# Learning from examples

# (Učenie na základe skúsenosti)

*Lubica Benuskova*

AIMA 3rd ed., Chap. 18.1 – 18.2 & 18.6.1 – 18.6.2

# Nature-inspired computing

- Current machines still fall short in performance compared to humans in certain complex tasks (like visual processing, language understanding, complex reasoning, lifelong learning, etc.).

- All living biological organisms effectively function in their environments. They have evolved to do so over the long time.

- Each biological organism (individual) with the brain
  - Is born with certain innate properties inherited from parents,
  - Has ability to learn other new things thanks to their brains.

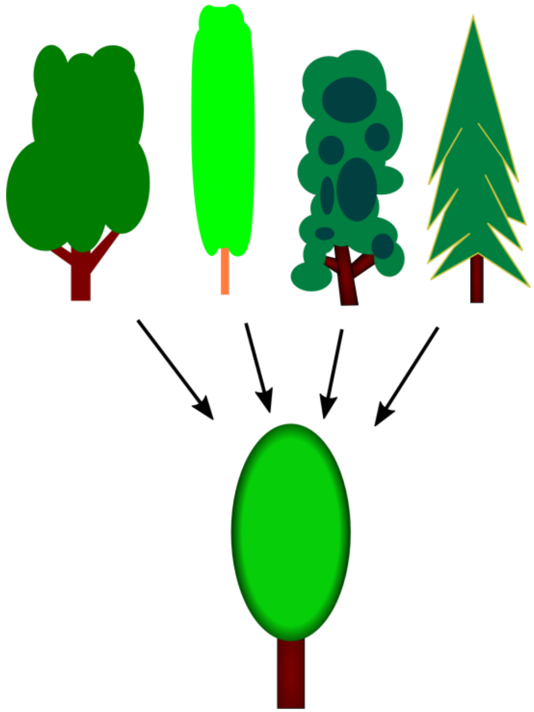
- Thus, nature can be an excellent source of inspiration for AI, provided we can express the cognitive processes by computational means (math, algorithms, etc.).

# Learning and generalization

- All our cognitive functions, including our sense of identity, are underpinned by what we have learned and what we can remember.

- The goal of learning is *generalization* (zovšeobecňovanie), i.e., evoking of a response learned to one stimulus by a different but similar stimulus.

- When the mind makes a generalization, it extracts the essence of a concept based on similarities from many discrete objects. The resulting general concept enables higher-level thinking.



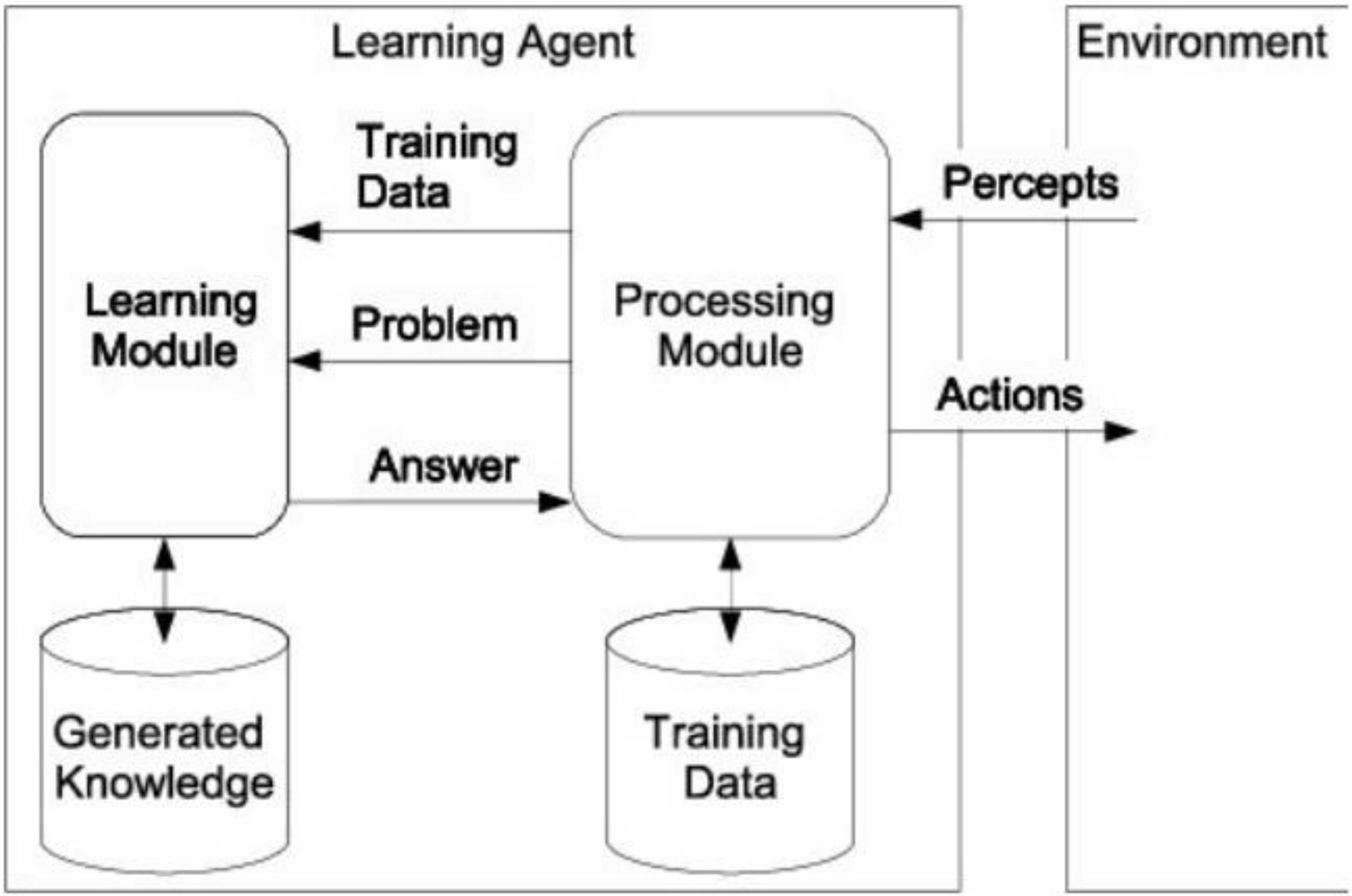
# Learning from observations

- In *observation*, biological or artificial agent senses and records data from environment (external and internal) through its sensory organs.

- Observation (*perception, sensory experience*) is used by a **process of learning** to improve agent's ability to act in the future.

