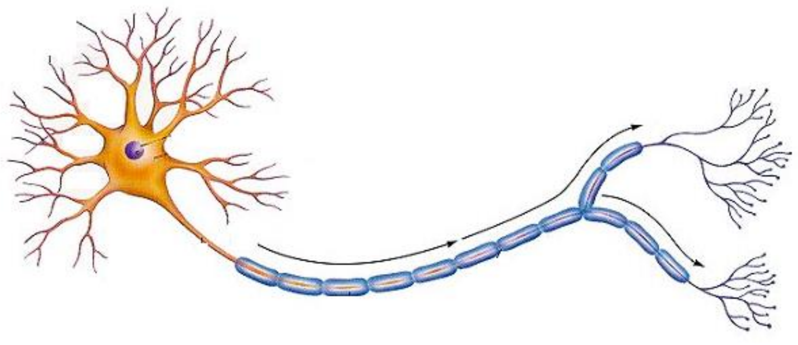
Artificial Neural Networks

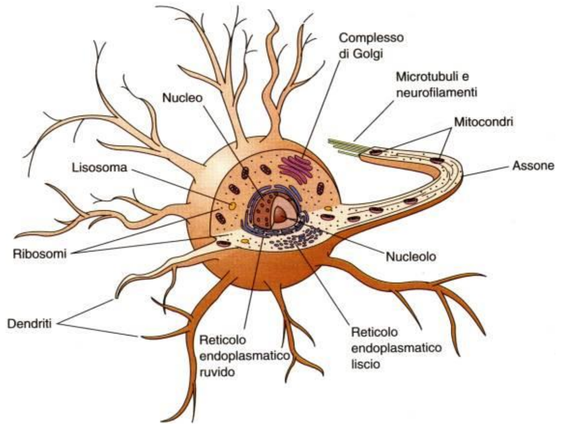
Davide Manca

Lecture 5 of "Dynamics and Control of Chemical Processes" – Master Degree in Chemical Engineering

Introduction

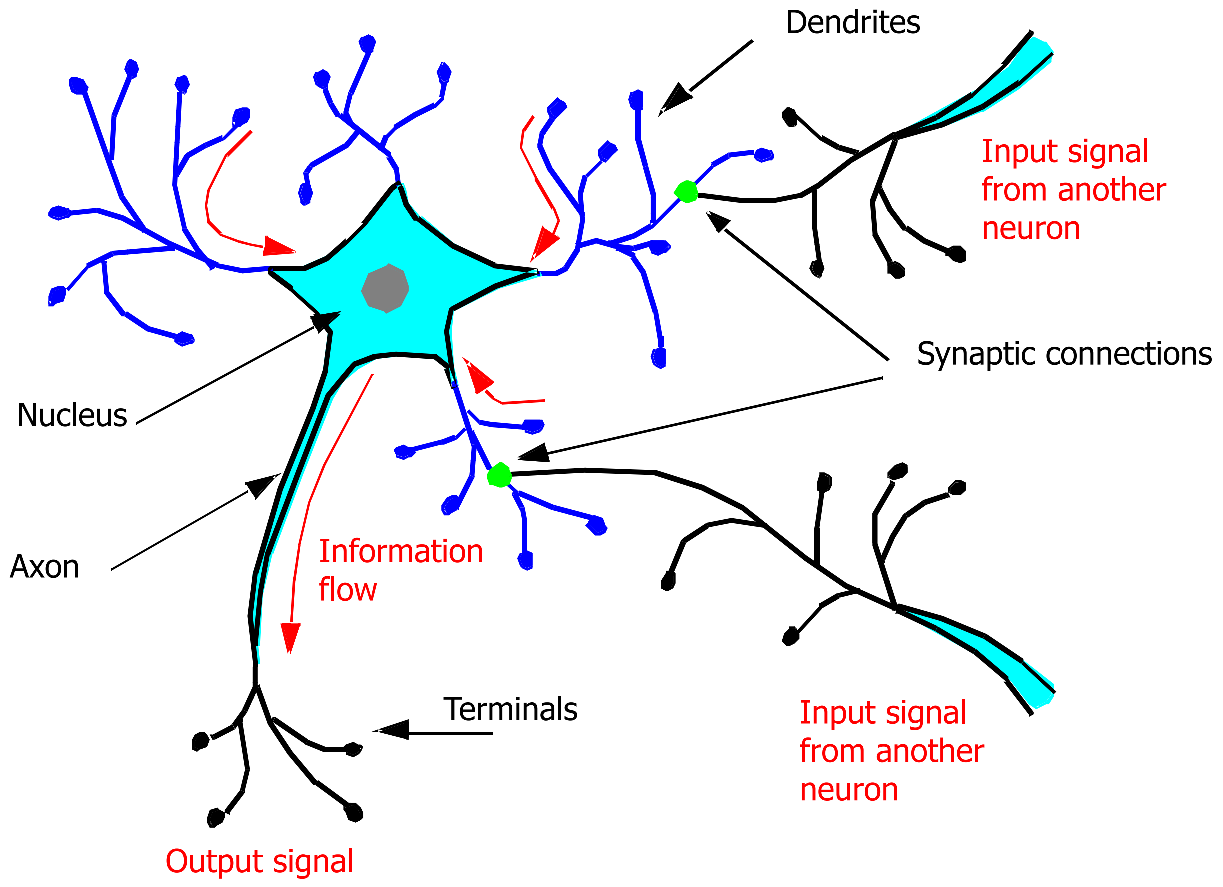
- **ANN**: Artificial Neural Network
- An ANN is an engine designed to simulate the operation of the **human brain**. It is physically implemented with electronic components or simulated on digital computers via software (Haykin, 1999).
- The ANN consists of simple elements (**neurons**, nodes, units).
- The information, i.e. the **signals**, flows through the neurons by means of connections.
- The **connections** are **weighed** to adjust the information flow.
- The information, the weighed signals, is loaded in the neurons and an **activation function** also known as transfer function (either linear or non-linear) transforms it into the output signal.



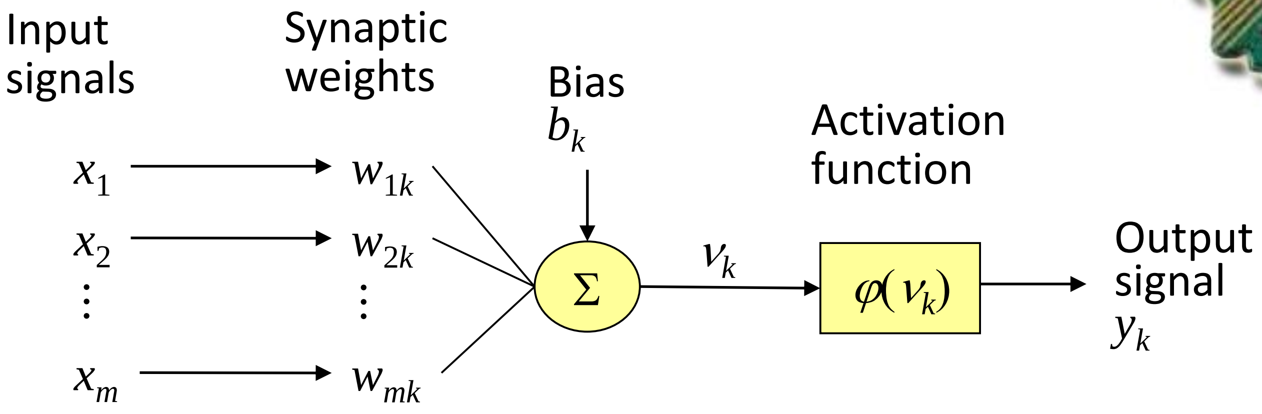

Complesso di Golgi
Microtubuli e neurofilamenti
Nucleo
Mitocondri
Lisosoma
Assone
Ribosomi
Nucleolo
Dendriti
Reticolo endoplasmatico ruvido
Reticolo endoplasmatico liscio

The biological neuron

- The central part of the neuron, the **nucleus**, sums the input signals coming from the **synapses**, which are connected to the **dendrites** of other neurons. When the signal reaches a limit threshold the neuron produces an output signal towards other neurons. In other words, the neuron "fires".



