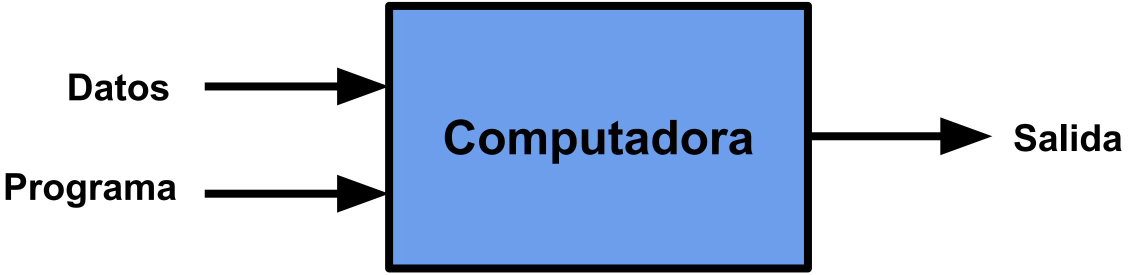
Introducción al aprendizaje automático

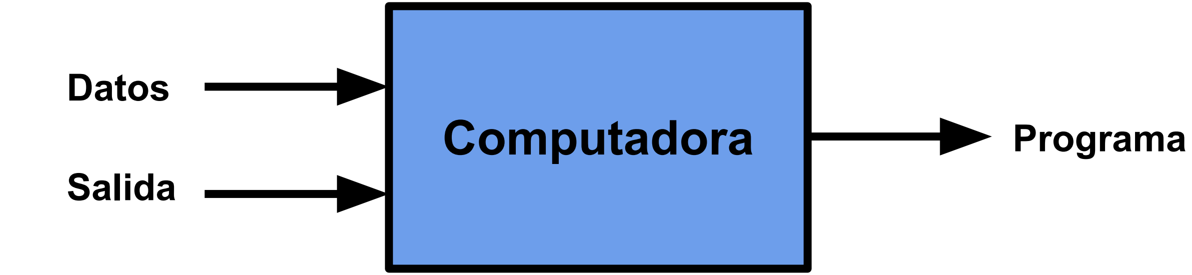
...

#1. Introducción al aprendizaje automático. Regresión. Clasificación. Naïve Bayes

Programación tradicional



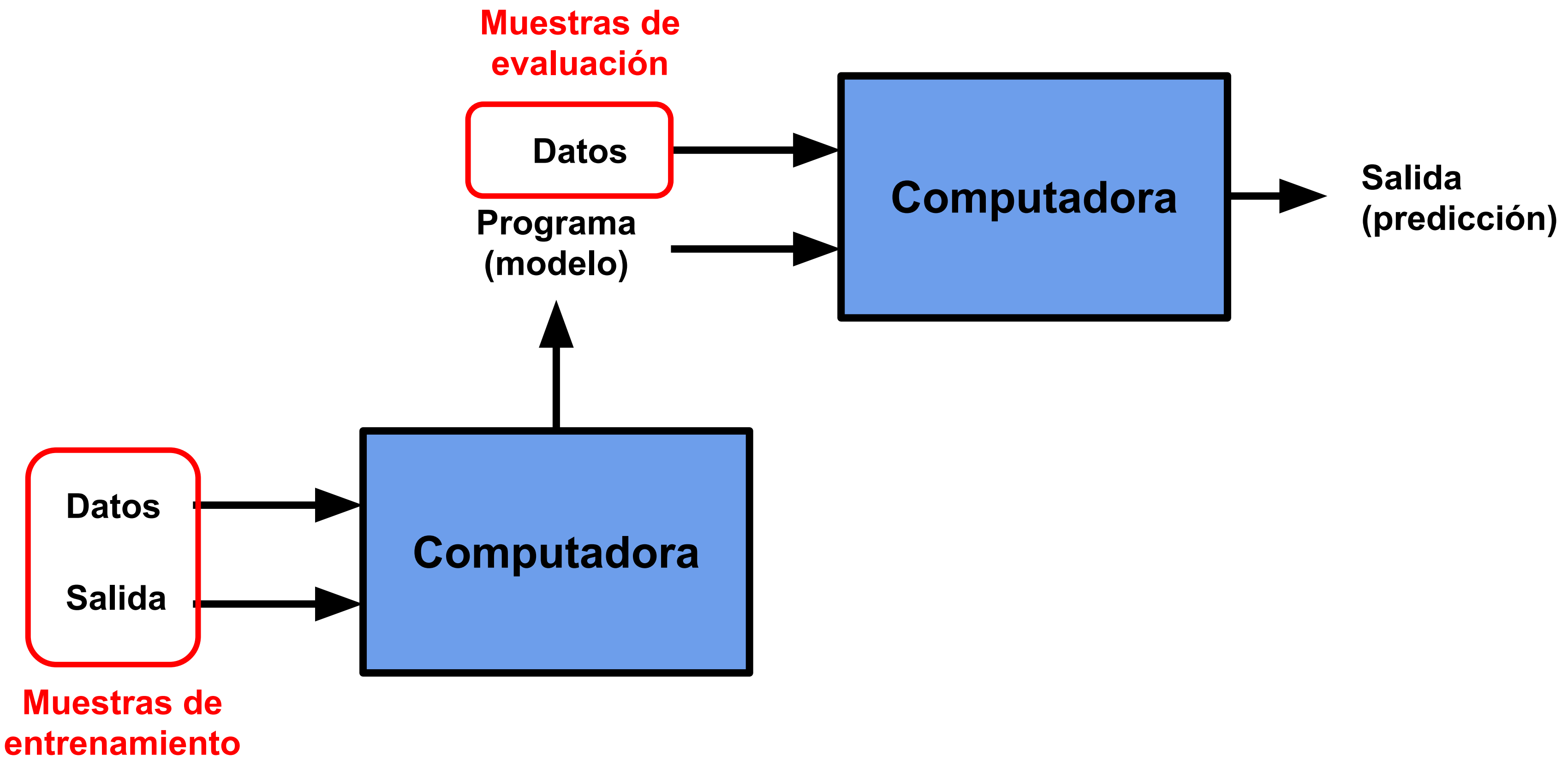
Aprendizaje automático



Tipos de aprendizaje

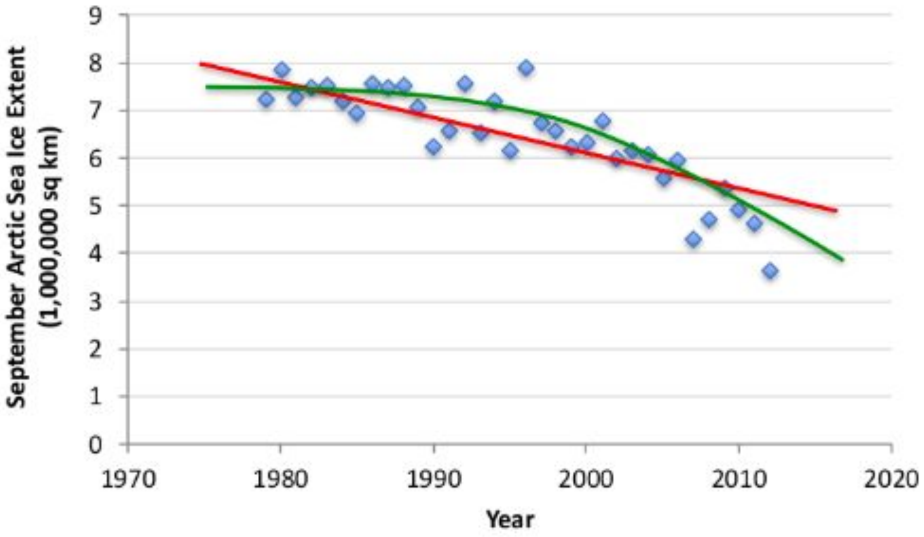
- **Aprendizaje supervisado (inductivo)**
  Datos de entrenamiento + salida esperada
- **Aprendizaje no supervisado**
  Datos de entrenamiento (sin salida esperada)
- **Aprendizaje semi-supervisado**
  Datos de entrenamiento + **pocas** salida esperadas
- **Aprendizaje por refuerzo**
  "Recompensas" por secuencias de acciones