- Design of an artificial agent's *learning module* is affected by
  - Which components of the agent's performance are to be learned
  - What feedback is available to teach these components
  - What representation (i.e. rule-based, probabilistic, etc.) is used for the components

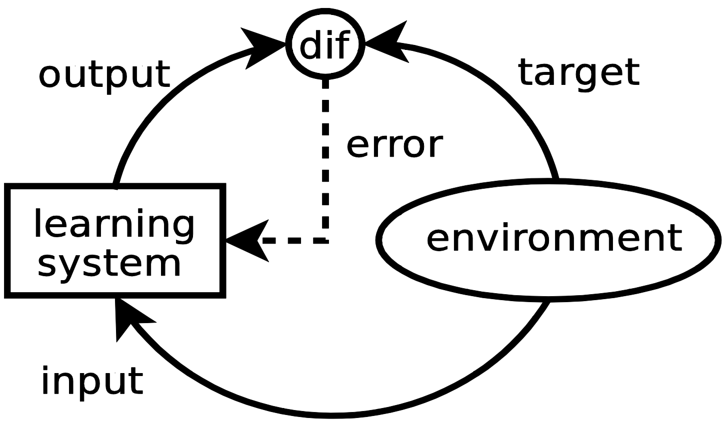
# Learning agent



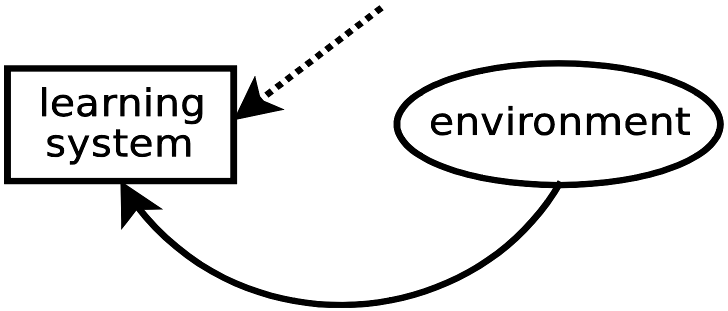

- Image provided by Bartłomiej Śnieżyński at ResearchGate.

# 3 ways of learning based on given feedback

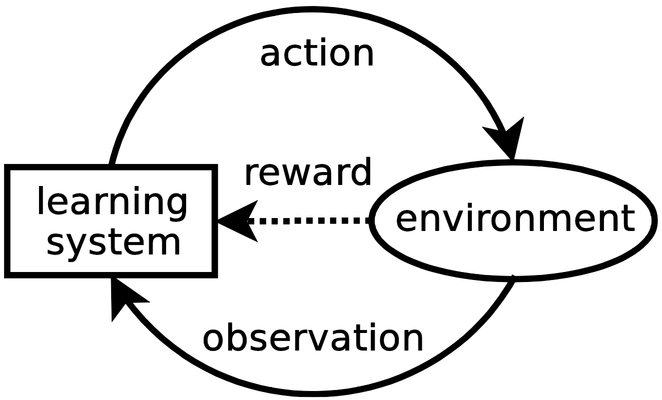
supervised (with teacher)
- učenie s učiteľom




unsupervised (self-organized)
- učenie bez učiteľa




reinforcement learning (partial feedback)
- učenie s posilňovaním



# Inductive learning

- Inductive learning: specific examples are used to infer a general rule
  - Deduction: general rule is used to explain specific examples

- Task of induction (inductive inference): given a **training set** of examples, return a function $h$ that approximates the unknown $f$
  - $f$ is the target function
  - an example is a pair $(x, f(x))$ = (input, output)
  - Function $h$ is a hypothesis such that $h \approx f$

- The fundamental problem of *induction* is to find a good hypothesis that generalizes well, i.e. will predict $f(x)$ for unseen input $x$

- This is a highly simplified model of real learning:
  - Ignores prior knowledge
  - Assumes representative examples are given

# Hypothesis space **H**

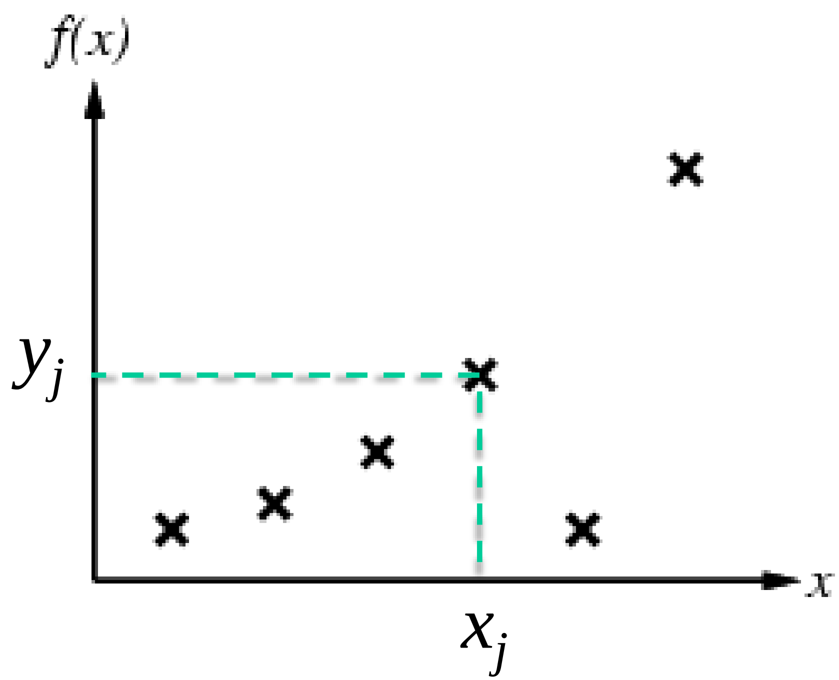
- **H** = set of hypotheses $\{h\}$ we will consider. E.g. to find a curve to fit the data points, we can work with:
  - the set of polynomials of finite degree or
  - the set of all trigonometric functions or
  - the set of all functions

- There is a *tradeoff* between the expressiveness of **H** and the computational complexity of finding the most simple hypothesis within **H,** which is consistent with the data (examples).

