

# **Support Vector Machines (SVMs) - Parte II**

Aprendizaje Automático INF-393 II-2018

---

Ricardo Ñanculef

UTFSM Campus San Joaquín

## Table of contents

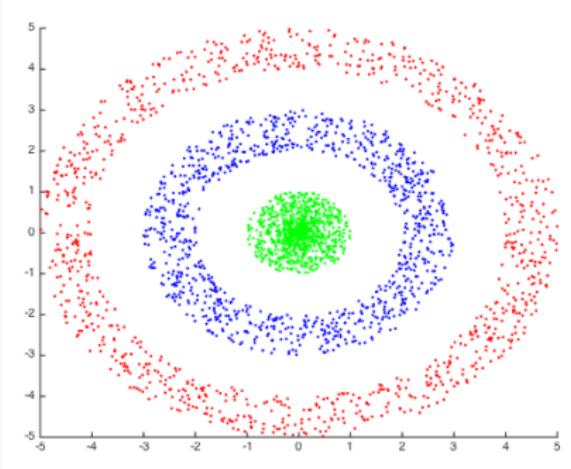
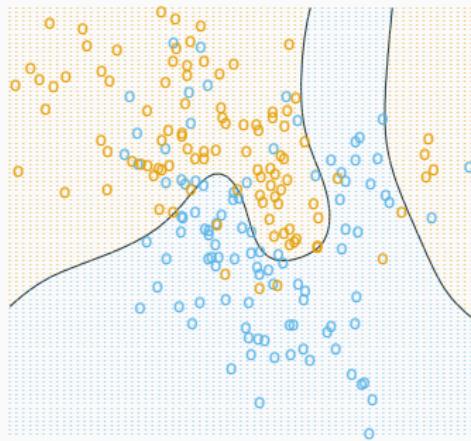
1. Extensión No-lineal de la SVM via Transformaciones
2. Extensión No-lineal de la SVM via Kernels
3. Kernels Admisibles
4. Extensiones para Regresión

## **Extensión No-lineal de la SVM vía Transformaciones**

---

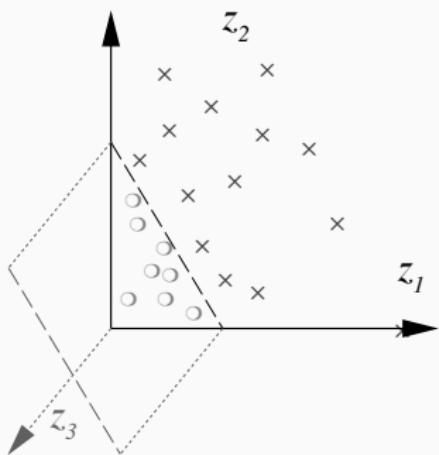
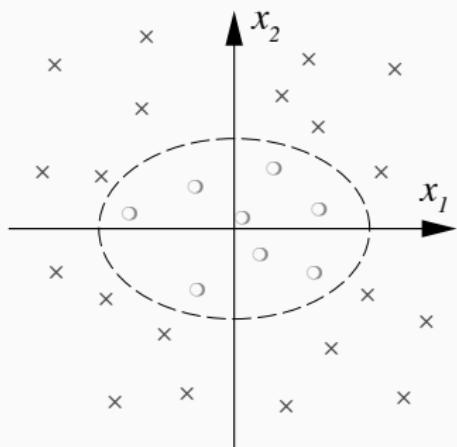
# Problemas Linealmente Inseparables

- La SVM que hemos obtenido en el capítulo anterior es un método potente para lidiar con problemas “casi” linealmente separables.
- El siguiente ingrediente de la “receta” de una SVM para resolver problemas de clasificación, es un “truco” para extender toda la maquinaria al diseño de fronteras no-lineales.



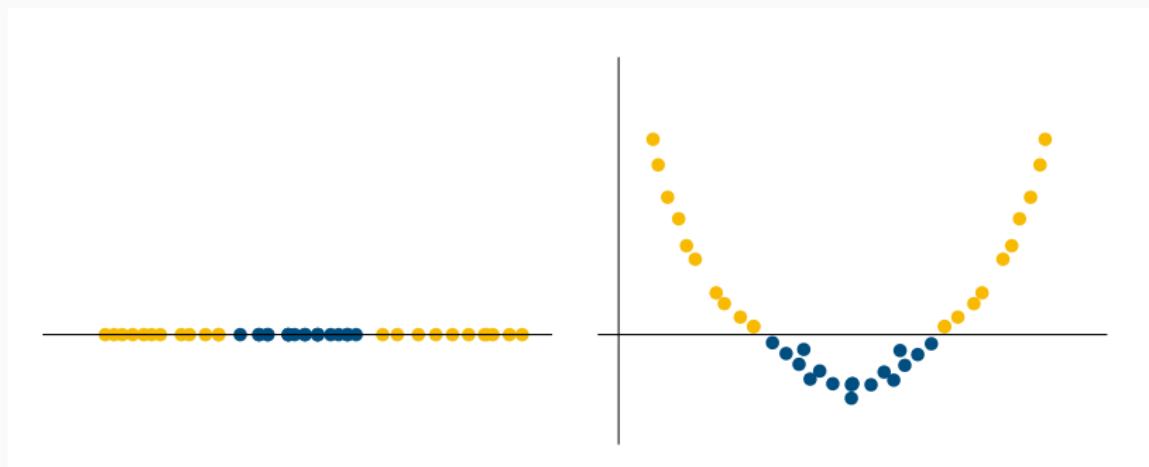
# La Clave es la Representación

- Idea: Un problema que parece linealmente inseparable puede volverse linealmente separable, cambiando la representación de los datos.



# La Clave es la Representación

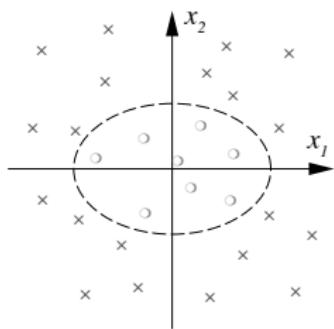
- Un problema que parece difícil en un espacio característico de cierta dimensionalidad puede ser muy fácil de resolver en un espacio de alta dimensionalidad.



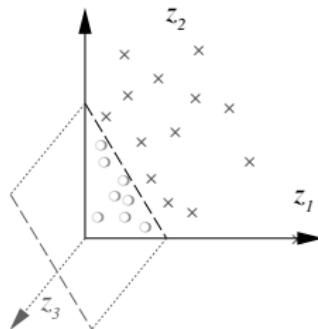
# Cambio de Representación

- Para cambiar la representación, introduciremos un mapa  $\phi : \mathbb{X} \rightarrow \mathcal{Z}$  desde el espacio original  $\mathbb{X}$  donde viven nuestros datos a un nuevo espacio de características  $\phi(\mathbb{X}) \subset \mathcal{Z}$ .

espacio original  $\mathbb{X}$



nuevo espacio  $\phi(\mathbb{X}) \subset \mathcal{Z}$



# SVM Lineal en el Nuevo Espacio

- En vez de construir la SVM en  $\mathbb{X}$ , la construiremos en  $\phi(\mathbb{X})$ .
- Dado  $\phi$  y un conjunto de entrenamiento  $S = \{(x^{(\ell)}, y^{(\ell)})\}_{\ell=1}^n$ , necesitamos calcular  $\phi(S) = \{(\phi(x^{(\ell)}), y^{(\ell)})\}_{\ell=1}^n$  y resolver

$$\begin{aligned} & \min_{w, b, \xi} \frac{1}{2} \|w\|^2 + C \cdot \sum_{\ell} \xi_{\ell} \\ \text{s.t. } & y^{(\ell)}(w^T \phi(x^{(\ell)}) + b) \geq 1 - \xi_{\ell} \quad \forall \ell \\ & \xi_{\ell} \geq 0 \quad \forall \ell \end{aligned}$$

donde  $\phi(x^{(\ell)})$  es la nueva representación del ejemplo  $x^{(\ell)}$ .

