



**Universitat**  
de les Illes Balears

# Machine Learning

## **Lesson 3: Supervised Learning - Classification**

Linear Models (LMS, Logistic Regression, Perceptron)

# Definition

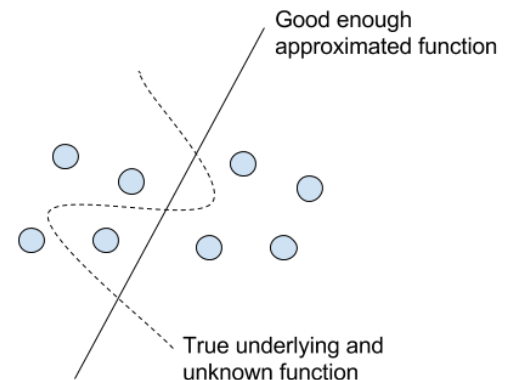
Se dice de un programa que aprende de la experiencia **E** con respecto a la tarea **T** y una medida de rendimiento **P** si el rendimiento **P** sobre **T** aumenta con la experiencia **E**

T. Mitchel, 1997

Campo de estudio que da a los ordenadores la capacidad de aprender sin ser programados de forma específica

A. Samuel, 1959

Podemos entender un algoritmo de ML como **un algoritmo que cambia su estado interno** para encontrar el mayor **mapeo** entre una variable de salida y sus características de entrada



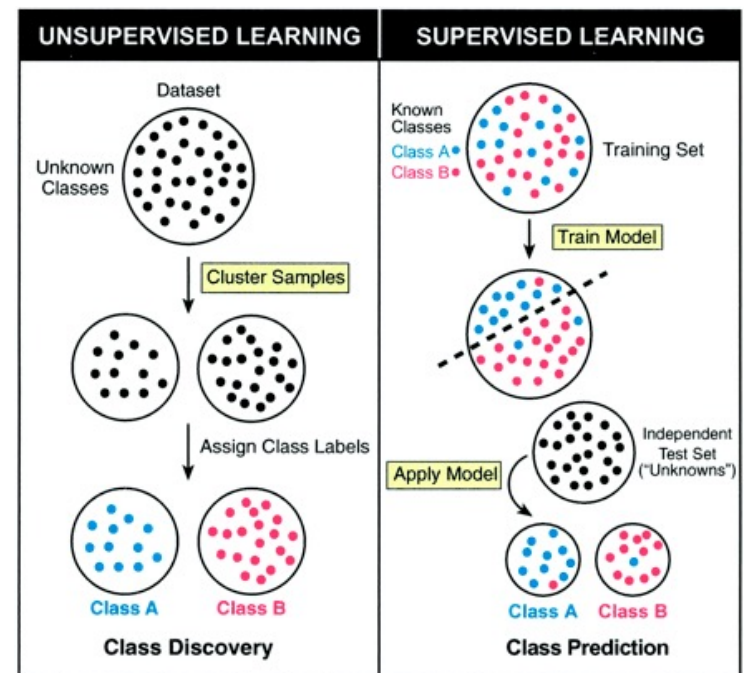
# Learning main types

## Supervised

Data is provided (through a "master") as pairs of cases-labels, indicating whether or not they belong to the type of association to be learned (classification, regression).  
Practical case: prediction

## Unsupervised

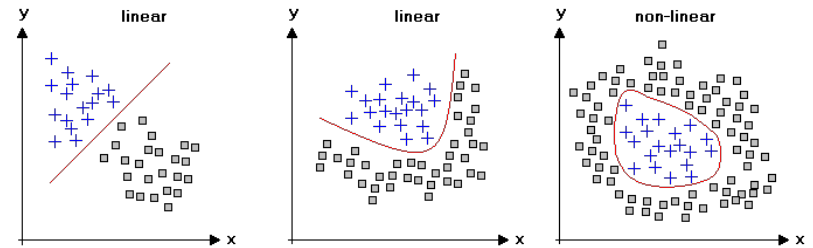
The algorithm must use other funds to obtain a "feedback" that tells him whether he does it well or not, since he does not have a "master" (clustering). Practical case: segmentation



# Task types

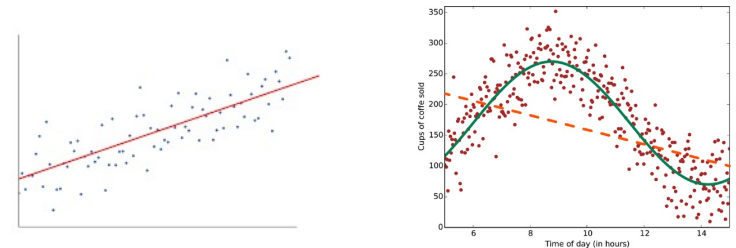
## Classification

Classification algorithms are used when the desired output is a discrete label.