- We must use some *prior knowledge* about the problem to come up with the plausible set **H.**

# Example: fitting a curve to data

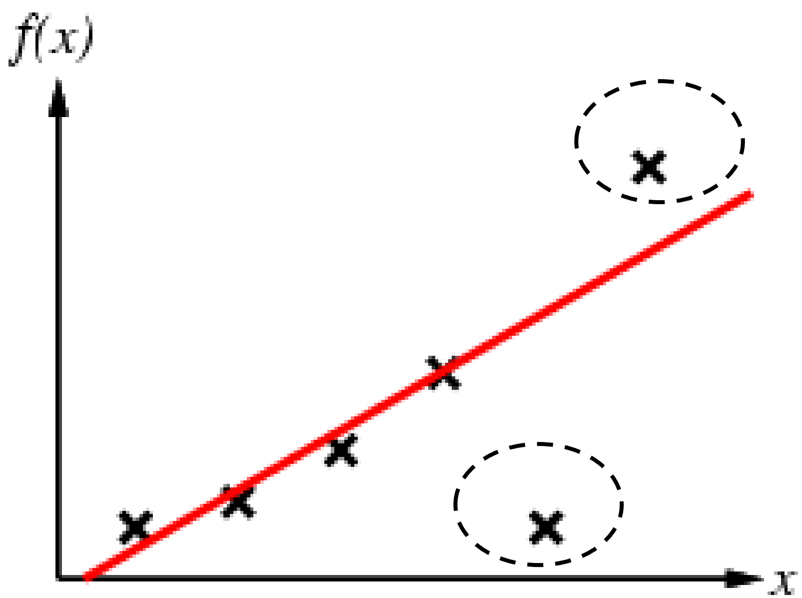
- Task: construct a hypothetical function $h$ that agrees with real (but unknown) function $y = f(x)$ on the training set.
- Training set is a set of real data points that are generated according to a real function $y = f(x)$.
- Hypothesis $h$ is consistent, if it agrees with $f$ on all examples, i.e. for all training data points $(x_j, y_j)$.



# The simplest hypothesis – a line

- Construct/find / adjust $h$ to agree with $f$ on the training set

- Data fitting by a **line** (1st degree polynomial) – two data points are not accounted for i.e. the hypothesis is **not consistent**



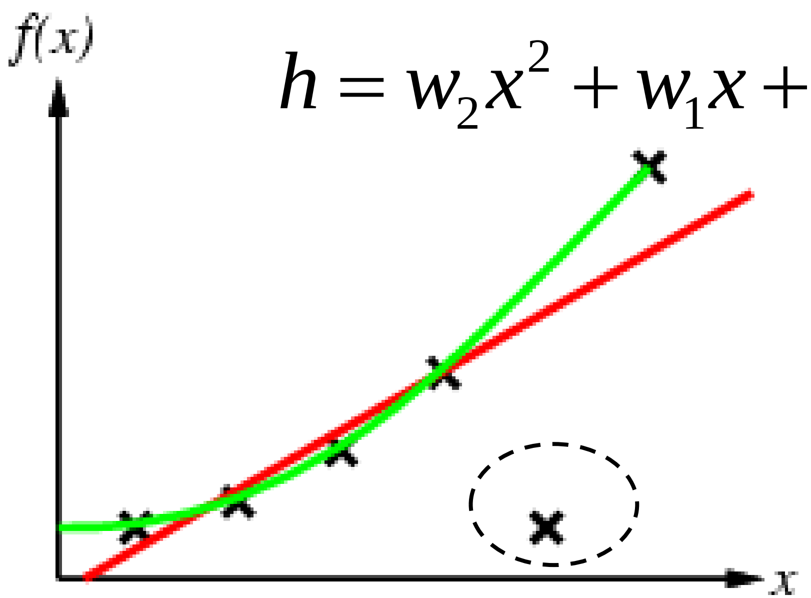

$$h = w_1 x + w_0$$

# More complex hypothesis – parabola

- Curve fitting by parabola (i.e., 2nd degree polynomial) – **not consistent** because one data point is off

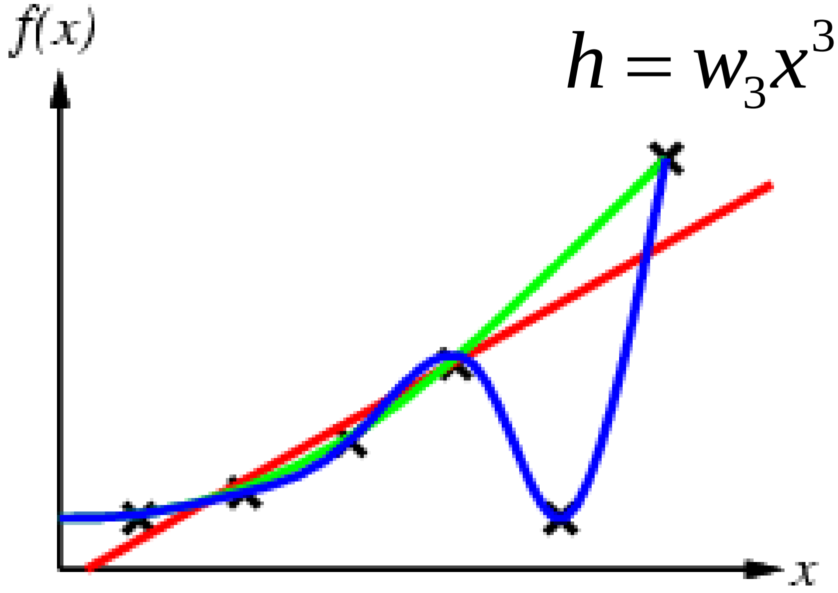
- Parameters $w_0$, $w_1$ and $w_2$ are coefficients of parabola

$$h = w_2 x^2 + w_1 x + w_0$$



# Even more complex hypothesis – 3rd degree curve

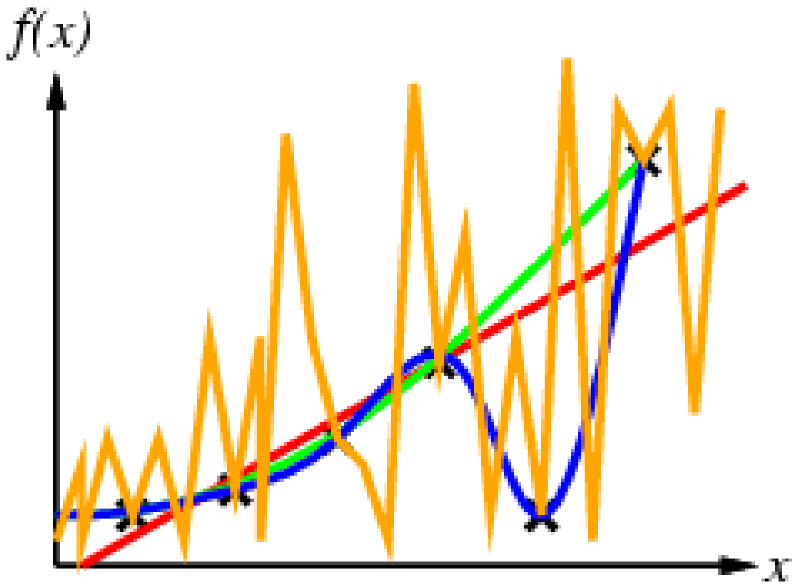
- Curve fitting by a 3rd degree polynomial – **consistent!**