- El clasificador obtenido es un hiperplano en  $\phi(\mathbb{X})$ .
- Para clasificar datos, necesitaremos primero mapearlos a nuevo espacio y luego aplicar el clasificador

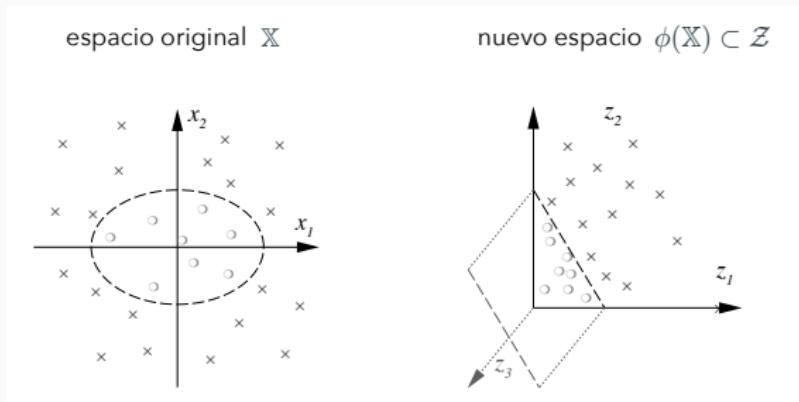
$$f(x) = \text{sign}(w^T \phi(x) + b) \tag{1}$$

# SVM No-Lineal en el Espacio Original

- El clasificador obtenido es un hiperplano en  $\phi(\mathbb{X})$ , pero si el mapa  $\mathbb{X} \rightarrow \phi(\mathbb{X})$  no es lineal, el clasificador “equivalente” en  $\mathbb{X}$  será **no lineal** (función no lineal de los atributos originales  $x$ ).

$$f(\mathbf{z}) = \text{sign} (\mathbf{w}^T \mathbf{z} + b) \quad (2)$$

$$f(\mathbf{x}) = \text{sign} (\mathbf{w}^T \phi(\mathbf{x}) + b)$$



# ¿Qué características debe tener $\phi$ ?

- Para obtener fronteras no lineales en  $\mathbb{X}$ , tendremos que elegir un mapa  $x \rightarrow \phi(x)$  no lineal.
- Para obtener separabilidad lineal (o casi), podemos usar un mapa  $x \rightarrow \phi(x)$  que cree un espacio con muchas dimensiones adicionales.

## Teorema<sup>1</sup>

Sea  $\phi$  una función de  $\mathbb{R}^d$  a  $\mathbb{R}^D$ ,  $M \in \mathbb{Z}$ . Para cualquier conjunto de  $n \in \mathbb{Z}$  ejemplos  $S = \{(x^{(\ell)}, y^{(\ell)})\}_{\ell=1}^n$ ,  $x^{(\ell)} \in \mathbb{R}^d$ ,  $y^{(\ell)} \in \{\pm 1\}$ , tenemos que  $\phi(S) = \{(\phi(x^{(\ell)}), y^{(\ell)})\}_{\ell=1}^n$  es linealmente separable con probabilidad

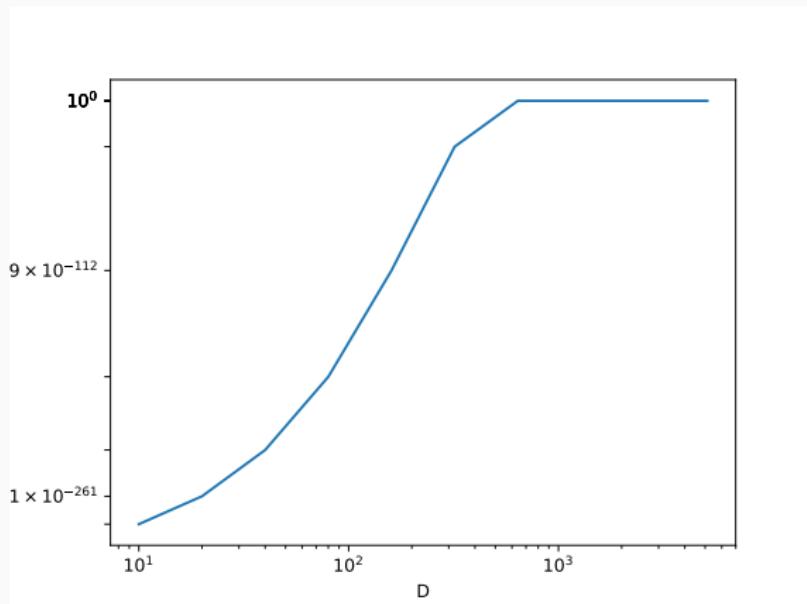
$$P(n, D) = \frac{1}{2^{n-1}} \sum_{k=0}^{D-1} \binom{n-1}{k}.$$

---

<sup>1</sup>T. Cover (1965). *Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition*. IEEE Transactions on Electronic Computers. Page 330.

## Teorema de Cover

- Teorema de Cover para  $n = 1000$  puntos. Eje y: probabilidad de separación lineal. Eje x: dimensionalidad del mapeo.



## ¿Qué características debe tener $\phi$ ?

- Por ejemplo, una transformación que induce un espacio de características altamente dimensional es el **mapa polinomial de grado  $p$** .
- Para  $x \in \mathbb{R}$  éste toma la forma

$$\phi(x) = (1, x, x^2, \dots, x^p)^T \quad (p \text{ características})$$

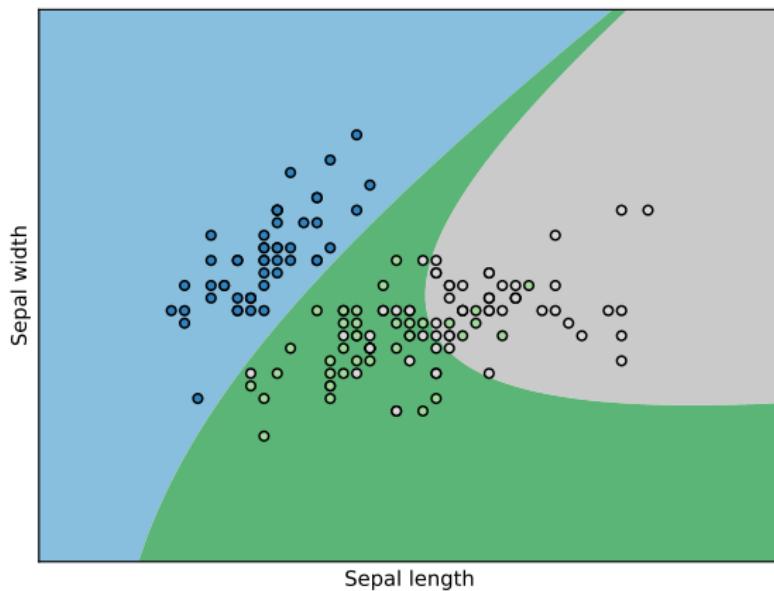
- Para  $x \in \mathbb{R}^2$  y  $p = 2$

$$\phi(x) = (1, x_1, x_2, x_1^2, x_1 \cdot x_2, x_2^2)^T \quad (\approx p^2 \text{ características})$$

- Para  $x \in \mathbb{R}^d$ , el nuevo vector de características tiene  $\approx p^d$  atributos.
- Ejemplo: para  $d = 100$  y  $p = 5$ , el nuevo espacio tiene 10 billones de dimensiones. **Tendremos separabilidad lineal casi segura.**

## SVM + Mapa Polinomial

En el dataset Iris (restringido a sus primeros dos atributos), una SVM entrenada usando un mapa polinomial de grado  $p = 3$ , da origen a las siguientes fronteras en el espacio original



## ¿Qué características debe tener $\phi$ ?

- Otro ejemplo de transformación genérica que induce un espacio de características altamente dimensional es el siguiente “mapa empírico”.
- Dado un conjunto de ejemplos  $\{x^{(\ell)}\}_{\ell=1}^n$ ,  $x^{(\ell)} \in \mathcal{X}$  y una función de similaridad  $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}_0^{+2}$ , un punto  $x$  se puede representar calculando cuán similar es a los ejemplos,

$$s\phi(x) = \left( k(x, x^{(1)}), k(x, x^{(2)}), \dots, k(x, x^{(n)}) \right)^T \quad (n \text{ características})$$

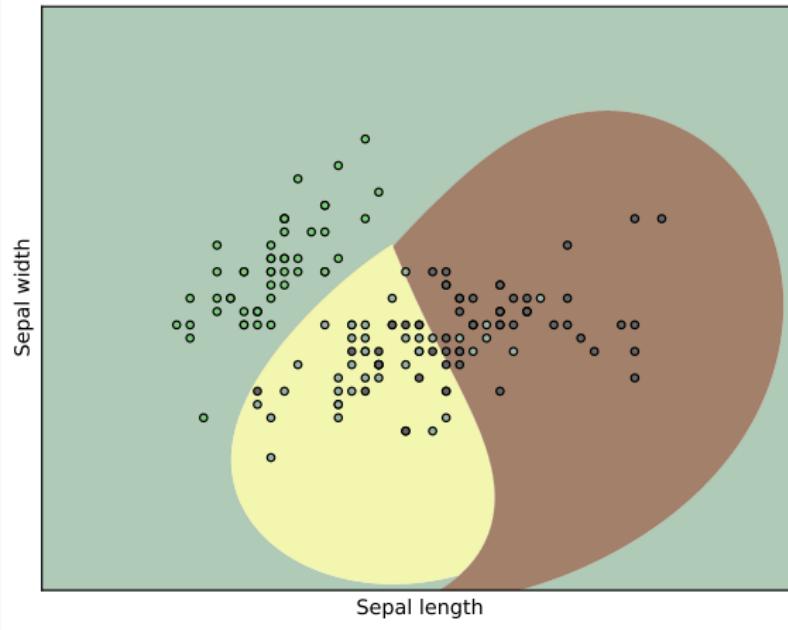
- El tamaño de la representación crece cuando crece el conjunto de puntos (bueno para aumentar la probabilidad de una separación lineal en el teorema de Cover).

