

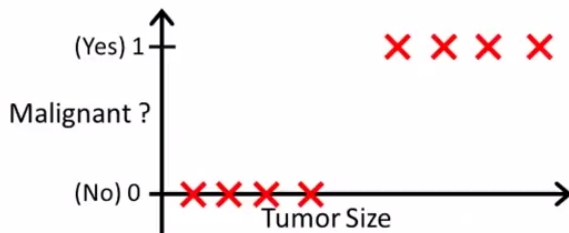


**Inteligencia Artificial**

## Problemas de clasificación.

- Email: Spam / Not Spam
- Tumores: Malignos / Benignos
- Clientes: Malos pagadores / Buenos Pagadores

**¿Es posible aplicar regresion lineal para resolver estos problemas?**



# Regresión logística.

Hipótesis previa.

$$h_{\theta}(x) = \Theta^T x$$

Ahora:

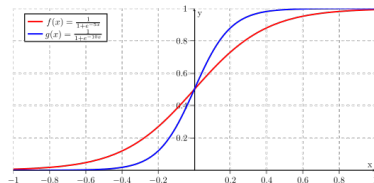
$$h_{\theta}(x) = g(\Theta^T x)$$

Donde:

$$g(z) = \frac{1}{1 + e^{-z}}$$

Por ende nos queda:

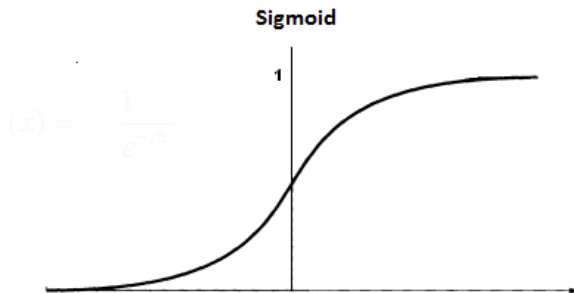
$$g(z) = \frac{1}{1 + e^{-\Theta^T x}}$$



- $g(z)$  se denomina **función logística** o **función sigmoidea**
- $h(x)$  vamos a decir que da la probabilidad de que **y = 0**

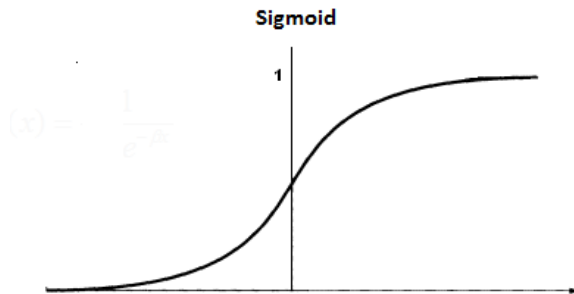
## Frontera de decisión.

- ¿Cuando  $y = 1$ ? ¿Cuando  $y = 0$ ?



## Frontera de decisión.

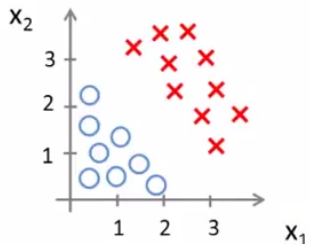
- ¿Cuando  $y = 1$ ? ¿Cuando  $y = 0$ ?



$$y=1 \Rightarrow h_{\theta}(x) \geq 0.5 \Rightarrow \Theta^T x \geq 0$$

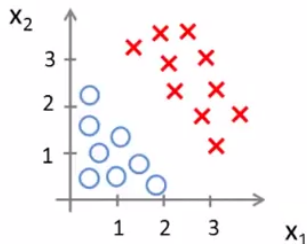
$$y=0 \Rightarrow h_{\theta}(x) < 0.5 \Rightarrow \Theta^T x < 0$$

## Frontera de decisión.



$$h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

## Frontera de decisión.

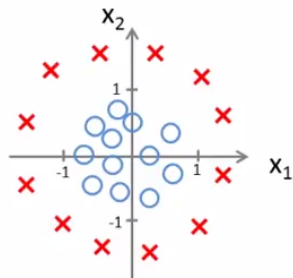


$$h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

La frontera de decisión es la región donde  $h(x) = 0.5$ .

