# From Neural Networks to parametric modeling

- Cesare Alippi

# What is machine learning

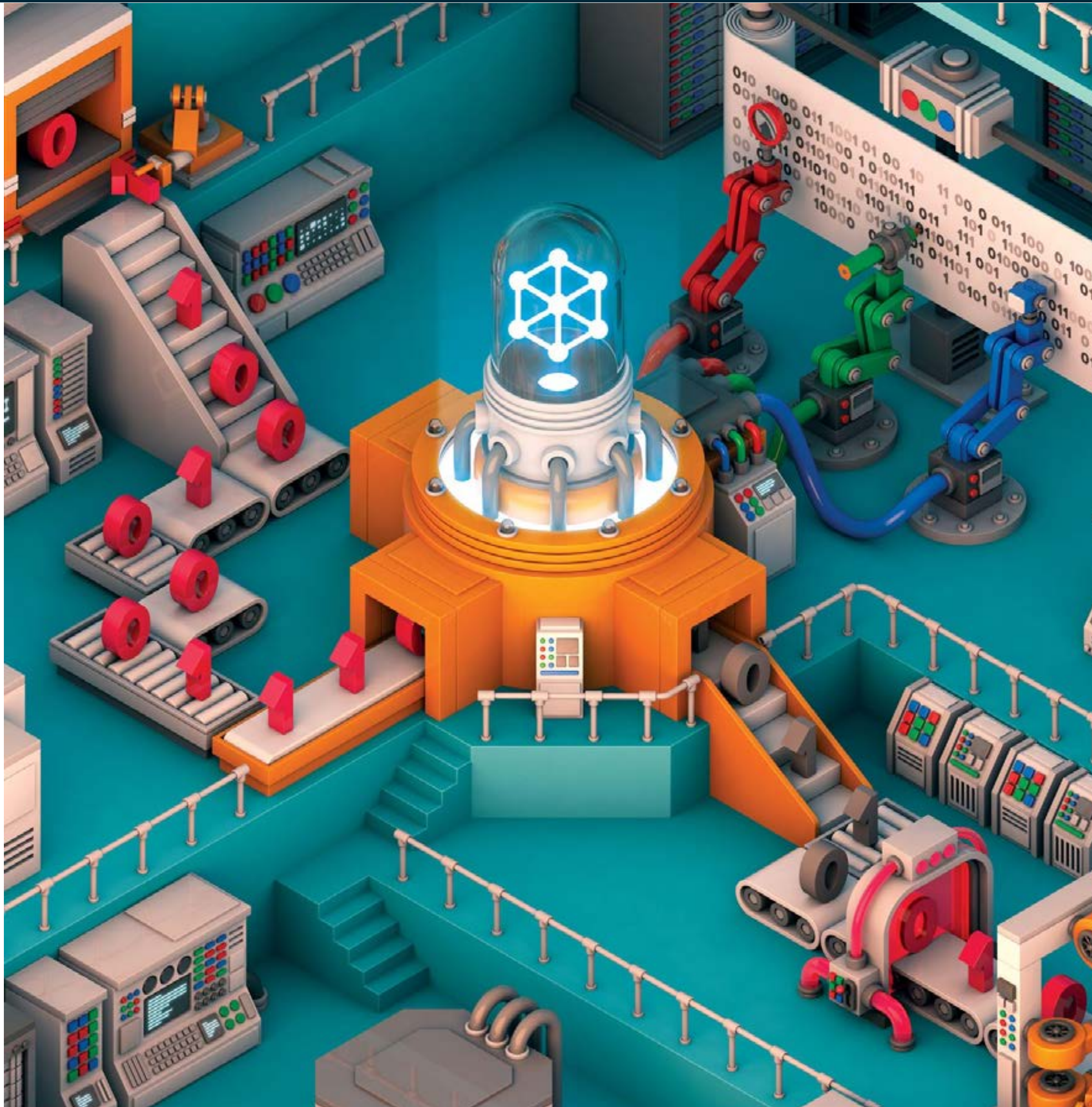
***«Yes, it is a hot topic»***

# What does it mean “to learn”?

- Hastie, Tibshirani, Friedman:
  - “Vast amounts of data are being generated in many fields, and the statisticians’s job is to make sense of it all: to extract important patterns and trends, and to understand “what the data says”. We call this *learning from data.*”
- Mitchell:
  - “The field of machine learning is concerned with the question of how to construct computer programs that automatically improve with experience.”
- Alippi:
  - “The ultimate goal of machine learning is to provide the simplest consistent method able to explain past and future data without requesting an explicitly programmed algorithm.”

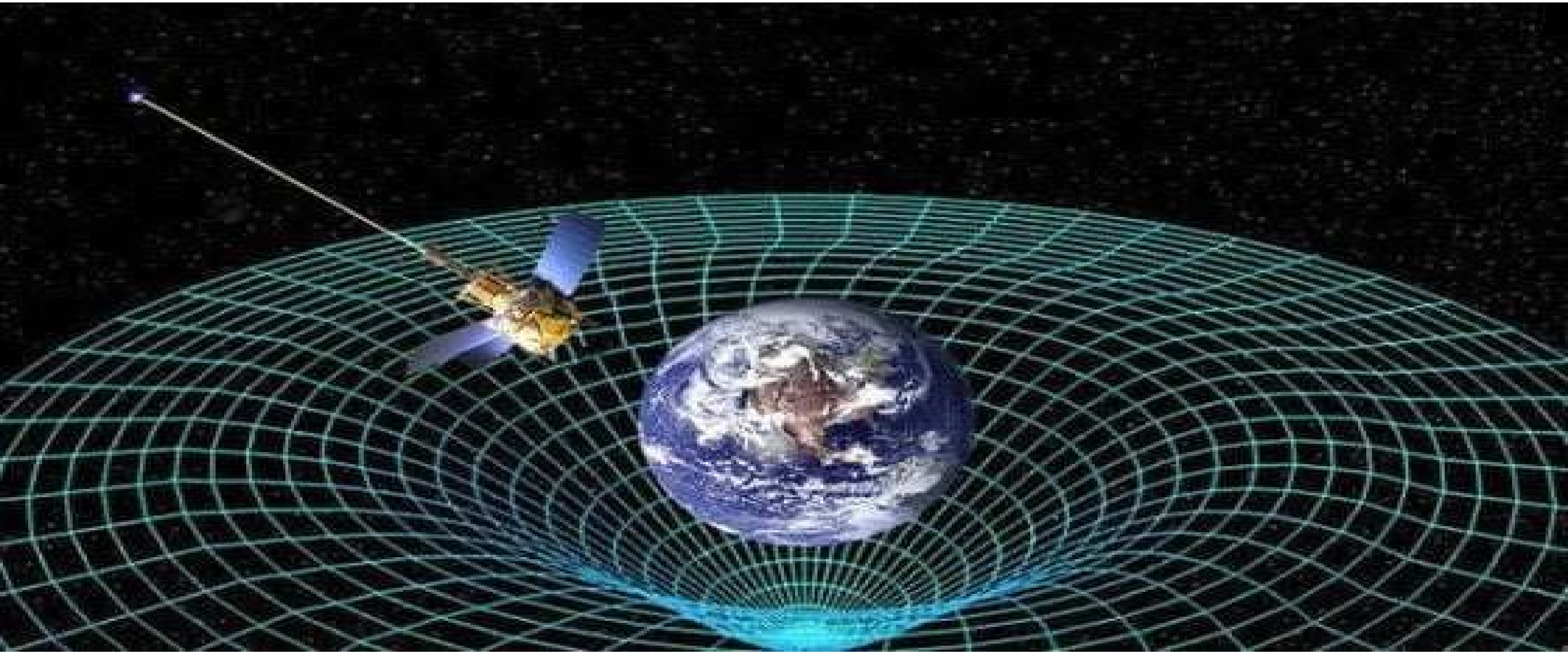
# Mitchell formalization of the learning framework

