Machine Learning 101

Métodos Kernel

Felipe Alonso Atienza

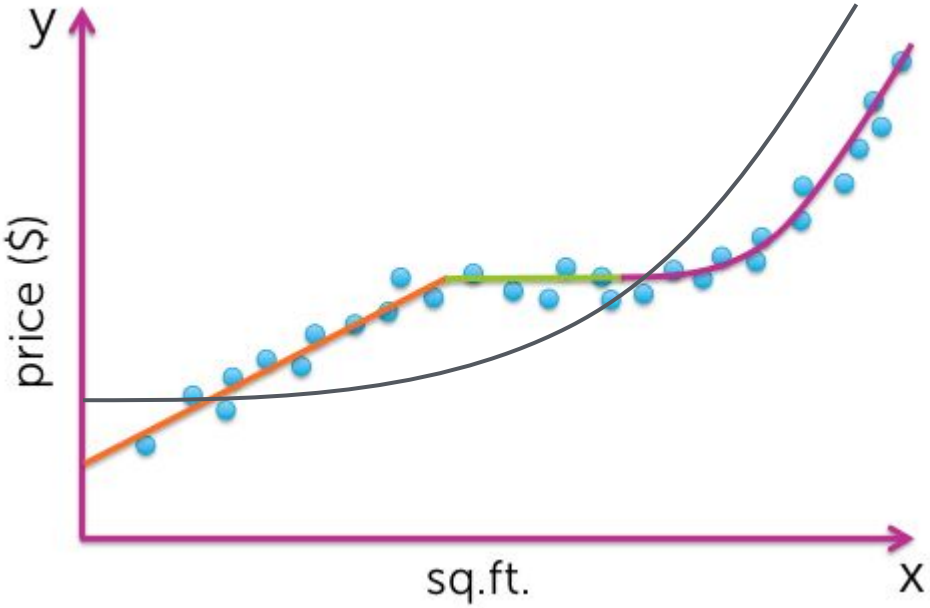
Data Scientist @BBVA

Índice

1. **K-nn en regresión**
2. Kernels y SVM
3. Otros algoritmos con Kernels

Motivación

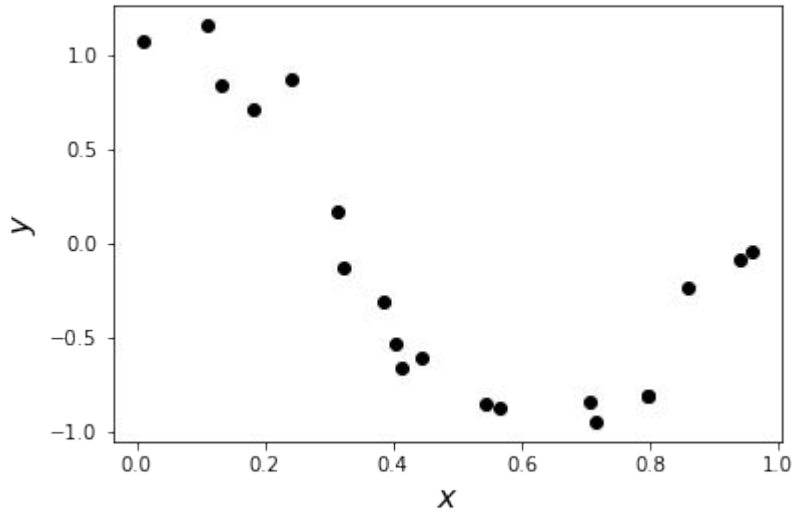
- Modelos paramétricos no siempre resultan adecuados
  - Modelo basado en datos (no paramétrico)



