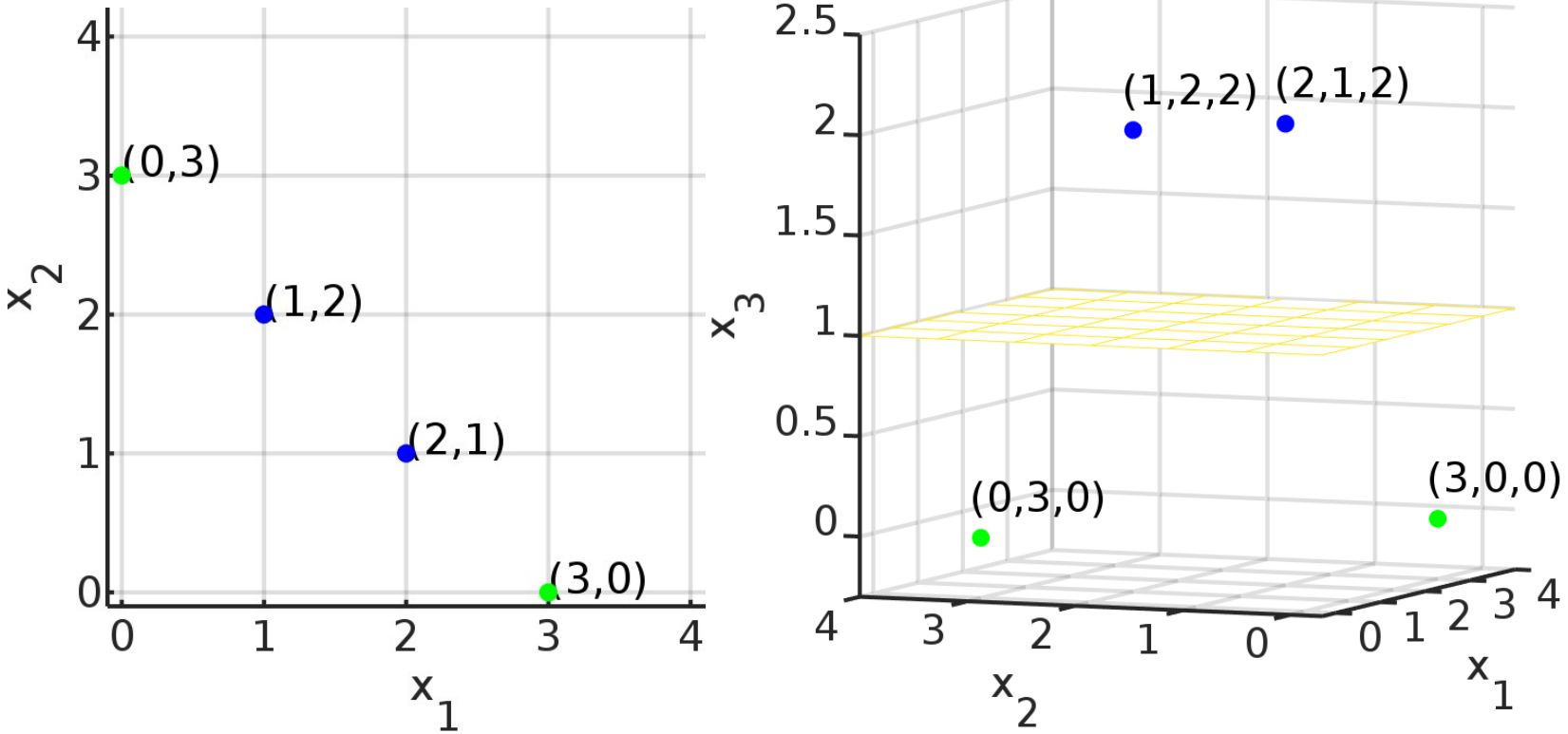
Kernels

- La idea es realizar dicha separación lineal en una dimensión mayor. Para ello se usa función, $\Phi(\boldsymbol{x})$, que mapee de los datos de entrada (del espacio de los ejemplos o espacio - $\mathcal{X}$) a un espacio dimensional superior (espacio de las características o espacio - $\mathcal{F}$).
- La frontera de decisión obtenida en el espacio de las características será lineal pero al mapearla al espacio de los ejemplos será una frontera de decisión no lineal.
- E.g. sea un $\boldsymbol{x} \in \mathbb{R}^n$ donde $n = 2$, se define una función de mapeo Φ que lleva a $\boldsymbol{x}$ de $\mathbb{R}^2$ a $\mathbb{R}^3$ como sigue:

$$\Phi(\boldsymbol{x}) = \begin{bmatrix} \phi_1(\boldsymbol{x}) \\ \phi_2(\boldsymbol{x}) \\ \phi_3(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_1 x_2 \end{bmatrix}$$

Kernels

- El conjunto no puede ser separado por una función lineal.



- Sea Φ una función de mapeo. Definimos una función Kernel, K tal que $K(\boldsymbol{x}, \boldsymbol{z}) = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{z}) \rangle = \Phi(\boldsymbol{x})^T \Phi(\boldsymbol{z})$.
- E.g., sean $\boldsymbol{x} \in \mathbb{R}^3$ y $\boldsymbol{z} \in \mathbb{R}^3$, para calcular $\langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{z}) \rangle$ usando la función de mapeo Φ definida como:

$$\Phi(\boldsymbol{x}) = \begin{bmatrix} x_1x_1 \\ x_1x_2 \\ x_1x_3 \\ x_2x_1 \\ x_2x_2 \\ x_2x_3 \\ x_3x_1 \\ x_3x_2 \\ x_3x_3 \end{bmatrix} \text{ obtenemos que } \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{z}) \rangle = \left\langle \begin{bmatrix} x_1x_1 \\ x_1x_2 \\ x_1x_3 \\ x_2x_1 \\ x_2x_2 \\ x_2x_3 \\ x_3x_1 \\ x_3x_2 \\ x_3x_3 \end{bmatrix}, \begin{bmatrix} z_1z_1 \\ z_1z_2 \\ z_1z_3 \\ z_2z_1 \\ z_2z_2 \\ z_2z_3 \\ z_3z_1 \\ z_3z_2 \\ z_3z_3 \end{bmatrix} \right\rangle =$$

$$\sum_{i=1}^{3}\sum_{j=1}^{3} x_i x_j z_i z_j = \left(\sum_{i=1}^{3} x_i z_i\right)\left(\sum_{j=1}^{3} x_j z_j\right) = (\boldsymbol{x}^T\boldsymbol{z})^2 = K(\boldsymbol{x}, \boldsymbol{z})$$

- De manera general (n dimensiones),

$$\begin{aligned}
\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{z}) \rangle &= \sum_{i,j=1}^{n} (x_i x_j)(z_i z_j) \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} x_i x_j z_i z_j \\
&= \left(\sum_{i=1}^{n} x_i z_i \right) \left(\sum_{j=1}^{n} x_j z_j \right) \\
&= (\boldsymbol{x}^T \boldsymbol{z})^2 \\
&= K(\boldsymbol{x}, \boldsymbol{z})
\end{aligned}$$

- Para calcular $\Phi(\boldsymbol{x})$ se requiere $O(n^2)$ mientras que $K(\boldsymbol{x}, \boldsymbol{z})$ solo requiere $O(n)$.

Kernels Válidos

- Un kernel es válido si se cumple el teorema de Mercer: *Sea* $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$. *Para que* K *sea un kernel válido, es necesario y suficiente que para cualquier* $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$, $(m < \infty)$, *la correspondiente matriz kernel sea simétrica y semidefinida positiva.*

- Otra interpretación: $K(\boldsymbol{x}, \boldsymbol{z})$ puede ser considerado como una medida de similaridad entre $\boldsymbol{x}$ y $\boldsymbol{z}$.
  - Si $\Phi(\boldsymbol{x})$ y $\Phi(\boldsymbol{z})$ son cercanos, $K(\boldsymbol{x}, \boldsymbol{z}) = \Phi(\boldsymbol{x})^T \Phi(\boldsymbol{z})$ debería ser una cantidad grande,
  - Si $\Phi(\boldsymbol{x})$ y $\Phi(\boldsymbol{z})$ están alejados entonces $K(\boldsymbol{x}, \boldsymbol{z}) = \Phi(\boldsymbol{x})^T \Phi(\boldsymbol{z})$ debería ser una cantidad pequeña.

