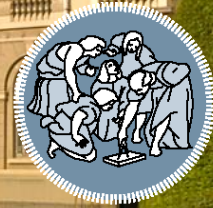


LASAR<sup>3</sup>



POLITECNICO  
MILANO 1863

# *Fault Diagnostics Methods: Neural Networks*

Ibrahim Ahmed  
Politecnico di Milano, Milan, Italy

[ibrahim.ahmed@polimi.it](mailto:ibrahim.ahmed@polimi.it)

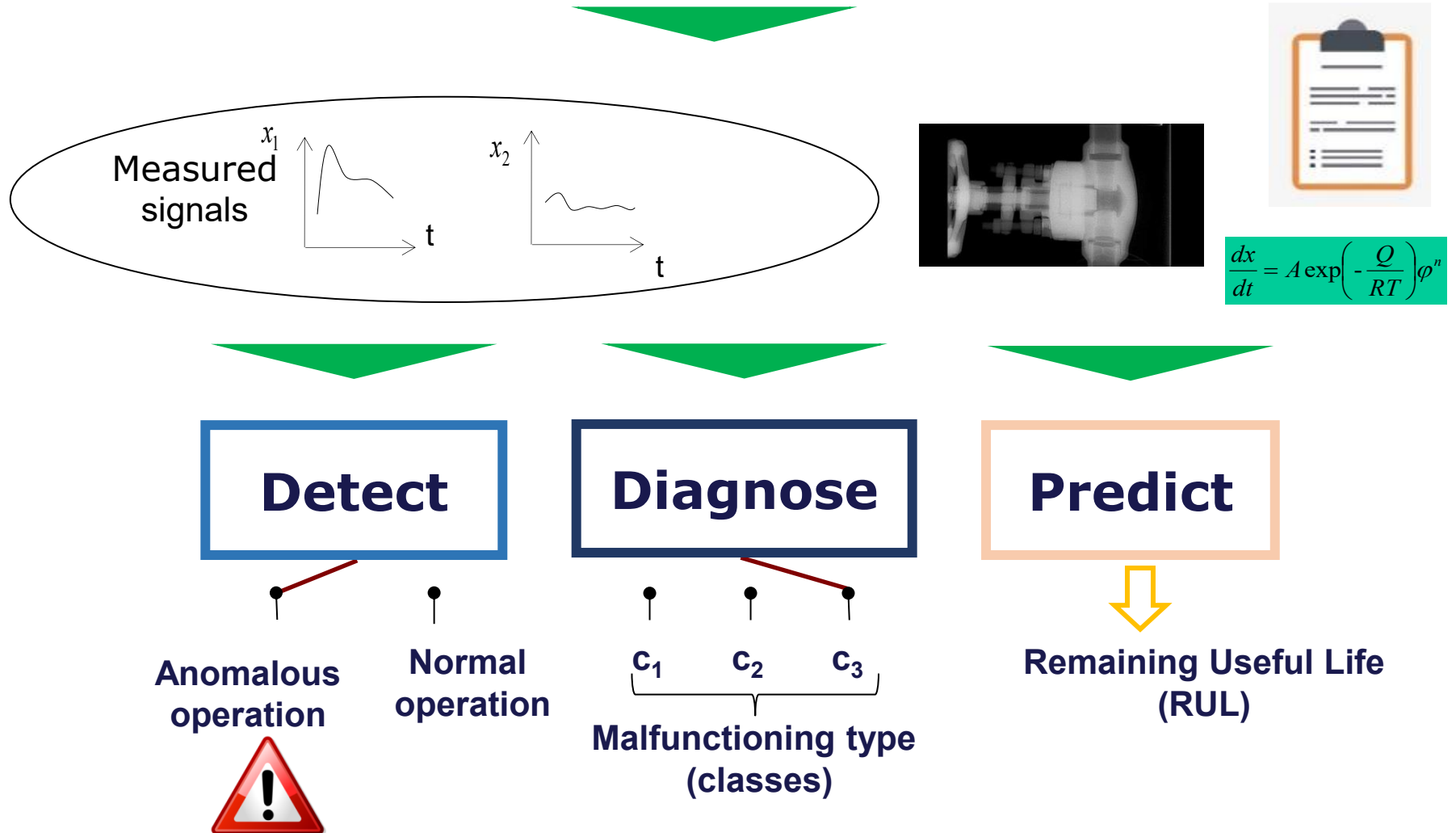
# Today's Lecture

---

1. **Lecture on Fault Diagnostics methods (this material)**
2. **Exercise Session on Fault Diagnostics methods (Please, bring your PC to the class. Note that Google account is required for using colab)**

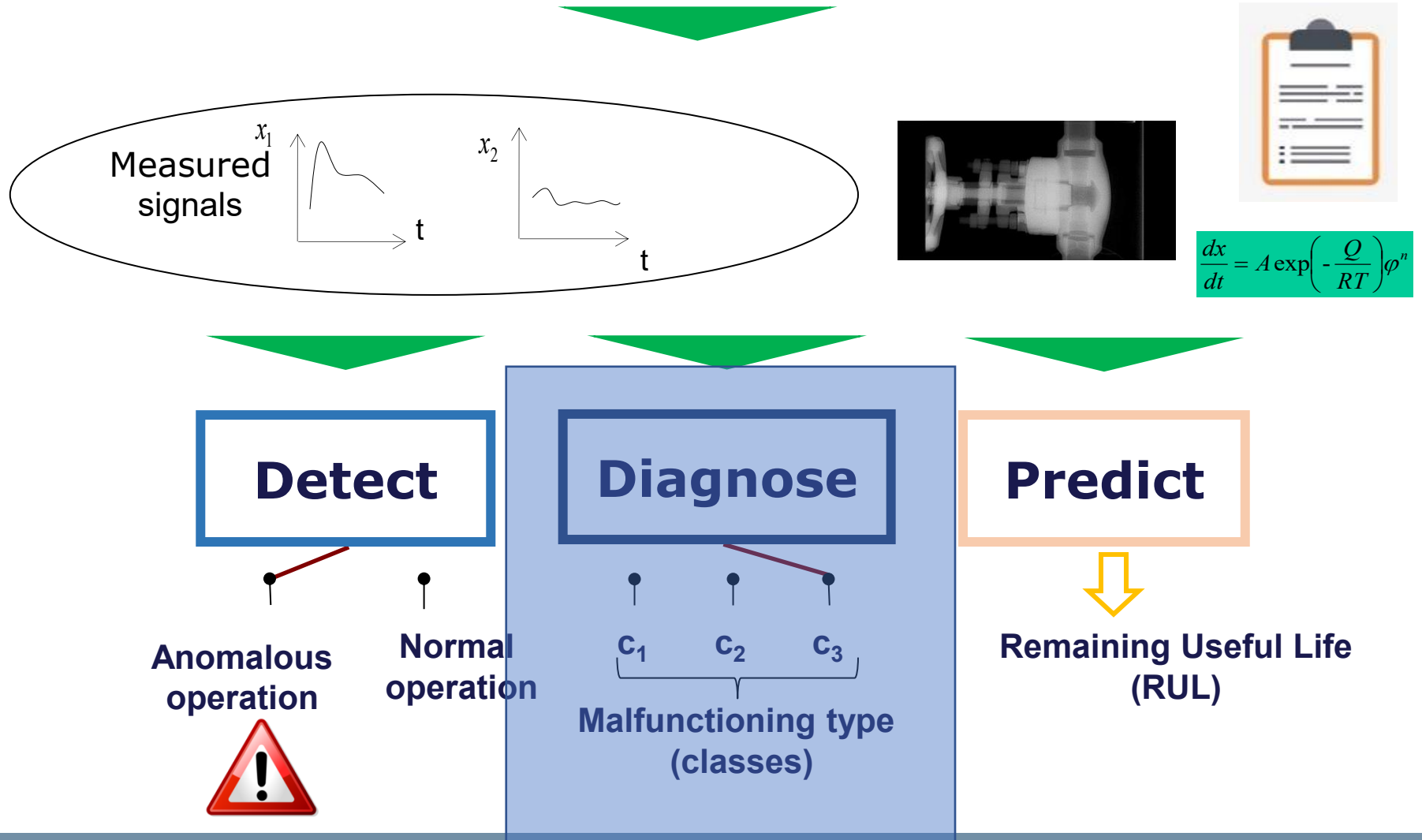
# Context: Prognostics and Health Management

## Equipment (System, Structure or Component)



# In This Lecture: Fault Diagnostics

## Equipment (System, Structure or Component)



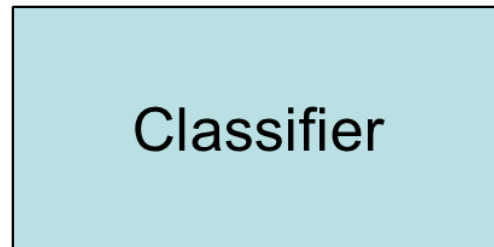
# Fault Diagnostics: Objective

Measured signals

$x_1(t)$  →

$x_2(t)$  →

$x_3(t)$  →



Classifier

• Fault Class 1

• Fault Class 2

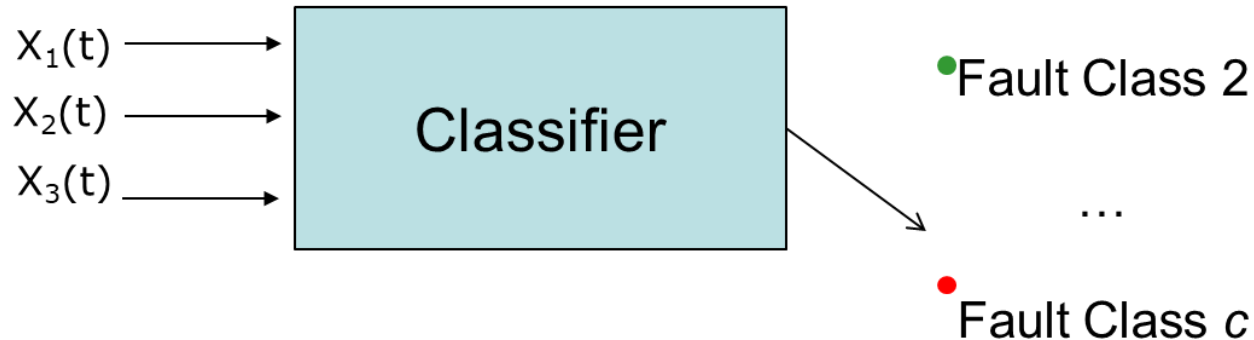
...