Aprendizaje supervisado: entrenamiento vs. evaluación



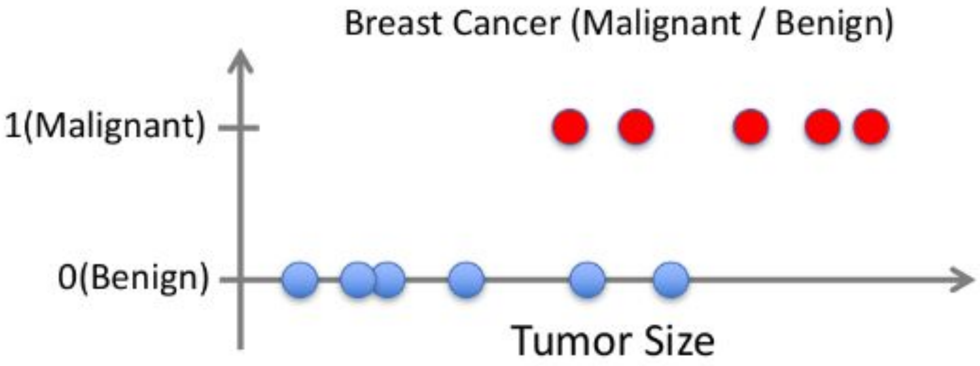
Aprendizaje supervisado: regresión

- Dados $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$
- Aprender una $f(x)$ que permita predecir y a partir de x
  - Si y está en $\mathbb{R}^n$ $\rightarrow$ **regresión**



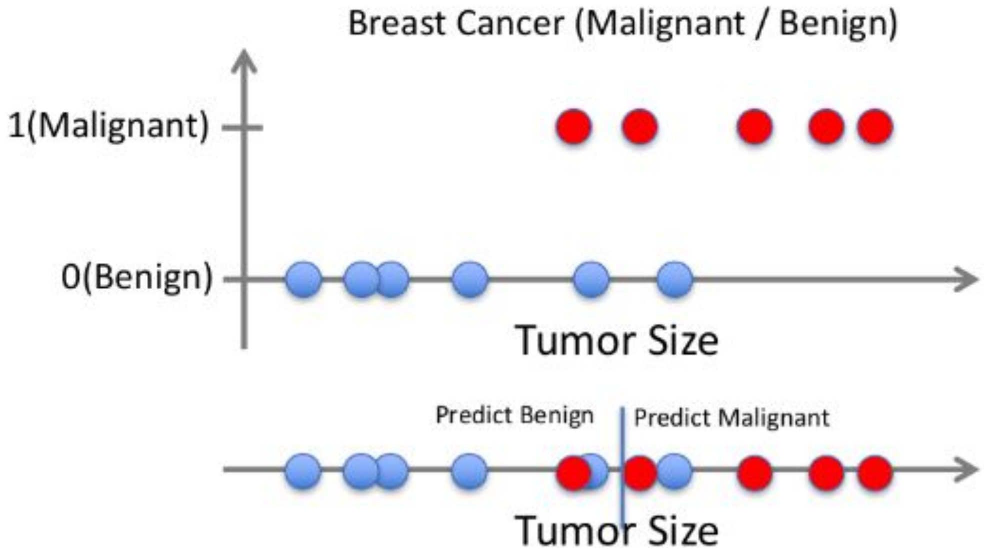
Aprendizaje supervisado: clasificación

- Dados $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$
- Aprender una $f(x)$ que permita predecir y a partir de x
  - Si y es categórica $\rightarrow$ **clasificación**



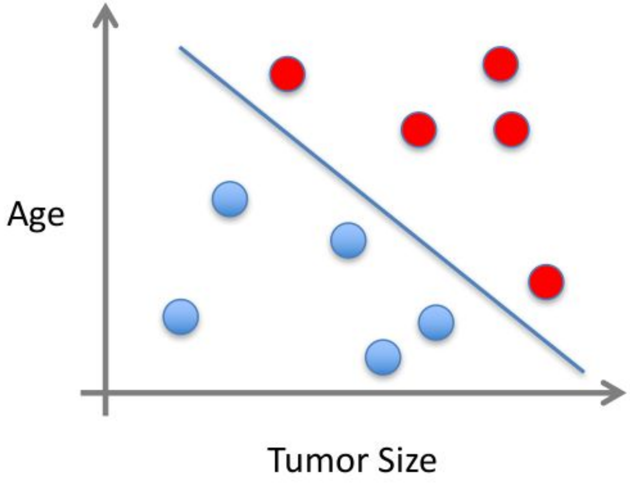
Aprendizaje supervisado: clasificación

- Dados $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$
- Aprender una $f(x)$ que permita predecir y a partir de x
  - Si y es categórica $\rightarrow$ **clasificación**



Supervised Learning

- x can be multi-dimensional
  - Each dimension corresponds to an attribute



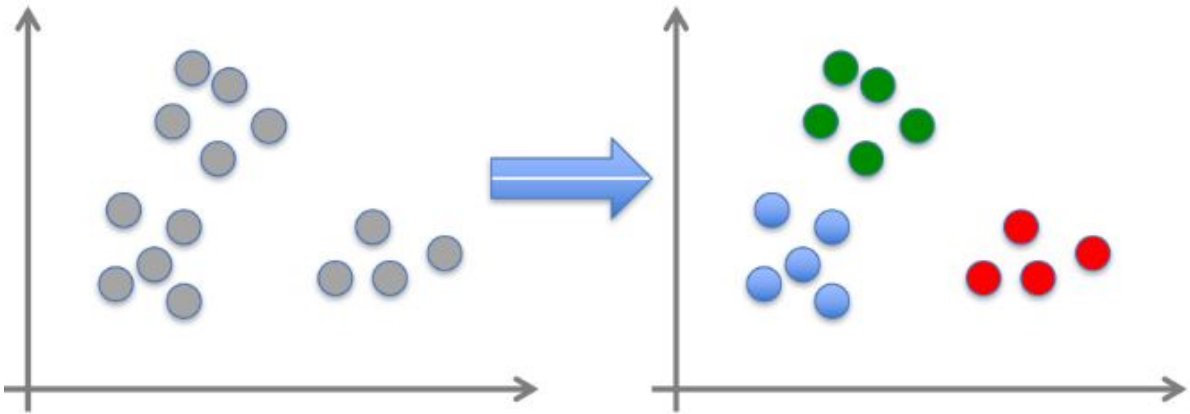
- Clump Thickness
- Uniformity of Cell Size
- Uniformity of Cell Shape

…

Based on example by Andrew Ng

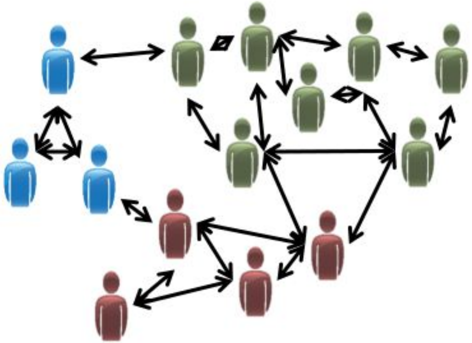
Aprendizaje no supervisado

- Dados $x_1, x_2, \ldots, x_n$
- Aprender la estructura interna de los datos
    - p.ej. *clustering*



Aprendizaje no supervisado

Organize computing clusters



Social network analysis

Market segmentation

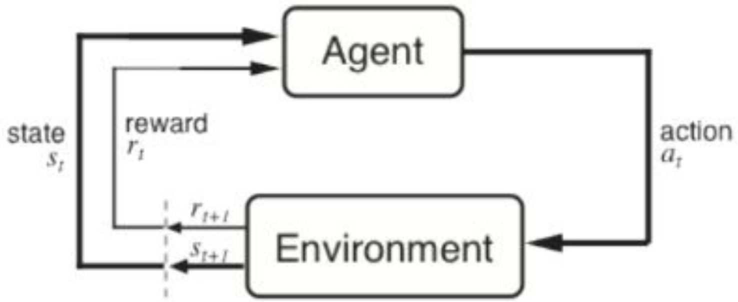
Astronomical data analysis

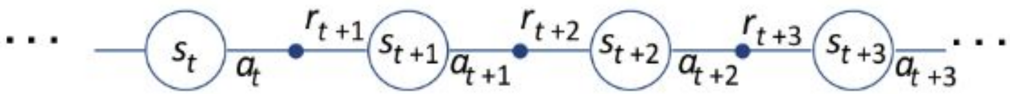
Aprendizaje por refuerzo

- Dada una secuencia de estados y acciones con recompensa (*reward*), generar una política (*policy*)
  - política = mapeo estados → acciones que nos dicen que hacer en un determinado estado
- Ejemplos:
  - Juegos
  - Navegación en robótica
  - Control
  - ...