$$\hat{y} \neq \omega_0 + \omega_1 x + \omega_2 x^2$$

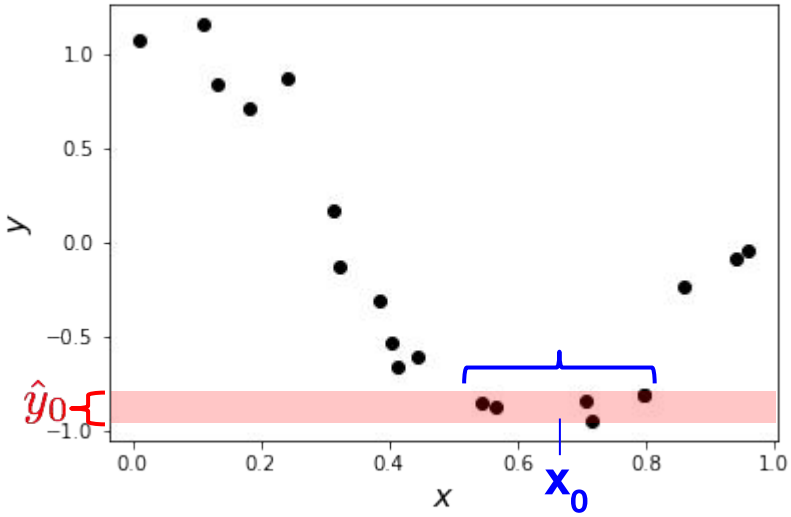
Opciones

- Solución 1: árboles de decisión en regresión
- Solución 2: K-nn regresión



K-NN regresión

- Queremos estimar el precio (y) a partir de sqm (x), en un nuevo punto x_0



1. Buscamos los K vecinos de x_0

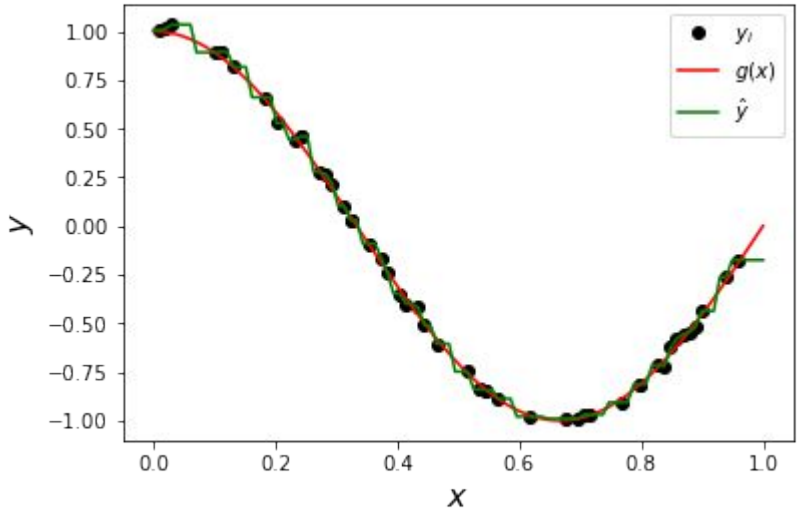
$$d_i = ||x_0 - x_i||_2$$

2. Valor estimado es la media de los vecinos

$$\hat{y}_0 = \frac{1}{K} \sum_{i=1}^{K} y_i$$

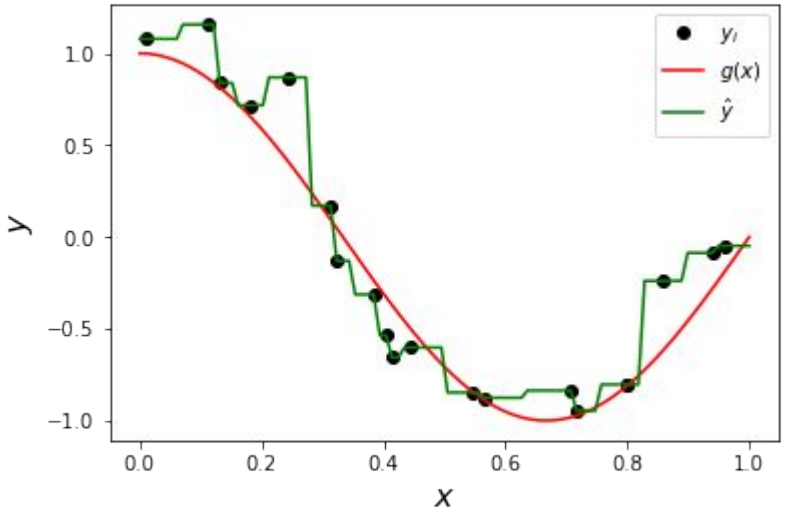
K = 1

