POLITECNICO DI MILANO

Soft Computing Methods Applied to Condition Monitoring and Fault Diagnosis for Maintenance:

Neural networks, genetic algorithms, fuzzy logic

Prof. Enrico Zio

Nuclear Engineering Department,
Politecnico di Milano, ITALY

SSARS 2007
Sopot, Poland, July, 2007

Condition monitoring and Fault Diagnosis

Ensuring system availability and reliability

1. ***Economic factors*** → service provision + maintenance and repair costs

2. ***Safety reasons*** → failures: prevention from injuries to workers, population and environment

Condition Monitoring

FAULT DIAGNOSIS → ***Fault Detection***

→ ***Fault Classification***

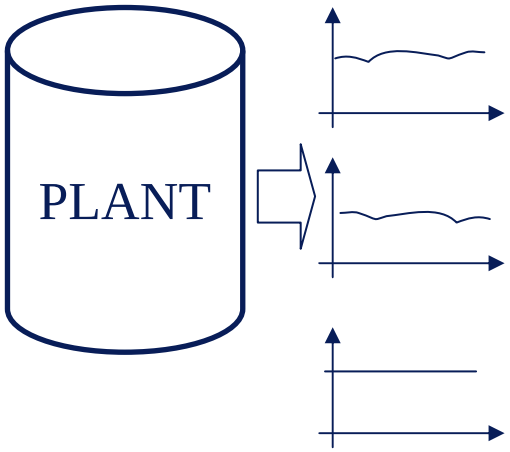
Problem statement:

What to maintain?
When to maintain?

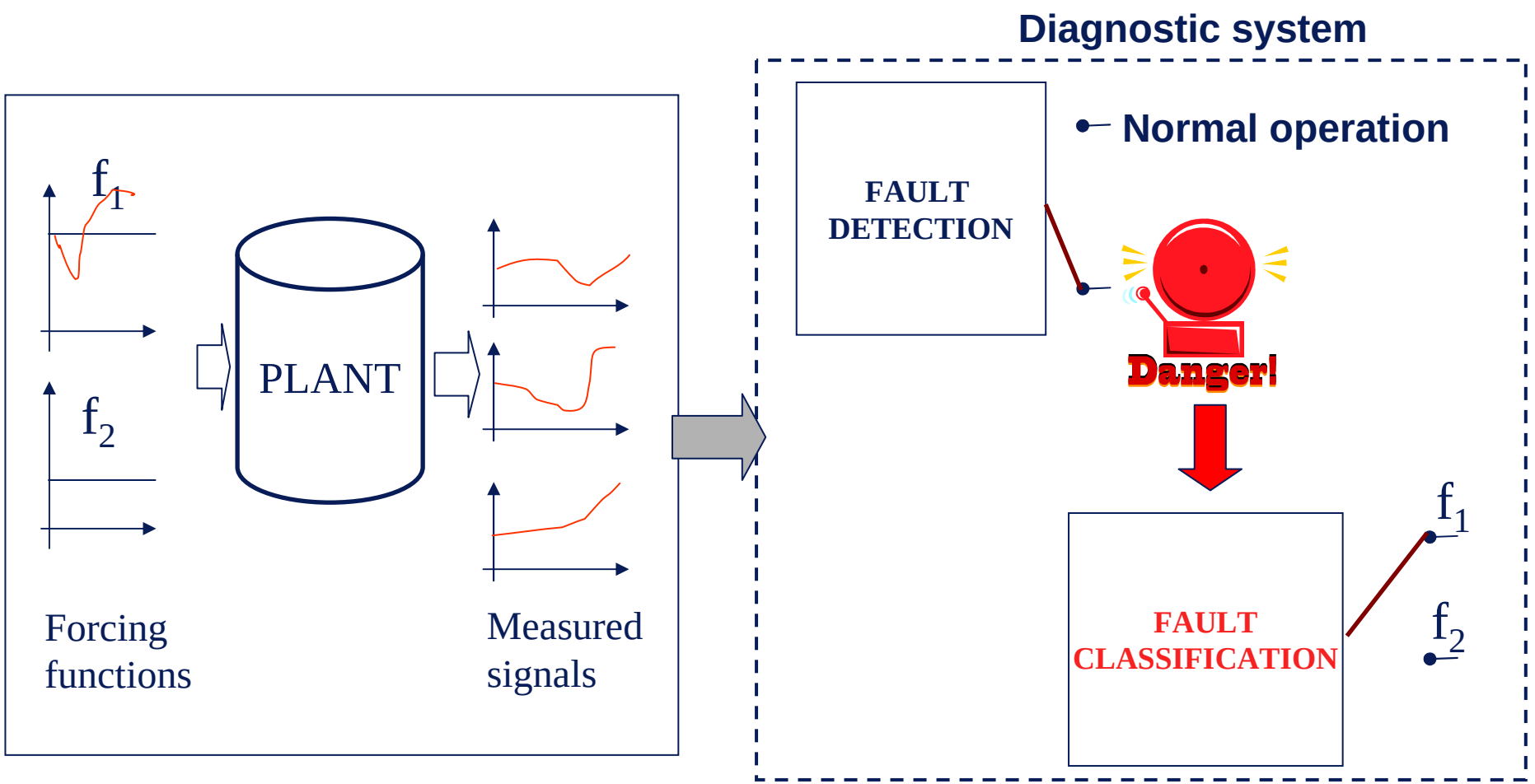
System health monitoring

Fault diagnosis

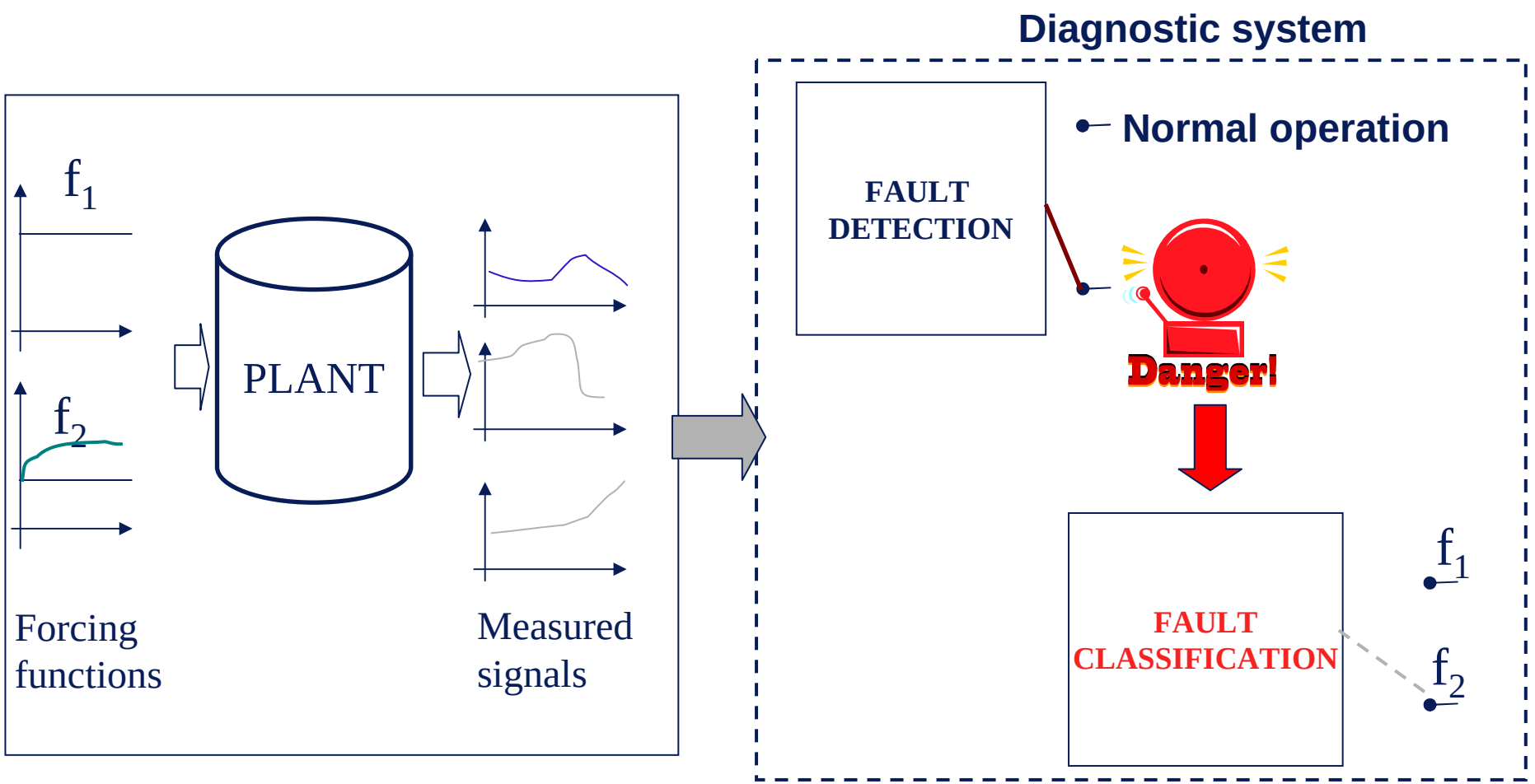
System diagnosis: fault detection and classification



