MAQUINAS DE VECTORES DE SOPORTE II

Margen Suave y Kernels

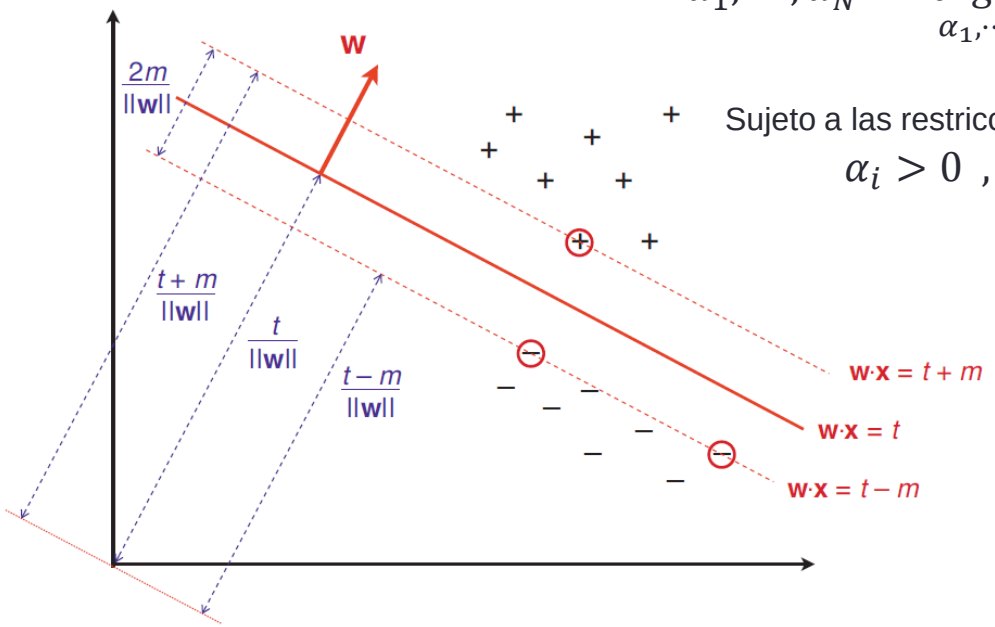
Dr. Jorge Hermosillo Valadez (jhermosillo@uaem.mx)
Depto. de Computación
CInC – IICBA, UAEM

Agosto - 2017

Resumen clase anterior

- Perceptron DUAL
  - $\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$
  - $y_i = \text{sign}\left(\sum_j \alpha y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle\right)$
  - única información necesaria: $\mathbf{G} = \mathbf{X}\mathbf{X}^{\mathrm{T}}$ (*matriz Gram*) que contiene todos los productos punto $\langle \mathbf{x}_i, \mathbf{x}_j \rangle, 1 \leq j \leq N, 1 \leq i \leq N$.
- Maquina de vectores de soporte (SVM)

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

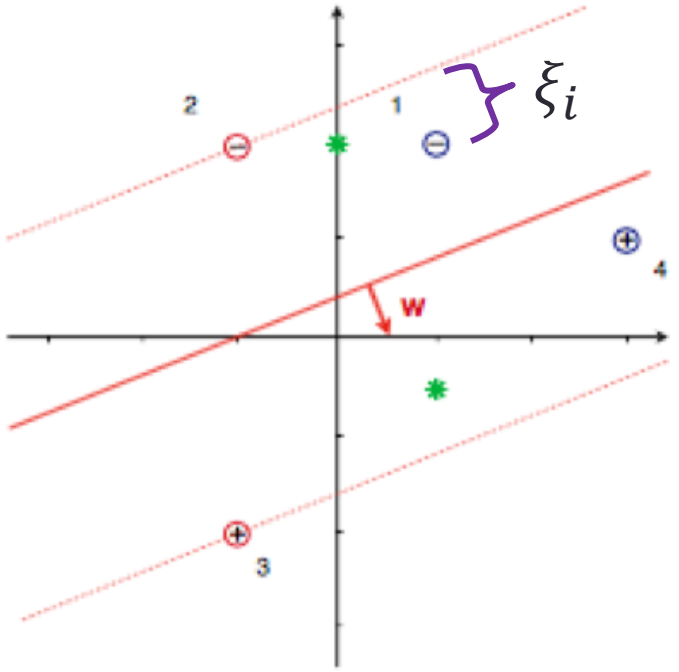


Sujeto a las restricciones:

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

SVM con 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 con margen suave

- El problema de optimización se vuelve:

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

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

- 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 con 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:

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

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

Significado de *C* como cota superior para α_i

- En el caso óptimo, para cada ejemplo (dato de entrada) se debe cumplir

$$C - \alpha_i - \beta_i = 0$$

- Distinguimos tres casos para los ejemplos:
  1. $\alpha_i = 0$ significa que están fuera o sobre el margen.
  2. $0 < \alpha_i < C$ estos son los vectores de soporte sobre el margen.
  3. $\alpha_i = C \Longrightarrow \beta_i = 0$ estos están sobre o dentro del margen.

