Aprendizaje Automático
Segundo Cuatrimestre de 2018

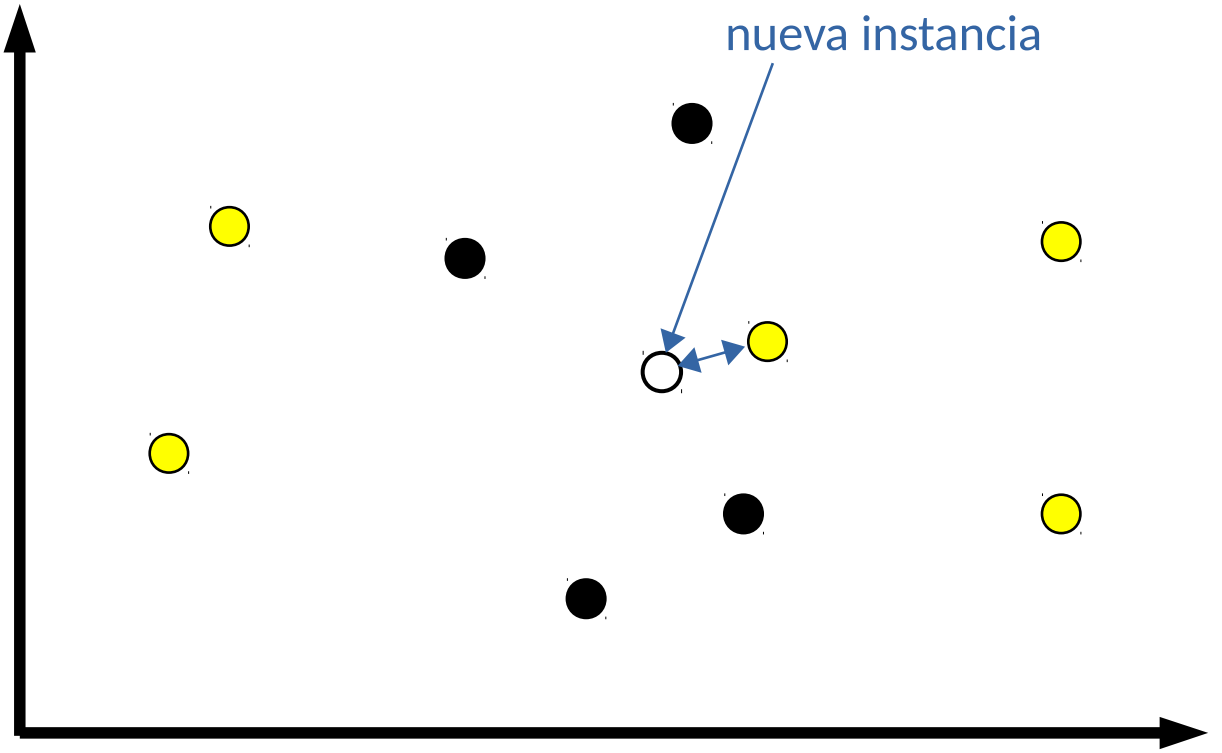
# Clasificadores

DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

# Aprendizaje Basado en Instancias

**Vecino Más Cercano:**

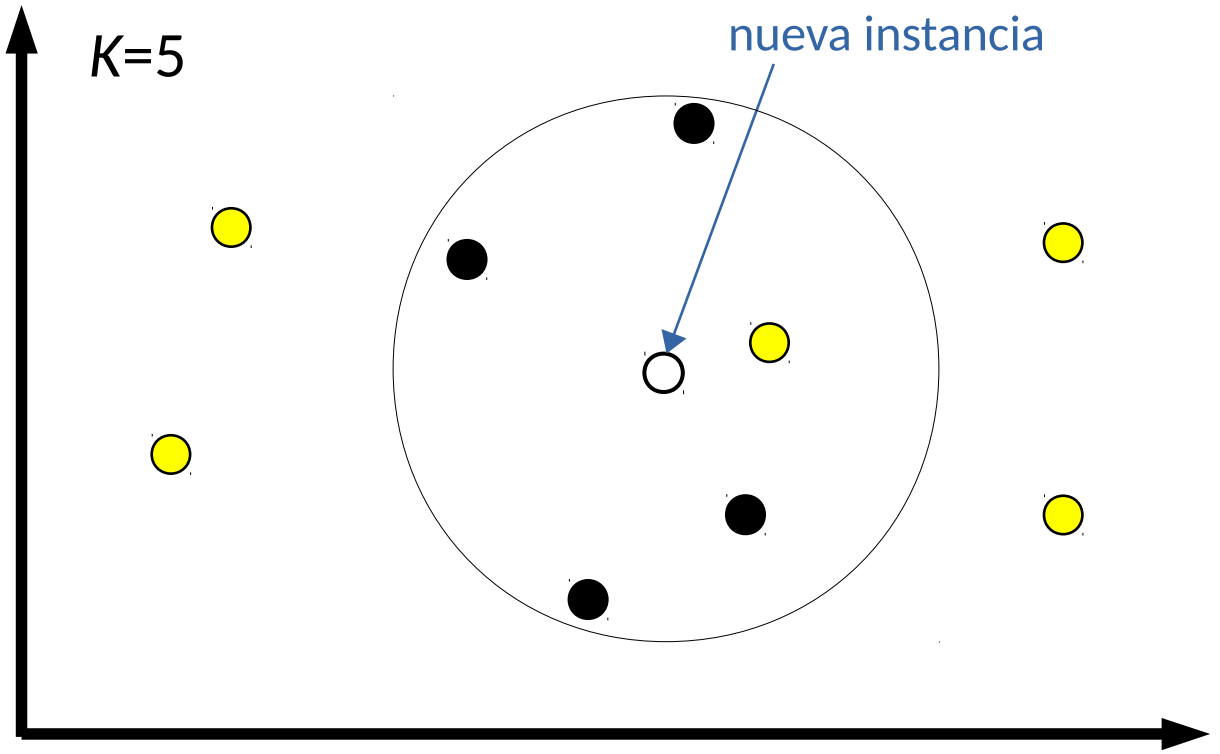
- Dada una nueva instancia, devolver la clase de la instancia más cercana en *D*.



# Aprendizaje Basado en Instancias

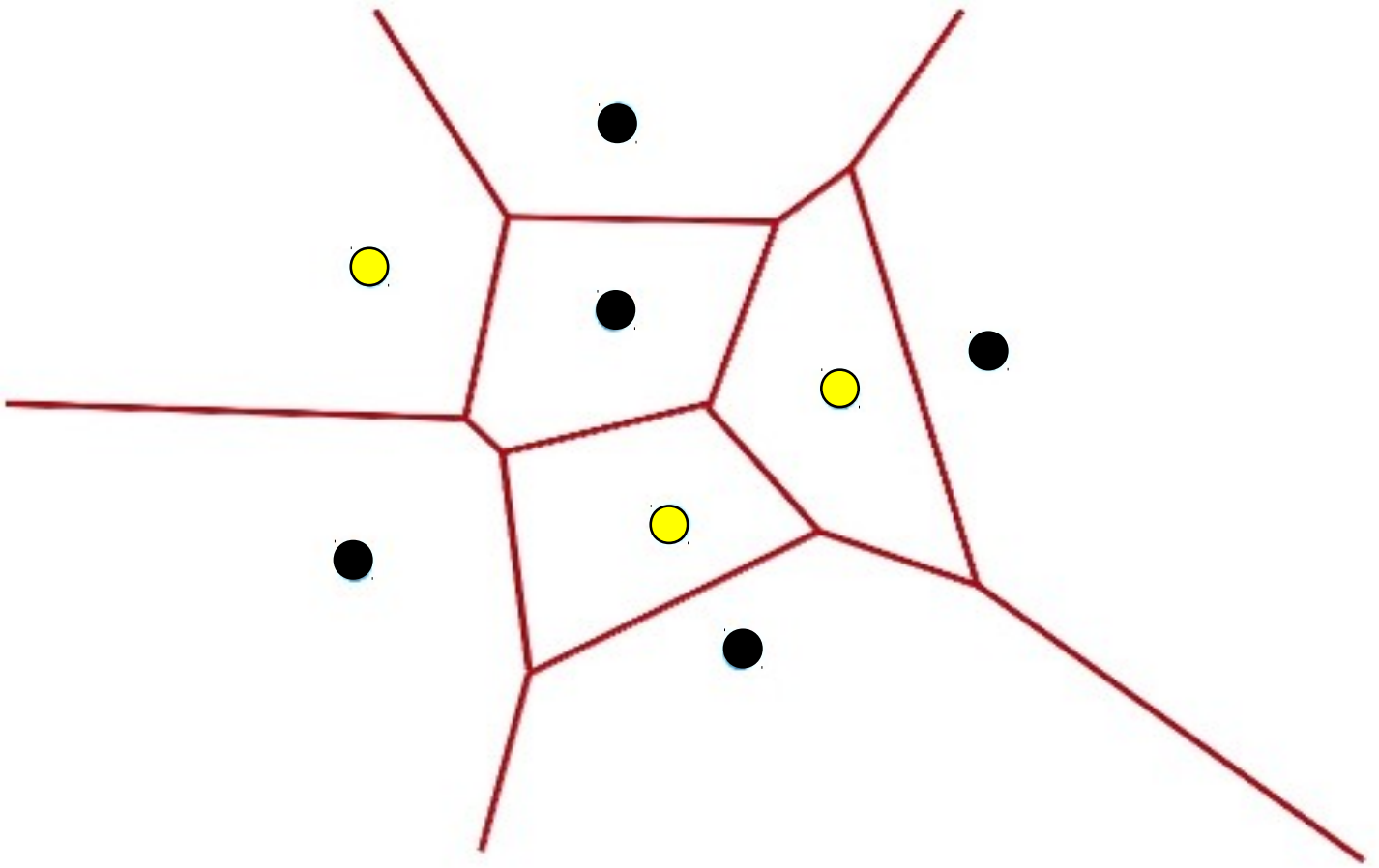
## *K* Vecinos Más Cercanos (KNN):