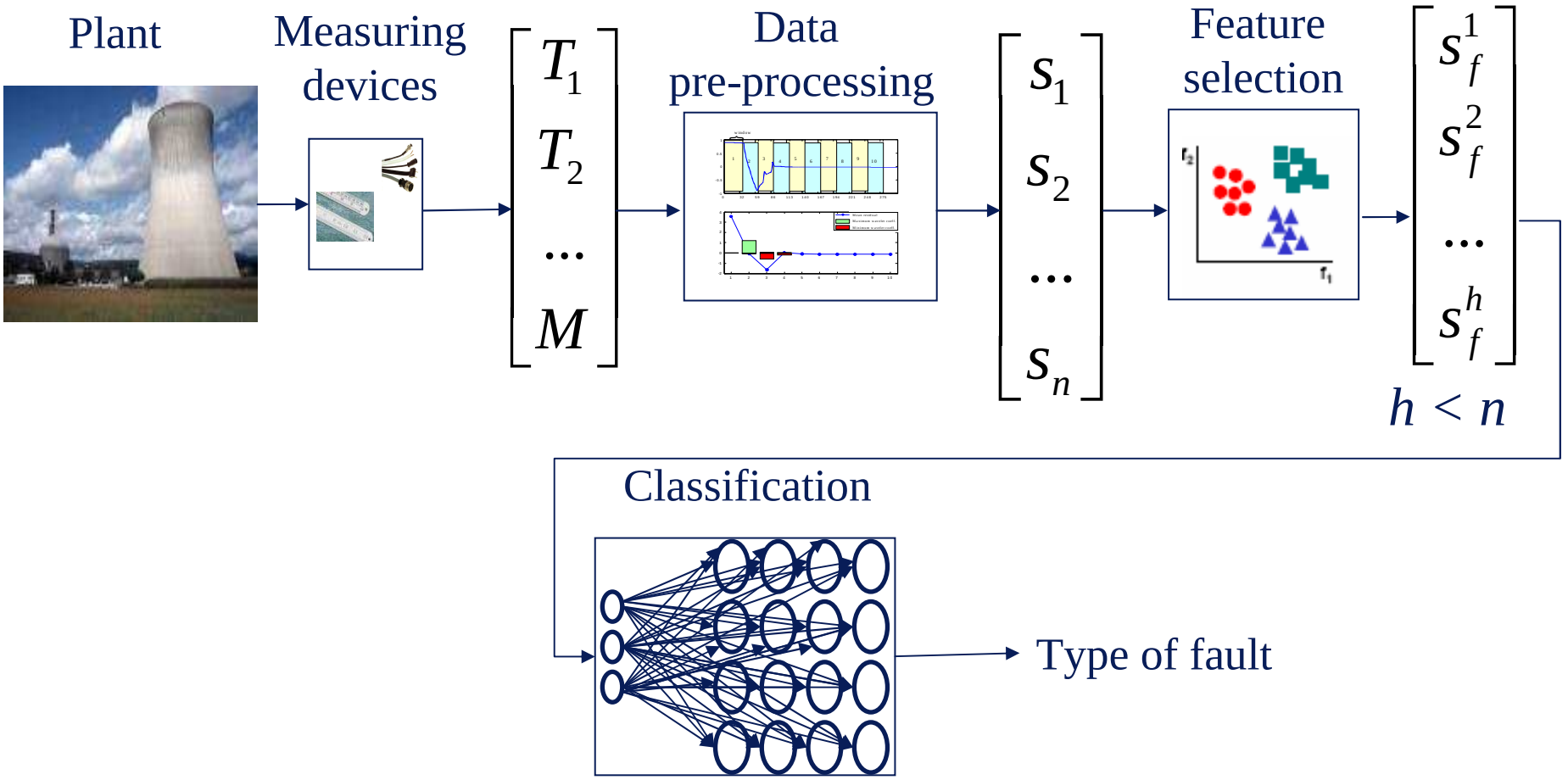
System diagnosis: fault detection and classification



System diagnosis: fault detection and classification



Diagnostic system: elements



Problem statement: practical difficulty

- Complex plants
- Many monitored signals with non-linear interrelationships

↓

Faults difficult to detect and classify

↓

No analytical but empirical models

↓

Soft computing techniques:

Neural Networks

Genetic Algorithms

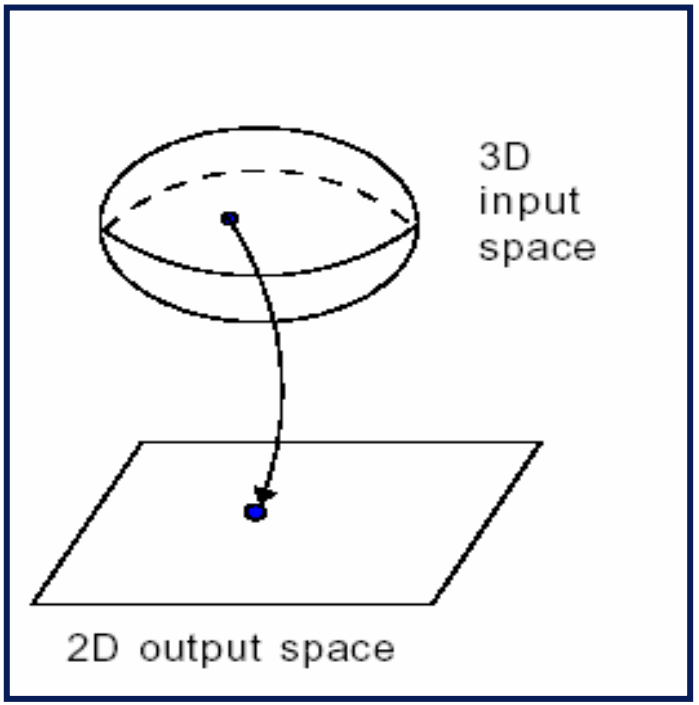
Fuzzy Logic

Artificial Neural Networks

What Are (Artificial) Neural Networks?

Multivariate, non linear interpolators

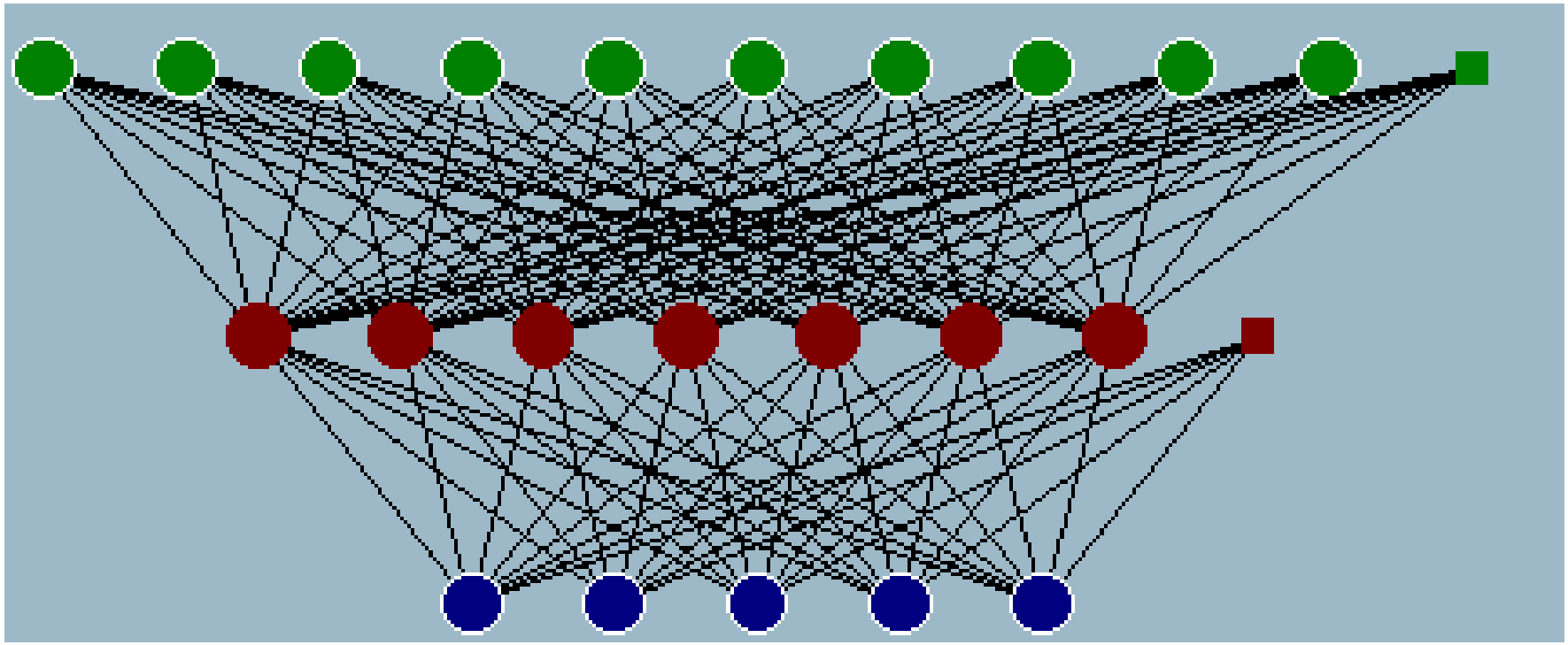
Techniques capable of reconstructing the underlying complex I/O nonlinear relations by combining multiple simple functions (modeling & computational advantage)



Empirical model
built by training
on input/output data
(modeling advantage)

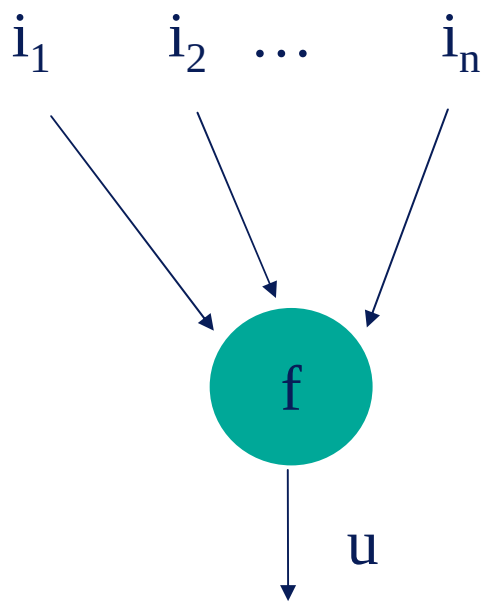
What Are (Artificial) Neural Networks?