- Buen ajuste si hay mucha densidad de puntos y bajo ruido



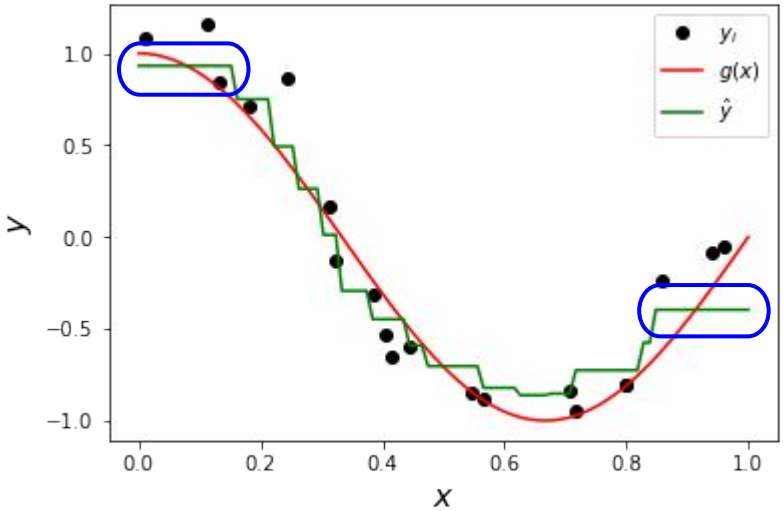
K = 1

- Región sin ejemplos, mal ajustada
- Sensible al ruido



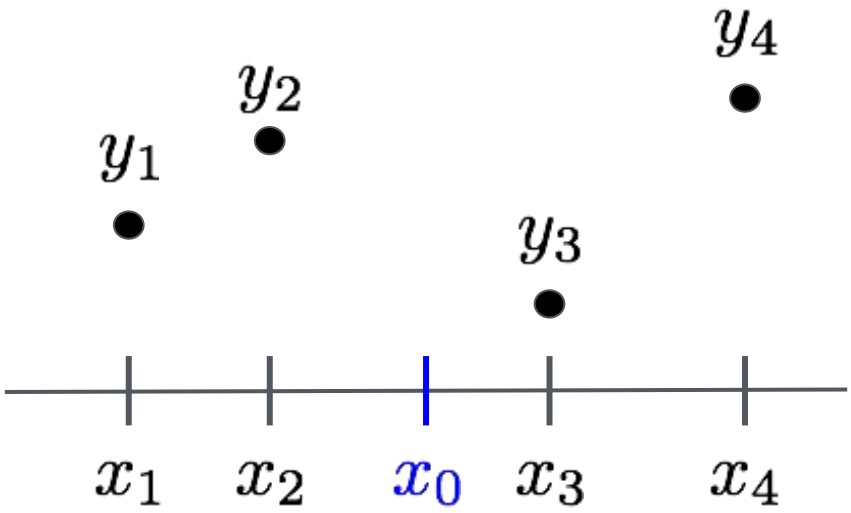
K = 5

- Como sabemos, deberíamos aumentar K
- Pero ello nos produce problemas de borde



- Solución: ponderar estimación por distancia entre vecinos
  - Dar más peso a los más cercanos
  - Reducir los más alejados

Ponderar la estimación (K=4)



$$y_0 = \frac{\theta_1 y_1 + \theta_2 y_2 + \theta_3 y_3 + \theta_4 y_4}{\sum_{i=1}^{K} \theta_i}$$