The abstract neuron



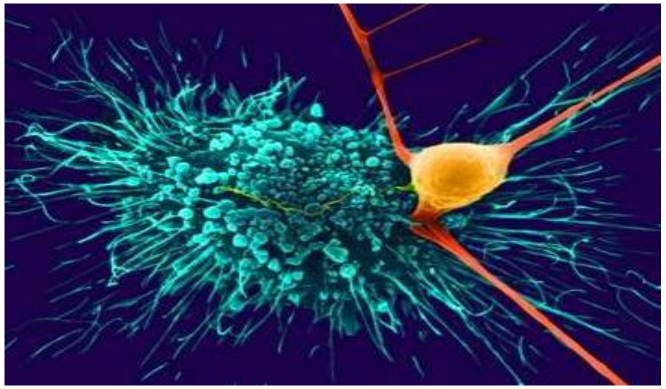
$$\mu_k = \sum_{j=1}^{m} w_{jk} x_j \Rightarrow v_k = \left(\mu_k + b_k\right) \Rightarrow y_k = \varphi\left(v_k\right)$$

A comparison of neurons and transistors

- The basic unit of the human brain is the neuron.
- The basic unit of a processor, CPU, is the transistor.
- Characteristic times:
  - neuron $\sim 10^{-3}$ s;
  - transistor $\sim 10^{-9}$ s.
- Energetic consumption for a single operation:
  - neuron $\sim 10^{-16}$ J;
  - transistor $\sim 10^{-7}$ J.
- Number of basic elements:
  - in the brain $\sim 10^{10}$ neurons;
  - in a processor $\sim 10^{9}$ transistors;
- Synapses number (interconnections) in the human brain: $\sim$60,000 billion.
- The **brain** can be compared to a complex system, for the information processing, that is highly **non-linear** and **parallel**.

More in depth…

- An ANN may be considered as a system capable of producing an answer to a question or provide an output after an input data.
- The input → output link, which is the transfer function of the network, is not explicitly programmed. On the contrary, it is simply obtained through a learning process grounded on empirical data.
- Under the supervised method, the learning algorithm modifies the characteristic parameters, i.e. weights and biases, so to bring the network forecast near to the supplied real experimental values.
- Therefore, the main features of the network come close to those of the brain:
  - capability to learn from the experience (measure of in-the-field data);
  - high flexibility to interpret the input data (*i.e.* "noise resistance" or "capability to interpret noisy data");
  - fair extrapolation capability.



The history

In diſem Turnier, Turnierte Ein Edler Ritter, heinrich von Gemmingen 1080

- The ANN history begins with **McCulloch** e **Pitts** in **1943**. McCulloch was a **psychiatrist** and a **neuro-anatomist**, whilst Pitts was a **mathematician**. The collaboration of these researchers took to the description of the logic calculus of the neural network, which combines neurophysiology to mathematical logics.
- In **1949** **Hebb** postulated the first law of self-organized learning (*i.e.* learning without a trainer/teacher).
- In **1958** **Rosenblatt**, in his perceptron work, proposed the first model of supervised learning (*i.e.* learning with a trainer/teacher).
- The perceptron introduced by Rosenblatt laid the foundations for the following neural networks, which can learn any kind of functional dependency.
- In **1969** **Minsky** and **Papert** showed mathematically that there are some basic limits to the learning capability of a single neuron. This result took the scientific community to temporarily leave the ANN research.
- In the **'80** of last century, the ANN gained new interest after the introduction of one or more **intermediate levels**. Such ANNs, capable of fixing their own errors, surpassed the perceptron limits of Rosenblatt and revitalized the research in that sector.

Main features

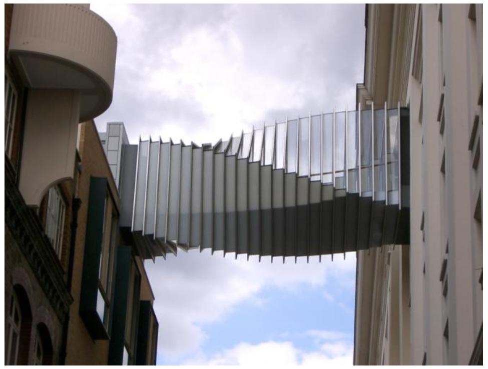
1. **Non-linearity**: the presence of non-linear activation functions makes the network intrinsically non-linear.
2. **Analysis and design uniformity**: the neurons are a common ingredient for ANNs. This makes viable sharing both learning theories and algorithms in different application fields.
3. **Input-output map**: the network learns by examples based on a sequence of input-output data (*i.e.* supervised learning). No *a priori* assumption is made on the model parameters.
4. **Adaptivity**: the ANN can adapt its weights to the changes of the surrounding environment. In particular, an ANN trained to operate in a given environment, can be trained again to operate in another one. In addition, if the network is operated in a dynamically evolving environment, it can be designed to change/modify its weights in real-time (*e.g.*, adaptive elaboration of the signals, adaptive control).
5. **Parallelism**: thanks to its structure, an ANN can be implemented directly at the hardware level for specific high-efficiency computational purposes.

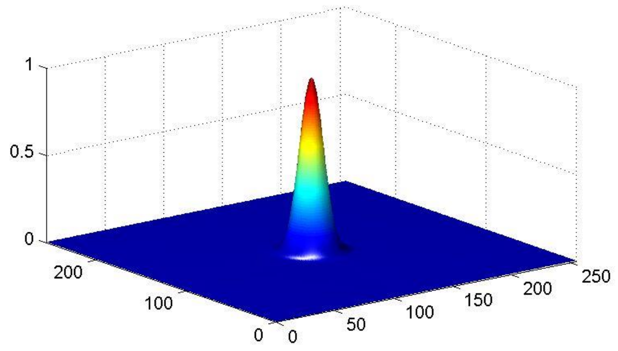
Main features

6. **Fault tolerant**: an hard-wired ANN shows a good tolerance to faults. Its performance deteriorates slowly in case of malfunctions. This is due to the high relocation of the information among the neurons, which reduces the risk of catastrophic faults and guaranties a gradual deterioration of its performance.
7. **Neuro-biological analogy**: the ANN design is motivated by its analogy with the human brain that is the living proof that the parallel, *fault tolerant*, computation is not only physically viable, but also fast and powerful.

Network features

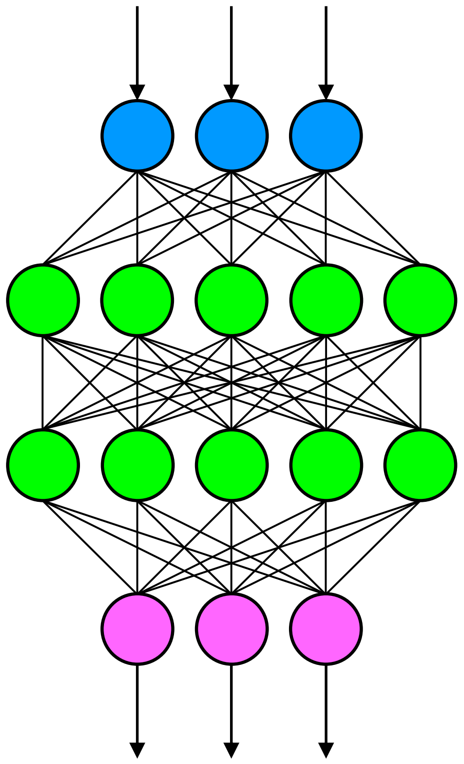
- There are three issues that describe and characterize an ANN:
  - **network architecture**. Structure and typology of the node connections, number of intermediate levels;
  - **learning method**. Evaluation of the intensity of the node connections (weights and biases computation);
  - **functional dependency** the link the input to the output of the neuron. Activation function typology.