In an artificial neural network, variables are associated with nodes in a graph.



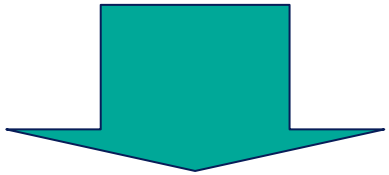
What Are (Artificial) Neural Networks?

The output (u) of a node is the result of nonlinear transformations (f) on the input variables (i) transmitted along the links of the graph.



What Are (Artificial) Neural Networks?

The variables and their interrelationships can be interpreted probabilistically.



NEURAL NETWORK = PROBABILISTIC MODEL

What Do Artificial Neural Networks Do?

- **Probability density estimation**

$$(x) \sim p(x)$$

What Artificial Do Neural Networks Do?

- **Regression**

$$(x, y), \quad y = \mu_y(x) + \varepsilon(x)$$

deterministic stochastic

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

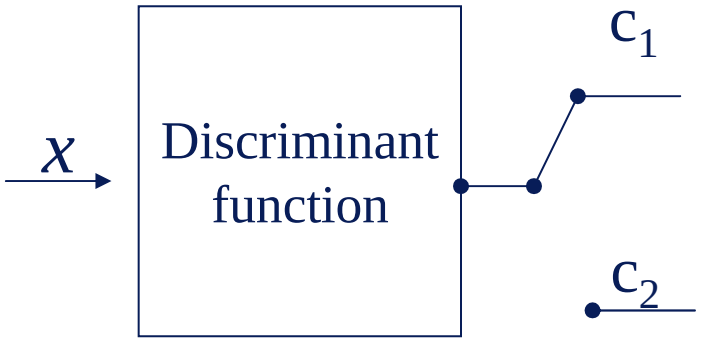
$$\sim \mu_y \qquad \sim y$$

regression prediction

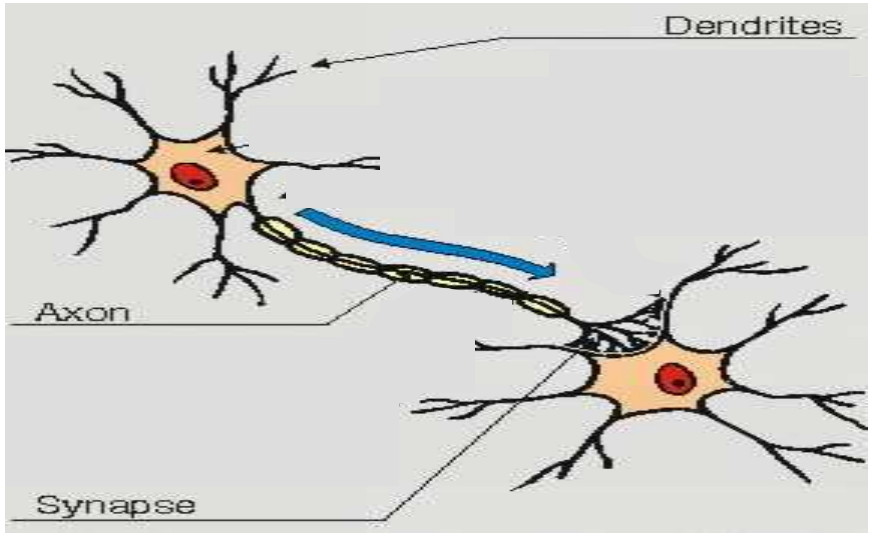
What Do Artificial Neural Networks Do?

- **Classification**

$$(x, c)$$



The Biological Neuron



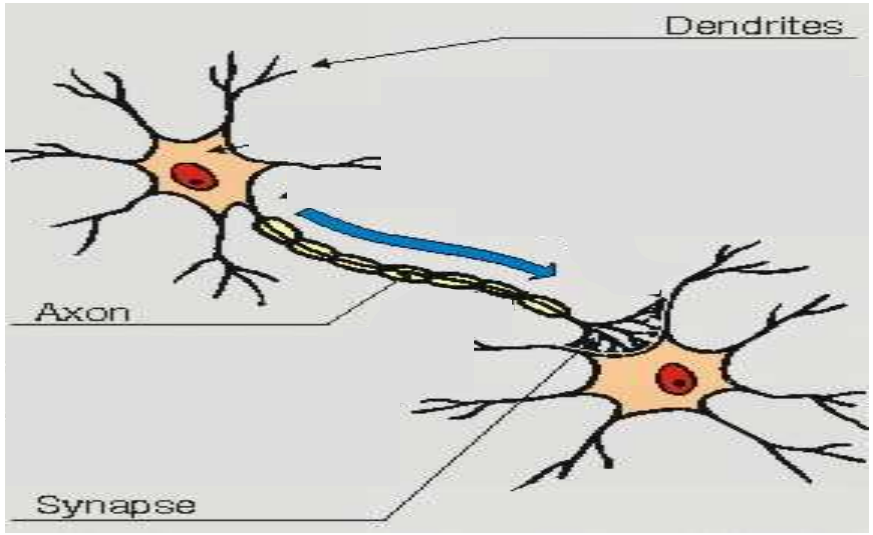
Neuron:
- cell
- axon
- dendrites
- synapses

Cell diameter: 10-80 μm

Electric pulses from other neurons are transformed into chemical information which is input to the cell body.

If the sum of the inputs received exceeds a given threshold, then it fires an electric pulse which activates the neuron function.

Biological Neural Networks



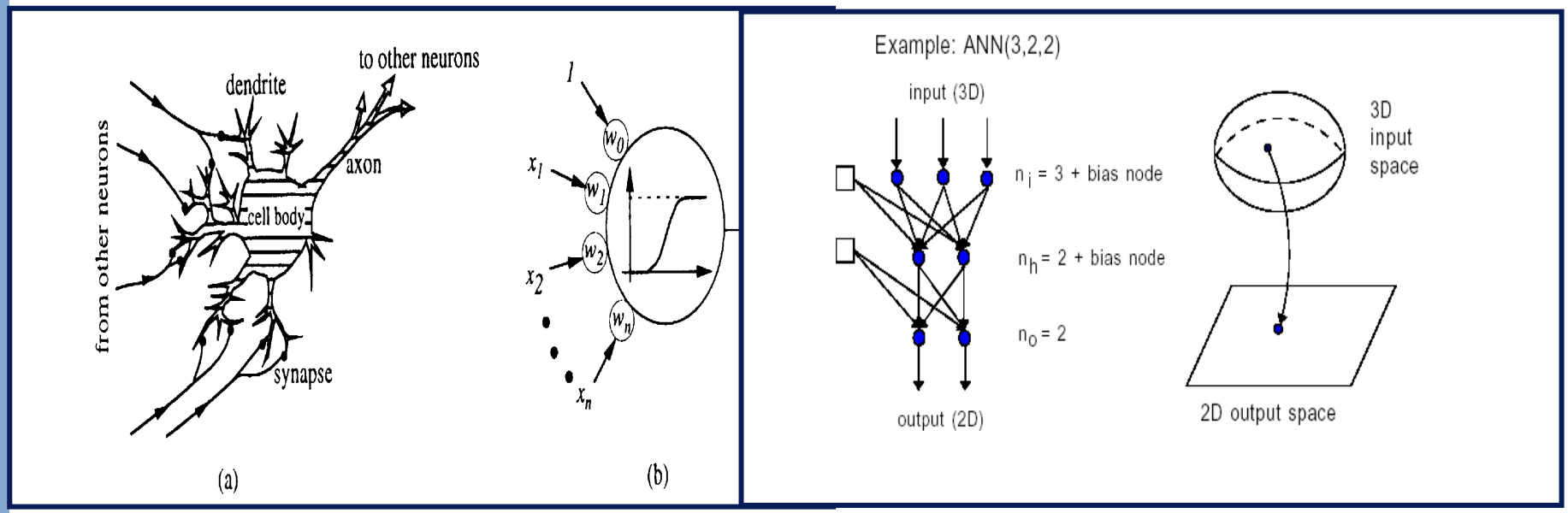
Neuron:
- cell
- axon
- dendrites
- synapses

Cell diameter: 10-80 μm

Number of neurons on the cerebral cortex (6 layers, 2mm thick)	**100 billion**
Number of synapses/neuron	**1000**
Total number of synapses	**100000 billion**
Operations/s/neuron	**100**
Total number of operations	**10000billion/s**
Neuronal density	**40000/mm^3**

How Do Artificial Neural Networks Work?

