POLITECNICO
MILANO 1863

lasar3
Laboratory of analysis of systems for the assessment of reliability, risk and resilience

Fault Diagnostics Methods

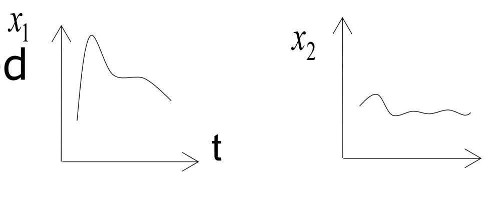
Joaquín Figueroa
Politecnico di Milano, Energy Department
joaquineduardo.figueroa@polimi.it

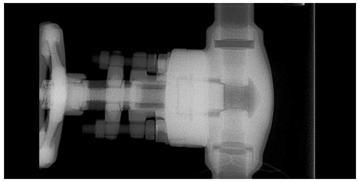
April 14th, 2025

Context: Prognostics and Health Management

Equipment (System, Structure or Component)

Measured signals





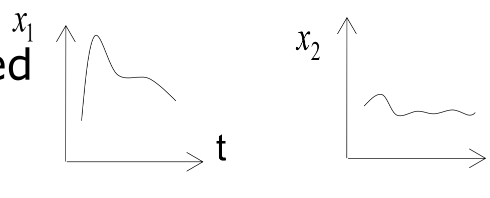
$$\frac{dx}{dt} = A\exp\left(-\frac{Q}{RT}\right)\varphi^n$$

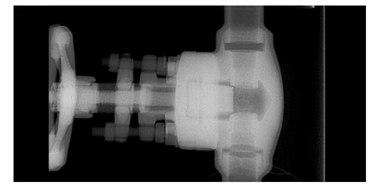
Detect	**Diagnose**	**Predict**
Anomalous operation / **Normal operation**	c_1 c_2 c_3 — **Malfunctioning type (classes)**	**Remaining Useful Life (RUL)**

In This Lecture: Fault Diagnostics

Equipment (System, Structure or Component)

Measured signals





$$\frac{dx}{dt} = A\exp\left(-\frac{Q}{RT}\right)\varphi^n$$

Detect

- Anomalous operation
- Normal operation

Diagnose

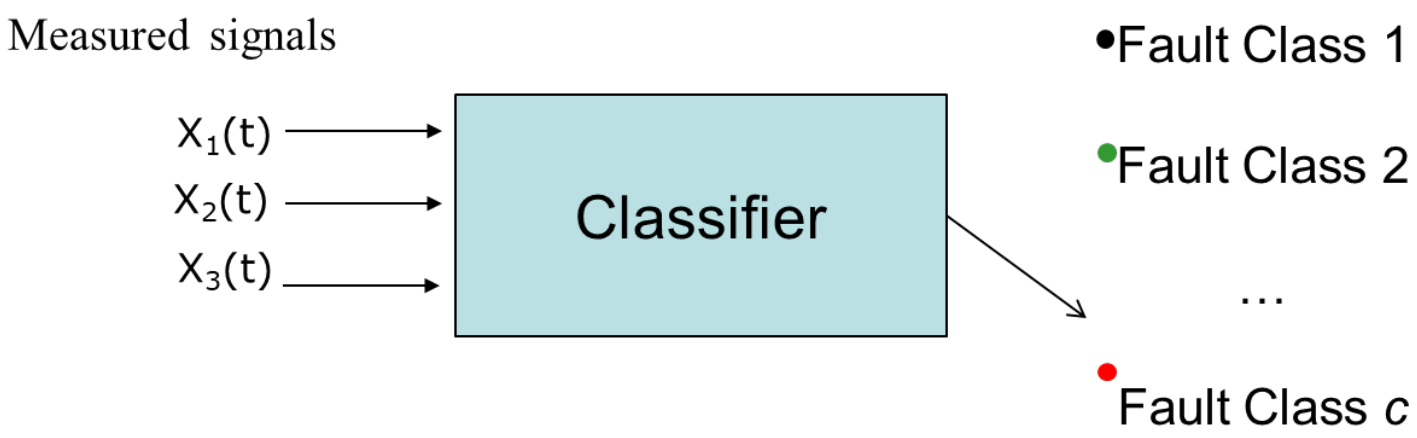
c_1 c_2 c_3

Malfunctioning type (classes)

Predict

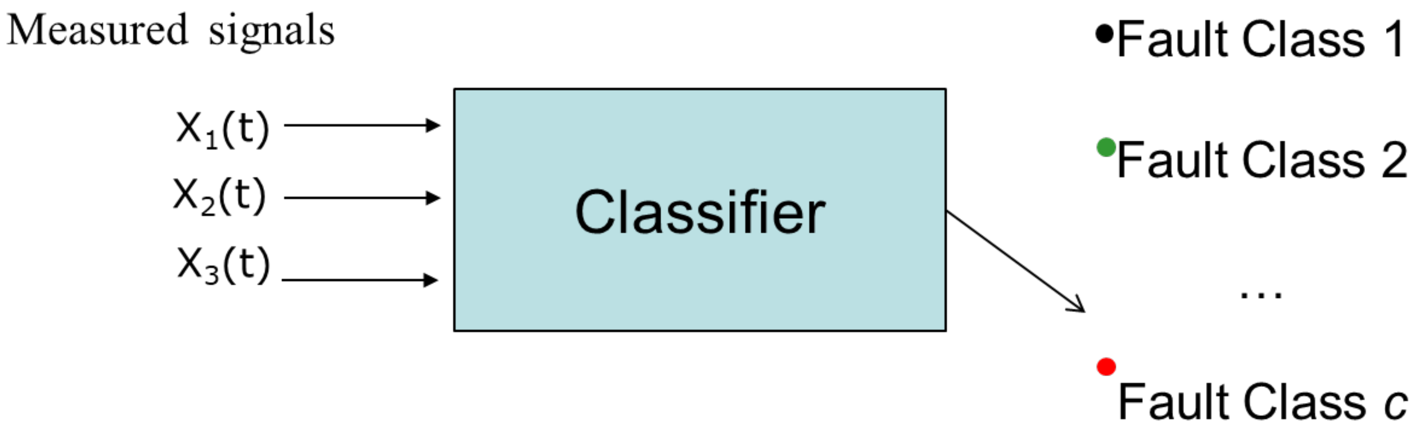
Remaining Useful Life (RUL)

Fault Diagnostics: Objective



Fault Diagnostics: Approaches

- Model-Based Approaces
  - Inverse Problem → Difficult to develop
- Data-driven Approaches



In This Lecture

1. Procedural steps for developing a fault diagnostic system
2. Supervised classification methods for fault diagnostics

Procedural steps for developing a fault diagnostic system: Step 1

1. Identify fault/degradation classes:
   - System Analysis (FMECA, Event Tree Analysis, …)
   - Good engineering sense of practice

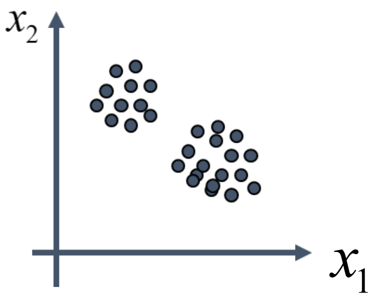
$\{C_1 \qquad C_2\}$

● ★

Procedural steps for developing a fault diagnostic system: Step 2

1. Identify fault/degradation classes
2. Analysis of the historical data

Case 1

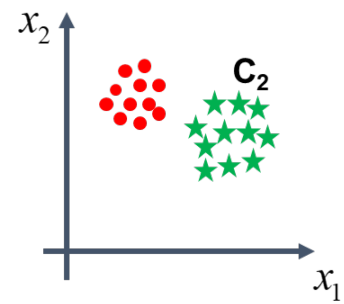


<measurements, unknown class>

=

$<x_1(k), x_2(k), ?>$

Case 2



<measurements, class>

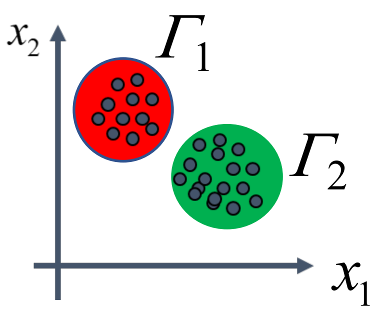
=

$<x_1(k), x_2(k), c(k)>$

Procedural steps for developing a fault diagnostic system: Step 3

1. Identify fault/degradation classes
2. Analysis of the historical data
3. Develop the clustering and/or classification methods :

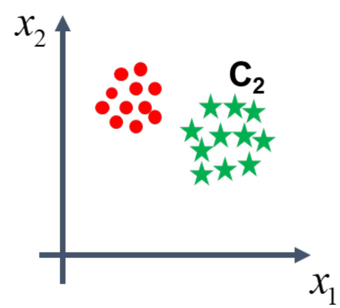
Case 1: clustering



What is Γ_2 ?

Expert

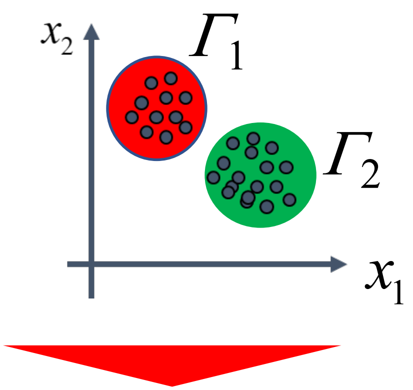
Case 2: classification



Procedural steps for developing a fault diagnostic system: Step 3

1. Identify fault/degradation classes
2. Analysis of the historical data
3. Develop the clustering and/or classification methods :

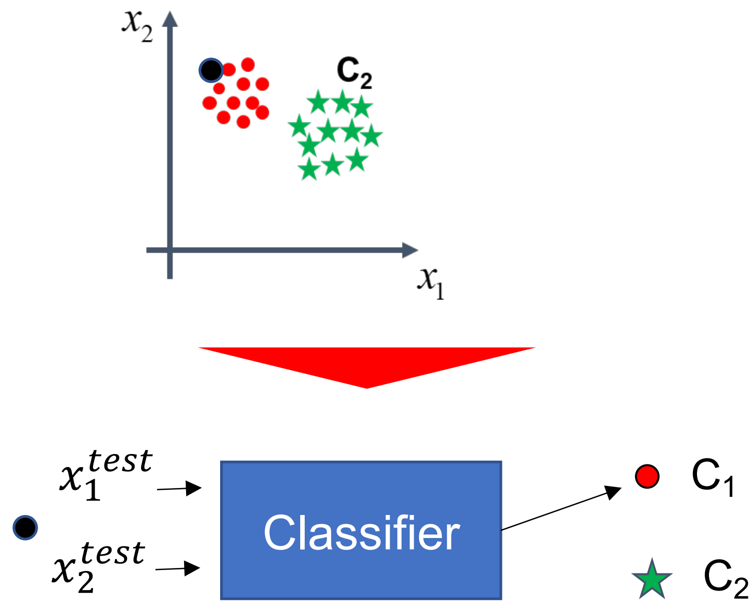
Case 1: clustering



What is Γ_2 ?