Tipos de Kernels

- Algunas funciones kernel que cumplen con el Teorema de Mercer son:
  - Kernel Lineal:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

  - Kernel Polinómico de grado p:

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \langle \mathbf{x}_i, \mathbf{x}_j \rangle + a)^p$$

  - Kernel Gaussiano:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma ||\mathbf{x}_i - \mathbf{x}_j||^2), \gamma > 0$$

- El parámetro γ del kernel gaussiano es inversamente proporcional a la desviación estándar σ^2 de una función gaussiana, i.e., $\gamma = \frac{1}{(2*\sigma^2)}$
- Mientras la desviación estándar crece, la campana gaussiana se expande, la varianza aumenta y γ disminuye.
- El kernel gaussiano se puede ver como medida de similitud entre dos puntos.
- Si γ es pequeño (gran varianza), entonces dos puntos son similares inclusive si están alejados uno del otro.
- Si γ es grande (pequeña varianza), entonces dos puntos son disimiles inclusive si son cercanos.

Software

- LIBSVM Chih-Chung Chang and Chih-Jen Lin Both C++ and Java sources CHANG, C.C. and C.J. LIN, 2001. LIBSVM: a library for support vector machines. http://www.csie.ntu.edu.tw/cjlin/libsvm.
- SVMlight Thorsten Joachims Written in C JOACHIMS, T., 1999. SVMLight: Support Vector Machine. Vector Machine http://svmlight.joachims.org/
- MATLAB Support Vector Machine Toolbox Gavin Cawley MATLAB toolbox CAWLEY, (2000) http://theoval.sys.uea.ac.uk/gcc/svm/toolbox. MATLAB Support Vector Machine Toolbox.
- R Language Chang and Lin 2001 - Package e1071 provides an interface to libsvm
- Python's Scikit-Learn