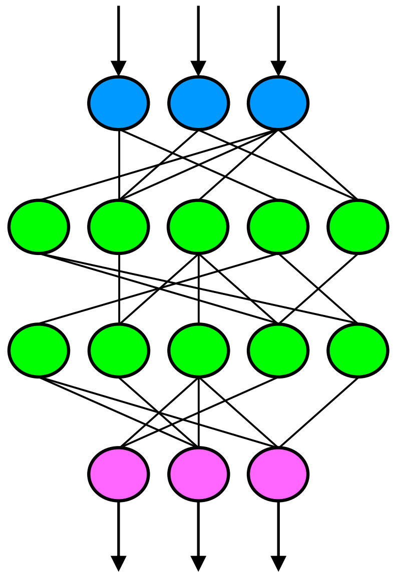
Network architecture

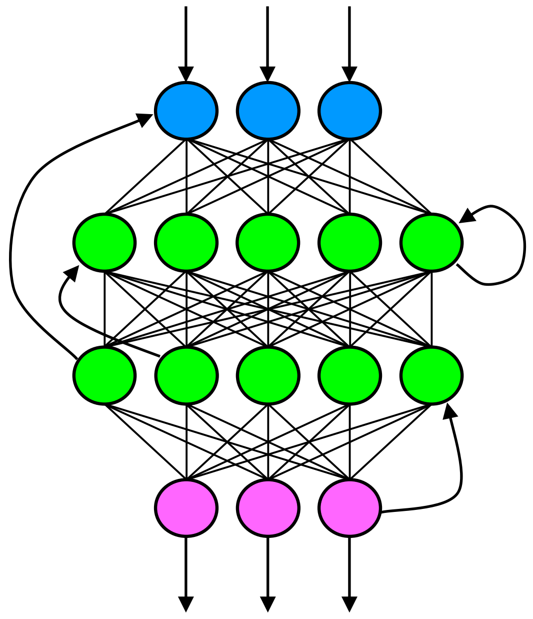
- The **input nodes** are the data source. They are not considered as a real level as they do not contribute to the data processing.
- The **output nodes** produce the final answer of the network.
- The nodes belonging to the **intermediate hidden layers** (if present), together with the output nodes, transform the input signal into the output one.
- The number of nodes for each level, the number of hidden layers, and the connections among the nodes define the **architecture** of the ANN.



Network architecture

- There are several ANN typologies:
  - networks without hidden layers, *i.e.* input-output networks;
  - networks with one or more hidden layers;
  - fully connected networks;
  - partially connected networks;
  - purely feedforward networks, unidirectional flow;
  - recursive networks with feedback flow;
  - stationary networks;
  - adaptive networks, with dynamic memory;
  - …





Network architecture

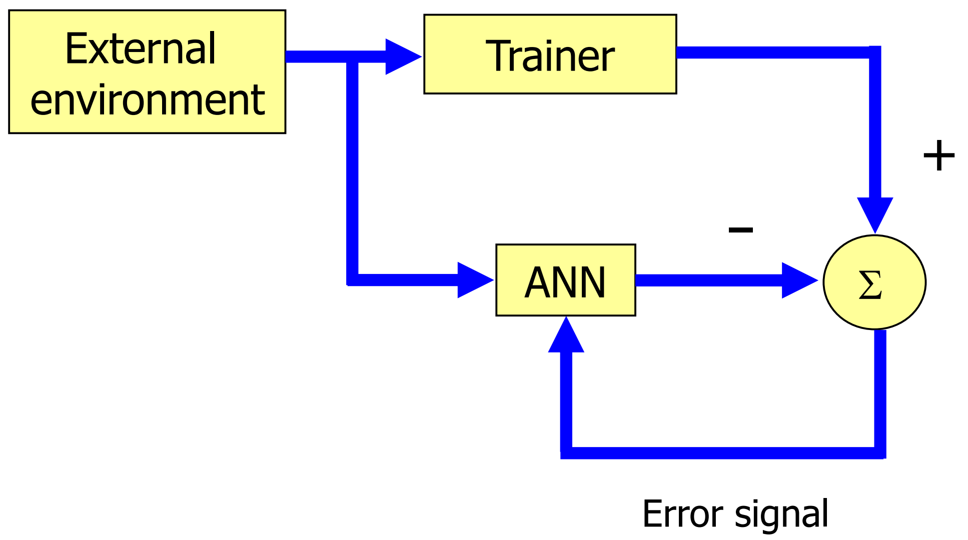
- The most basic structure of an ANN consists of:
  - **one input layer**;
  - **one or more hidden layers**;
  - **one output layer**.
- Every layer has a variable number of nodes. Often, the whole ANN structure is triangular moving from the top (input) to the bottom (output).
- **Cybenko**, 1989, showed that a single hidden layer is sufficient to describe any continuous function.
- **Haykin**, 1992 (1999), added that the choice of a single hidden layer does not always produce the best possible configuration (mainly in case of identification of highly non-linear processes).

George Cybenko

Simon Haykin

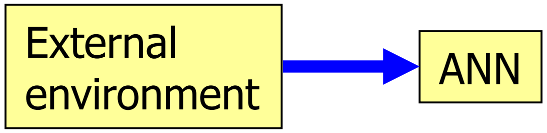
Learning methods

- The network training, which consists in adjusting (*i.e.* evaluating) the optimal synaptic weights, can happen according to two distinct methods:
  - **SUPERVISED TRAINING (with trainer/teacher)**
    - During the training, a trainer/teacher supplies the correct answers respect to an input data set.
    - The weights are modified as a function of the error made by the network respect to the real output data.



Learning methods

- **UNSUPERVISED TRAINING (without trainer/teacher)**
  - There is not a trainer/teacher. Therefore, the feedback contribution is missing.
  - The network discovers independently the correlations among the supplied input data.
  - The weights change along the training procedure according to an ***a priori* rule** which does not use the error respect the real environment (*i.e.* process).
  - The method uses a **measure** that is independent of the specific problem.
  - Some heuristic rules may be used to transform the external input signal into either an "**award**" or a "**punishment**".



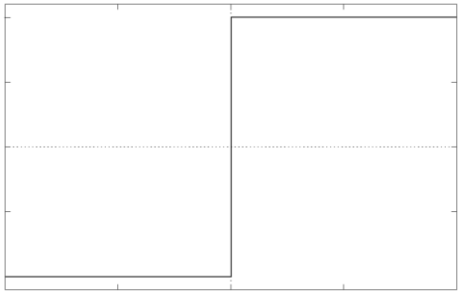
Activation function

- The activation function, $\varphi(v)$, defines the neuron output as a function of the activity level of its inputs.

$$\mu_k = \sum_{j=1}^{m} w_{jk} x_j \quad \Rightarrow \quad v_k = (\mu_k + b_k) \quad \Rightarrow \quad y_k = \varphi(v_k)$$

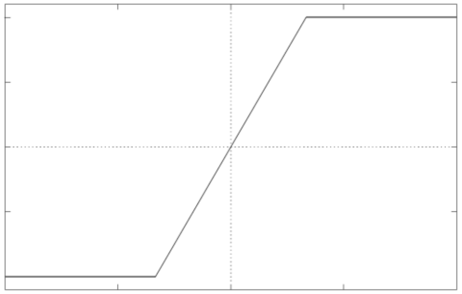
- **Step (or threshold) function**

$$\varphi(v) = \begin{cases} 1 & v \geq 0 \\ 0 & v < 0 \end{cases}$$



- **Linear piece-wise (or ramp) function**

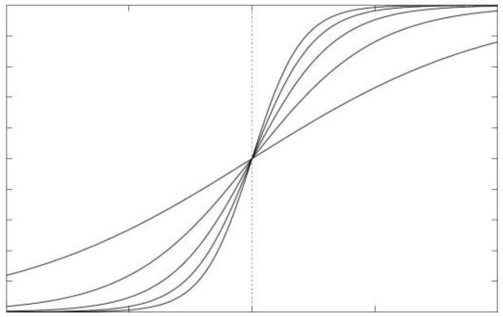
$$\varphi(v) = \begin{cases} 1 & v \geq 1/2 \\ \mu v & -1/2 \leq v \leq 1/2 \\ 0 & v \leq -1/2 \end{cases}$$