La interfaz agente-entorno

- El agente y el entorno interactúan a instantes discretos de tiempo
  - $t=0,1,...,K$
  - el agente observa el estado S_t en el paso t
  - produce una acción a_t en el paso t
  - obtiene una recompensa r_{t+1} en el paso $t+1$
  - genera un nuevo estado s_{t+1} en el paso $t+1$

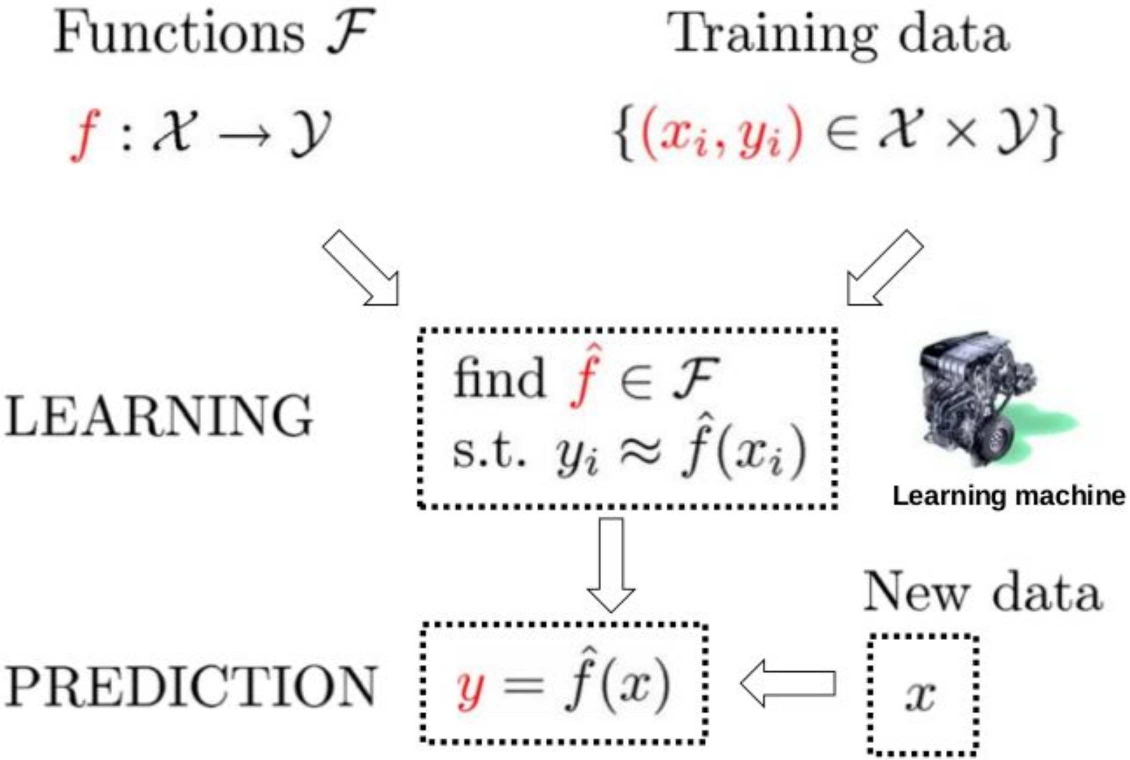




Sobre "aprendizaje"

- Se puede ver como la utilización directa o indirecta de la experiencia para aproximar una determinada función.
- La aproximación de dicha función corresponde a una búsqueda en un espacio de hipótesis (espacio de funciones) por aquella que mejor ajusta el conjunto de datos de entrenamiento.
- Distintos métodos de aprendizaje automático asumen distintos espacios de hipótesis o utilizan distintas estrategias de búsqueda.

Aprendizaje supervisado



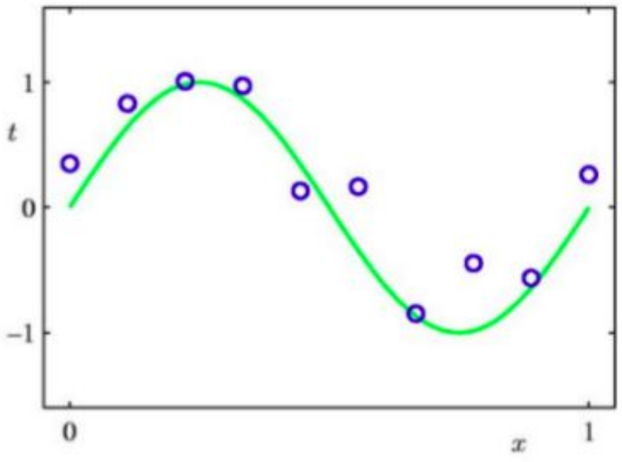
Regresión

Regresión

- Disponemos de N pares de entrenamiento (observaciones)