- Dada una nueva instancia, devolver la clase más frecuente entre las *K* instancias más cercanas en *D*.



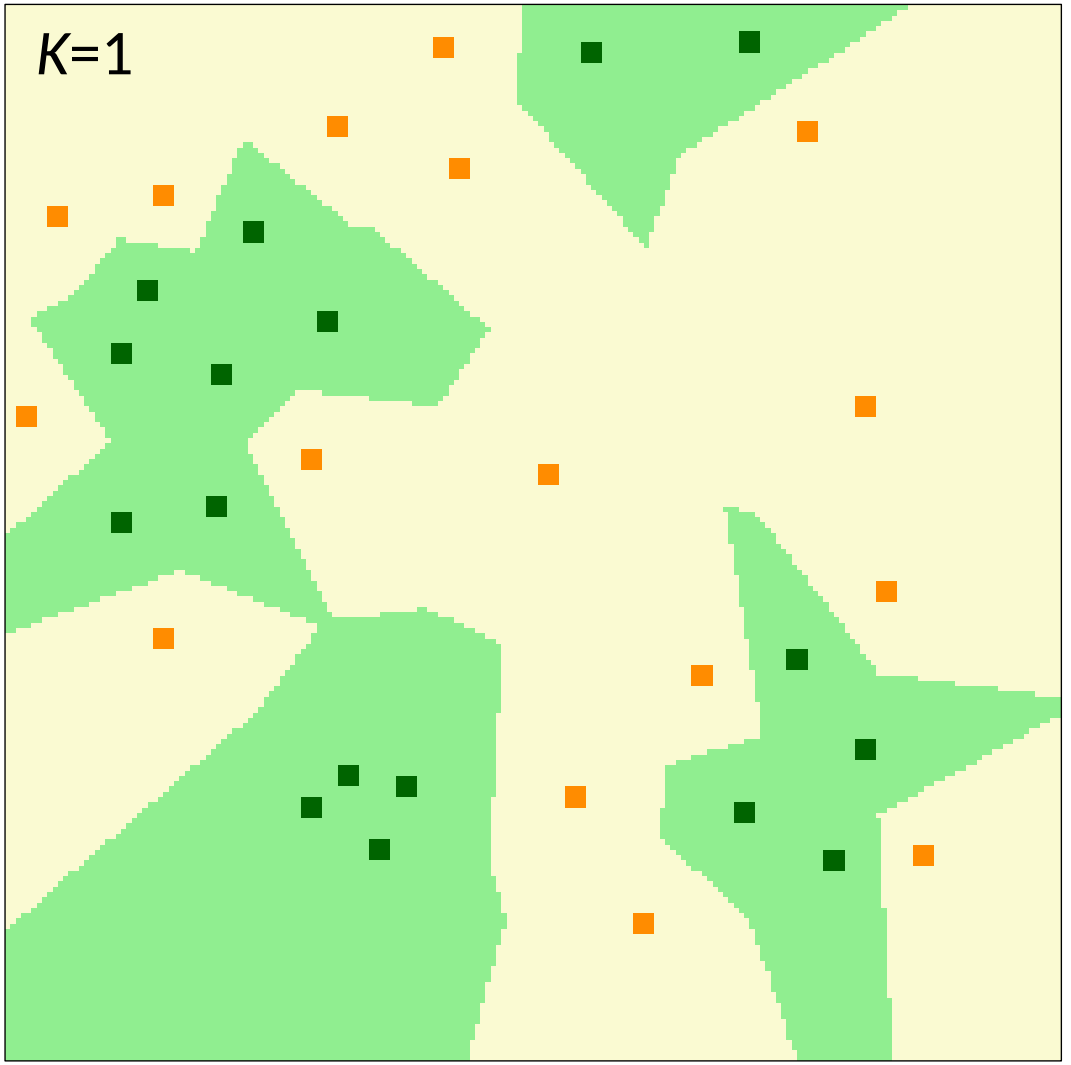
# ¿Qué significa un *K* más grande?