- Como todavía tenemos: $\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$ los últimos dos casos contribuyen a expandir el margen.

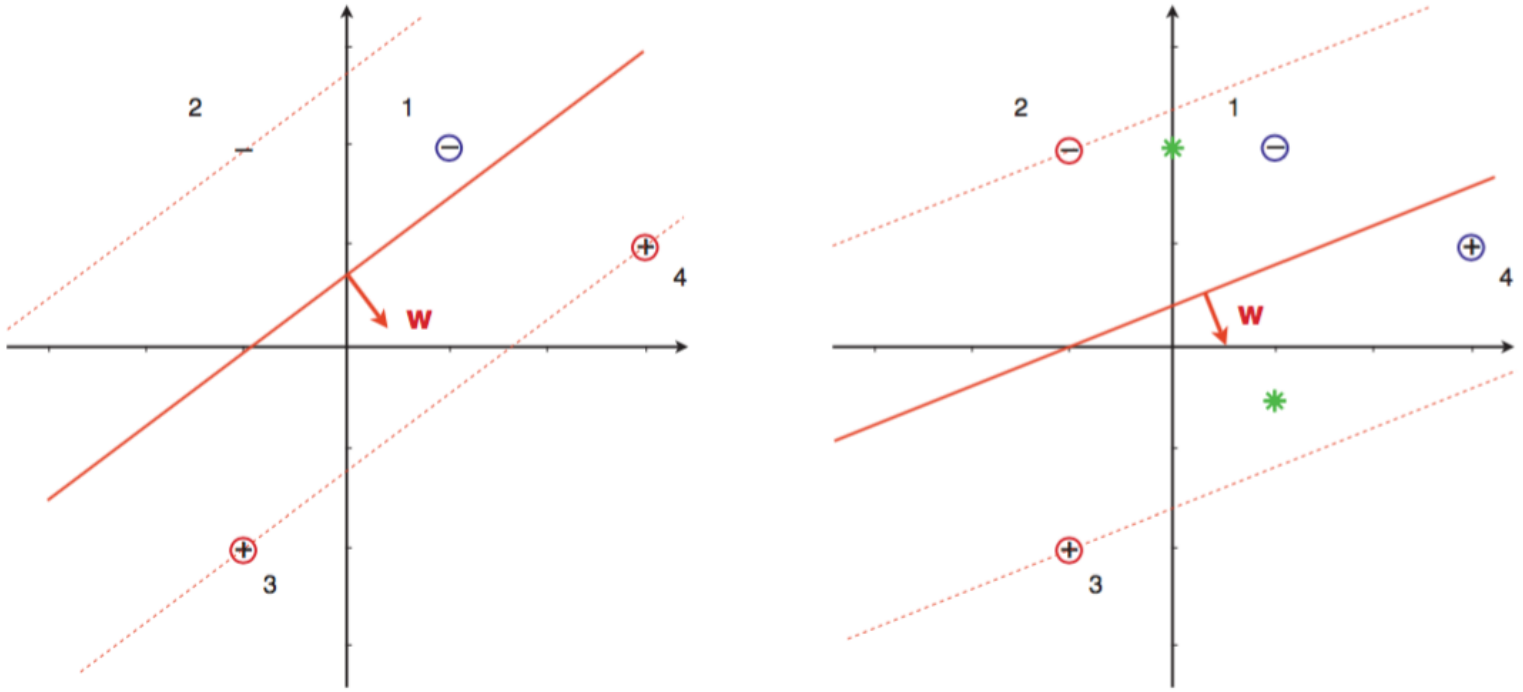


Figure 7.9. **(left)** The soft margin classifier learned with $C = 5/16$, at which point $\mathbf{x}_2$ is about to become a support vector. **(right)** The soft margin classifier learned with $C = 1/10$: all examples contribute equally to the weight vector. The asterisks denote the class means, and the decision boundary is parallel to the one learned by the basic linear classifier.