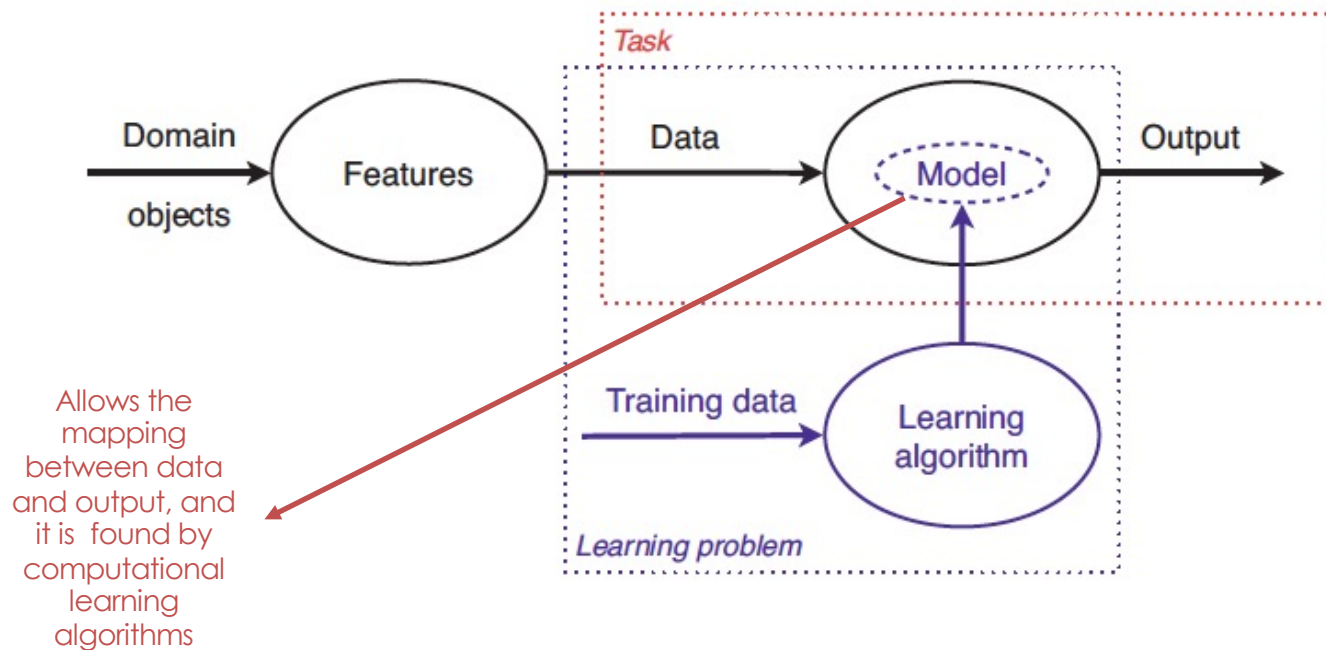


## Regression

Approximate a continuous function (predicting outputs that are continuous)



# Task: machine learning model



# Summary of Linear Regression

## Linear regression

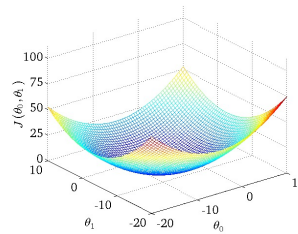
### Hypothesis:

the model is linear

$$h_{\theta}(x) = \sum_{j=0}^n \theta_j x_j$$

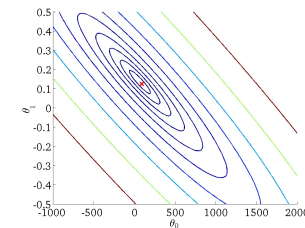
## Gradient descent

**Idea:** to make  $h_{\theta}$  close to  $y$  by means minimizing a cost function

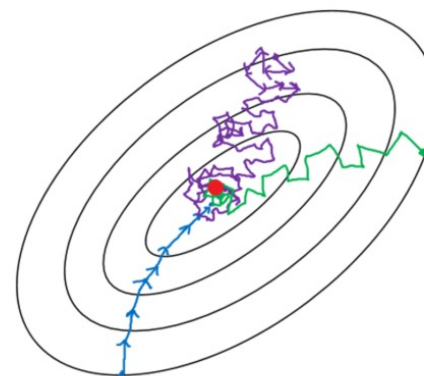
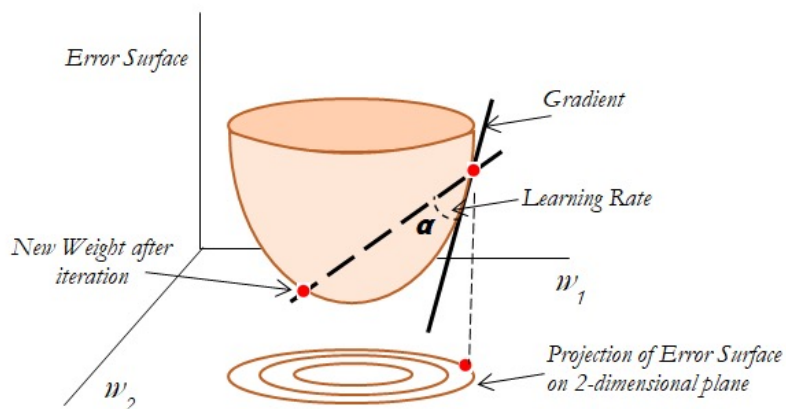


## LMS

Using the gradient to find the minimum (batch & incremental)