**Diagrama de Voronoi:**

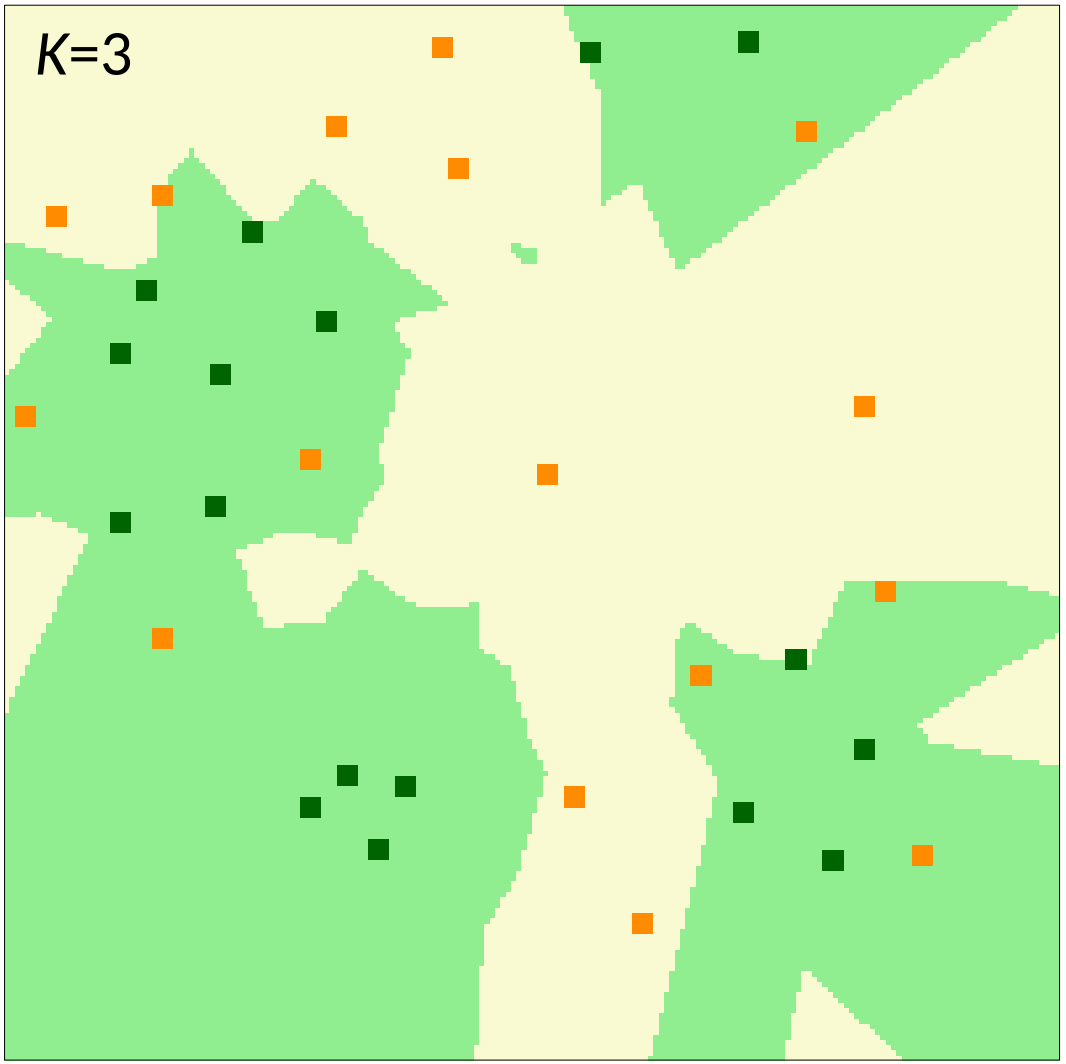


*K*=1

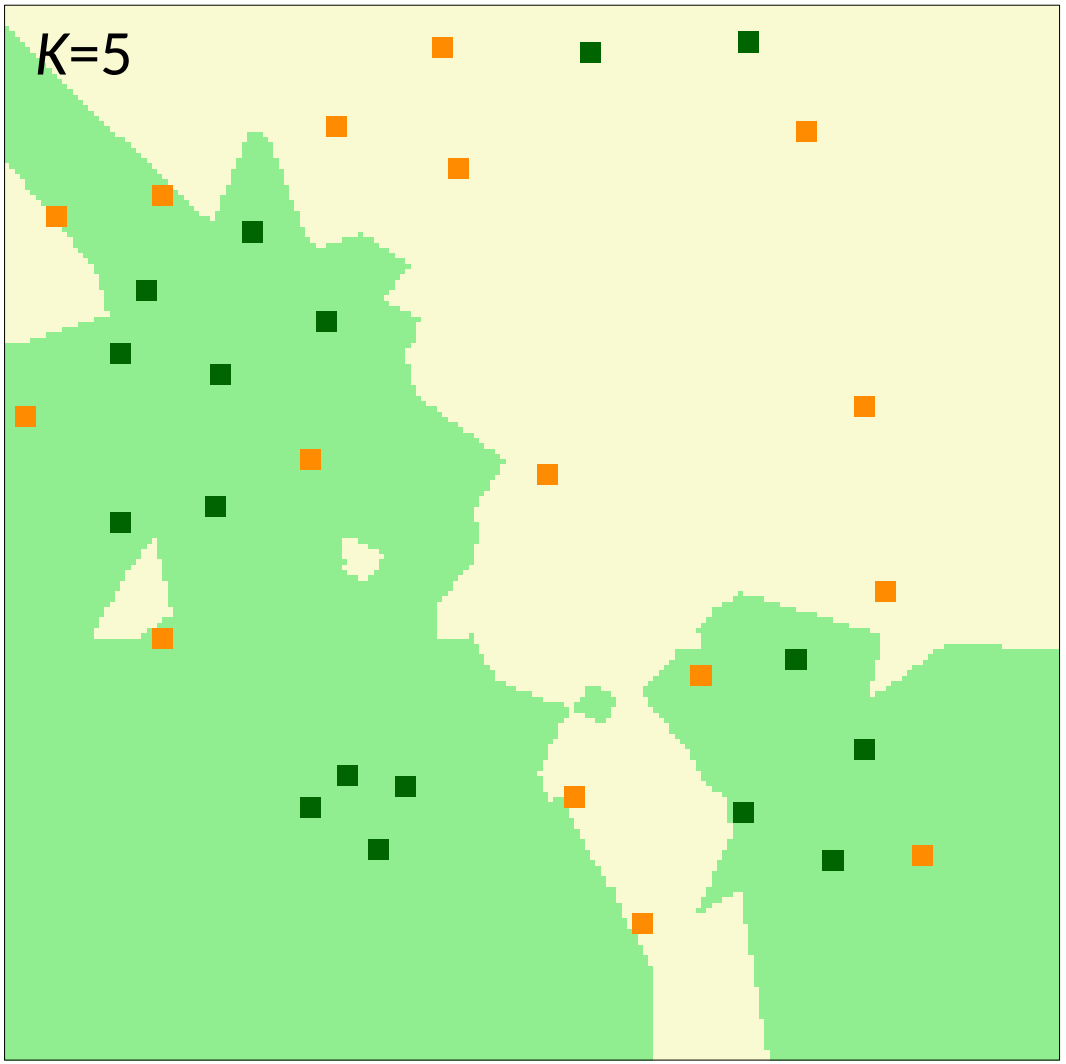
# ¿Qué significa un *K* más grande?



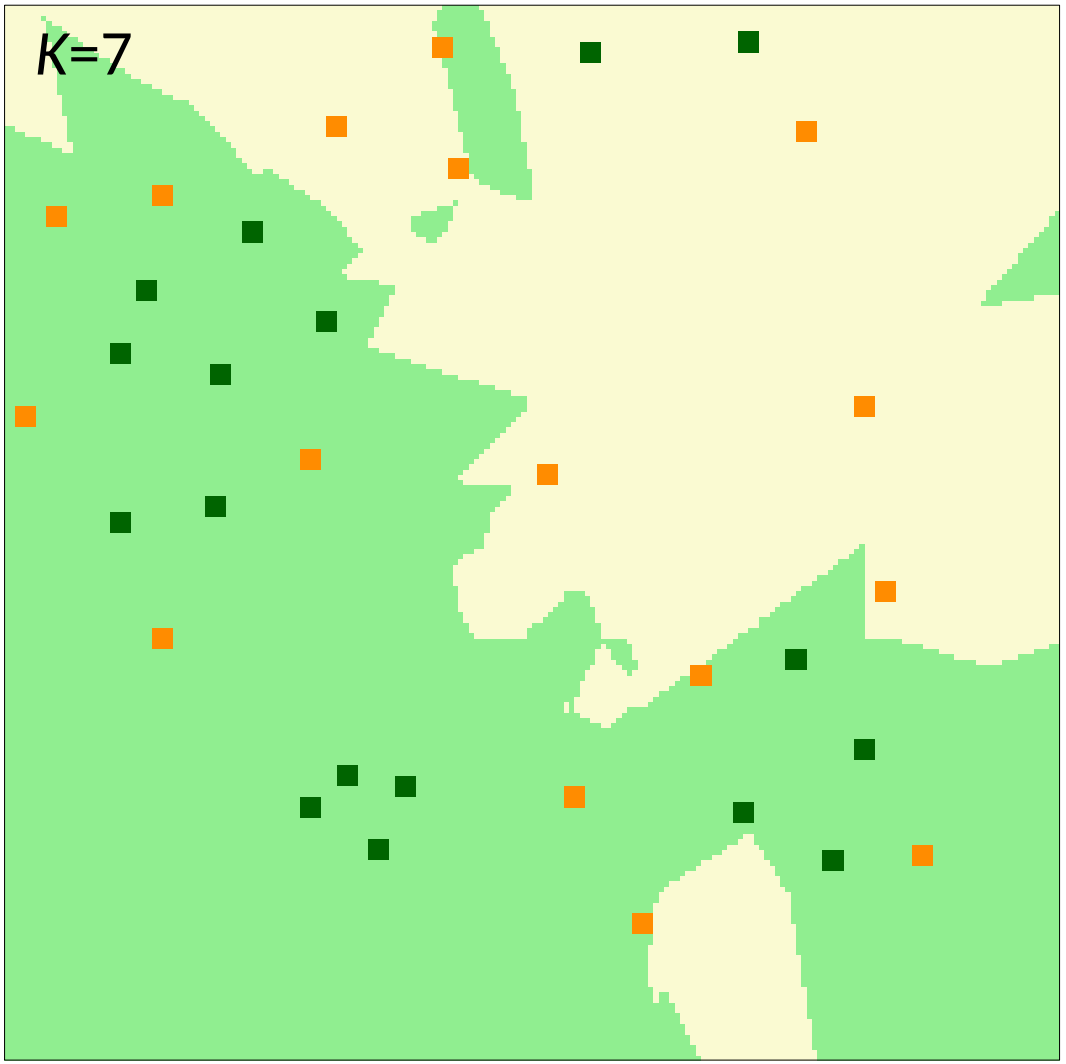
# ¿Qué significa un *K* más grande?



# ¿Qué significa un *K* más grande?



# ¿Qué significa un *K* más grande?




... ¿y si *K=n*?