A computer program is said to **learn** from **experience *E*** with respect to some class of **tasks *T*** and **performance measure *P***, if its performance at tasks in *T*, as measured by *P*, improves with experience *E*.



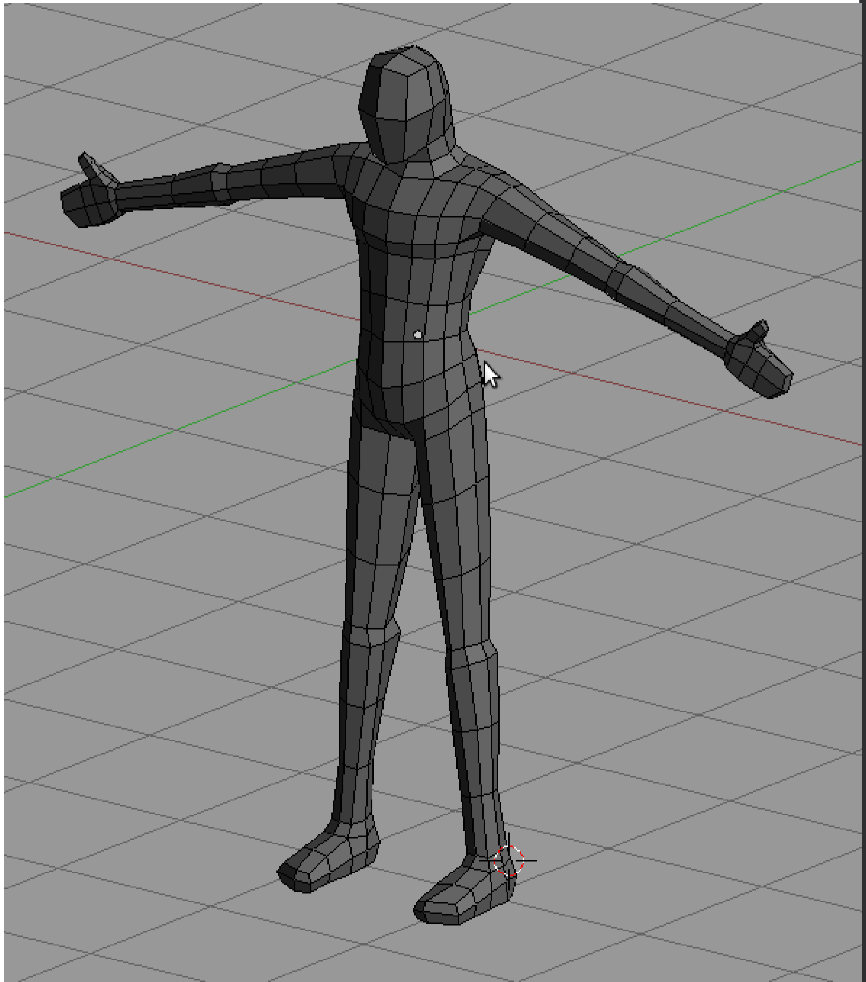
# The regression problem

- **Linear regression**
- **Nonlinear regression**
- **Feed-forward neural networks**