$$\{(x_i, y_i)\}_{i=1}^{N} = \{(x_1, y_1), \cdots, (x_N, y_N)\}$$

- El problema de regresión consiste en estimar *f(x)* a partir de estos datos

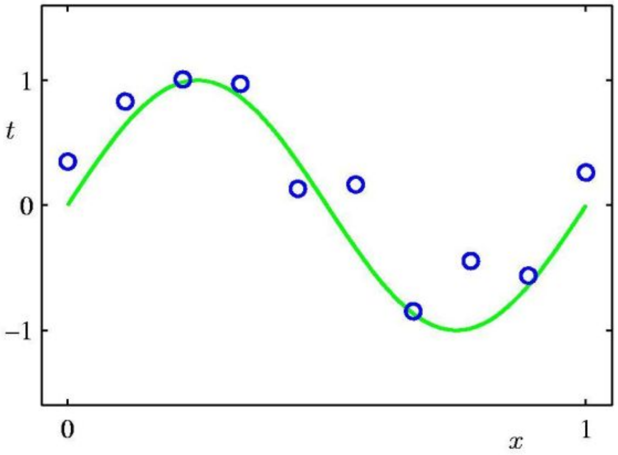


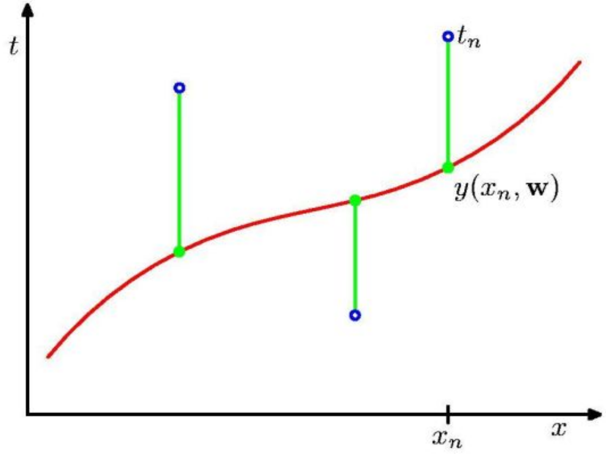
Regresión polinomial

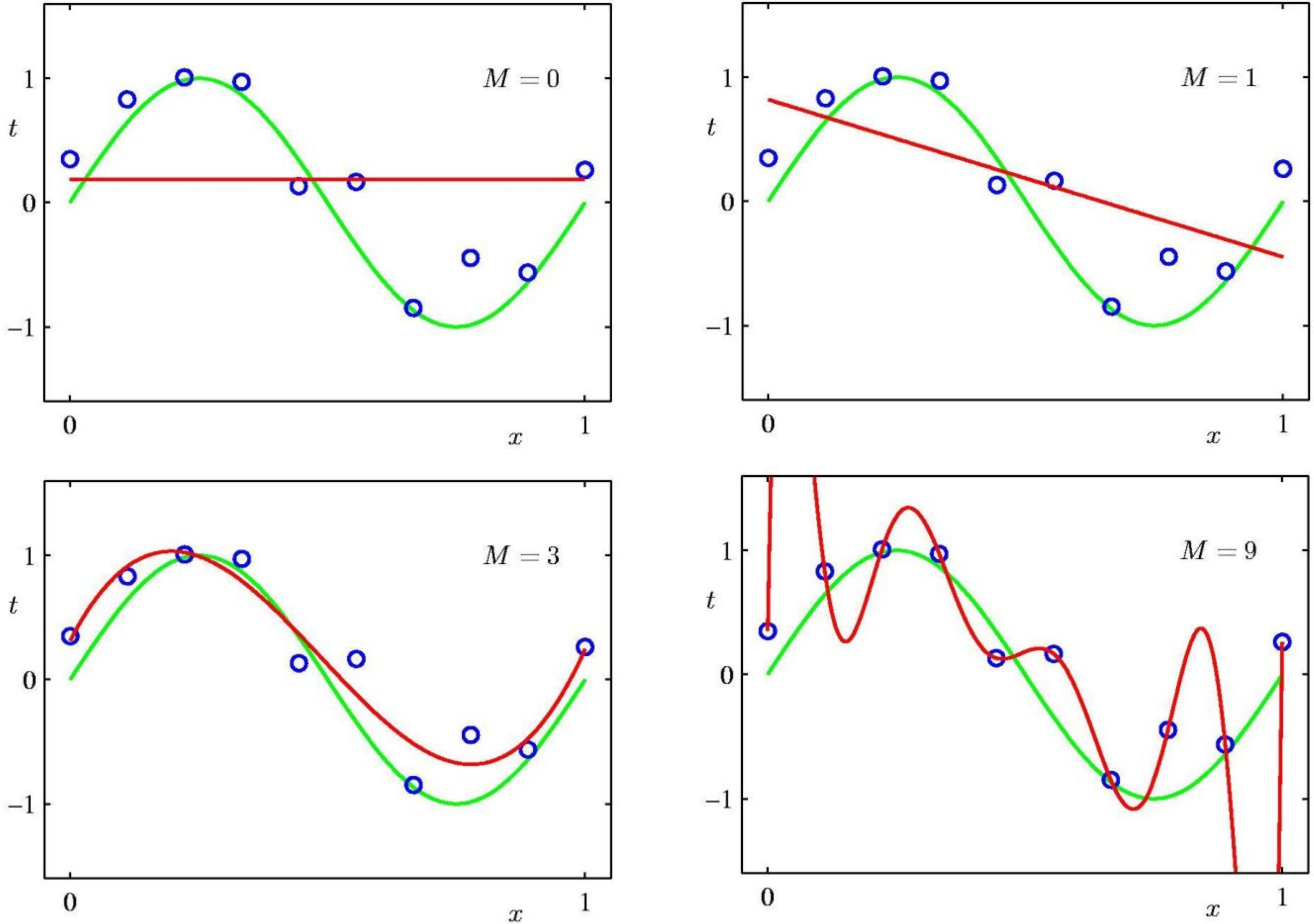
- En verde se ilustra la función "verdadera" (inaccesible)
- Las muestras son uniformes en x y poseen ruido en y
- Utilizaremos una **<u>función de costo</u>** (error cuadrático) que mida el error en la predicción de y mediante $f(x)$

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n,\mathbf{w}) - t_n\}^2$$

$$y(x,\mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$



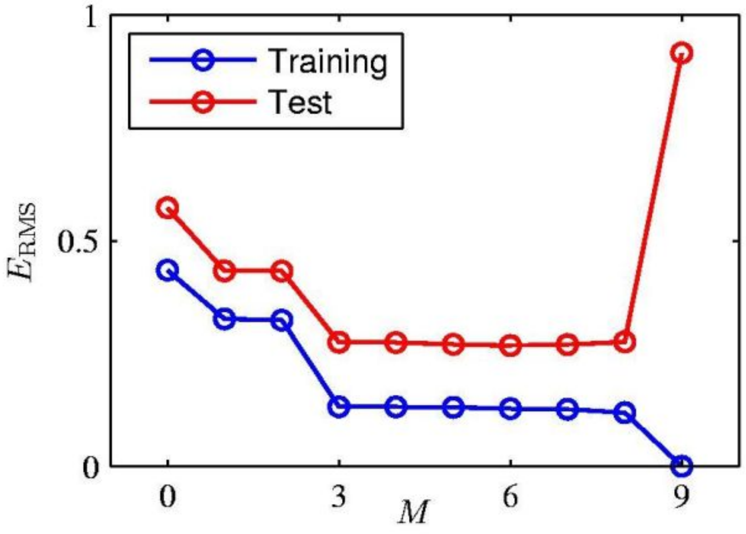




M = 0
M = 1
M = 3
M = 9
t
x
0
1
−1

Sobreajuste (*overfitting*)

- Datos de test: otra muestra de los misma función subyacente
- El error de entrenamiento se hace cero, pero el de test crece con M



Root-Mean-Square (RMS) Error: $E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^\star)/N}$

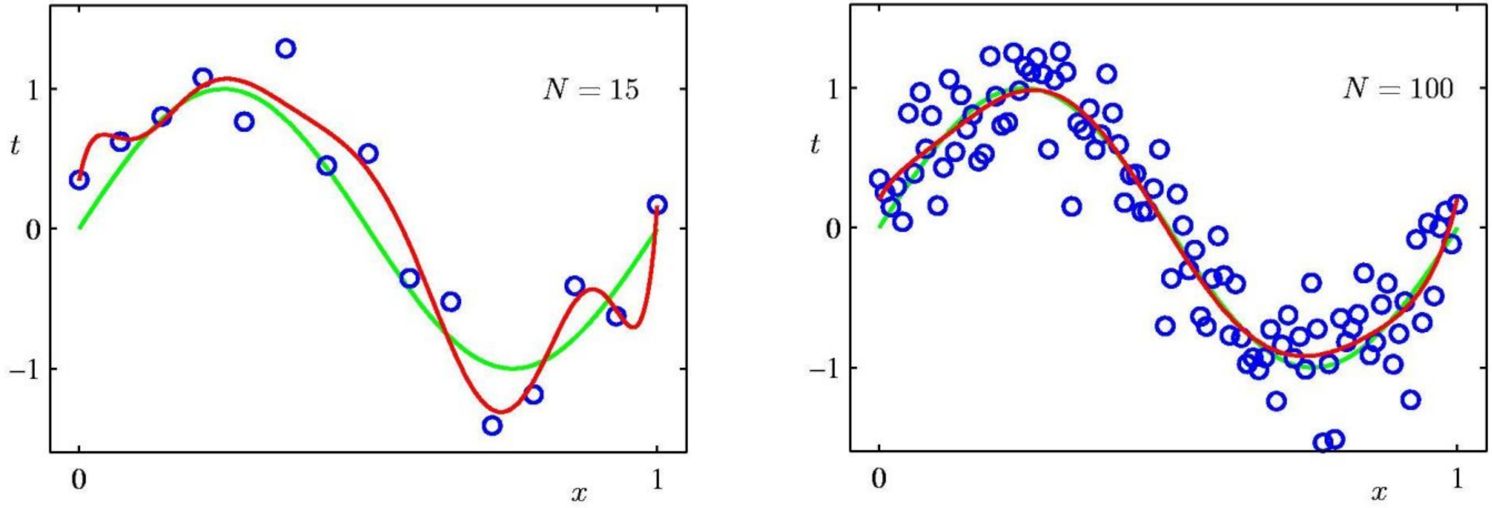
Bondad de ajuste vs. complejidad de modelo

- Si el modelo tiene tantos grados de libertad como los presentes en los datos de entrenamiento, puede ajustarlos perfectamente
- El objetivo en aprendizaje automático no es el ajuste perfecto, sino la generalización a conjuntos no vistos
- Podemos decir que un modelo generaliza, si puede explicar los datos empleando una complejidad acotada

	$M=0$	$M=1$	$M=3$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

Prevenir el sobreajuste (I)

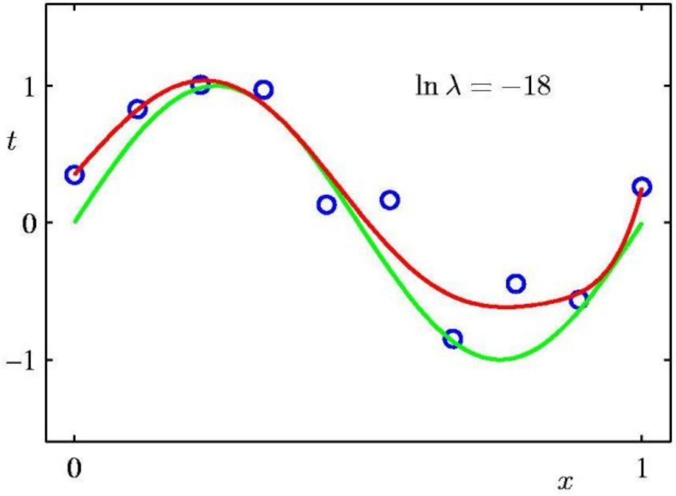
- Agregar más datos (más que la "complejidad" del modelo)