## Stochastic Gradient Descent with Batch size “1”



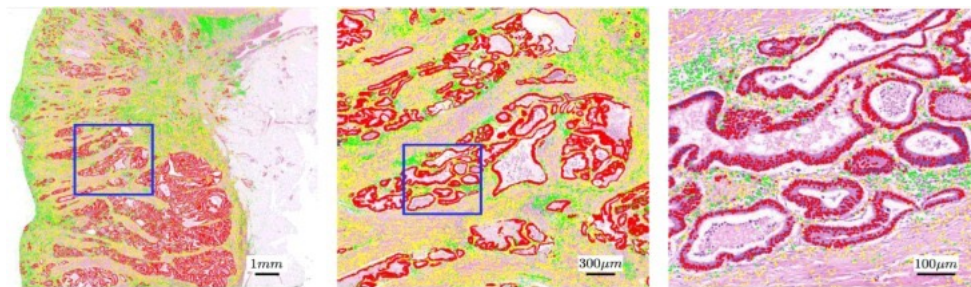
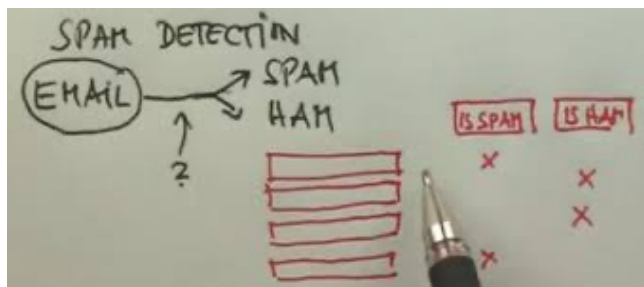
- Batch gradient descent
- Mini-batch gradient Descent
- Stochastic gradient descent

### Gradient descent

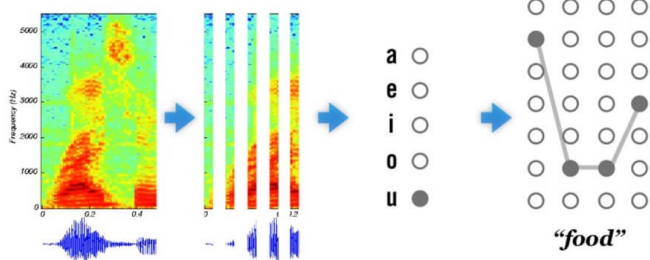
minimize an objective function  $J(\theta)$  by updating the parameters in the opposite direction of the gradient of the objective function.

The learning rate determines the size of the steps taken to reach the minimum

- Batch gradient descent: all training observations utilized in each iteration
- SGD: one observation per iteration
- Mini batch gradient descent: size of about 50 training observations for each iteration



## SIRI TEARDOWN



# Classification

In machine learning, **classification** is the problem of identifying to which of a set of categories a new observation belongs, on the basis of a training set of data containing observations whose category membership is known.



# Binary classification

There are only two categories

$$y = \{0,1\},$$

where

0: negative class (-)

1: positive class (+)

**Task:** assign a class for a new input.

Email: Spam / Not Spam  
Fraudulent online transactions: Yes / No  
Tumor: Malignant / Benign

Dataset =  $\{(x^{(i)}, y^{(i)}); i = 1, \dots, m\}$

**Quiz**

$(x^{(i)}, y^{(i)})$  = Training example

$x^{(i)}$  = "input" variable (features),  $x \in \mathcal{X}$

$y^{(i)}$  = "output" variable (target),  $y \in \mathcal{Y}$

Size (feet <sup>2</sup> )	Number of bedrooms	Number of floors	Age of home (years)	Price (\$1000)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...

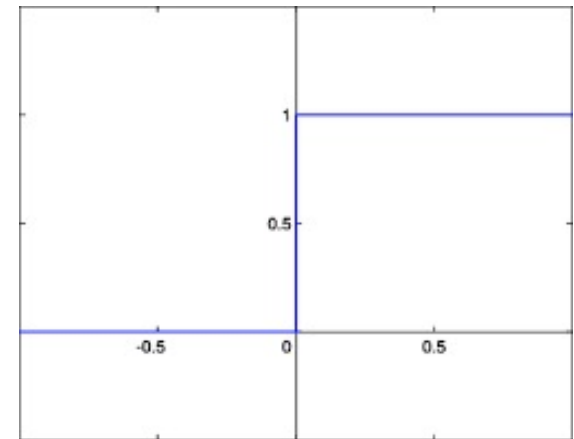
# Binary classification – Linear model

**Hypothesis:** the model is linear

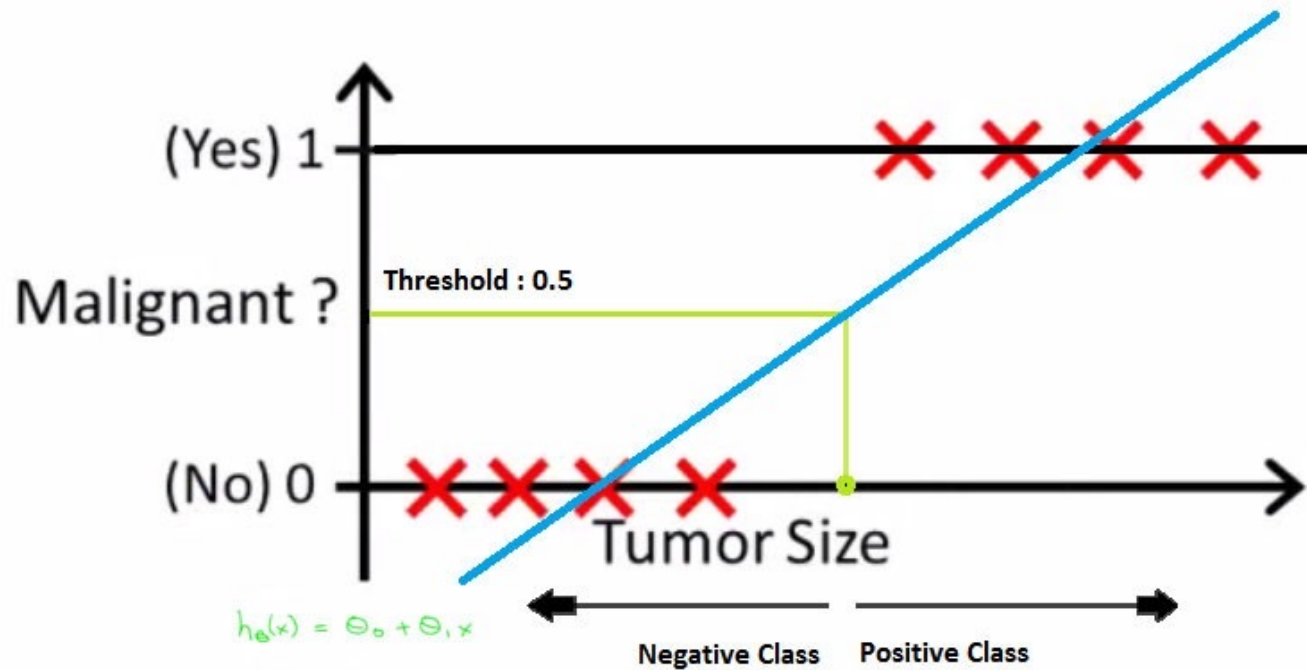
$$h_{\theta}(\mathbf{x}) = \sum_{j=0}^n \theta_j x_j$$