Activation function

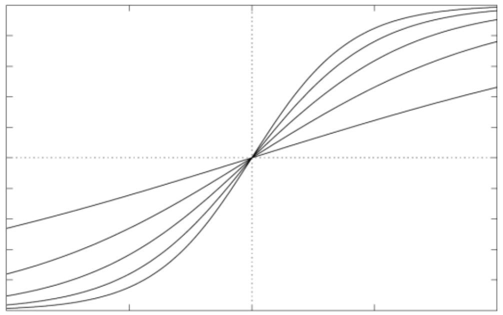
- **Sigmoid function**

$$\varphi(v) = \frac{1}{1+\exp(-\mu v)}$$



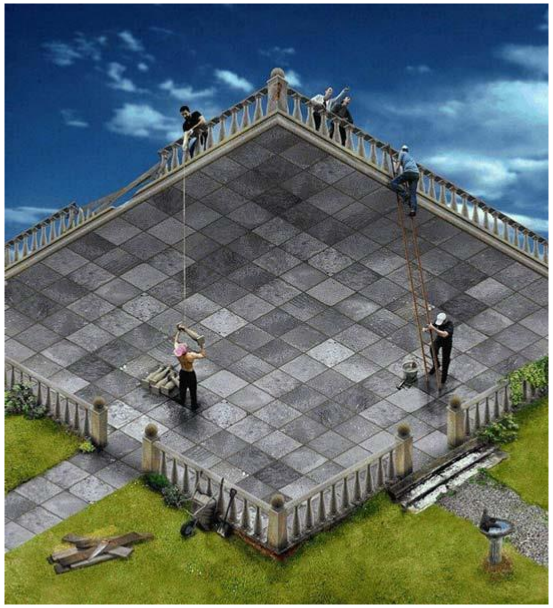
- **Hyperbolic tangent function**

$$\varphi(v) = \mathrm{Th}\left(\frac{\mu v}{2}\right) = \frac{1-\exp(\mu v)}{1+\exp(\mu v)}$$



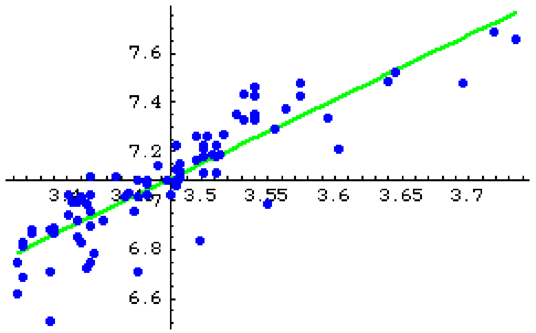
N.B.: the activation functions take over values belonging to the $0,\ldots 1$ or $-1,\ldots 1$ intervals. This is consistent with the fact that the inputs to the network are normalized to keep limited and controlled the signals flowing in the network.

BP learning



- The most renowned method for supervised training is the Error **B**ack **P**ropagation Algorithm, **BP**.
- This algorithm comprises two phases:
  - **forward** phase;
  - **backward** phase;
- In the **forward** phase, the learning patterns (*i.e.* input-output data pairs) are presented to the input nodes. The network response depends on the hidden layer(s) downwards the output nodes. During this phase both **weights** and **biases** remain **constant** (*i.e.* unchanged).
- In the **backward** phase, the error between the real process and the network output is first evaluated and then back propagated (upwards) through the nodes of the different layers. Suitable update formula **modify** the **weights** and **biases** up to the first input layer of the network.

BP learning

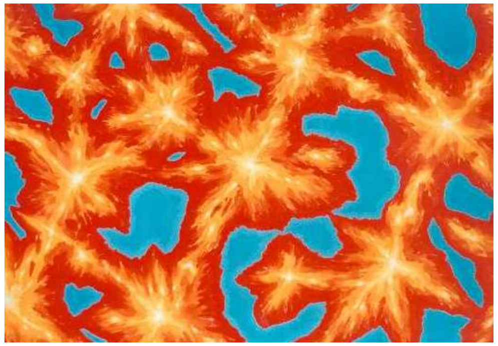


- Somehow, the weights update along the training procedure may be seen as a **parameter regression** that is the optimization of an objective function that **minimizes the average error** made on the set of learning patterns.
- A critical point is the **overfitting** problem (*aka* **overlearning**).
  The risk is that the network learns in an optimal way the input-output patterns provided along the training procedure and loses the generalization capability to deal with input data that have not been supplied yet.
- A **validation** procedure of the training procedure (**cross-validation**) is mandatory. This is based on a set of input-output patterns different from that used during the training procedure.

Further parameters

- The training procedure is based on two further parameters, both limited in the $0, \ldots 1$ interval:
  - **learning rate**, α
    - it determines the learning rate of the ANN: high values increase the learning process but may bring on convergence problems and instability.
  - **momentum factor**, β
    - it accounts for the dynamic evolution of the weights: its role consists in increasing the procedure velocity while keeping high values of the learning rate without incurring into oscillations and convergence problems.

An example

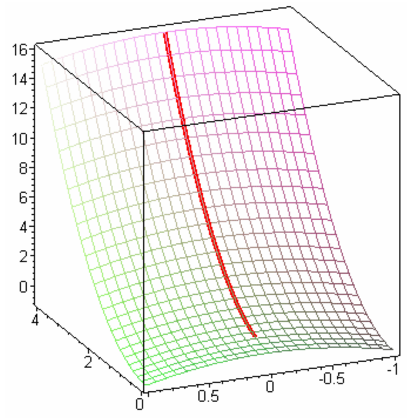


```
Neural network type:   Back propagation with momentum factor
Activation function:   Logistic sigmoid + linear function
Learning Algorithm :   History Stack Adaptation

  4  ! Layers  number (including input and output layers)
  2  ! Neurons number at level: 1
  4  ! Neurons number at level: 2
  2  ! Neurons number at level: 3
  1  ! Neurons number at level: 4
---------------------------------------------------------
  200                    ! Total learning patterns
 .50000000               ! alfa
 .90000000               ! beta
 .10000000               ! mu
---------------------------------------------------------
 Biases for each layer
 Layer n.   2   -1.10191428411417    -.306607804280631   .396983628052738   .229701066848763E-01
 Layer n.   3    1.32438158863708     .895365411961350
 Layer n.   4    12.1461254510109
---------------------------------------------------------
 Weights for each layer
 Layer n.   1 --> 2
  -.522050637568751    .247668247408662       1.36848442775682        .784256667945155
  -2.36688633725843    -2.47312321369078      1.81072640204618        .871974901958030
 Layer n.   2 --> 3
  1.78023653514858     1.68340336532185
  -.751951327532474    .906457768573295
  -1.66948863690154    -.743060896398274
  -.766335553627057    -.263315123659310
 Layer n.   3 --> 4
  -4.61959708139503
  -2.87806024443973

 End of neural network
---------------------------------------------------------
```

Identification algorithms

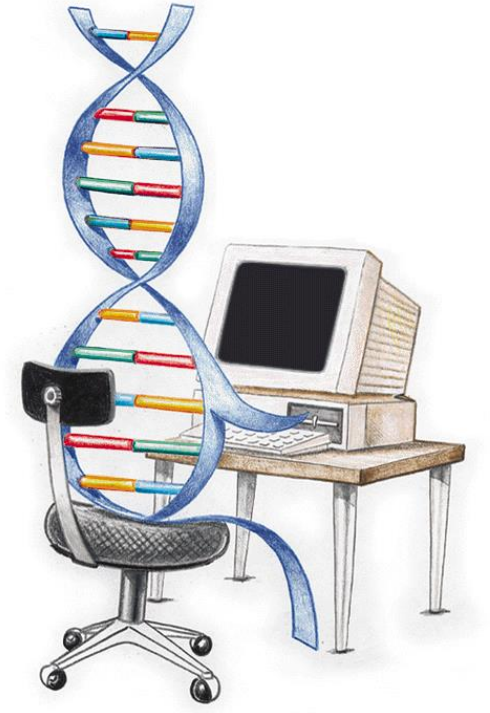
- **LOCAL SEARCH**
  - **First order** methods
    - **Gradient** (steepest descent)
      - rather simple;
      - possible local minima;
      - possible divergence along the search.

  - **Second order** methods
    - **Levenberg-Marquardt**
      - more complex;
      - possible local minima;
      - higher memory allocation;
      - longer training times;
      - deeper minimization of the objective function.



Identification algorithms

- **GLOBAL SEARCH**
  - Simulated Annealing;
  - Genetic algorithms;
  - Tabu search;
  - Multidimensional specific methods (robust methods).
  - Features:
    - complex implementation;
    - high cost in terms of memory and CPU time;
    - Identification of the global minimum.