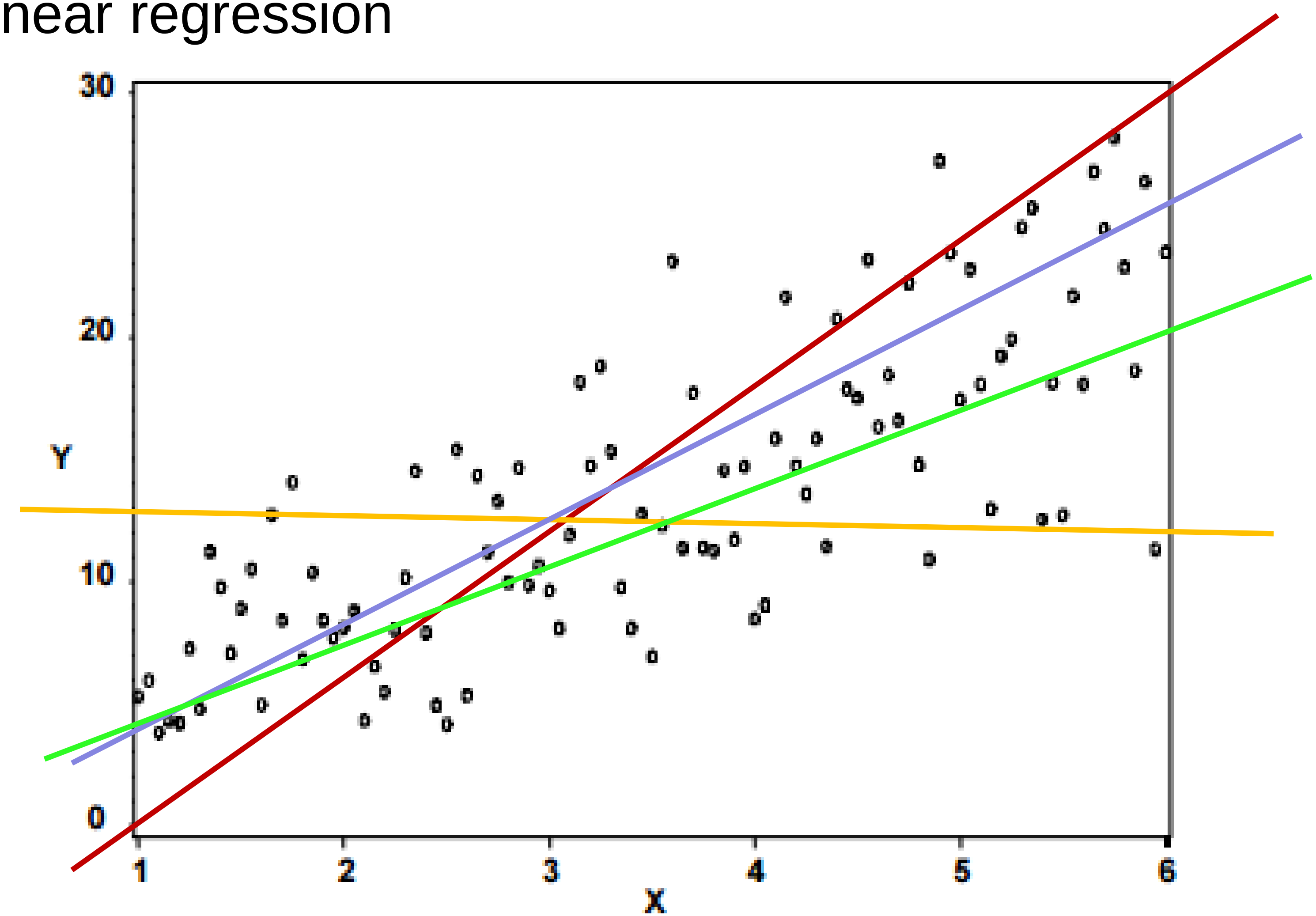
# Linear regression

***«Keep the model simple»***



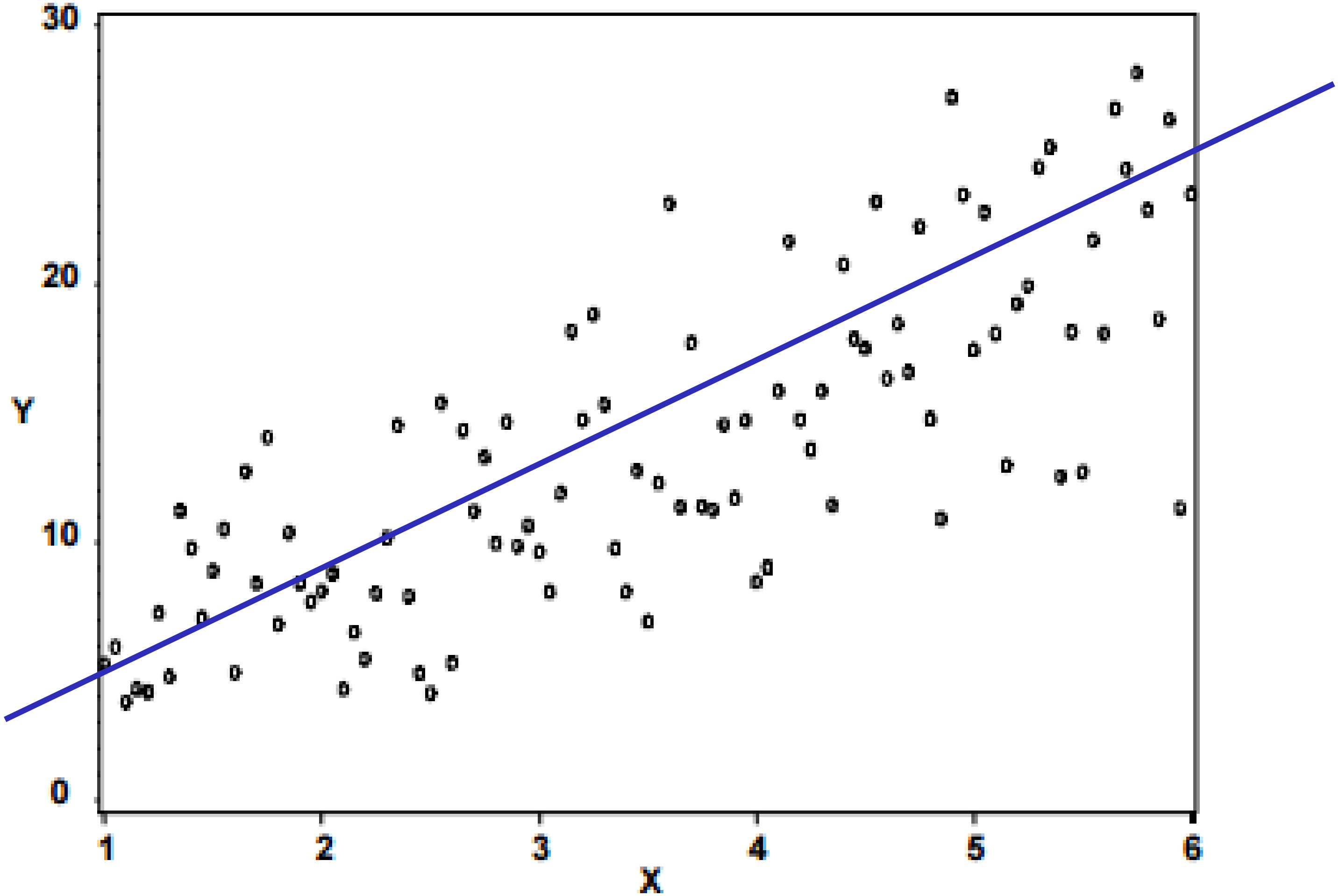
# Some examples

- linear regression



# Some examples

- linear regression: least mean square algorithm



# Multiple Linear regression (as they taught you)

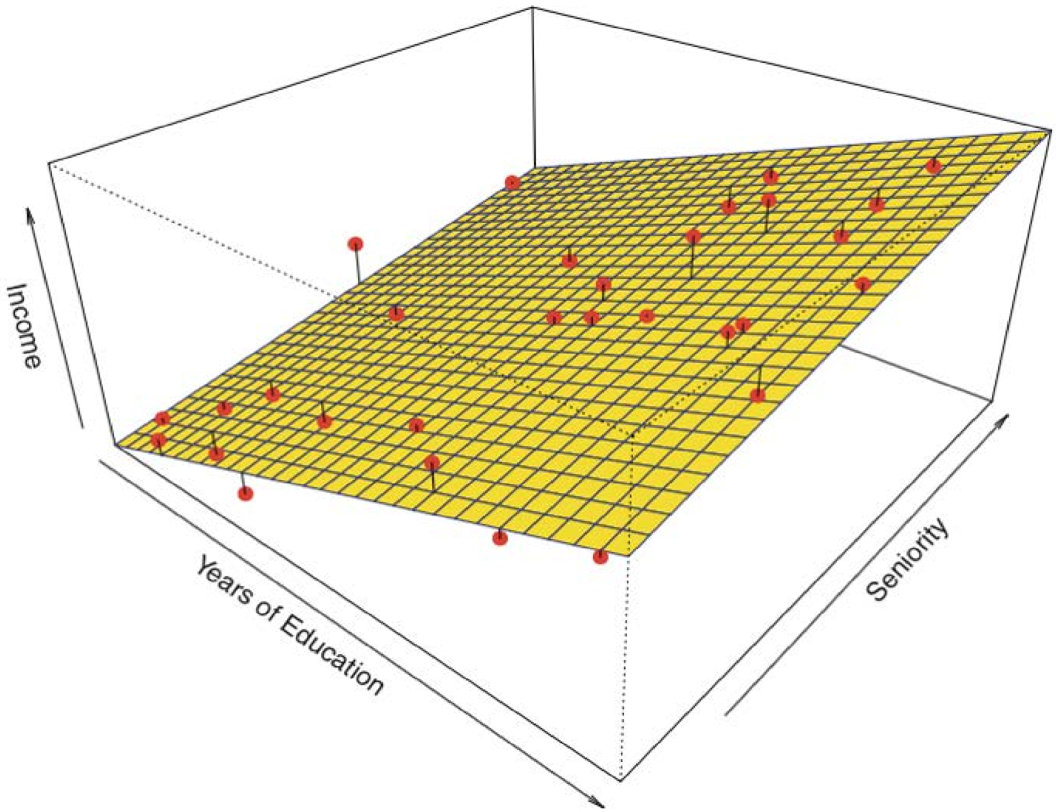
A set of n data couples (training set)

$$\{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$$

is given

$$x \in \mathbb{R}^d, y \in \mathbb{R}$$

*x* is column vector



# Multiple linear regression

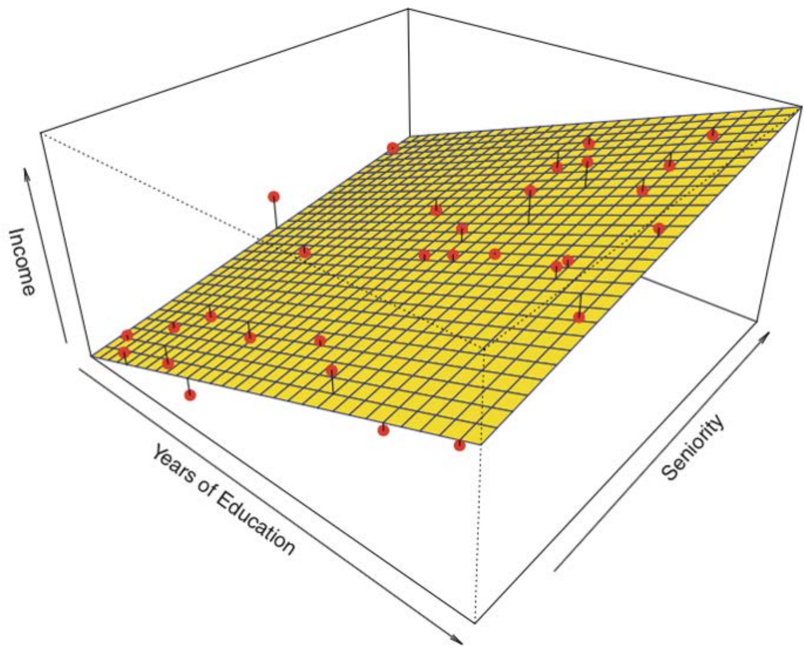
**Assume** that the unknown function that generates the data is linear and that there is a gaussian uncertainty affecting measurements in an additive way

$$y(x) = \theta_1^o + \theta_2^o x_2 + \cdots \theta_d^o x_d + \eta$$

Canonical form for system model

$$y(x) = x^T \theta^o + \eta$$

$$\theta^o \in \mathbb{R}^d \qquad \eta = N(0, \sigma_\eta^2)$$




Optimal parameters, grouped in a column vector, are unknown, as it is the variance of noise

# Multiple linear regression

We know that the unknown system model is linear. Which family of models should we consider to best fit the available data?