However... It does no sense for  $h_{\theta}$  to take values larger than 1 or smaller than 0, therefore we must redefine our hypothesis:

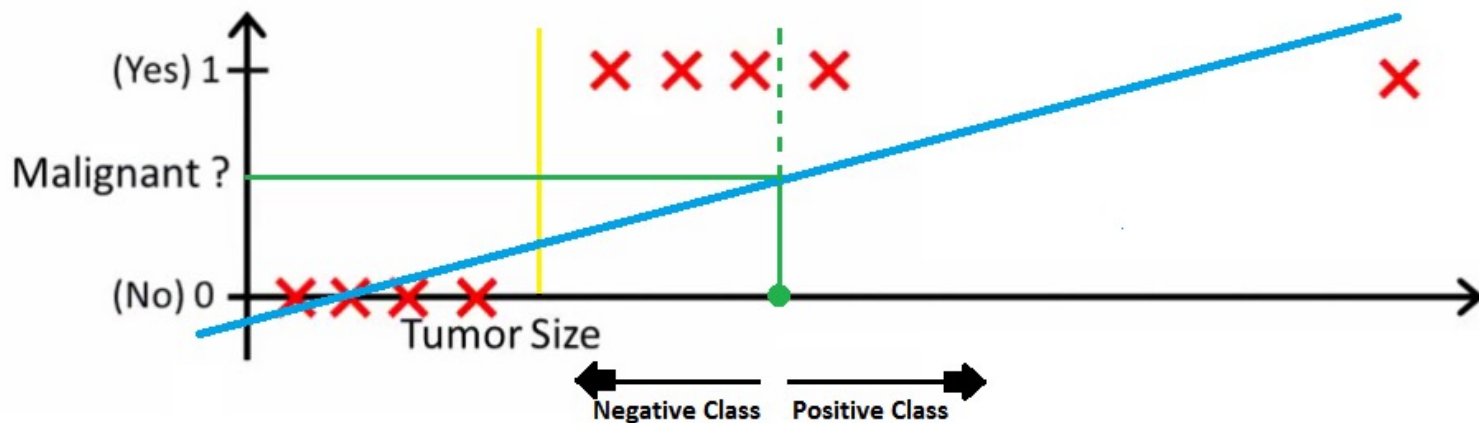
$$h_{\theta}(\mathbf{x}) = g\left(\sum_{j=0}^n \theta_j x_j\right) = \begin{cases} 1, & \sum_{j=0}^n \theta_j x_j \geq 0 \\ 0, & \text{otherwise} \end{cases}$$



# Binary classification – Linear model



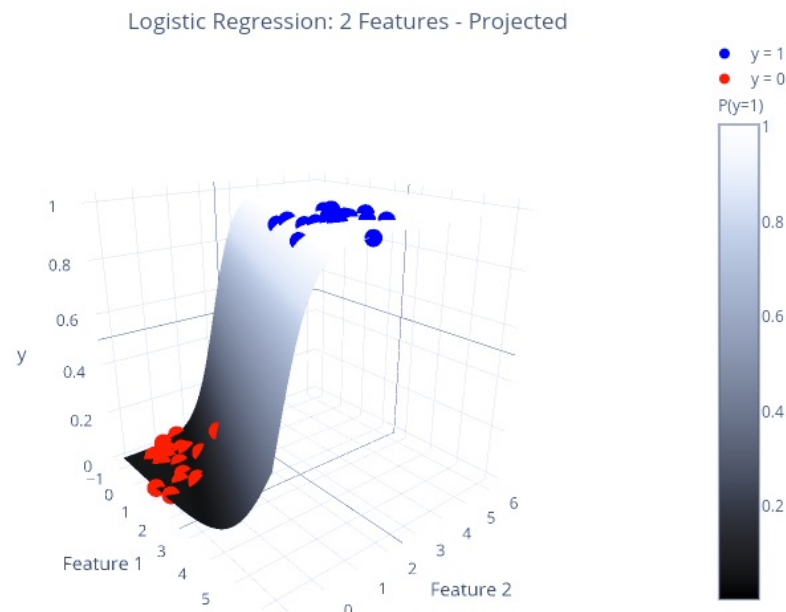
# Binary classification – Linear model



# Probabilistic approach

$h_{\theta}(x)$ , can be interpreted as the estimated probability that  $y = 1$  on input  $x$

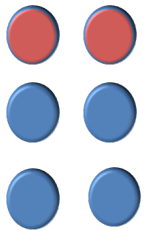
Example:  $h_{\theta}(\mathbf{x}) = 0.7$ .... there is a 70% that the email be spam.