---

<sup>2</sup>Una función de similaridad en  $\mathcal{X}$  cuantifica el grado de similitud de dos objetos  $x_1, x_2 \in \mathcal{X}$ . En el caso más simple, una función de similaridad tiene la forma  $k(x_1, x_2) = \varphi(d(x_1, x_2))$  donde  $d$  es una métrica y  $\varphi$  es una función monótonamente decreciente. Varios trabajos teóricos han estudiado posibles definiciones axiomáticas del concepto.

## SVM + Empírico

En el dataset Iris (restringido a sus primeros dos atributos), un mapa empírico, con  $k(x_1, x_2) = \exp(-\gamma \|x_1 - x_2\|^2)$ , da origen a las siguientes fronteras en el espacio original



## SVM y Alta Dimensionalidad

- Como sabemos, trabajar en un espacio de alta dimensionalidad es peligroso. *¿Cómo logra una SVM escapar a este problema?*
- Veremos que, aún si la SVM aprende en un espacio de altísima dimensionalidad, la complejidad del clasificador queda limitada por el número de datos de entrenamiento. Concretamente, la formulación dual tiene  $\mathcal{O}(n)$  parámetros libres, y muchos de ellos son cero al final del entrenamiento.
- La SVM sortea (en buena parte) la maldición de la dimensionalidad porque regulariza el aprendizaje del hiperplano en el espacio transformado.

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2} \|w\|^2 + C \cdot \sum_{\ell} \xi_{\ell} \\ \text{s.t. } & y^{(\ell)}(w^T \phi(x^{(\ell)}) + b) \geq 1 - \xi_{\ell} \quad \forall \ell \\ & \xi_{\ell} \geq 0 \quad \forall \ell \end{aligned}$$

## SVM y Alta Dimensionalidad

- Es fácil demostrar que el problema anterior es equivalente (para cierta elección de  $\lambda$ ) a resolver el problema:

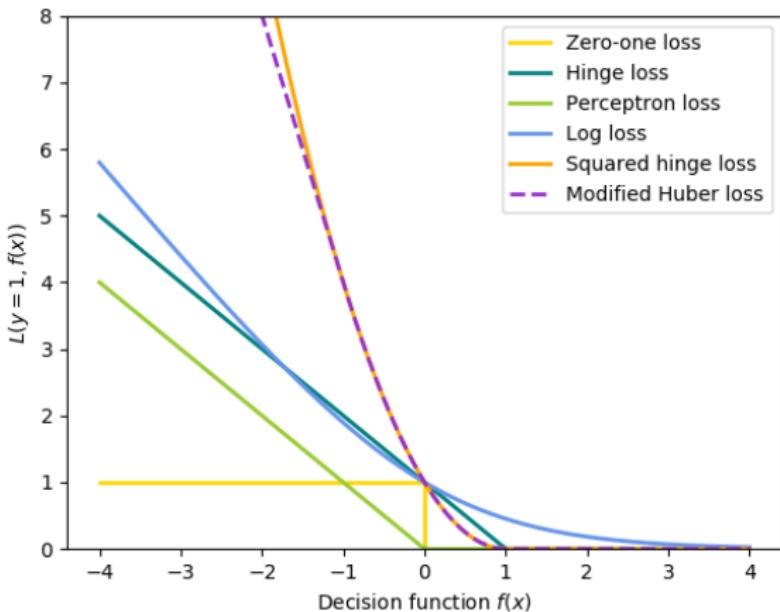
$$\min_{w,b} \sum_{\ell} L(y^{(\ell)}, h(x^{(\ell)})) + \frac{\lambda}{2} \|w\|^2$$

donde  $h(x^{(\ell)}) = w^T \phi(x^{(\ell)}) + b$  y  $L(y, h(x)) = (1 - yh(x))_+$  es una función de pérdida (loss) denominada **hinge loss**.

- Notemos que si  $yh(x) < 0$  (error de clasificación), la hinge loss es siempre positiva y  $> 1$ .
- Si  $0 < yh(x) < 1$  (dato clasificado correctamente, pero con margen insuficiente), la hinge loss también es positiva y queda en  $(0, 1)$ .
- La hinge loss es cero si y solo si  $yh(x) \geq 1$  (dato clasificado correctamente con margen suficiente).

# Hinge Loss

- Hinge loss y otras funciones de pérdida para clasificación.



## **Extensión No-lineal de la SVM via Kernels**

---

## Problemas

- Para aprender de un conjunto de datos  $S = \{(x^{(\ell)}, y^{(\ell)})\}_{\ell=1}^n$ , necesitaremos primero mapearlos al nuevo espacio.
- Si  $\phi(x)$  es de alta dimensionalidad, esto podría ser muy muy costoso computacionalmente.
- Es decir, si bien la regularización del aprendizaje permite escapar a la parte estadística de la maldición de la dimensionalidad, no permite escapar a sus consecuencias computacionales.
- La forma de escapar a este problema se denominará **el truco del kernel** (the kernel trick).

## Kernel Trick

- Supongamos que:
  - Nuestro algoritmo de aprendizaje accede a los datos exclusivamente a través de productos puntos de la forma

$$\langle x, x' \rangle = x^T x'.$$

- Para el mapa  $\phi$  que hemos elegido, existe una función  $k : \mathbb{X} \times \mathbb{X} \rightarrow \mathbb{R}$  tal que:

$$\phi(x)^T \phi(x') = k(x, x') \quad \forall x, x' \in \mathbb{X}.$$

- En este caso, no es necesario ejecutar el mapeo de los datos  $S = \{(x^{(\ell)}, y^{(\ell)})\}_{\ell=1}^n$  para entrenar la máquina, ya **basta sustituir cualquier aparición de  $\langle x, x' \rangle$  por  $k(x, x')$ .**
- Notemos que esta función se computa en el espacio original.

# Kernel Trick

- Una función  $k$  que cumpla con la propiedad anterior se denominará *kernel* del mapa  $\phi$ .
- Por ejemplo, para un mapa polinomial de orden 2, con  $d = 3$

$$\phi(x) = (x_1x_1, x_1x_2, x_1x_3, x_2x_2, x_2x_1, x_2x_3, x_3x_3, x_3x_1, x_3x_2)^T$$

$$\phi(x') = (x'_1x'_1, x'_1x'_2, x'_1x'_3, x'_2x'_2, x'_2x'_1, x'_2x'_3, x'_3x'_3, x'_3x'_1, x'_3x'_2)^T$$

$$\phi(x)^T \phi(x') = (x^T x')^2 .$$

- El kernel anterior se denomina *kernel polinomial de orden 2*.
- Notemos que efectuar el mapeo y luego el cálculo nos cuesta muchísimo más que operar con el kernel.

## Kernel Trick

- Para un mapa polinomial arbitrario, la complejidad de efectuar el mapeo crece exponencialmente en  $p$  y en  $d$ .
- Esto repercutirá también en la complejidad del entrenamiento que suele ser proporcional a la dimensionalidad del espacio de características.
- El **kernel polinomial general**

$$\phi(x)^T \phi(x') = (x^T x' + c_0)^p ,$$

permite efectuar **implícitamente** la aplicación del mapa polinomial sobre **cualquier algoritmo** que esté construido solo en base a productos punto  $\langle x, x' \rangle$  entre datos.

- Un algoritmo que cumpla con esta propiedad se denominará **kernelizable**. ¿Es nuestra SVM kernelizable?