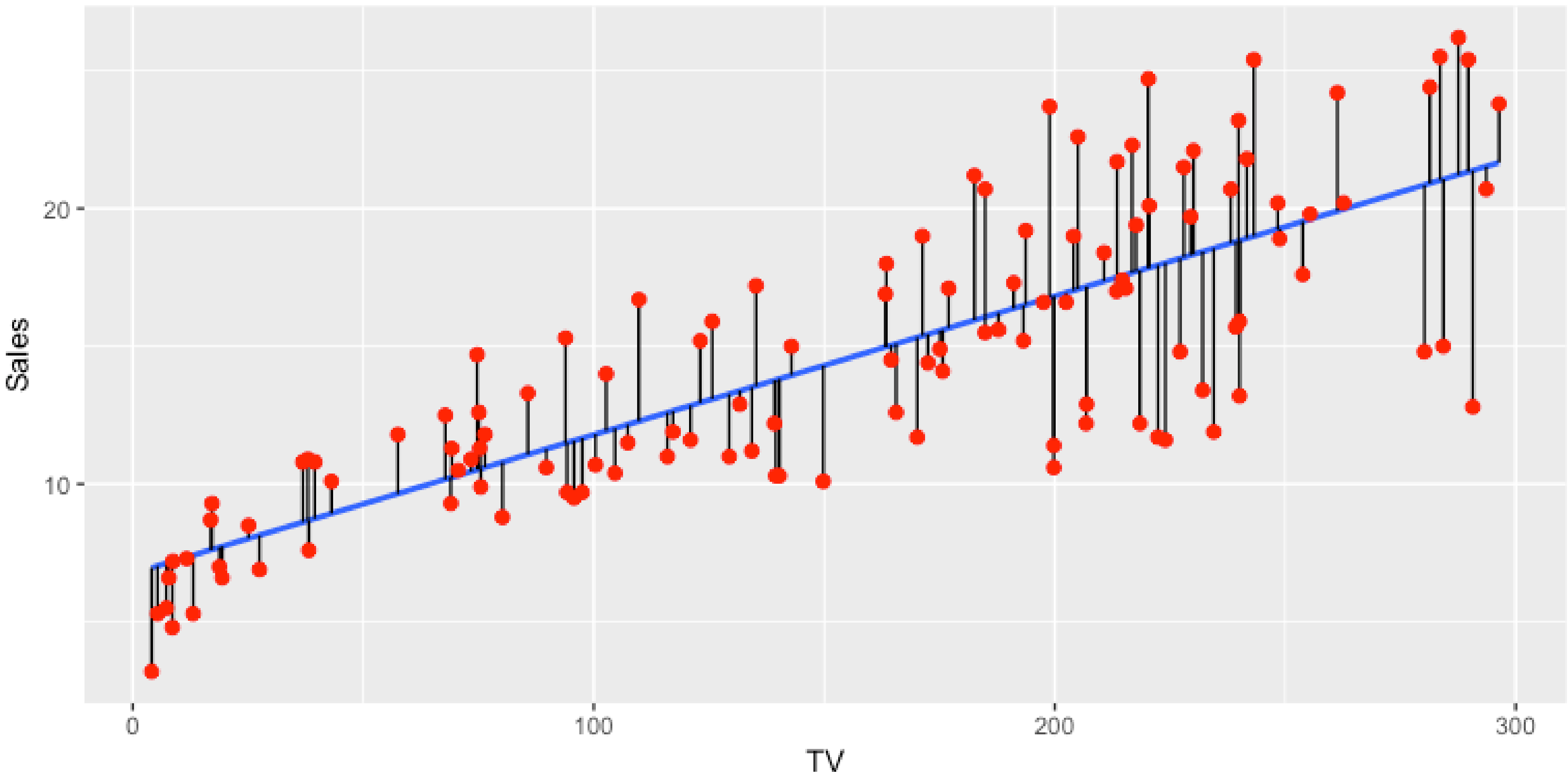
$$\hat{y}(x) = f(\theta, x) = x^T \theta$$

How to determine the best estimated parameter $\hat{\theta}$
so that we generate the model

$$f(\hat{\theta}, x) = x^T \hat{\theta}$$

approximating unknow function $x^T \theta^o$ ?

# Multiple linear regression




Idea: select the linear function that minimizes the average distance between given points and the linear function: we obtain the **Least Mean Square –LMS-** procedure

# Multiple linear regression

Performance function

$$V_n(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(y(x_i) - f(\theta, x_i)\right)^2$$

The parameter vector to be chosen is

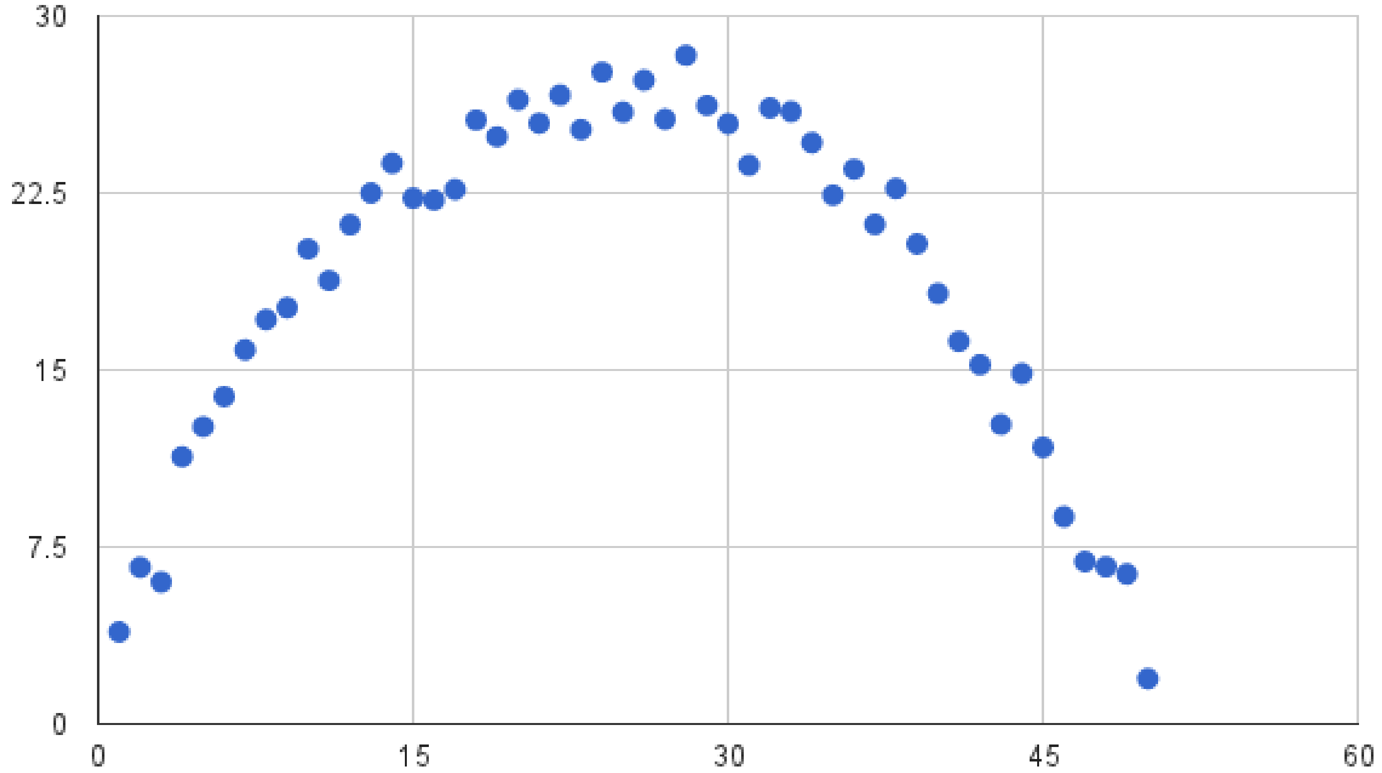
$$\hat{\theta} = \arg\min_{\theta \in \Theta} V_n(\theta)$$