$$h = w_3 x^3 + w_2 x^2 + w_1 x + w_0$$

# There are many solutions – which one is right?

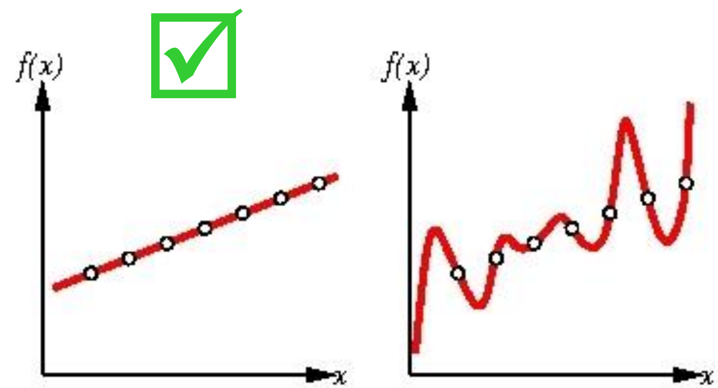
- Curve fitting by (big n)$^{th}$ degree polynomial – **consistent!**
- In fact, there are many solutions, i.e. many different hypotheses that are consistent with the data. Which one is correct?
- How well each of them generalises to a new $x$ ?



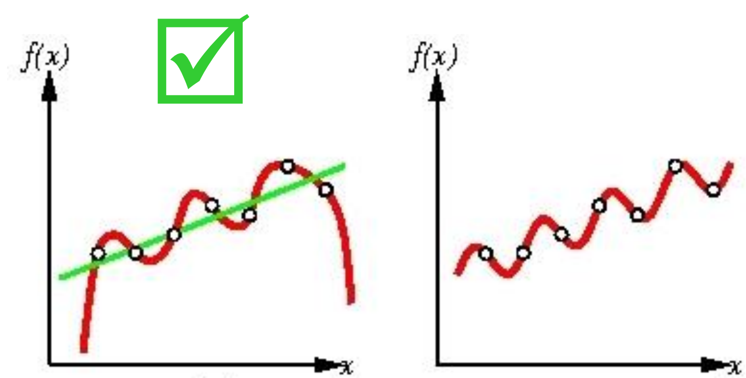
# Ockam's razor

How do we choose from among several consistent hypotheses?

- Ockham's razor: prefer the ***simplest hypothesis*** consistent with the data
  - Ockham's razor or law of parsimony attributed to the 14th century Franciscan logician William of Ockham (Occam)

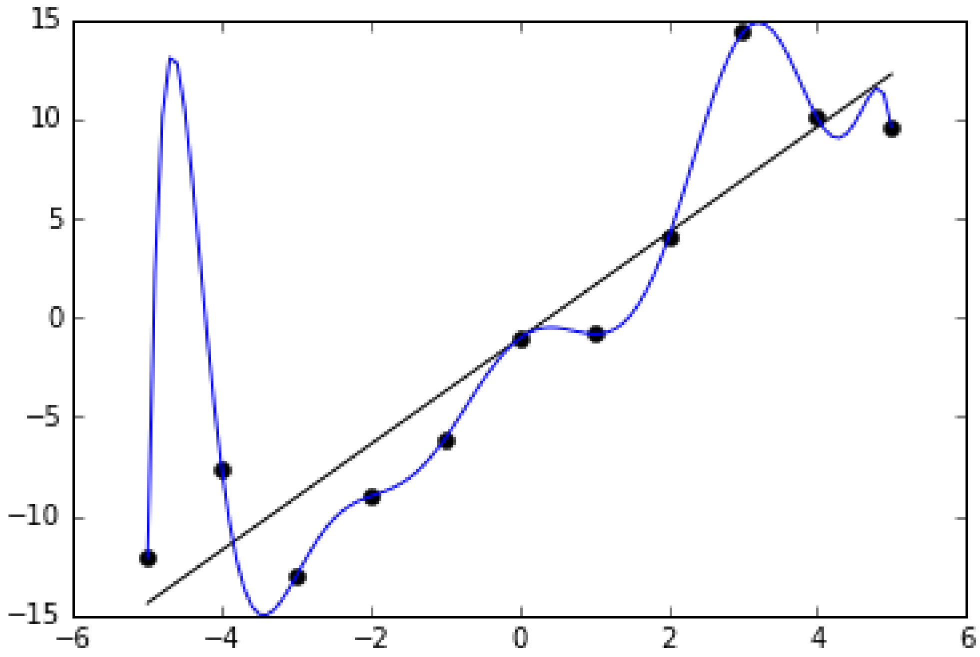



- In general, there's a tradeoff between the complexity of function and fitting the data
  - Preference to the simpler hypothesis even if it does not fit data perfectly