# Distance-Weighted KNN

Podríamos querer que los vecinos más cercanos tengan más influencia en la votación…

Cada vecino $x^{(i)}$ aporta $w^{(i)}$ votos (y no 1 como en KNN), donde

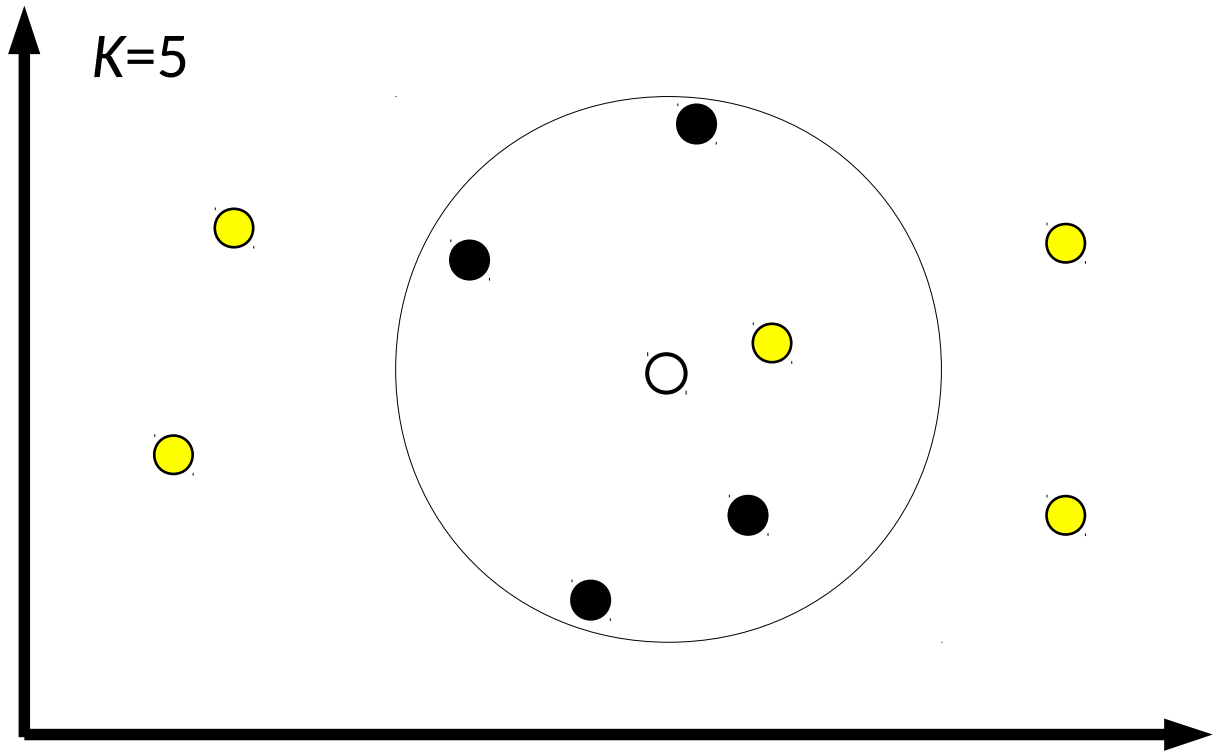
$$w^{(i)} = \frac{1}{d(x^{(q)}, x^{(i)})^2}$$

$x^{(q)}$ es la instancia a clasificar, y $d(\cdot,\cdot)$ es la distancia entre dos instancias.

Quizá podemos usar *todas* las instancias en *D*, en lugar de sólo las *K* más cercanas.

# KNN – Devolviendo Probabilidades

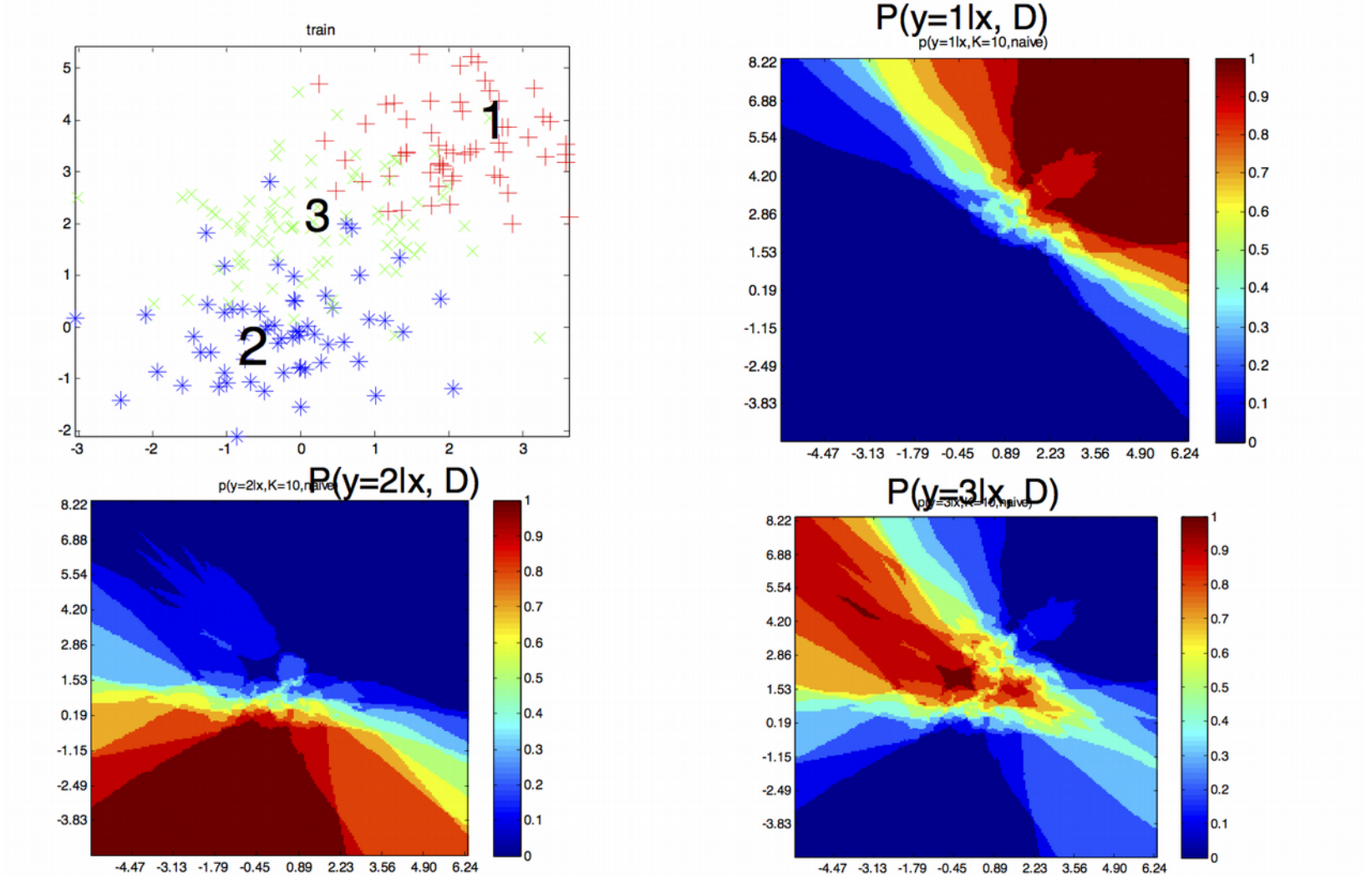
$$P(Y = y \,|\, X = x^{(q)}) \approx \sum_{x^{(i)} \in Vecinos(x^{(q)}, K, D)} I(y^{(i)} = y) \cdot \frac{1}{K}$$




Para la nueva instancia: *P(amarillo) = 1/5*
*P(negro) = 4/5*

# KNN – Devolviendo Probabilidades

$$P(Y = y \,|\, X = x^{(q)}) \approx \sum_{x^{(i)} \in Vecinos(x^{(q)}, K, D)} I(y^{(i)} = y) \cdot \frac{1}{K}$$



# KNN

- 👍 Técnica simple que a veces permite aproximar conceptos muy complejos.
- 👍 El entrenamiento (¿entrenamiento?) es muy rápido.
- 👎 La consulta es muy lenta. Requiere AyED eficientes.
- 👎 El modelo (¿modelo?) ocupa mucho espacio en disco.
- ? Para pensar: La distancia se calcula con todos los atributos. ¿Qué pasa si algunos son irrelevantes? ¿Qué pasa si están en escalas muy distintas?

# Bayes Classifier

- Dada una nueva instancia $x$, su clase más probable a posteriori $k$ puede expresarse así:

$$k_{\text{MAP}} = \underset{k}{\operatorname{argmax}} \; P(Y = k \mid X = x)$$

(Como vimos, KNN estima esta probabilidad en forma directa.)

Bayes

$$k_{\text{MAP}} = \underset{k}{\operatorname{argmax}} \; \frac{P(Y = k) \cdot P(X = x \mid Y = k)}{\cancel{P(X = x)}}$$

→ no depende de *k*

$$k_{\text{MAP}} = \underset{k}{\operatorname{argmax}} \; P(Y = k) \cdot P(X = x \mid Y = k)$$

*Bayes Classifier*

- Entonces, para clasificar una nueva instancia $x$, alcanza con calcular $P(Y = k)$ (*prior* de la clase *k*) y $P(X = x \mid Y = k)$ (distribución de instancias en la clase *k*), y listo…
- Problema: conocer $P(X = x \mid Y = k)$ es casi siempre imposible.
- Necesitamos estimar $P(Y = k)$ y $P(X = x \mid Y = k)$.