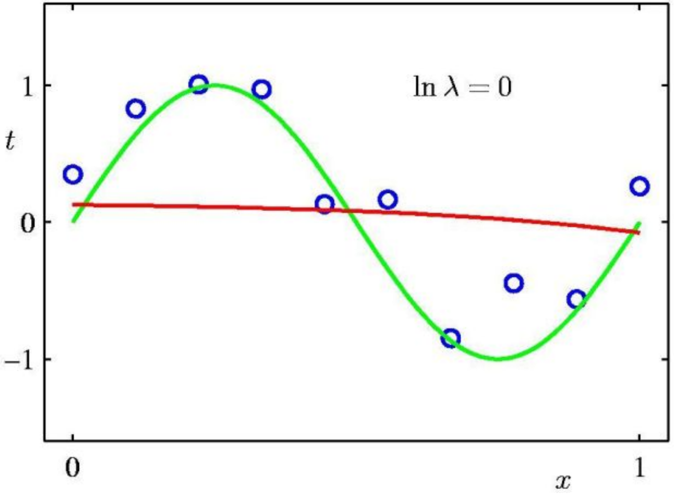


Prevenir el sobreajuste (II)

- Regularización: penalizar valores grandes de los coeficientes

$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n,\mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

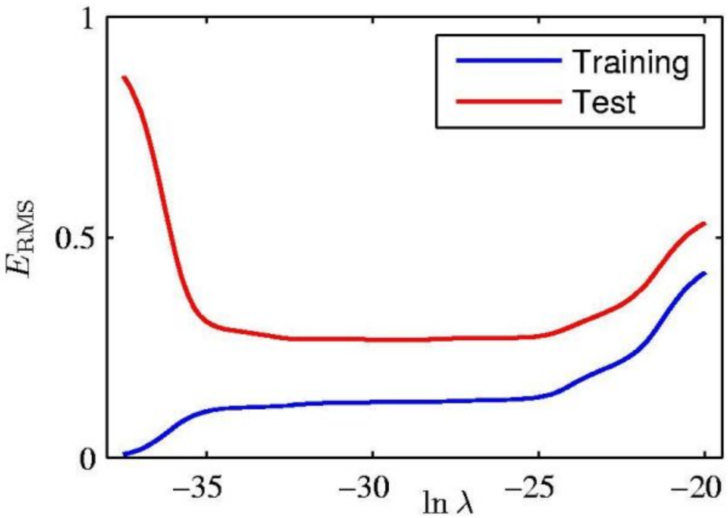




Prevenir el sobreajuste (II)

- Regularización: penalizar valores grandes de los coeficientes

$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n,\mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$



Término de regularización (*ridge*)

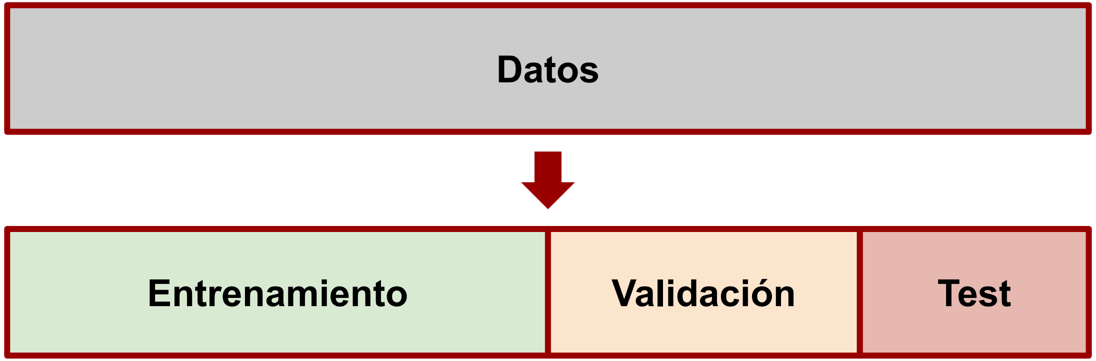
λ = hiperparámetro

	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01

Elección de *hiperparámetros*

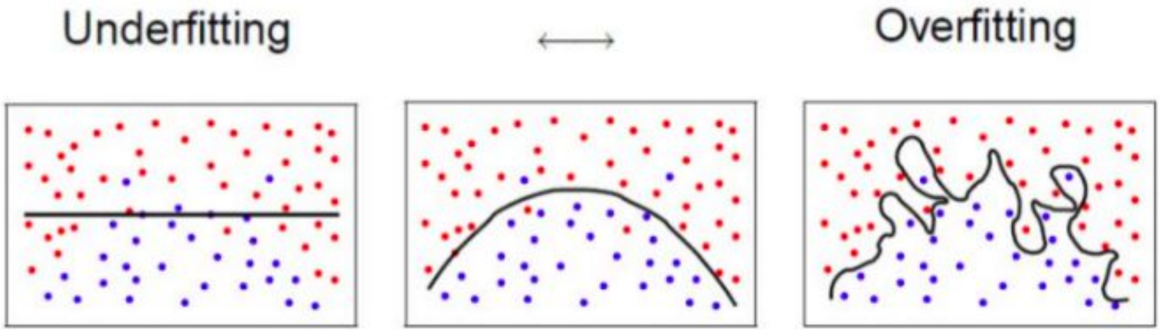
Dividir el conjunto total de ejemplos en tres subconjuntos

- **Entrenamiento**: aprendizaje de variables del modelo
- **Validación**: ajuste/elección de hiperparámetros
- **Test**: estimación <u>final</u> de la performance del modelo entrenado (y con hiperparámetros elegidos adecuadamente



Generalización en clasificación

- Complejidad del modelo ⇔ complejidad de la frontera de decisión



Clasificación

Clasificación binaria

- Disponemos de N pares de entrenamiento (observaciones)

$$\{(x_i, y_i)\}_{i=1}^{N} = \{(x_1, y_1), \cdots, (x_N, y_N)\}$$

con $x_i \epsilon \mathbb{R}^n$, $y_i \epsilon \{-1, +1\}$.

- Aprender una $f(x)$ tal que

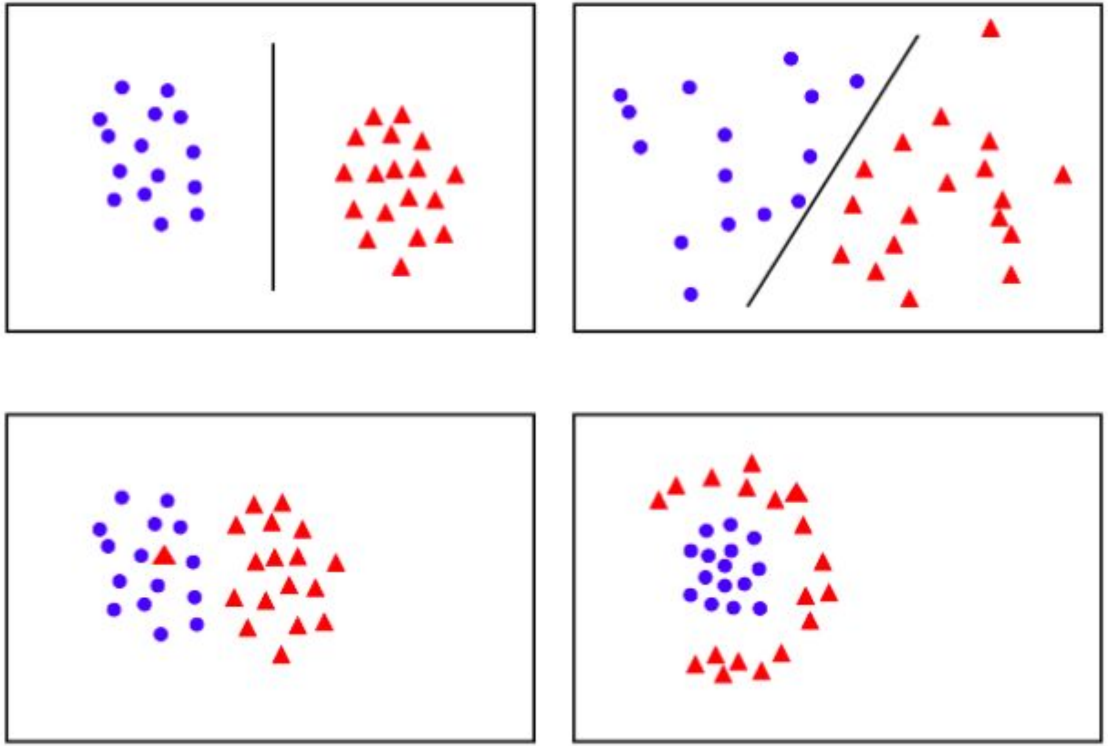
$$f(\mathbf{x}_i) \begin{cases} \geq 0 & y_i = +1 \\ < 0 & y_i = -1 \end{cases}$$

es decir: $y_i\, f(x_i) > 0$ para una clasificación correcta.