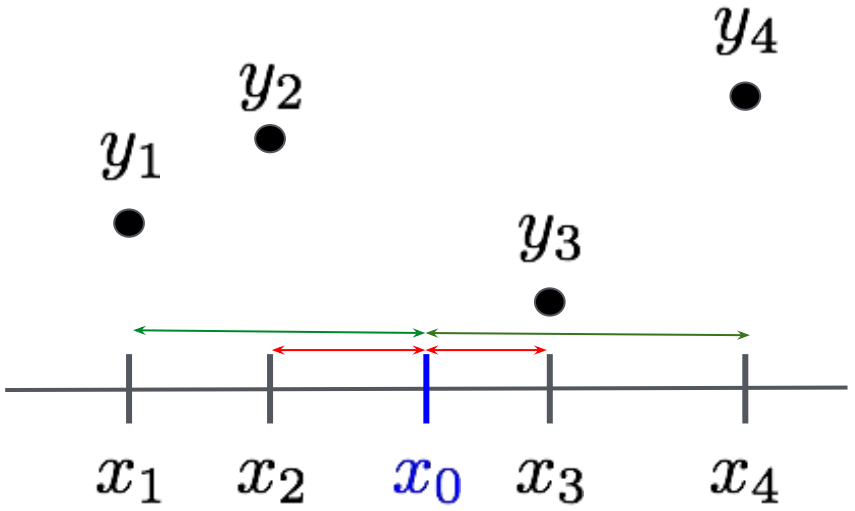
¿Cómo elegimos θ_i?

- En función de la distancia

$$\theta_i = \frac{1}{d_i}$$

donde

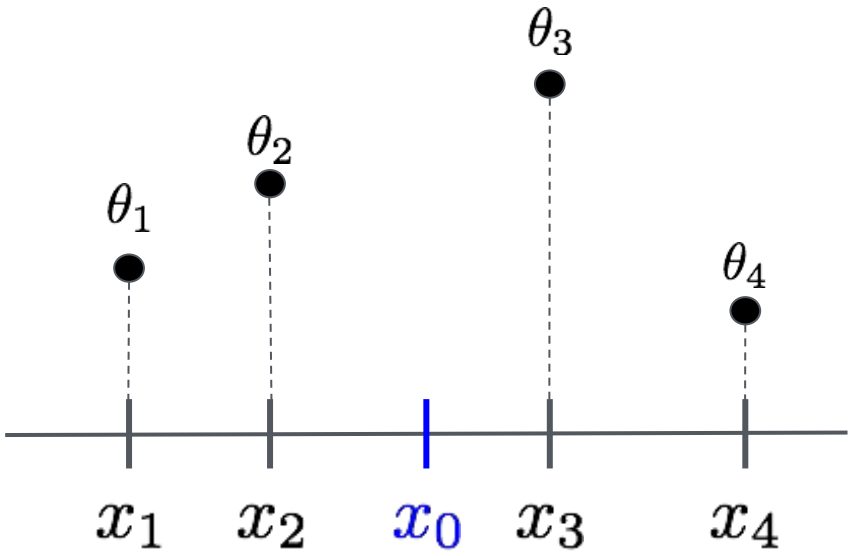
$$d_i = ||x_0 - x_i||_2$$



$$y_0 = \frac{\theta_1 y_1 + \theta_2 y_2 + \theta_3 y_3 + \theta_4 y_4}{\sum_{i=1}^{K} \theta_i}$$

¿Cómo elegimos θ_i?

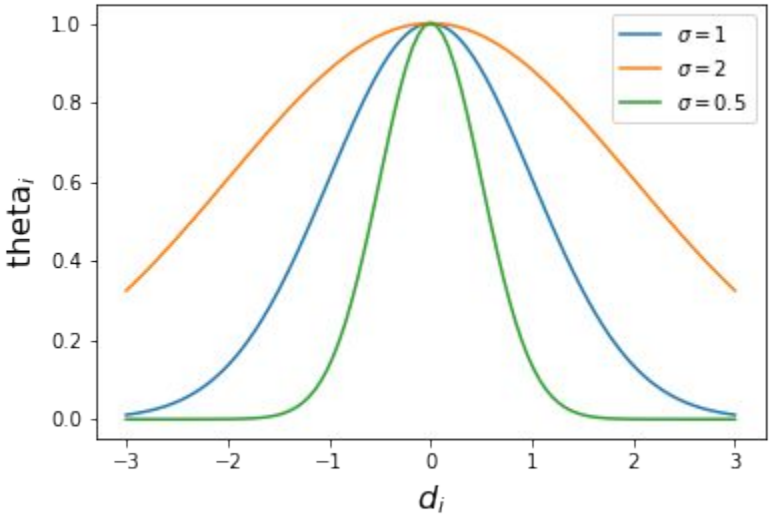
- Si $d_i \downarrow \Rightarrow \theta_i \uparrow$
- Si $d_i \uparrow \Rightarrow \theta_i \downarrow$