• Fault Class  $c$

# Fault Diagnostics: Approaches

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

Measured signals



# 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

8

## 1. Identify fault/degradation classes:

- System Analysis (FMECA, Event Tree Analysis, ...)
- Good engineering sense of practice

{C<sub>1</sub>    C<sub>2</sub>}

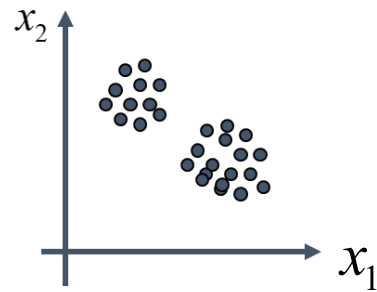


# Procedural steps for developing a fault diagnostic system: Step 2

9

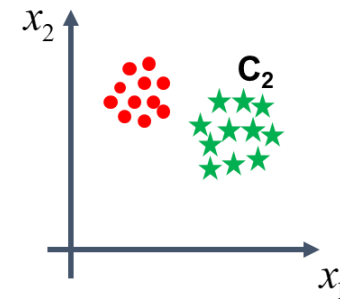
1. Identify fault/degradation classes
2. Analysis of the historical data

Case 1



$$\begin{aligned} &\langle \text{measurements, unknown class} \rangle \\ &= \\ &\langle x_1(k), x_2(k), ? \rangle \end{aligned}$$

Case 2



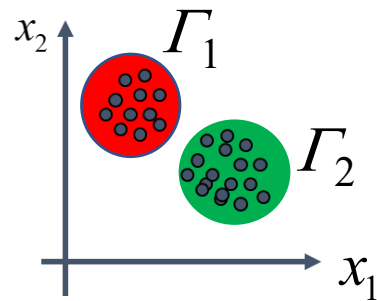
$$\begin{aligned} &\langle \text{measurements, class} \rangle \\ &= \\ &\langle x_1(k), x_2(k), c(k) \rangle \end{aligned}$$

# Procedural steps for developing a fault diagnostic system: Step 3

10

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

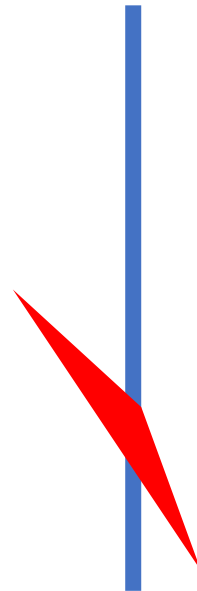
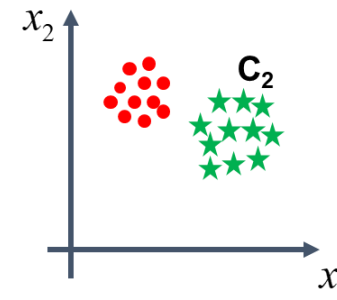
Case 1: clustering



Expert

What is  $\Gamma_2$  ?

Case 2: classification

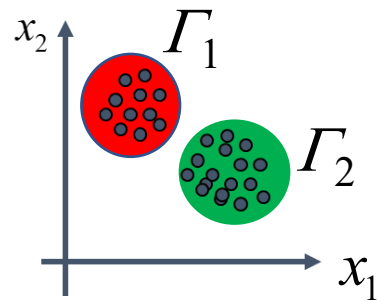


# Procedural steps for developing a fault diagnostic system: Step 3

11

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

Case 1: clustering

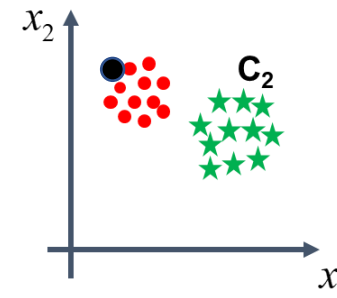


What is  $\Gamma_2$  ?



Expert

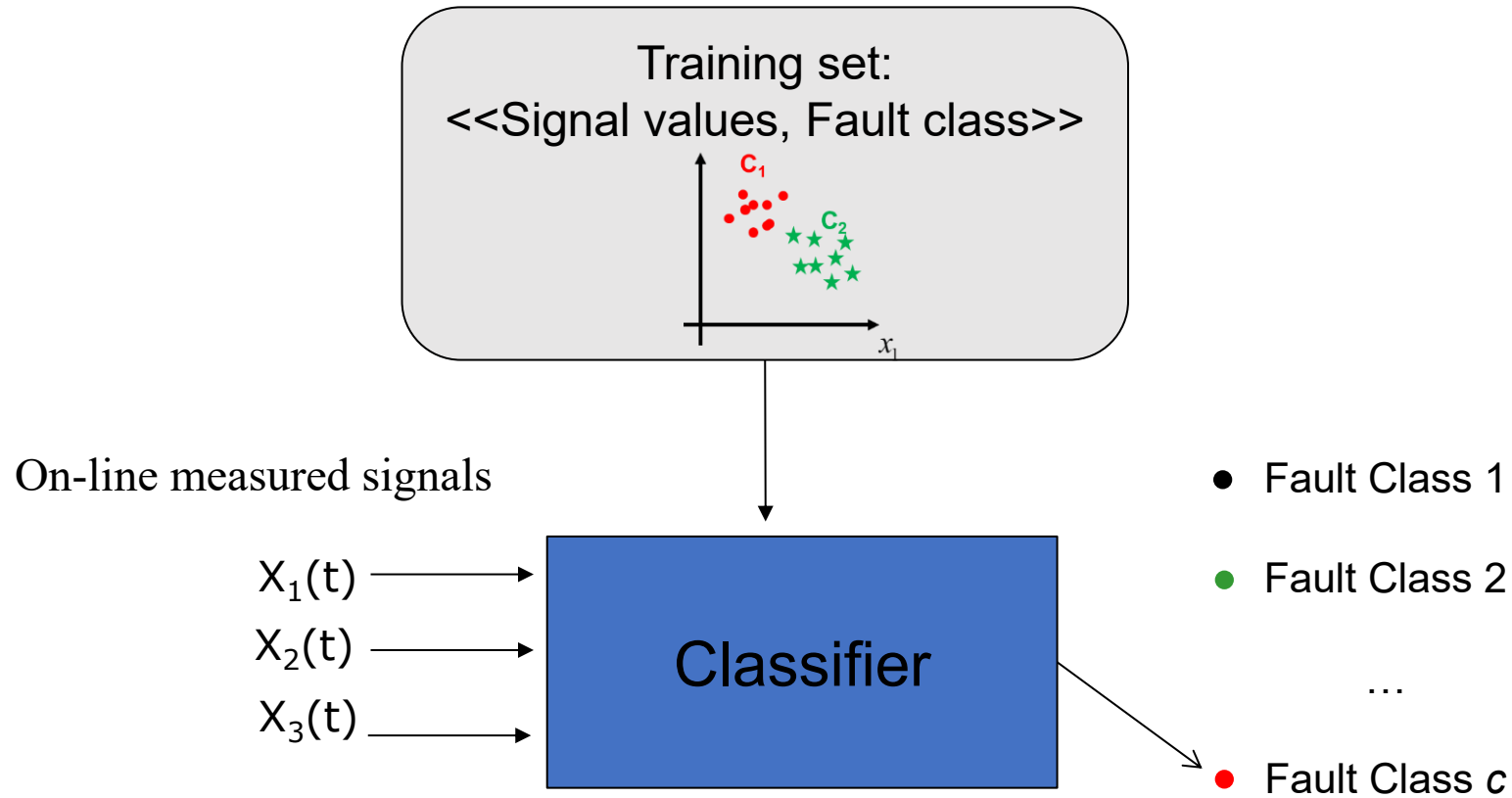
Case 2: classification



# In This Lecture

---

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



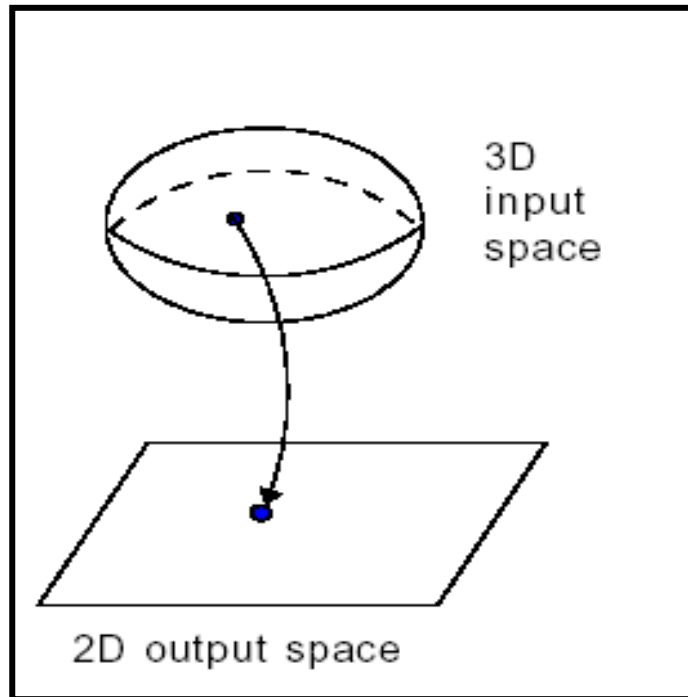
- K-Nearest Neighbor Classifier
- Support Vector Machines
- Fuzzy similarity-based approaches
- Artificial Neural Networks (ANNs)
  - Deep Neural Networks (DNNs)
  - Convolutional Neural Networks (CNNs)
  - Generative Adversarial Networks (GANs)
- Neurofuzzy Systems
- Relevant Vector Machines
- Ensemble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

- 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)

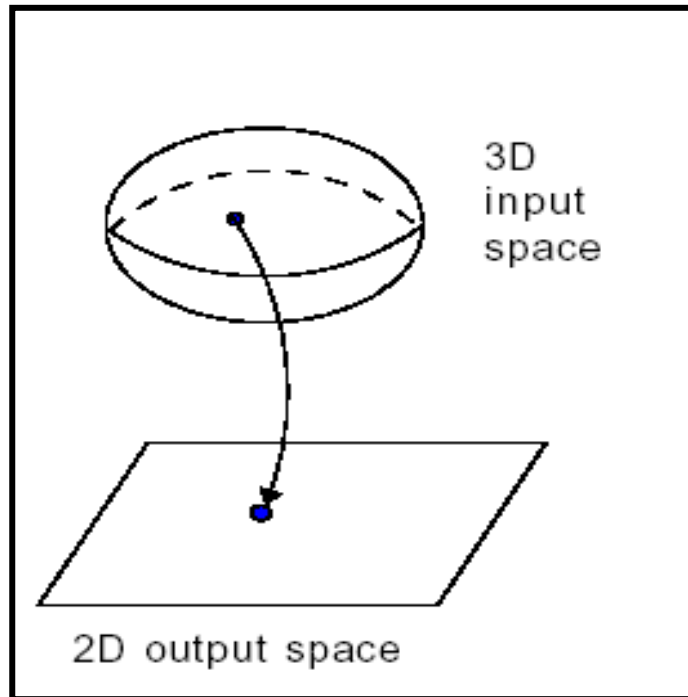
$$t = g(x)$$



$g$  is unknown!

- $g$  is highly non linear
- $g$  is complex

$$t = g(x)$$



$g$  is unknown!