Separabilidad lineal

linealmente
separable

no
linealmente
separable



Clasificadores lineales

- La entrada es un vector x_i de dimensionalidad n
- La salida es una etiqueta $y_i \in \{-1, +1\}$
- Clasificador = función de predicción + función de decisión

$$g(f(x)) \rightarrow \{-1, +1\}$$

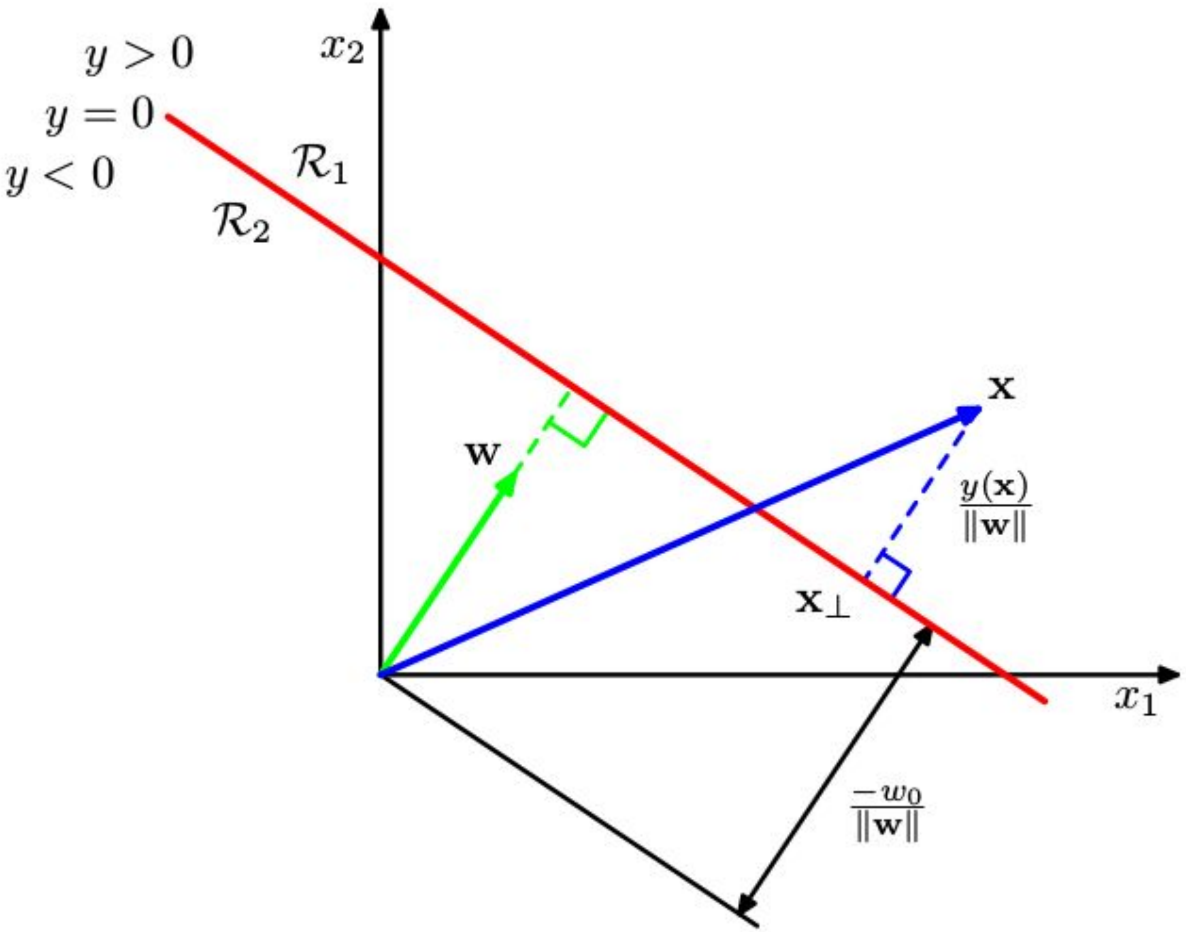
- Función de predicción **lineal**

$$f(x) = w^{\mathrm{T}}x + w_0$$

- Función de decisión

$$g(z) = sign(z)$$

$$g(f(x)) = sign(w^{\mathrm{T}}x + w_0)$$



$y > 0$
$y = 0$
$y < 0$
$\mathcal{R}_1$
$\mathcal{R}_2$
x_2
x_1
$\mathbf{w}$
$\mathbf{x}$
$\mathbf{x}_\perp$
$\frac{y(\mathbf{x})}{\|\mathbf{w}\|}$
$\frac{-w_0}{\|\mathbf{w}\|}$

El algoritmo del "perceptrón"

- Propuesto por Rosemblatt en 1958

- El objetivo es encontrar un hiperplano de separación
    - Si los datos son linealmente separables, lo encuentra

- Es un algoritmo *online* (procesa un ejemplo a la vez)

- Muchas variantes ...

El algoritmo del "perceptrón"

Entrada:

- una secuencia de pares de entrenamiento $(x_1,y_1), (x_2, y_2)$ ...
- Una tasa de aprendizaje r

Algoritmo:

- Inicializar $w^{(0)} \in \mathbb{R}^{n}$
- Para cada ejemplo (x_i,y_i)
  - Predecir $y_i' = sign(w^{T}x_i + w_0)$
  - Si $y_i' \neq y_i$:

    $w^{(t+1)} \leftarrow w^{(t)} + r\,(y_i x_i)$

El algoritmo del "perceptrón"

Entrada:

- una secuencia de pares de entrenamiento (x_1,y_1), (x_2, y_2) ...
- Una tasa de aprendizaje r (número pequeño y menor a 1)

Algoritmo:

- Inicializar $w^{(0)} \in \mathbb{R}^n$
- Para cada ejemplo (x_i,y_i)
  - Predecir $y_i' = sign(w^T x_i + w_0)$
  - Si $y_i' \neq y_i$:

    $w^{(t+1)} \leftarrow w^{(t)} + r\,(y_i x_i)$

Nota: el término de bias se puede contemplar definiendo las entrada como $(x_i^T\ 1)^T \in \mathbb{R}^{n+1}$.

Pregunta: ¿qué implica que $w_0=0$?

El algoritmo del "perceptrón"

Entrada:

- una secuencia de pares de entrenamiento (x_1,y_1), (x_2, y_2) ...
- Una tasa de aprendizaje r (número pequeño y menor a 1)

Algoritmo:

- Inicializar $w^{(0)} \epsilon \mathbb{R}^n$
- Para cada ejemplo (x_i,y_i)
  - Predecir $y_i' = sign(w^T x_i)$
  - Si $y_i' \neq y_i$:

    $w^{(t+1)} \leftarrow w^{(t)} + r\,(y_i x_i)$

Actualiza solo cuando comete un error

Error en positivos:

$$w^{(t+1)} \leftarrow w^{(t)} + r\,x_i$$

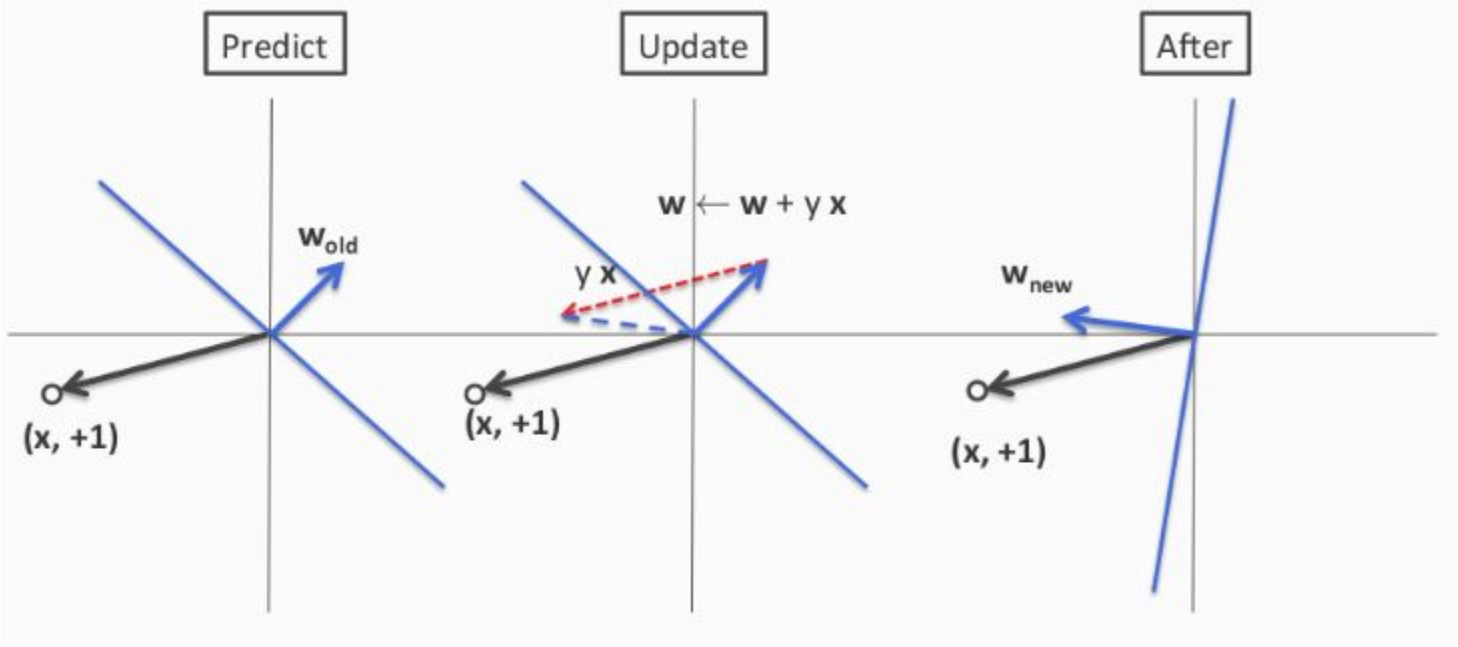
Error en negativos:

$$w^{(t+1)} \leftarrow w^{(t)} - r\,x_i$$

Si $y_i w^T x_i \leq 0 \rightarrow$ error

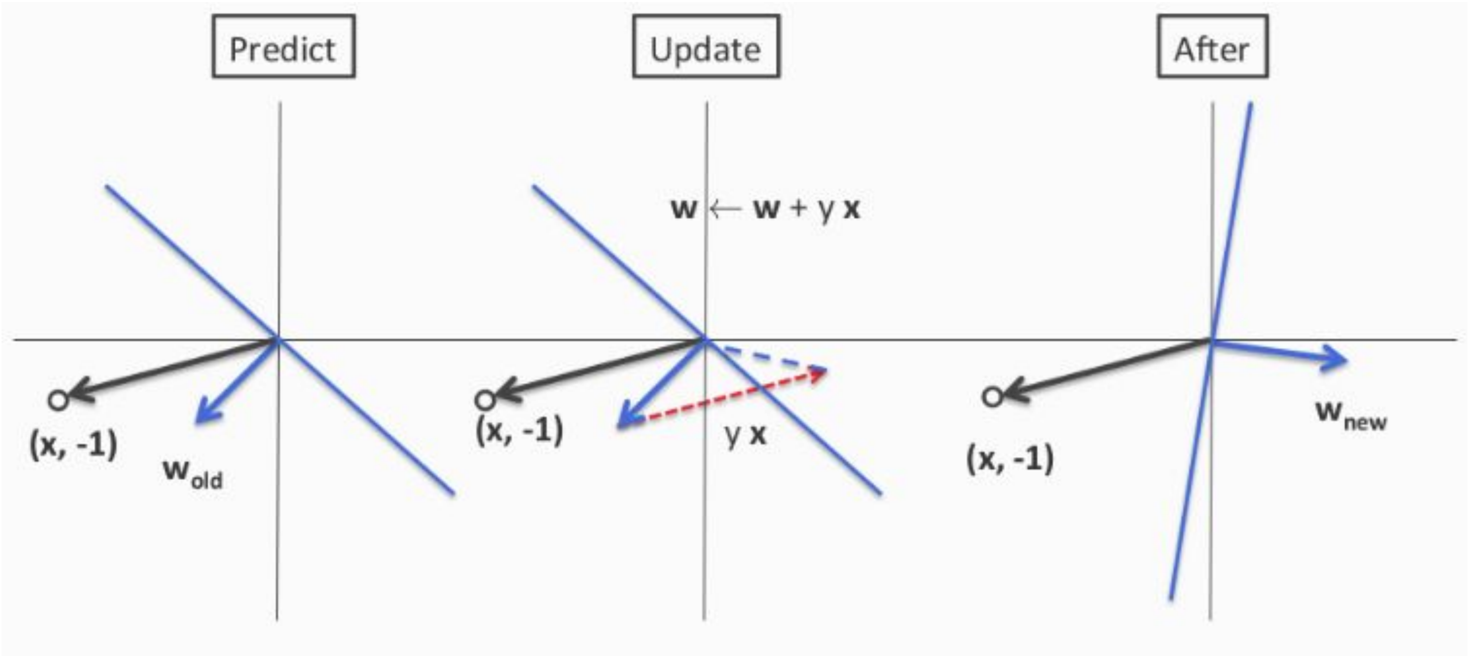
Dinámica de actualización

Error en ejemplo **positivo**:



Dinámica de actualización

Error en ejemplo **negativo**:



1. The "standard" algorithm

Given a training set D = $\{(\mathbf{x}_i, y_i)\}$, $\mathbf{x}_i \in \Re^n$, $y_i \in \{-1,1\}$

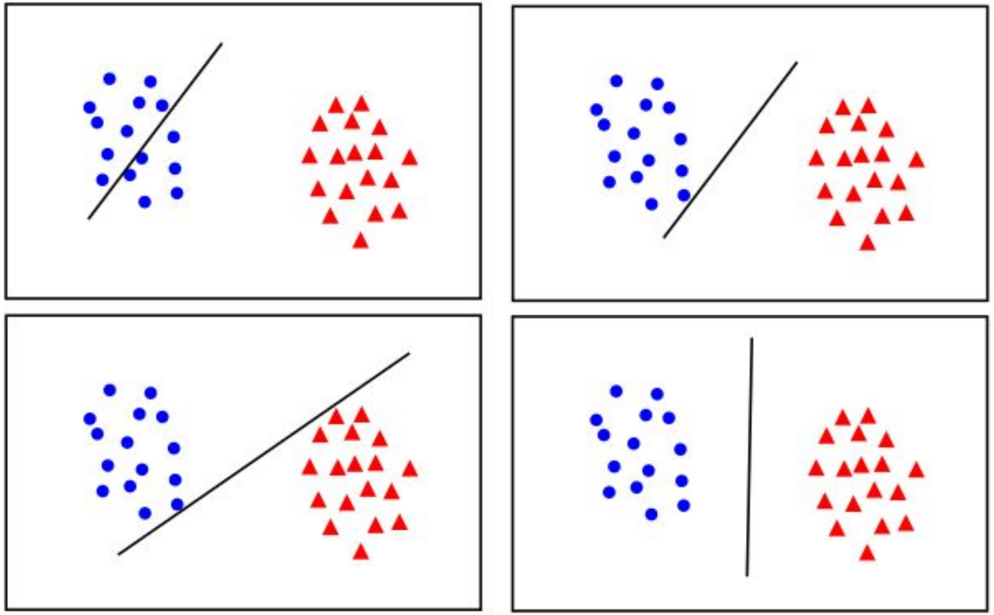
1. Initialize $\mathbf{w} = 0 \in \Re^n$
2. For epoch = 1 … T:
    1. Shuffle the data
    2. For each training example $(\mathbf{x}_i, y_i) \in$ D:
        - If $y_i\, \mathbf{w}^T\mathbf{x}_i \leq 0$, update $\mathbf{w} \leftarrow \mathbf{w} + r\, y_i\, \mathbf{x}_i$
3. Return $\mathbf{w}$

T is a hyper-parameter to the algorithm

Another way of writing that there is an error

Prediction: sgn($\mathbf{w}^T\mathbf{x}$)

¿Cuál es el mejor w?



Solución de **margen máximo**: el hiperplano más estable ante perturbaciones de la entrada

Naïve Bayes

Regla de Bayes

- Dos formas de factorizar una distribución en dos variables:

$$P(x, y) = P(x|y)P(y) = P(y|x)P(x)$$

- Operando:

$$P(x|y) = \frac{P(y|x)}{P(y)}P(x)$$

- ¿Porqué es útil?
  - Nos permite "revertir" el condicional
  - A veces una dirección es difícil de calcular, pero la otra no
  - Es la base de muchos modelos

El clasificador de Bayes

- Distribución conjunta sobre $X_1, \ldots, X_n$ e Y

- Podemos definir una función de predicción de la forma:

$$\arg\max_Y P(Y|X_1, \ldots, X_n)$$

  - por ejemplo: ¿cuál es la probabilidad de que una imagen represente un "5" dado el valor de sus píxeles?

