

Introducción al aprendizaje automático

...

#1. Introducción al aprendizaje automático.

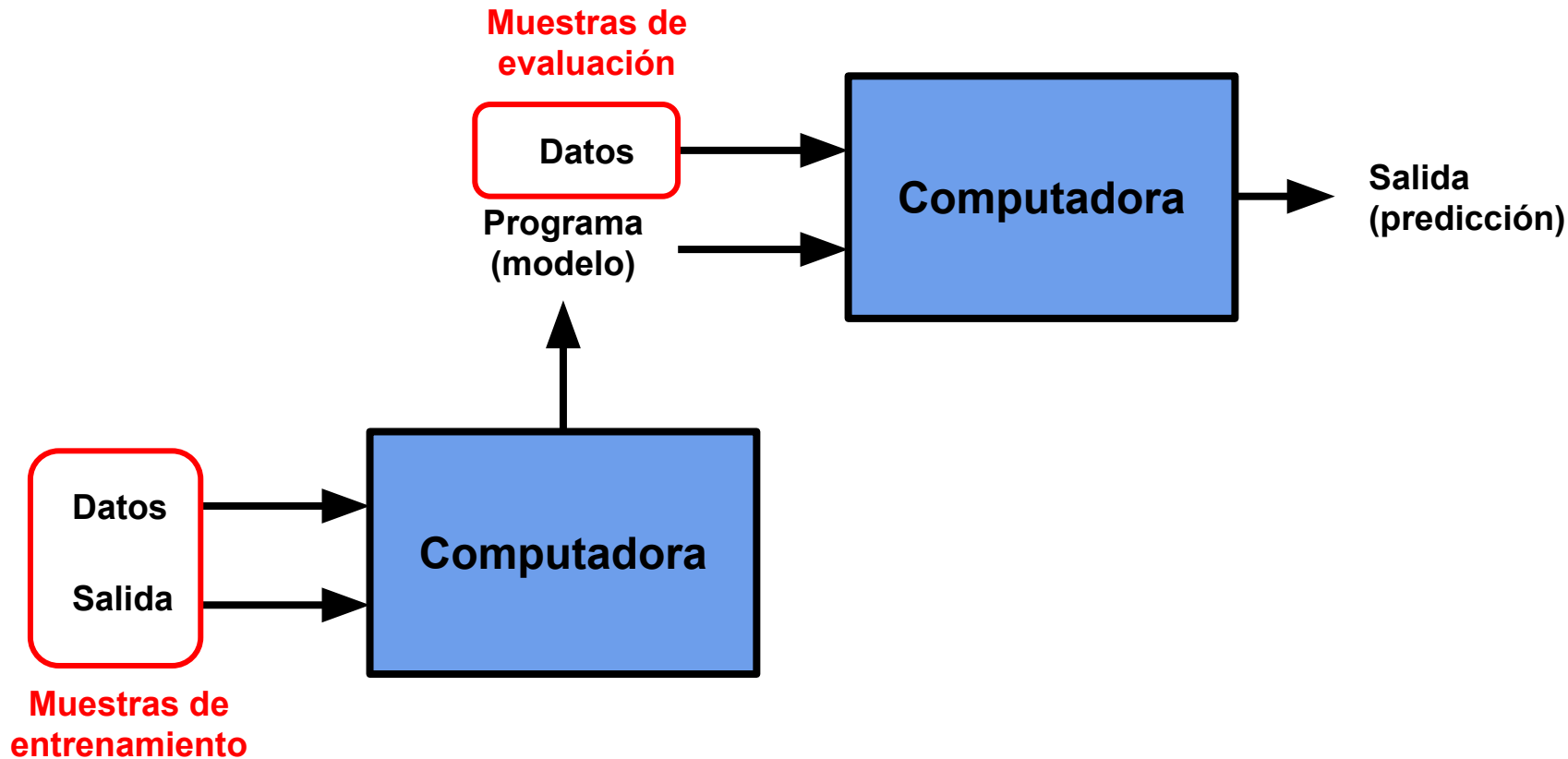
Programación tradicional



Aprendizaje automático



Aprendizaje automático: entrenamiento vs. evaluación

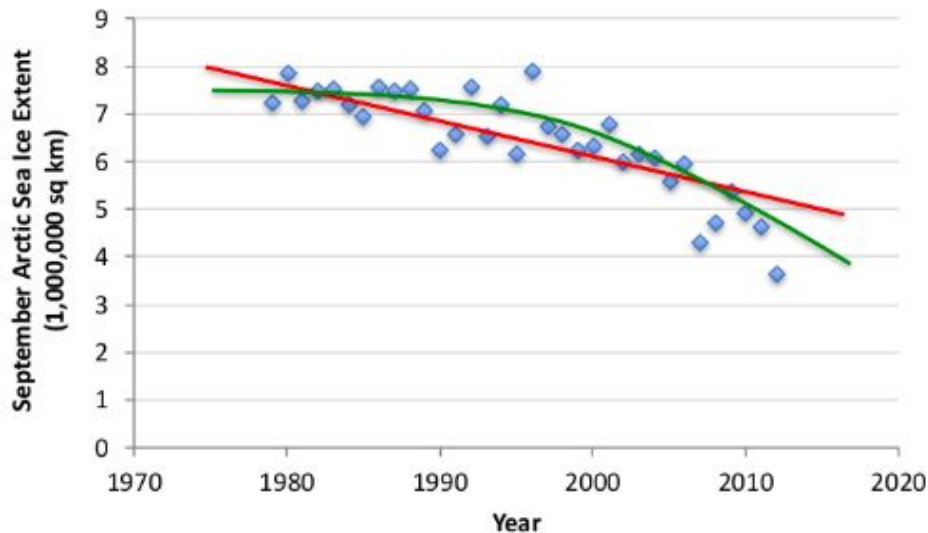


Tipos de aprendizaje

- **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

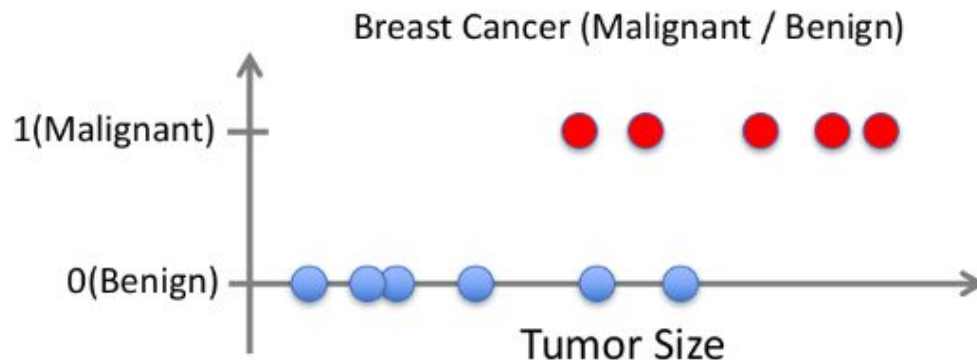
Aprendizaje supervisado: regresión

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



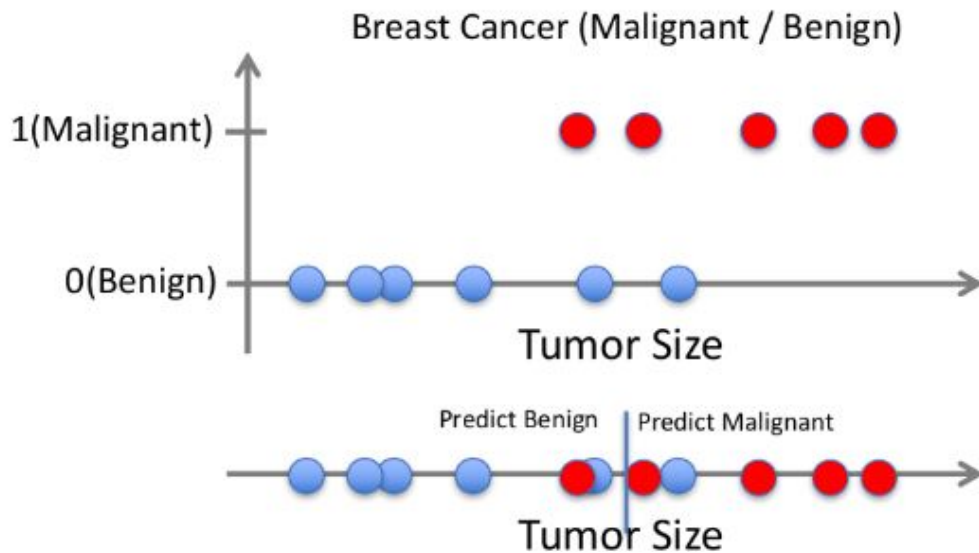
Aprendizaje supervisado: clasificación

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



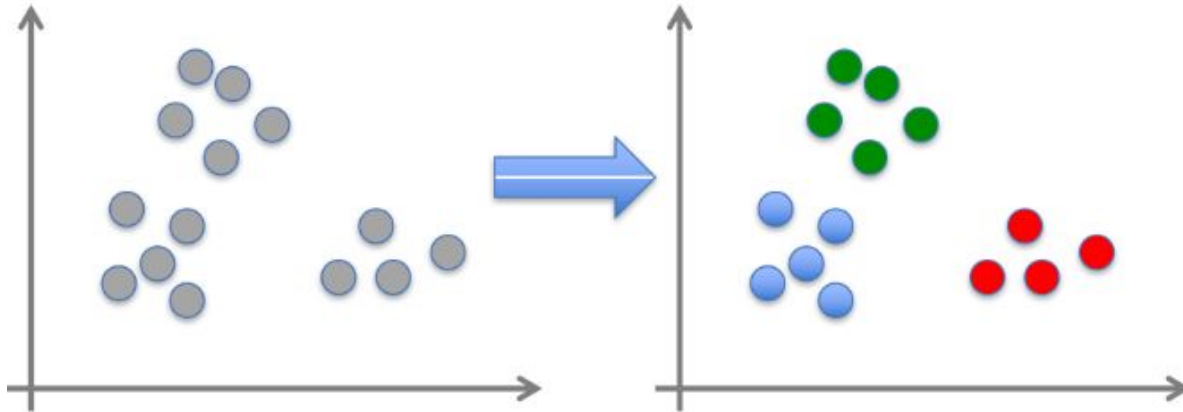