Clasificador lineal “básico”

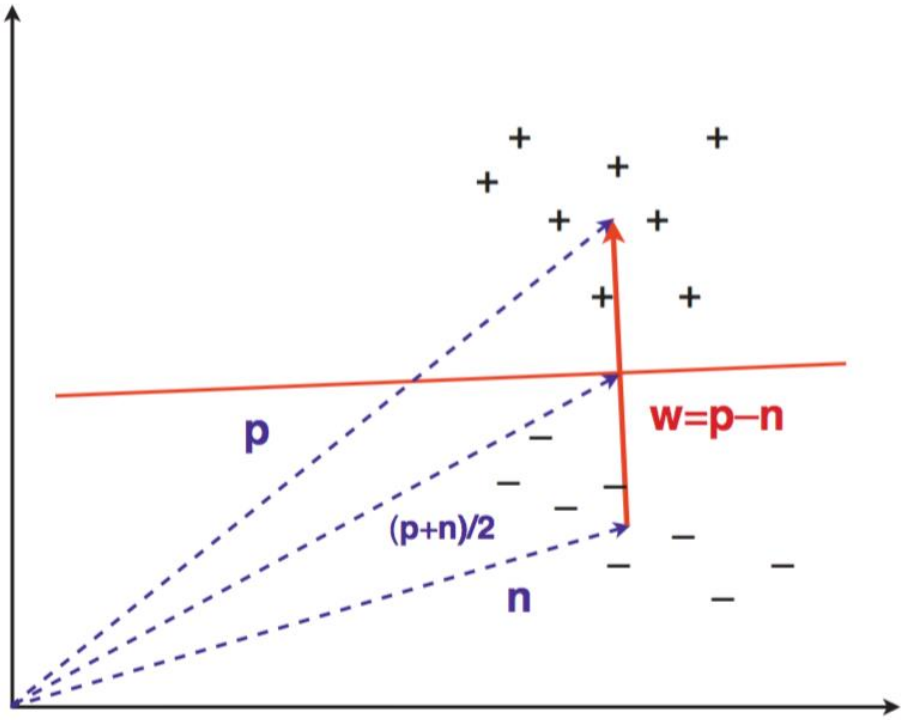


Figure 1.1. The basic linear classifier constructs a decision boundary by half-way intersecting the line between the positive and negative centres of mass. It is described by the equation $\mathbf{w} \cdot \mathbf{x} = t$, with $\mathbf{w} = \mathbf{p} - \mathbf{n}$; the decision threshold can be found by noting that $(\mathbf{p} + \mathbf{n})/2$ is on the decision boundary, and hence $t = (\mathbf{p} - \mathbf{n}) \cdot (\mathbf{p} + \mathbf{n})/2 = (||\mathbf{p}||^2 - ||\mathbf{n}||^2)/2$, where $||\mathbf{x}||$ denotes the length of vector $\mathbf{x}$.