- $g$  is highly non linear
- $g$  is complex



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

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

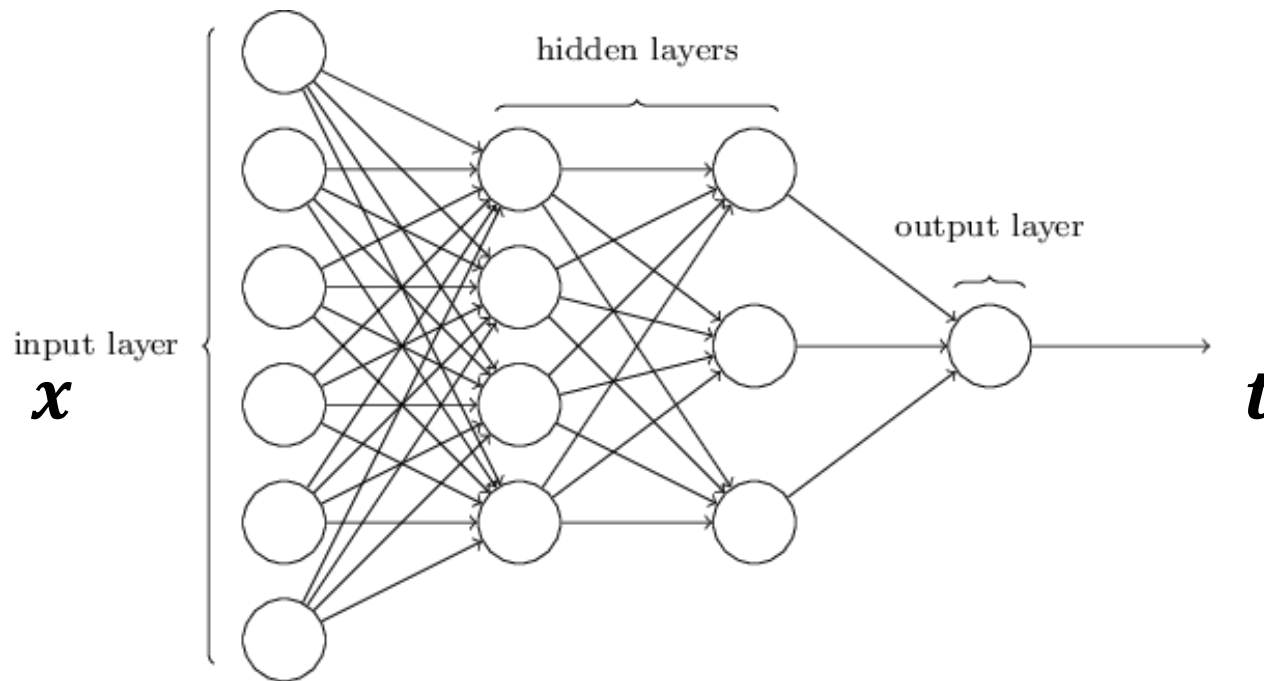
$$x_1^{(2)}, x_2^{(2)}, x_2^{(2)} \mid t_1^{(2)}, t_2^{(2)}$$

...

# What Are (Artificial) Neural Networks?

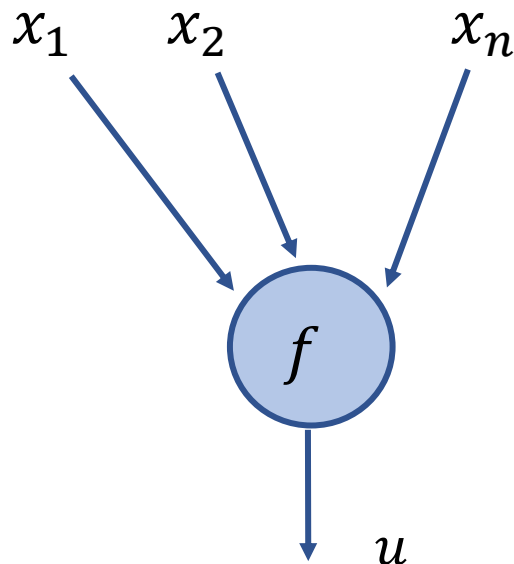
19

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




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.

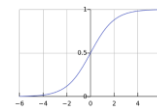


$$x^* = \sum_{i=1}^n x_i$$


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

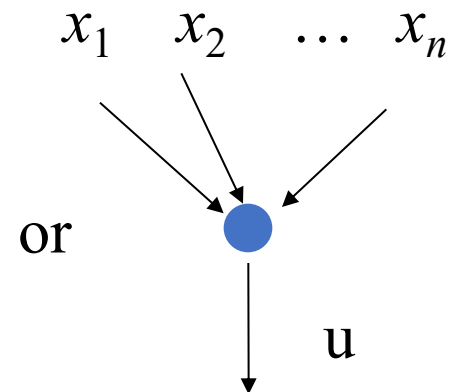
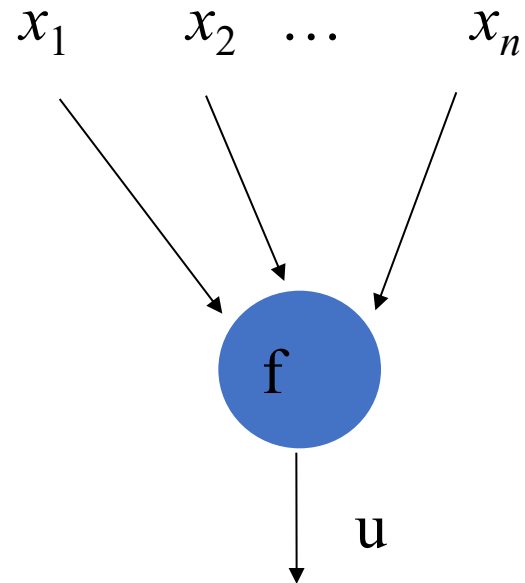
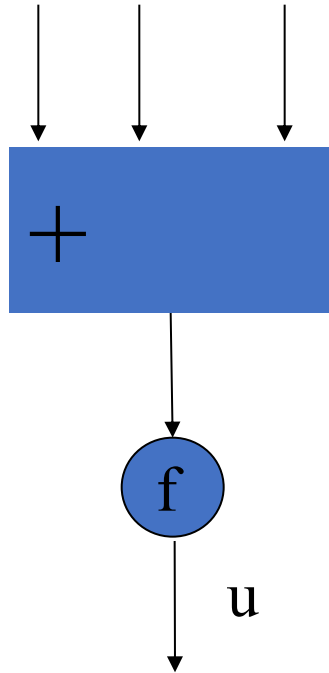
Activation function  $f$ :

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



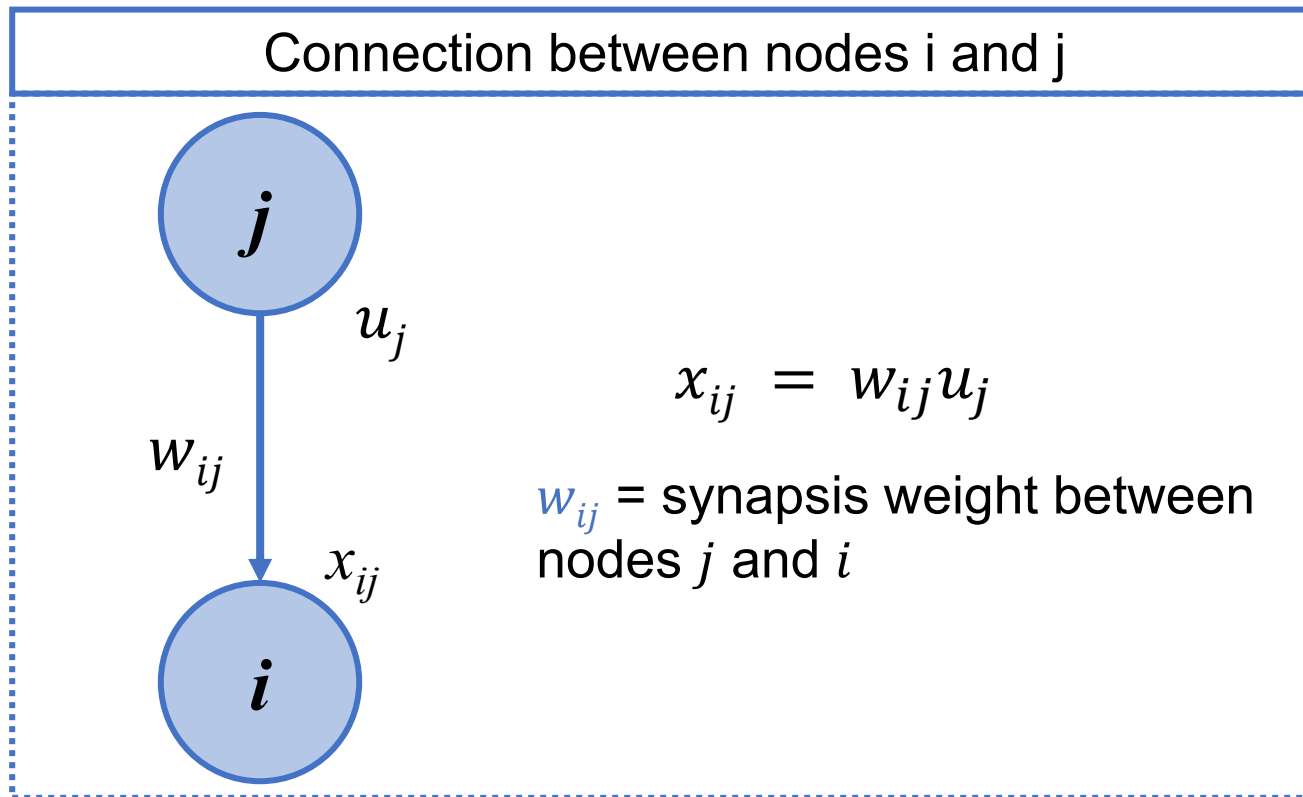
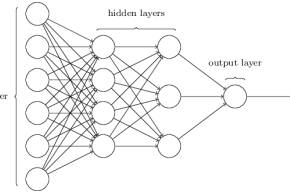
$$f' = f(1 - f)$$

- ReLU  $f(x) = \max(0, x)$
- ...

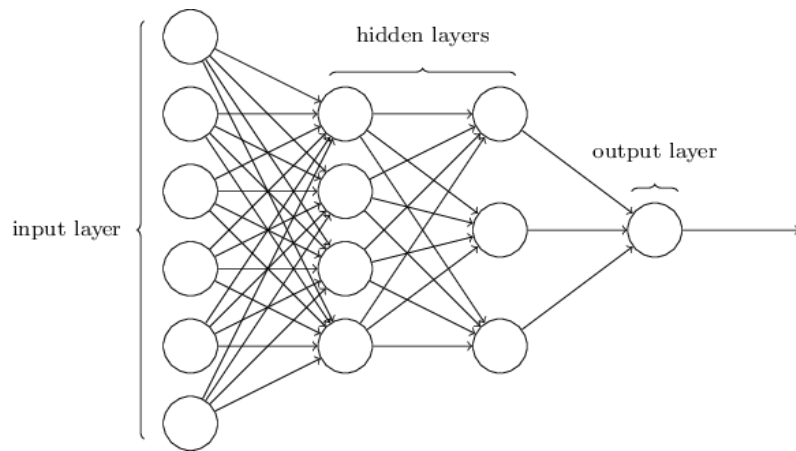


# The weighted connection

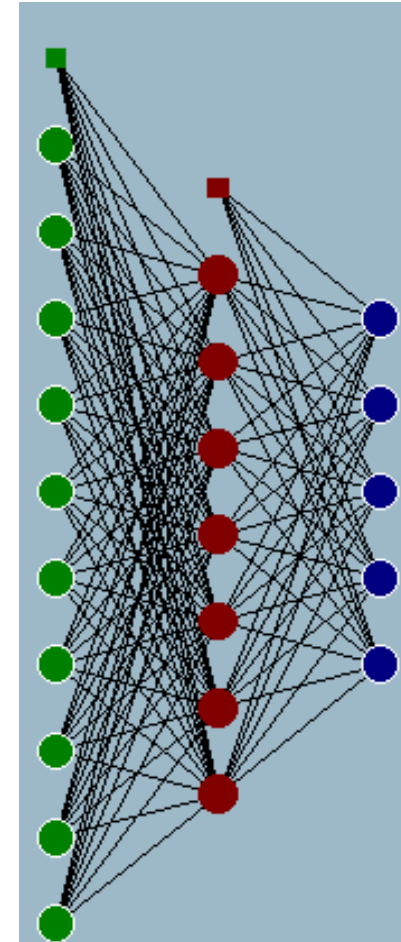
An artificial neural network is composed by several simple **computational units** (also called nodes or neurons) directionally connected by **weighted connections**