- Analogy with human brain $\Rightarrow$ mapping obtained by combining a large number of simple sigmoidal, radial or other parameterized functions which are adjusted by means of appropriate parameters and *synaptic* weights



Neural networks are universal approximators of multivariate non-linear functions.

KOLMOGOROV (1957):

For any real function $f(x_1, x_2, ..., x_n)$ continuous in $[0,1]^n$, $n \geq 2$, there exist $n(2n+1)$ functions $\psi_{ml}(\xi)$ continuous in $[0,1]$ such that

$$f(x_1, x_2, ..., x_n) = \sum_{l=1}^{2n+1} \phi_l \left(\sum_{m=1}^{n} \psi_{ml}(x_m) \right)$$

where the $2n+1$ functions ϕ_l's are real and continuous.

Thus, a total of $n(2n+1)$ functions $\psi_{ml}(\xi)$ and $2n+1$ functions ϕ_l of one variable represent a function of n variables

Neural networks are universal approximators of multivariate non-linear functions.

CYBENKO (1989):

Let $\sigma(\bullet)$ be a sigmoidal continuous function. The linear combinations

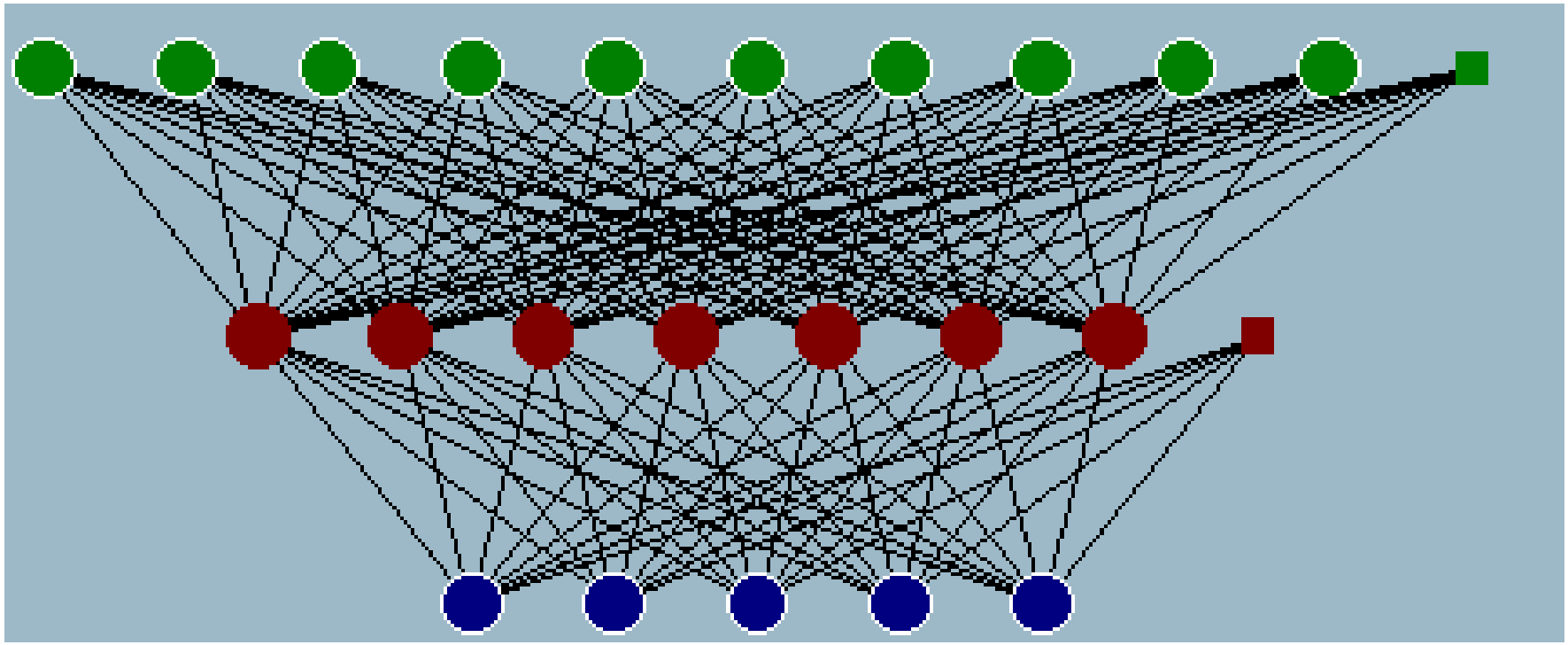
$$\sum_{j=0}^{N} \alpha_j \sigma\left(\sum_{i=1}^{n} x_i w_{ij} + \vartheta_j \right)$$

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

In other words, any function f: $[0,1]^n \rightarrow \Re$ can be approximated by a linear combination of sigmoidal functions.
Note that N is not specified.

ARTIFICIAL NEURAL NETWORKS

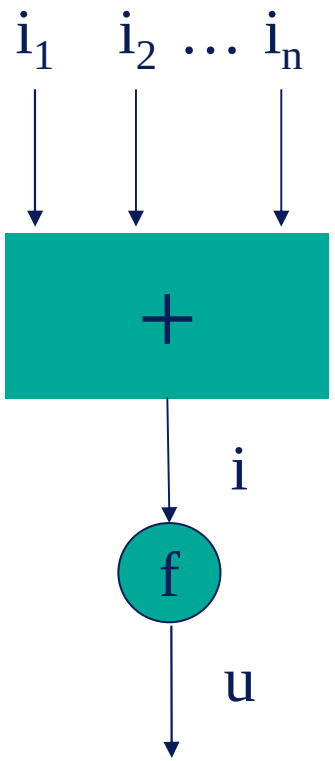
INPUT



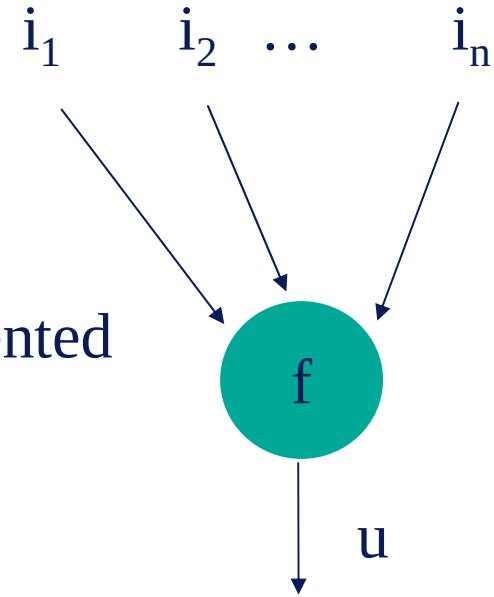
Number of connections
$11 \times 7 + 8 \times 5 = 117$

OUTPUT

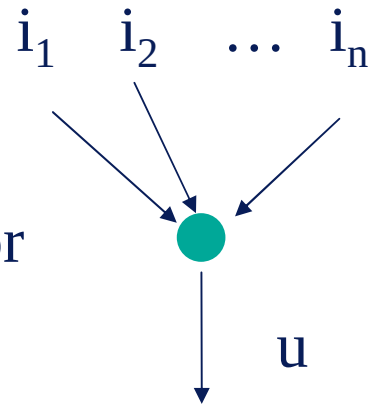
The Artificial Neuron:



represented as



or

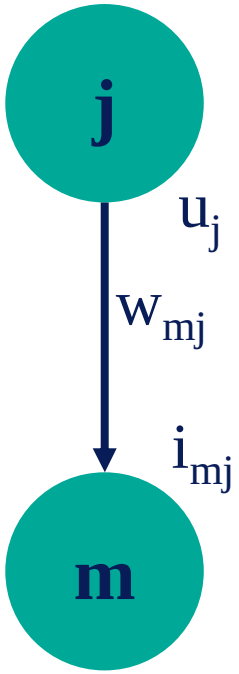


where $u = f(i)$; linear case: $u = i$

sigmoidal case: $u = 1/(1+e^{-i})$

The Artificial Synapsis

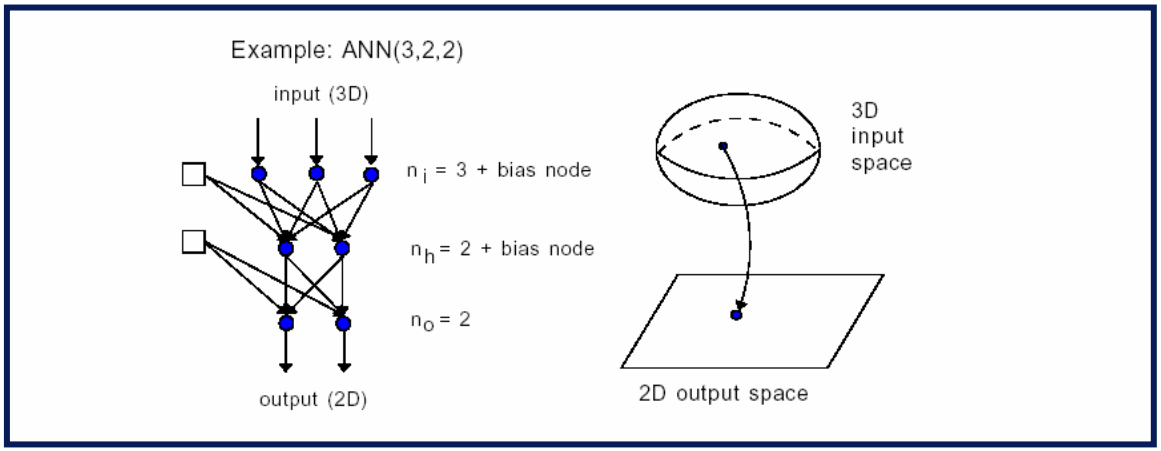
Connection between nodes i and j



$$i_{mj} = w_{mj} u_j$$

Forward Calculation (input-hidden)