Expert

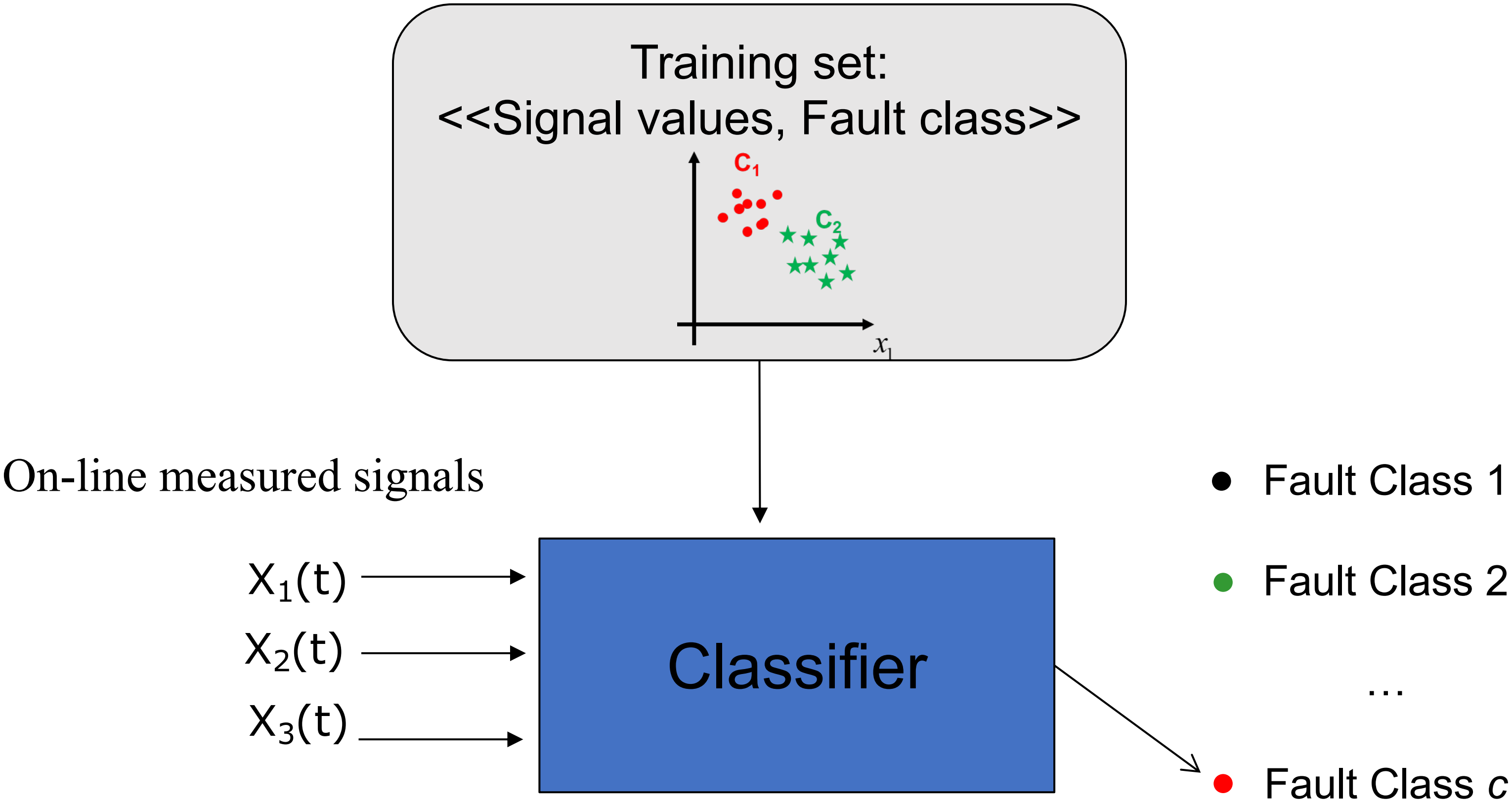
Case 2: classification



In This Lecture

1. Procedural steps for developing a fault diagnostic system
2. Supervised classification methods for fault diagnostics

Supervised Classification



Classification Algorithms

- K-Nearest Neighbor Classifier
- Support Vector Machines
- Fuzzy similarity-based approaches
- Artificial Neural Networks (ANNs)
  - Deep Neural Networks (DNNs)
  - Convolutional Neural Networks (CNNs)
  - Generative Adversarial Networks (GANs)
- Neurofuzzy Systems
- Relevant Vector Machines
- Ensemble Systems
- …

Classification Algorithms

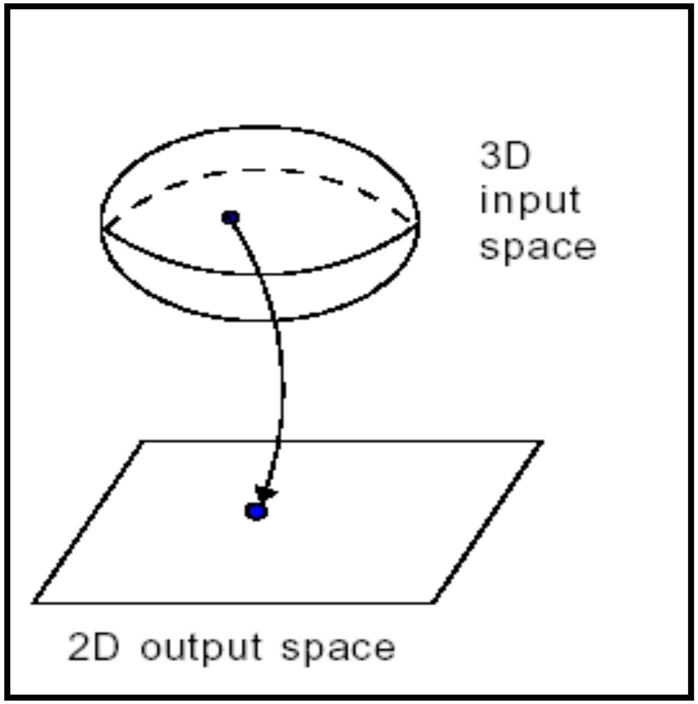
- K-Nearest Neighbor Classifier
- Support Vector Machines
- Fuzzy similarity-based approaches
- Artificial Neural Networks (ANNs)
  - Deep Neural Networks (DNNs)
  - Convolutional Neural Networks (CNNs)
  - Generative Adversarial Networks (GANs)
- Neurofuzzy Systems
- Relevant Vector Machines
- Ensemble Systems
- …

FAULT DIAGNOSTICS METHODS

- **Artificial Neural Networks (ANNs)**

The problem

$$\boldsymbol{t} = g(\boldsymbol{x})$$



g is unknown!

- *g* is highly non linear
- *g* is complex

How to build an empirical model
using input/output data?

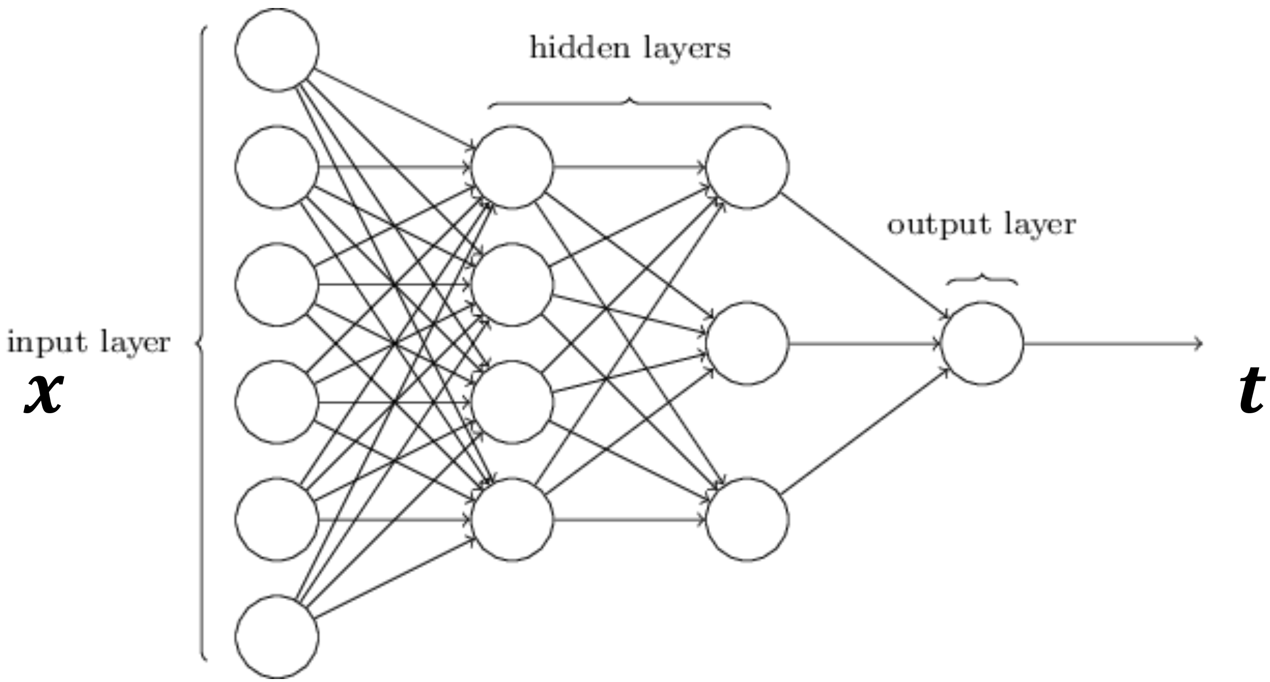
$$x_1^{(1)}, x_2^{(1)}, x_3^{(1)} \mid t_1^{(1)}, t_2^{(1)}$$

$$x_1^{(2)}, x_2^{(2)}, x_2^{(2)} \mid t_1^{(2)}, t_2^{(2)}$$

...

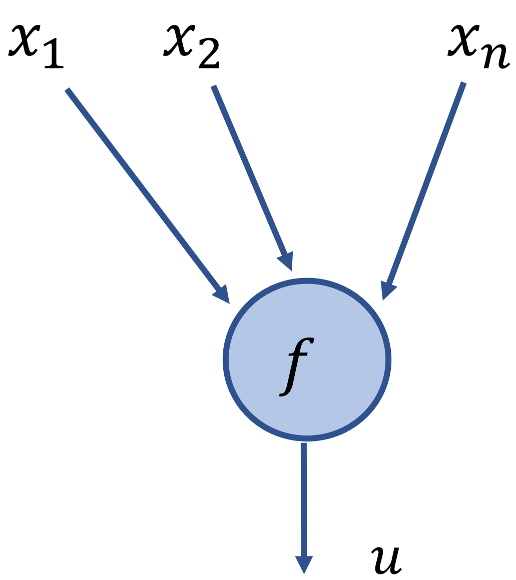
What Are (Artificial) Neural Networks?

An artificial neural network is composed by several simple computational units (also called nodes or neurons) directionally connected by weighted connections organized in a proper architecture



The computational unit

Computational unit (node, neuron)
The output (u) of a node is the result of a (possibly nonlinear) transformation f on the input variables $(x_1, x_2, \dots, x_n)$ transmitted along the links of the graph.



$$x^* = \sum_{i=1}^{n} x_i$$

$$u = f(x^*)$$

Activation function f:

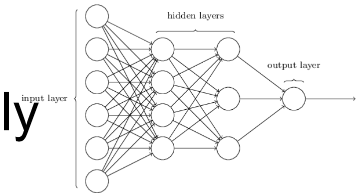
- Linear $f(x^*) = x^*$
- Sigmoid $f(x^*) = 1/(1 + e^{-x^*})$

  $f' = f(1 - f)$

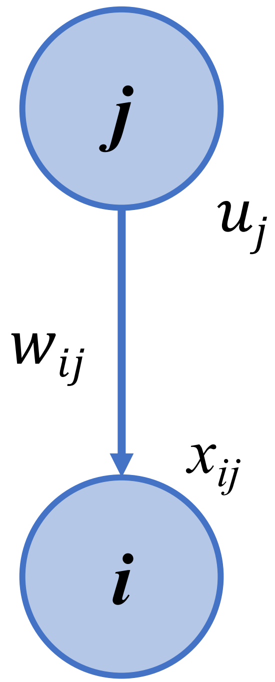
- ReLU $f(x) = \max(0, x)$
- …

The weighted connection

An artificial neural network is composed by several simple computational units (also called nodes or neurons) directionally connected by weighted connections



Connection between nodes i and j

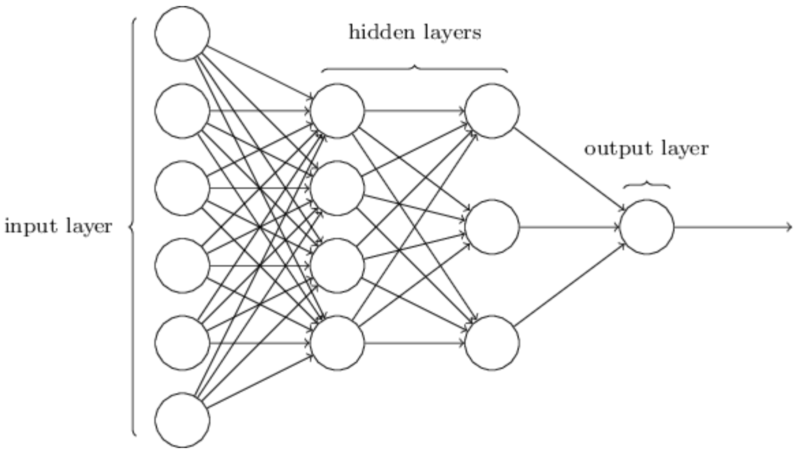


$$x_{ij} = w_{ij}u_j$$

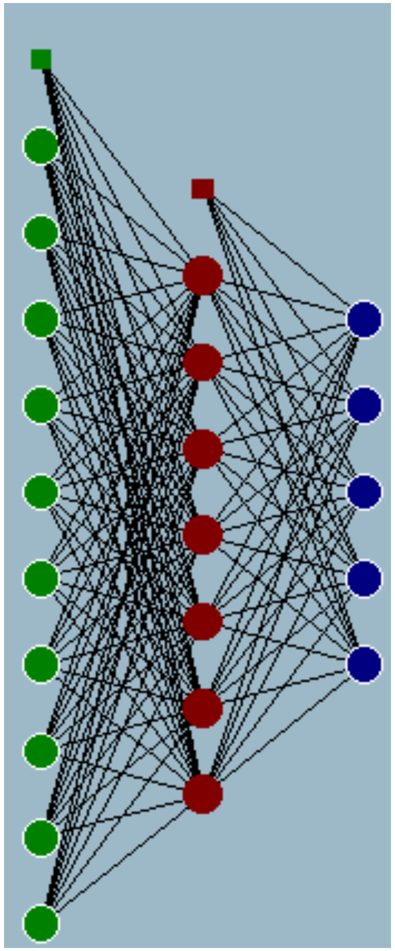
w_{ij} = synapsis weight between nodes j and i