## Multilayered Feedforward ANN

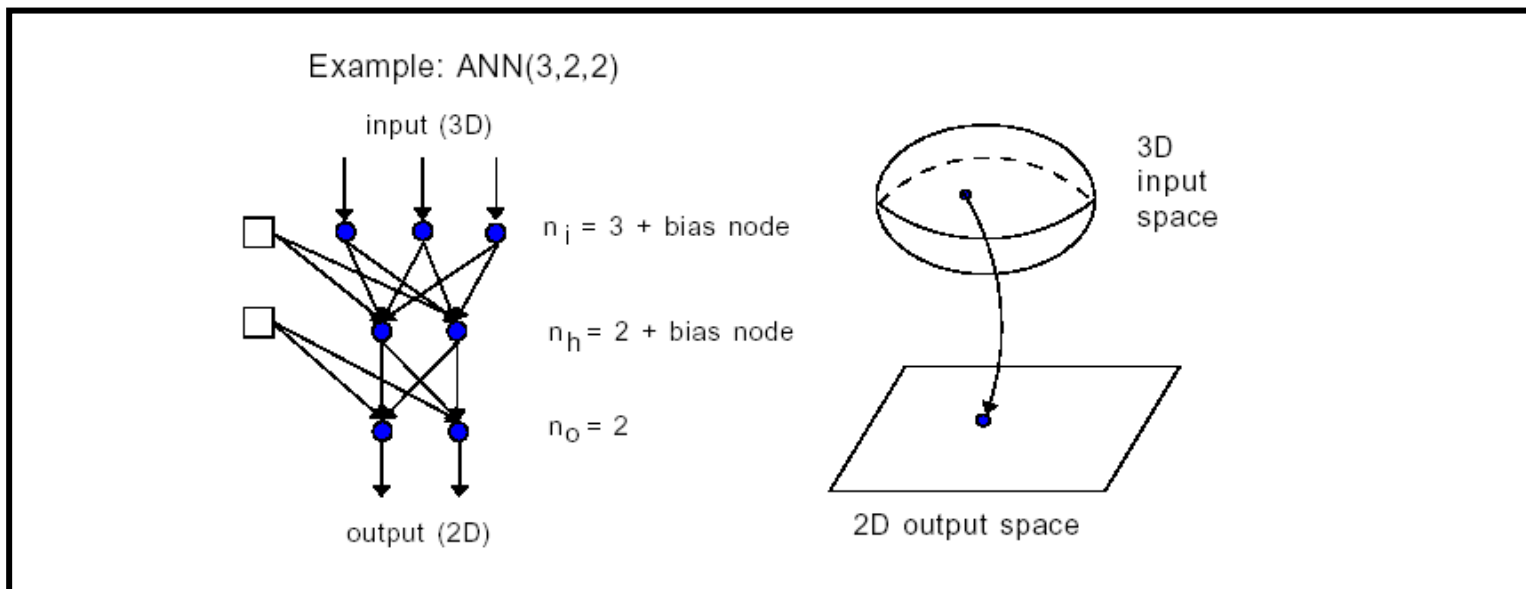


## 1 hidden layer feedforward ANN



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

24



## INPUT LAYER:

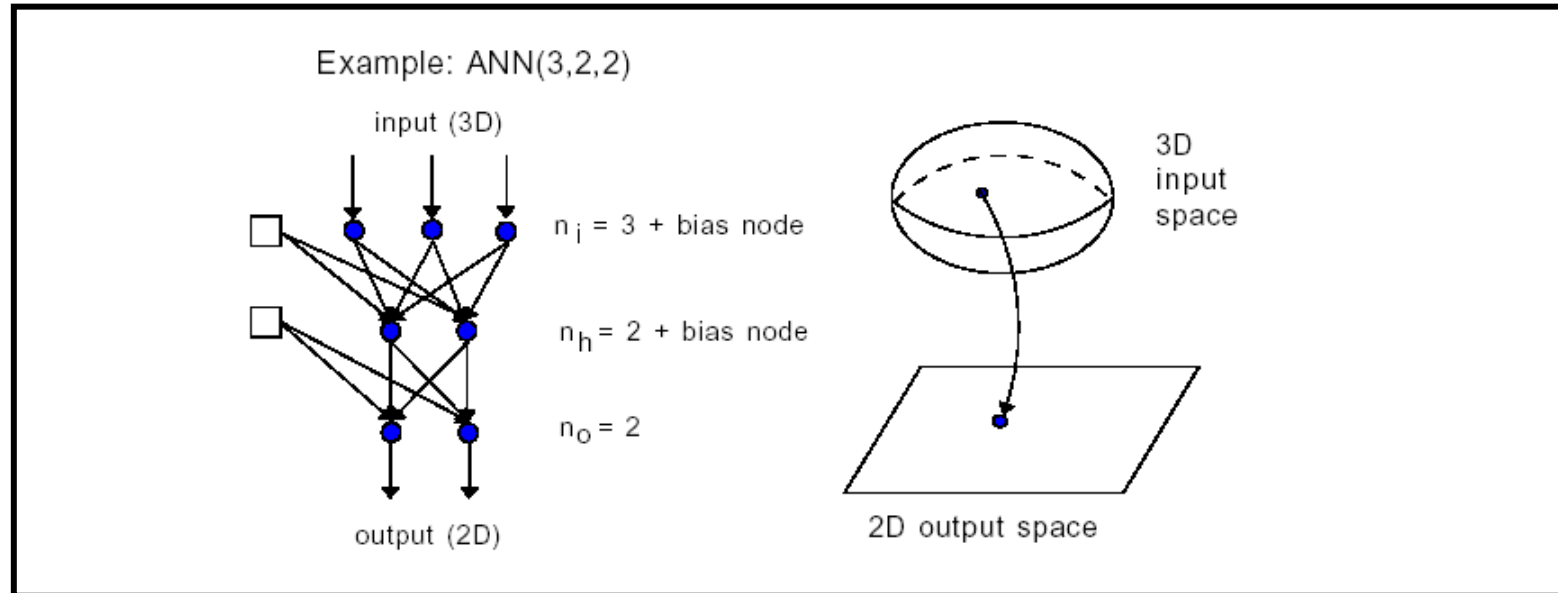
Each  $k$ -th node ( $k = 1, 2, \dots, n_i$ ) receives the value of the  $k$ -th component of the input vector  $x$  and delivers the same value

## HIDDEN LAYER:

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

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

25



## OUTPUT LAYER:

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

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/softmax

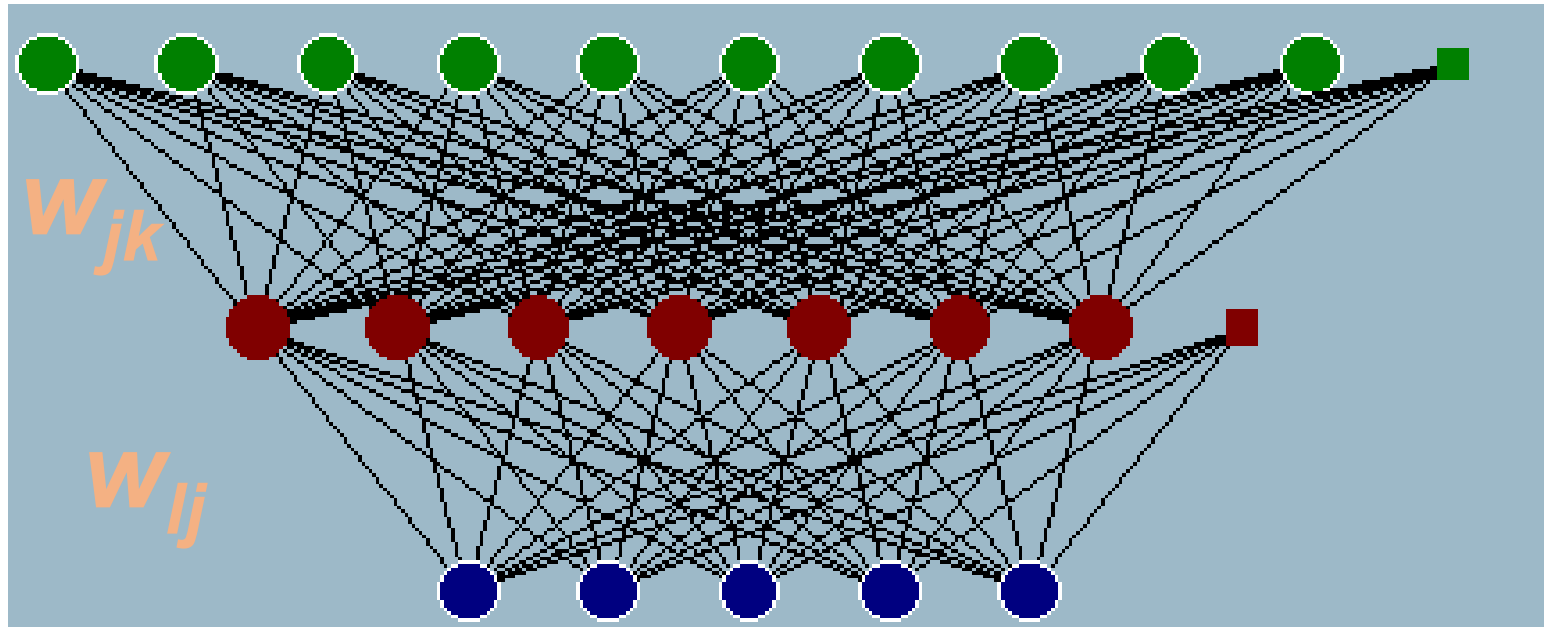
# How to train an Artificial Neural Networks?



# ANN Training: the ANN parameters to be estimated

27

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



## Available input/output patterns

$$x_1^{(1)}, x_2^{(1)} \mid t_{(1)}$$

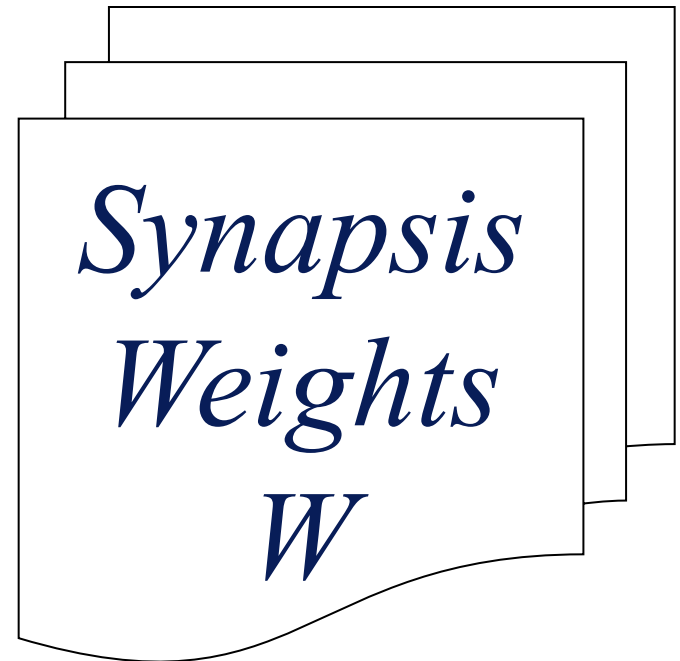
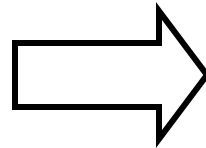
$$x_1^{(2)}, x_2^{(2)} \mid t_{(2)}$$

.....

$$x_1^{(p)}, x_2^{(p)} \mid t_{(p)}$$

.....