Otras opciones

- Podemos utilizar otras medidas de **similitud**

$$\theta_i(\sigma) = e^{-\frac{||\mathbf{x}_0 - \mathbf{x}_i||_2^2}{2\sigma^2}}$$



Kernel RBF

- RBF: Radial Basis Function

$$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{||\mathbf{x}_i - \mathbf{x}_j||_2^2}{2\sigma^2}}$$

que se expresa como

$$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma||\mathbf{x}_i - \mathbf{x}_j||_2^2}$$

Otros Kernels

- Lineal

$$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) =< \mathbf{x}_i, \mathbf{x}_j >= \mathbf{x}_i^T \cdot \mathbf{x}_j$$

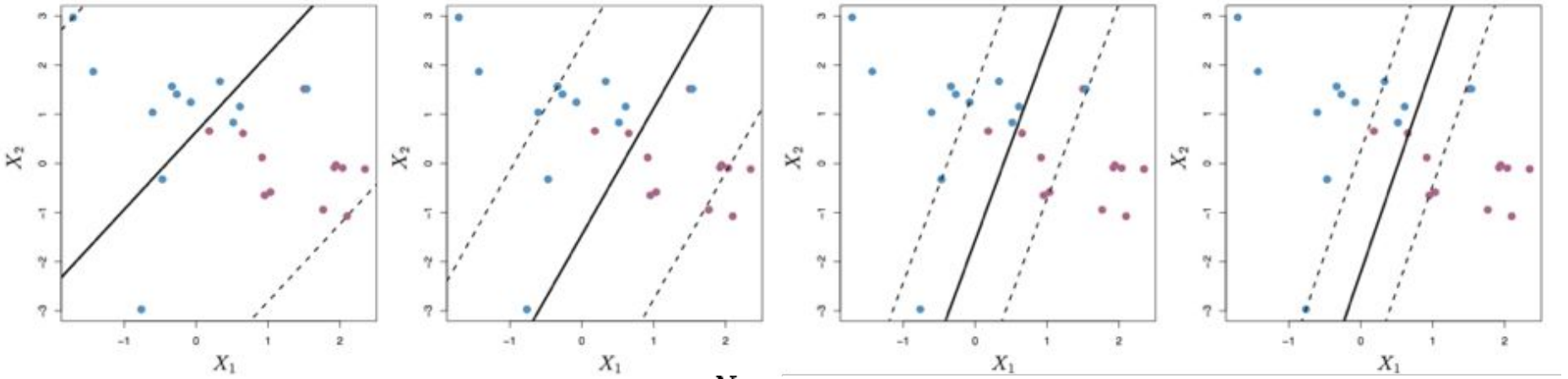
- Polinómico

$$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = (\gamma < \mathbf{x}_i, \mathbf{x}_j > +r)^d$$

Índice

1. K-nn en regresión
2. **Kernels y SVM**
3. Otros algoritmos con Kernels

SVCs (recap)



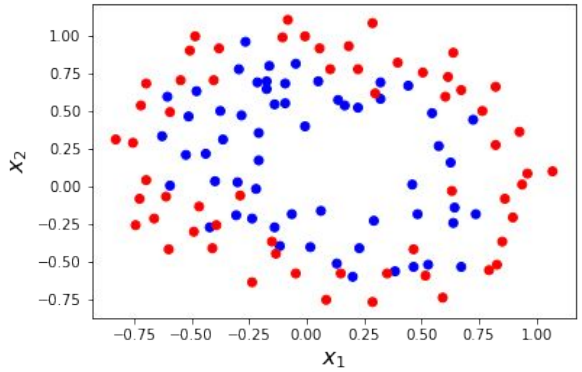
- Problemas lineales

$$\hat{\boldsymbol{\omega}} = \sum_{i=1}^{N} \alpha_i y^{(i)} \mathbf{x}^{(i)}$$

$$\hat{y} = f(\mathbf{x}) = \text{sign}\left(\hat{\boldsymbol{\omega}}^T \mathbf{x} + \hat{\omega}_0\right) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y^{(i)} \mathbf{x}^T \mathbf{x}^{(i)} + \hat{\omega}_0\right)$$