Training methods

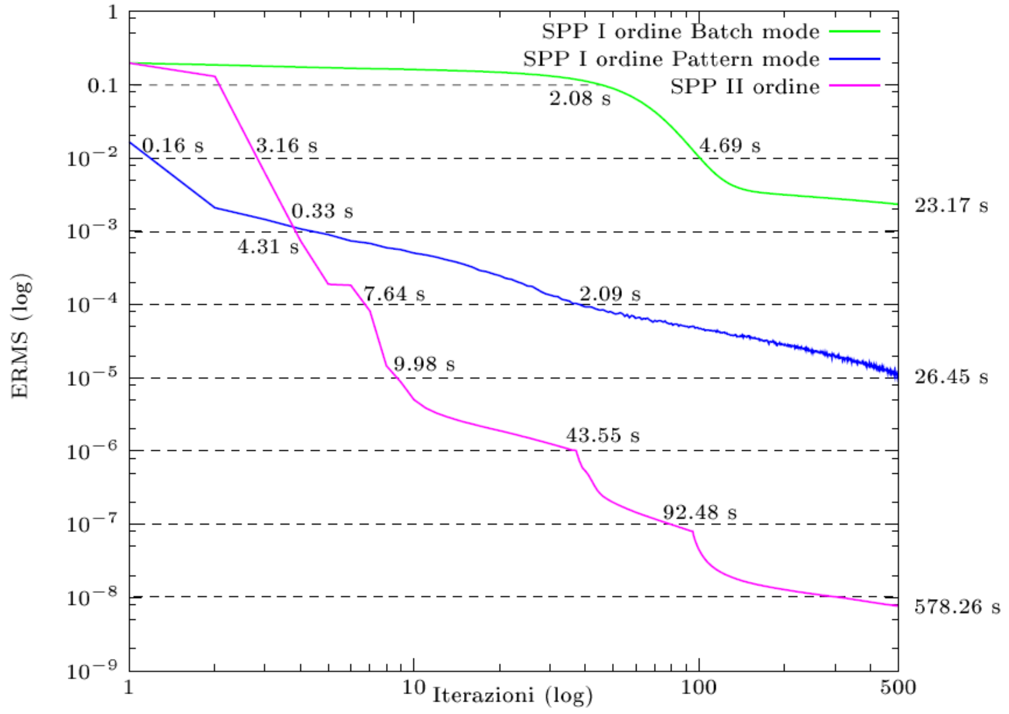
- Once the learning set is available, it is time for the network training.
- The provision/implementation of a whole set of patterns is called **epoch**.
- The training is based on the iterative provision of the epochs according to a **stochastic sequence** of the patterns to avoid any specific learning and any loss of generality.
- **PATTERN MODE training**
  - The update of both weights and biases is made **after every supplied pattern** to the network. Every pattern undergoes a **forward** and **backward** procedure. If a training set comprises N patterns, then there are N weight and bias updates for each epoch.
- **BATCH MODE training**
  - The update of both weights and biases is made **only after having supplied all the patterns** that is at the end of the epoch. The **forward** procedure supplies N patterns. The N local errors are stored together with their local gradients. Eventually, these values are averaged and the **backward** procedure is carried out.

Convergence criteria

There are not universal criteria to assess the network learning.

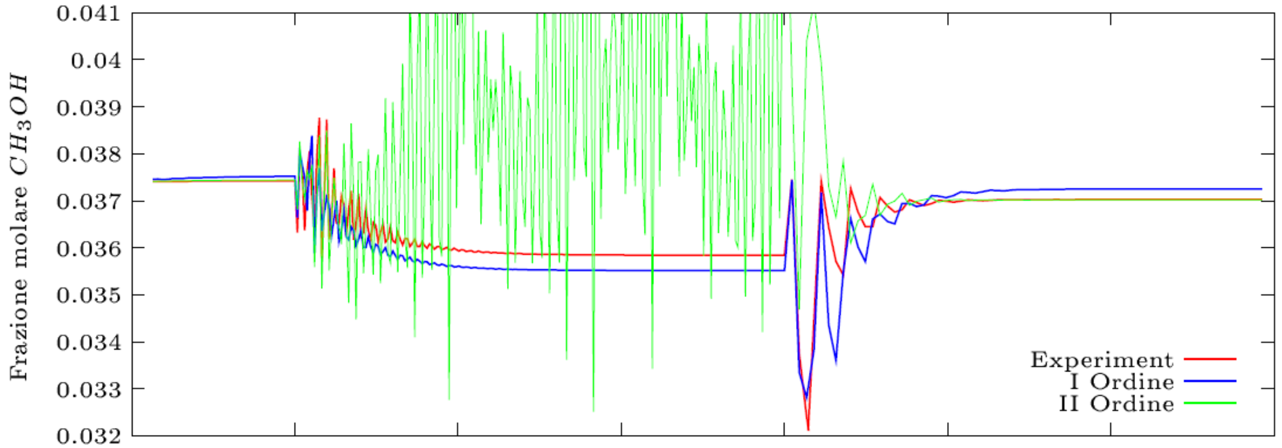
- It is worth focusing the attention on the identification degree respect to the set of the **training pattern**. For instance, it is possible to check:
  - If the weights and biases vary a little from an iteration and the following one;
  - The average error;
  - The decrease of the average error from an iteration and the following one.

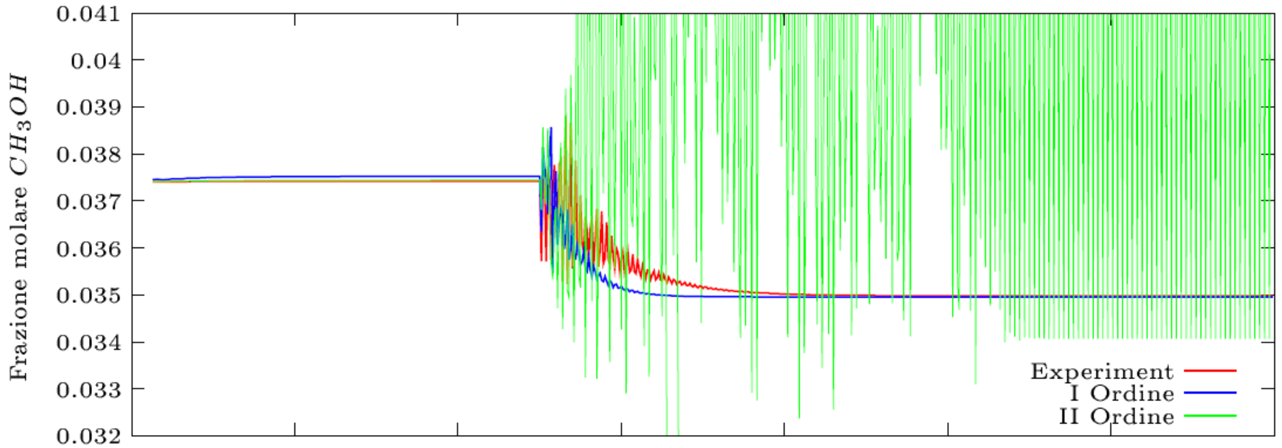
- The real risk for these control techniques of convergence is the **overfitting** of the network.



Overfitting examples

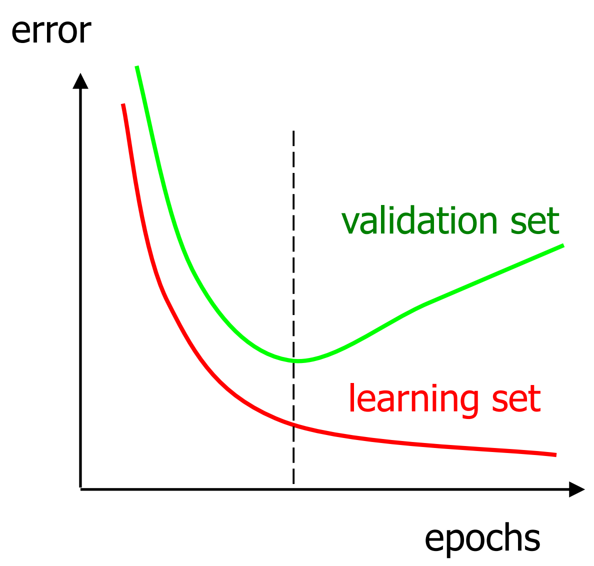
- **Network of simulated moving-bed reactors for the methanol synthesis.**