Multilayered Feedforward NN



INPUT LAYER:

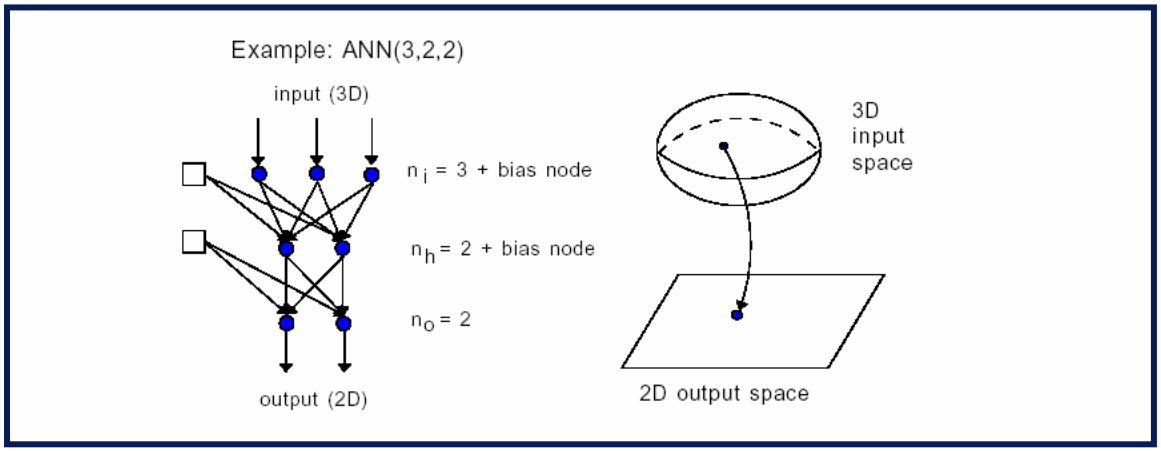
each k-th node ($k=1, 2, \ldots, n_i$) receives the (normalized) value of the k-th component of the input vector $\vec{x}$ and delivers the same value

HIDDEN LAYER:

each j-th node ($j=1, 2, \ldots, n_h$) receives $\sum_{k=1}^{n_i} x_k w_{jk} + w_{j0}$

and delivers $z_j = f\left(\sum_{k=1}^{n_i} x_k w_{jk} + w_{j0} \right)$ with f typically sigmoidal

Forward Calculation (hidden-output)



OUTPUT LAYER:

each l-th node ($l=1, 2, \ldots, n_o$) receives $\sum_{j=1}^{n_h} z_j w_{lj} + w_{l0}$

and delivers $u_l = f\left(\sum_{j=1}^{n_h} z_j w_{lj} + w_{l0} \right)$ f typically linear or sigmoidal

Setting the NN Parameters: Training Phase

Available input/output patterns

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

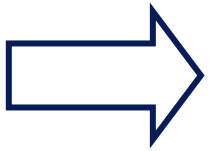
$x_1^{(2)}, x_2^{(2)} \mid t_{(2)}$

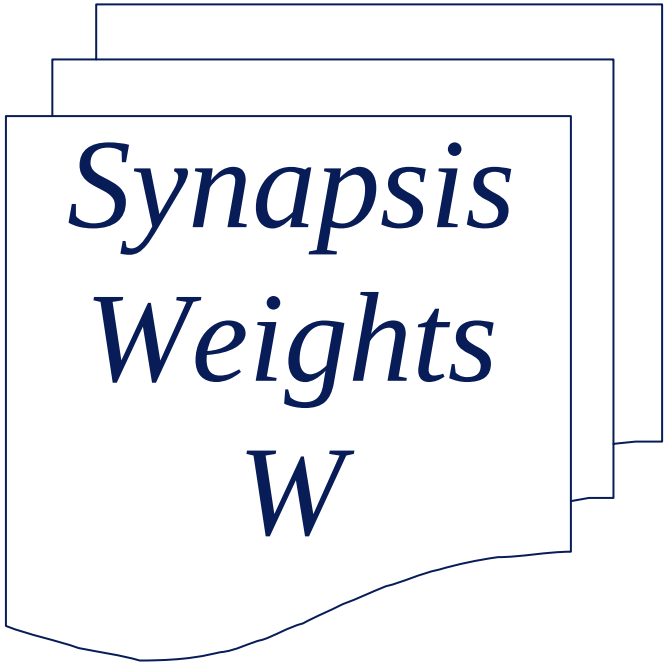
.....

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

.....

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





Setting The NN Parameters: Error Backpropagation

Initialize weights to random values

Update weights so as to minimize the *average squared output deviation error:*

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

Minimization by *gradient descent* algorithms

Error Backpropagation (output-hidden)

Updating the w_{lj} weights: $\Delta w_{lj} = -\eta \frac{\partial E}{\partial w_{lj}}$:

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

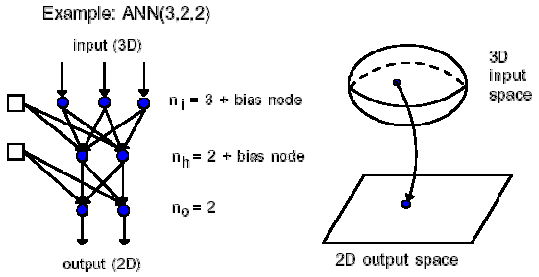
$$= \phi_l^+ (u_l^o - t_l) f'(y_l^o) u_j^h = -\delta_l u_j^h$$

where

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

Therefore

$$\Delta w_{lj} = \eta \, \delta_l \, u_j^h$$



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

Error Backpropagation (output-hidden)

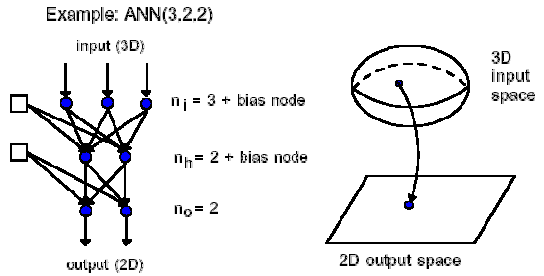
Updating the $\bar{w}_{jk}$ weights: $\Delta \bar{w}_{jk} = -\eta \frac{\partial E}{\partial \bar{w}_{jk}}$

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

$$= \sum_{l=1}^{n_o} \phi_l^+ (u_l^o - t_l) f'(y_l^o) w_{lj} f'(y_j^h) u_k^i = -\bar{\delta}_j u_i^k$$

where

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



$$\Delta w_{jk}(n) = \frac{1}{n_o} \eta \delta_j u_j^i + \alpha \Delta w_{jk}(n-1)$$

- **TRAINING SET**

Input			Output			
			ANN		Target	
x_{11}	x_{12}	x_{13}	o_{11}	o_{12}	t_{11}	t_{12}
x_{21}	x_{22}	x_{23}	o_{21}	o_{22}	t_{21}	t_{22}
x_{i1}	x_{i2}	x_{i3}	o_{i1}	o_{i2}	t_{i1}	t_{i2}

- **TRAINING PROCEDURE**

t_{i1} t_{i2} o_{i1} o_{i2} $\Rightarrow$ $E = \frac{1}{2}\left[(o_{i1} - t_{i1})^2 + (o_{i2} - t_{i2})^2\right]$ $\Rightarrow$ Backpropagation & updating

- **TEST SET**

Input			Output				
			ANN		Target		
x'_{11}	x'_{12}	x'_{13}	o'_{11}	o'_{12}	t'_{11}	t'_{12}	(for check only)
x'_{21}	x'_{22}	x'_{23}	o'_{21}	o'_{22}	t'_{21}	t'_{22}	
x'_{i1}	x'_{i2}	x'_{i3}	o'_{i1}	o'_{i2}	t'_{i1}	t'_{i2}	

- **OPERATION SET**

x''_{11} x''_{12} x''_{13}	ANN
x''_{21} x''_{22} x''_{23}	results

Utilization of the Neural Network

After training:

- **Synaptic weights fixed**
- **New input ⇨ retrieval of information in the weights ⇨ output**

Capabilities:

- **Nonlinearity of sigmoids ⇨ NN can learn nonlinear mappings**
- **Each node independent and relies only on local info (synapses)**

⇩

Parallel processing and fault-tolerance

Sensitivity Analysis

Neural Networks

The ANN sensitivity is defined as the matrix $\mathbf{S}$ whose elements are

$$S_{kl} = \frac{\partial u_l^o}{\partial x_k} = \frac{\partial u_l^o}{\partial y_l^o} \sum_{j=0}^{n_h} \frac{\partial y_l^o}{\partial u_j^h} \frac{\partial u_j^h}{\partial y_j^h} \frac{\partial y_j^h}{\partial u_k^i}$$

$$= f'(y_l^o) \sum_{j=0}^{n_h} w_{lj} \, f'(y_j^h) \, \bar{w}_{jk}$$

NEST - Neural Simulation Tool

1. Have a Matrix File

2. Choose Input and O

train.nst - Blocco note

File Modifica Formato ?