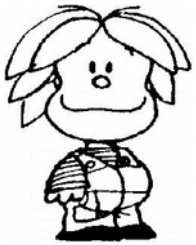
# Naive Bayes Classifier

$$k_{\text{MAP}} = \underset{k}{\operatorname{argmax}}\ P(Y = k) \cdot P(X = x \mid Y = k)$$

→ *Bayes Classifier*

$$k_{\text{MAP}} = \underset{k}{\operatorname{argmax}}\ P(Y = k) \cdot P(X_1 = x_1 \wedge X_2 = x_2 \ldots \wedge X_p = x_p | Y = k)$$

Suposición "naive": los *n* atributos son independientes.

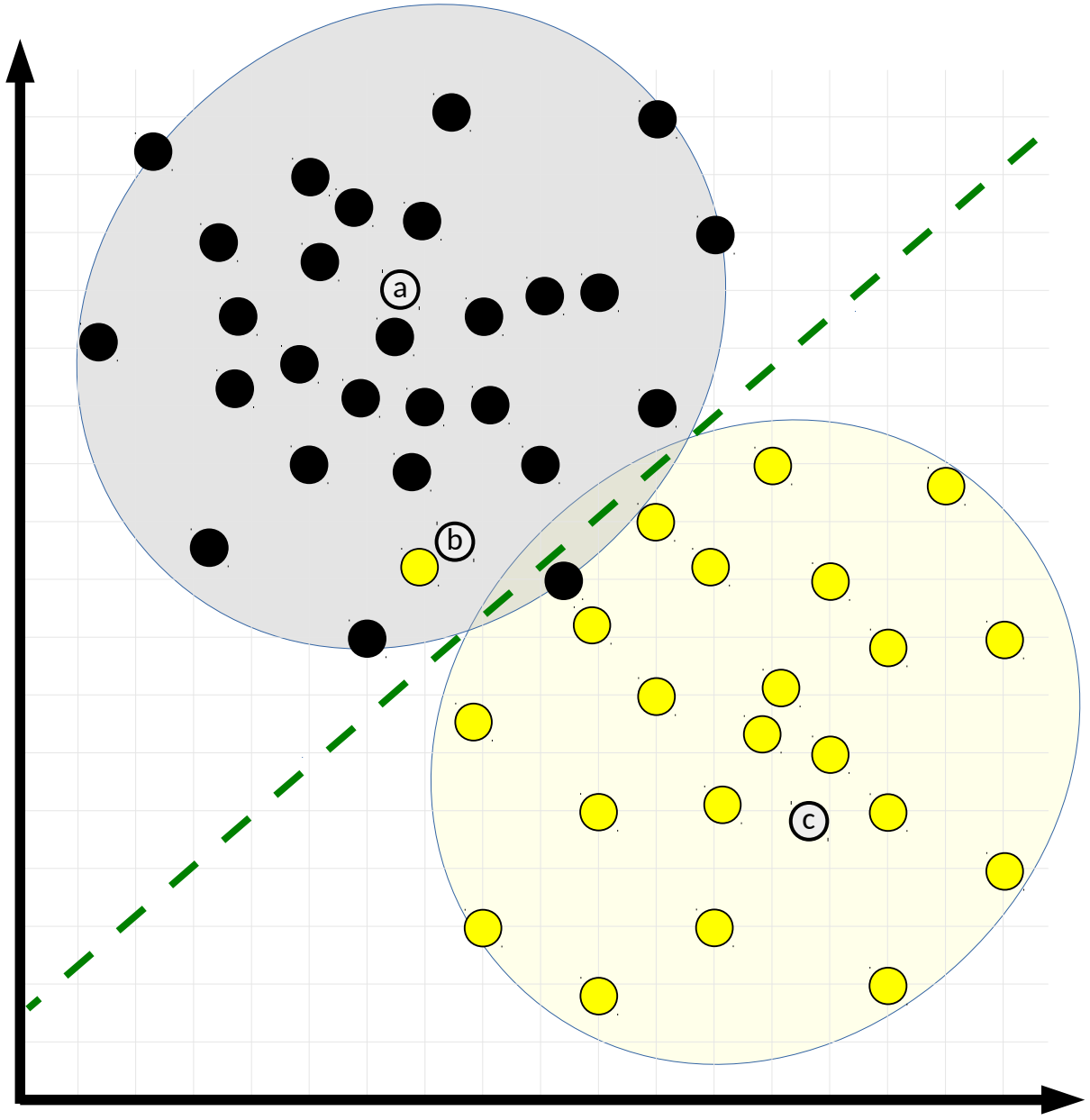


$$k_{\text{NB}} = \underset{k}{\operatorname{argmax}}\ P(Y = k) \cdot \prod_{i=1}^{p} P(X_i = x_i \mid Y = k)$$

→ *Naive Bayes Classifier*

- ¿Cómo podemos estimar $P(X_i = x_i \mid Y = k)$, con $X_i$ categórico?
- ¿Y con $X_i$ continuo?
- ¿Y cómo estimamos $P(Y = k)$?

# Linear Discriminant Analysis




¿De qué color serán a, b, c?

Idea de LDA:

1) modelar la distribución (normal) de puntos de cada clase;

2) asignar nuevas instancias a la clase que tiene probabilidad máxima a posteriori (MAP).

La recta verde es la frontera de decisión entre las 2 clases (en este caso, es una recta).

# Linear Discriminant Analysis

- La tarea de clasificación consiste en encontrar la clase *k* que maximice:

Bayes

$$P(Y=k|X=x) \overset{\text{Bayes}}{=} \frac{P(Y=k)P(X=x|Y=k)}{P(X=x)} = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)}$$

$P(Y=k) = \pi_k$ : prior de la clase $k$

$P(X=x|Y=k) = f_k(x)$ : función de densidad de $X$ para la clase $k$

- Suponemos que $f_k(x)$ tiene distribución normal $\mathrm{N}(\mu_k, \Sigma)$.
  Es decir, los puntos en cada clase fueron generados por una distribución con media particular, pero todos con la misma matriz de covarianza $\Sigma$.
- Con esta suposición, para modelar la distribución de instancias de cada clase, $P(X=x \mid Y=k)$, alcanza con estimar $\mu_k$, $\Sigma$, $\pi_k$ usando los datos de entrenamiento.
- Luego, encontrar la clase *k* con probabilidad máxima a posteriori sale directo.

# Linear Discriminant Analysis

Veamos el **caso *p*=1** (es decir, las instancias tienen un único atributo).

$$\operatorname*{argmax}_k \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)} = \operatorname*{argmax}_k \; \pi_k \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{(x-\mu_k)^2}{-2\sigma^2}\right)$$

$$f_k(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{(x-\mu_k)^2}{-2\sigma^2}\right), \text{ porque } f_k(x) \sim N(\mu_k, \sigma^2)$$

- Es fácil ver (tomar log de todo, operar y reordenar → ejercicio) que esto equivale a encontrar la clase *k* con máxima función discriminante $\delta_k(x)$:

$$\delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log \pi_k$$

- Los parámetros $\pi_k$, $\mu_k$, $\sigma^2$ pueden estimarse así:

$$\hat{\pi}_k = n_k/n$$ → proporción de instancias en la clase *k*

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i$$ → media de las instancias en la clase *k*

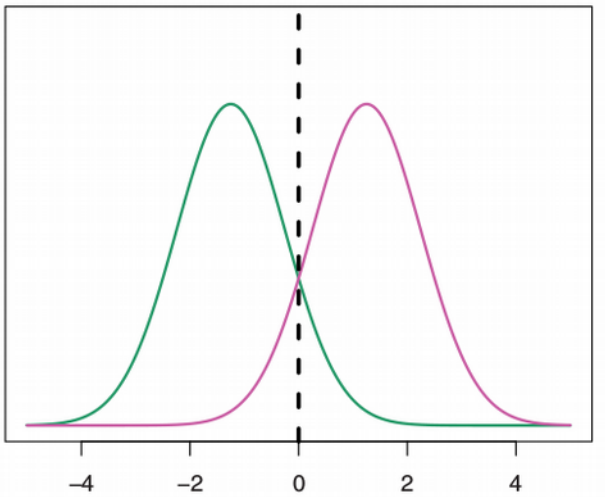
$$\hat{\sigma}^2 = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)^2$$ → promedio ponderado de las varianzas de las instancias en cada clase

# Linear Discriminant Analysis

$$\hat{\delta}_k(x) = x \cdot \frac{\hat{\mu}_k}{\hat{\sigma}^2} - \frac{\hat{\mu}_k^2}{2\hat{\sigma}^2} + \log \hat{\pi}_k$$

**Ejemplo simple:**

$p = 1$ (una variable)
$k = 2$ (dos clases)
$\pi_1 = \pi_2$ (clases equiprobables)




Fuente:
ISLR, p.140

Para cada nueva instancia *x*, le asignamos la clase con máxima función discriminante.