Implementación

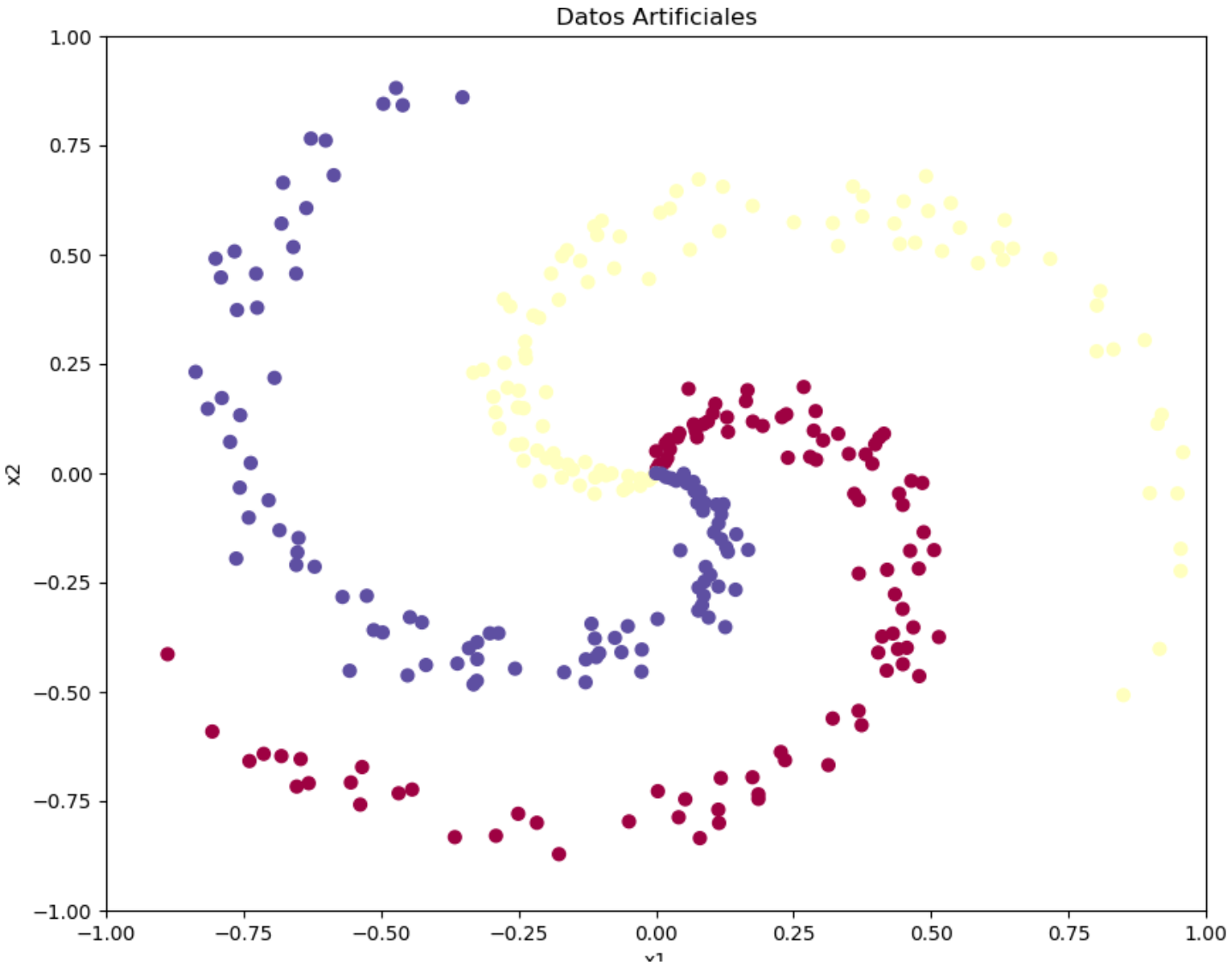
```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import svm

# Parametros de las figura
plt.rcParams['figure.figsize']      = (10.0, 8.0)
np.random.seed(0)
N = 100 # 100 datos por clase
D = 2   # los datos seran de dos dimensiones
K = 3   # tres clases
X = np.zeros((N*K, D))
num_train_examples = X.shape[0]
y = np.zeros(N*K, dtype='uint8')

# Generación de Datos
for j in range(K):
    ix  = range(N * j, N * (j + 1))
    r   = np.linspace(0.0,        1, N)
    t   = np.linspace(j*4, (j+1)*4, N) + np.random.randn(N) * 0.2
    X[ix] = np.c_[r * np.sin(t),  r * np.cos(t)]
    y[ix] = j

plot_data(X, y)
```