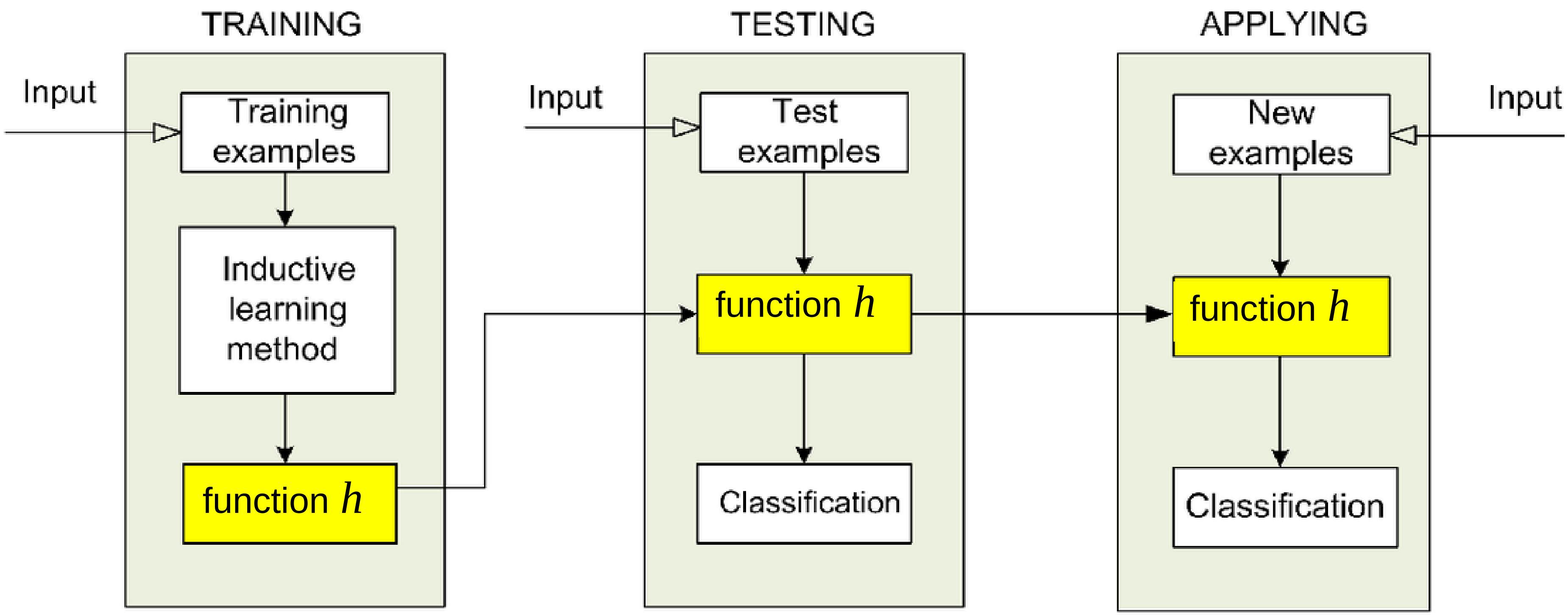


# Overfitting (overtraining, overlearning)

- In overfitting, a model is affected by random error or noise instead of the underlying relationship (e.g. a polynomial vs. a line).
- Overfitting occurs when a model is excessively complex, such as having too many parameters relative to the number of observations.
- A model that has been overfit has poor predictive performance, i.e. poor generalization to new data.

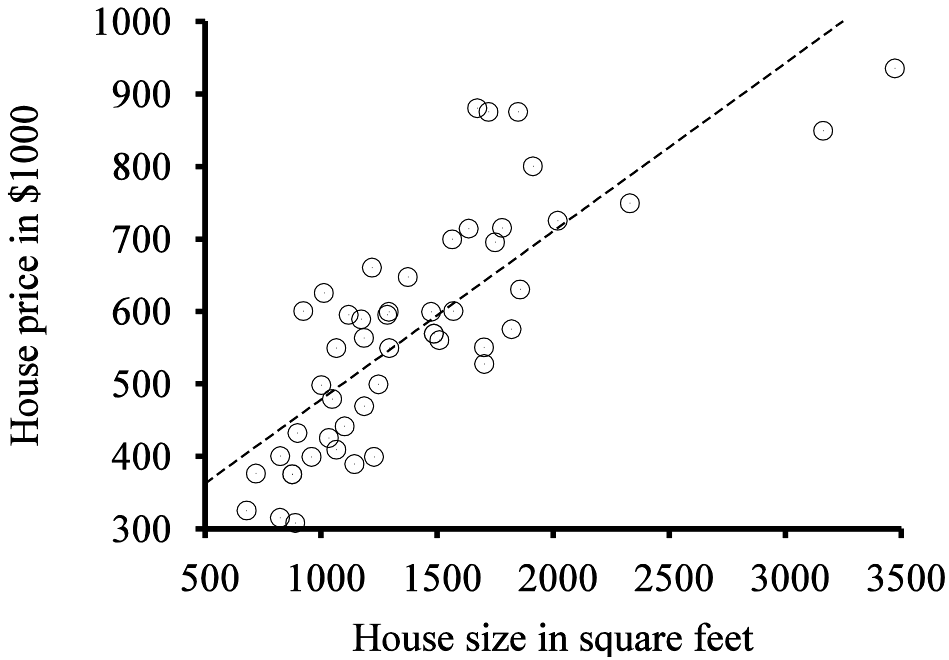


# What happens after training




- Inductive learning method leads to finding / learning the concrete values of the real-valued coefficients of the target function *h*.

# Example of real data




- Q: can I predict the selling price of my house based on this data?

- **A: Yes, if you build a good model of how the price depends on the house size.**

# Regression analysis

- In mathematics, regression analysis is the process of estimating the relationships among variables (i.e. finding the right model, function $h$).

- The focus is on the relationship between a dependent variable (e.g. house selling price) and one or more independent variables (e.g. house size).

- We'll start with the simplest case: regression with a univariate linear function, otherwise known as "fitting a straight line" to the data.

- This will be our (simplest possible) hypothesis, i.e. that the variables have an underlying linear relationship.

# Univariate linear regression

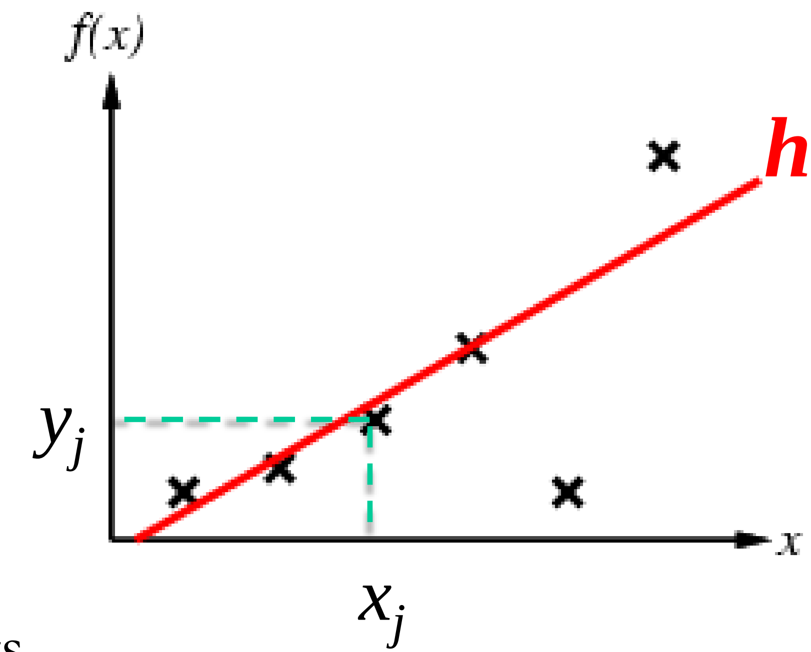
- Univariate linear function (a straight line) with input $x$ and output $h$ has the mathematical form →

$$h = w_1 x + w_0$$

- Input $x$ represents an independent variable and output $y = f(x)$ represents the values of dependent variable.