La frontera de decisión puede calcularse con $\delta_1(x) = \delta_2(x)$ y así llegar (ejercicio) a:

$$x = \frac{\hat{\mu}_1 + \hat{\mu}_2}{2}$$

# Linear Discriminant Analysis

**Caso general (*p*>1):**

Ahora *x*, $\mu_k$ son vectores de *p* dimensiones; $\Sigma$ es una matriz de *p*x*p*.

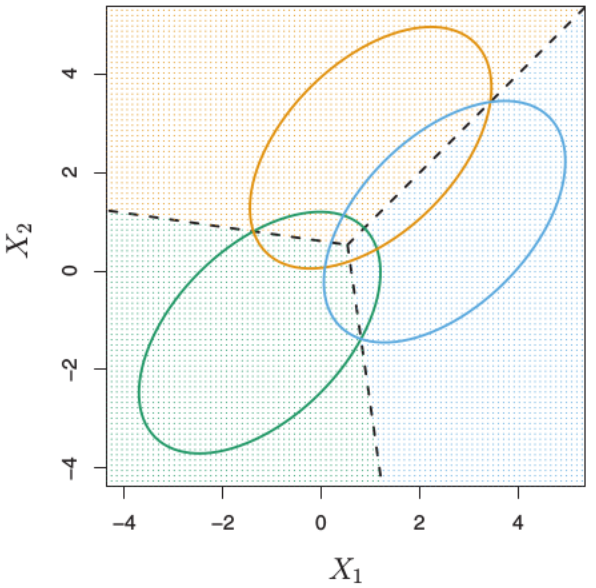
Con el mismo razonamiento, llegamos a estas funciones discriminantes:

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log \pi_k$$

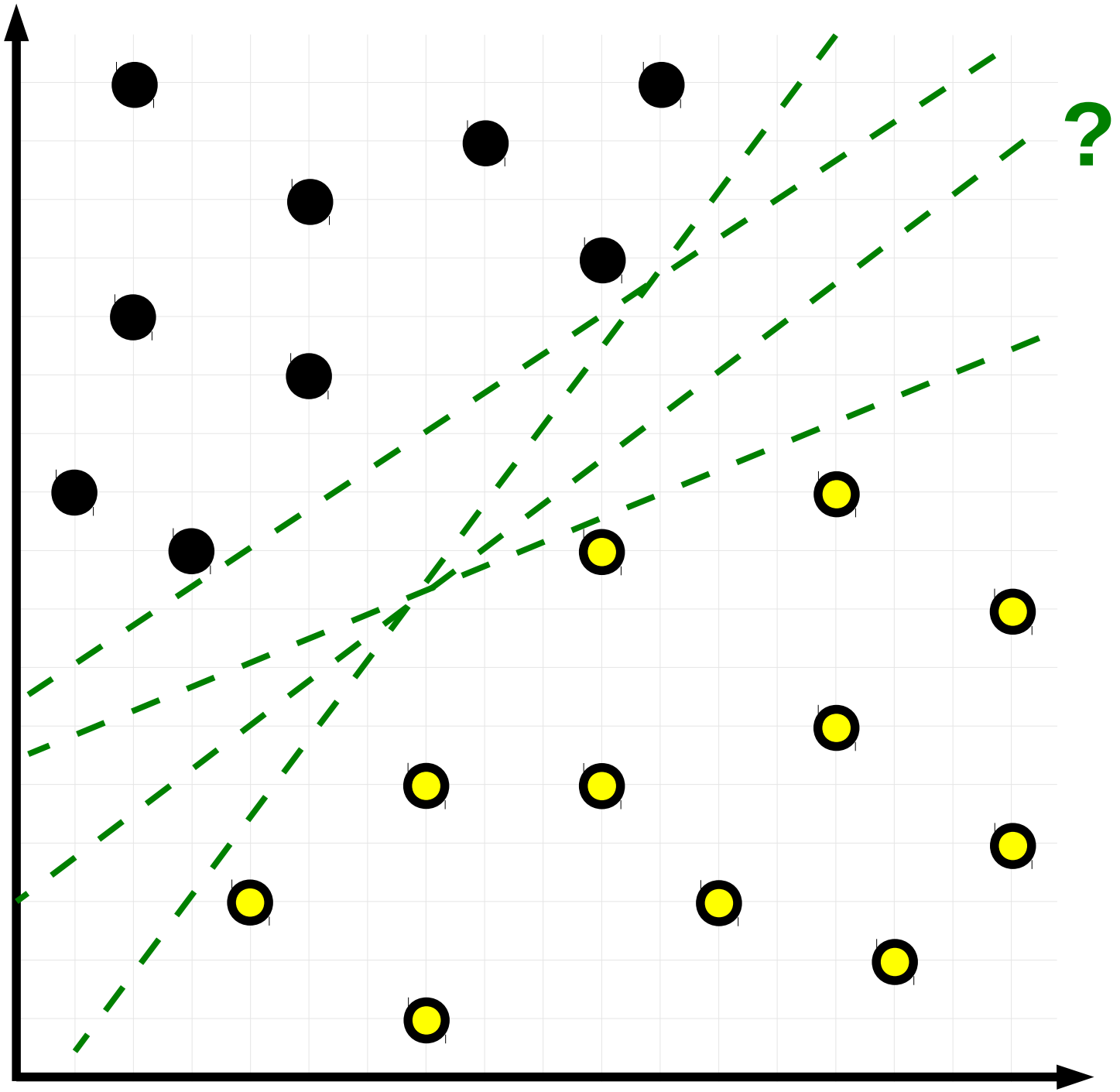
(versión vectorial/matricial de las que vimos antes).

**Ejemplo:**

| | |
|---|---|
| $p = 2$ | (dos variables) |
| $k = 3$ | (tres clases) |
| $\pi_1 = \pi_2 = \pi_3$ | (clases equiprobables) |