Aprendizaje supervisado: clasificación

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



Aprendizaje no supervisado

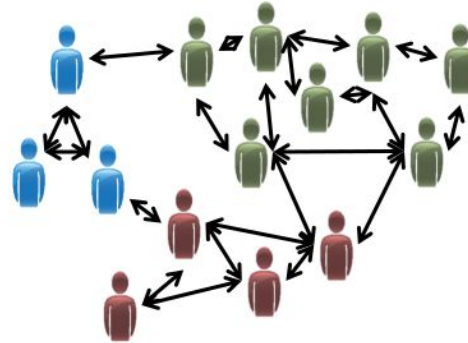
- Datos x_1, x_2, \dots, x_n
- Aprender la estructura interna de los datos
 - p.ej. *clustering*



Aprendizaje no supervisado



Organize computing clusters



Social network analysis



Market segmentation



Astronomical data analysis

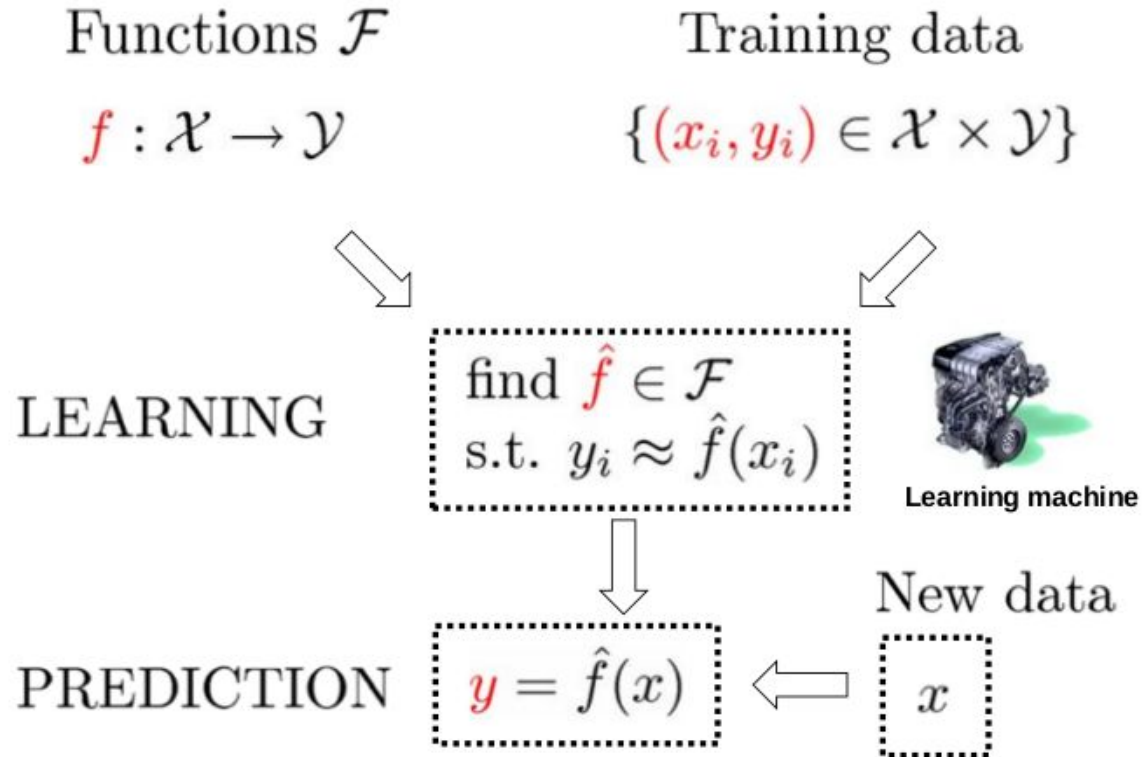
Aprendizaje por refuerzo

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

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



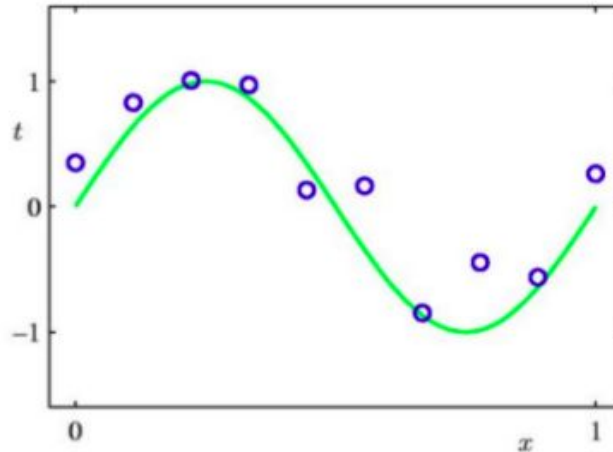
Regresión

Regresión

- Disponemos de N pares de entrenamiento (observaciones)