# Linearity must be intended according to the parameters!!!

- Linear regression

$$y = a + bz + cz^2 + dz^3$$

$$\theta = [a, b, c, d]; x = [1, z, z^2, z^3]$$



# Multiple linear regression: how to estimate parameters

By grouping data as

$$Y = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \mathbf{y}_3 \\ \cdots \\ \mathbf{y}_n \end{bmatrix} \qquad \boldsymbol{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \mathbf{x}_3^T \\ \cdots \\ \mathbf{x}_n^T \end{bmatrix}$$

We can rewrite the loss function in a canonical form

$$V_n^*(\theta) = \sum_{i=1}^{n} \left(y(x_i) - x_i^T \theta\right)^2 = (Y - X\theta)^T (Y - X\theta)$$

# Multiple linear regression

Stationary points are those for which

$$\frac{\partial V_n^*(\theta)}{\partial \theta} = -2X^TY + 2X^TX\theta = 0$$

Therefore, the parameter vector minimizing the performance function is

$$\hat{\theta} = \left(X^TX\right)^{-1} X^TY$$

and the best approximating model is

$$f(\hat{\theta}, x) = x^T \hat{\theta}$$

# Multiple linear regression

- A computer program is said to **learn** from experience *E* with respect to some class of tasks *T* and performance measure *P*, if its performance at tasks in *T*, as measured by *P*, improves with experience *E*.
- Let's list the different actors

  - Task *T*: regression
  - Experience *E*: $\{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$
  - Performance measure *P*: Mean square error
  - Performance at task: $V_n(\theta) = \frac{1}{n} \sum_{i=1}^{n} (y(x_i) - f(\theta, x_i))^2$

# Performance at task: mmh….

- In a typical classroom the teacher provides solution examples/instances related to a concept, e.g., what a cat is.



- However, at exam time, the problems the teacher provides you to test your understanding are not identical to the ones you saw during the course. E.g.,


# Performance at target: mmh….

- Instead, we propose different instances



- Why?
- Performance at task assessed according to

$$V_n(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(y(x_i) - f(\theta, x_i)\right)^2$$

is biased to the training set

# How to measure performance at task?

- Much more to come later in the course
- For now, based on our exam experience, we consider another data set, called test set,

$$\{(\bar{x}_1, \bar{y}_1), (\bar{x}_2, \bar{y}_2), \cdots, (\bar{x}_l, \bar{y}_l)\}$$

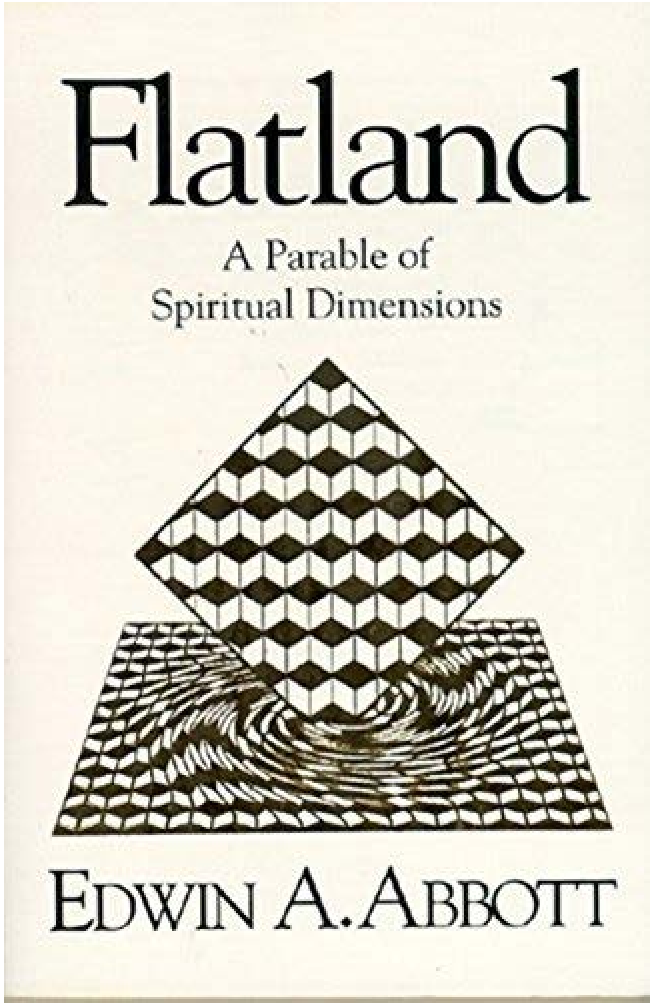
- and evaluate performance on it

$$V_l(\hat{\theta}) = \frac{1}{l} \sum_{i=1}^{l} \left(\bar{y}_i - \bar{x}^T \hat{\theta}\right)^2$$

The procedure is named cross-validation

# Nonlinear regression

***«I like straight lines but sometimes curves are appreciable»***



# Non-linear regression: formalization

The *stationary* process generating the data

$$y = g(x) + \eta$$

provides, given input $x_i$ output

$$y_i = g(x_i) + \eta_i$$

Collect a set of couples (training set)

$$Z_N = \{(x_1, y_1), \ldots, (x_N, y_N)\}$$

The goal of supervised learning is to design the simplest approximating model able to explain past $Z_N$ data and future instances provided by the data generating process.