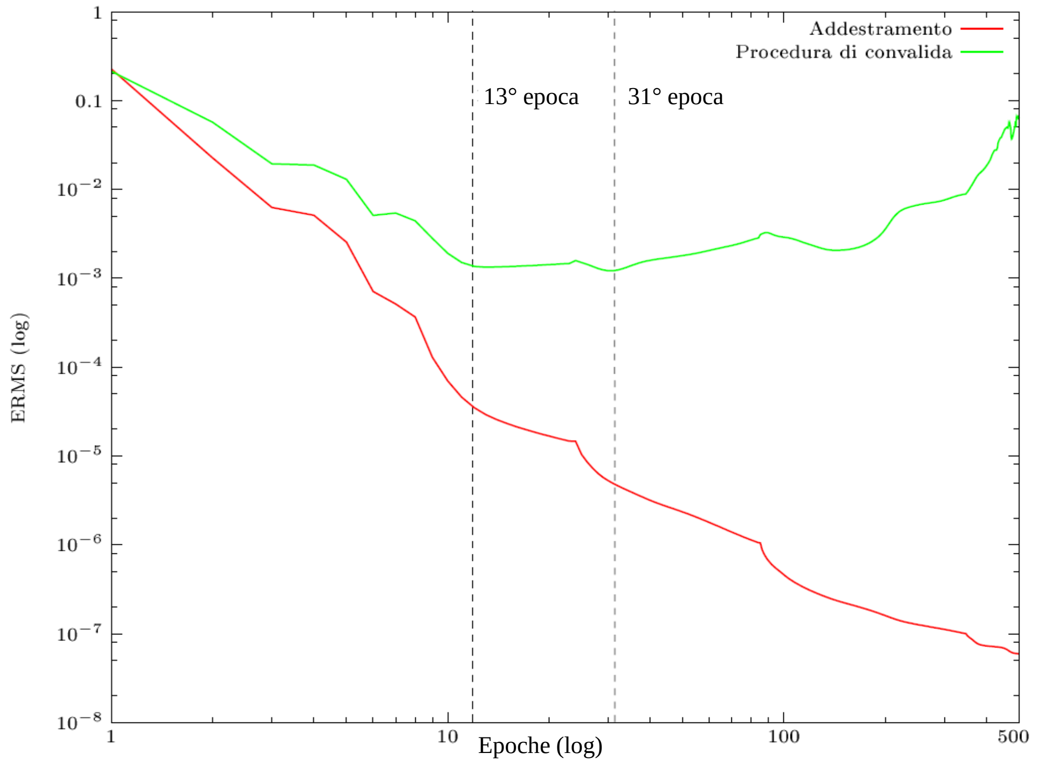




Convergence criteria

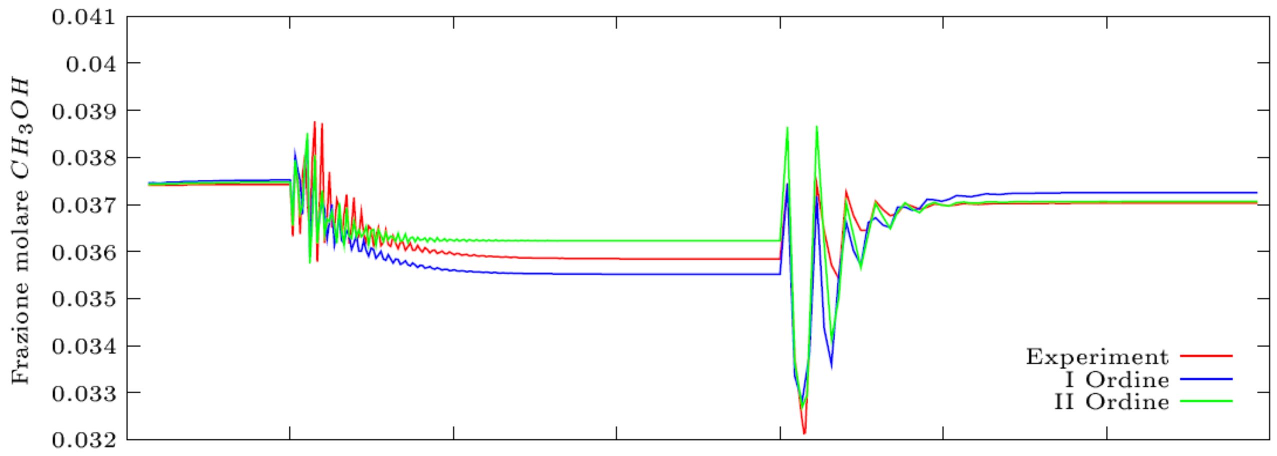
- It is worth focusing on the identification quality respect to the set of **validation patterns**. The average error is checked respect to the validation data set rather than the learning data set.

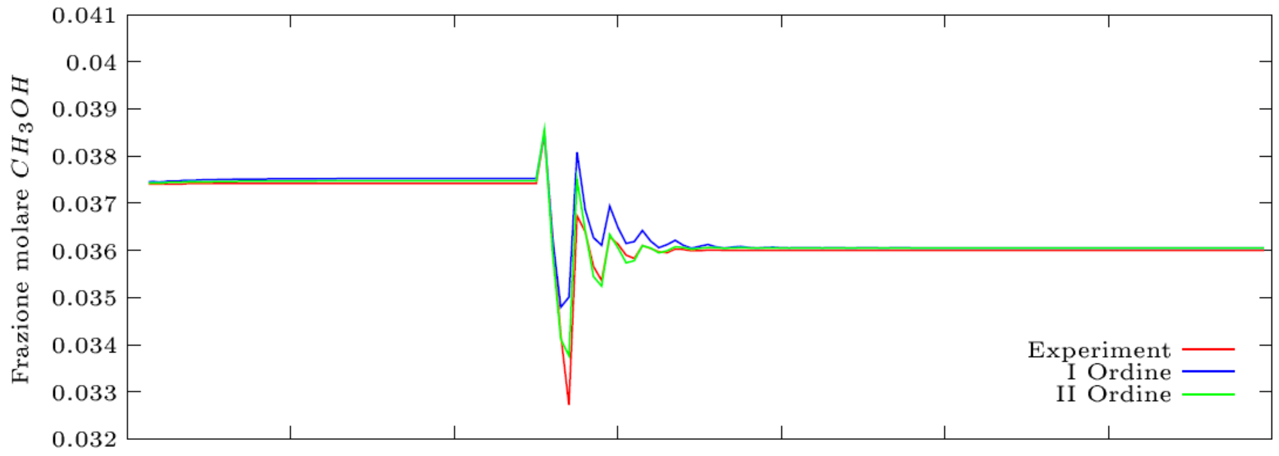




Overfitting removal

- **Network of simulated moving-bed reactors for the methanol synthesis.**





Static and dynamic approach

- An ANN can identify the link (*i.e.* dependency) between the input and output variables of the black-box process.
- The input-output dependency may refer to some **stationary conditions** or better to some functional dependencies that **do not depend** neither explicitly nor implicitly on the **time**.
- In turn, the network can identify the **dynamic behavior** of a process. In this case, the network must be able to describe the system dynamics respect to external disturbances on the input variables.
- As it happens for the ARX, ARMAX, and NARX systems, the **input variables to the network** are not only those of the process inputs but there are also some output variables referred to past values.