0.5951568500E-02	0.2905495700E-02	0.1169987100E-01
0.2073757100E-02	0.1123025000E-01	0.1235445400E-01
0.9200700600E-02	0.1042550100E-01	0.1448815900E-01
0.7949112000E-02	0.1389126900E-01	0.6061822200E-02
0.4224140700E-02	0.1773664200E-01	0.9058212900E-02
0.8325046900E-02	0.1209950400E-01	0.1221689400E-01
0.9361996300E-02	0.9921600100E-02	0.1536235000E-01

2.19087 1.79281 1.39475 ... -0.993621 -1.39168 -1.78974

A USER-FRIENDLY SOFTWARE

TO BUILD

ARTIFICIAL NEURAL NETWORKS

2 level parameters

Repetitions 5 Red. coeff. 0.1

Error rec 1000 Seme 12345

Examp./Batch 20 Noise % 0

5. Test the Trained ANN

LASAR

Laboratorio Analisi di Segnale e Analisi di Rischio
POLITECNICO DI MILANO
DIPARTIMENTO DI INGEGNERIA NUCLEARE

CONCLUSIONS

Advantages:

- No physical/mathematical modelling efforts.
- Automatic parameters adjustment through a training phase based on available input/output data. Adjustments such as to obtain the best interpolation of the functional relation between input and output.

Disadvantages:

"black box" : difficulties in interpreting the underlying physical model.

FINAL REMARKS

- Neuro-computing complement rule-based computing, but cannot replace it.
- Since neuro-computing is based on the establishment of input-output relationships through training on examples, rather than on the implementation of *if-then* logic, its domain of successful application is expected to be different from that of rule-based programmed computation (brain vs. computers).
- Neuro-computing is NOT a panacea for all problems (otherwise humans would never have needed to invent rule-based computers) and its full capabilities are yet to be assessed.

Application:

Detection of malfunctions in the secondary system of a nuclear power plant by neural networks

Introduction

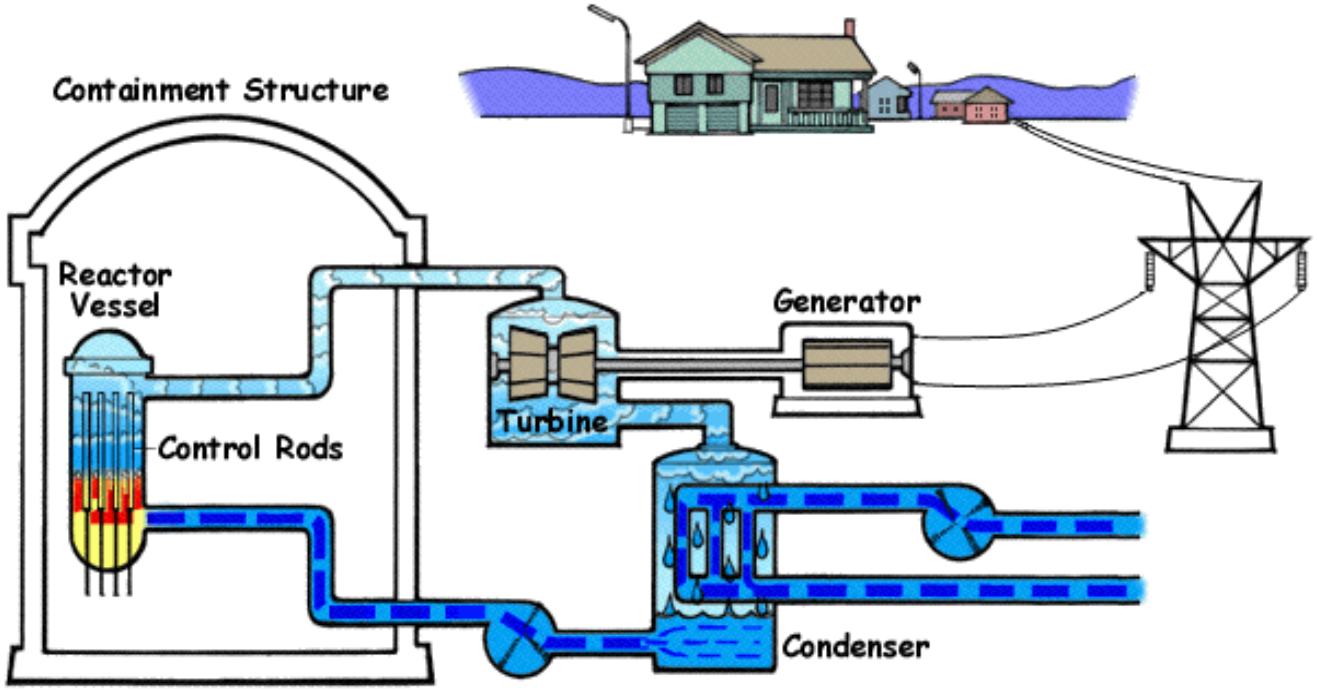
- Early identification of transients is of importance for the safe and efficient plant operation

- Neural networks can capture the non-linear behavior and build the relationships existing between the transient causes and the corresponding process variables evolution

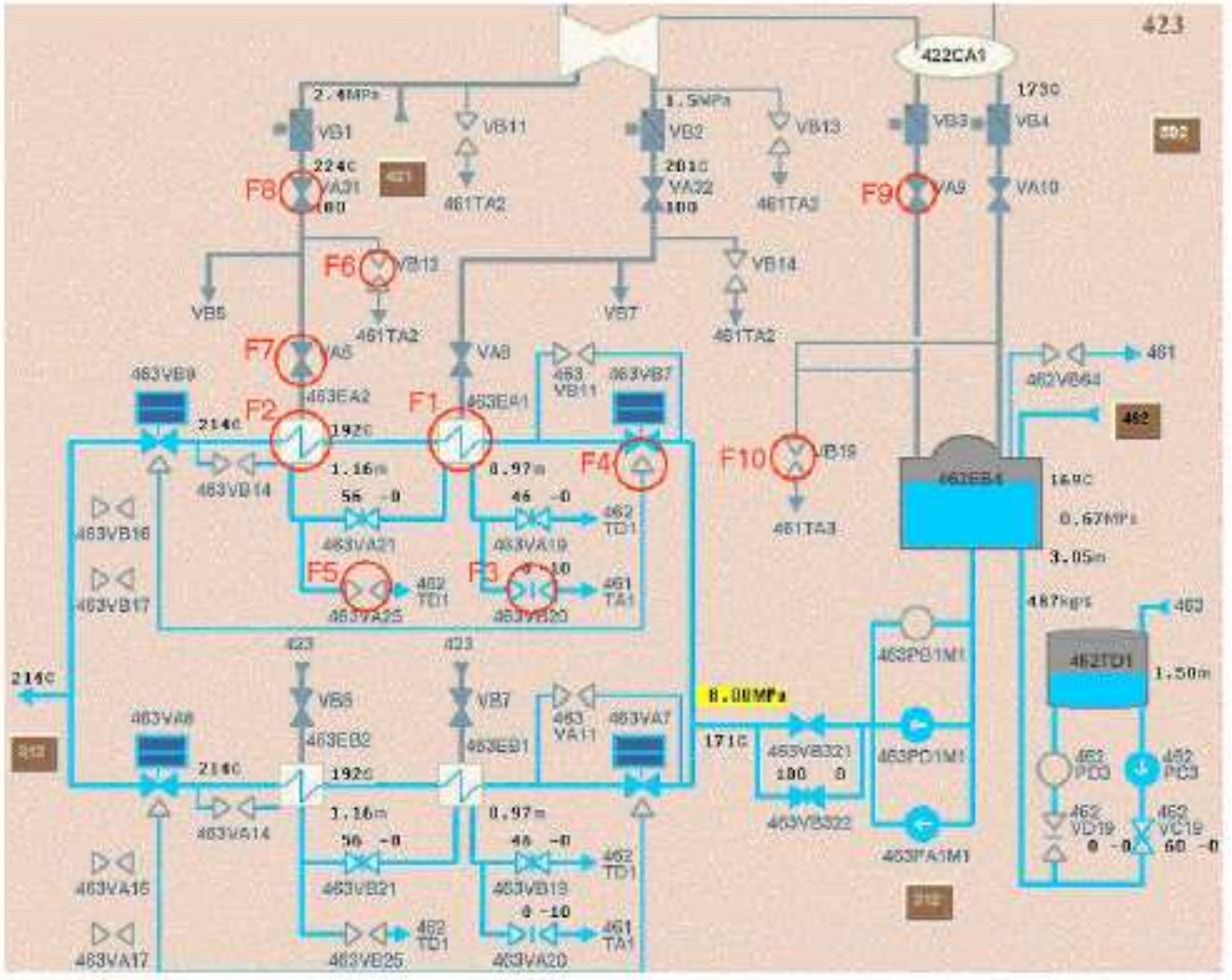
Objective

- **Build and train a neural network to classify different malfunctions in the secondary system of a boiling water reactor**