# Kernelización de la SVM

- En el capítulo anterior, vimos que resolver la formulación **primal** de la SVM

$$\begin{aligned} & \min_{w,b,\xi} \frac{1}{2} \|w\|^2 + C \cdot \sum_{\ell} \xi_{\ell} \\ \text{s.t. } & y^{(\ell)}(w^T x^{(\ell)} + b) \geq 1 - \xi_{\ell} \forall \ell \\ & \xi_{\ell} \geq 0 \forall \ell \end{aligned} \tag{3}$$

es equivalente a resolver el problema **dual**

$$\begin{aligned} \max_{\alpha} \quad & W(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y^{(i)} y^{(j)} \langle x^{(i)}, x^{(j)} \rangle \\ \text{s.t.} \quad & \sum_i \alpha_i y^{(i)} = 0, \quad 0 \leq \alpha_i \leq C \forall i . \end{aligned} \tag{4}$$

# Kernelización de la SVM

- Pre-mapeando los datos de entrenamiento con la transformación  $\phi$ , obtenemos

$$\begin{aligned} & \min_{w,b,\xi} \frac{1}{2} \|w\|^2 + C \cdot \sum_{\ell} \xi_{\ell} \\ \text{s.t. } & y^{(\ell)}(w^T \phi(x^{(\ell)}) + b) \geq 1 - \xi_{\ell} \forall \ell \\ & \xi_{\ell} \geq 0 \forall \ell \end{aligned} \tag{5}$$

cuyo **dual** toma la forma

$$\begin{aligned} & \max_{\alpha} W(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y^{(i)} y^{(j)} \langle \phi(x^{(i)}), \phi(x^{(j)}) \rangle \\ \text{s.t. } & \sum_i \alpha_i y^{(i)} = 0, \quad 0 \leq \alpha_i \leq C \forall i. \end{aligned} \tag{6}$$

# Kernelización de la SVM

- Si tenemos un kernel para  $\phi$  el dual

$$\begin{aligned} \max_{\alpha} W(\alpha) &= \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y^{(i)} y^{(j)} k(x^{(i)}, x^{(j)}) \quad (7) \\ \text{s.t. } \sum_i \alpha_i y^{(i)} &= 0, \quad 0 \leq \alpha_i \leq C \forall i. \end{aligned}$$

se puede resolver sin mapear explícitamente los ejemplos al nuevo espacio característico.

# Kernelización de la SVM

- En el capítulo anterior, también vimos que la solución primal  $w, b$  se puede obtener desde la solución dual  $\alpha$  mediante de las ecuaciones

$$w = \sum_{\ell} \alpha_{\ell} y^{(\ell)} x^{(\ell)} \quad (8)$$

$$\begin{aligned} b &= |B|^{-1} \sum_{\ell: B} \left( y^{(\ell)} - w^T x^{(\ell)} \right) \\ &= |B|^{-1} \sum_{\ell: B} \left( y^{(\ell)} - \sum_{\ell'} \alpha_{\ell'} y^{(\ell')} \langle x^{(\ell')} x^{(\ell)} \rangle \right), \end{aligned}$$

con  $B : \{\ell : 0 < \alpha_{\ell} < C\}$ <sup>3</sup>.

---

<sup>3</sup>Para los support vectors con  $\alpha_{\ell}$  estrictamente en  $(0, C)$ , se tiene, por las KKT que  $\beta_{\ell} \neq 0$ , lo que implica  $\xi_{\ell} = 0$ . Esto ultimo implica que  $y^{(\ell)}(w^T x^{(\ell)} + b) = 1$ . Desde aquí se puede despejar  $b$ . Para un cálculo más robusto, se suele tomar la media entre todos los sv estrictos.

# Kernelización de la SVM

- Pre-mapeando la data de entrenamiento, tenemos

$$w = \sum_{\ell} \alpha_{\ell} y^{(\ell)} \phi(x^{(\ell)}) \quad (9)$$

$$b = |B|^{-1} \sum_{\ell: B} \left( y^{(\ell)} - \sum_{\ell'} \alpha_{\ell'} y^{(\ell')} \langle \phi(x^{(\ell')}), \phi(x^{(\ell)}) \rangle \right),$$

de modo que si se conoce un kernel para  $\phi$ , podemos kernelizar el cálculo de  $b$ , pero no de  $w$ !

$$w = \sum_{\ell} \alpha_{\ell} y^{(\ell)} \phi(x^{(\ell)}) \quad (10)$$

$$b = |B|^{-1} \sum_{\ell: B} \left( y^{(\ell)} - \sum_{\ell'} \alpha_{\ell'} y^{(\ell')} k(x^{(\ell')}, x^{(\ell)}) \right),$$

## Kernelización de la Función de Decisión

- En realidad, no necesitamos explícitamente  $w$ . Sólo necesitamos poder clasificar datos nuevos, es decir, computar

$$f(x) = \text{sign} (w^T \phi(x) + b) = \text{sign} \left( \sum_{\ell} \alpha_{\ell} y^{(\ell)} \phi(x^{(\ell)})^T \phi(x) + b \right)$$

- Este cálculo se puede kernelizar fácilmente

$$f(x) = \text{sign} (w^T \phi(x) + b) = \text{sign} \left( \sum_{\ell} \alpha_{\ell} y^{(\ell)} k(x^{(\ell)}, x) + b \right)$$

## Kernelización de la Función de Decisión

- En el capítulo anterior (SVM lineal), la expansión en términos de los support vectors (ejemplos con  $\alpha_\ell \neq 0$ )

$$f(x) = \text{sign} (w^T \phi(x) + b) = \text{sign} \left( \sum_{\ell} \alpha_{\ell} y^{(\ell)} \color{red}{k(x^{(\ell)}, x)} + b \right),$$

era opcional. Podíamos calcular explícitamente  $w$  y aplicar la función clásica

$$f(x) = \text{sign} (w^T \phi(x) + b).$$

- La SVM kernelizada en cambio debe expresarse necesariamente como expansión en términos de los support vectors. Esto hace más serios los problemas computacionales que aparecen cuando el support set es muy grande, porque los support vectors deben ser almacenados explícitamente.

## SVM como Modelo No-Paramétrico

- La forma final de la función de decisión

$$f(x) = \text{sign} (w^T \phi(x) + b) = \text{sign} \left( \sum_{\ell} \alpha_{\ell} y^{(\ell)} k(x^{(\ell)}, x) + b \right),$$

y el problema de optimización correspondiente al criterio de entrenamiento,

$$\begin{aligned} \max_{\alpha} W(\alpha) &= \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y^{(i)} y^{(j)} k(x^{(i)}, x^{(j)}) \\ \text{s.t. } \sum_i \alpha_i y^{(i)} &= 0, \quad 0 \leq \alpha_i \leq C \forall i. \end{aligned}$$

demuestra que la SVM es un método no-paramétrico, en el sentido de que el número de parámetros del modelo aumenta cuando aumenta el número datos de entrenamiento.

## “Complejidad” Máxima de la SVM

- La forma final de la función de decisión

$$f(x) = \text{sign} (w^T \phi(x) + b) = \text{sign} \left( \sum_{\ell} \alpha_{\ell} y^{(\ell)} k(x^{(\ell)}, x) + b \right), \quad (11)$$

demuestra también que la complejidad del modelo resultante estará limitada por el número de datos de entrenamiento, aún si la transformación introducida tiene muchos más atributos, por ejemplo, infinitos.

- Notemos que sin dualidad, el número de parámetros crecería proporcionalmente al número de características del espacio transformado.

## Kernels Admisibles

---

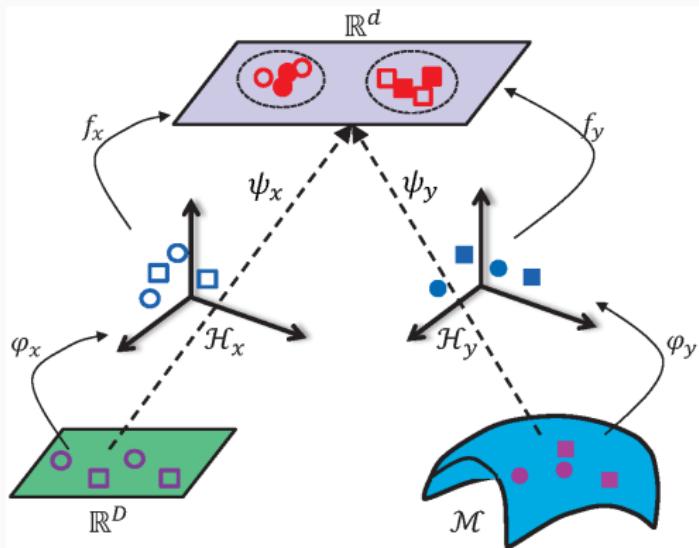
- Hemos visto que si se satisfacen los siguientes requisitos:
  - Nuestro algoritmo de aprendizaje accede a los datos exclusivamente a través de productos puntos de la forma  $\langle x, x' \rangle$ .
  - Para la transformación  $\phi$  que hemos elegido, existe una función  $k : \mathbb{X} \times \mathbb{X} \rightarrow \mathbb{R}$  tal que  $\phi(x)^T \phi(x') = k(x, x'), \forall x, x' \in \mathbb{X}$ .

un método lineal puede extenderse fácilmente vía kernels.

- ¿Cómo encontrar el kernel correspondiente a una transformación  $\phi$ ?
- **Idea:** en vez de diseñar/elegir la transformación  $\phi$  y luego buscarle un kernel, podemos elegir directamente el kernel.

## Rol del Kernel

- La idea anterior está basada en la intuición de que una transformación del espacio de características, debiese ser equivalente a **cambiar la métrica con la que se comparan cosas en el espacio original**.