$$x_1^{(np)}, x_2^{(np)} \mid t_{(np)}$$



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

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

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

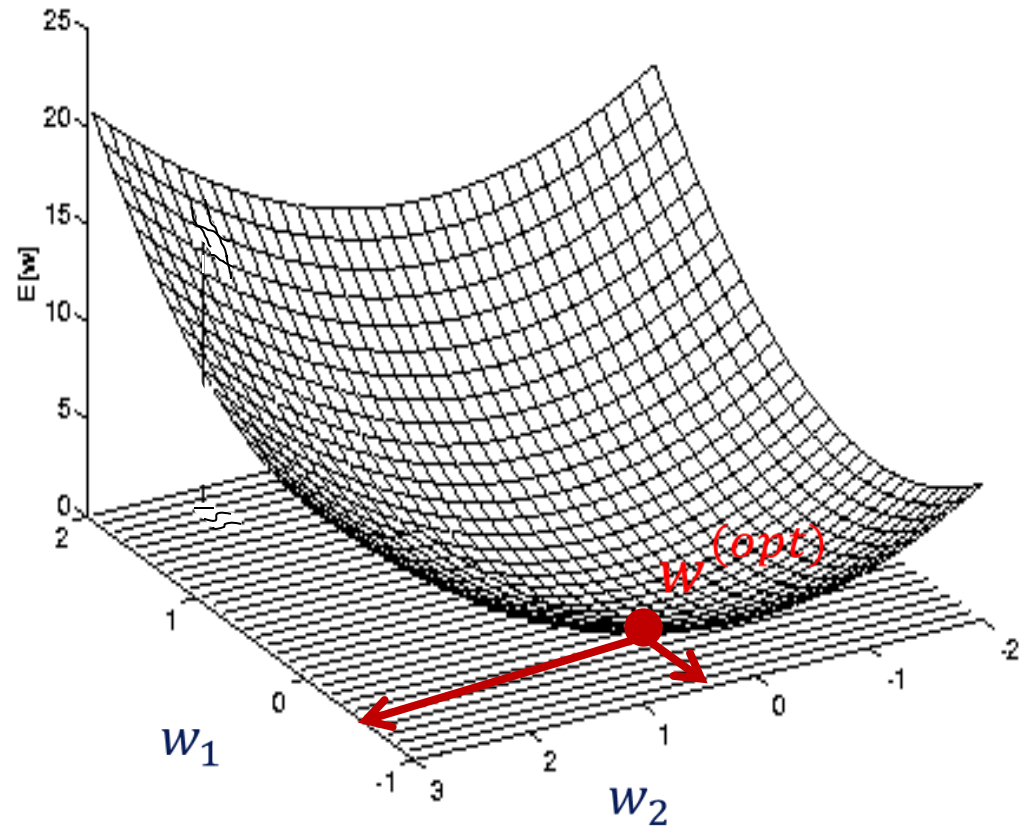
TRUE  $l$ -th output of  
the  $p$ -th training  
pattern

# The Training objective: graphical representation

$E$  is a function of  
the ANN outputs

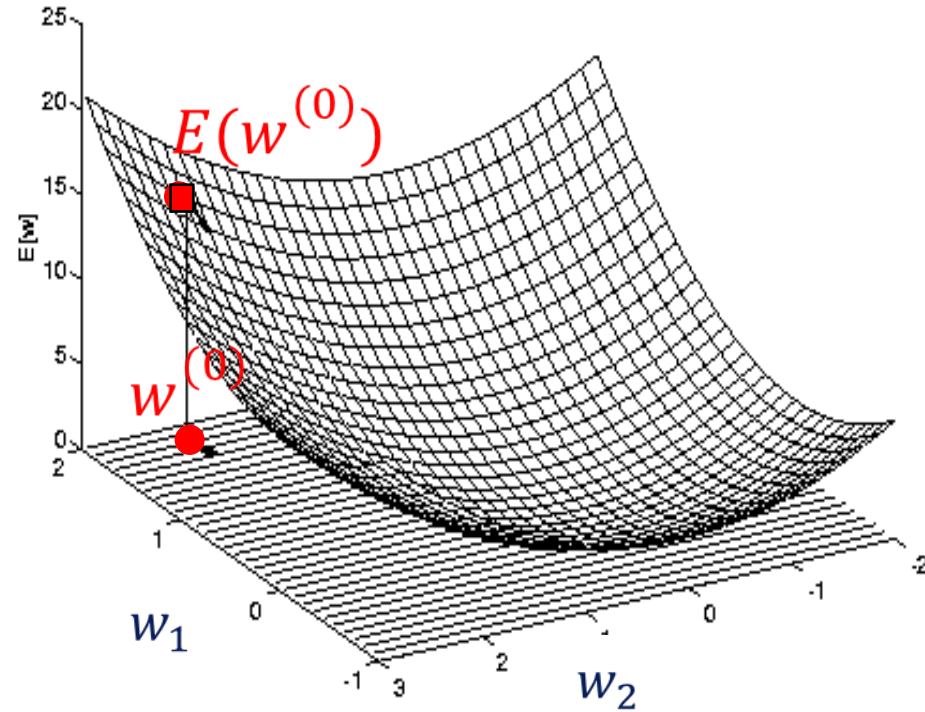


$E$  is a function of the  
synapsis weights  $[w_{jk}]$



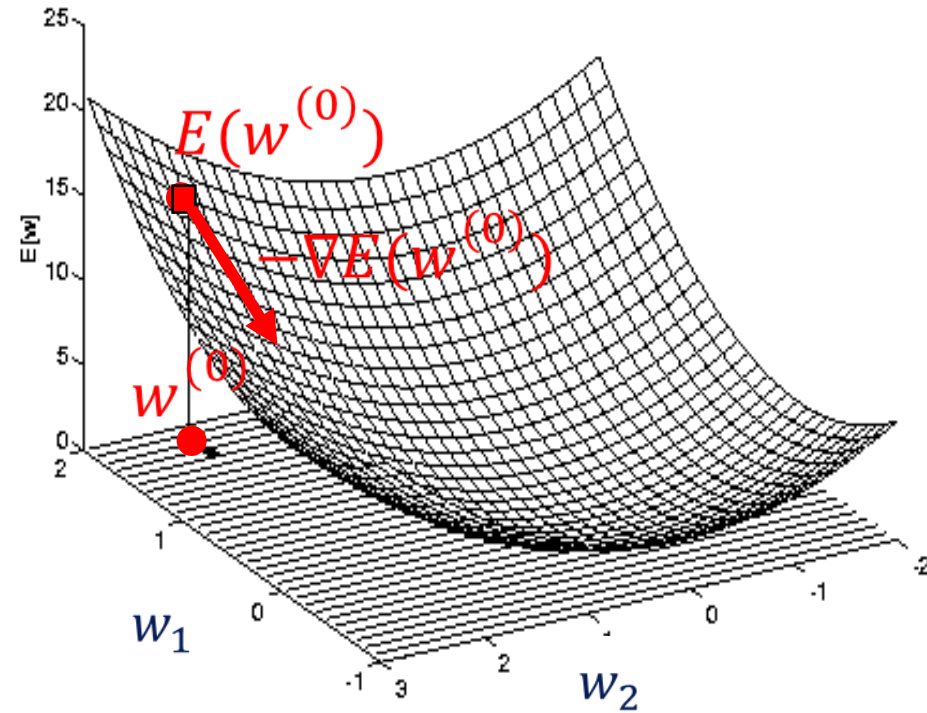
1. Initialize weights to random values:

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



1. Initialize weights to random values:

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



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

1. Initialize weights to random values:

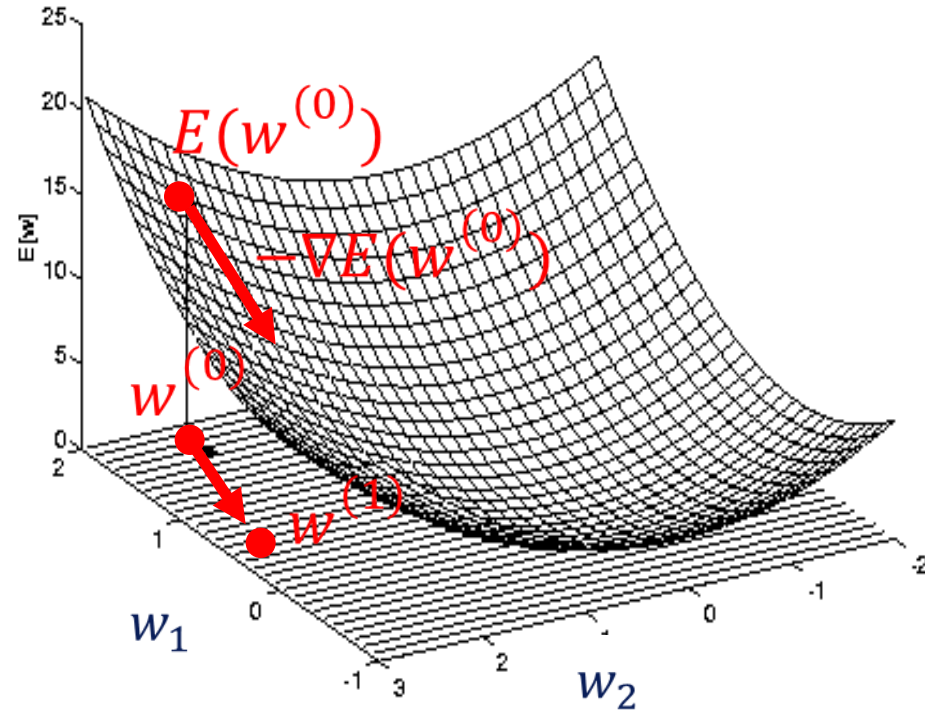
$$w_{jk}^{(0)} = \text{rand}$$

2. While  $E$  is small:

- Update  $w_{jk}^{(i)}$  using the gradient descent method:

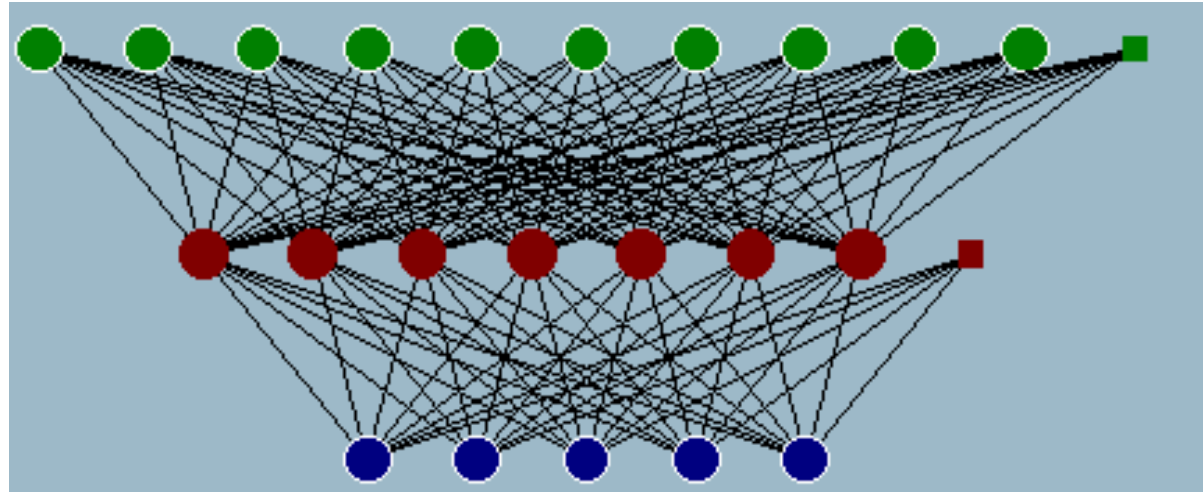
$$w_{jk}^{(i+1)} = w_{jk}^{(i)} - \eta \frac{\partial E}{\partial w_{jk}}$$

Learning coefficient



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

# Example



Error Backpropagation for weights of the  
Hidden-output layer

- Updating of  $w_{lj}$ :