Boiling Water Reactor



Secondary System



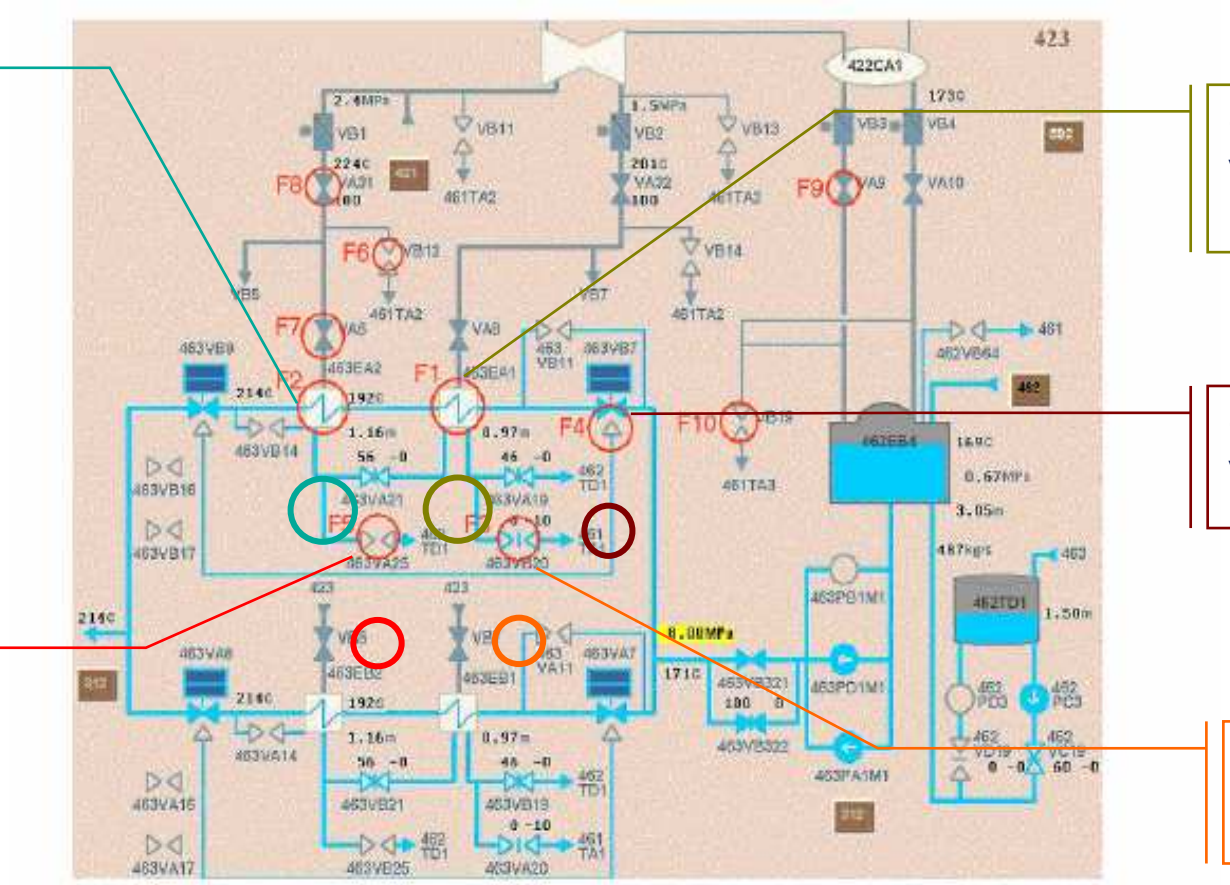
Outputs

- 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

Transients

Class 2:
Valve EA2

Class 5:
Valve VA25



Class 1:
Valve EA1

Class 4:
Valve VB7

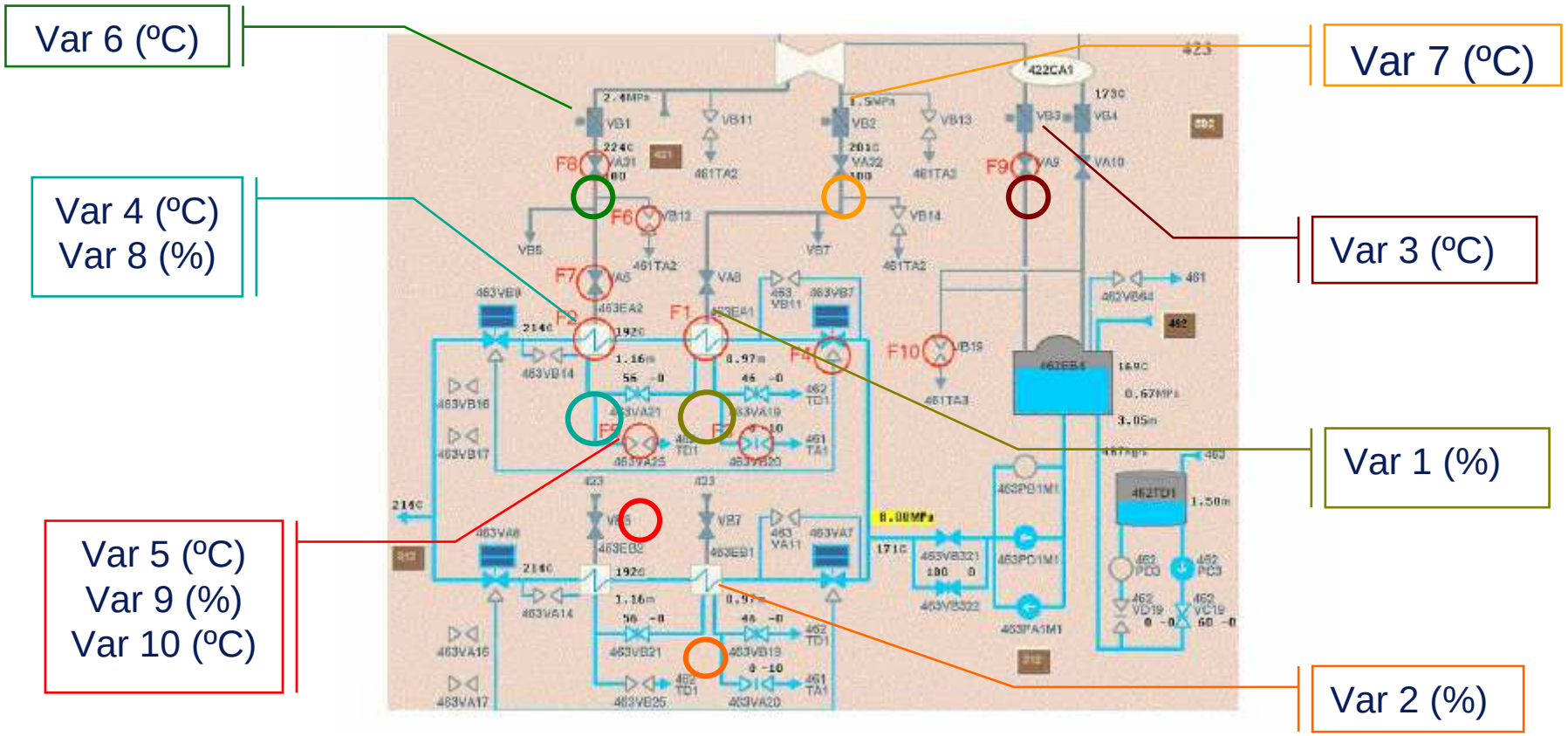
Class 3:
Valve VB20

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

Sensors position

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



Input/Output patterns

- **Input: The class assignment is performed dynamically as a two-step time window of the measured signals shifts to the successive time t+1**

- **Output: number of the class to which the variables belong**

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

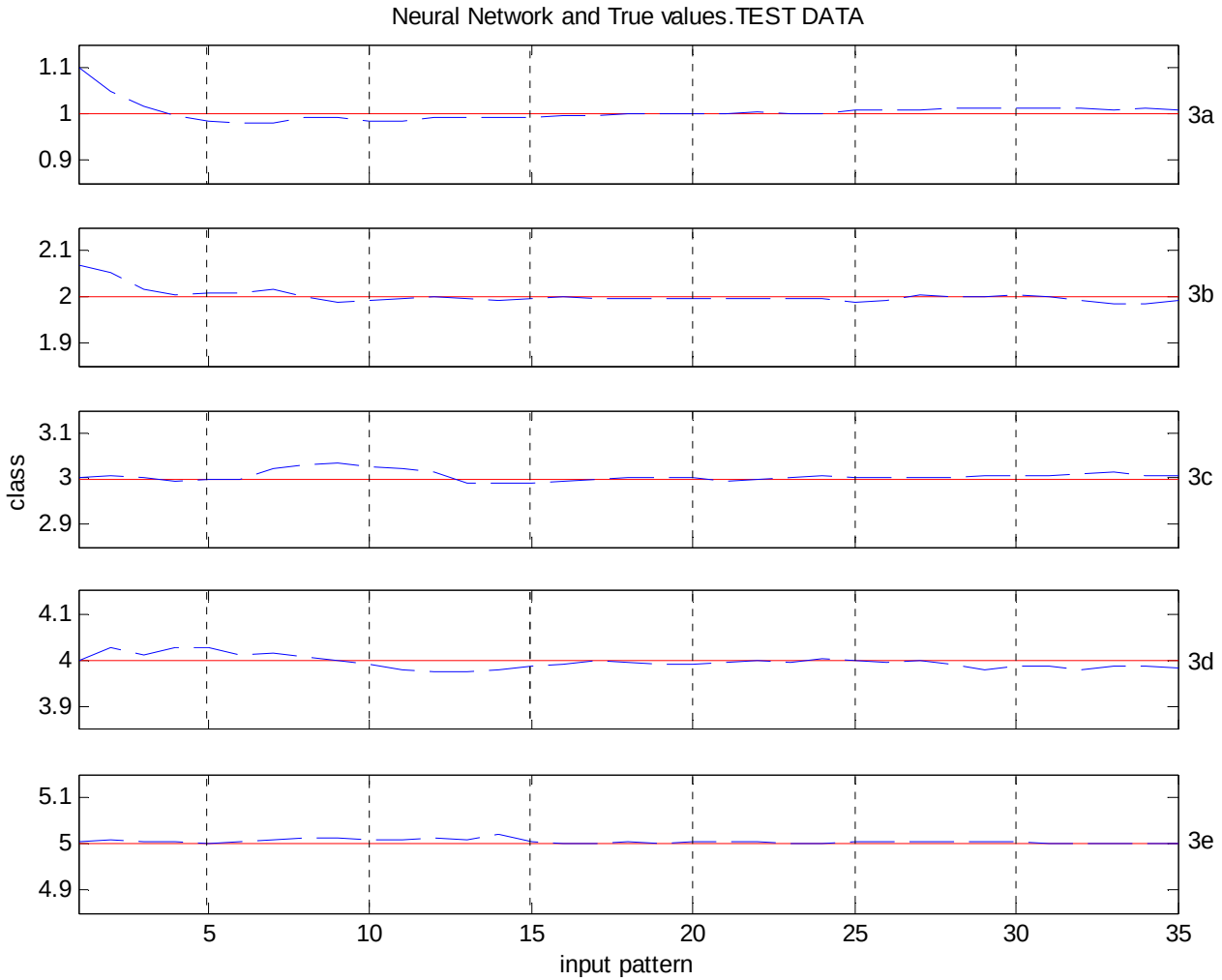
$x_1(2)$	$x_1(1)$	$x_2(2)$	$x_2(1)$	…	$x_{10}(2)$	$x_{10}(1)$	$class$
⋮							⋮
$x_1(t)$	$x_1(t-1)$	$x_2(t)$	$x_2(t-1)$	…	$x_{10}(t)$	$x_{10}(t-1)$	$class$
$x_1(t+1)$	$x_1(t)$	$x_2(t+1)$	$x_2(t)$	…	$x_{10}(t+1)$	$x_{10}(t)$	$class$
⋮							⋮
$x_1(36)$	$x_1(35)$	$x_2(36)$	$x_2(35)$	…	$x_{10}(36)$	$x_{10}(35)$	$class$