# Probability vs. Odds

Probability: favorable cases over all cases

Odds: favorable cases over non-favorable cases



$$P(\text{redBall}) = \frac{2}{6} = 0.33$$

$$O(\text{redBall}) = \frac{2}{4} = 0.5$$

$$O(\text{redBall}) = \frac{P(\text{redBall})}{1 - P(\text{redBall})}$$

# Heuristic approach

$h_{\theta}(x)$ , can be interpreted as the estimated probability that  $y = 1$  on input  $x$

Example:  $h_{\theta}(\mathbf{x}) = 0.7$ .... there is a 70% that the email be spam.

There have been several efforts to adapt linear regression methods to a domain where the output is a probability value,  $(0,1)$ , instead of any real number  $(-\infty, \infty)$

In many cases, such efforts have focused on modeling this problem by mapping the range  $(0,1)$  to  $(-\infty, \infty)$  and then running the linear regression on these transformed values.

# Heuristic approach

$h_{\theta}(x)$ , can be interpreted as the estimated probability that  $y = 1$  on input  $x$

Example:  $h_{\theta}(\mathbf{x}) = 0.7$ .... there is a 70% that the email be spam.



# Logistic regression

We need a function similar to the threshold, but it has to be **continuous and derivable**. We redefine our hypothesis:

$$h_{\theta}(\mathbf{x}) = g\left(\sum_{j=0}^n \theta_j x_j\right) \\ = \frac{1}{1 + e^{-\sum_{j=0}^n \theta_j x_j}}$$

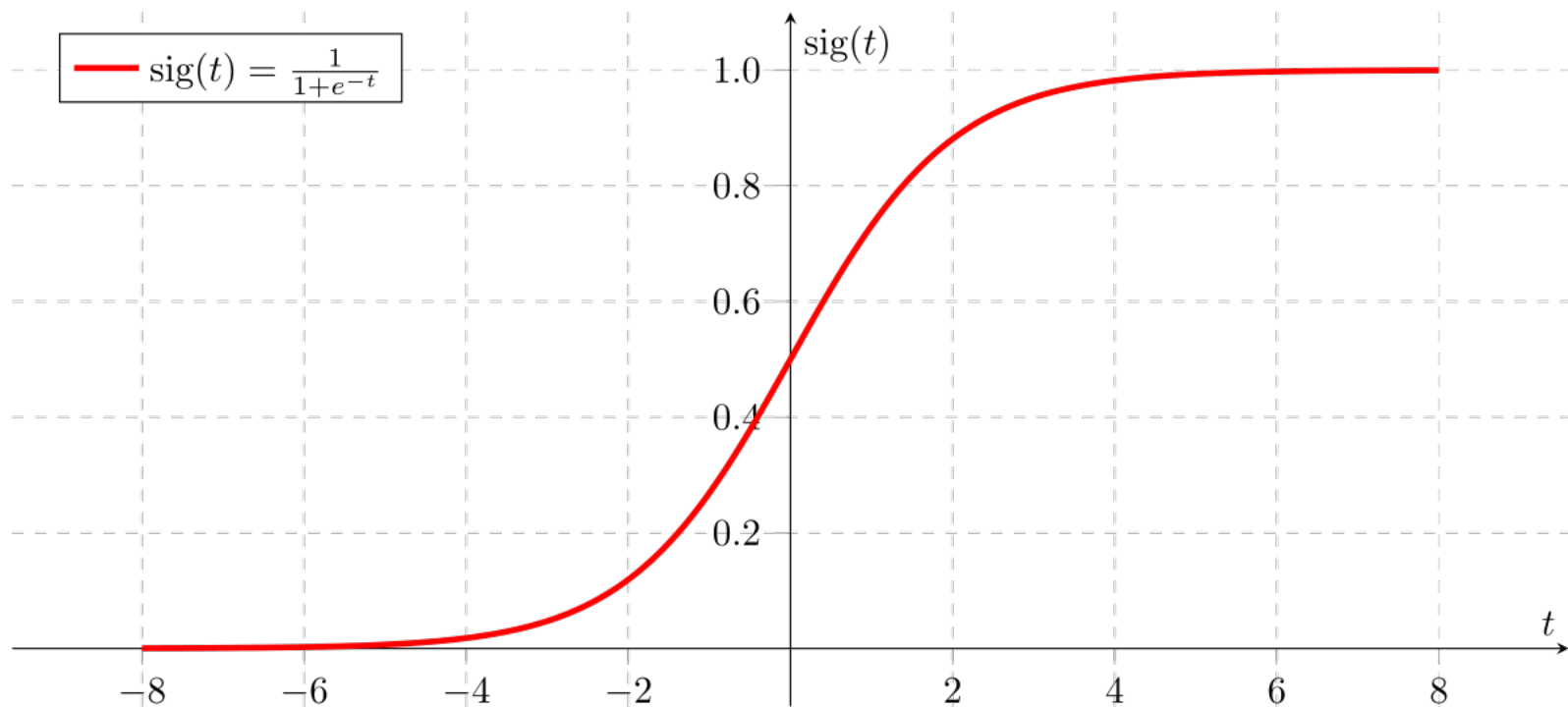
The **logistic** function...