Architecture

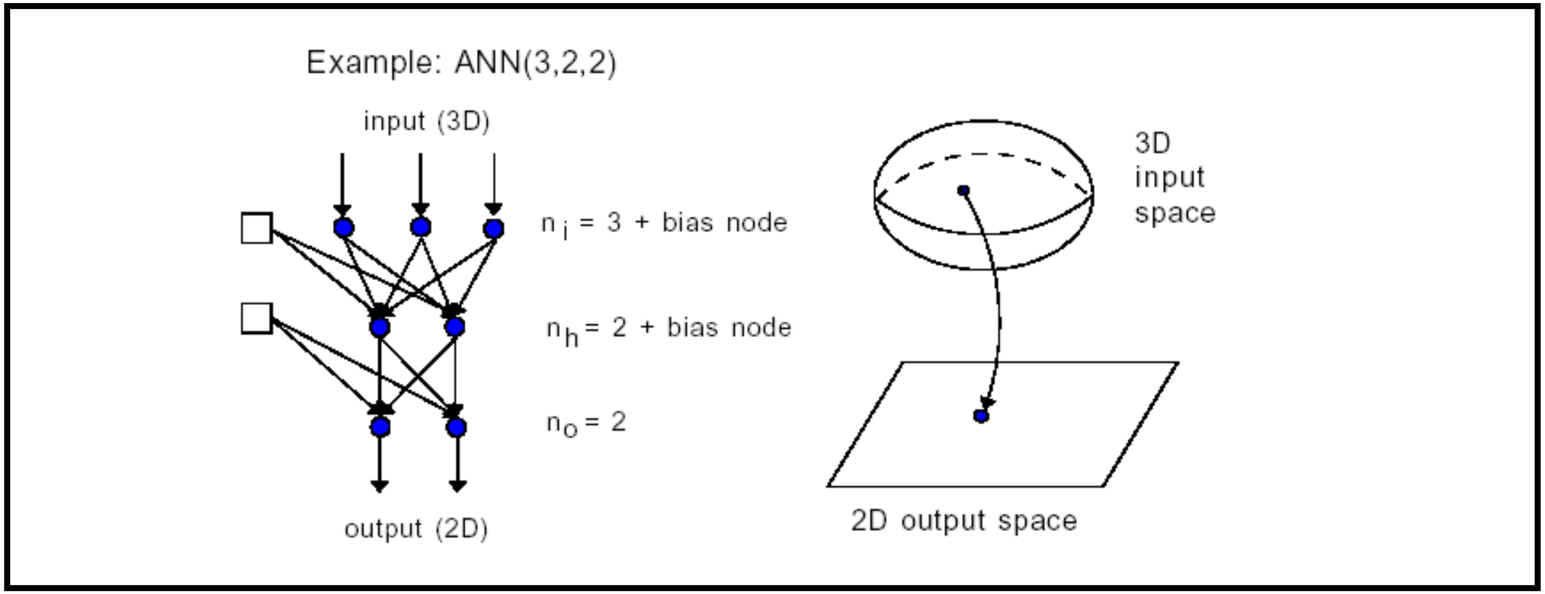
Multilayered Feedforward ANN



1 hidden layer feedforward ANN



1 Hidden Layer ANN: Forward Calculation (input-hidden)



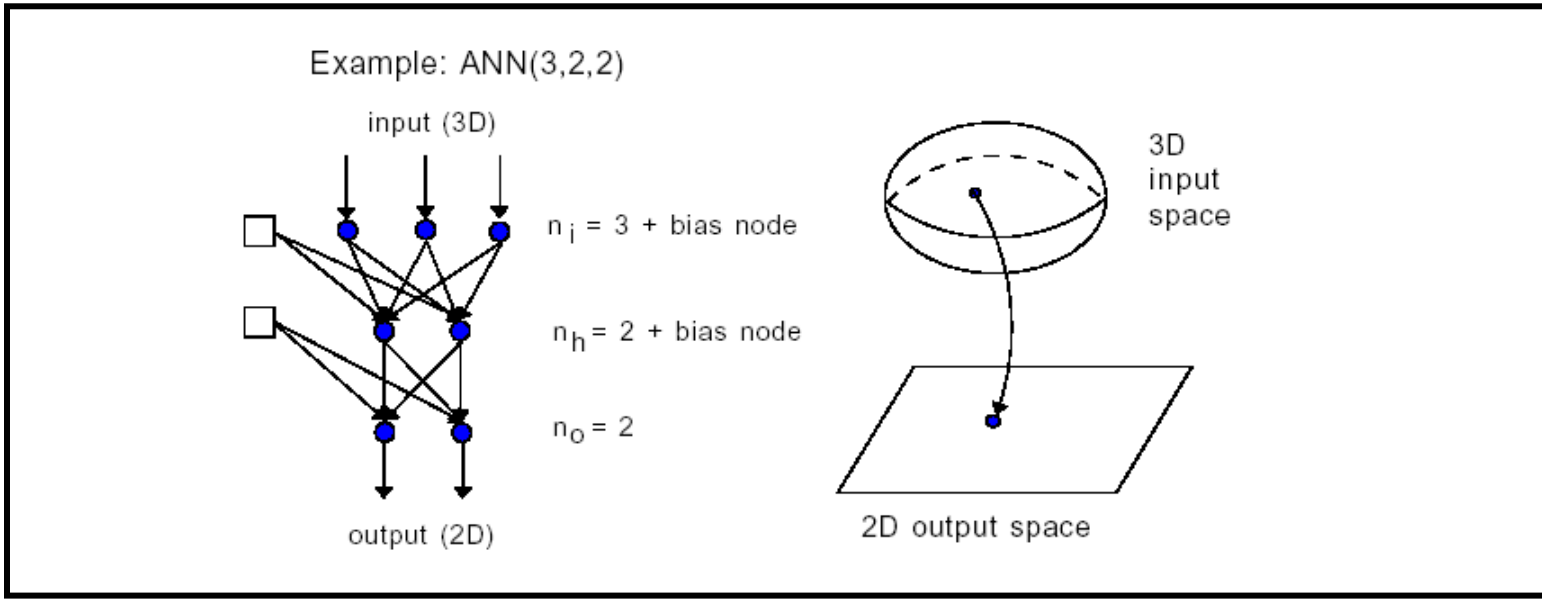
INPUT LAYER:

Each k-th node ($k = 1, 2, \ldots, n_i$) receives the value of the k-th component of the input vector $\boldsymbol{x}$ and delivers the same value

HIDDEN LAYER:

Each j-th node ($j = 1, 2, \ldots, n_h$) receives: $x_1 w_{j1}, \ldots, x_i w_{ji}, \ldots, x_{n_i} w_{jn_i}, w_{j0}$ and delivers $u_j^h = f^h\left(\sum_{k=1}^{n_i} x_k w_{jk} + w_{j0}\right)$ with f^h typically sigmoidal/ReLU

1 Hidden Layer ANN: Forward Calculation (hidden-output)



OUTPUT LAYER:

Each l-th node (l=1, 2, …, n_o) receives: $u_1^h w_{l1}, \dots, u_j^h w_{lj}, \dots, u_{n_h}^h w_{l n_h}, w_{l0}$

and delivers $u_l^o = f^o\left(\sum_{j=1}^{n_h} u_j^h w_{lj} + w_{l0}\right)$

f^o typically linear or sigmoidal

What are the mathematical bases behind Artificial Neural Networks?

Cybenko Theorem

Let σ(•) be a sigmoidal continuous function. The linear combinations:

$$\sum_{j=1}^{N} \alpha_j \sigma\left(\sum_{k=1}^{n} x_k w_{kj} + \vartheta_{k0}\right)$$

are dense in $[0,1]^n$

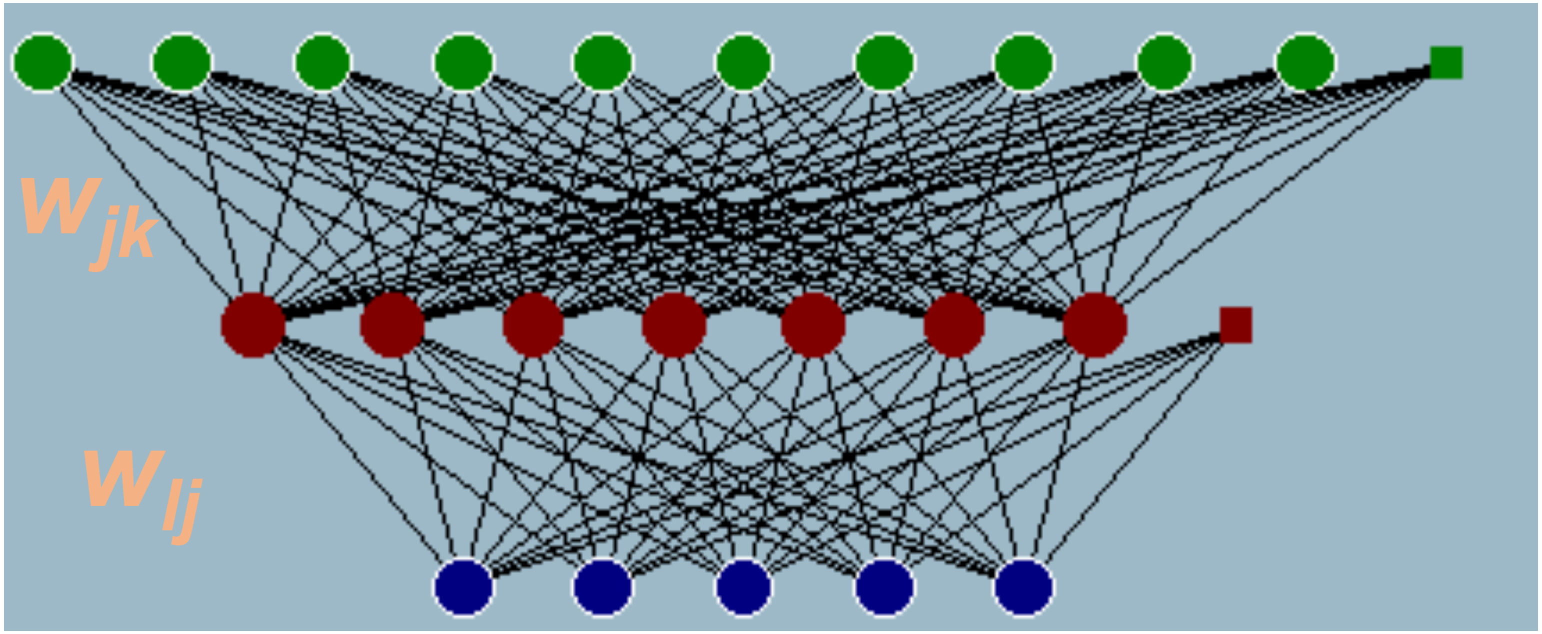
Any function $f: [0,1]^n \rightarrow \mathcal{R}$ can be approximated by a linear combinations of sigmoidal functions

Notice that the theorem does not specify the values of *N*, w_{ij} and α_j!

How to train an Artificial Neural Networks?