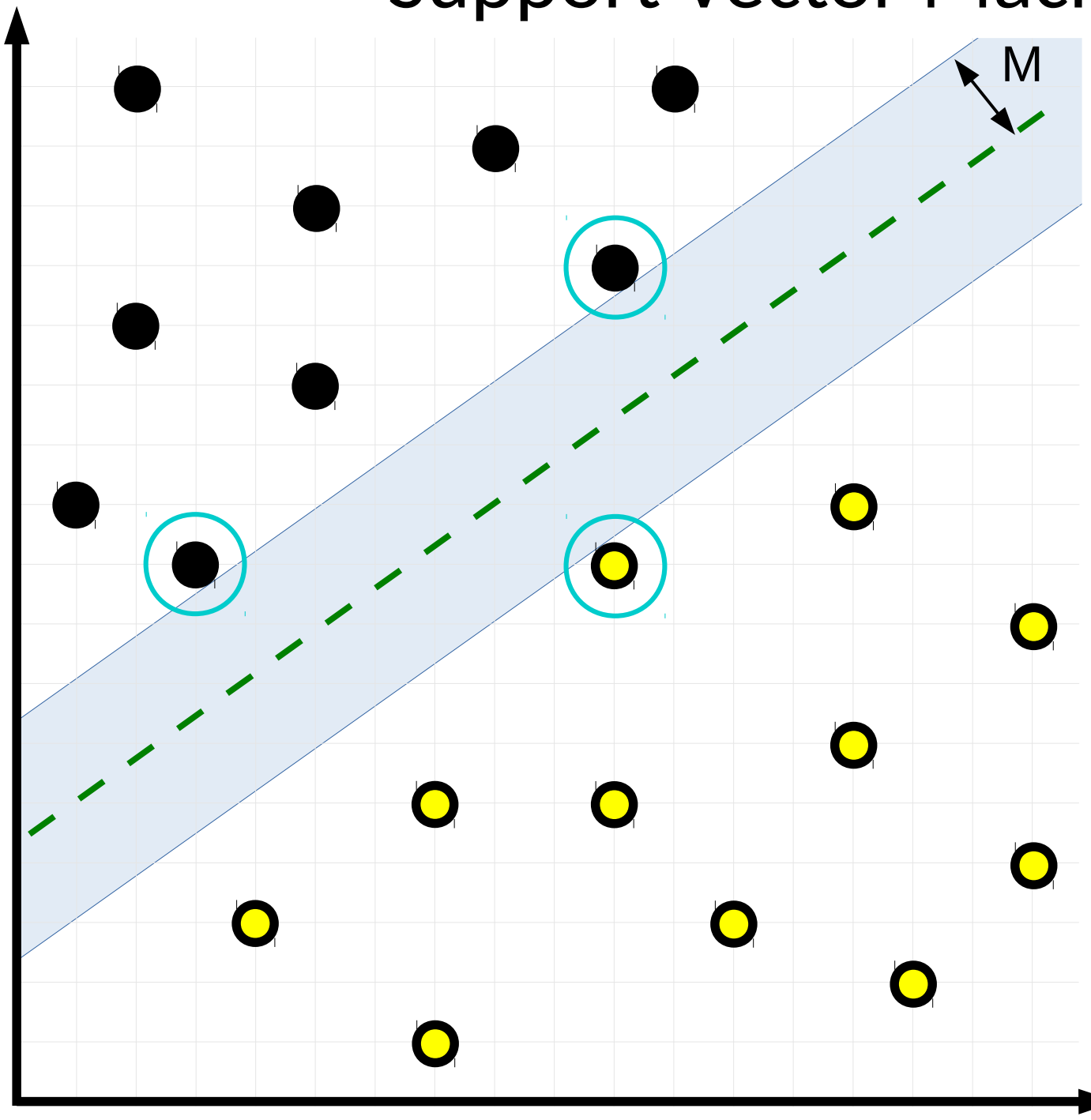

Fuente:
ISLR, p.143




Otra idea: buscar una recta que separe las clases lo mejor posible, sin tener que modelar la distribución de los datos de cada clase.

¿Cómo podemos elegir esa recta?

# Support Vector Machines




Margen M: Distancia de las instancias más cercanas a la recta (hiperplano) de decisión.

Instancias más cercanas: "support vectors".

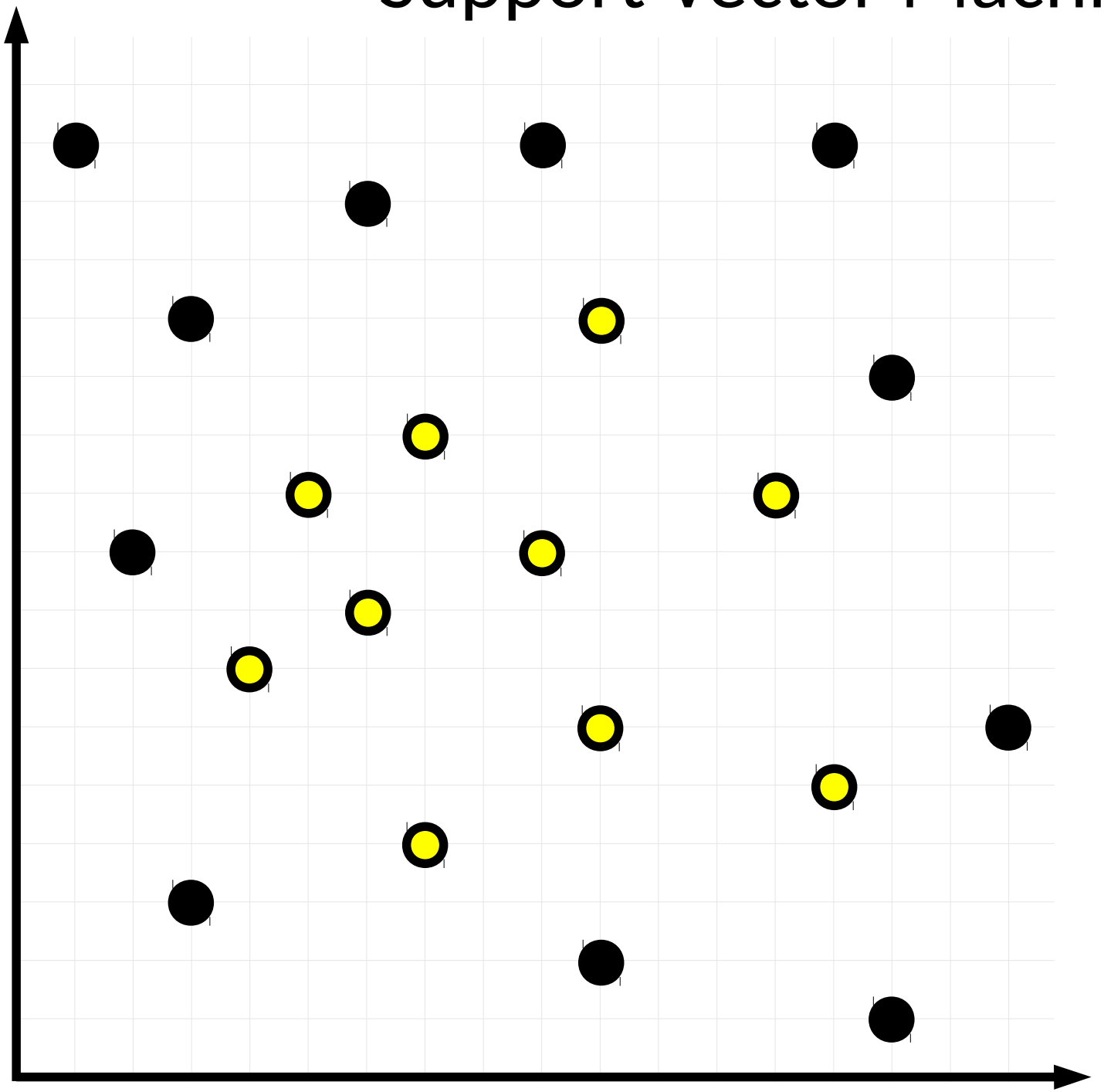
SVM: Busca maximizar M.

Problema de optimización.

Solución eficiente: programación cuadrática (QP).

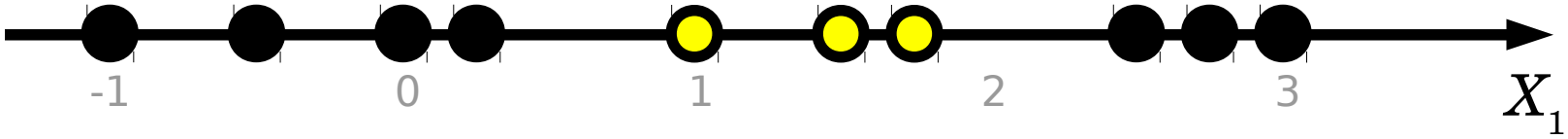
Ejercicio: ¿para qué sirve el hiperparámetro *C*?

# Support Vector Machines



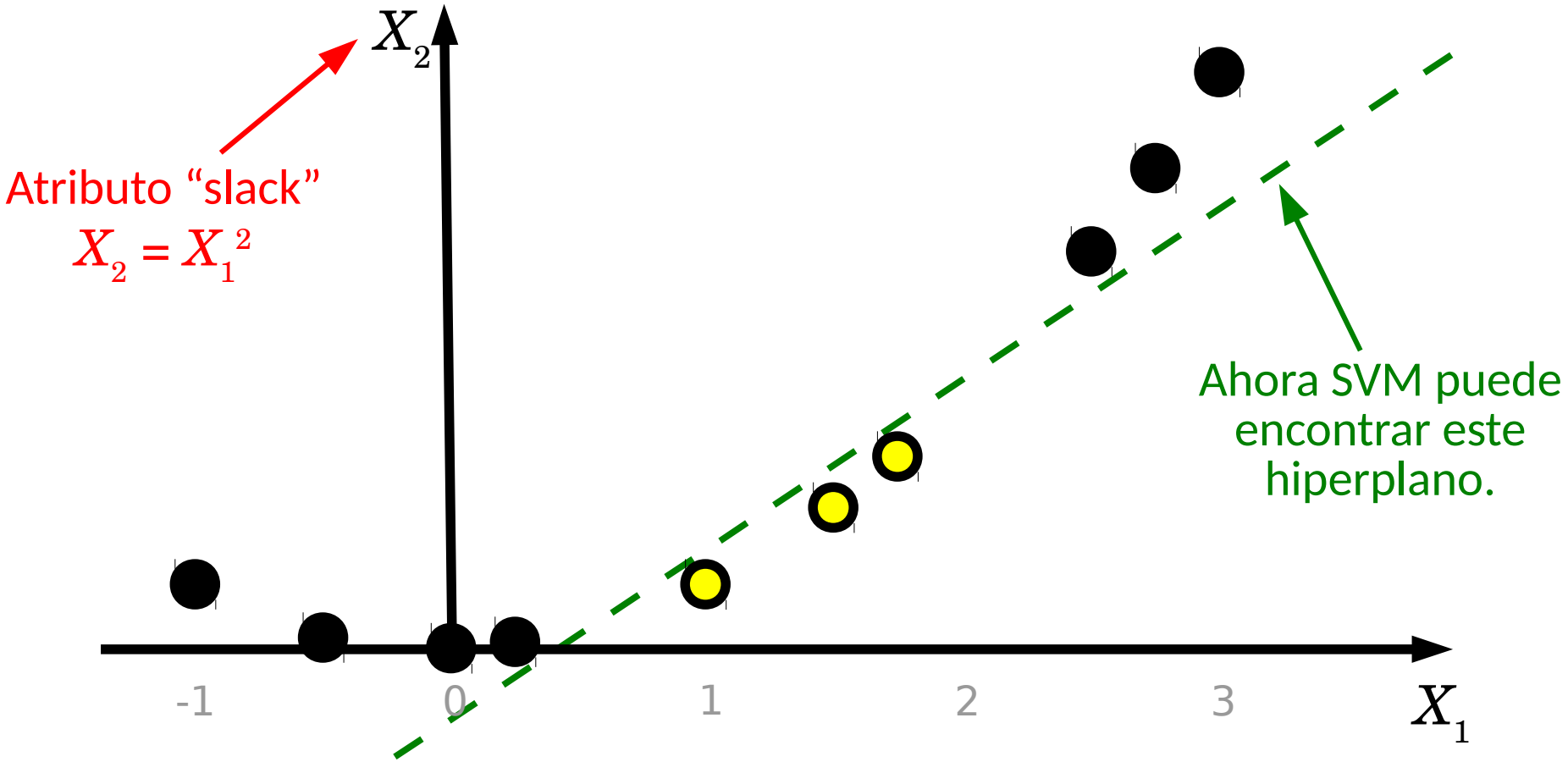
¿Qué hacemos si las instancias no son linealmente separables?