- Function $h$ represents our hypothesis.

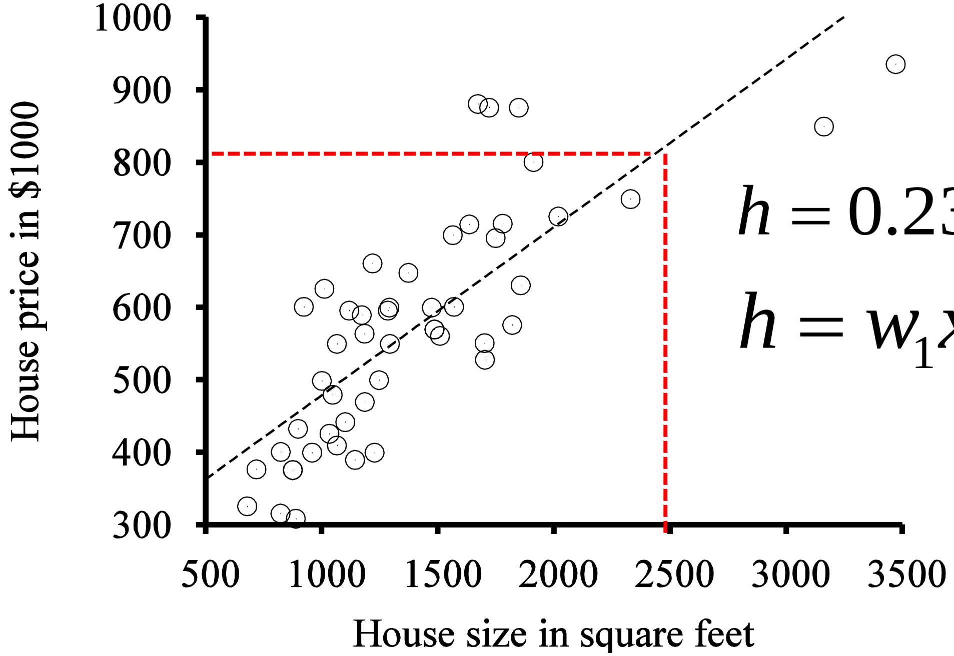
- The task of finding / learning the concrete values of the real-valued coefficients $w_1$ and $w_0$ so that $h$ best fits the data is called **linear regression**.



# Solution of linear regression for house data

- For our example with the house prices, the values of coefficients are $w_1 = 0.232$ and $w_0 = 246$.

- Now we can use this function to estimate the price of a new house based on its size, e.g. the house of size 2500 feet$^2$ will cost 800 000$.




$$h = 0.232x + 246$$

$$h = w_1 x + w_0$$

# How to find/derive values of coefficients?

- To fit $h$ to the data, we have to find values of the real-valued coefficients $w_1$ and $w_0$ that ***minimize*** the empirical loss function.

- Mathematicians Gauss and Legendre introduced formula for the squared loss function L2, summed over all the training examples:

$$L_2 = \sum_{j=1}^{N} \left( y_j - h(x_j) \right)^2 = \sum_{j=1}^{N} \left( y_j - \left( w_1 x_j + w_0 \right) \right)^2$$

- Here, $y_j = f(x_j)$ is the value of the dependent variable (house price) for the independent variable $x_j$ (house area) for $N$ data points
- $h(x_j)$ is the value of our approximation function (i.e. line) for $x_j$
- The ***training set*** is the set of $N$ pairs of values $(x_j, y_j)$

# Solving partial derivatives of $L_2$

- If we want to find a minimum of the function, we have to calculate partial derivatives of that function with respect to the sought after variables (in this case $w_1$ and $w_0$ ) and solve these derivates when they are equal to zero.

$$\frac{\partial}{\partial w_0}\sum_{j=1}^{N}\left(y_j-\left(w_1x_j+w_0\right)\right)^2=0 \quad and \quad \frac{\partial}{\partial w_1}\sum_{j=1}^{N}\left(y_j-\left(w_1x_j+w_0\right)\right)^2=0$$

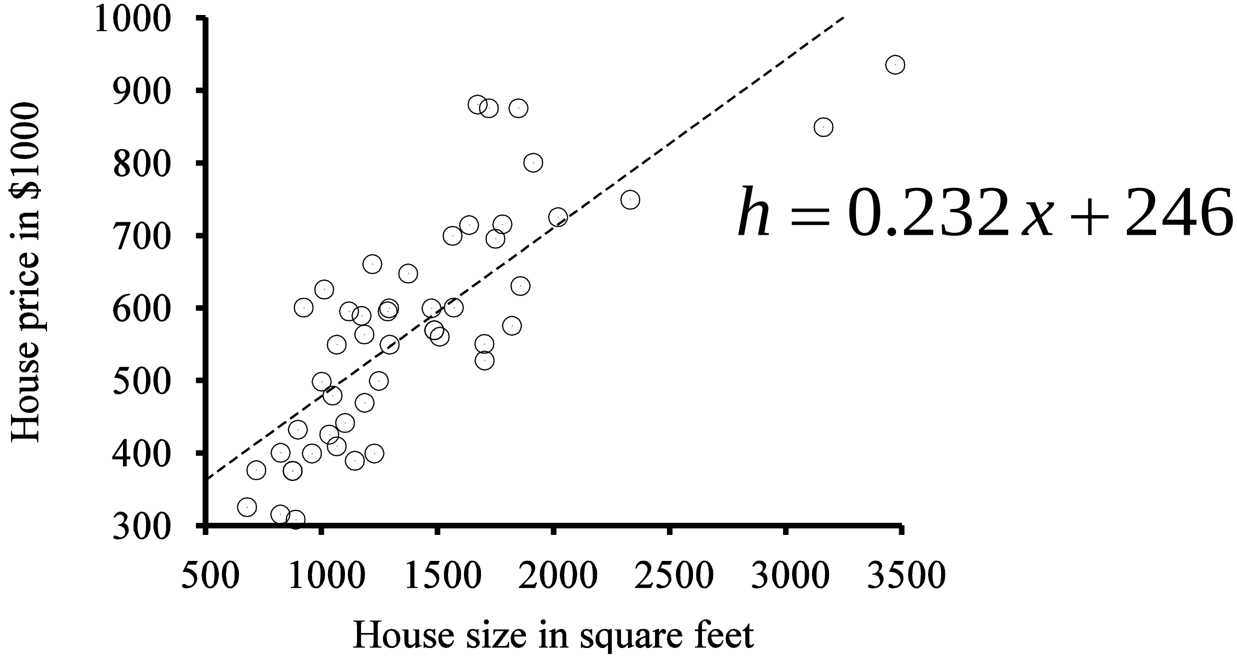
- These equations have unique analytical solutions which are:

$$w_0=\frac{\left(\sum y_j-w_1\left(\sum x_j\right)\right)}{N} \quad and \quad w_1=\frac{N\left(\sum x_jy_j\right)-\left(\sum x_j\right)\left(\sum y_j\right)}{N\left(\sum x_j^2\right)-\left(\sum x_j\right)^2}$$