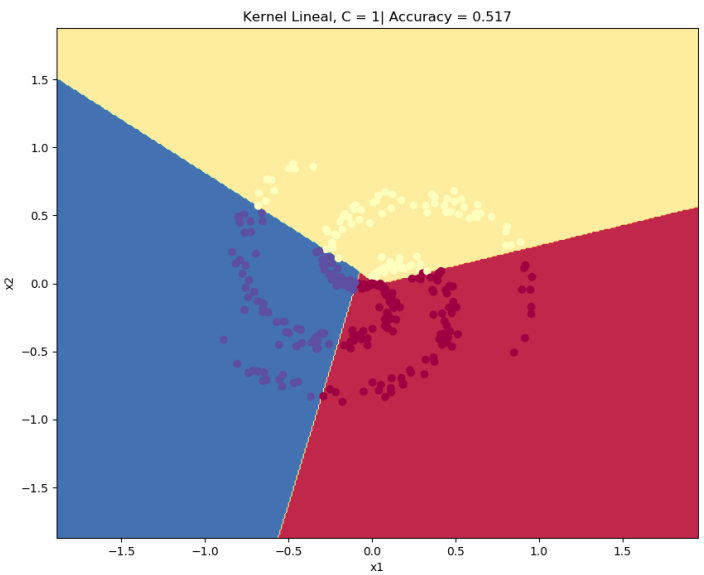
Implementación

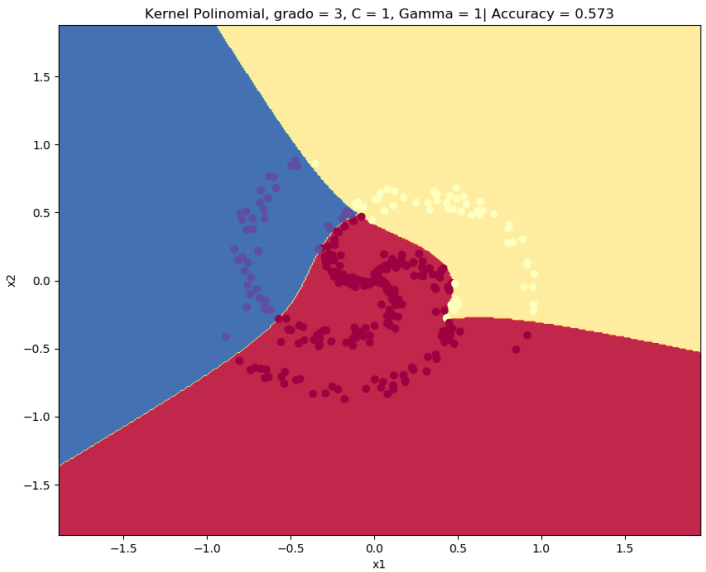


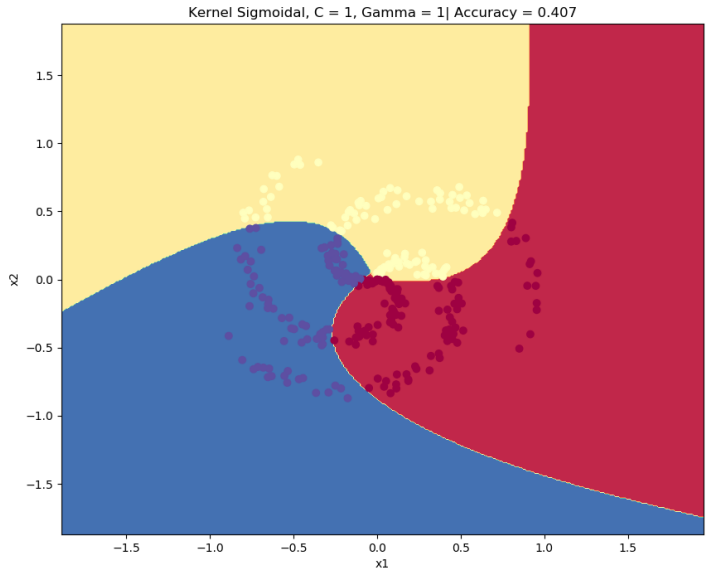
Implementación

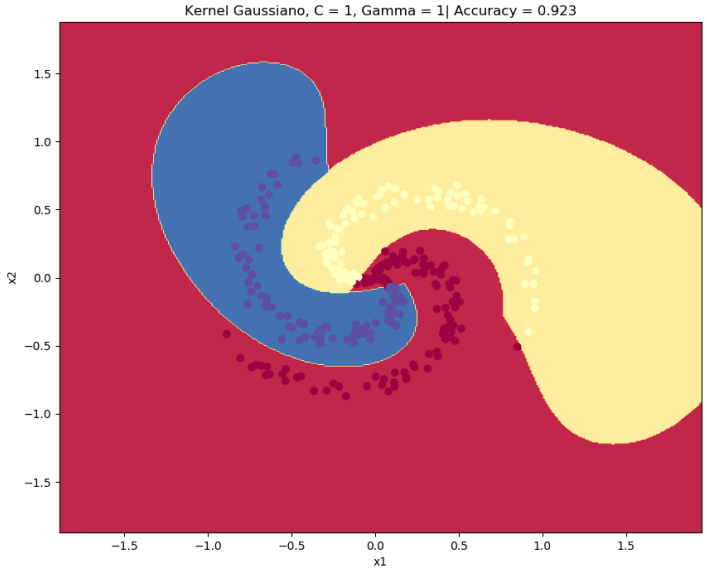
```
C=1
gamma =1
# Kernel Lineal
linear_svc = svm.SVC(kernel='linear', C=C)
linear_svc.fit(X, y)
y_pred     = linear_svc.predict(X)
Accuracy      = linear_svc.score(X, y)
titulo        = "Kernel Lineal, C = "+str(C)+ "| Accuracy = %.3f"%(Accuracy)
plot_decision_boundary( linear_svc, X, y_pred, titulo)
# Kernel Polinomial
poly_svc   = svm.SVC(kernel='poly', C=C, degree=3, gamma=gamma)
poly_svc.fit(X,y)
y_pred     = poly_svc.predict(X)
Accuracy      = poly_svc.score(X, y)
titulo        = "Kernel Polinomial, grado = 3, C = "+str(C)+", Gamma = "
                          +str(gamma)+ "| Accuracy = %.3f"%(Accuracy)
plot_decision_boundary( poly_svc,  X, y_pred, titulo)
# Kernel sigmoidal
sigmoid_svc= svm.SVC(kernel='sigmoid', C=C, gamma=gamma)
sigmoid_svc.fit(X,y)
y_pred     = sigmoid_svc.predict(X)
Accuracy      = sigmoid_svc.score(X, y)
titulo        = "Kernel Sigmoidal, C = "+str(C)+", Gamma = "+str(gamma)+
                                    "| Accuracy = %.3f"%(Accuracy)
plot_decision_boundary( sigmoid_svc, X, y_pred, titulo)
# Kernel Gaussiano
gaussian_svc =  svm.SVC(kernel='rbf', C=C, gamma=gamma)
gaussian_svc.fit(X,y)
y_pred     = gaussian_svc.predict(X)
Accuracy      = gaussian_svc.score(X, y)