SVCs: fronteras no lineales

- La formulación de las SVMs y LR es similar
- Si queremos definir fronteras de separación no lineal en LR, ¿qué teníamos que hacer?
  - Ejemplo 2



$$x_1, x_2 \longrightarrow x_1, x_2, x_1^2, x_2^2, x_1 x_2$$

$D = 2$ $\qquad$ $D = 5$

Aumentamos la dimensionalidad del espacio de características

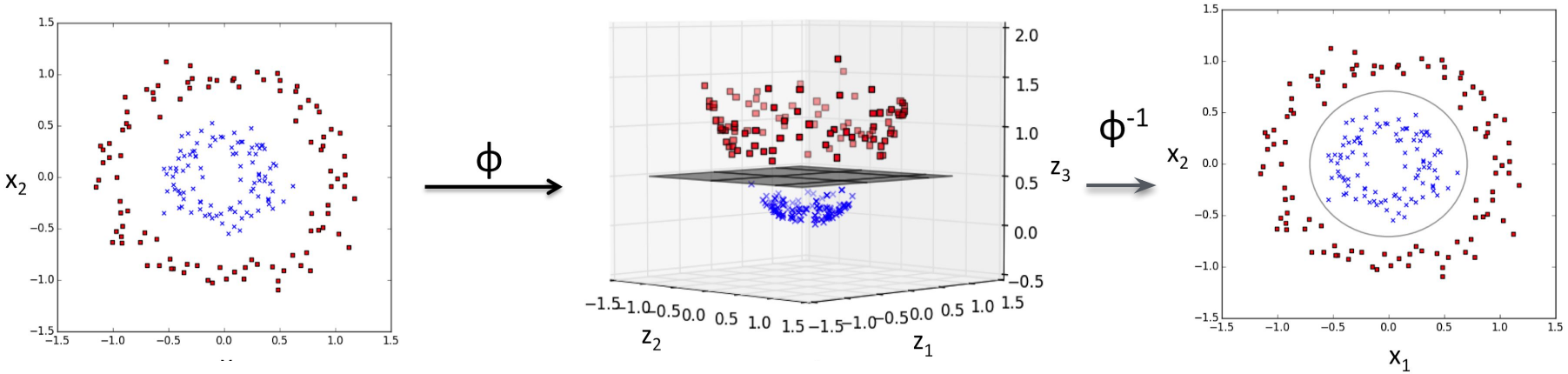
SVCs: fronteras no lineales

- En LR tenemos, bajo nuestro mejor criterio, que elegir la función de transformación

$$\mathbf{x}_1 \Rightarrow \phi(\mathbf{x}_1) \Rightarrow \mathbf{x}_1^2$$
$$\mathbf{x}_2 \Rightarrow \phi(\mathbf{x}_2) \Rightarrow \mathbf{x}_2^2$$

SVCs: fronteras no lineales

- ¿Qué buscamos con esta transformación?



SVCs: fronteras no lineales

- La formulación SVM permite no tener que conocer la transformación $\phi(\mathbf{x})$
  - **Truco del Núcleo**

Si tengo $< \mathbf{x}^{(i)}, \mathbf{x}^{(j)} >$ y quiero aplicar $< \phi(\mathbf{x}^{(i)}), \phi(\mathbf{x}^{(j)}) >$ no necesito conocer $\phi(\mathbf{x})$, **¡aplico un Kernel!**

Allí donde tenga $< \mathbf{x}^{(i)}, \mathbf{x}^{(j)} >$, utilizaré un kernel como

$$\mathcal{K}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = e^{-\gamma ||\mathbf{x}^{(i)} - \mathbf{x}^{(j)}||_2^2}$$

Formulación

- Así, podemos transformar la frontera lineal

$$\hat{y} = f(\mathbf{x}) = \text{sign}\left(\hat{\boldsymbol{\omega}}^T \mathbf{x} + \hat{\omega}_0\right) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y^{(i)} \mathbf{x}^T \mathbf{x}^{(i)} + \hat{\omega}_0\right)$$

en una no lineal

$$\hat{y} = f(\mathbf{x}) = \text{sign}\left(\hat{\boldsymbol{\omega}}^T \mathbf{x} + \hat{\omega}_0\right) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y^{(i)} \mathcal{K}(\mathbf{x}, \mathbf{x}^{(i)}) + \hat{\omega}_0\right)$$