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

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

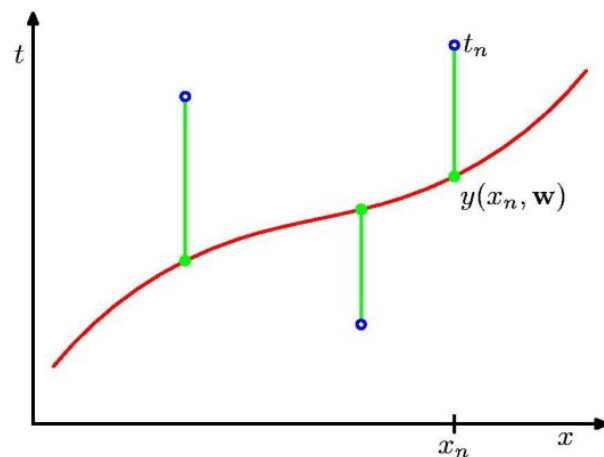
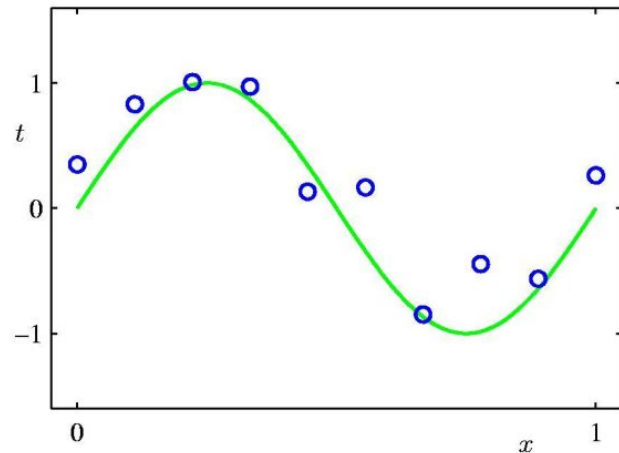


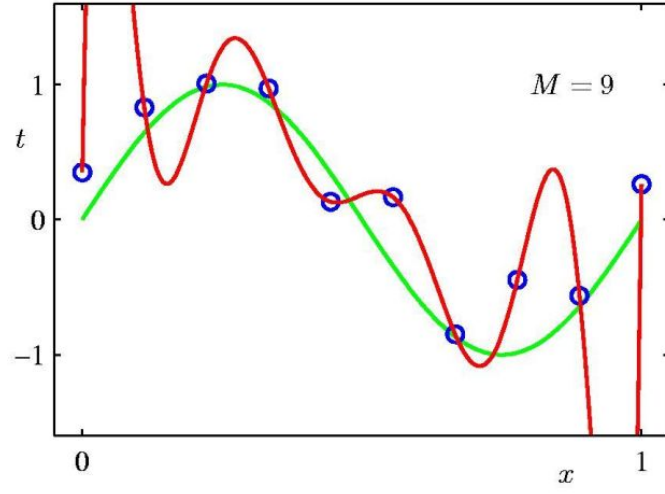
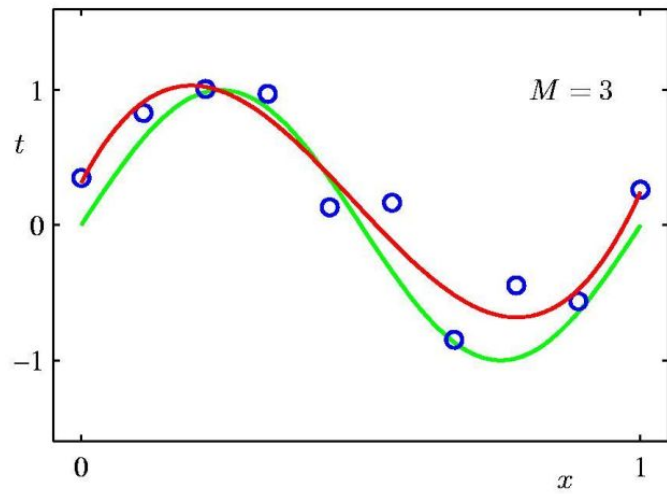
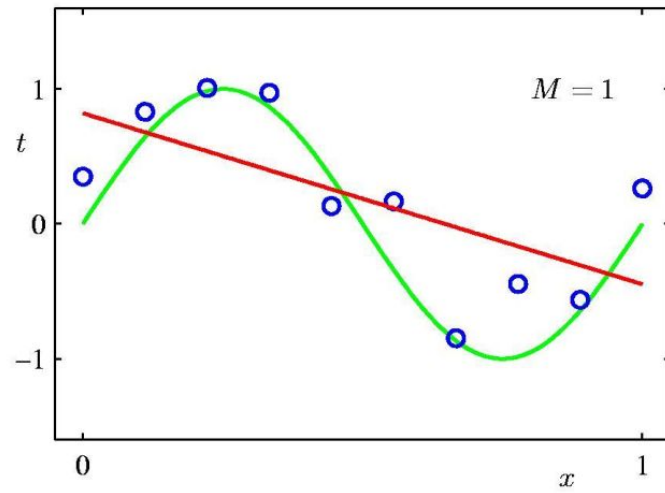
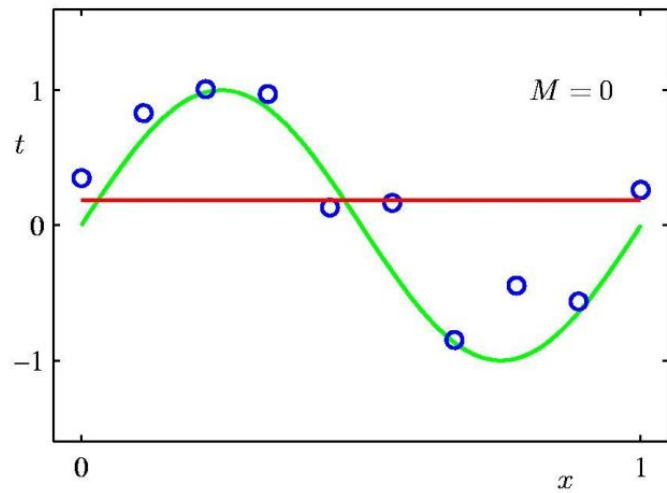
Regresión polinomial

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

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

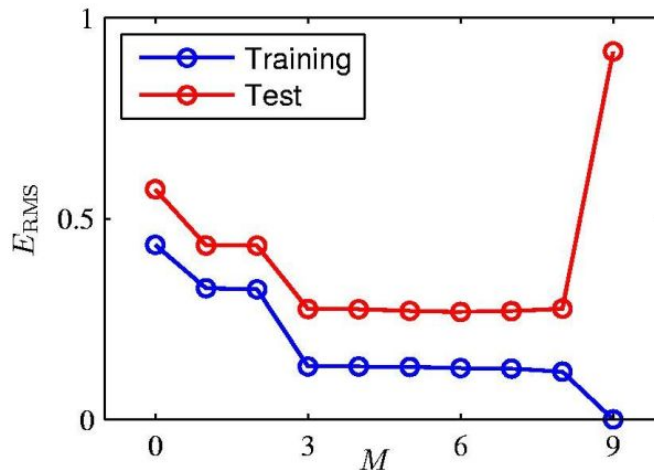
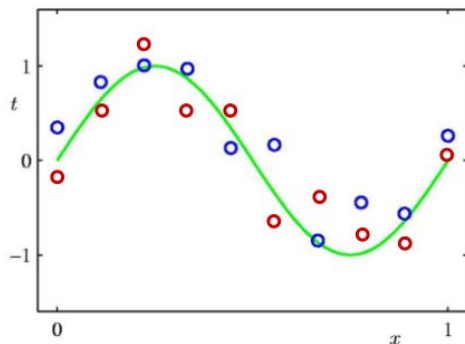
$$y(x, \mathbf{w}) = w_0 + w_1x + w_2x^2 + \dots + w_Mx^M = \sum_{j=0}^M w_jx^j$$