titulo        = "Kernel Gussiano, C = "+str(C)+", Gamma = "+str(gamma)+
                                    "| Accuracy = %.3f"%(Accuracy)
plot_decision_boundary( gaussian_svc  , X, y_pred, titulo)
```

Implementación









Referencias

- Asa Ben-Hur et al. "Support Vector Machines and Kernels for Computational Biology.". In: *PLoS Computational Biology* 4.10 (2008). URL: http://dblp.uni-trier.de/db/journals/ploscb/ploscb4.html#Ben-HurOSSR08.
- Bernhard E. Boser, Isabelle M. Guyon and Vladimir N. Vapnik. "A Training Algorithm for Optimal Margin Classifiers". In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. COLT '92. Pittsburgh, Pennsylvania, USA: ACM, 1992, pp. 144–152. ISBN: 0-89791-497-X. DOI: 10.1145/130385.130401. URL: http://doi.acm.org/10.1145/130385.130401.
- Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine Learning* 20.3 (Sept. 1995), pp. 273–297. ISSN: 1573-0565. DOI: 10.1007/BF00994018. URL: https://doi.org/10.1007/BF00994018.
- Andrew Ng. *CS229 Lecture notes, Part V Support Vector Machines*. URL: http://cs229.stanford.edu/notes/cs229-notes3.pdf.
- Bernhard Scholkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2001. ISBN: 0262194759.
- Enrique J. Carmona Suárez. *Tutorial sobre Máquinas de Vectores Soporte (SVM)*. URL: http://www.ia.uned.es/~ejcarmona/publicaciones/[2013-Carmona]%20SVM.pdf.
- V. Vapnik and A. Chervonenkis. "A note on one class of perceptrons". In: *Automation and Remote Control* 25 (1964).