Es la frontera donde la predicción cambia de valor.

## Frontera de decisión no lineal.



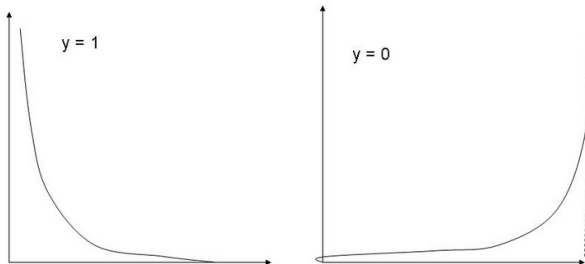
$$h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$



**Función de costo.**

- Si usamos error cuadrático como función de costo,  $J(w)$  será no convexa
- En cambio vamos a usar:

$$\text{Cost}(h_{\theta}(x), y) = \begin{cases} -\log(h_{\theta}(x)) & \text{if } y = 1 \\ -\log(1 - h_{\theta}(x)) & \text{if } y = 0 \end{cases}$$



**Función de costo.**

- Si juntamos todo nos queda:

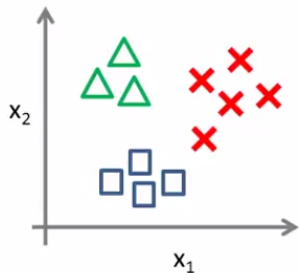


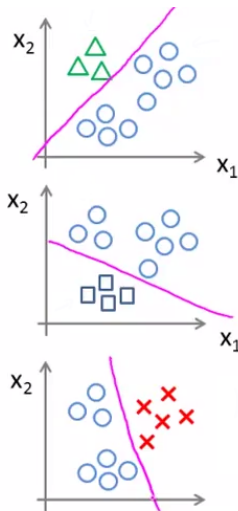

$$\begin{aligned} J(\theta) &= \frac{1}{m} \sum_{i=1}^m \text{Cost}(h_{\theta}(x^{(i)}), y^{(i)}) \\ &= -\frac{1}{m} \left[ \sum_{i=1}^m y^{(i)} \log h_{\theta}(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_{\theta}(x^{(i)})) \right] \end{aligned}$$

- La regla de actualización para GD nos queda:

$$\theta_j := \theta_j - \alpha \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

## Clasificación multiclase.

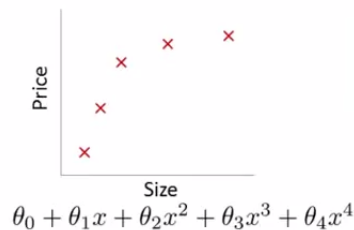
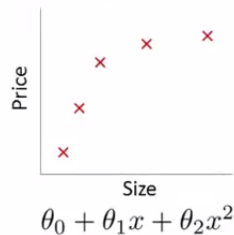
One-vs-all (one-vs-rest):

Class 1: Class 2: Class 3: 

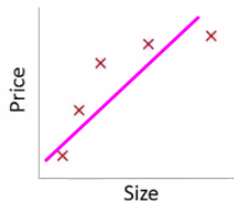
**Clasificación multiclase.**

- Entrenamos un clasificador binario por cada clase
- Para hacer una predicción:
  - Obtenemos la probabilidad de  $y = 1$  por cada predictor
  - Elegimos la clase con la probabilidad más alta
- La normalización de los features sigue siendo necesaria