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

with

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

- Updating of  $w_{lj}$ :

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

with

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

- Without loss of generality, set  $n_p=1$

$$E = \frac{1}{2n_o} \sum_{l=1}^{n_o} (u_l^o - t_l)^2 \quad \rightarrow \quad \frac{\partial E}{\partial w_{lj}} = ?$$

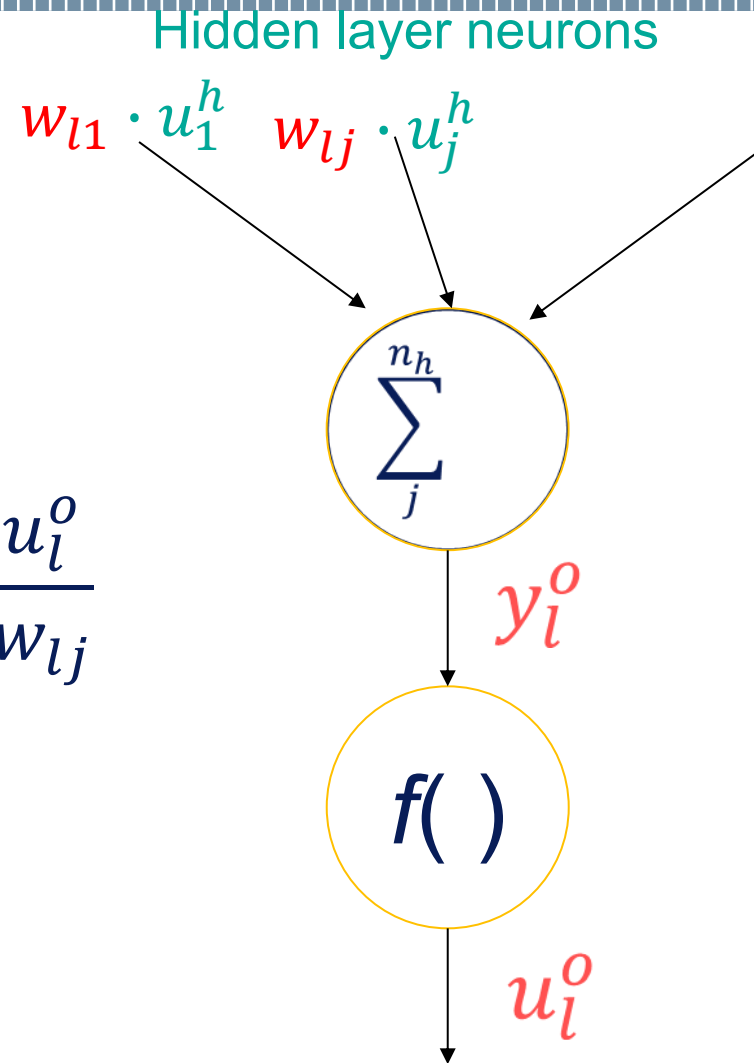
# Error Backpropagation (hidden-output)

37

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



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



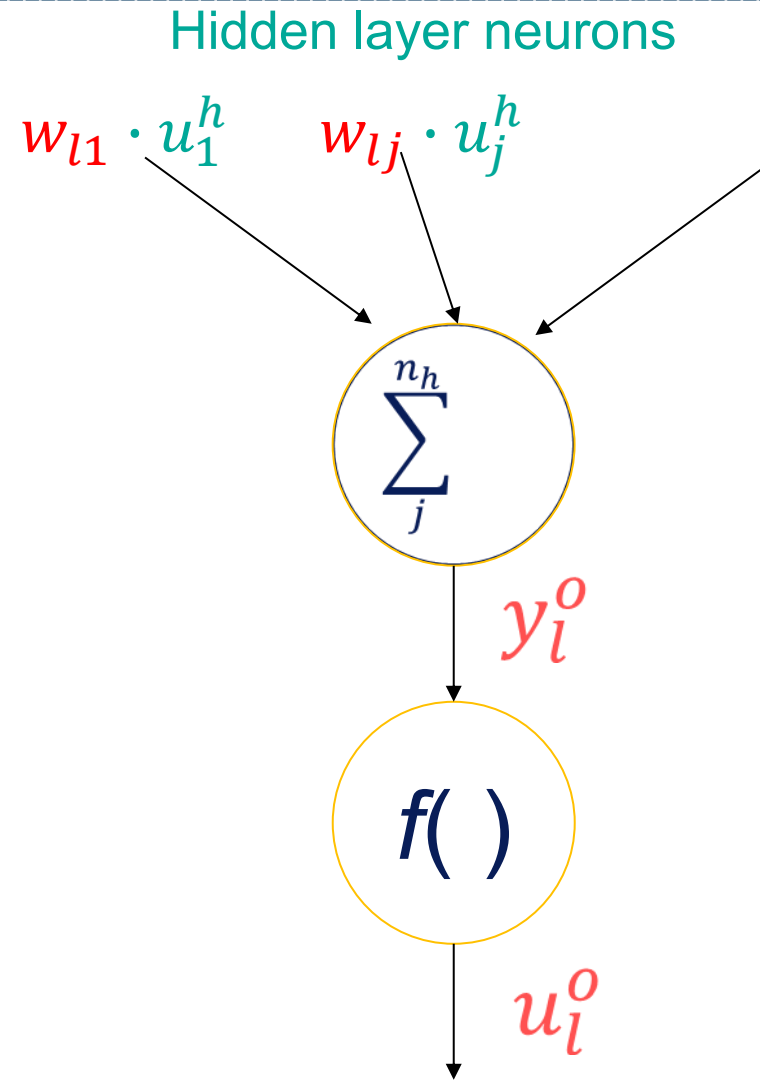
Output layer neuron

# Error Backpropagation (hidden-output)

38

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

$$\begin{aligned} \frac{\partial E}{\partial w_{lj}} &= \frac{2(u_i^o - t_l)}{2n_o} \frac{\partial u_i^o}{\partial w_{lj}} \\ &= \frac{(u_i^o - t_l)}{n_o} \frac{\partial u_i^o}{\partial y_i^o} \frac{\partial y_i^o}{\partial w_{lj}} = \end{aligned}$$



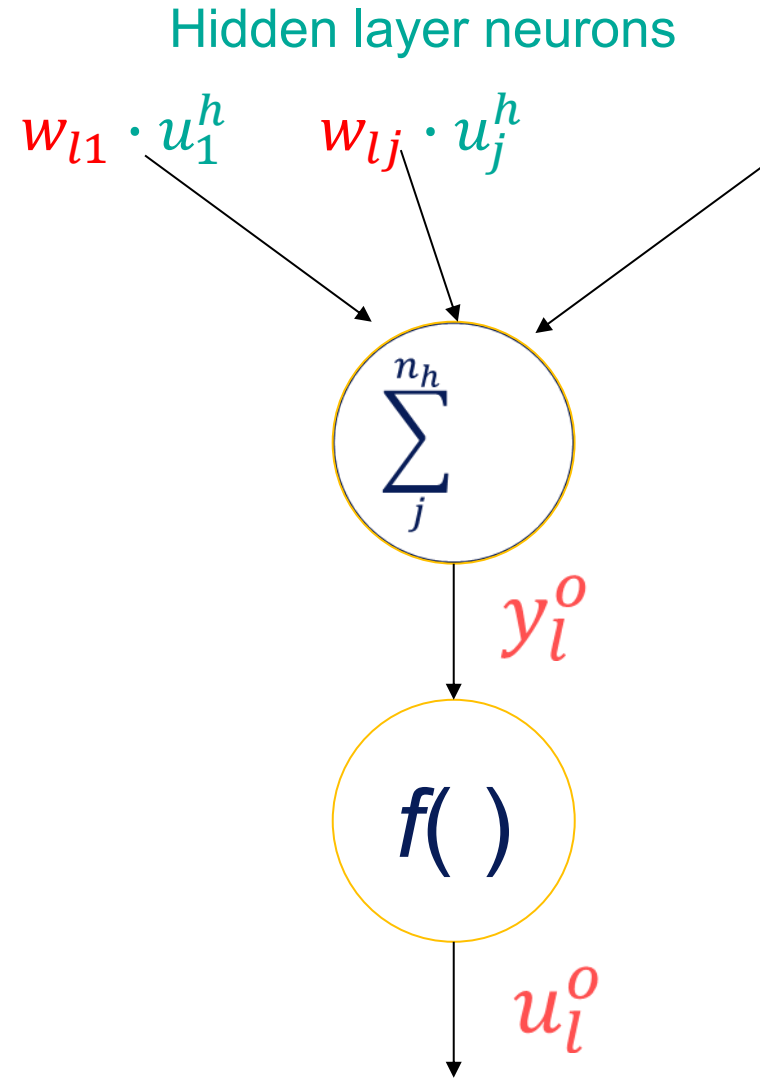
Output layer neuron

# Error Backpropagation (hidden-output)

39

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

$$\begin{aligned} \frac{\partial E}{\partial w_{lj}} &= \frac{2(u_l^o - t_l)}{2n_o} \frac{\partial u_l^o}{\partial w_{lj}} \\ &= \frac{(u_l^o - t_l)}{n_o} \frac{\partial u_l^o}{\partial y_l^o} \frac{\partial y_l^o}{\partial w_{lj}} \\ &= \frac{(u_l^o - t_l)}{n_o} f'(y_l^o) \frac{\partial (\sum_{j'=1}^{n_h} w_{lj'} \cdot u_{j'}^h + w_{l0})}{\partial w_{lj}} \end{aligned}$$



# Error Backpropagation (hidden-output)

40

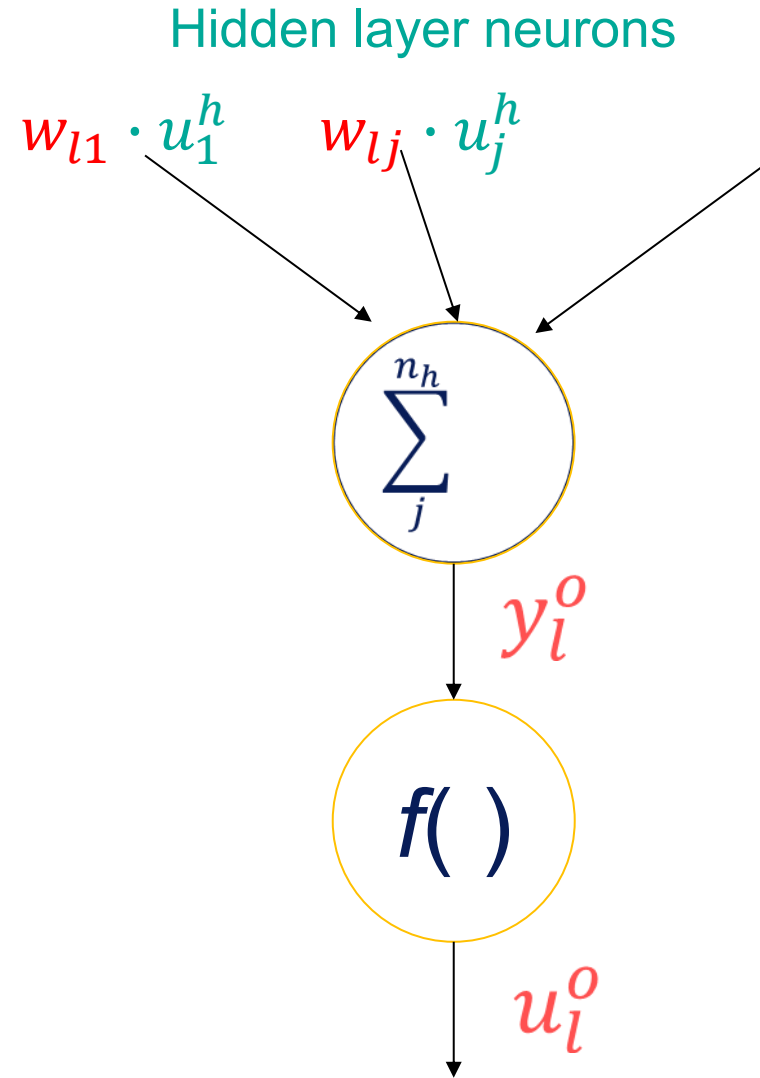
$$E = \frac{1}{2n_o} \sum_{l'=1}^{n_o} (u_{l'}^o - t_{l'})^2$$