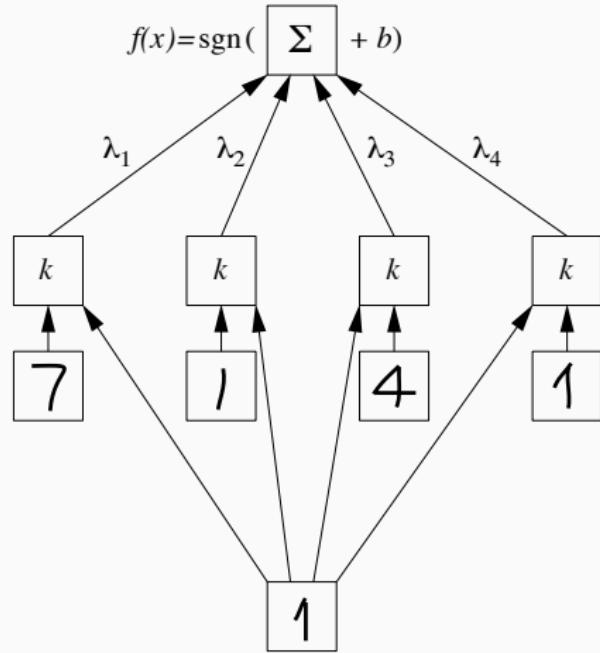
## Rol del Kernel

- Estudiando la función de decisión de nuestra SVM kernelizada

$$f(x) = \text{sign} (w^T \phi(x) + b) = \text{sign} \left( \sum_{\ell} \alpha_{\ell} y^{(\ell)} \color{red}{k(x^{(\ell)}, x)} + b \right),$$

vemos que el rol del kernel es esencialmente el de “comparar” el dato que se quiere clasificar con los datos de entrenamiento. Si  $x$  se parece más a los ejemplos positivos, dominará este signo en la decisión final.

# Rol del Kernel



## Rol del Kernel

- Comparando ahora la función de decisión de nuestra SVM lineal

$$f(x) = \text{sign} \left( \sum_{\ell} \alpha_{\ell} y^{(\ell)} \langle x^{(\ell)}, x \rangle + b \right),$$

con la versión kernelizada

$$f(x) = \text{sign} \left( \sum_{\ell} \alpha_{\ell} y^{(\ell)} k(x^{(\ell)}, x) + b \right),$$

vemos que el rol del kernel es generalizar el producto punto ordinario  $\langle x^{(\ell)}, x \rangle = x^{(\ell)T} x$  a una operación no-lineal en  $\mathbb{X}$ .

- ¿Qué generalizaciones del producto punto resultan admisibles en modo siga siendo válida la formulación de la máquina de aprendizaje?

## Admisibilidad de un Kernel

- Para la maquinaria lineal se preserve, un kernel  $\mathbb{X} \times \mathbb{X} \rightarrow \mathbb{R}$  definido sobre el espacio de características original es admisible si podemos encontrar un mapa  $\phi$  entre  $\mathbb{X}$  y un espacio vectorial  $\mathcal{Z}$  equipado con un producto punto  $\langle a, b \rangle$  tal que

$$k(x, x') = \langle \phi(x), \phi(x') \rangle.$$

- Para permitir la máxima generalidad posible, admitiremos que producto punto en  $\mathcal{Z}$  sea lo más general posible

$$\langle a, b \rangle = \langle b, a \rangle \quad \forall a, b \in \mathcal{Z}$$

$$\langle a, a \rangle \geq 0, \text{ con } \langle a, a \rangle = 0 \Leftrightarrow a = 0 \quad \forall a \in \mathcal{Z}$$

$$\langle \alpha_1 a_1 + \alpha_2 a_2, b \rangle = \alpha_1 \langle a_1, b \rangle + \alpha_2 \langle a_2, b \rangle \quad \forall a, b \in \mathcal{Z}, \forall \alpha_1, \alpha_2$$

$$\langle a, \beta_1 b_1 + \beta_2 b_2 \rangle = \beta_1 \langle a, b_1 \rangle + \beta_2 \langle a, b_2 \rangle \quad \forall a, b \in \mathcal{Z}, \forall \beta_1, \beta_2$$

# Matriz Gram y Kernels Positivos

- Para cualquier colección finita de puntos  $S = \{x^{(\ell)}, y^{(\ell)}\}_{\ell=1}^n$  un kernel  $k$ , induce una matriz de  $\mathbf{K} \in \mathbb{R}^{n \times n}$  con  $K_{ij} = k(x^{(i)}, x^{(j)})$  que se denomina *Matriz Gram* asociada al kernel  $k$  en el dataset  $S$ .

## Definición (Kernel Positivo)

Un kernel se dice positivo, si es simétrico y si para cualquier  $n$  y cualquier  $S = \{x^{(\ell)}, y^{(\ell)}\}_{\ell=1}^n$ , su matriz Gram correspondiente es semi-definida positiva, es decir, si  $\forall \alpha \in \mathbb{R}^n$

$$\alpha^T \mathbf{K} \alpha \geq 0.$$

# Teorema de Mercer

## Teorema (Mercer, 1909)<sup>4</sup>

Un kernel es admisible si y sólo si es positivo.



---

<sup>4</sup> James Mercer. *Functions of positive and negative type and their connection with the theory of integral equations*, Philosophical Transactions of the Royal Society A, 209, 441–458, 1909.

## Teorema de Mercer

- (Admissible  $\Rightarrow$  Positivo)

$$\begin{aligned}\alpha^T \mathbf{K} \alpha &= \sum_{i,j} \alpha_i \alpha_j k(x^{(i)}, x^{(j)}) && (12) \\&= \sum_{i,j} \alpha_i \alpha_j \langle \phi(x^{(i)}), \phi(x^{(j)}) \rangle \quad (\text{el kernel es admisible}) \\&= \left\langle \sum_i \alpha_i \phi(x^{(i)}), \sum_j \alpha_j \phi(x^{(j)}) \right\rangle \quad (\text{bi-linealidad}) \\&= \langle a, a \rangle \text{ para algún } a \in \mathbb{Z} \quad (\text{clausura de } \mathbb{Z}) \\ \therefore \alpha^T \mathbf{K} \alpha &\geq 0.\end{aligned}$$

Además, por la simetría del producto punto,

$$k(x^{(i)}, x^{(j)}) = \langle \phi(x^{(i)}), \phi(x^{(j)}) \rangle = \langle \phi(x^{(j)}), \phi(x^{(i)}) \rangle = k(x^{(j)}, x^{(i)}) \quad (13)$$

## Teorema de Mercer

- La demostración (Positivo  $\Rightarrow$  Admisible) es más complicada y no es única, en el sentido de que, dado un kernel positivo, es posible construir muchos mapas  $\phi$  que satisfacen las propiedades requeridas.
- La más famosa se le debe a Mercer ...

# Teorema de Mercer

## Teorema (Mercer, 1909)

Para cualquier kernel positivo  $k : \mathbb{X} \times \mathbb{X}$ , existe una colección (posiblemente infinita) de funciones ortonormales  $\{\psi_k\}_k$  ( $\psi_k \in L_2(\mathbb{X})$ ) y constantes  $\{\lambda_k\}_k$ ,  $\lambda_k > 0$ , tales que

$$k(x^{(i)}, x^{(j)}) = \sum_k \lambda_k \psi_k(x^{(i)}) \psi_k(x^{(j)}) \quad (14)$$

para (casi) todo  $x^{(i)}, x^{(j)} \in \mathbb{X}$ . Además, las funciones  $\{\psi_k\}_k$  corresponden a las funciones propias del operador integral de Hilbert–Schmidt  $T_k : L_2(\mathbb{X}) \rightarrow L_2(\mathbb{X})$

$$T_k [f] (x) = \int_{\mathbb{X}} k(x, x') f(x) f(x') d\mu(x'), \quad (15)$$

y las constantes  $\{\lambda_k\}_k$  son sus correspondientes valores propios.

## Teorema de Mercer

- Lo anterior implica que para un kernel positivo, podemos siempre construir una transformación de la forma

$$\phi(x) = \left( \sqrt{\lambda_1} \psi_1, \sqrt{\lambda_2} \psi_2, \dots, \sqrt{\lambda_N} \psi_N \right)^T,$$

tal que

$$\phi(x^{(i)})^T \phi(x^{(j)}) = \sum_k \lambda_k \psi_k(x^{(i)}) \psi_k(x^{(j)}) = k(x^{(i)}, x^{(j)}).$$

- Es importante notar que  $N$ , la dimensionalidad de  $\phi(x)$  puede ser infinita.

# Kernels Populares

Selección de un kernel en *sklearn*.



The scikit-learn logo consists of a blue circle containing a white 's' shape, followed by the word 'scikit' in a small sans-serif font and 'learn' in a larger orange sans-serif font.

Home Installation Documentation Examples Google Custom Search

Previous 4.7 Kernel A... Next 4.8 Transfer... Up 4. Dataset M...

scikit-learn v0.20.0 Other versions

Please cite us if you use the software.