# Sobre-entrenamiento.

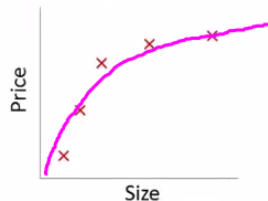


# Sobre-entrenamiento.

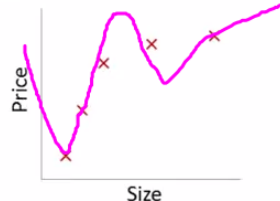


$$\theta_0 + \theta_1 x$$

Underfit



$$\theta_0 + \theta_1 x + \theta_2 x^2$$

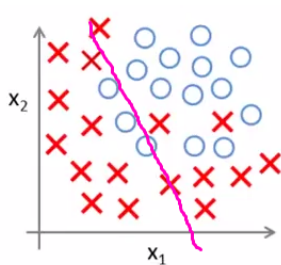


$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

Overfit

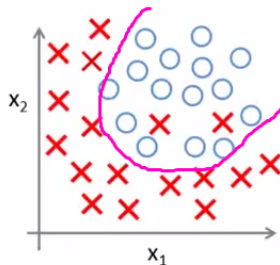
Decimos que el modelo está **sobreentrenado** cuando la hipótesis se ajusta muy bien al dataset de entrenamiento, pero falla en **generalizar** nuevos ejemplos

# Sobre-entrenamiento.

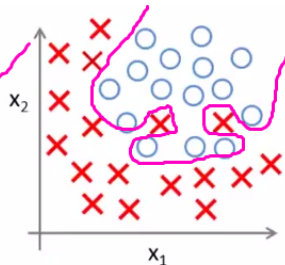


$$h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

(  $g$  = sigmoid function)



$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2)$$



$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \dots)$$

# Sobre-entrenamiento.

**¿Cómo podemos preveer el sobreentrenamiento?**

1. Reducir el número de features:

- Selección manual o Algoritmo de selección
- Puede ser que todos los features contribuyan en la predicción

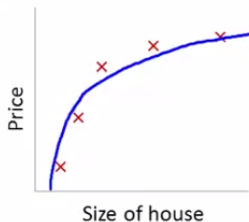
2. Regularización:

- Mantener todos los features pero reducir el valor de los parámetros
- Funciona bien cuando todos los features contribuyen en la predicción

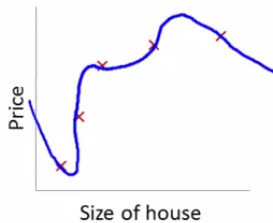


# Regularización.

## Intuition



$$\theta_0 + \theta_1 x + \theta_2 x^2$$



$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

Suppose we penalize and make  $\theta_3, \theta_4$  really small.

$$\min_{\theta} \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

## Regularización.

$$J(\theta) = \frac{1}{2m} \left[ \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n \theta_j^2 \right]$$

- **lambda** es el parámetro de regularización. Establece un tradeoff entre "complejidad de la hipótesis" y "mantener bajo el error".

¿Cómo funciona?

## Regularización.

$$J(\theta) = \frac{1}{2m} \left[ \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n \theta_j^2 \right]$$

- **lambda** es el parámetro de regularización. Establece un tradeoff entre "complejidad de la hipótesis" y "mantener bajo el error".

¿Cómo funciona?

- A valores bajos de lambda, la hipótesis es compleja y el error va a ser bajo (sobreentrenamiento)
- A valores altos de lambda, la hipótesis es simple y el error va a ser alto (bajo entrenamiento)

## Regularización.

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)} - y^{(i)})x_0^{(i)}$$

$$\theta_j := \theta_j - \frac{1}{m} \left[ \alpha \sum_{i=1}^m (h_{\theta}(x^{(i)} - y^{(i)})x_j^{(i)} + \lambda \theta_j \right]$$

$$\theta_j := \theta_j (1 - \alpha \frac{\lambda}{m}) - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})x_j^{(i)}$$

**Bibliografía y enlaces útiles.**

- Russell S., Norvig P.: Artificial Intelligence: A modern Approach. Third Edition. Chapter 18.
- Curso de Machine Learning dictado por Andrew Ng - <https://www.coursera.org/learn/machine-learning/>