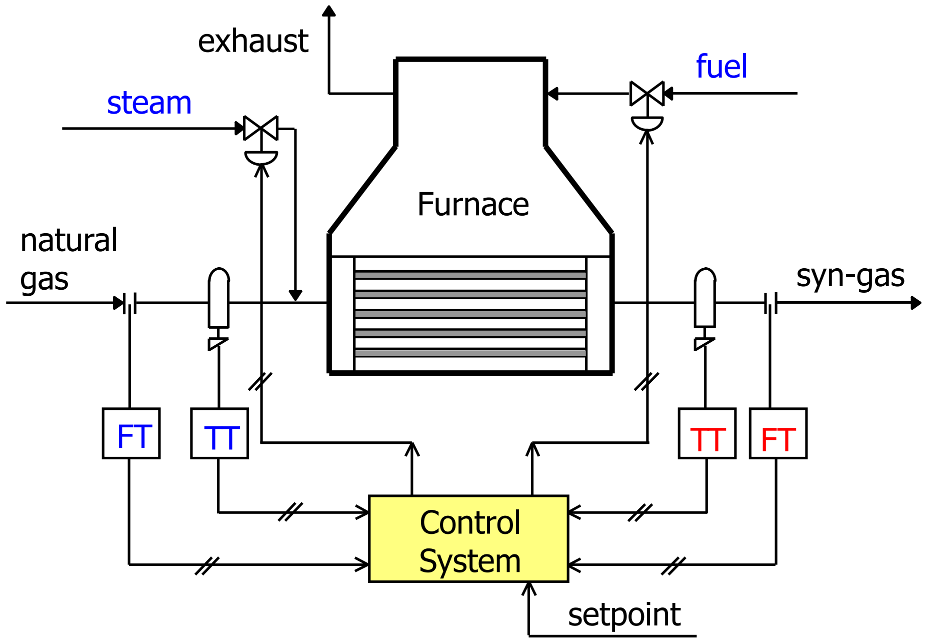
$$\underbrace{\hat{\mathbf{y}}(k)}_{\text{Model forecast}} = \boldsymbol{\varphi}\left[\mathbf{y}(k-1), \underbrace{\mathbf{y}}_{\text{Process output}}(k-2), \ldots\ldots, \mathbf{y}(k-\mathbf{n}_a), \mathbf{u}(k-1-\underbrace{\boldsymbol{\tau}}_{\text{Time delay}}), \mathbf{u}(k-2-\boldsymbol{\tau}), \ldots\ldots, \mathbf{u}(k-\mathbf{n}_b-\boldsymbol{\tau})\right]$$

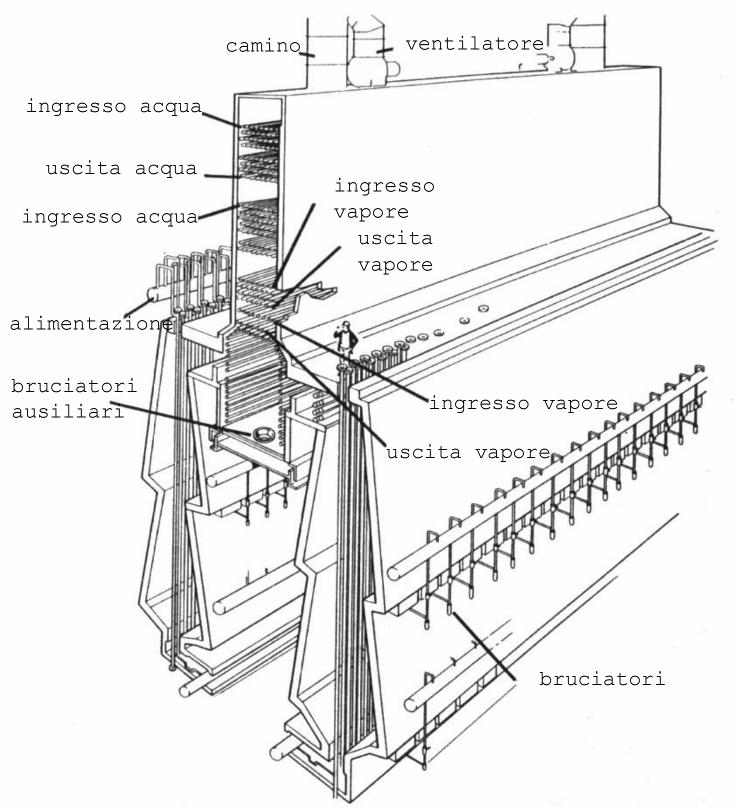
Dynamic identification of processes

- Example: **Steam Reforming process**.

$$CH_4 + H_2O \Leftrightarrow CO + 3H_2 \quad \Delta H = +206 \text{ kJ/mol}$$

$$CO + H_2O \Leftrightarrow CO_2 + H_2 \quad \Delta H = -41 \text{ kJ/mol}$$





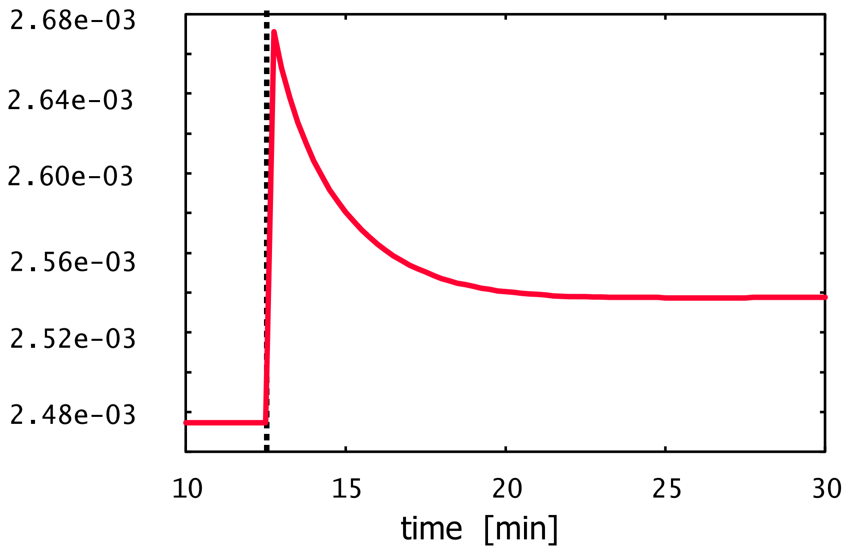
Special Cr/Ni alloy pipes resistant up to 1150 °C.
Granular catalyst: Nickel supported on Alumina.

Network input: manipulated variables and measurable disturbances

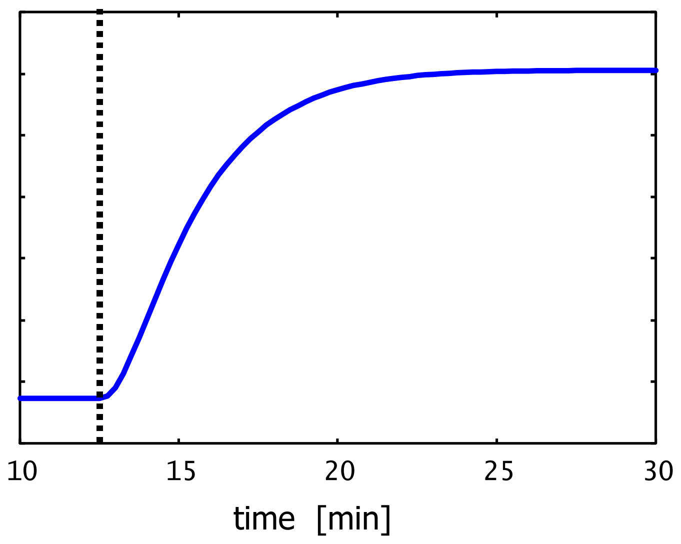
Network output: controlled variables

Open-loop analysis of the system

Hydrogen flowrate [kmol/s]