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

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

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

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




- Updating of  $w_{lj}$ :

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

with

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

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


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

- Updating of  $w_{lj}$ :

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

with

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

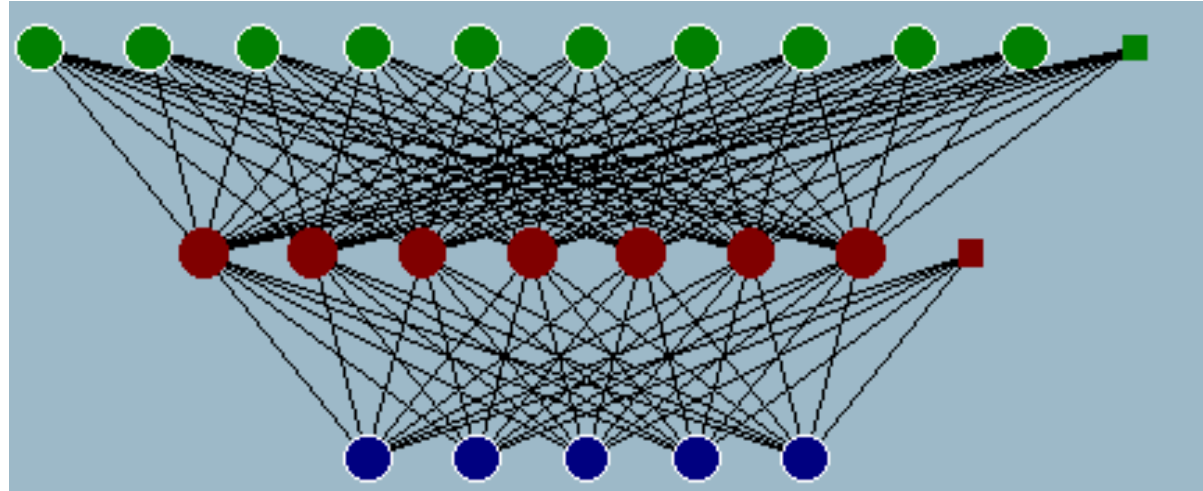


$$w_{lj}^{(i+1)} = w_{lj}^{(i)} + \Delta w_{lj}^{(i)} + \alpha \Delta w_{lj}^{(i-1)}$$

momentum



# Example



Error Backpropagation for weights of the  
input-hidden layer

- Updating of  $w_{kj}$ :

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

with

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

- Without loss of generality, set  $n_p=1$

$$\begin{aligned} \frac{\partial E}{\partial w_{kl}} &= \frac{1}{2n_o} \sum_{l=1}^{n_o} \frac{2(u_l^o - t_l)}{2n_o} \frac{\partial u_l^o}{\partial w_{kj}} = \frac{1}{n_o} \sum_{l=1}^{n_o} (u_l^o - t_l) \frac{\partial u_l^o}{\partial y_l^o} \frac{\partial y_l^o}{\partial u_j^h} \frac{\partial u_j^h}{\partial y_j^h} \frac{\partial y_j^h}{\partial w_{kj}} = \\ &= \sum_{l=1}^{n_o} \frac{(u_l^o - t_l)}{n_o} f'(y_l^o) w_{jl} f'(y_j^h) u_k^i = u_k^i f'(y_j^h) \end{aligned}$$

# Error Backpropagation (hidden-input)

Similarly to the updating of the output-hidden weights,

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

$$\begin{aligned} \frac{\partial E}{\partial \bar{w}_{jk}} &= \frac{1}{n_o} \sum_{l=1}^{n_o} \phi_l^+(u_l^o - t_l) \frac{\partial u_l^o}{\partial y_l^o} \frac{\partial y_l^o}{\partial u_j^h} \frac{\partial u_j^h}{\partial y_j^h} \frac{\partial y_j^h}{\partial \bar{w}_{jk}} \quad \text{being } \begin{matrix} y_k^i = x_k, u_k^i = y_k^i \\ y_j^h = \sum_{k=1}^{n_i} \bar{w}_{jk} u_k^i + \bar{w}_{j_o}, u_j^h = f(y_j^h) \end{matrix} \\ &= \frac{1}{n_o} \sum_{l=1}^{n_o} \phi_l^+(u_l^o - t_l) f'(y_l^o) w_{lj} f'(y_j^h) u_k^i \\ &= -\frac{1}{n_o} \bar{\delta}_j u_k^i \end{aligned}$$

$$\bar{\delta}_j = f'(y_j^h) \sum_{l=1}^{n_o} \delta_l w_{lj}$$

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

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

Learning coefficient

Momentum

## **Advantages:**

- No physical/mathematical modelling efforts
- Able to learn nonlinear mappings
- After training: very fast computations

## **Disadvantages:**

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

# Application

A decorative horizontal line consisting of a series of vertical bars of varying heights, creating a textured effect. The bars are light blue and set against a white background. Below this line is a solid blue rectangular area that occupies the bottom half of the page.