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



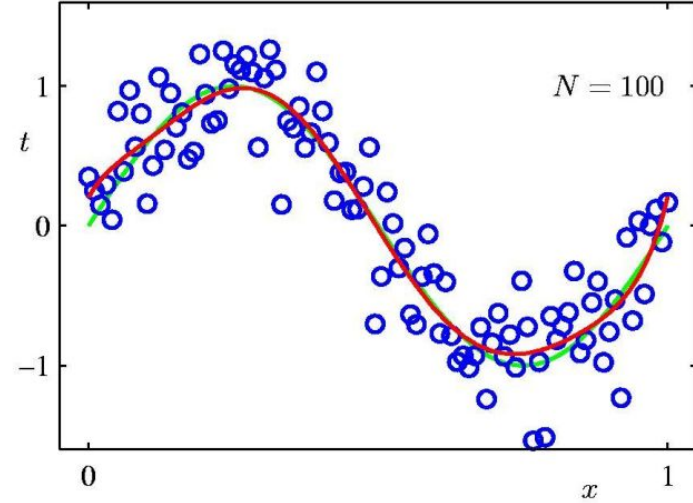
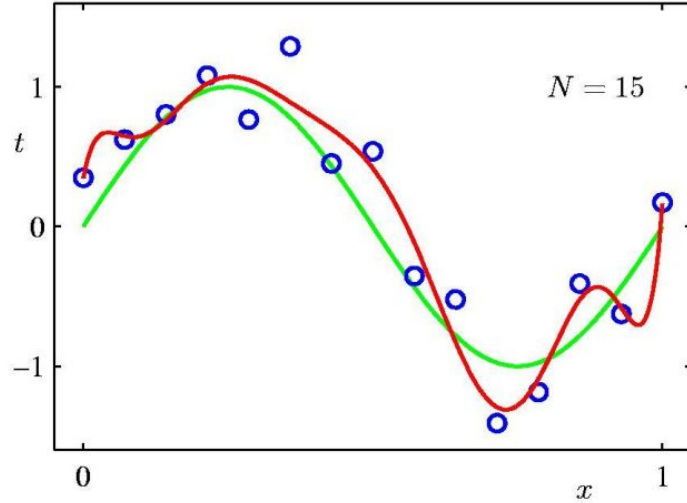
Root-Mean-Square (RMS) Error: $E_{\text{RMS}} = \sqrt{2E(\mathbf{w}^*)/N}$

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 nuevos (no vistos en entrenamiento)
- Podemos decir que un modelo generaliza, si puede explicar los datos empleando una complejidad acotada

Prevenir el sobreajuste (I)

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



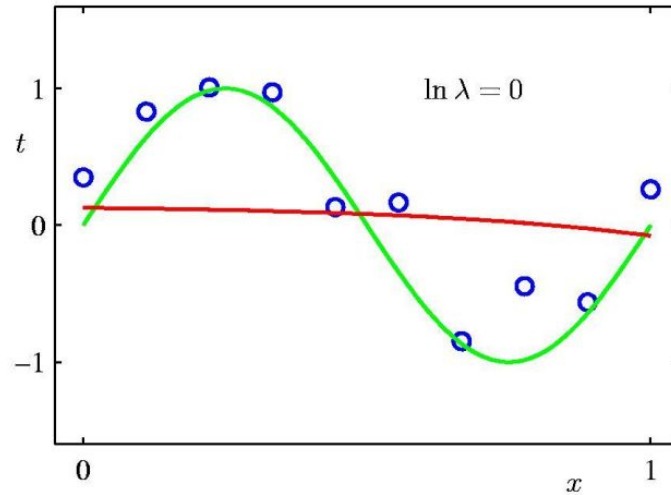
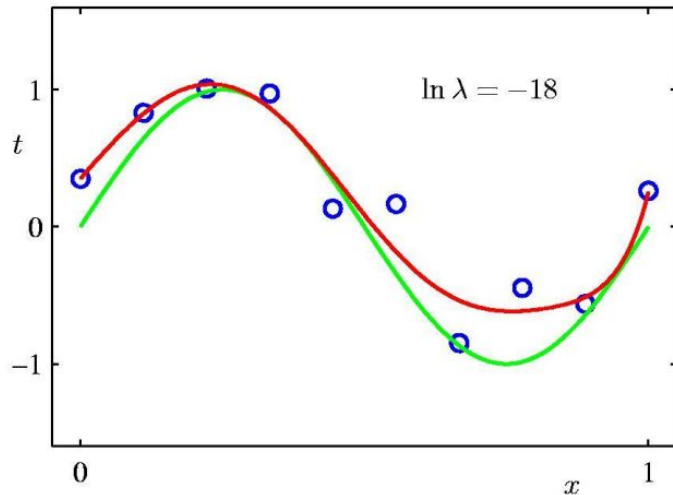
Prevenir el sobreajuste (II)

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

Prevenir el sobreajuste (II)

- Regularización: penalizar valores grandes de los coeficientes

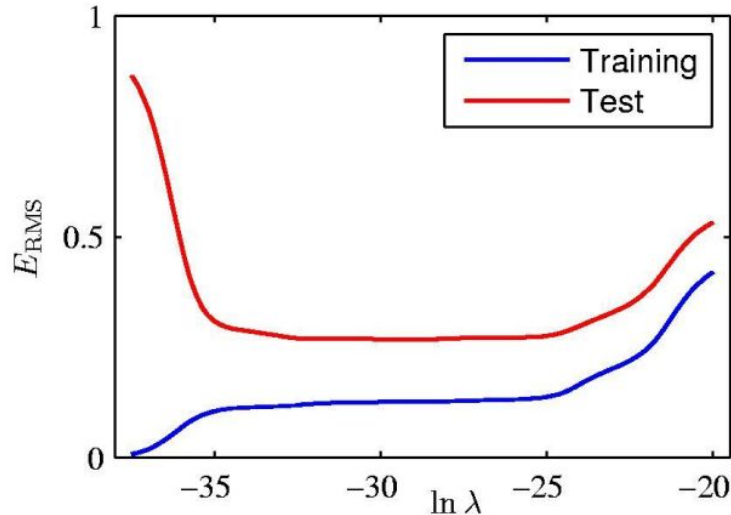
$$\tilde{E}(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2$$



Prevenir el sobreajuste (II)

- Regularización: penalizar valores grandes de los coeficientes

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



**Término de
regularización
(ridge)**

λ = hiperparámetro

Prevenir el sobreajuste (II)

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

Regresión polinomial como regresión lineal

$$x \mapsto \mathbf{z} = \begin{pmatrix} x \\ x^2 \\ \vdots \\ x^M \end{pmatrix} \quad \mathbf{w} = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_M \end{pmatrix}$$

$$\begin{aligned} y(x; \mathbf{w}) &= w_0 + w_1x + w_2x^2 + \cdots + w_Mx^M \\ &= w_0 + \sum_{j=1}^M w_j x^j = w_0 + \sum_{j=1}^M w_j z_j \\ &= w_0 + \mathbf{w}^T \mathbf{z} \end{aligned}$$

$$\mathbf{a}^T \mathbf{b} \equiv \langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i=1}^D a_i b_i$$