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

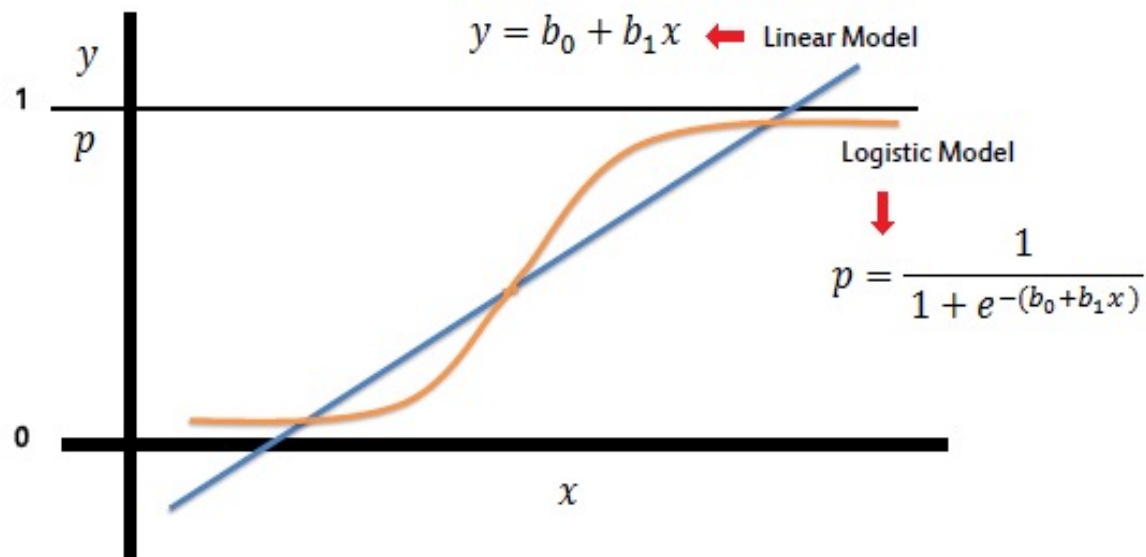
...derived

$$\frac{dg(t)}{dt} = g(t) \cdot (1 - g(t))$$

# The logistic function



# Graphical interpretation



# Exercise

Derive the expression of the logistic regression from the logit of odds

$$\text{logit}(p_i) = \ln\left(\frac{p_i}{1-p_i}\right) = \beta_0 + \beta_1 x_{1,i} + \cdots + \beta_k x_{k,i}.$$

$$p_i = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_{1,i} + \cdots + \beta_k x_{k,i})}}$$

# Loss function

*Given a training set, how do we learn the parameters?*

Basic idea: to make  $h_\theta$  close to  $y$

We've to define a function that measures, for each parameters' values, how close the  $h_\theta(x^{(i)})$ 's are to the corresponding  $y^{(i)}$ 's

# Quadratic Loss function

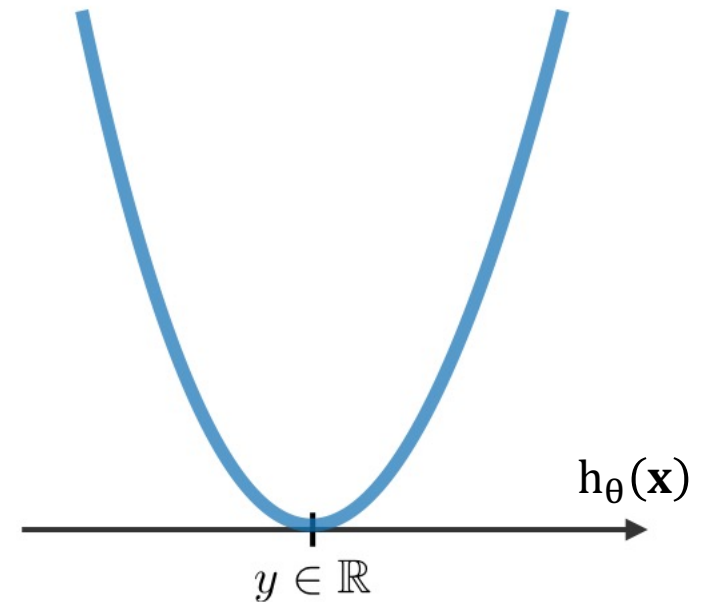
A **loss function** is a function

$$Loss: (h_{\theta}(\mathbf{x}), y) \in \mathbb{R} \times Y \rightarrow Loss(h_{\theta}(\mathbf{x}), y) \in \mathbb{R},$$

that takes as inputs the predicted value,  $h_{\theta}(\mathbf{x})$ , corresponding to the real data value,  $y$ , and outputs how different they are.

Basic idea: 1-sample cost function

$$\mathbf{Q}: Loss(h_{\theta}(\mathbf{x}), y) = (h_{\theta}(\mathbf{x}) - y)^2$$

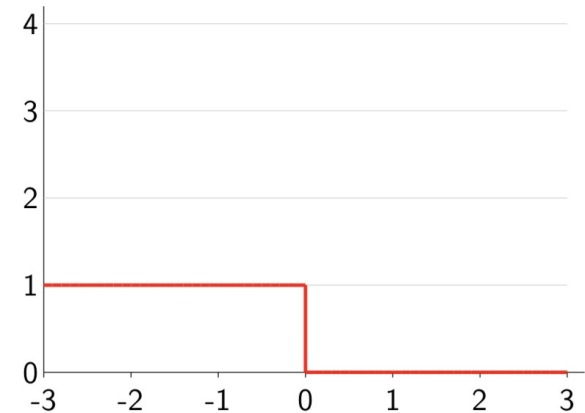


# Binary Loss function

Gradient of Loss 0-1 is 0 everywhere, therefore the stochastic gradient descent is not applicable.