- **Objective:**

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

- **Input/Output patterns**

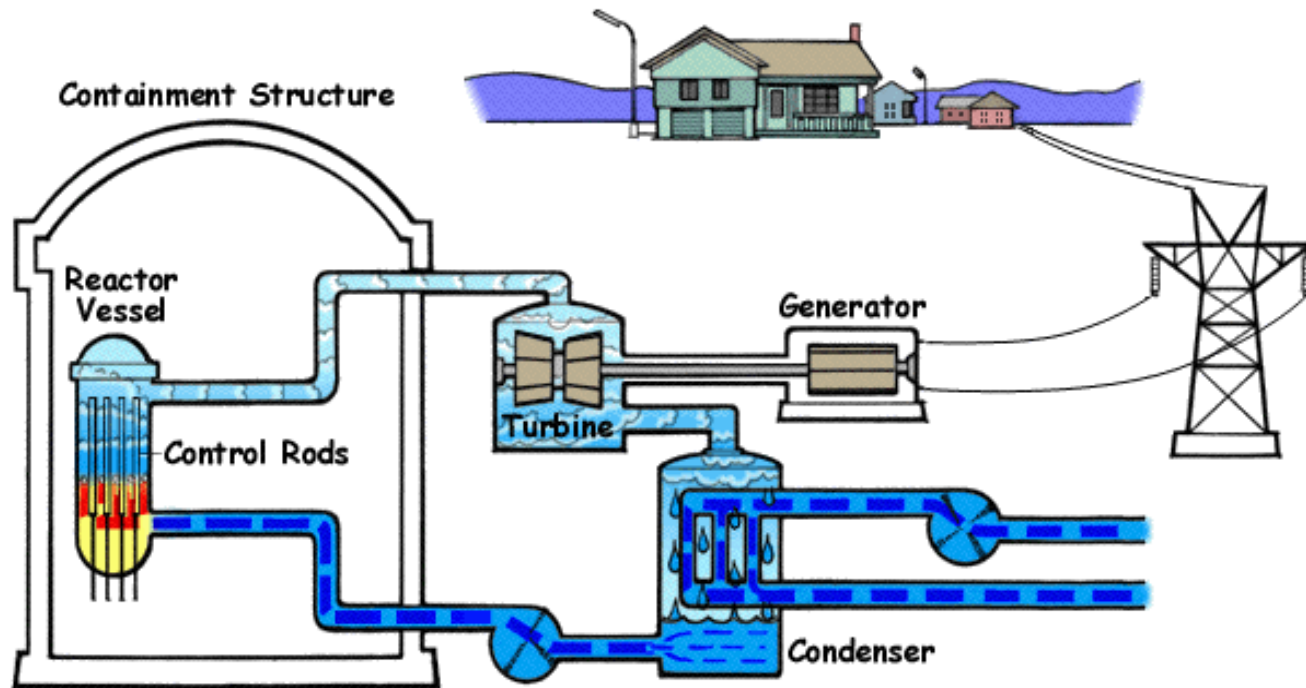
Input: signal measurements

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

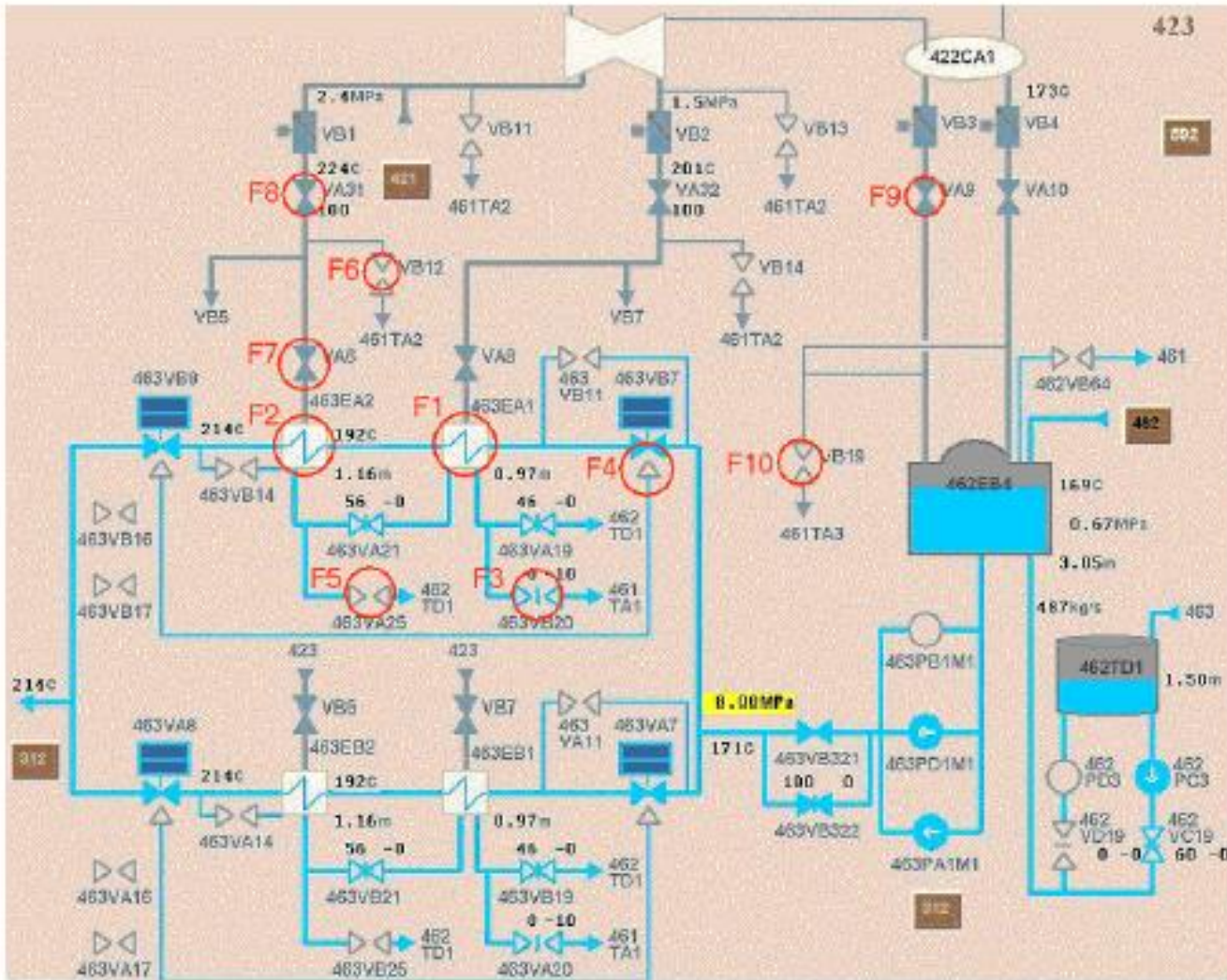
# ANN Example: Boiling Water Reactor

49

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



# ANN Example: Secondary System



- 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

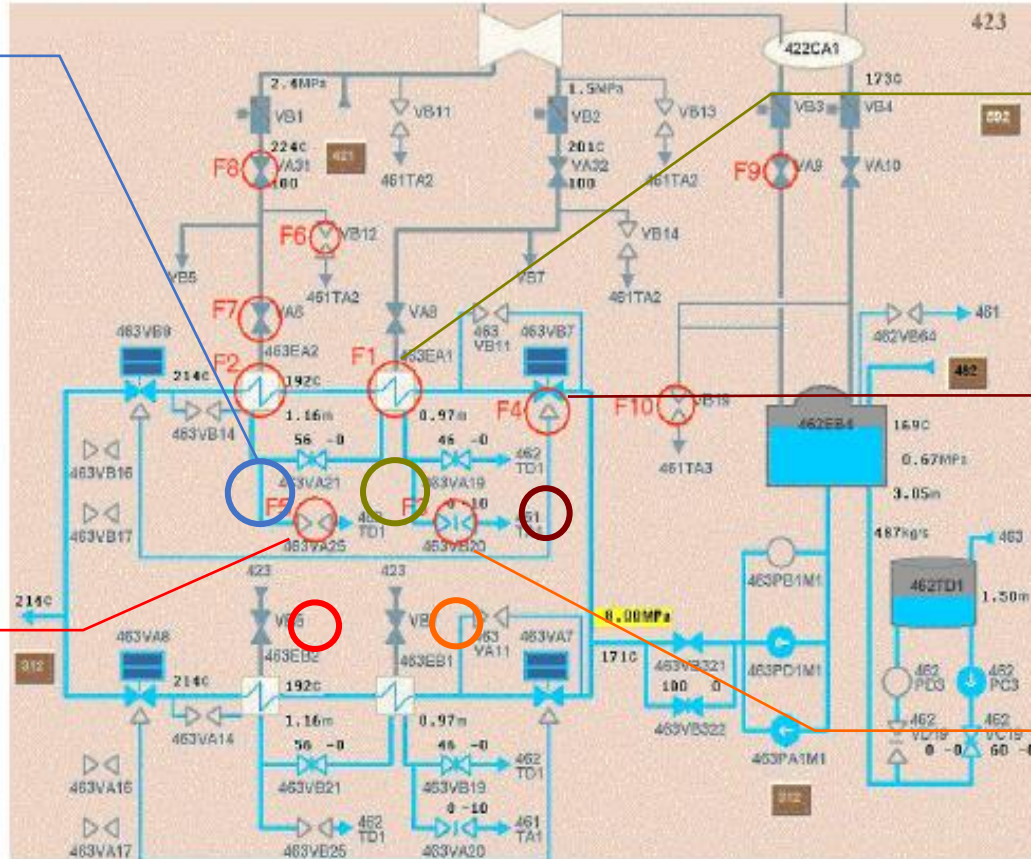
**Class 2:  
Preheater  
EA2**

**Class 1:  
Preheater  
EA1**

**Class 4:  
Valve VB7**

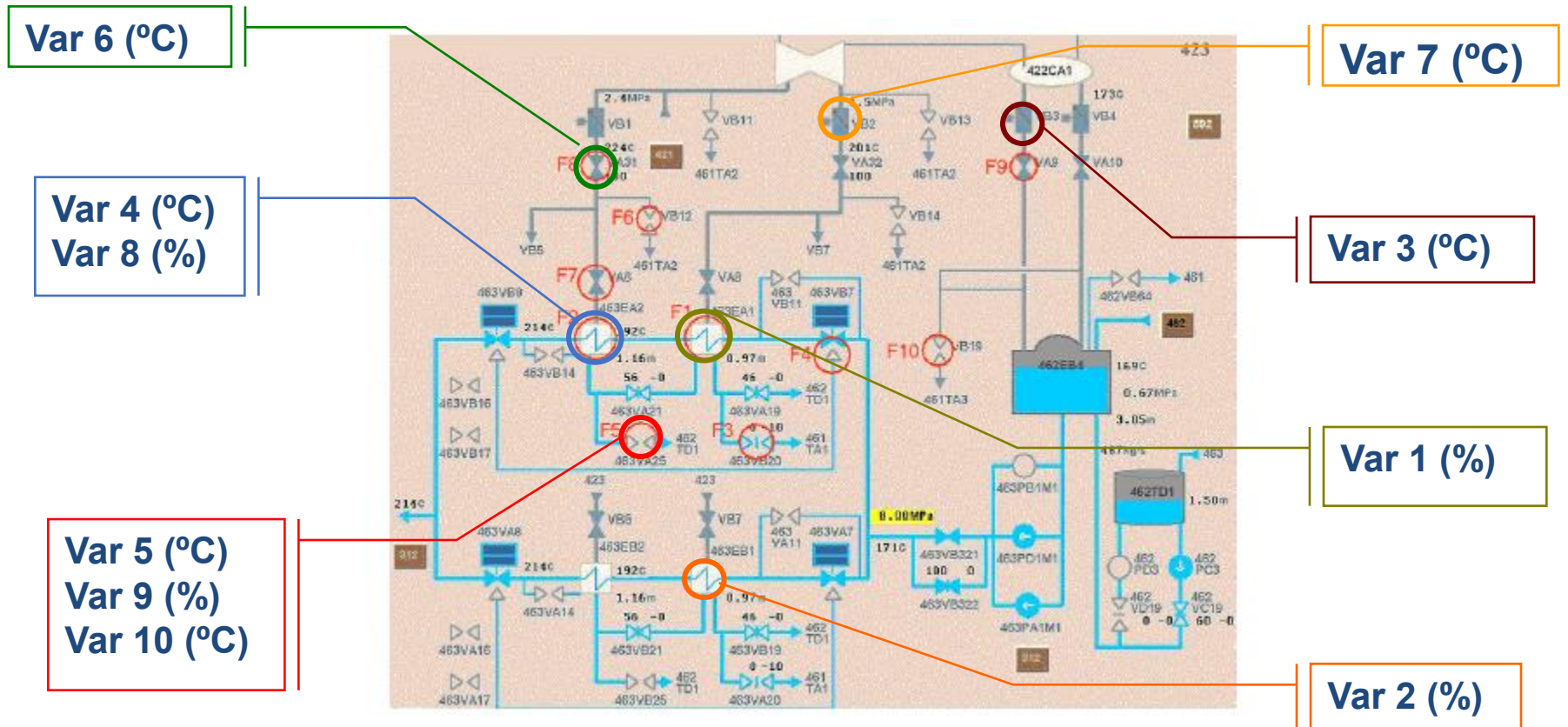
**Preheater 5:  
Valve VA25**

**Class 3:  
Valve VB20**

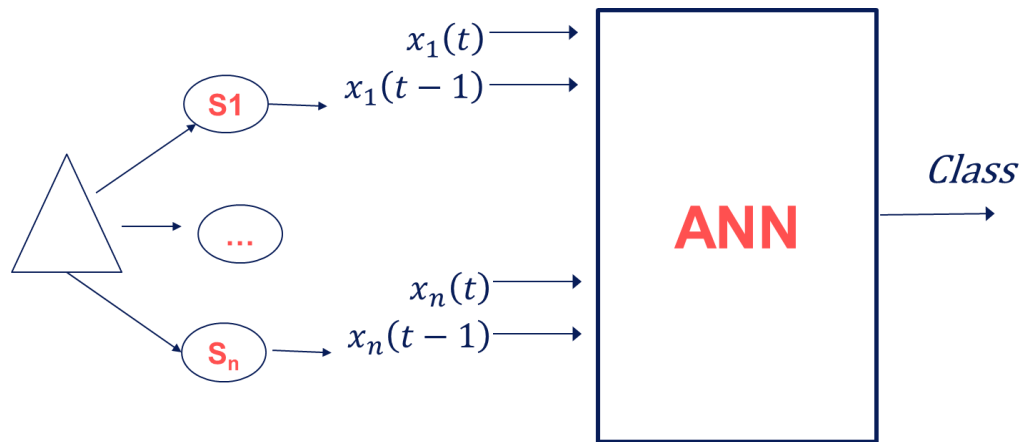


Variable	Signal	Unit
1	Position level for control valve EA1	%
2	Position level for control valve EB1	%
3	Temperature drain before VB3	°C
4	Temperature feedwater after EA2 train A	°C
5	Temperature feedwater after EB2 train B	°C
6	Temperature drain 6 after VB1	°C
7	Temperature drain 5 after VB2	°C
8	Position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

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



- 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



- 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

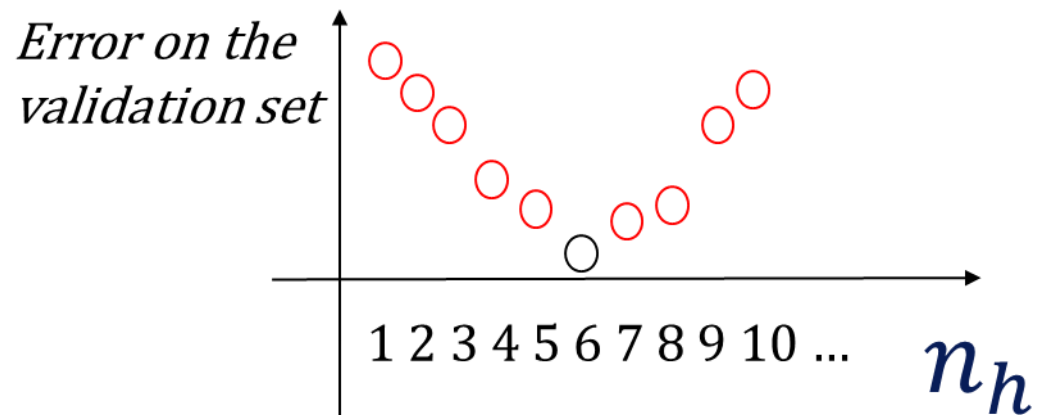
$x_1(2)$	$x_1(1)$	$x_2(2)$	$x_2(1)$	...	$x_{10}(2)$	$x_{10}(1)$	<i>class</i>
$\vdots$							$\vdots$
$x_1(t)$	$x_1(t-1)$	$x_2(t)$	$x_2(t-1)$	...	$x_{10}(t)$	$x_{10}(t-1)$	<i>class</i>
$x_1(t+1)$	$x_1(t)$	$x_2(t+1)$	$x_2(t)$	...	$x_{10}(t+1)$	$x_{10}(t)$	<i>class</i>
$\vdots$							$\vdots$
$x_1(36)$	$x_1(35)$	$x_2(36)$	$x_2(35)$	...	$x_{10}(36)$	$x_{10}(35)$	<i>class</i>

## ❑ 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



- $N_h, \eta, \alpha \rightarrow$  grid optimization
- Optimal values
  - $N_h : 17$
  - $\eta : 0.55$
  - $\alpha : 0.6$