# Solution of linear regression for house data

- For our example with the house prices, by solving these equations we get the values of $w_1 = 0.232$ and $w_0 = 246$.

- The line with these values is shown as a dashed line in the figure.

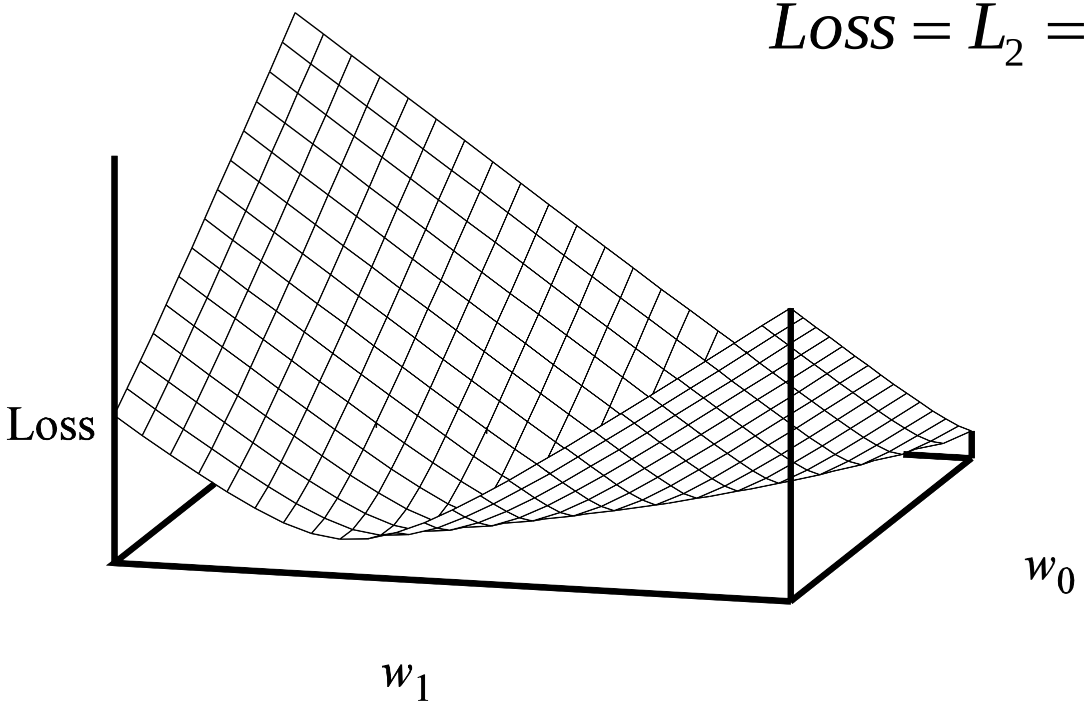


# Landscape (profile) of the $L_2$ for the house data

- We can plot the values of empirical loss function for all data points ($x_j$ , $y_j$) and all possible values of $w_1$ and $w_0$ .

- The loss is minimal only for values of $w_1$ = 0.232 and $w_0$ = 246.

$$Loss = L_2 = \sum_{j=1}^{N}\left(y_j - \left(w_1 x_j + w_0\right)\right)^2$$



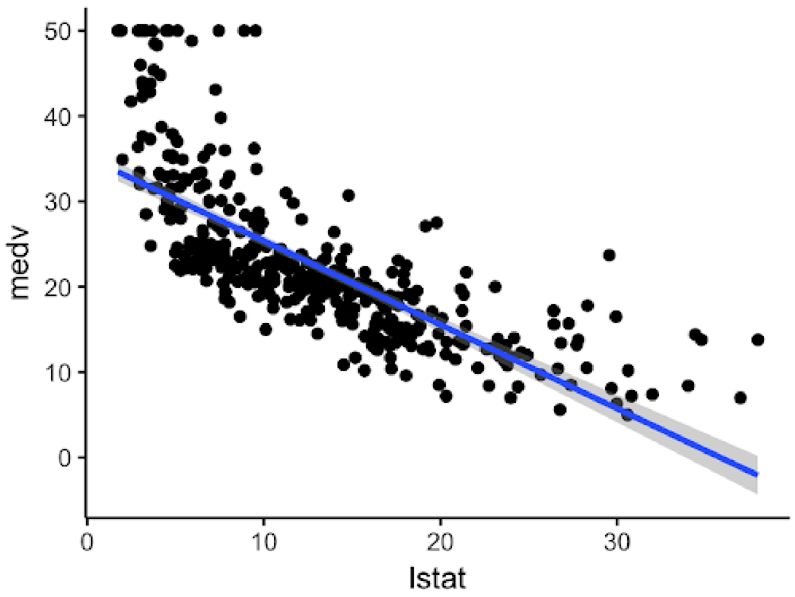

**Mean Squared Error**

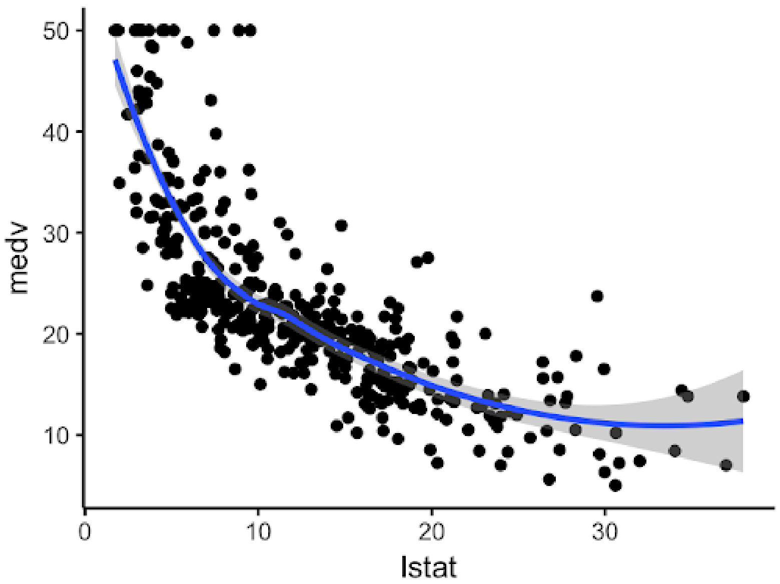
$$MSE = \frac{1}{N} L_2$$

# Linear versus nonlinear regression analysis

- But what if the linear model is not correct – how do we find out?