Ejemplo

- Supongamos la transformación

$$\phi(\mathbf{x}^{(i)}) = \left[\left(x_1^{(i)}\right)^2, \left(x_2^{(i)}\right)^2, \sqrt{2}x_1^{(i)}x_2^{(i)}\right]$$

con $\mathbf{x}^{(i)} = \left[x_1^{(i)}, x_2^{(i)}\right]$ Vamos a demostrar que

$$< \phi(\mathbf{x}^{(i)}), \phi(\mathbf{x}^{(j)}) >= \mathcal{K}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \left(< \mathbf{x}^{(i)}, \mathbf{x}^{(j)} >\right)^2$$

Ejemplo

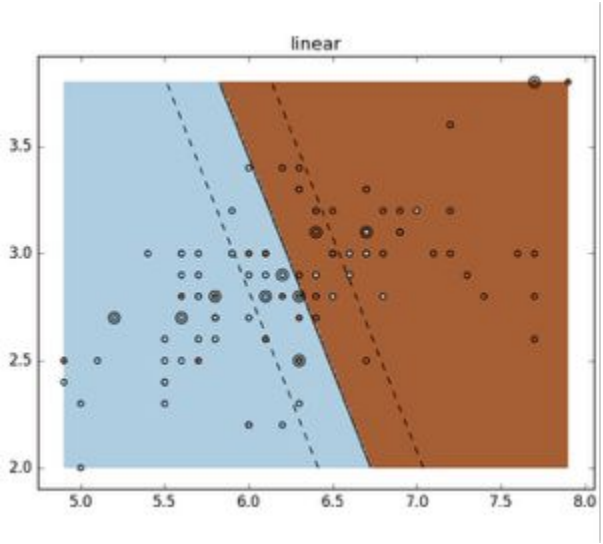
$$\mathbf{x}^{(i)} = \left[x_1^{(i)}, x_2^{(i)}\right]$$

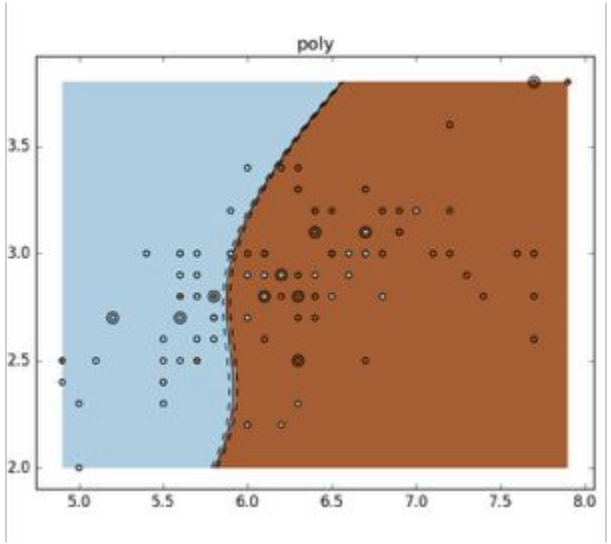
$$\phi(\mathbf{x}^{(i)}) = \left[\left(x_1^{(i)}\right)^2, \left(x_2^{(i)}\right)^2, \sqrt{2}x_1^{(i)}x_2^{(i)}\right]$$

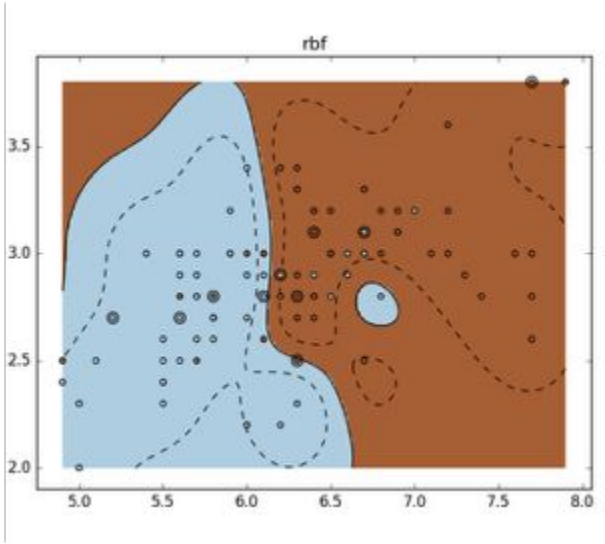
$$\phi(\mathbf{x}^{(j)}) = \left[\left(x_1^{(j)}\right)^2, \left(x_2^{(j)}\right)^2, \sqrt{2}x_1^{(j)}x_2^{(j)}\right]$$