In addition, Loss 0-1 is insensitive to how badly model messed up.

$$\mathbf{L}_{0-1}: \text{Loss}(h_{\theta}(\mathbf{x}), y) = \mathbf{1}[h_{\theta}(\mathbf{x}) \cdot y \leq 0]$$

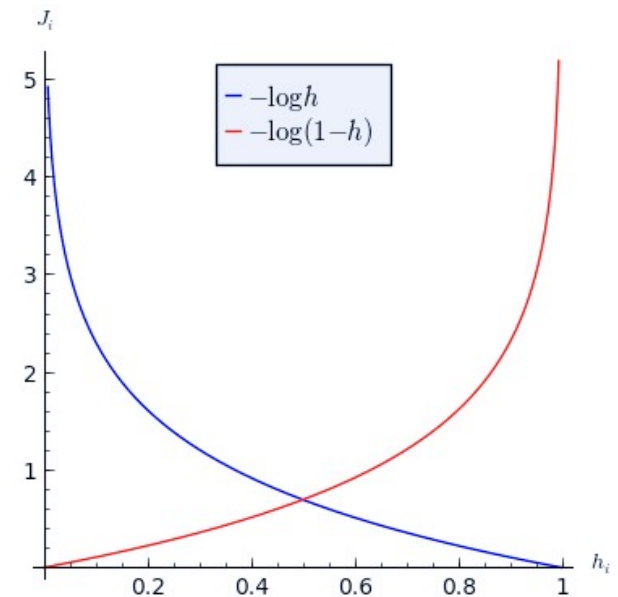


# Logistic Loss function

*Given a training set, how do we learn the parameters?*

Basic idea: to make  $h_\theta$  close to  $y$

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





# Logistic Loss function vs. MSE

$$\text{MSE}(h_{\theta}(\mathbf{x}), y) = (h_{\theta}(\mathbf{x}) - y)^2$$

$$\text{LogLoss}(\theta) = -y \cdot \log h_{\theta}(\mathbf{x}) - (1 - y) \cdot \log(1 - h_{\theta}(\mathbf{x}))$$

Y_real	Y_pred	MSE	LogLoss
1	1		
1	0		

MSE is not convex for Logistic Regression → could not really find optima !

# Logistic regression learning

Loss function:

$$J(\theta) = -y \cdot \log h_{\theta}(\mathbf{x}) - (1 - y) \cdot \log(1 - h_{\theta}(\mathbf{x}))$$

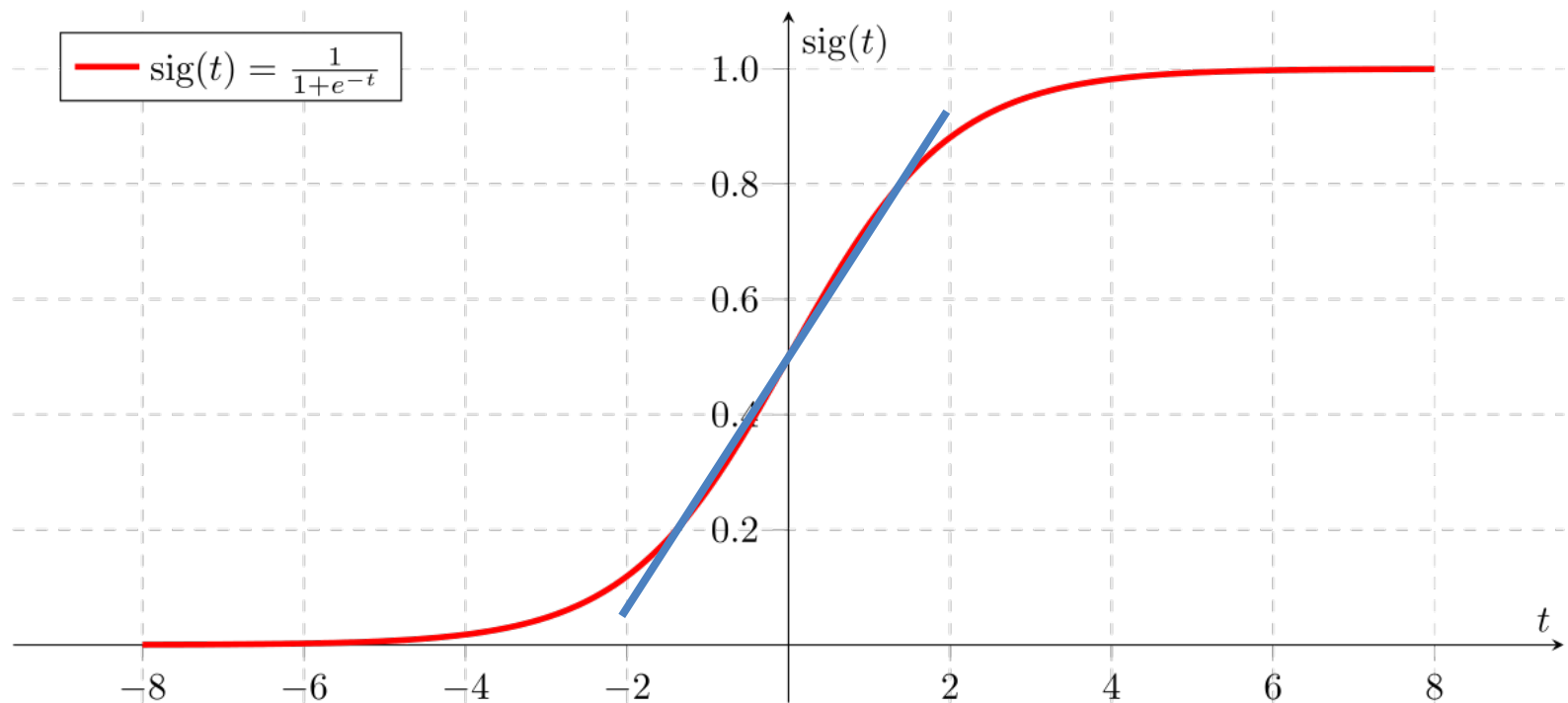
↓  
minimize  $J(\theta)$

<http://sambfok.blogspot.com.es/2012/08/partial-derivative-logistic-regression.html>

$$\theta_j := \theta_j - \alpha \cdot \frac{\partial}{\partial \theta_j} J(\theta) \longrightarrow \theta_j := \theta_j + \alpha \cdot (y - h_{\theta}(\mathbf{x}))x_j$$

Is it the same update rule as LMS?