Regresión polinomial como regresión lineal

$$x \mapsto \mathbf{z} = \begin{pmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^M \end{pmatrix} \quad \mathbf{w} = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_M \end{pmatrix}$$

$$\begin{aligned} y(x; \mathbf{w}) &= w_0 + w_1x + w_2x^2 + \cdots + w_Mx^M \\ &= \sum_{j=0}^M w_j x^j \\ &= \mathbf{w}^T \mathbf{z} \end{aligned}$$

Regresión lineal: solución de mínimos cuadrados

- Dataset: $\{(x_1, t_1), \dots, (x_N, t_N)\} \mapsto \{(\mathbf{z}_1, t_1), \dots, (\mathbf{z}_N, t_N)\}$
- Función de costo: $E(\mathbf{W}) = \frac{1}{2} \sum_{i=1}^N (y(x_i; \mathbf{w}) - t_i)^2 = \frac{1}{2} \sum_{i=1}^N (\mathbf{w}^T \mathbf{z}_i - t_i)^2$

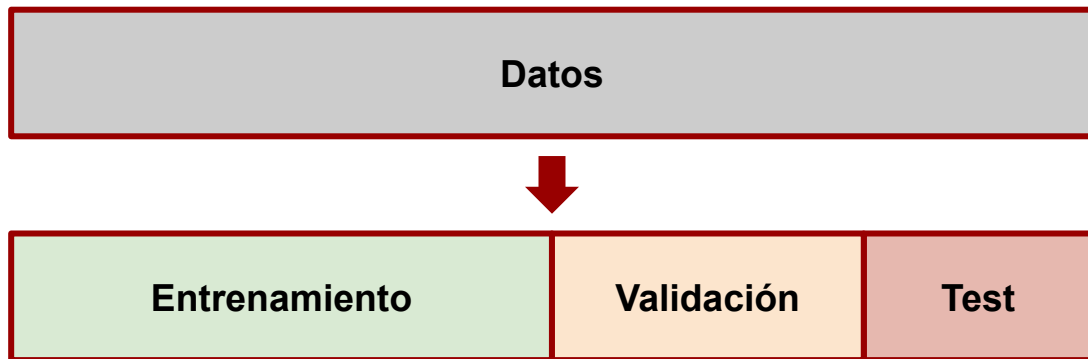
$$\mathbf{Z} = \begin{pmatrix} - & \mathbf{z}_1^T & - \\ & \vdots & \\ - & \mathbf{z}_N^T & - \end{pmatrix} \in \mathbb{R}^{N \times M} \quad \mathbf{y} = \begin{pmatrix} t_1 \\ \vdots \\ t_N \end{pmatrix} \in \mathbb{R}^{N \times 1} \quad \|\mathbf{w}\|^2 = \mathbf{w}^T \mathbf{w}$$

$$\begin{aligned} E(\mathbf{w}) &= (\mathbf{Z}\mathbf{w} - \mathbf{y})^T (\mathbf{Z}\mathbf{w} - \mathbf{y}) & \rightarrow & \mathbf{w}^* = (\mathbf{Z}^T \mathbf{Z})^{-1} \mathbf{Z}^T \mathbf{y} \\ E(\mathbf{w}) &= (\mathbf{Z}\mathbf{w} - \mathbf{y})^T (\mathbf{Z}\mathbf{w} - \mathbf{y}) + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w} & \rightarrow & \mathbf{w}^* = (\mathbf{Z}^T \mathbf{Z} + \lambda \mathbf{I})^{-1} \mathbf{Z}^T \mathbf{y} \end{aligned}$$

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 final de la performance del modelo entrenado (y con hiperparámetros elegidos adecuadamente)



Clasificación

Clasificación binaria

- Disponemos de N pares de entrenamiento (observaciones)

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

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

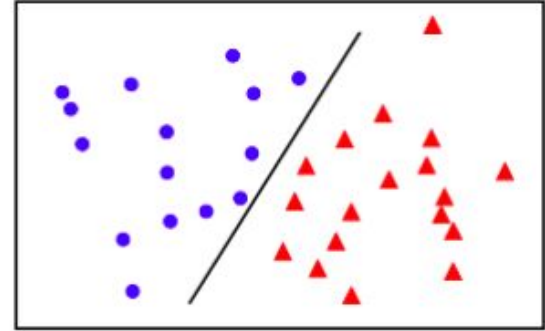
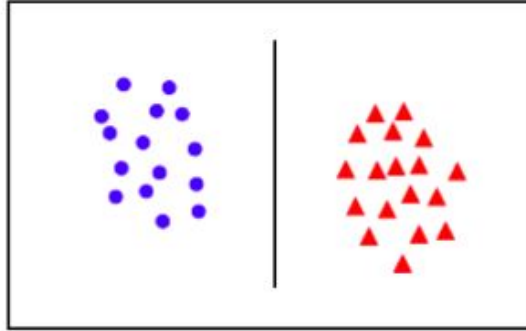
- Aprender una $f(x)$ tal que

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

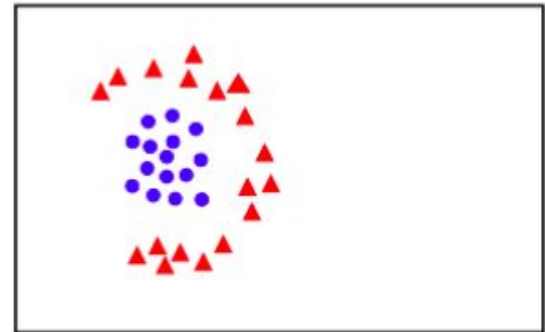
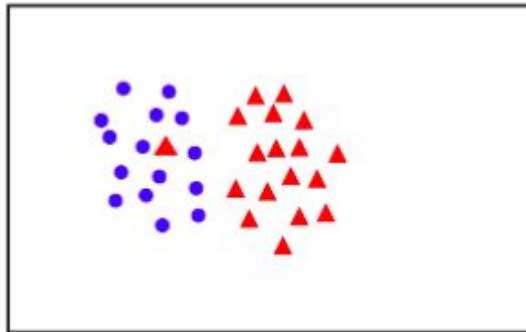
es decir: $y_i f(x_i) > 0$ para una clasificación correcta.