ANN Training: the ANN parameters to be estimated

Once the ANN architecture has been fixed (number of layers, number of nodes for layer), the only parameters to be set are the **synapsis weights** (w_{jk}, w_{lj})



Setting the ANN Parameters: Available Information

Available input/output patterns

$x_1^{(1)}, x_2^{(1)} \mid \mathrm{t}_{(1)}$

$x_1^{(2)}, x_2^{(2)} \mid \mathrm{t}_{(2)}$

$\cdots\cdots\cdots$

$x_1^{(p)}, x_2^{(p)} \mid \mathrm{t}_{(p)}$

$\cdots\cdots\cdots$

$x_1^{(np)}, x_2^{(np)} \mid \mathrm{t}_{(np)}$

$\Rightarrow$

Synapsis
Weights
W

The Training Objective/ Loss function

Training Objective: minimize the *average squared output deviation error* (also called Energy Function)*:*

$$\boxed{E = \frac{1}{2 n_p n_o} \sum_{p=1}^{n_p} \sum_{l=1}^{n_o} \left(u_{pl}^{o} - t_{pl}\right)^2}$$

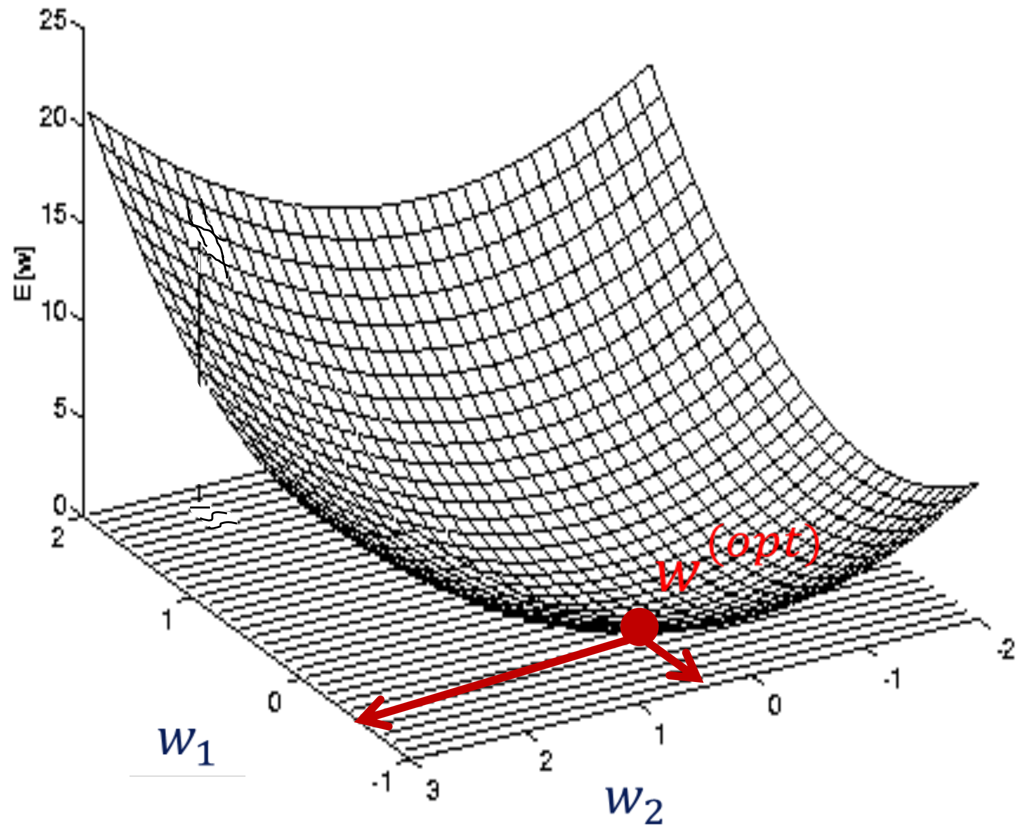
TRUE *l*-th output of the *p*-th training pattern

ANN *l*-th output of the *p*-th training pattern

The Training objective: graphical representation

E is a function of
the ANN outputs

E is a function of the synapsis weights $[w_{jk}]$



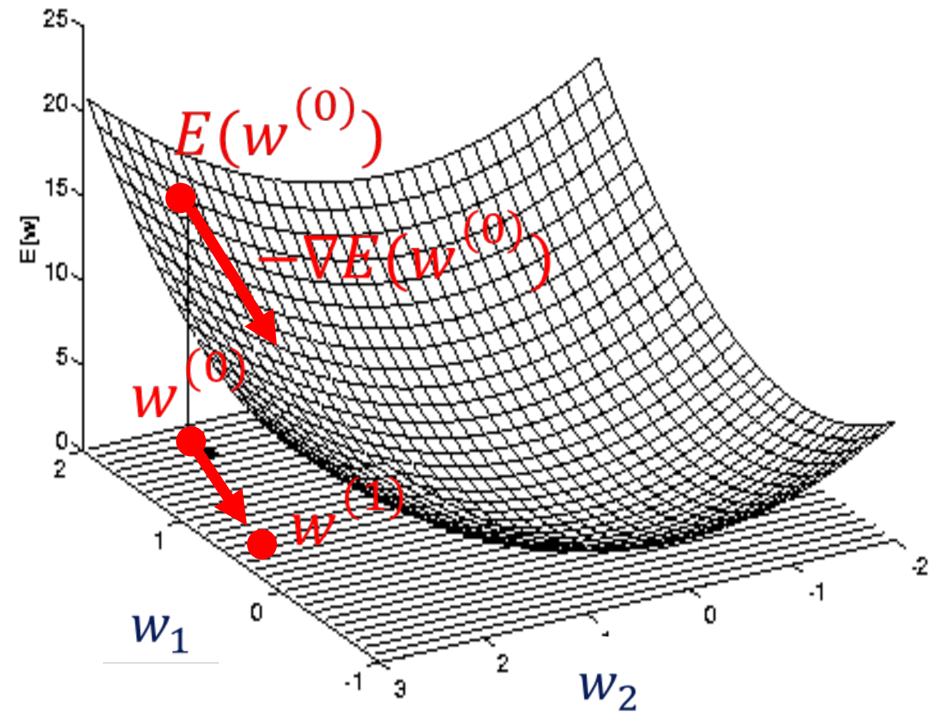
1. Initialize weights to random values:

$$w_{jk}^{(0)} = rand$$

2. While E is small:
   - Update $w_{jk}^{(i)}$ using the gradient descent method:

$$w_{jk}^{(i+1)} = w_{jk}^{(i+1)} - \eta \frac{\partial E}{\partial w_{jk}}$$

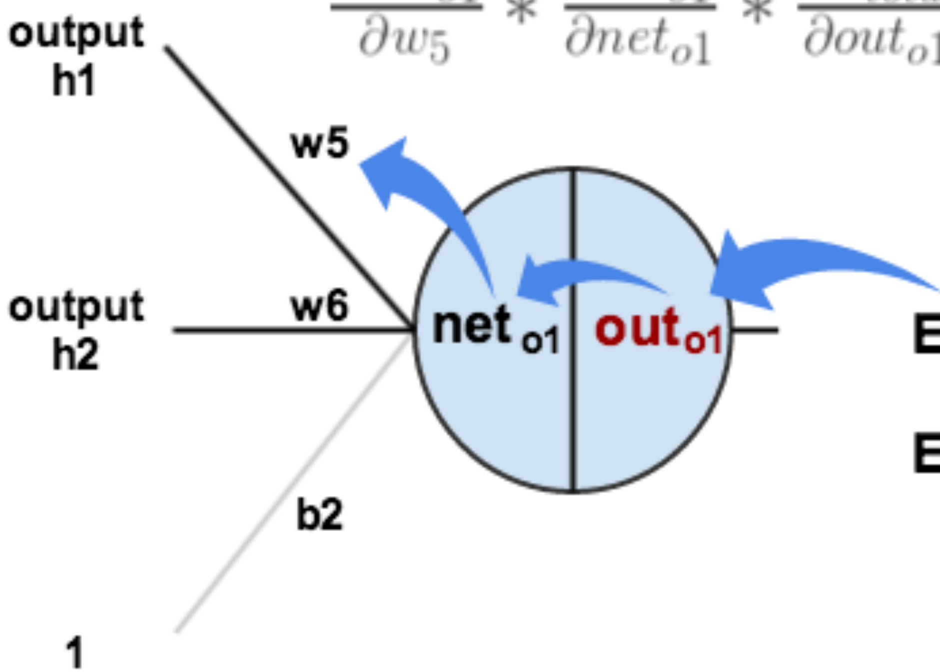
Learning coefficient



gradient $\nabla E(w) = (\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2})$

The Error Backpropagation Algorithm

$$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$$



$E_{o1} = ½(target_{o1} - out_{o1})^2$

$E_{total} = E_{o1} + E_{o2}$

Error Backpropagation (hidden-output)

- Updating of w_{lj}:

$$w_{lj}^{(i+1)} = w_{lj}^{(i)} - \eta \frac{\partial E}{\partial w_{lj}}$$

with

$$E = \frac{1}{2 n_p n_o} \sum_{p=1}^{n_p} \sum_{l=1}^{n_o} \left(u_{pl}^{o} - t_{pl}\right)^2$$

- Without loss of generality, set n_p=1

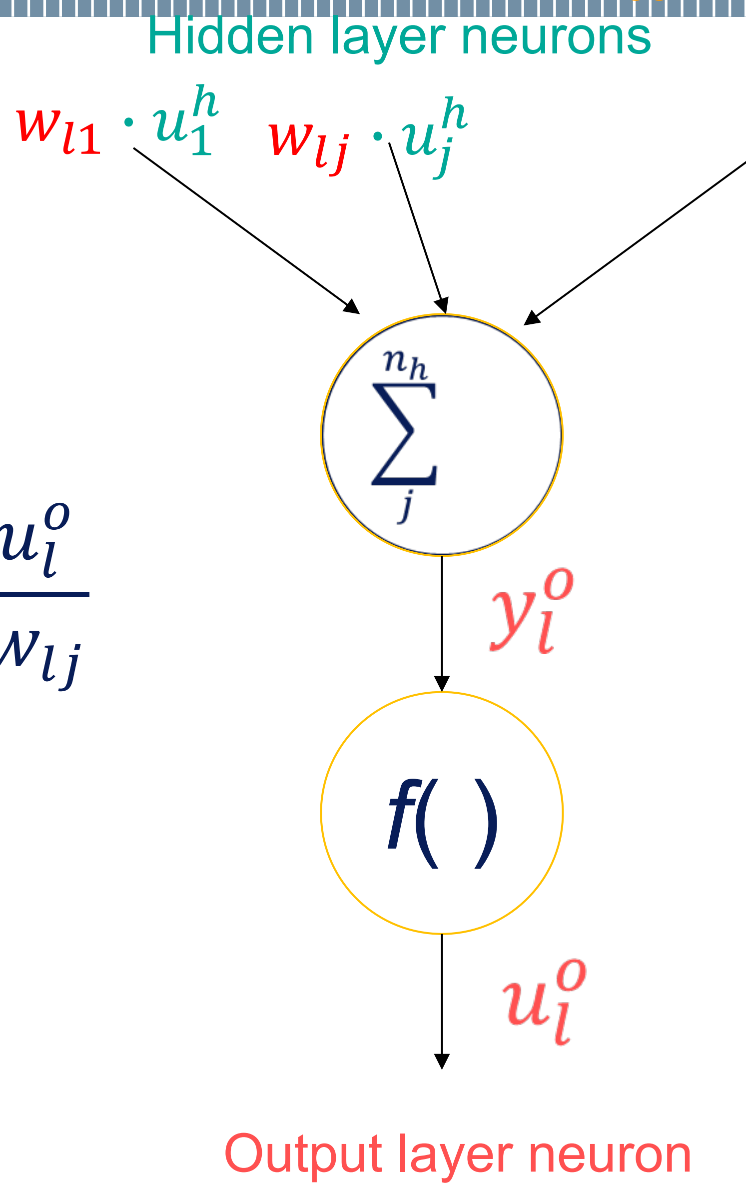
$$E = \frac{1}{2 n_o} \sum_{l=1}^{n_o} \left(u_l^{o} - t_l\right)^2$$

$$\frac{\partial E}{\partial w_{lj}} = ?$$

Error Backpropagation (hidden-output)

$$E = \frac{1}{2n_o} \sum_{l'=1}^{n_o} \left(u_{l'}^{o} - t_{l'}\right)^2$$

$$\frac{\partial E}{\partial w_{lj}} = \frac{\partial E}{\partial u_l^o} \frac{\partial u_l^o}{\partial w_{lj}} = \frac{2(u_l^o - t_l)}{2n_o} \frac{\partial u_l^o}{\partial w_{lj}}$$



Error Backpropagation (hidden-output)

$$E = \frac{1}{2n_o}\sum_{l'=1}^{n_o}\left(u_{l'}^o - t_{l'}\right)^2$$

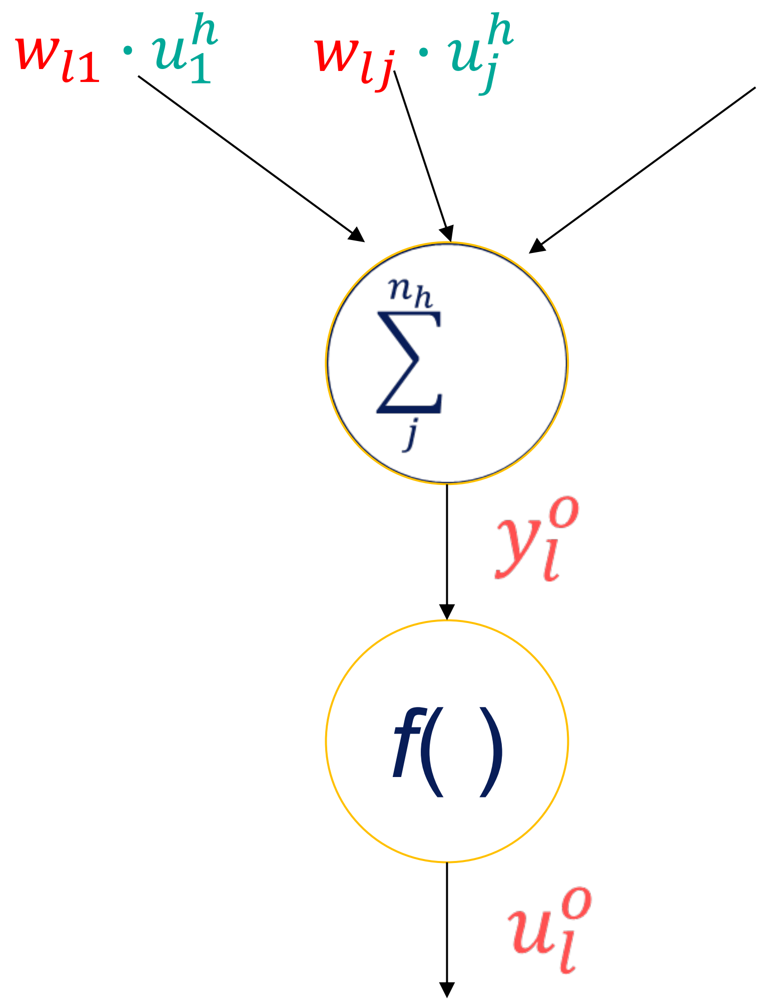
$$\frac{\partial E}{\partial w_{lj}} = \frac{2(u_l^o - t_l)}{2n_o}\boxed{\frac{\partial u_l^o}{\partial w_{lj}}}$$

$$= \frac{(u_l^o - t_l)}{n_o}\frac{\partial u_l^o}{\partial y_l^o}\frac{\partial y_l^o}{\partial w_{lj}} =$$

$$= \frac{(u_l^o - t_l)}{n_o} f'(y_l^o)\frac{\partial\left(\sum_{j'=1}^{n_h} w_{lj'}\cdot u_{j'}^h + w_{l0}\right)}{\partial w_{lj}} =$$

$$= \frac{(u_l^o - t_l)}{n_o} f'(y_l^o)\, u_j^h$$

Hidden layer neurons



Output layer neuron

Error Backpropagation (hidden-output)