Mas allá de la clasificación lineal: métodos de Kernel

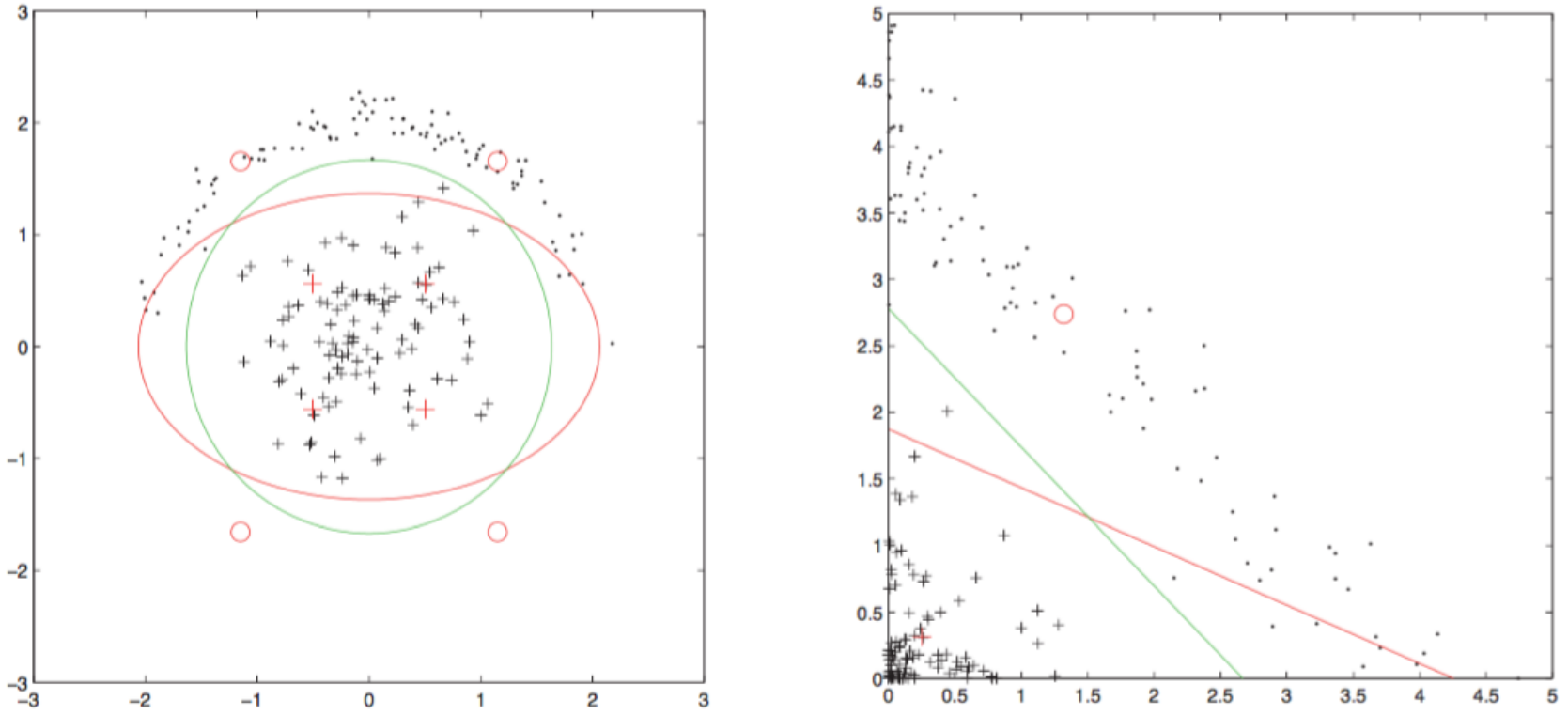


Figure 7.14. (left) Decision boundaries learned by the basic linear classifier and the perceptron using the square of the features. **(right)** Data and decision boundaries in the transformed feature space.

Términos empleados

- Kernel: en términos generales un Kernel es un factor multiplicativo en una sumatoria o integral.
- En nuestro contexto un Kernel es una transformación de los datos que debe cumplir con ciertas propiedades específicas.
- El espacio de nuevos datos (transformados) se le llama comúnmente espacio de atributos ($feature\ space$), en contraste con el espacio original o espacio de entrada ($input\ space$).

Perceptrón DUAL: un algoritmo de conteo

$\boldsymbol{\alpha} \coloneqq (\alpha_1, \alpha_2, \cdots, \alpha_N) = 0$

$converge = \boldsymbol{Falso}$

mientras $converge == \boldsymbol{Falso}$:

$converge = \boldsymbol{Verdadero}$

para i **en** $|\mathbf{X}|$:

si $y_i \sum_{j=1}^{|\mathbf{X}|} \alpha_j y_j \langle \mathbf{x_i}, \mathbf{x_j} \rangle \leq 0$ **entonces**: #xi mal clasificado

$\alpha_i = \alpha_i + 1$

$converge = \boldsymbol{Falso}$

fin

fin

fin

Producto punto en el espacio cuadrático

$$\mathbf{x}_i = (x_i, y_i)\ ;\ \mathbf{x}_j = \left(x_j, y_j\right) \ \rightarrow \langle \mathbf{x}_i, \mathbf{x}_j \rangle = x_i x_j + y_i y_j$$

- Instancias en el espacio de atributos al cuadrado:

$$\mathbf{x}_i^2 = \left(x_i^2, y_i^2\right);\ \mathbf{x}_j^2 = (x_j^2, y_j^2)$$

- Producto punto de estas instancias:

$$\langle \mathbf{x}_i^2, \mathbf{x}_j^2 \rangle = x_i^2 x_j^2 + y_i^2 y_j^2$$

- Casi igual a:

$$\langle \mathbf{x}_i, \mathbf{x}_j \rangle^2 = x_i^2 x_j^2 + y_i^2 y_j^2 + 2 x_i x_j y_i y_j$$