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

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

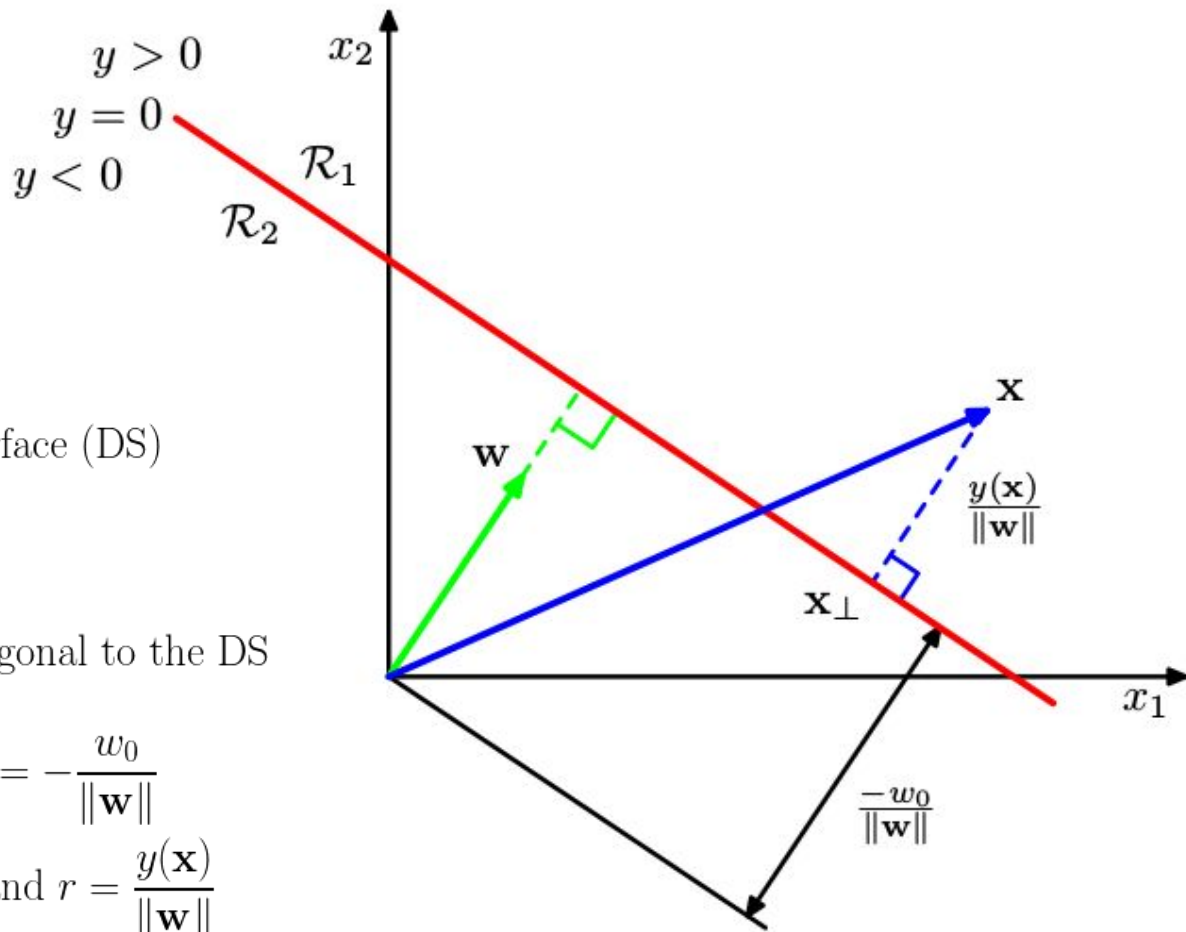
- Función de predicción **lineal**

$$f(x) = w^T x + w_0$$

- Función de decisión

$$g(z) = \text{sign}(z)$$

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



$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$$

$\mathbf{x}_A, \mathbf{x}_B$: points on the decision surface (DS)

$$y(\mathbf{x}_A) = y(\mathbf{x}_B) = 0$$

$\mathbf{w}^T(\mathbf{x}_A - \mathbf{x}_B) = 0 \implies \mathbf{w}$ is orthogonal to the DS

$$\text{if } \mathbf{x} \text{ on the DS, } y(\mathbf{x}) = 0 \rightarrow \frac{\mathbf{w}^T \mathbf{x}}{\|\mathbf{w}\|} = -\frac{w_0}{\|\mathbf{w}\|}$$

$$\text{if } \mathbf{x} = \mathbf{x}_\perp + r \frac{\mathbf{w}}{\|\mathbf{w}\|} \rightarrow y(\mathbf{x}_\perp) = 0 \text{ and } r = \frac{y(\mathbf{x})}{\|\mathbf{w}\|}$$

El algoritmo del "perceptrón"

- Propuesto por Roseblatt 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 $(x_1, y_1), (x_2, y_2) \dots$
- 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 $y_i' = \text{sign}(w^T x_i + w_0)$
 - 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 $(x_1, y_1), (x_2, y_2) \dots$
- 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 $y_i' = \text{sign}(w^T x_i + w_0)$
 - 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 $(x_1, y_1), (x_2, y_2) \dots$
- 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 $y_i' = \text{sign}(w^T x_i)$
 - 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:

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

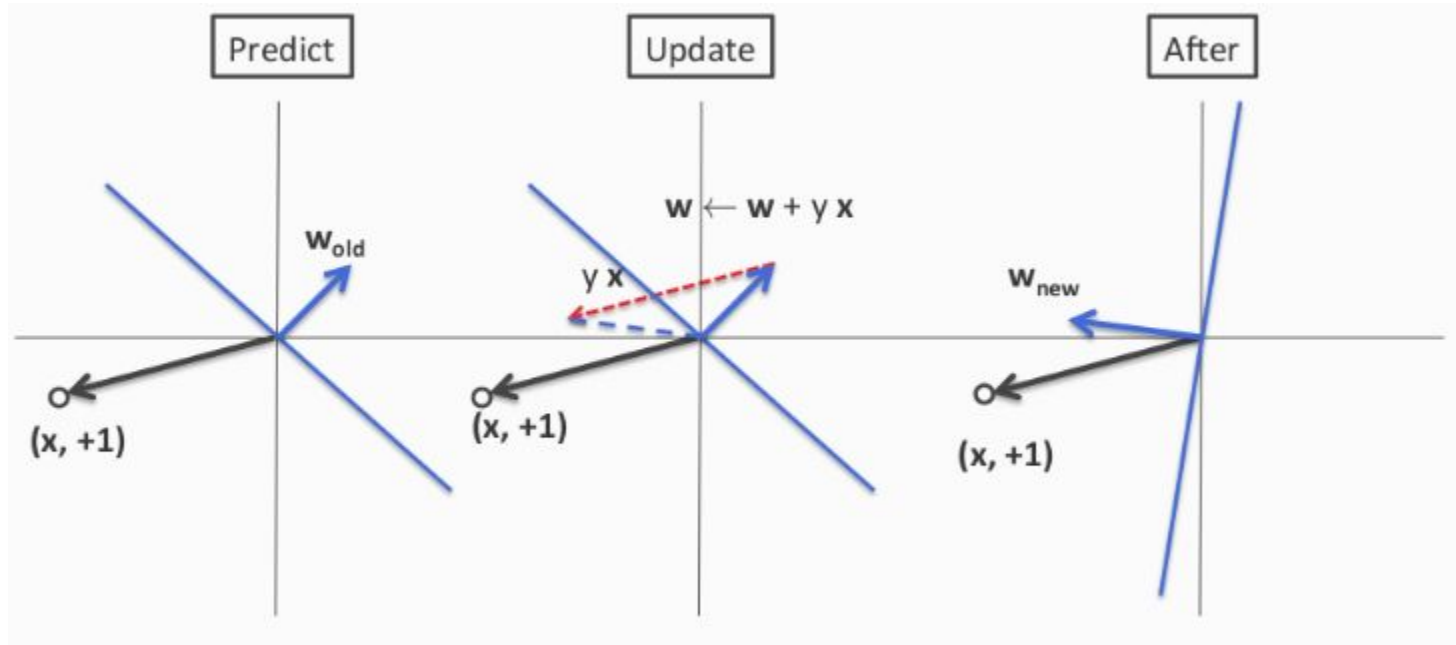
Error en negativos:

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

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

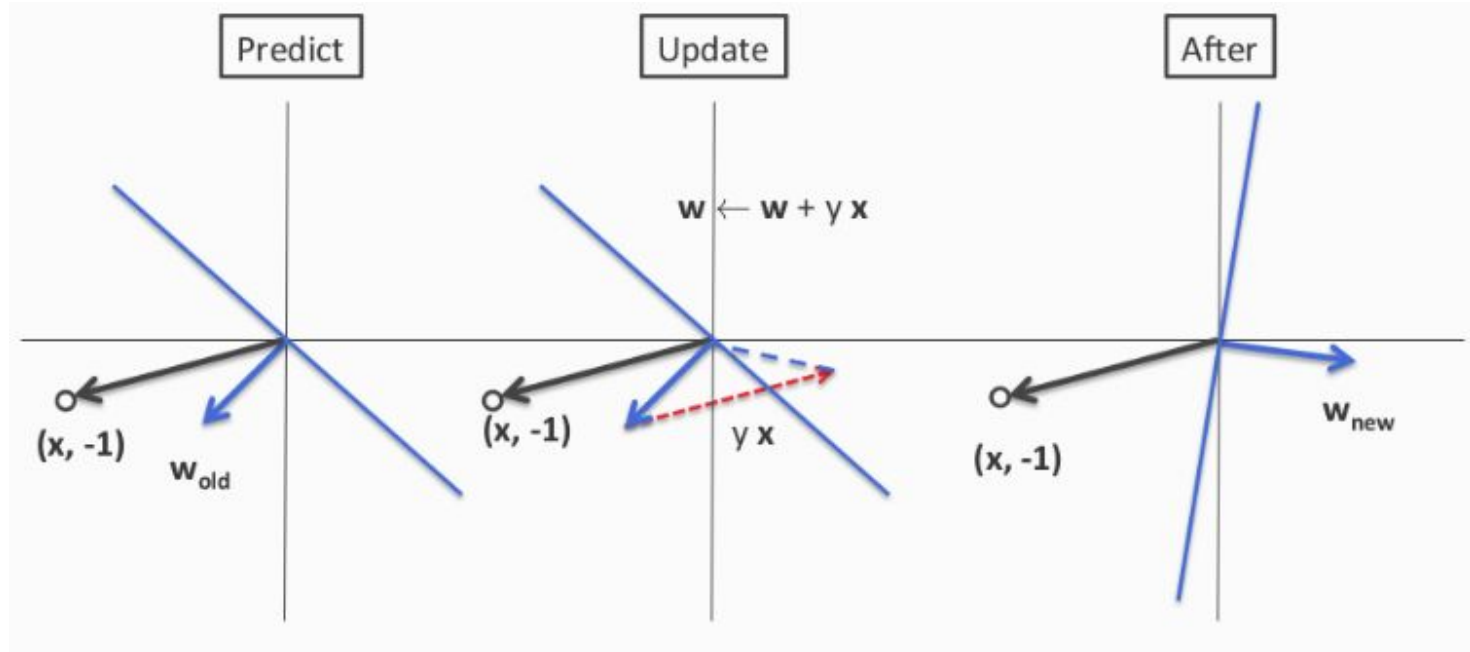
Dinámica de actualización

Error en ejemplo **positivo**:



Dinámica de actualización

Error en ejemplo **negativo**:



El algoritmo "estándar"

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

1. Initialize $\mathbf{w} = \mathbf{0} \in \mathbb{R}^n$

2. For epoch = 1 ... T:

T is a **hyper-parameter** to the algorithm

1. Shuffle the data

2. For each training example $(\mathbf{x}_i, y_i) \in D$:

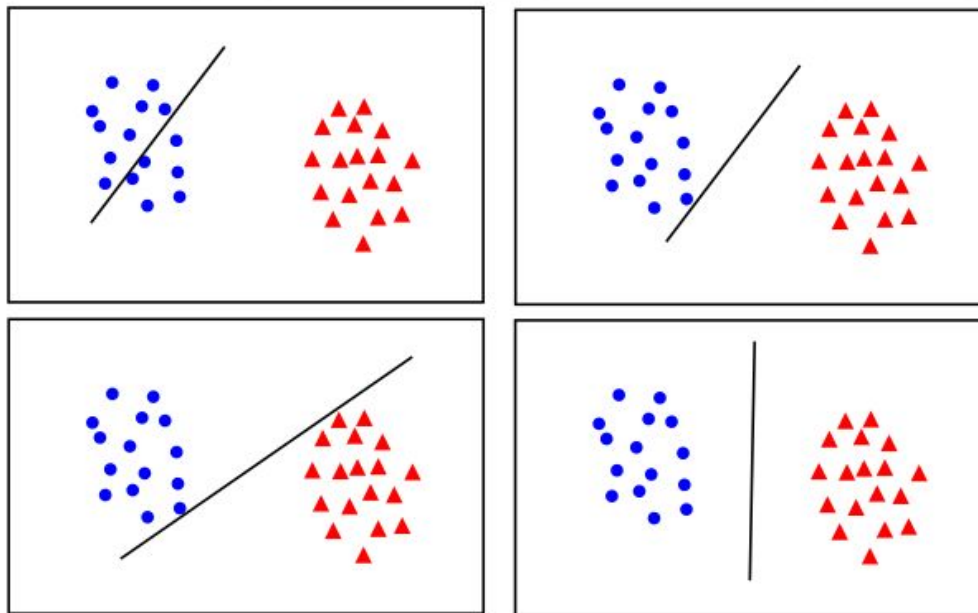
- If $y_i \mathbf{w}^T \mathbf{x}_i \leq 0$, update $\mathbf{w} \leftarrow \mathbf{w} + r y_i \mathbf{x}_i$

3. Return \mathbf{w}

Another way of writing that
there is an error

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

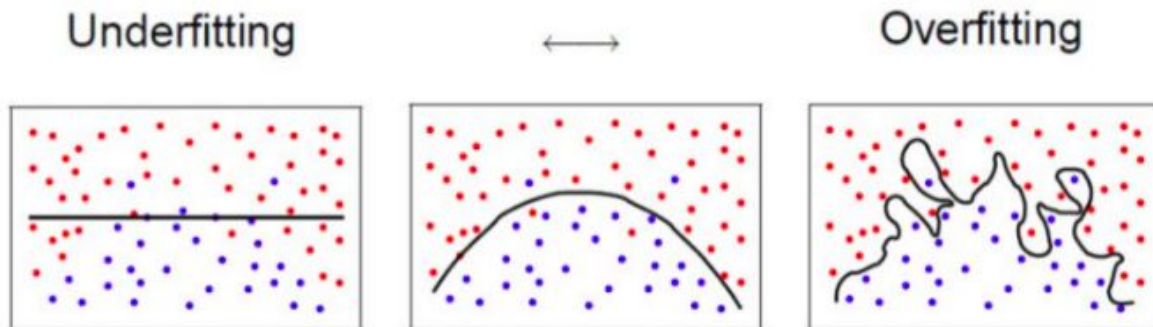
¿Cuál es el mejor w ?