## 4.8. Pairwise metrics, Affinities and Kernels

The `sklearn.metrics.pairwise` submodule implements utilities to evaluate pairwise distances or affinity of sets of samples.

This module contains both distance metrics and kernels. A brief summary is given on the two here.

Distance metrics are functions  $d(a, b)$  such that  $d(a, b) < d(a, c)$  if objects  $a$  and  $b$  are considered "more similar" than objects  $a$  and  $c$ . Two objects exactly alike would have a distance of zero. One of the most popular examples is Euclidean distance. To be a 'true' metric, it must obey the following four conditions:

1.  $d(a, b) \geq 0$ , for all  $a$  and  $b$
2.  $d(a, b) = 0$ , if and only if  $a = b$ , positive definiteness
3.  $d(a, b) = d(b, a)$ , symmetry
4.  $d(a, c) \leq d(a, b) + d(b, c)$ , the triangle inequality

Kernels are measures of similarity, i.e.  $s(a, b) \geq s(a, c)$  if objects  $a$  and  $b$  are considered "more similar" than objects  $a$  and  $c$ . A kernel must also be positive semi-definite.

## Kernels Populares

- El *kernel lineal*  $k(x, x') = x^T x'$  (corresponde a **mantener** la geometría del espacio original).
- El kernel *polinomial*, con parámetros  $\gamma$  (escala),  $c_0$  and  $p$

$$k(x, x') = (\gamma \cdot x^T x' + c_0)^p.$$

- El kernel *RBF (or Gausiano)*, con parámetro  $\gamma$  (escala)

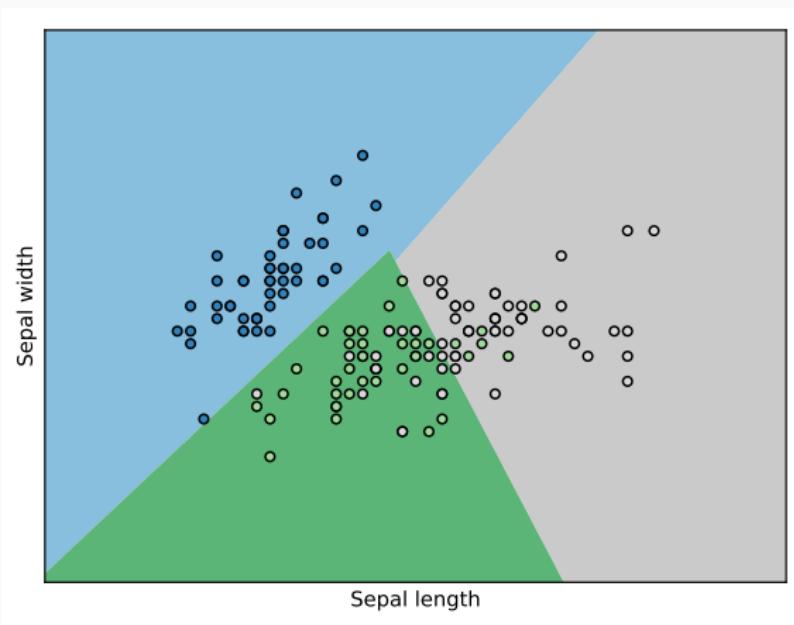
$$k(x, x') = \exp(-\gamma \|x - x'\|^2).$$

- El kernel *Laplaciano*, con parámetro  $\gamma$  (escala)

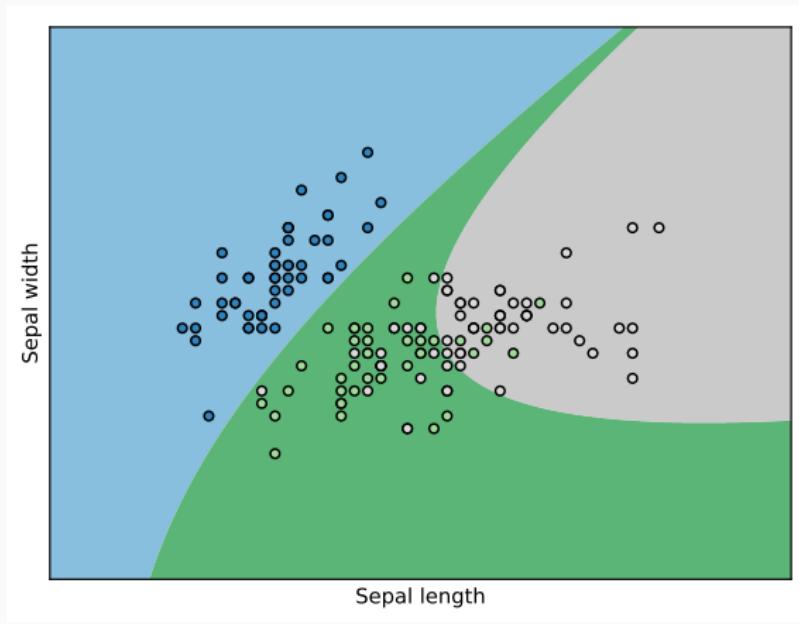
$$k(x, x') = \exp(-\gamma \|x - x'\|_{l1}).$$

donde  $\|x - x'\|_{l1} = \sum_j |x_j - x'_j|$ .

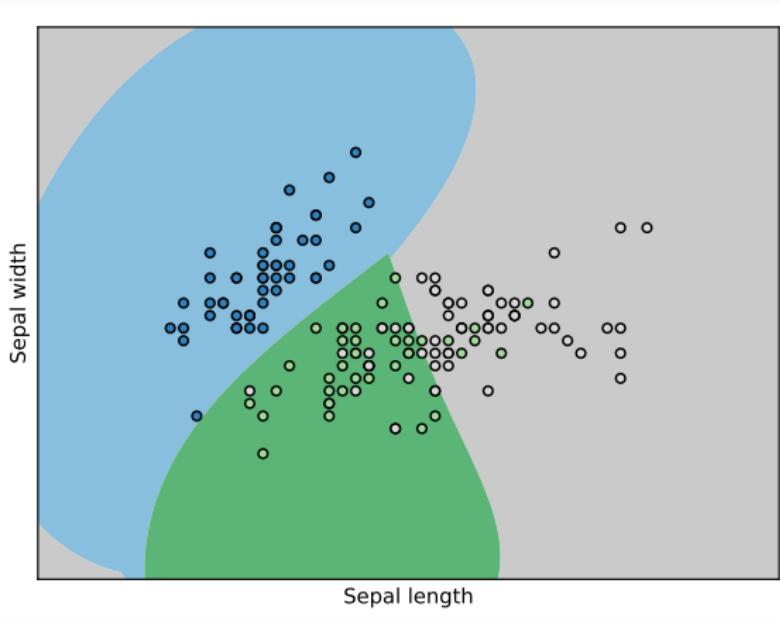
# SVM + Kernel Lineal



## SVM + Kernel Poly



# SVM + Kernel RBF



## Más Kernels

- El kernel denominado *sigmoidal* con parámetros  $\gamma$  y  $c_0$

$$k(x, x') = \tanh(\gamma \cdot x^T x' + c_0),$$

no es un kernel admisible, pero en ocasiones da buenos resultados.

- Suma de Kernels. La suma de kernels admisibles con  $\alpha_1, \alpha_2 > 0$

$$k(x, x') = \alpha_1 k_1(x, x') + \alpha_2 k_2(x, x'),$$

es admisible.

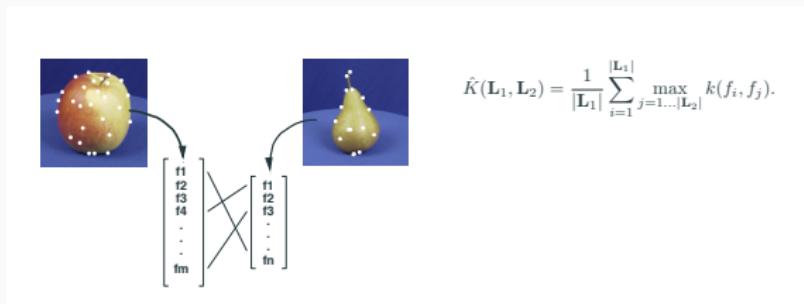
- El producto de dos kernels admisibles  $k_1$  and  $k_2$ , and  $\alpha_1, \alpha_2 > 0$

$$k(x, x') = \alpha_1 \alpha_2 \cdot k_1(x, x') \cdot k_2(x, x').$$

es admisible.

# Kernels Especializados

- Muchos kernels (que respetan las propiedades necesarias) han sido desarrollados para aplicaciones específicas.
- Una ventaja de la teoría que hemos revisado es que si bien  $\phi$  mapea a un espacio vectorial,  $k$  puede estar definido sobre espacios  $\mathbb{X}$  que no necesariamente lo son.
- Ejemplo: Pyramid Match Kernel usando en Computer Vision.