- We should try also the nonlinear functions and for each of them calculate $L_2$. Then we will see which curve fits the data the best, i.e. for which model we get a minimal value of $L_2$.



# Univariate nonlinear regression

- Univariate linear function, a polynomial of the first degree (a straight line) with input $x$ and output $h$ has the mathematical form

$$h = w_1 x + w_0$$

- Univariate polynomial of the 2nd degree (parabola) has the form:

$$h = w_2 x^2 + w_1 x + w_0$$

- Univariate polynomial of the 3rd degree has the form

$$h = w_3 x^3 + w_2 x^2 + w_1 x + w_0$$

# Loss functions for univariate regression

- Squared loss function $L_2$ for univariate linear function reads:

$$L_2 = \sum_{j=1}^{N} \left(y_j - h(x_j)\right)^2 = \sum_{j=1}^{N} \left(y_j - (w_1 x_j + w_0)\right)^2$$

- $L_2$ for univariate polynomial of the 2nd degree (parabola) has the form:

$$L_2 = \sum_{j=1}^{N} \left(y_j - h(x_j)\right)^2 = \sum_{j=1}^{N} \left(y_j - (w_2 {x_j}^2 + w_1 x_j + w_0)\right)^2$$

- Univariate polynomial of the 3rd degree has the form

$$L_2 = \sum_{j=1}^{N} \left(y_j - h(x_j)\right)^2 = \sum_{j=1}^{N} \left(y_j - (w_3 {x_j}^3 + w_2 {x_j}^2 + w_1 x_j + w_0)\right)^2$$

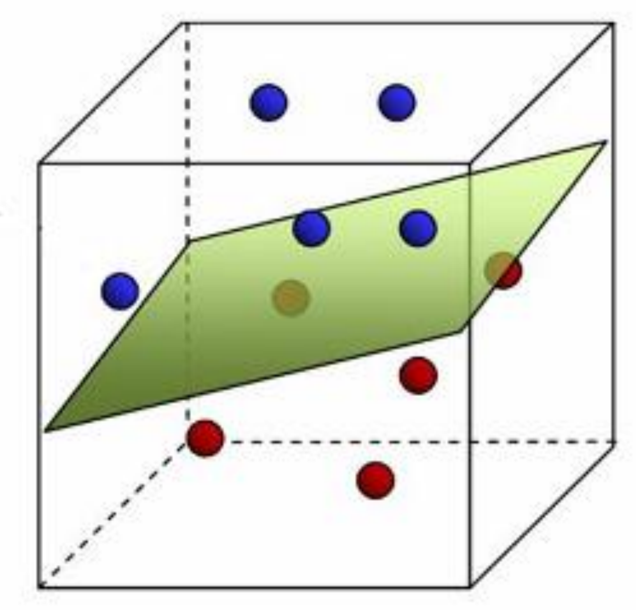
# Classification and regression

- In machine learning and statistics, classification is identifying to which of a set of categories (classes) an observation belongs.

- An algorithm (or mathematical function) that implements classification, is known as a **classifier**.

- The simplest boundary that separates two classes of objects is a linear boundary.

- In this example we have two classes of objects that can be separated by a plane.

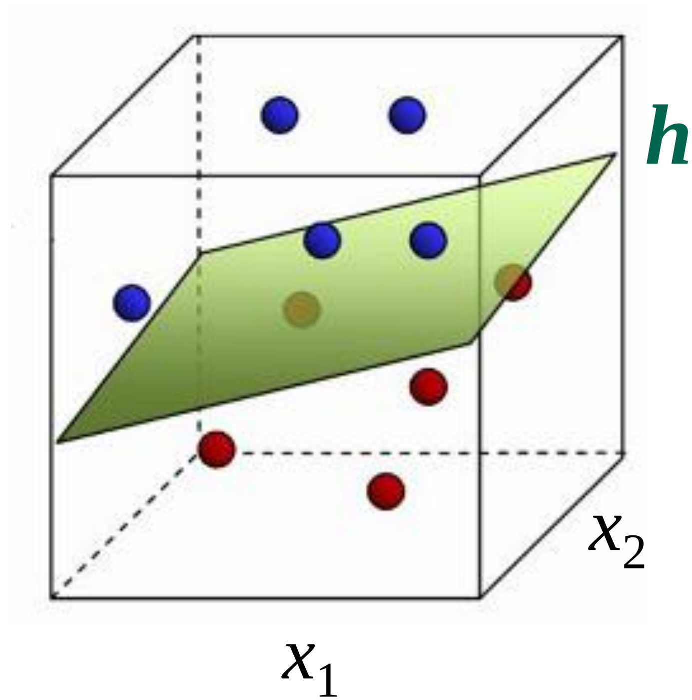


# Classification and regression

- Linear function (a hyperplane) with input vector $\boldsymbol{x}$ and output $h$ has the mathematical form →

$$h = w_0 + \sum_{i=1}^{n} w_i x_i$$

- Input vector $\boldsymbol{x} = (x_1, x_2)$ represents independent variables.
- Function $h$ represents our hypothesis for division boundary.
- The task is to **derive** concrete values of the real-valued coefficients $w_i$ and $w_0$ so that $h$ best fits the data.



# Multivariate regression analysis

- We can extend the regression theory to a multivariate regression problems, in which the function $f$ is a function of many independent variables, $x_1$, $x_2$, $x_3$,…, $x_i$, …$x_n$.

- The values of coefficients $w_i$ can be obtained either analytically (from $L_2$) or by using the method of gradient descent (e.g., error-backpropagation) or some method of stochastic optimisation (e.g., genetic algorithm).

- **Perceptron and multi-layer perceptron (MLP) automatically perform multivariate regression analysis (topic of next lectures).**