Disturbance on the dry flowrate

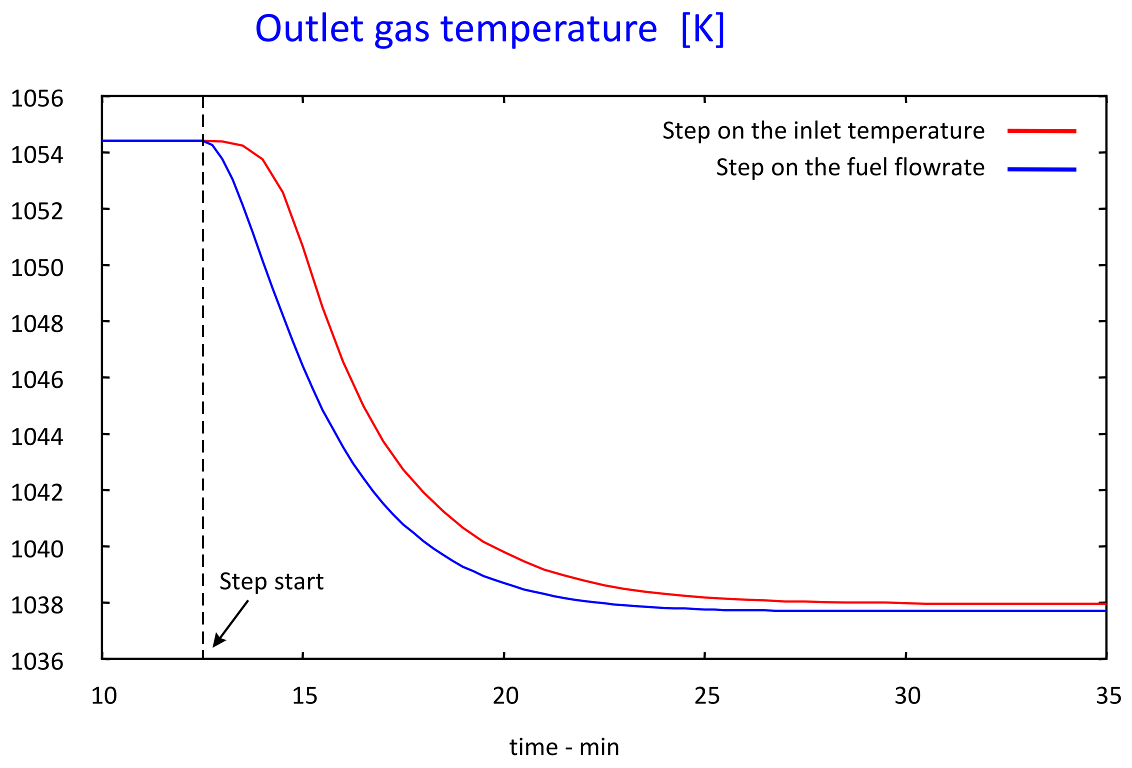


Disturbance on the furnace fuel flowrate

Main features of the process to be identified:

Overshoot presence, slow dynamics, and fast dynamics.

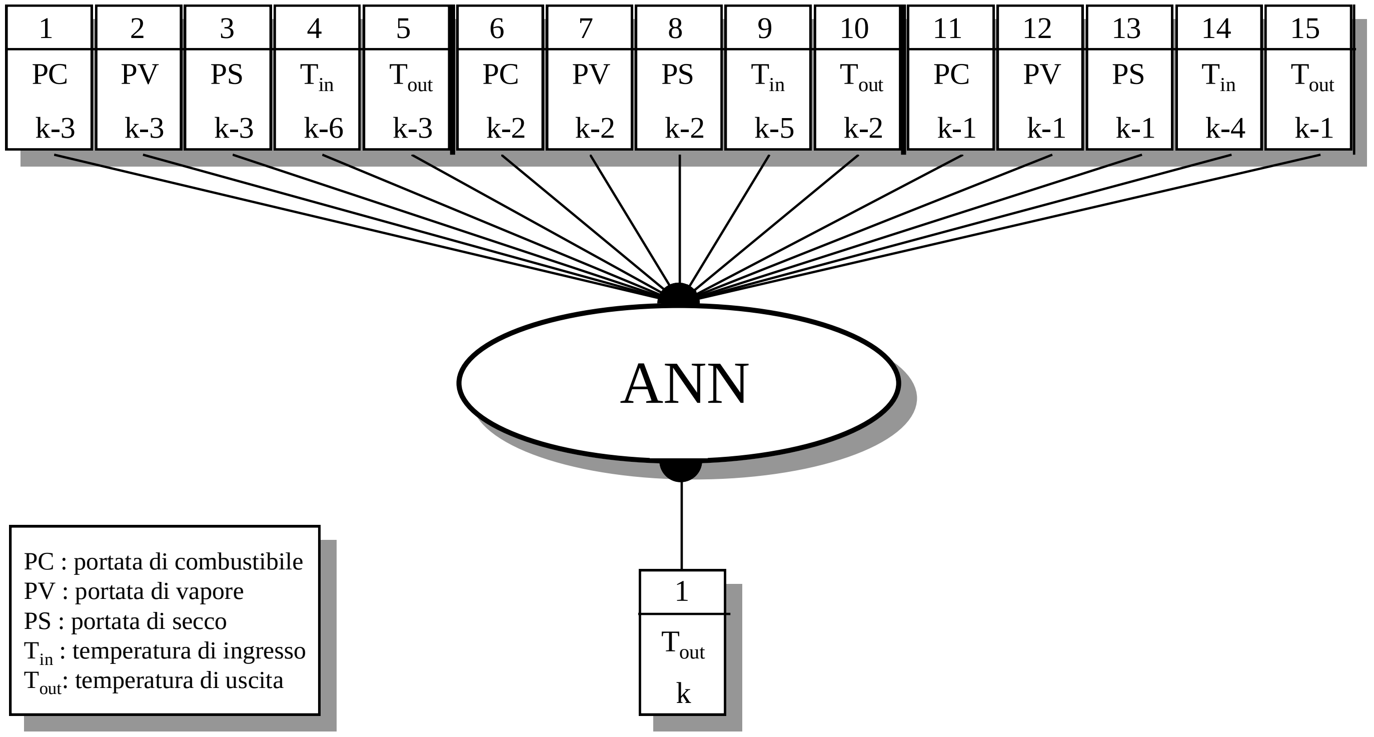
Open-loop analysis of the system



N.B.: a disturbance on the inlet gas temperature has a delayed effect on the outlet gas temperature from the reactor. It is necessary to account for this delay in the definition of the ANN structure in terms of time delay, τ.

Dynamic identification of a system

- ANN features: **two separate (and dedicated) MISO networks**.



N.B.: the second MISO network implements (instead of the reactor outlet temperature) the produced hydrogen flowrate as output variable.

Bibliography

- Battiti R., First and Second Order Methods for Learning: Between Steepest Descent and Newton's Method, Neural Computation, 4, 2, 1992
- Cholewo T.J., J.M. Zurada, Exact Hessian Calculation in Feedforward FIR Neural Networks, IEEE International Joint Conference on Neural Networks, 1998
- Churchland P.S., T.J. Sejnowski, The Computational Brain, MIT Press, Cambridge, 1992
- Cybenko G., Approximation by Superpositions of a Sigmoidal Function, Mathematics of Control Signals and Systems, 1989
- Hagan M.T., H.B. Demuth, M. Beale, Neural Network Design, PWS Publishing Company, Boston, 1996
- **Haykin S., Neural Networks: a Comprehensive Foundation, Prentice-Hall, Upper Saddle River, New Jersey, 1999**
- Henson M.A., D.E. Seborg, Nonlinear Process Control, Prentice Hall, 1998
- Hilgard E.R., R.L. Atkinson, R.C. Atkinson, Introduction to Psychology, Harcourt Brace Jovanovich, 1979
- Hrycej T., Neurocontrol towards an Industrial Control Methodology, John Wiley & Sons, New York, 1997
- Narendra K.S., A.U. Levin, Identification using Feedforward Networks, Neural Computation, 7, 1995

Bibliography

- Narendra K.S., K. Parthasarathy, Identification and Control of Dynamical Systems Using Neural Networks, IEEE Trans. on Neural Networks, 1, 1990
- **Principe J. C., Euliano N. R. and Lefebvre W. C., Neural and Adaptive Systems: Fundamentals through Simulations, John Wiley & Sons, New York, 2000**
- Ranga Suri N.N.R., D. Deodhare, P. Nagabhushan, Parallel Levenberg-Marquardt-based Neural Network Training on Linux Clusters - A Case Study, Indian Conference on Computer Vision Graphics and Images Processing, 2002
- Scattolini R., S. Bittanti, On the Choice of the Horizon in Long-range Predictive Control. Some Simple Criteria, Automatica, 26, 5, 1999
- **Tadé M.O., P.M. Mills, A.Y. Zomaya, Neuro-Adaptive Process Control: a Practical Approach, John Wiley & Sons, New York, 1996**
- Wan E.A., F. Beaufays, Diagrammatic Derivation of Gradient Algorithms for Neural Networks, Neural Computation, 8, 1, 1996
- Werbos P.J., The Roots of Backpropagation - From Ordered Derivatives to Neural Networks and Political Forecasting, John Wiley & Sons, New York, 1994
- Wilamowski B.M. , Neural Network Architectures and Learning, IEEE-Neural Networks Society, 2003
- Wilamowski B.M., S. Iplikci, O. Kaynak, An Algorithm for Fast Convergence in Training Neural Networks, IEEE International Joint Conference on Neural Networks, 2001