# Kernels Especializados

- Una ventaja de la teoría que hemos revisado es que si bien  $\phi$  mapea a un espacio vectorial,  $k$  puede estar definido sobre espacios  $\mathbb{X}$  que no necesariamente lo son.
  - Ejemplo: String kernels diseñados para clasificación de texto y bio-informática.

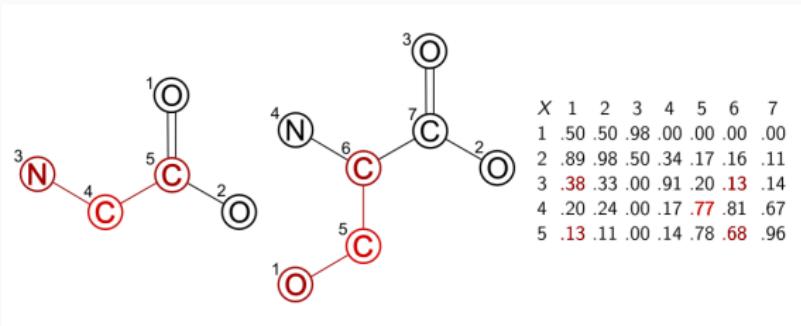
$$k(s_1, s_2) = w_7 + w_1 + w_2 + w_2 + w_3$$

S1 → AGTCAGATAGAGGACATCAGTAGACAGATTAAA →  
          |||||           |||           |||  
S2 → TTATAGATAGACAAAGACATCAGTAGACTTATT →

**Figure 4.1:** Given two sequences  $x_1$  and  $x_2$  of equal length, the kernel consists of a weighted sum to which each match in the sequences makes a contribution  $w_B$  depending on its length  $B$ , where longer matches contribute more significantly.

# Kernels Especializados

- Una ventaja de la teoría que hemos revisado es que si bien  $\phi$  mapea a un espacio vectorial,  $k$  puede estar definido sobre espacios  $\mathbb{X}$  que no necesariamente lo son.
- Ejemplo: Graph kernels diseñados para bio-informática.



# Aprendizaje del Kernel

- Una corriente importante en la literatura de SVMs consiste en el aprendizaje del kernel, idea muy en línea con lo que hoy se denomina **metric learning**.
- Dos formulaciones populares consisten en aprender  $k$ 
  - Como suma de kernels positivos, es decir, aprender del espacio

$$\mathcal{K} = \left\{ \sum_{i=1}^p \mu_i k_i : \mu \in \mathcal{M} \right\}.$$

con  $\mathcal{M} = \{\mu \geq 0 : \|\mu\|_p \leq \gamma\}$  y  $p \geq 1$ .

- Como suma de productos de kernels positivos,

$$\mathcal{K} = \left\{ \sum_{i=1}^p \mu_i \mu_j k_i k_j : \mu \in \mathcal{M} \right\}.$$

con  $\mathcal{M} = \{\mu \geq 0 : \|\mu\|_p \leq \gamma\}$  y  $p \geq 1$ .

# Aprendizaje del Kernel

- El criterio de entrenamiento de una SVM

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y^{(i)} y^{(j)} k(x^{(i)}, x^{(j)}) \text{ s.t. } \alpha \in \mathcal{C}$$

depende de una matriz  $\mathbf{K}$  inducida por el kernel

$$\max_{\alpha} W(\alpha) = \alpha^T \mathbf{1} - \frac{1}{2} \alpha^T \mathbf{K} \alpha \text{ s.t. } \alpha \in \mathcal{C} \quad (16)$$

$$K_{ij} = y_i y_j k(x^{(i)}, x^{(j)})^5.$$

- Podemos aprender el kernel  $k_\mu \in \mathcal{K}$ , introduciendo  $\mu$  en el objetivo.

$$\max_{\alpha, \mu} W(\alpha, \mu) = \alpha^T \mathbf{1} - \frac{1}{2} \alpha^T \mathbf{K}_\mu \alpha \text{ s.t. } \alpha \in \mathcal{C}, \mu \in \mathcal{M},$$

donde  $\mathbf{K}_\mu$  es la matriz inducida por  $k_\mu$ .

<sup>5</sup>Escribimos  $\mathcal{C}$  para representar la región factible.

# Aprendizaje del Kernel

- Típicamente  $W(\alpha, \mu)$  es diferenciable en  $\mu$  y el algoritmo de entrenamiento toma la forma

---

**Algorithm 1:** Algoritmo Genérico de Aprendizaje del Kernel.

---

- 1 Inicializar  $\mu \in \mathcal{M}$ .
  - 2 **do**
  - 3     Para  $\mu$  fijo, actualizar  $\alpha$  resolviendo
$$\max_{\alpha, \mu} W(\alpha, \mu) \text{ s.t. } \alpha \in \mathcal{C}.$$
  - 4     Para  $\alpha$  fijo, actualizar  $\mu$  via gradiente  $\mu \leftarrow \mu + \eta \frac{\partial W(\alpha, \mu)}{\partial \mu}$ .
  - 5     Proyectar  $\mu$  en  $\mathcal{M}$ .
  - 6 **while** *not convergence*;
-

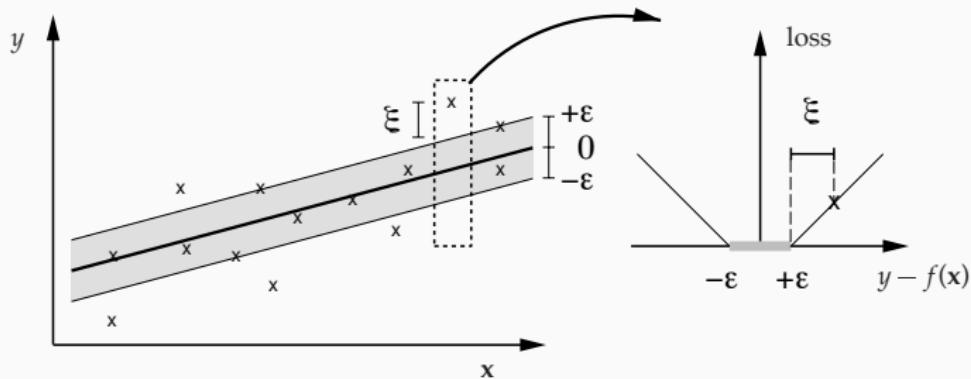
## Extensiones para Regresión

---

- La efectividad de una SVM para resolver problemas de clasificación está basada en los siguientes ingredientes:
  - Un criterio de entrenamiento que prevenga overfitting.
  - Una formulación que permita la introducción de algoritmos de entrenamiento eficientes.
  - Una formulación que permita extensiones no-lineales simples vía kernels y que de paso generen modelos no-paramétricos.
- Nos gustaría obtener algo similar para regresión.
- Hay muchas formulaciones posibles y discutiremos brevemente dos: **C-SVR** y **Kernel Ridge Regression**.

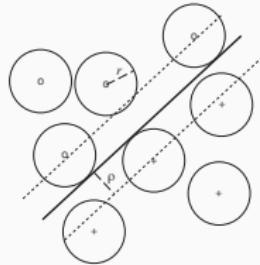
## $\epsilon$ -Insensitive Loss

- Esta formulación se basa en el uso de (1) una función de error (loss function) robusta y (2) un regularizador clásico.
- La función de error adoptada se denomina  $\epsilon$ -insensitive loss y tiene la forma  $L(y, f(x)) = (|y - f(x)| - \epsilon)_+$ .
- $\epsilon$  será un hiper-parámetro a calibrar con algún método de selección de modelos (e.g. validación cruzada).



## $\epsilon$ -Insensitive Loss

- La elección del valor absoluto en vez del error cuadrático garantiza robustez frente a observaciones extremas (outliers).
- Esta elección permitirá obtener restricciones lineales en el dual.
- La zona de insensibilidad intenta reproducir la tolerancia que otorga un margen amplio a perturbaciones en la de entrenamiento (perturbando los datos de la forma  $x^{(i)} + \epsilon$  obtendremos casi la misma solución).



- Esta última elección permitirá obtener menos support vectors en la versión kernelizada.

- Dado un conjunto de entrenamiento  $S = \{(x^{(\ell)}, y^{(\ell)})\}_{\ell=1}^n$  con  $y^{(\ell)} \in \mathbb{R}$ , el modelo toma la forma

$$f(x) = w^T x + b \quad (17)$$

y el entrenamiento corresponde a la solución de

$$\min_{w,b} \sum_{\ell} \left( |y^{(\ell)} - f(x^{(\ell)})| - \epsilon \right)_+ + \frac{\lambda}{2} \|w\|^2$$

donde  $\|w\|^2$  es el regularizador clásico de Tikhonov.