- Updating of w_{lj}:

$$w_{lj}^{(i+1)} = w_{lj}^{(i+1)} - \eta \frac{\partial E}{\partial w_{lj}}$$

with

$$\frac{\partial E}{\partial w_{lj}} = \frac{(u_l^o - t_l)}{n_o} f'(y_l^o) u_j^h = \frac{1}{n_o} \delta_l u_j^h$$

$$\delta_l = (u_l^o - t_l)\, f'(y_l^o)$$

$$w_{lj}^{(i+1)} = w_{lj}^{(i)} \overbrace{- \eta \frac{1}{n_o} \delta_l u_j^h}^{\Delta w_{lj}^{(i)}}$$

- Updating of w_{lj}:

$$w_{lj}^{(i+1)} = w_{lj}^{(i+1)} - \eta \frac{\partial E}{\partial w_{lj}}$$

with

$$\frac{\partial E}{\partial w_{lj}} = \frac{(u_l^o - t_l)}{n_o} f'(y_l^o) u_j^h = \frac{1}{n_o} \delta_l u_j^h$$

momentum

$$w_{lj}^{(i+1)} = w_{lj}^{(i)} + \Delta w_{lj}^{(i)} + \alpha \Delta w_{lj}^{(i-1)}$$

Error Backpropagation (input- hidden)

- Updating of w_{kj}:

$$w_{kj}^{(i+1)} = w_{kj}^{(i)} - \eta \frac{\partial E}{\partial w_{kj}}$$

with

$$E = \frac{1}{2n_p n_o} \sum_{p=1}^{n_p} \sum_{l=1}^{n_0} \left(u_{pl}^o - t_{pl}\right)^2$$

- Without loss of generality, set n_p=1

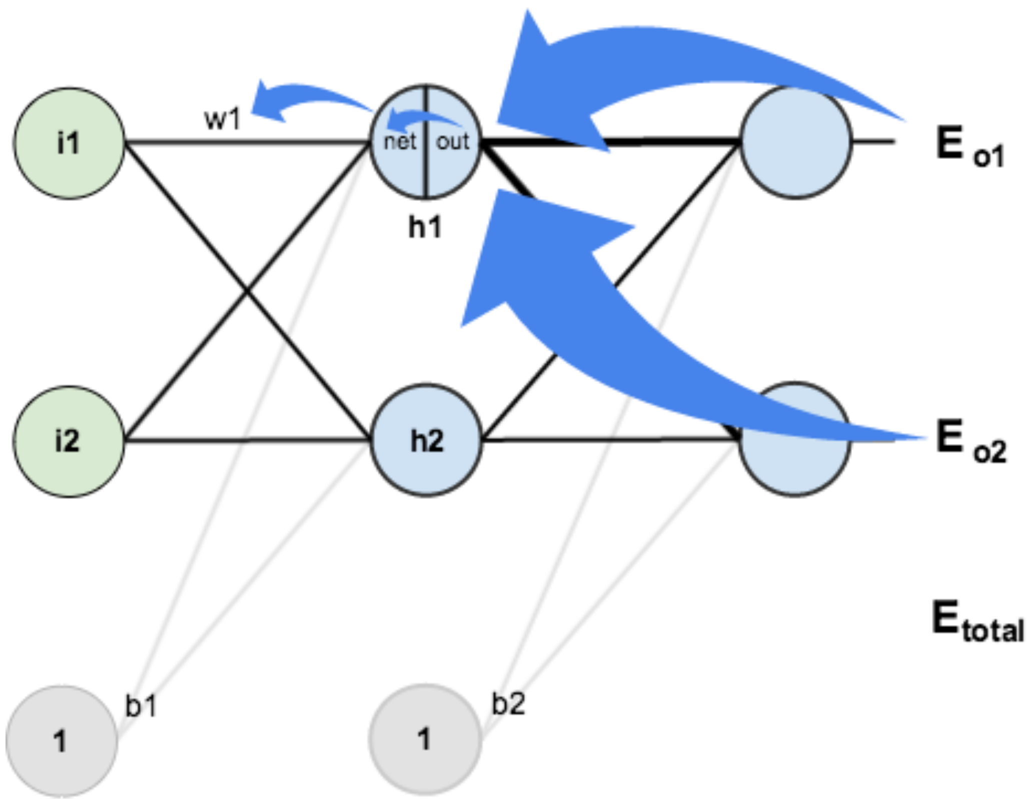
$$\frac{\partial E}{\partial w_{kl}} = \frac{1}{2n_o} \sum_{l=1}^{n_o} \frac{2(u_l^o - t_l)}{2n_o} \frac{\partial u_l^o}{\partial w_{kj}} = \frac{1}{n_o} \sum_{l=1}^{n_o} (u_l^o - t_l) \frac{\partial u_l^o}{\partial y_l^o} \frac{\partial y_l^o}{\partial u_j^h} \frac{\partial u_j^h}{\partial y_j^h} \frac{\partial y_j^h}{\partial w_{kj}} =$$

$$= \sum_{l=1}^{n_o} \frac{(u_l^o - t_l)}{n_o} f'(y_l^o) w_{jl} f'\left(y_j^h\right) u_k^i \quad = u_k^i f'\left(y_j^h\right)$$

Error Backpropagation (input- hidden)

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$



$E_{total} = E_{o1} + E_{o2}$

Error Backpropagation (hidden-input)

Similarly to the updating of the output-hidden weigths,

Updating weight w_{jk} (hidden-input connections) $\Delta \bar{w}_{jk} = -\eta \frac{\partial E}{\partial \bar{w}_{jk}}$

$$\frac{\partial E}{\partial \bar{w}_{jk}} = \frac{1}{n_o} \sum_{l=1}^{n_o} (u_l^o - t_l) \frac{\partial u_l^o}{\partial y_l^o} \frac{\partial y_l^o}{\partial u_j^h} \frac{\partial u_j^h}{\partial y_j^h} \frac{\partial y_j^h}{\partial \bar{w}_{jk}}$$

being $y_k^i = x_k \ , \ u_k^i = y_k^i$

$y_j^h = \sum_{k=1}^{n_i} \bar{w}_{jk} u_k^i + \bar{w}_{jo} \ , \ u_j^h = f(y_j^h)$

$$= \frac{1}{n_o} \sum_{l=1}^{n_o} (u_l^o - t_l) f'(y_l^o) w_{lj} f'(y_j^h) u_k^i$$

$$= -\frac{1}{n_o} \bar{\delta}_j u_k^i \qquad \bar{\delta}_j = f'(y_j^h) \sum_{l=1}^{n_o} \delta_l \, w_{lj}$$

$$\Delta \bar{w}_{jk} = \frac{1}{n_o} \eta \, \bar{\delta}_j u_k^i$$

$$\boxed{\Delta \bar{w}_{jk}(n) = \frac{1}{n_o} \eta \bar{\delta}_j u_j^i + \alpha \Delta \bar{w}_{jk}(n-1)}$$

Learning coefficient Momentum

ANN remarks

Advantages:

- No physical/mathematical modelling efforts
- Able to learn nonlinear mappings

Disadvantages:

- “black box” : difficulties in interpreting the underlying physical model.

Application

ANN for Fault Diagnostics

- **Objective:**

Build and train a neural network to classify different types of malfunctions of plant components