**Datos originales:**




Un único atributo $X_1$. Instancias no linealmente separables.

**Datos transformados:**



# Kernel Trick

- Transformación de vectores de atributos. Ej: $\Phi([X_1]) = ([X_1, X_1^2])$
- Expandir las transformaciones explícitamente es muy costoso.
- Lo evitamos mediante kernels.
  - Kernel: Generalización del producto interno que nos permite operar con nuevos atributos en forma implícita.

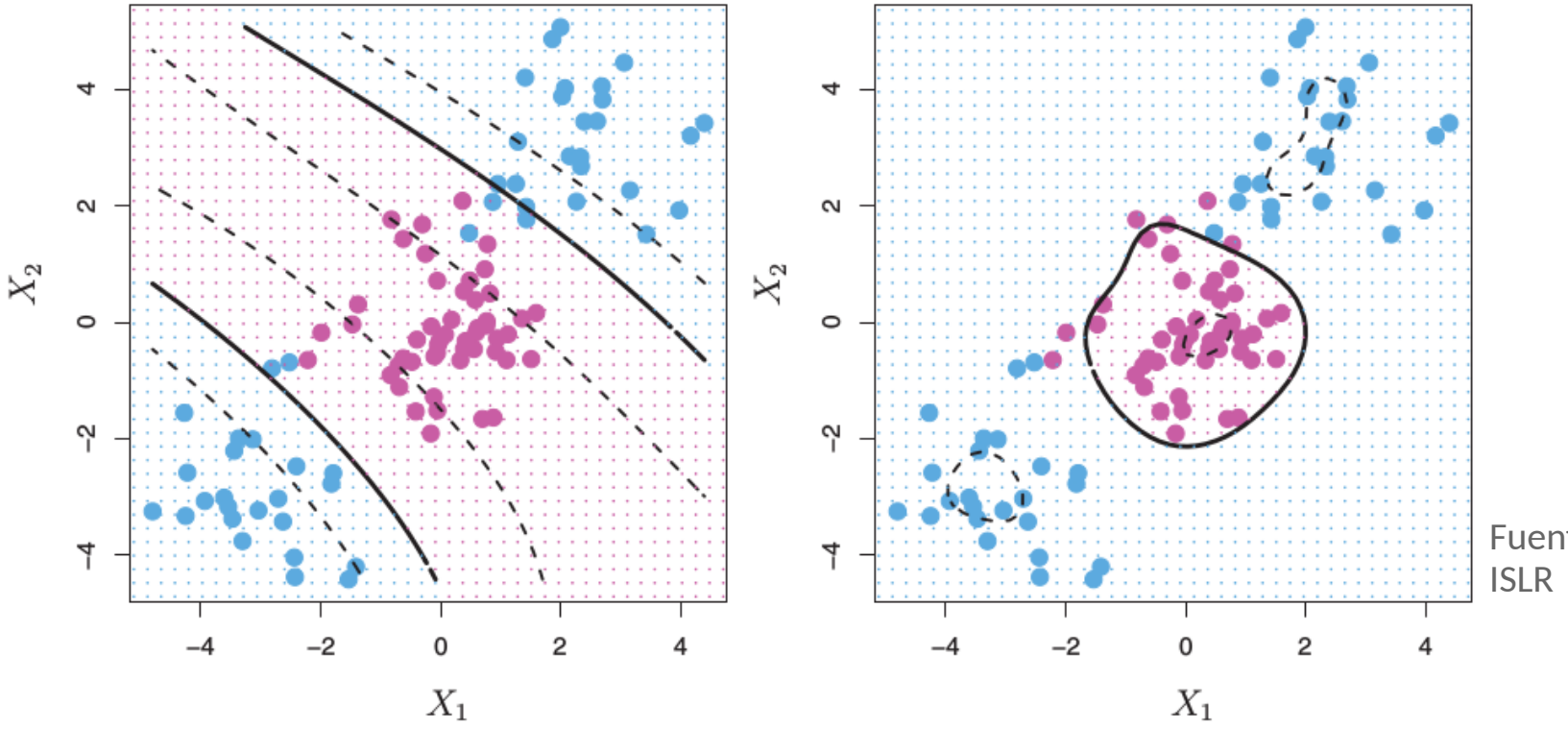
$$K(x^{(1)}, x^{(2)}) = <\Phi(x^{(1)}), \Phi(x^{(2)})>$$

donde $x^{(1)}$, $x^{(2)}$ son instancias.

  - Si un algoritmo (ej. SVM) puede expresarse en términos de productos internos entre instancias, reemplazamos las apariciones de $<x^{(1)}, x^{(2)}>$ por $K(x^{(1)}, x^{(2)})$.
  - Así, ejecutamos SVM implícitamente en dimensiones superiores.

# Support Vector Machines


- Kernels más usados: lineal, polinomial, sigmoideo, RBF.
  - Souza (2010), "Kernel Functions for Machine Learning Applications"




Fuente: ISLR

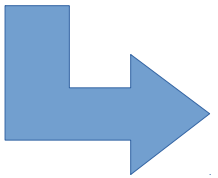
**FIGURE 9.9.** Left: *An SVM with a polynomial kernel of degree 3 is applied to the non-linear data from Figure 9.8, resulting in a far more appropriate decision rule.* Right: *An SVM with a radial kernel is applied. In this example, either kernel is capable of capturing the decision boundary.*

# SVM – Clases múltiples

- Hasta ahora: clasificación binaria.
- *K* clases:
  - Para cada clase *k*, entrenar un SVM para discriminar *k* del resto (clasificación *OVA*: *one-versus-all*).
  - Para una nueva instancia $x^{(q)}$, correr los *K* clasificadores y retornar la clase para la cual $x^{(q)}$ queda a mayor distancia del hiperplano de decisión (i.e., la clase con mayor confianza).

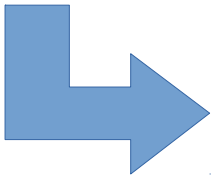
# Atributos Categóricos

- EstadoCivil: {Solterx, Casadx, Viudx, Divorciadx, Otro}



EstadoCivil_Solterx: {0, 1}
EstadoCivil_Casadx: {0, 1}
EstadoCivil_Viudx: {0, 1}
EstadoCivil_Divorciadx: {0, 1}
EstadoCivil_Otro: {0, 1}

- Palabras: BagOfWords



Palabras_hola: $\mathbb{N}$
Palabras_mundo: $\mathbb{N}$
Palabras_la: $\mathbb{N}$
Palabras_y: $\mathbb{N}$
Palabras_cuando: $\mathbb{N}$
...

# Repaso

- Clasificadores:
  - K Vecinos más Cercanos (KNN)
  - Bayes Classifier
  - Naive Bayes Classifier
  - Linear Discriminant Analysis (LDA)
  - Support Vector Machines (SVM)

- Próximos temas:
  - Sesgo y varianza de algoritmos.
  - Ensambles de modelos.