## $C$ -SVR (Primal)

- Es fácil ver que el problema anterior se puede escribir como

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2} \|w\|^2 + C \sum_{\ell} (\xi_{\ell}^+ + \xi_{\ell}^-) \\ \text{s.t.} \quad & f(x^{(\ell)}) - y^{(\ell)} \leq \epsilon + \xi_{\ell}^+ \\ & y^{(\ell)} - f(x^{(\ell)}) \leq \epsilon + \xi_{\ell}^- \\ & \xi_{\ell}^+, \xi_{\ell}^- \geq 0. \end{aligned}$$

- Esta formulación permitirá hacer uso de la misma teoría de optimización que usamos para la kernelización de la SVM para clasificación.

## $C$ -SVR (Dual)

- En efecto, el dual de este problema tiene la forma

$$\begin{aligned} \max_{\alpha^+, \alpha^-} \quad & \sum_{\ell} y^{(\ell)} (\alpha_{\ell}^- - \alpha_{\ell}^+) - \sum_{\ell} \epsilon (\alpha_{\ell}^- + \alpha_{\ell}^+) \\ & - \frac{1}{2} \sum_{\ell, m} (\alpha_{\ell}^- - \alpha_{\ell}^+) (\alpha_m^- - \alpha_m^+) \langle x^{(\ell)}, x^{(m)} \rangle \\ \text{s.t.} \quad & \sum_{\ell} (\alpha_{\ell}^+ - \alpha_{\ell}^-) = 0, \quad 0 \leq \alpha_{\ell}^+, \alpha_{\ell}^- \leq C \forall \ell, \end{aligned}$$

que admite una kernelización inmediata

$$\begin{aligned} \max_{\alpha^+, \alpha^-} \quad & \sum_{\ell} y^{(\ell)} (\alpha_{\ell}^- - \alpha_{\ell}^+) - \sum_{\ell} \epsilon (\alpha_{\ell}^- + \alpha_{\ell}^+) \\ & - \frac{1}{2} \sum_{\ell, m} (\alpha_{\ell}^- - \alpha_{\ell}^+) (\alpha_m^- - \alpha_m^+) \mathbf{k}(x^{(\ell)}, x^{(m)}) \\ \text{s.t.} \quad & \sum_{\ell} (\alpha_{\ell}^+ - \alpha_{\ell}^-) = 0, \quad 0 \leq \alpha_{\ell}^+, \alpha_{\ell}^- \leq C \forall \ell, \end{aligned}$$

## Kernelización del Predictor

- Usando las KKT, se obtiene que

$$w = \sum_{\ell} \alpha_{\ell} x^{(\ell)},$$

con  $\alpha_{\ell} = (\alpha_{\ell}^- - \alpha_{\ell}^+)$  de modo que el predictor  $f(x) = w^T x + b$  admite también un kernelización inmediata

$$f(x) = \sum_{\ell} \alpha_{\ell} \mathbf{k}(x^{(\ell)}, x) + b,$$

con todas las consecuencias que hemos discutido para el caso de clasificación.

## KKT

- Usando las KKT, se obtiene también que

$$\alpha^+ \left( \epsilon + \xi_\ell^+ - y^{(\ell)} + w^T x^{(\ell)} + b \right) = 0$$

$$\alpha^- \left( \epsilon + \xi_\ell^- + y^{(\ell)} - w^T x^{(\ell)} - b \right) = 0$$

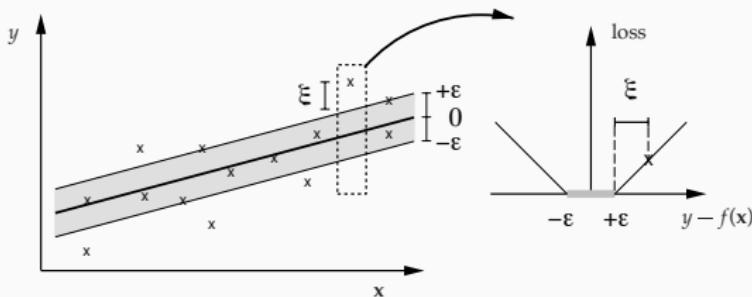
$$\xi_\ell^+ (C - \alpha^+) = 0$$

$$\xi_\ell^- (C - \alpha^-) = 0,$$

- Estas ecuaciones permiten encontrar el valor de  $b$ .
- Además, permiten caracterizar los support vectors!

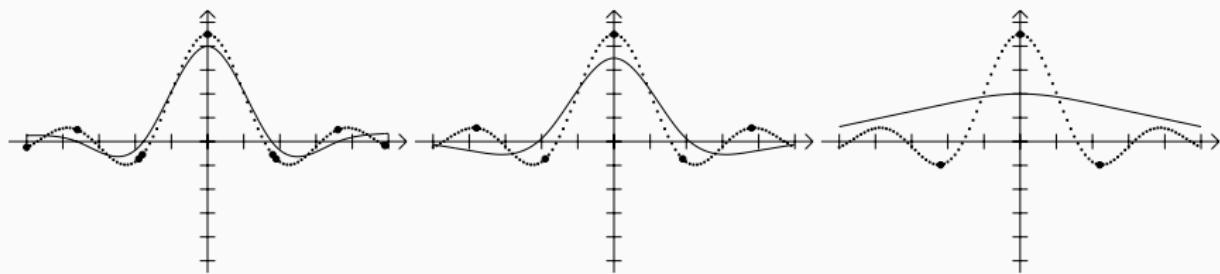
# Tipos de Support Vector

- Si para un ejemplo  $|y^{(\ell)} - f(x^{(\ell)})| < \epsilon$  (está estrictamente dentro del tubo de insensibilidad), el ejemplo no puede ser un support vector,  $\alpha^+ = \alpha^- = 0$ .
- Los support vectors tales que  $\alpha^+ = C$  ó  $\alpha^- = C$  ( $\alpha^+\alpha^-$  es siempre 0), están estrictamente fuera del tubo.
- Los support vectors tales que  $0 < \alpha^+ < C$  ó  $0 < \alpha^- < C$  están sobre el tubo.



## Support Vectors vs $\epsilon$

- En general, a un mayor valor de  $\epsilon$  obtendremos un modelo mucho más simple, con menor cantidad de support vectors.



**Figure 1:** Número de SVs (puntos negros) para diferentes valores de  $\epsilon$  en un experimento con una SVR entrenada para aproximar la función  $\text{sinc}(x) = \sin(x)/x$ .

# Kernel Ridge Regression

- **Idea:** Sacrificar la dispersión inducida por el tubo de insensibilidad para obtener una solución analítica del dual.
- Dado un conjunto de entrenamiento  $S = \{(x^{(\ell)}, y^{(\ell)})\}_{\ell=1}^n$  con  $y^{(\ell)} \in \mathbb{R}$ , el modelo toma la forma

$$f(x) = w^T x + b \tag{18}$$

y el entrenamiento corresponde a la solución de

$$\min_{w,b} \sum_{\ell} \left( y^{(\ell)} - f(x^{(\ell)}) \right)^2 + \frac{\lambda}{2} \|w\|^2$$

que es simplemente el modelo que hemos denominado **ridge regression**

## Kernel Ridge Regression

- La forma matricial del criterio de entrenamiento (esta forma requiere la eliminación de  $b$  del problema, que se puede lograr de varias formas),

$$\min_{w,b} \|y - Xw\|^2 + \frac{\lambda}{2} \|w\|^2$$

permite obtener la solución rápidamente,

$$w = (X^T X + \lambda I)^{-1} X^T y$$

- Esta solución no es directamente kernelizable. Notemos que  $(X^T X)_{ij} \neq \langle x^{(i)}, x^{(j)} \rangle$ .

## Kernel Ridge Regression

- Una solución aparece al recordar la identidad

$$(P^{-1} + B^T R^{-1} B)^{-1} B^T R^{-1} = P B^T (B P B^T + R)^{-1}$$

que aplicada a nuestra ecuación  $w = (X^T X + \lambda I)^{-1} X^T y$  produce

$$w = X^T (X X^T + \lambda I)^{-1} y$$

- Notemos que ahora  $(X X^T)_{ij} = \langle x^{(i)}, x^{(j)} \rangle$  y por lo tanto esta parte se puede kernelizar.

# Kernel Ridge Regression

- Además, el predictor se puede escribir como

$$f(x) = w^T x + b = x^T w + b = x^T X^T (X X^T + \lambda I)^{-1} y + b$$

que es equivalente a

$$f(x) = \sum_{\ell} \alpha_{\ell} \langle x^{(\ell)}, x \rangle + b,$$

con

$$\alpha = (X X^T + \lambda I)^{-1} y.$$

- Desafortunadamente, ahora  $\alpha$  es denso.

# Referencias

-  Alex J Smola and Bernhard Schölkopf.  
**Learning with Kernels.**  
MIT press, 2001.
-  Sebastian Raschka.  
**Python Machine Learning.**  
Packt Publishing Ltd, 2015.
-  M Bishop Christopher.  
**Pattern Recognition and Machine Learning.**  
Springer-Verlag New York, 2016.