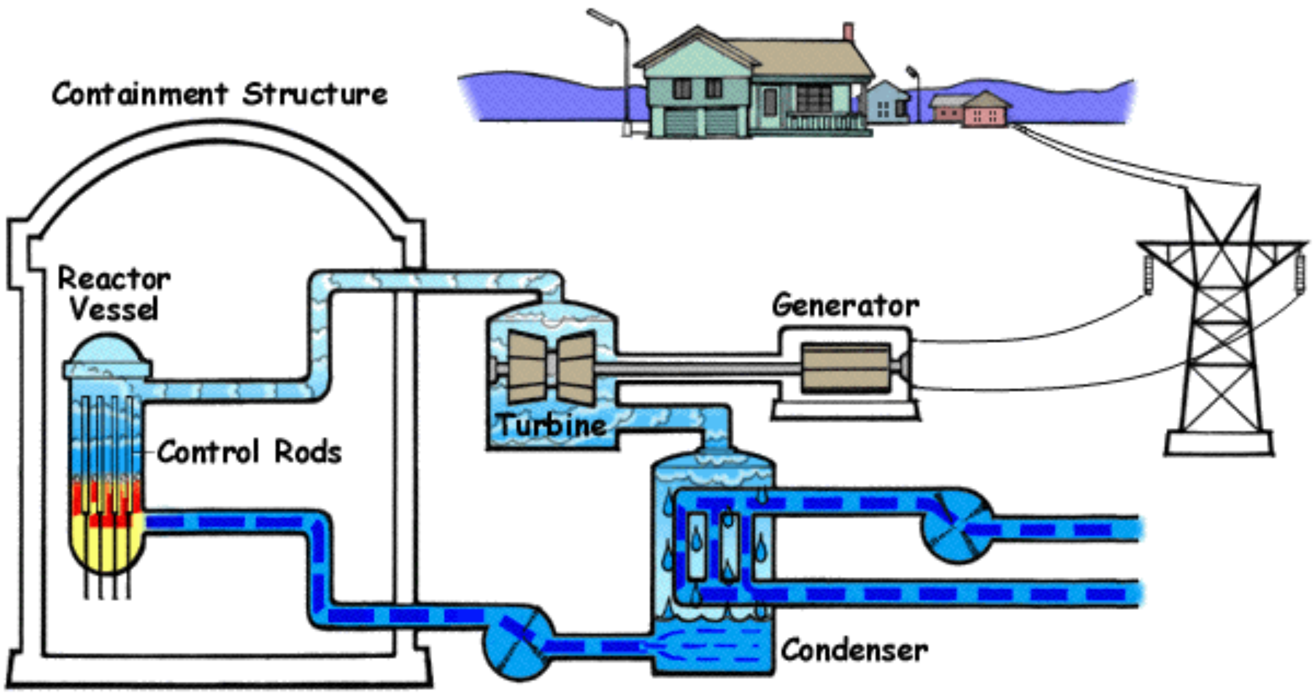
- **Input/Output patterns**

Input: signal measurements

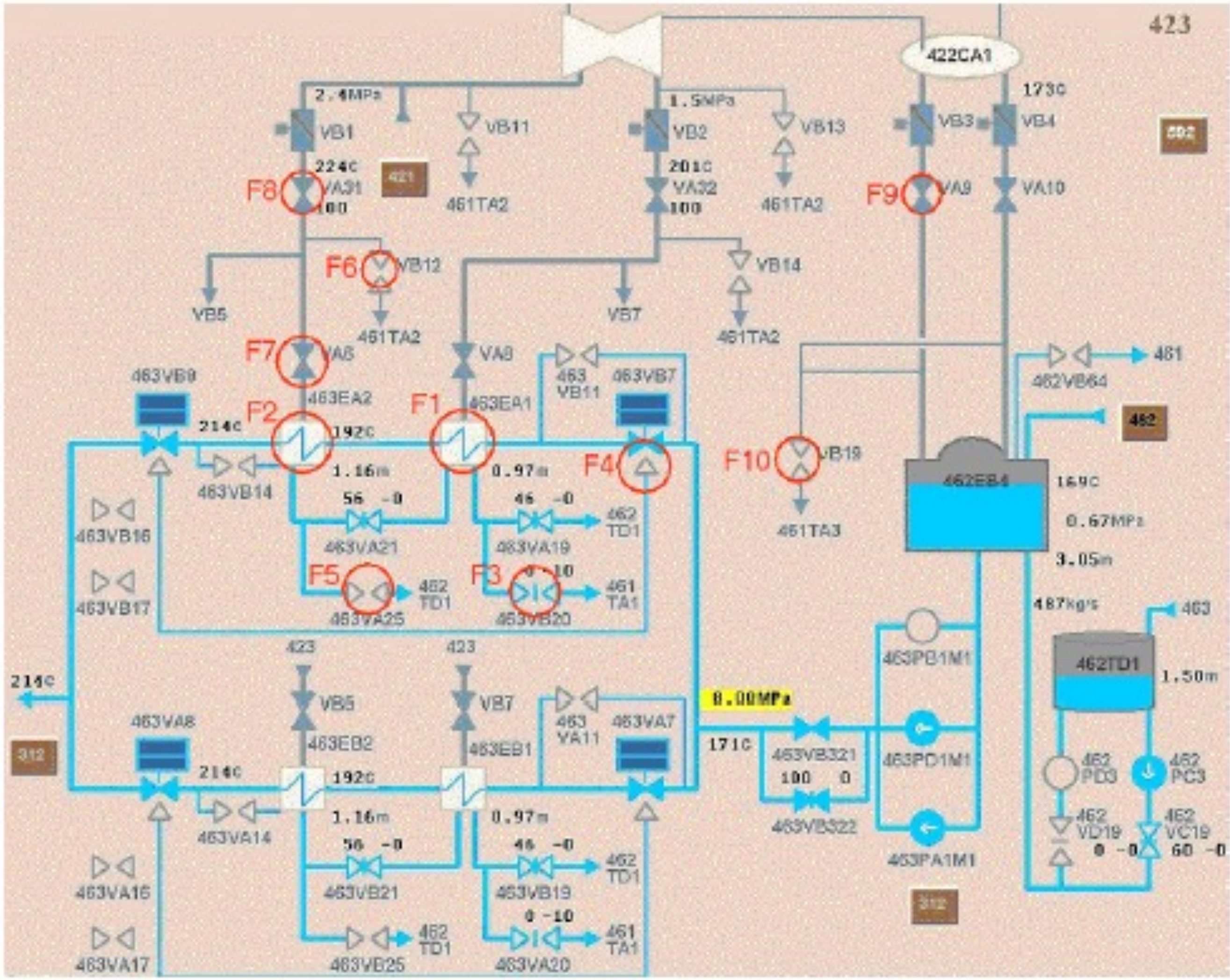
Output: number (label) of the class of the failure

ANN Example: Boiling Water Reactor

Transient classification in a Feedwater System of a Boiling Water Reactor (BWR)



ANN Example: Secondary System

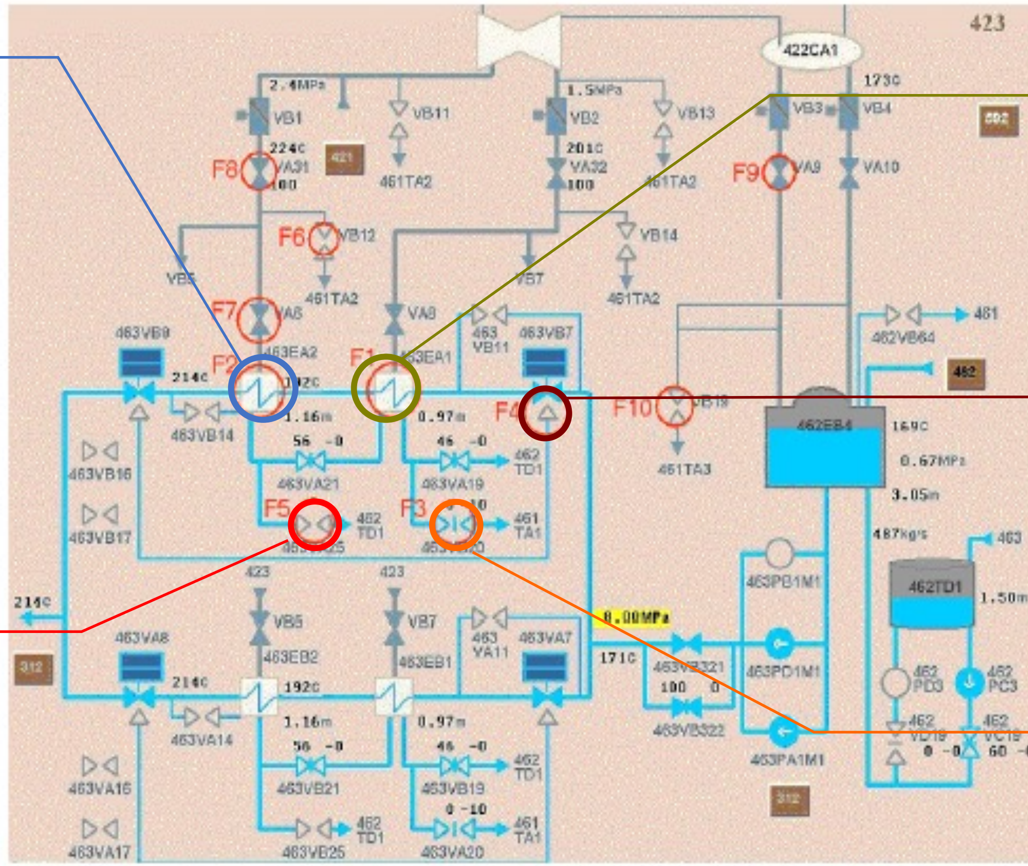


ANN Example: Outputs

- Class 1: Leakage through the second high-pressure preheater
- Class 2: Leakage in the first high-pressure preheater to the drain tank
- Class 3: Leakage through the first high-pressure preheater drain back-up valve to the condenser
- Class 4: Leakage through high-pressure preheaters bypass valve
- Class 5: Leakage through the second high-pressure preheater drain back-up valve to the feedwater tank

ANN Example: Faults

Class 2: Preheater EA2



Class 1: Preheater EA1

Class 4: Valve VB7

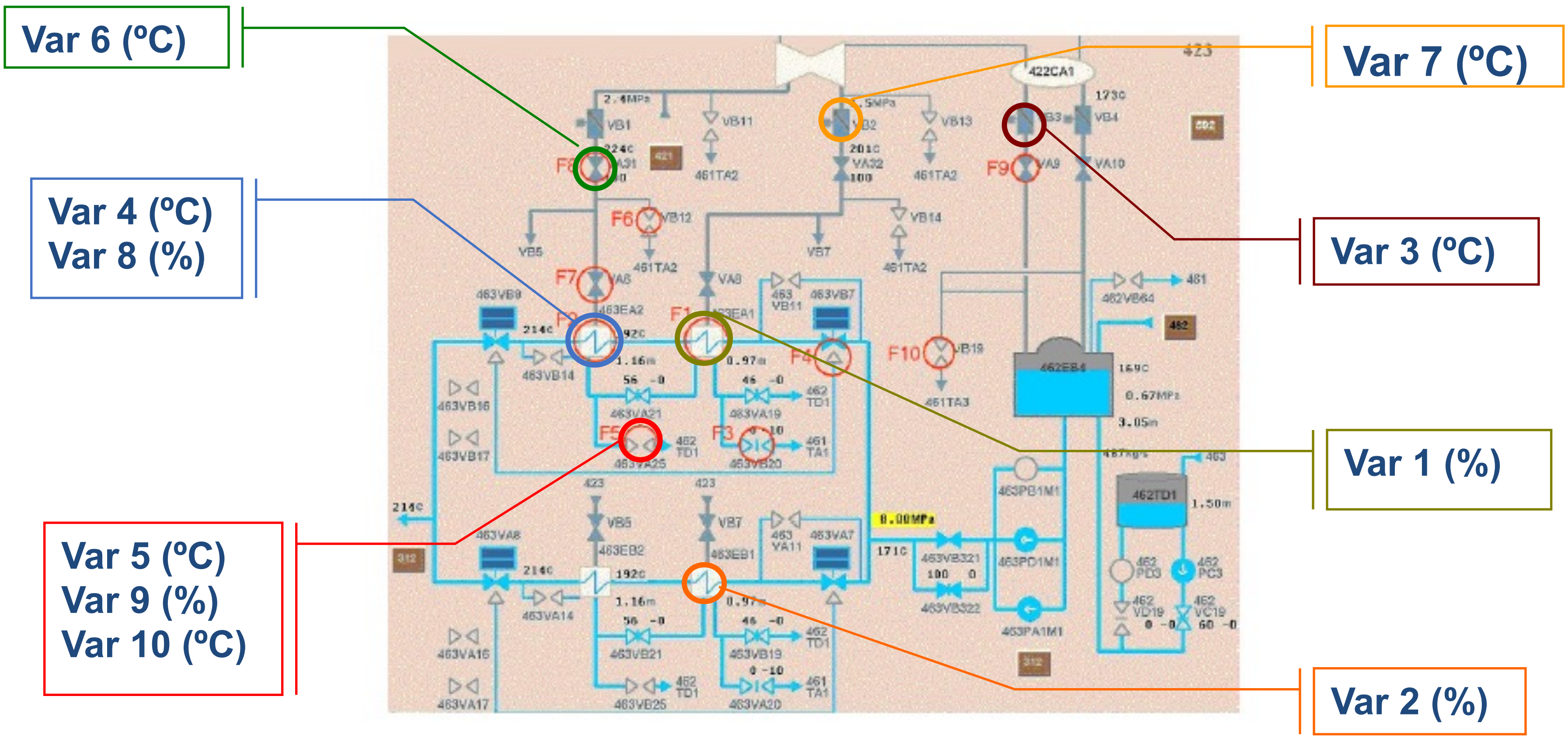
Class 5: Valve VA25

Class 3: Valve VB20

Variables inputs

Variable	Signal	Unit
1	Position level for control valve EA1	%
2	Position level for control valve EB1	%
3	Temperature drain before VB3	ºC
4	Temperature feedwater after EA2 train A	ºC
5	Temperature feedwater after EB2 train B	ºC
6	Temperature drain 6 after VB1	ºC
7	Temperature drain 5 after VB2	ºC
8	Position level control valve before EA2	%
9	Position level control valve before EB2	%
10	Temperature feedwater before EB2 train B	ºC

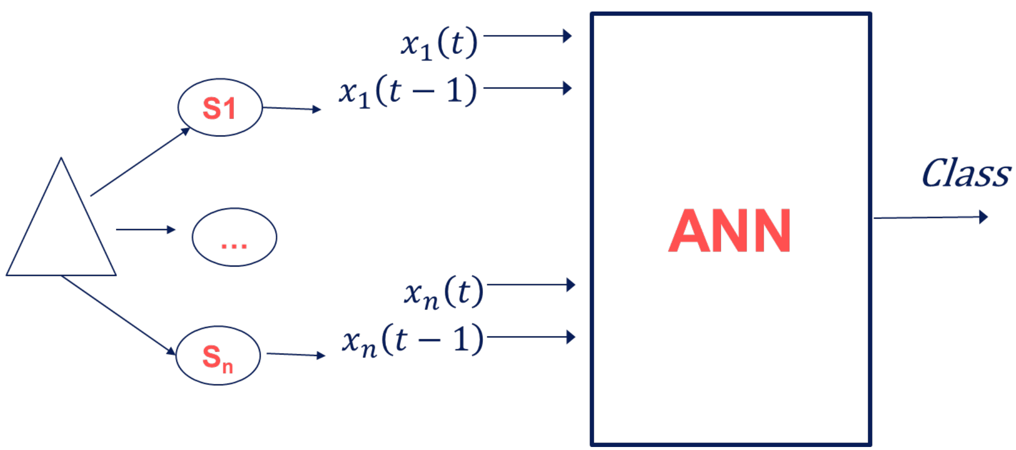
- **Measurement:**
36 sampling instants in [80, 290]s, one each 6s.



Input/Output patterns

- Input: a two-step time window of the measured signals at times t-1 and t

- Output: label of the class to which the transient belongs



Training set

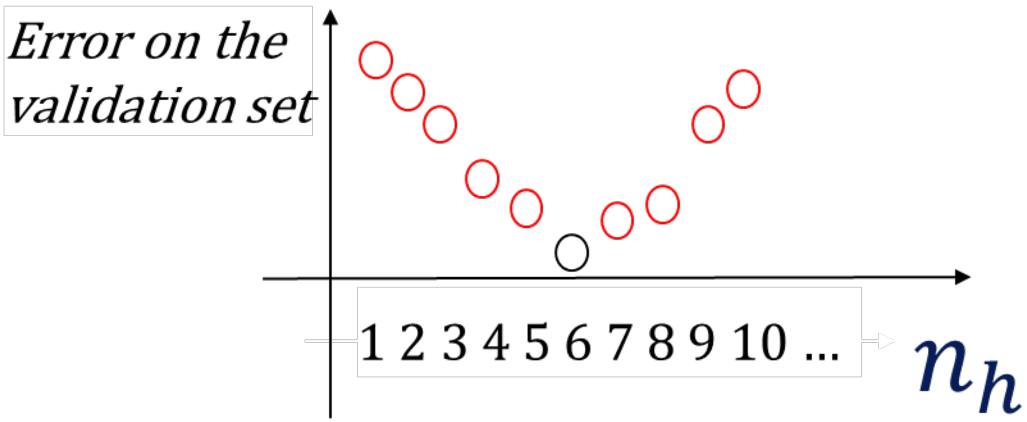
- A training set was constructed containing 8 transients for each class.