# Non-linear regression: formalization

Model unknown function $g(x)$

with parameterized family of models $f(\theta,x)$

Consider loss function

$$L(y(x), f(\theta,x))$$

e.g.,

$$L(y(x), f(\theta,x)) = (y(x) - f(\theta,x))^2$$

# The traditional statistical approach

The structural risk

$$\bar{V}(\theta) = \int L(y, f(\theta, x))\, p_{x,y} dxy$$

$$\theta^o = \arg\min_{\theta \in \Theta} \bar{V}(\theta)$$

The empirical risk

$$V_N(\theta) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(\theta, x_i))$$

$$\hat{\theta} = \arg\min_{\theta \in \Theta} V_N(\theta)$$

# Non-linear regression: formalization

Determine the parameter estimate through a minimization problem

$$\hat{\theta} = \arg\min_{\theta \in \Theta} V_N(\theta)$$

carried out by a learning procedure

$$\theta_{i+1} = \theta_i - \varepsilon_L \frac{\partial V_N(\theta)}{\partial \theta}\Big|_{\theta_i}$$

and get model

$$f(\hat{\theta}, x)$$

# Comments

The risk associated with the model (generalization ability, performance at task) can be decomposed in three terms

$$\bar{V}(\hat{\theta}) = \left(\bar{V}(\hat{\theta}) - \bar{V}(\theta^{o})\right) + \left(\bar{V}(\theta^{o}) - V_{I}\right) + V_{I}$$

# Comments

The inherent risk

$$V_I$$

depends only on the structure of the learning problem.  This term can be improved only by improving the problem itself (e.g., by reducing the instrumentation noise)

# Comments

The approximation risk

$$\bar{V}(\theta^{o}) - V_{I}$$

depends only on how close the approximating model family is to the process generating the data. As such, a more appropriate model family reduces the risk

# Comments

The estimation risk

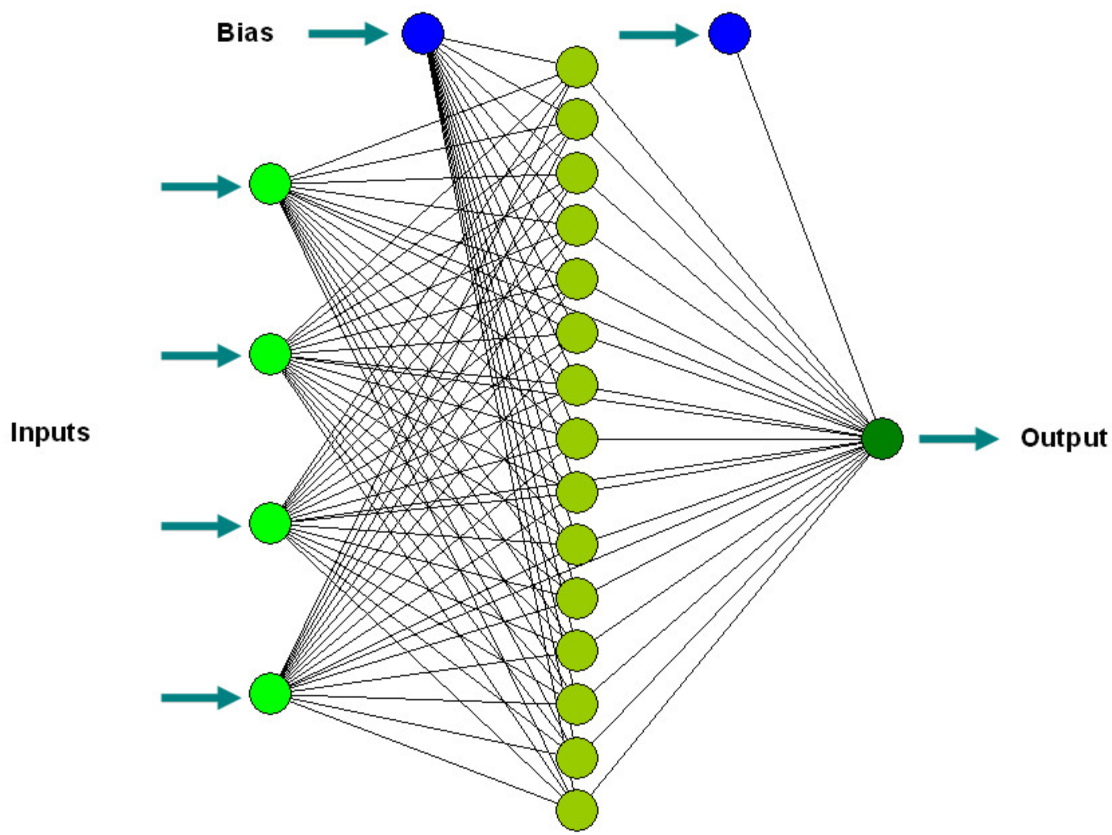
$$\bar{V}\left(\hat{\theta}\right) - \bar{V}\left(\theta^{o}\right)$$

depends on the effectiveness of the learning procedure. As such, a more effective learning algorithm reduces the risk