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

Generalización en clasificación

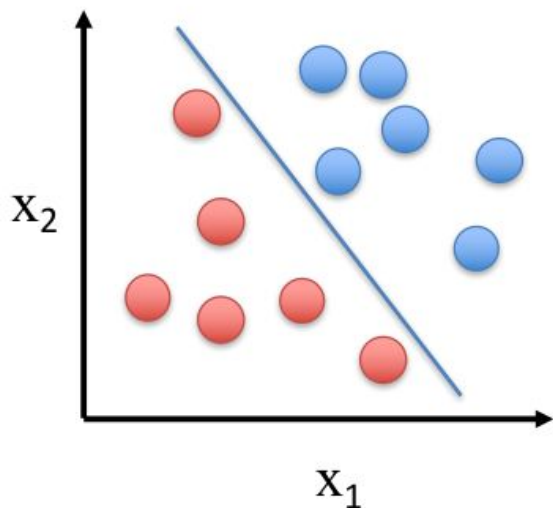
- Complejidad del modelo \Leftrightarrow complejidad de la frontera de decisión



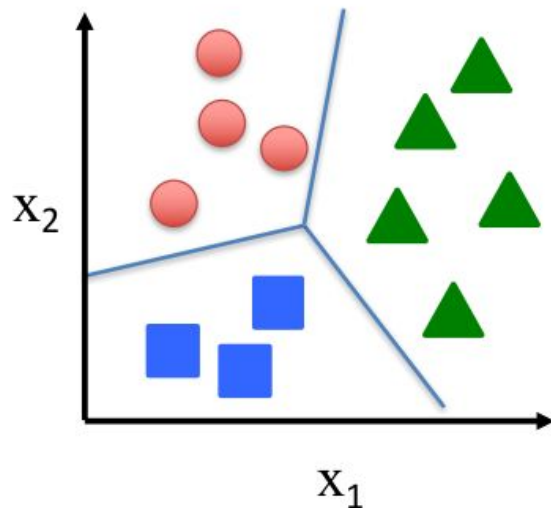
Problemas multiclase

Multi-Class Classification

Binary classification:



Multi-class classification:



What is multiclass classification?

- An input can belong to one of K classes
- **Training data**: Input associated with class label (a number from 1 to K)
- **Prediction**: Given a new input, predict the class label

Each input belongs to exactly one class. Not more, not less.

- Otherwise, the problem is not multiclass classification
- If an input can be assigned multiple labels (think tags for emails rather than folders), it is called *multi-label classification*

Binary to multiclass

- Can we use a binary classifier to construct a multiclass classifier?
 - Decompose the prediction into multiple binary decisions
- How to decompose?
 - One-vs-all
 - All-vs-all
 - Error correcting codes

1. One-vs-all classification

- **Assumption:** Each class individually separable from *all* the others

- **Learning:** Given a dataset $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}$ $\begin{matrix} \mathbf{x} \in \mathbb{R}^n \\ \mathbf{y} \in \{1, 2, \dots, K\} \end{matrix}$
 - Decompose into K binary classification tasks
 - For class k , construct a binary classification task as:
 - **Positive examples:** Elements of D with label k
 - **Negative examples:** All other elements of D
 - Train K binary classifiers $\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_K$ using any learning algorithm we have seen

1. One-vs-all classification

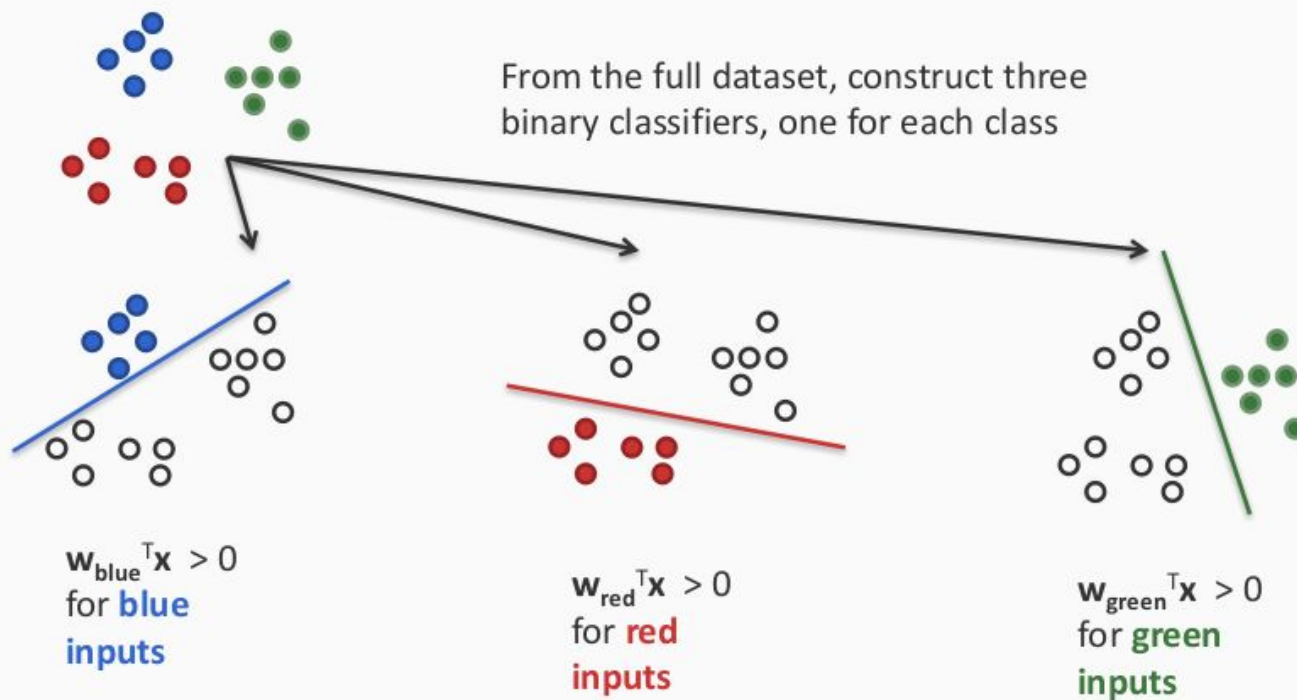
- **Assumption:** Each class individually separable from *all* the others

- **Learning:** Given a dataset $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}$ $\begin{matrix} \mathbf{x} \in \mathbb{R}^n \\ \mathbf{y} \in \{1, 2, \dots, K\} \end{matrix}$
 - Train K binary classifiers $\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_K$ using any learning algorithm we have seen

- **Prediction:** “Winner Takes All”

$$\operatorname{argmax}_i \mathbf{w}_i^\top \mathbf{x}$$

Visualizing One-vs-all



2. All-vs-all classification

Sometimes called one-vs-one

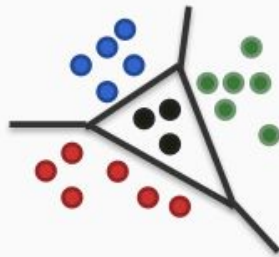
- **Assumption:** *Every* pair of classes is separable
- **Learning:** Given a dataset $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}$, $\mathbf{x} \in \mathbb{R}^n$
 $\mathbf{y} \in \{1, 2, \dots, K\}$
 - For every pair of labels (j , k), create a binary classifier with:
 - **Positive examples:** All examples with label j
 - **Negative examples:** All examples with label k
 - Train $\binom{K}{2} = \frac{K(K-1)}{2}$ classifiers to separate every pair of labels from each other

2. All-vs-all classification

Sometimes called one-vs-one

- **Assumption:** Every pair of classes is separable
- **Learning:** Given a dataset $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}$, $\begin{matrix} \mathbf{x} \in \mathbb{R}^n \\ \mathbf{y} \in \{1, 2, \dots, K\} \end{matrix}$
 - Train $\binom{K}{2} = \frac{K(K-1)}{2}$ classifiers to separate every pair of labels from each other
- **Prediction:** More complex, each label get K-1 votes
 - How to combine the votes? Many methods
 - Majority: Pick the label with maximum votes
 - Organize a tournament between the labels

All-vs-all classification



- Every pair of labels is linearly separable here
 - When a pair of labels is considered, all others are ignored
- Problems
 1. $O(K^2)$ weight vectors to train and store
 2. Size of training set for a pair of labels could be very small, leading to overfitting of the binary classifiers
 3. Prediction is often ad-hoc and might be unstable
 - Eg: What if two classes get the same number of votes? For a tournament, what is the sequence in which the labels compete?