$$\begin{aligned} < \phi(\mathbf{x}^{(i)}), \phi(\mathbf{x}^{(j)}) > &= \left(x_1^{(i)}\right)^2 \left(x_1^{(j)}\right)^2 + \left(x_2^{(i)}\right)^2 \left(x_2^{(j)}\right)^2 + 2x_1^{(i)}x_2^{(i)}x_1^{(j)}x_2^{(j)} \\ &= \left(x_1^{(i)}x_1^{(j)}\right)^2 + \left(x_2^{(i)}x_2^{(j)}\right)^2 + 2x_1^{(i)}x_2^{(i)}x_1^{(j)}x_2^{(j)} \\ &= \left(x_1^{(i)}x_1^{(j)} + x_2^{(i)}x_2^{(j)}\right)^2 = \left(< \mathbf{x}^{(i)}, \mathbf{x}^{(j)} >\right)^2 = \mathcal{K}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \end{aligned}$$

Resultado







- Hay que fijar los parámetros libres del Kernel
  - Hiperparámetro adicional
  - CV

Índice

1. K-nn en regresión
2. **Kernels y SVM**
3. Otros algoritmos con Kernels

Métodos Kernel

- Allí donde tengamos un producto escalar de la forma $< \mathbf{x}^{(i)}, \mathbf{x}^{(j)} >$

o en forma matricial una expresión como $\mathbf{X}\mathbf{X}^T$

- Entonces, podemos **aplicar un Kernel**
  - Ridge Regression
    - Muy similar a SVR
  - PCA
    - Extensión no lineal de PCA

Kernel Ridge Regression

- La solución, en forma matricial del algoritmo Ridge es

$$\hat{\boldsymbol{\omega}} = \left(\mathbf{X}^T\mathbf{X} + \alpha\mathbf{I}_D\right)^{-1}\mathbf{X}^T\mathbf{y}$$

la cual aplicando el *matrix inversion lemma*, puede expresarse como

$$\hat{\boldsymbol{\omega}} = \mathbf{X}^T\left(\mathbf{X}\mathbf{X}^T + \alpha\mathbf{I}_N\right)^{-1}\mathbf{y}$$

- Lo que permite aplicar la extensión Kernel

$$\hat{\boldsymbol{\omega}} = \mathbf{X}^T\left(\mathcal{K} + \alpha\mathbf{I}_N\right)^{-1}\mathbf{y}$$

Kernel Ridge Regression

- Misma solución que SVR

$$\hat{\omega} = \mathbf{X}^T \boldsymbol{\beta}$$

donde $\boldsymbol{\beta} = (\mathcal{K} + \alpha \mathbf{I}_N)^{-1} \mathbf{y}$

Salvo que la función de coste es distinta (error cuadrático medio)

Conclusiones sobre SVMs & Kernels

- Algoritmos muy potentes, grandes prestaciones
- El algoritmo no calcula la probabilidad, se estima a partir de heurística (no muy fiable)
- Computacionalmente intenso:
  - Si alta dimensionalidad (muchas variables), Kernel lineal
  - Valores de C elevados cuesta mucho entrenar
  - Cálculo del Kernel cuando el problema tiene muchas muestras
- RBF es capaz de aprender casi todo, kernel universal.
- Kernels usan medidas de distancia/similitud: ¡¡ ESCALADO !!

Referencias

- Felipe Alonso-Atienza, Tesis Doctoral (uc3m)
- Machine Learning, a probabilistic perspective
  - Capítulo 14
- Support Vector Machines, Chris Williams, Universidad de Edimburgo

Hora de practicar