$\boxed{\text{All consistent learning procedures are equally effective}}$

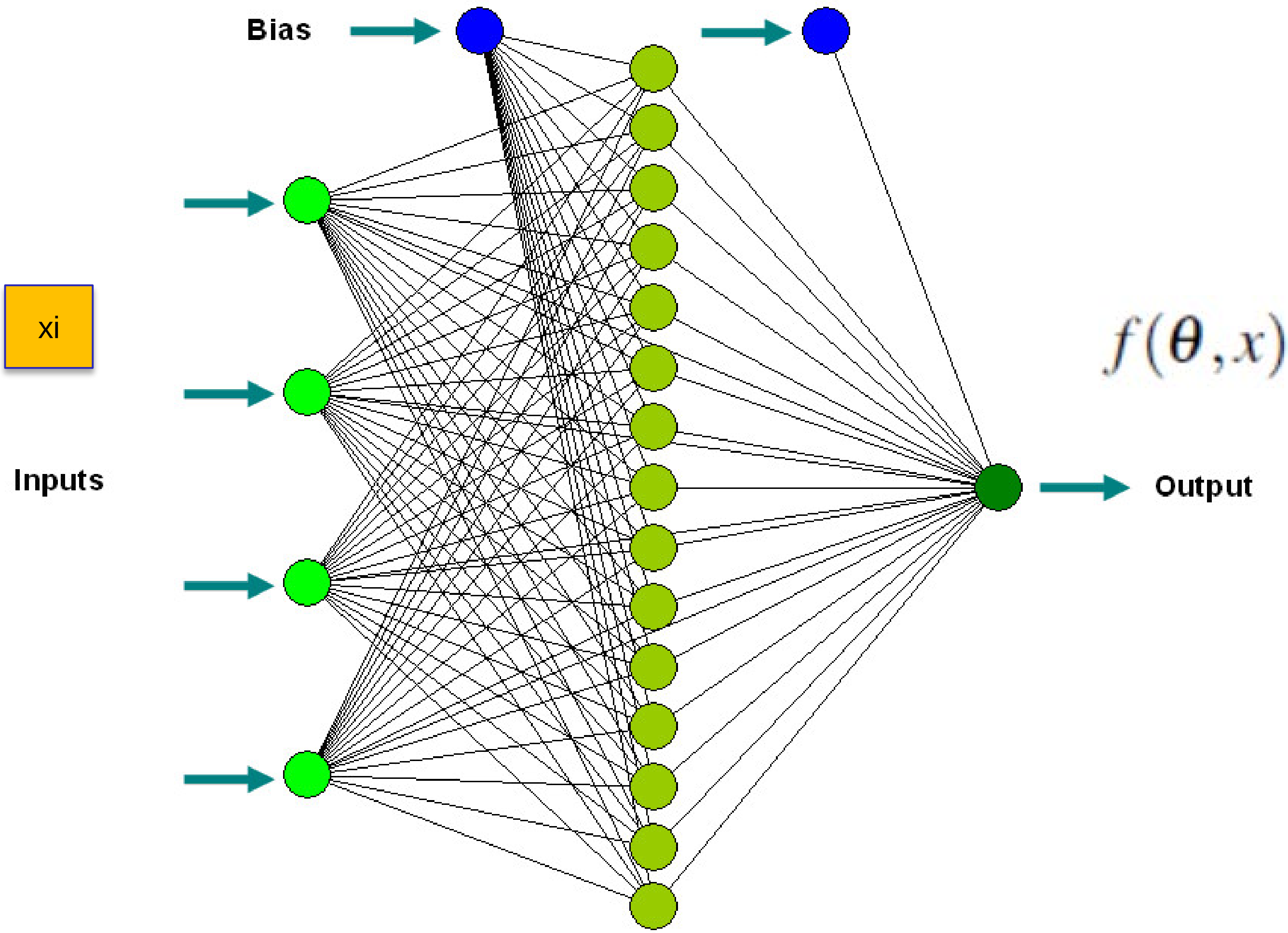
# Feed-forward neural networks

***«An interesting computational paradigm»***



# Feedforward Neural Networks

Not rarely $f(\theta, x)$ is chosen as

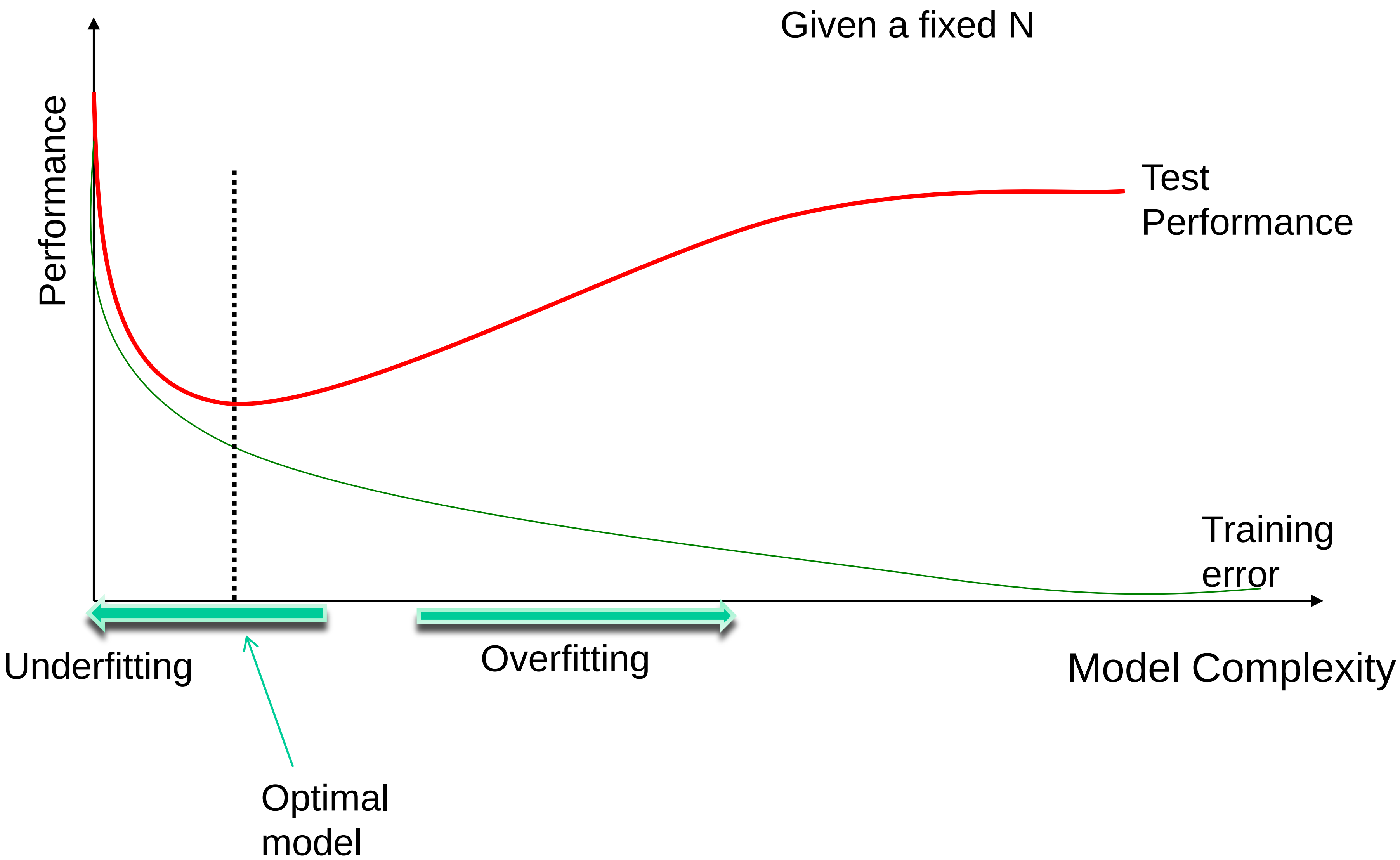


# Universal approximation theorem

A feedforward network with a single hidden layer containing a finite number of neurons and a linear output neuron approximates any continuous function defined on compact subsets

K. Hornik, "Approximation Capabilities of Multilayer Feedforward Networks", *Neural Networks*, No.4 Vol. 2, 251–257, 1991

# Approximation performance vs. Model complexity



# Controlling the model complexity

- Trial & error
- Early stopping
- Dropout
- Tikhonov regularization