- For a transient of a given class the 35 patterns used to train the network take the following form

$$
\begin{array}{cccccccc}
x_1(2) & x_1(1) & x_2(2) & x_2(1) & \dots & x_{10}(2) & x_{10}(1) & class \\
\vdots & & & & & & & \vdots \\
x_1(t) & x_1(t-1) & x_2(t) & x_2(t-1) & \dots & x_{10}(t) & x_{10}(t-1) & class \\
x_1(t+1) & x_1(t) & x_2(t+1) & x_2(t) & \dots & x_{10}(t+1) & x_{10}(t) & class \\
\vdots & & & & & & & \vdots \\
x_1(36) & x_1(35) & x_2(36) & x_2(35) & \dots & x_{10}(36) & x_{10}(35) & class
\end{array}
$$

How to set the ANN architecture

- **Paramers to be set:**
  - Number of neurons in the hidden layer (n_h)
  - Learning coefficient
  - Momentum
- **Method:**
  - Divide the training data into:
    - A training set (it will be used to find the synapsis weight
    - A validation set (it will be used to find the optimal value of the parameters)
  - Proceed by trial-and-error

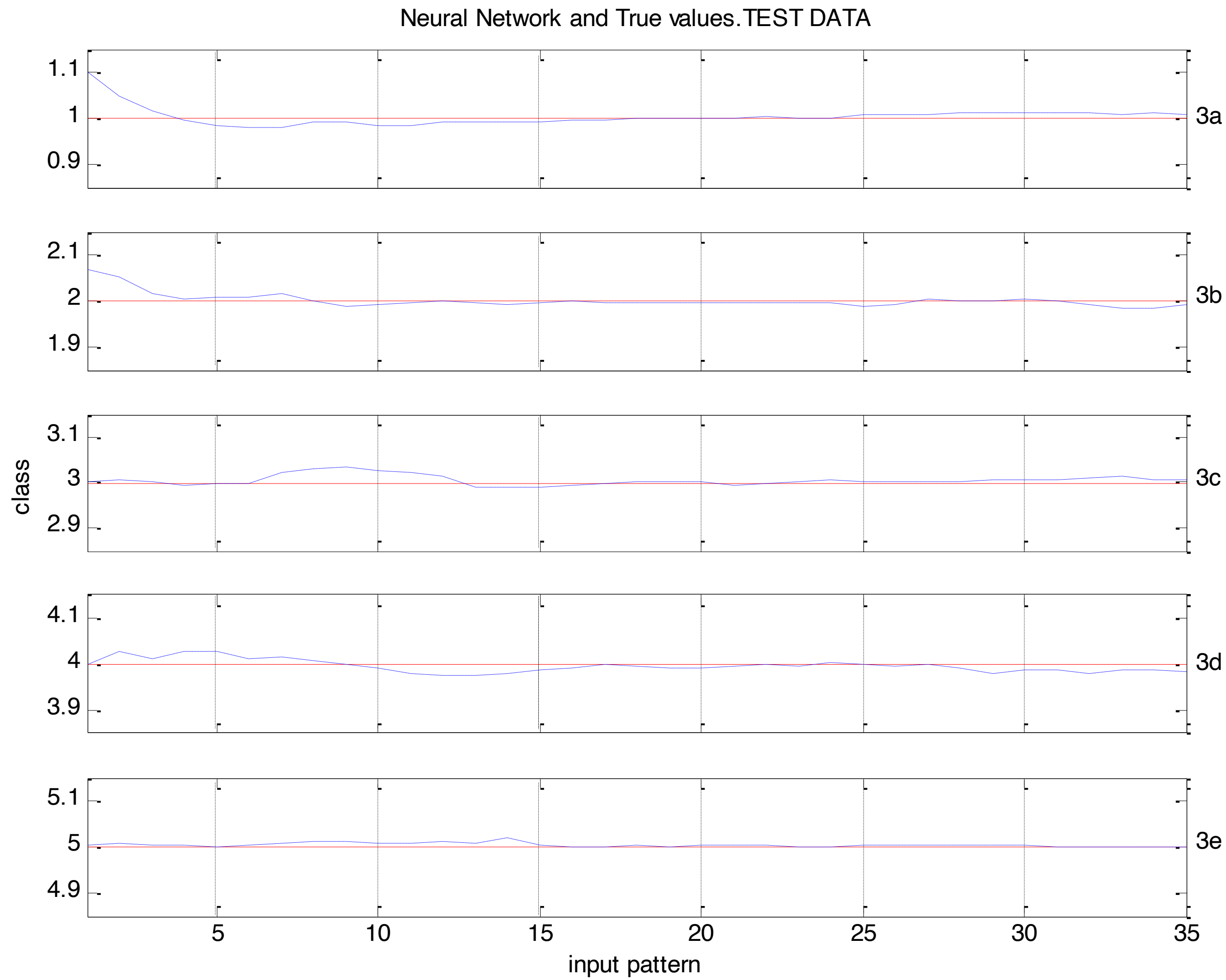


Network architecure

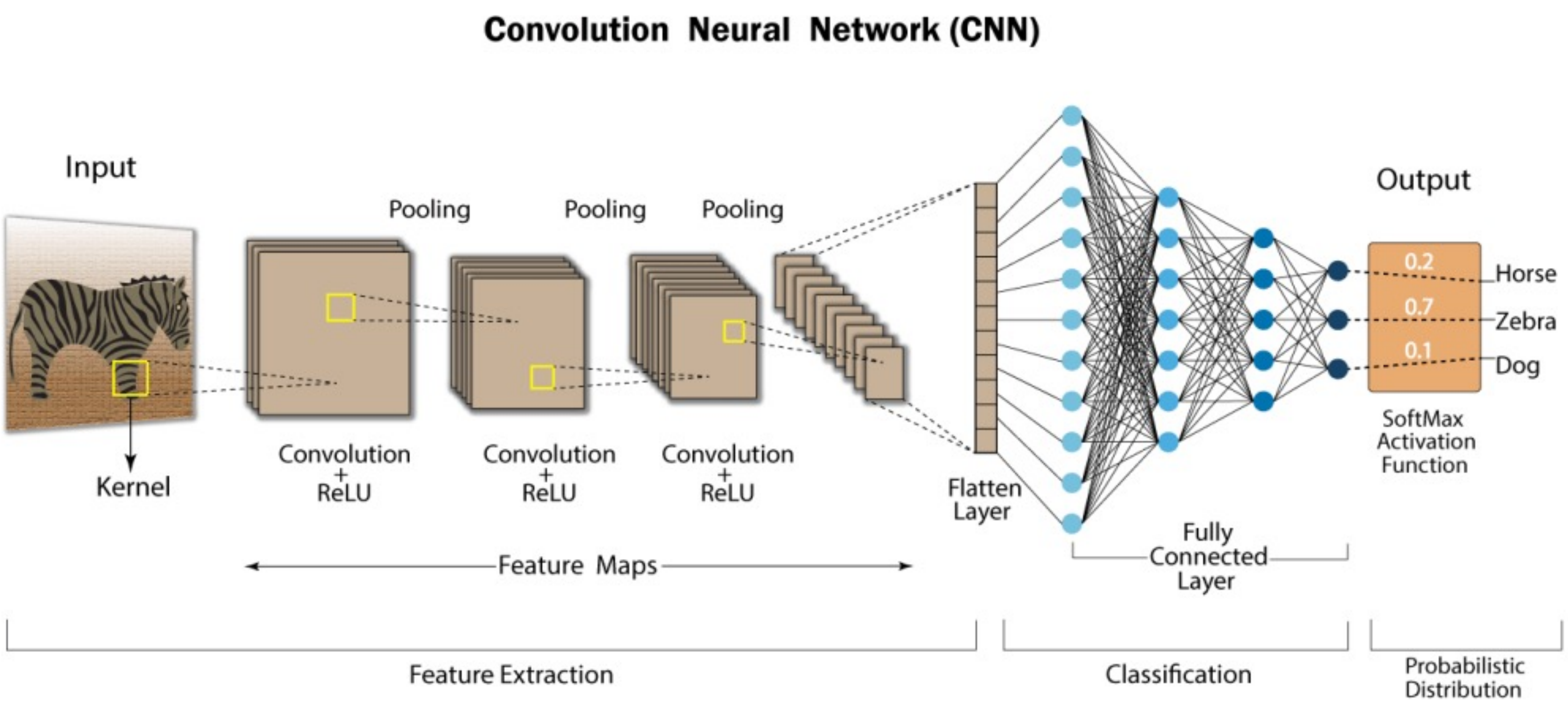
- N_h, η, α → grid optimization

- Optimal values
  - N_h : 17
  - η: 0.55
  - α : 0.6

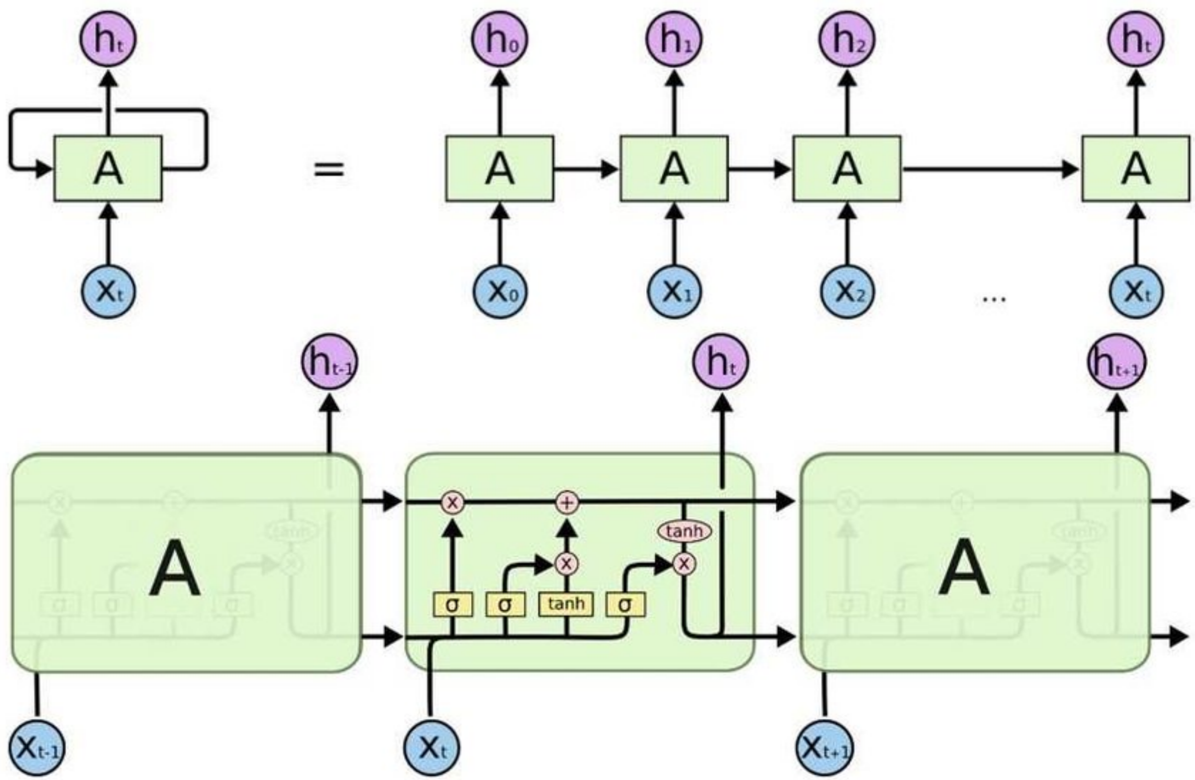
Test results



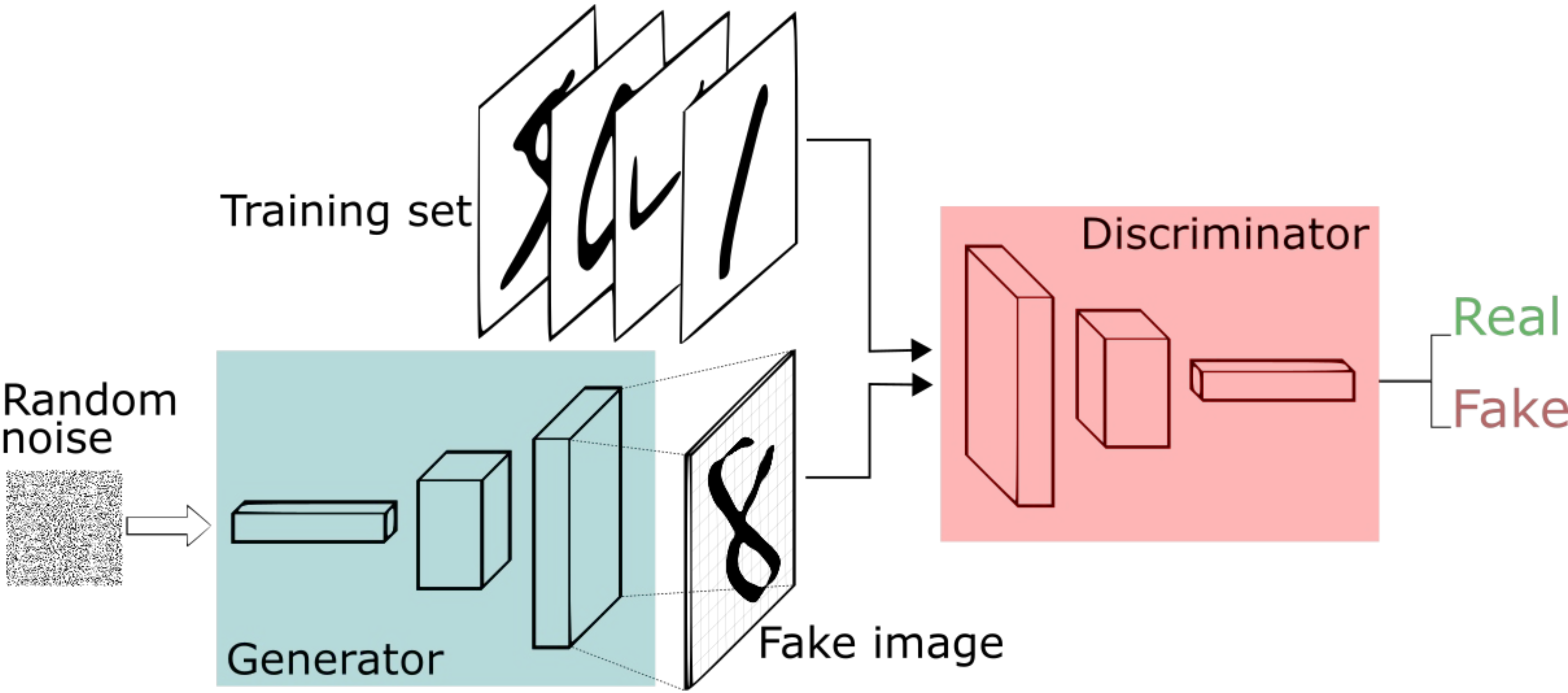
- Convolutional Neural Networks



- Recurrent Neural Networks

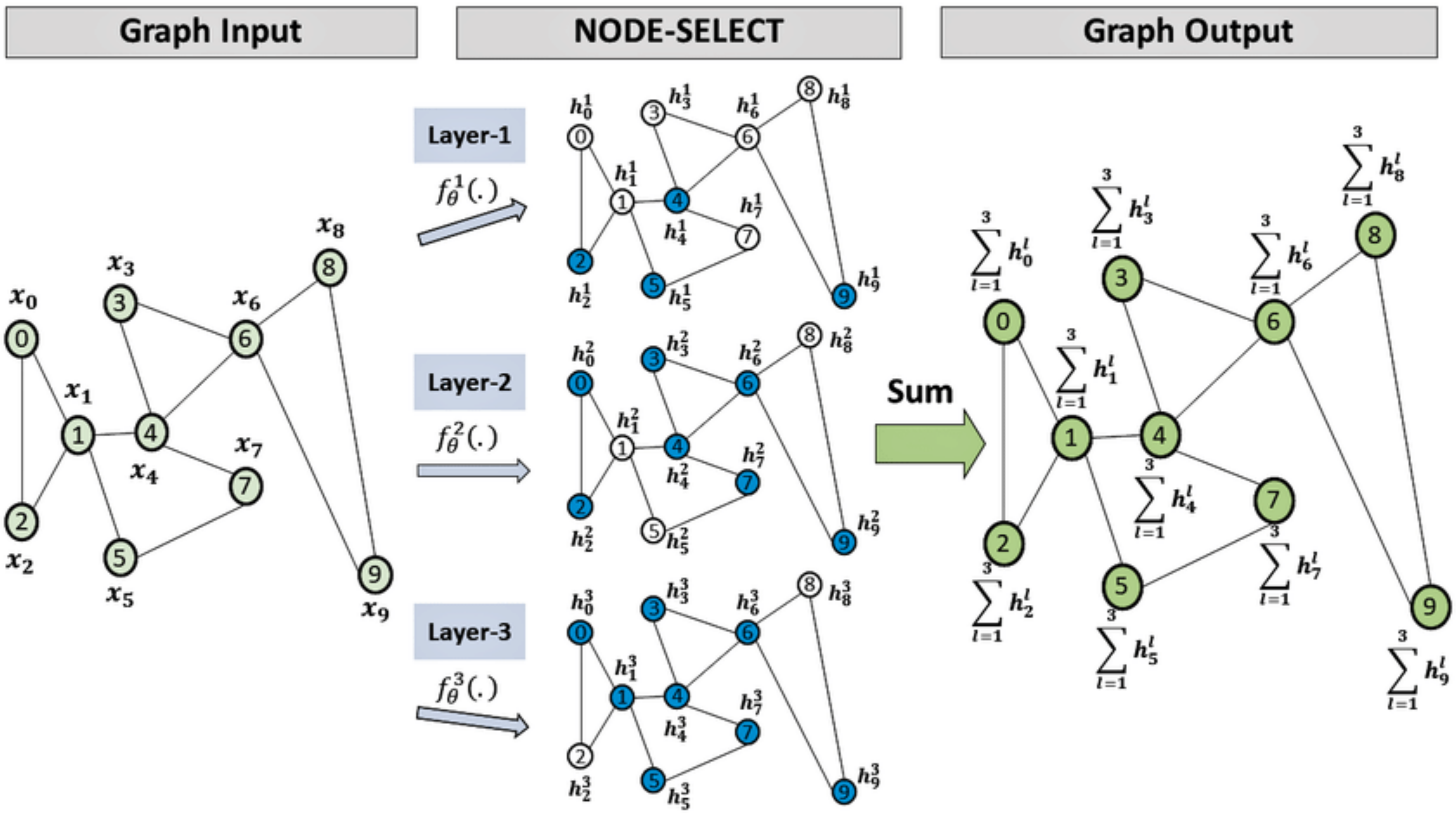


- Generative Adversarial Networks



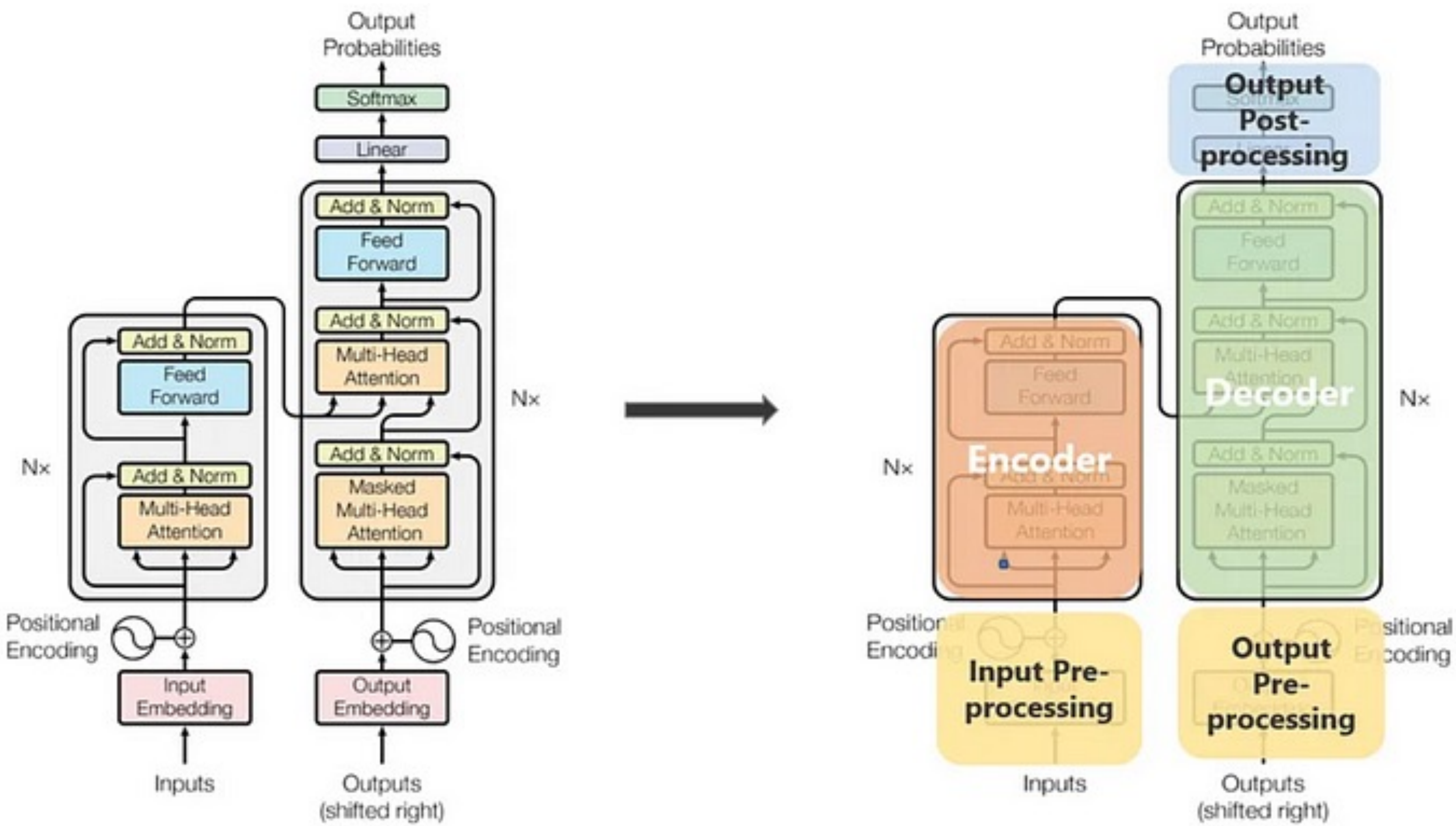
Other kinds of Neural Networks

- Graph Neural Networks



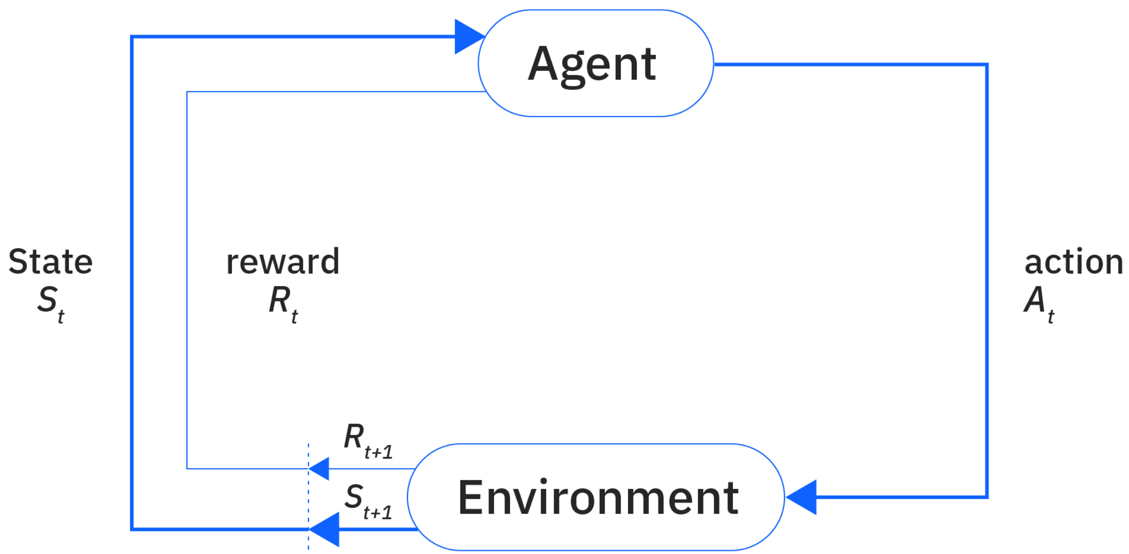
Other kinds of Neural Networks

- Transformers



Other kinds of Neural Networks

- Reinforcement Learning



POLITECNICO
MILANO 1863

lasar³
Laboratory of analysis of systems for the assessment of reliability, risk and resilience

Fault Diagnostics Methods

Joaquín Figueroa

Politecnico di Milano, Energy Department

joaquineduardo.figueroa@polimi.it

April 14th, 2025