Kernel cuadrático como un producto punto

- Para que sean iguales, podemos transformar los datos agregando un tercer atributo: $\sqrt{2}xy$

$$\phi(\mathbf{x}_i) = \left(x_i^2, y_i^2, \sqrt{2}x_i y_i\right) \; ; \; \phi(\mathbf{x}_j) = \left(x_j^2, y_j^2, \sqrt{2}x_j y_j\right)$$

- Calculando ahora el producto punto:

$$\left\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \right\rangle = x_i^2 x_j^2 + y_i^2 y_j^2 + 2x_i x_j y_i y_j = \left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle^2$$

- Definimos ahora:

$$\kappa\left(\mathbf{x}_i, \mathbf{x}_j\right) = \left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle^2$$

- Sustituimos $\kappa\left(\mathbf{x}_i, \mathbf{x}_j\right)$ en lugar de $\left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle$ en el algoritmo del perceptron.

Perceptrón dual con kernel

Entrada: datos etiquetados en coordenadas homogeneas

funcion de kernel $\kappa(\mathrm{x}_i, \mathrm{x}_j)$

Salida: coeficientes α_i que definen una frontera no-lineal

$\boldsymbol{\alpha} \coloneqq (\alpha_1, \alpha_2, \cdots, \alpha_N) = 0$

$converge = \boldsymbol{Falso}$

mientras $converge == \boldsymbol{Falso}$:

- $converge = \boldsymbol{Verdadero}$
- **para** i **en** $|\mathbf{X}|$: #|X|=N
  - **si** $y_\mathrm{i} \sum_{j=1}^{N} \alpha_j y_j \kappa(\mathbf{x_i}, \mathbf{x_j}) \leq 0$ **entonces**: #xi mal clasificado
    - $\alpha_i = \alpha_i + 1$
    - $converge = \boldsymbol{Falso}$
  - **fin**
- **fin**

fin

Kernel polinomial y gaussiano

- Con base en lo anterior podemos definir kernels de grado p > 2, por ejemplo:

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle^p$$

- En general podemos definir un kernel polinomial:

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\langle \mathbf{x}_i, \mathbf{x}_j \rangle + c)^p, c \geq 0$$

- No cualquier función puede ser utilizada como kernel
- Nota que: $\kappa(\mathbf{x}, \mathbf{x}) = \langle \phi(\mathbf{x}), \phi(\mathbf{x}) \rangle = \|\phi(\mathbf{x})\|^2$ para cualquier kernel que cumpla con ciertas propiedades referidas como transformaciones “semidefinidas positivas”.

Kernels básicos

- Aunque nuevos kernels aparecen en la literatura, los siguientes cuatro son básicos y ampliamente utilizados:

Lineal:	$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$
Polinomial:	$\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\langle \mathbf{x}_i, \mathbf{x}_j \rangle + r)^p, r \geq 0$
Gaussiano (*Radial Basis Function* – RBF):	$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right) = \exp\left(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2\right)$
Sigmoide:	$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma \langle \mathbf{x}_i, \mathbf{x}_j \rangle + r)$

donde r, p, γ son parámetros de los modelos.

SVM's con kernels

- El "truco" del kernel (*kernel trick*) es comúnmente empleado con las máquinas de vectores de soporte:

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

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

- No perder de vista:
  1. La frontera de decisión no puede representarse como un simple vector de pesos en el espacio de entrada.
  2. Para clasificar un nuevo dato $\mathbf{x}_i$ se necesita evaluar:

$$y_{\mathrm{i}} \sum_{j=1}^{N} \alpha_j y_j \kappa(\mathbf{x}_i, \mathbf{x}_j)$$

Procedimiento para clasificar con SVM's

- El siguiente procedimiento se propone en:
  - Hsu, C., Chang, C., & Lin, C. (2016). *A practical guide to support vector classification*. Department of Computer Science National Taiwan University, Taipei 106, Taiwan (https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf)

1. Transforma los datos en un formato conveniente para usar SVM's
2. Realiza un escalamiento sencillo de los datos
3. Considera el kernel RBF (gaussiano)
4. Utiliza validación cruzada (cross-validation) para elegir la mejor combinación de C y γ para entrenar el modelo con todos los datos
5. Prueba