- Problema: ¿cómo computamos $P(Y|X_1, \ldots, X_n)$? …

El clasificador de Bayes

- … ¡Usando regla de Bayes!

$$P(Y|X_1,\ldots,X_n) = \frac{\overbrace{P(X_1,\ldots,X_n|Y)}^{\text{Likelihood}}\,\overbrace{P(Y)}^{\text{Prior}}}{\underbrace{P(X_1,\ldots,X_n)}_{\text{Normalization Constant}}}$$

  - Ahora podemos pensar en modelar cómo los píxeles de la imágen son "generados" dado el número "5".

Naïve Bayes

- Hipótesis: los X_i son independientes dado Y

$$\begin{aligned} P(X_1, X_2|Y) &= P(X_1|X_2, Y)P(X_2|Y) \\ &= P(X_1|Y)P(X_2|Y) \end{aligned}$$

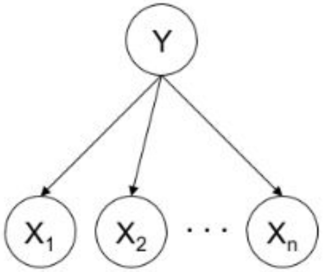
- O en forma más general:

$$\boxed{P(X_1 \ldots X_n|Y) = \prod_i P(X_i|Y)}$$

- Si los X_i consisten en n valores binarios, ¿cuántos parámetros necesito especificar para $P(X_i|Y)$?

El clasificador naïve Bayes

- Dado:
  - Distribución a priori $\mathrm{P}(Y)$
  - n features X_i condicionalmente independientes dada la clase Y



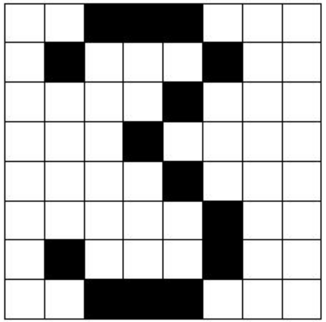
- Para cada X_i, especificar $\mathrm{P}(X_i \mid Y)$

- Función de decisión:

$$
\begin{aligned}
y^* = h_{NB}(\mathbf{x}) &= \arg\max_y P(y) P(x_1, \ldots, x_n \mid y) \\
&= \arg\max_y P(y) \prod_i P(x_i | y)
\end{aligned}
$$

Ejemplo: reconocimiento de dígitos

- Input: pixel grids



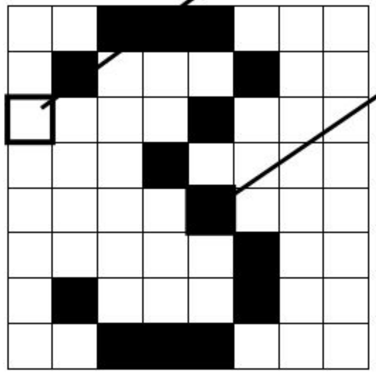
- Output: a digit 0-9

Pregunta: ¿cuán realista es la hipótesis del clasificador naïve Bayes en este ejemplo?

Ejemplo: reconocimiento de dígitos

$P(Y)$

1	0.1
2	0.1
3	0.1
4	0.1
5	0.1
6	0.1
7	0.1
8	0.1
9	0.1
0	0.1



$P(F_{3,1} = on|Y)$

1	0.01
2	0.05
3	0.05
4	0.30
5	0.80
6	0.90
7	0.05
8	0.60
9	0.50
0	0.80

$P(F_{5,5} = on|Y)$

1	0.05
2	0.01
3	0.90
4	0.80
5	0.90
6	0.90
7	0.25
8	0.85
9	0.60
0	0.80

Estimación de parámetros por MV

- Dado un conjunto de datos, obtener Count(A=a, B=b) , es decir, el número de ejemplos en donde A=a y B=b.
- MV para naïve Bayes sobre variables discretas:
  - Prior:

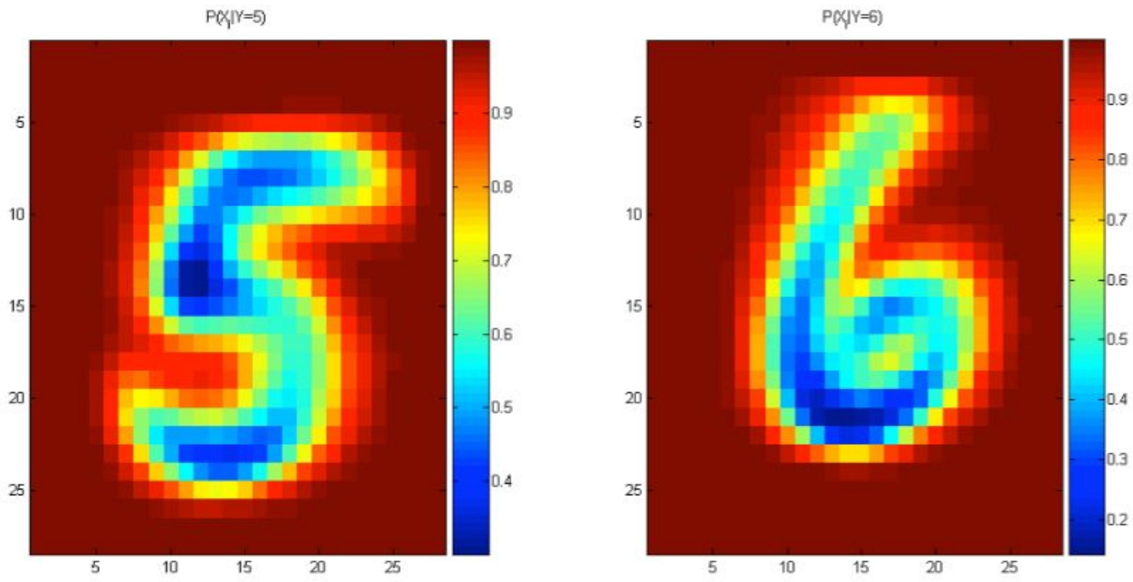
$$P(Y=y) = \frac{Count(Y=y)}{\sum_{y'} Count(Y=y')}$$

  - Distribución condicionales (observación):

$$P(X_i=x|Y=y) = \frac{Count(X_i=x, Y=y)}{\sum_{x'} Count(X_i=x', Y=y)}$$

Ejemplo: estimación de parámetros por MV

- El entrenamiento consiste en promediar los ejemplos para cada clase



MAP estimation for NB

- Given dataset
  - Count(A=a,B=b) $\leftarrow$ number of examples where A=a and B=b
- MAP estimation for discrete NB, simply:
  - Prior:

$$P(Y=y) = \frac{Count(Y=y)}{\sum_{y'} Count(Y=y')}$$

  - Observation distribution:

$$P(X_i=x|Y=y) = \frac{Count(X_i=x, Y=y) \textbf{ + a}}{\sum_{x'} Count(X_i=x', Y=y) \textbf{ + |X\_i|*a}}$$

- Called "smoothing". Corresponds to Dirichlet prior!

Estimación de parámetros por MAP

- Dado un conjunto de datos, obtener Count(A=a, B=b) , es decir, el número de ejemplos en donde A=a y B=b.
- MV para naïve Bayes sobre variables discretas:
  - Prior:

$$P(Y=y) = \frac{Count(Y=y)}{\sum_{y'} Count(Y=y')}$$

  - Distribución condicionales (observación):

$$P(X_i=x|Y=y) = \frac{Count(X_i=x, Y=y)+a}{\sum_{x'} Count(X_i=x', Y=y)+a|X_i|}$$

Estimación de parámetros por MAP

- Dado un conjunto de datos, obtener Count(A=a, B=b) , es decir, el número de ejemplos en donde A=a y B=b.
- MV para naïve Bayes sobre variables discretas:
  - Prior:

$$P(Y = y) = \frac{Count(Y = y)}{\sum_{y'} Count(Y = y')}$$

  - Distribución condicionales (observación):

smoothing
(Dirichlet prior)

$$P(X_i = x | Y = y) = \frac{Count(X_i = x, Y = y) + a}{\sum_{x'} Count(X_i = x', Y = y) + a|X_i|}$$