# Logistic regression learning



# Logistic regression exercise

## Probabilistic interpretation

This Table shows the number of hours each student spent studying ML, and whether they passed (1) or failed (0).

0.50	0.75	1.00	1.25	1.50	1.75	1.75	2.00	2.25	2.50	2.75	3.00	3.25	3.50	4.00	4.25	4.50	4.75	5.00	5.50
0	0	0	0	0	1	1	0	1	0	1	0	1	0	1	1	1	1	1	1

If we use a Logistic Regression model with parameters  $\theta_0 = -4.0777$  and  $\theta_1 = 1.5046$  as a learning result, what would be the probability function of passing conditioned to the number of study hours? What would be the minimum number of hours to have more likely to pass the exam?

# Logistic regression exercice

From the Dataset, the number of the input data dimensions is  $n=1$ , then:

$$h_{\theta}(\mathbf{x}) = g(\theta_0 + \theta_1 x_1) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x_1)}} = \frac{1}{1 + e^{-(-4.07 + 1.5x_1)}}$$

that could be interpreted as a function of the probability to pass the exam conditioned to the student number of study hours,  $P(\text{"pass the exam"}|\mathbf{x}, \theta_0, \theta_1)$ .

# Logistic regression exercise

For the second question we need to compute:

$$h_{\theta}(\mathbf{x}) > 0.5$$

Odds of passing exam

$$\frac{1}{1 + e^{-(-4.07 + 1.5x_1)}} > \frac{1}{2}$$

$$e^{-(-4.07 + 1.5x_1)} < 1$$

$$\log(e^{-(-4.07 + 1.5x_1)}) < \log(1)$$

$$4.07 - 1.5x_1 < 0$$

$$x_1 > 2.71$$

hours	odds
1	1:13
2	1:3
3	3:2
4	7:1
5	31:1

2.71 is the minimum number of hours to have more likely to pass the exam.

# Summary: Linear models

$$\theta_j := \theta_j + \alpha \cdot (y - h_{\theta}(\mathbf{x})) \cdot x_j$$

LMS

$$h_{\theta}(\mathbf{x}) = \sum_{j=0}^n \theta_j x_j$$

Logistic regression

$$h_{\theta}(\mathbf{x}) = \frac{1}{1 + e^{-\sum_{j=0}^n \theta_j x_j}}$$

Perceptron

$$h_{\theta}(\mathbf{x}) = \begin{cases} 1, & \sum_{j=0}^n \theta_j x_j \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

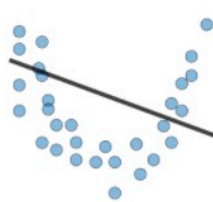

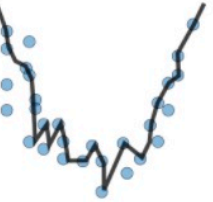
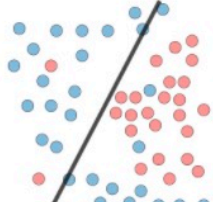
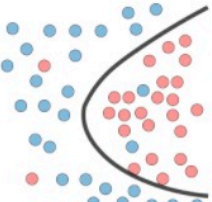
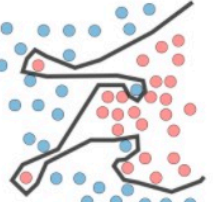



# Bias-Variance Tradeoff

**Alto bias:** Simplificaciones sobre la forma de los datos, *menos flexibles*: Linear models

**Baja Varianza:** Pequeños cambios en las predicciones: Linear models

**Bajo bias:** Pocos supuestos sobre la forma de los datos, *más flexibles*: Decision trees, knn, ...

**Alta Varianza:** Grandes cambios en las predicciones: Decision trees, knn, ...

	Underfitting	Just right	Overfitting
Symptoms	<ul style="list-style-type: none"> <li>• High training error</li> <li>• Training error close to test error</li> <li>• High bias</li> </ul>	<ul style="list-style-type: none"> <li>• Training error slightly lower than test error</li> </ul>	<ul style="list-style-type: none"> <li>• Very low training error</li> <li>• Training error much lower than test error</li> <li>• High variance</li> </ul>
Regression illustration			
Classification illustration			
Deep learning illustration			
Possible remedies	<ul style="list-style-type: none"> <li>• Complexify model</li> <li>• Add more features</li> <li>• Train longer</li> </ul>		<ul style="list-style-type: none"> <li>• Perform regularization</li> <li>• Get more data</li> </ul>



# Bias-Variance Tradeoff

La única forma de comprobar que nuestro modelo generaliza bien es guardando una parte de los datos para medir el error *out-of-sample*, es decir, el error sobre los datos que no ha visto el modelo durante la fase de entreno.

Con tal de poder elegir el mejor modelo y sus parametros se dividirá inicialmente el dataset en dos subsets:

- subset de entrenamiento: normalmente se utiliza el 80% del tamaño de la muestra
- subset de testeo: el 20% restante.