Network architecure

- **N_h, η, α optimized with grid optimization**

- **Optima values**
  - N_h : 17
  - η: 0.55
  - α : 0.6

Test results



Comments

- The real-valued outputs provided by the neural network closely approximate the class integer values since early times.
- As time progresses, the network output class assignment approaches more and more closely the true class integer.

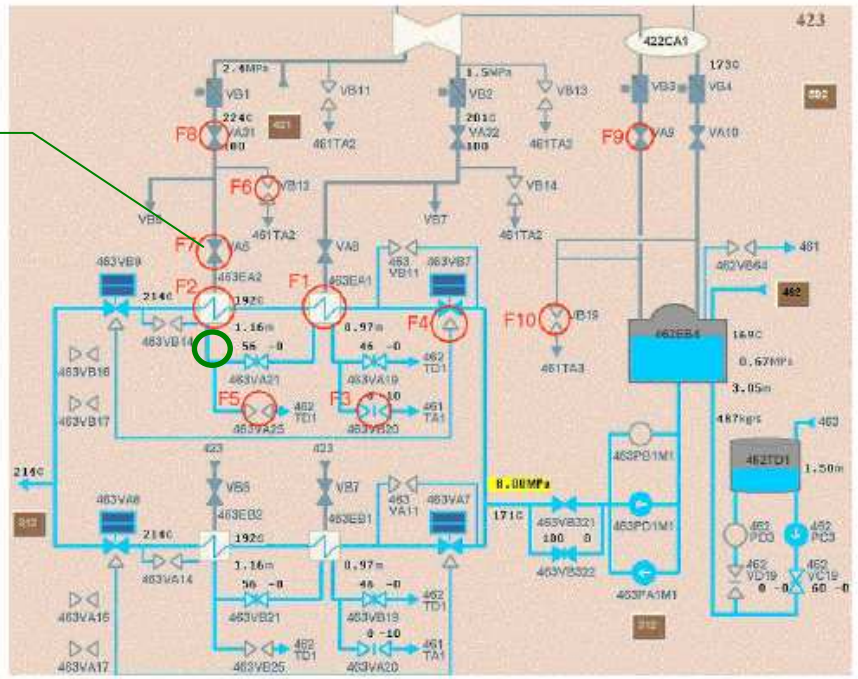
New transients

- **Response of the neural network performance desirable when a new transient is found**
  - Identification that something happens
  - Classification as don’t know

New transients

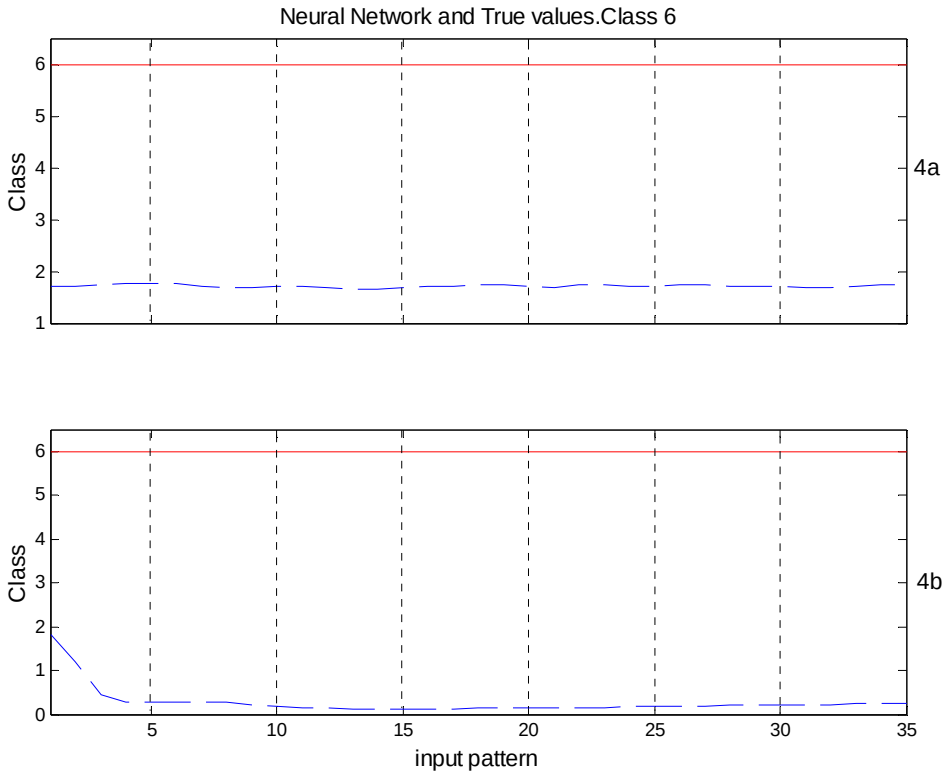
- **New class**
  - **Class 6: Steam line valve to the second high-pressure preheater closing**

Class 6:
Valve VA6



New transients

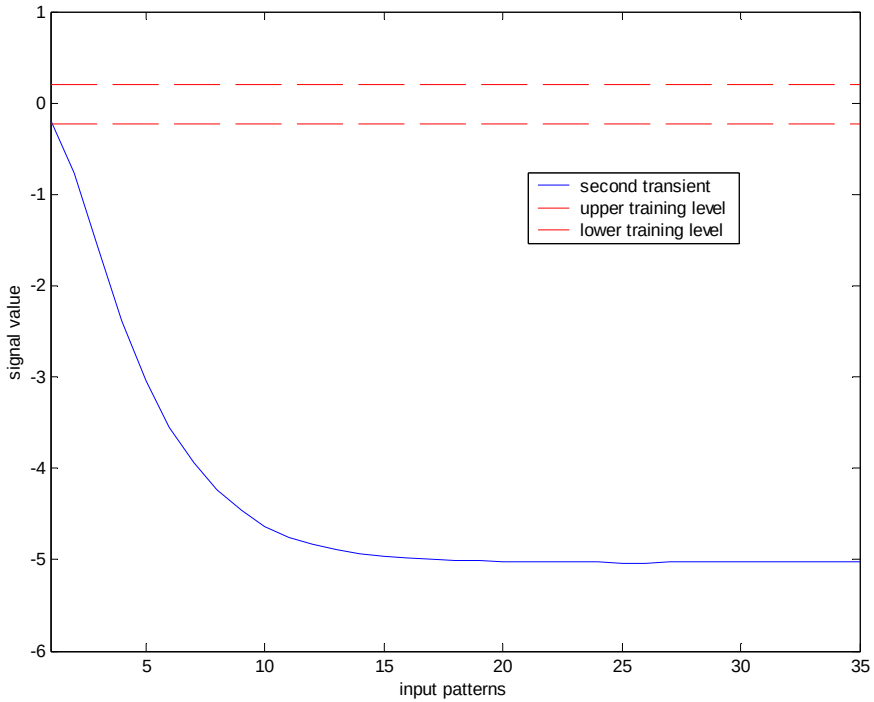
- Close to Class 2 but with a larger deviation
- Value provided by the neural network close to 0



New transient

- **the large output error is due to the fact that some of the inputs which are fed into the network fall outside the ranges of the input values of the training transients**

Temperature drain after VB2



Conclusions

- A neural network has been constructed and trained to classify different malfunctions in the secondary system of a boiling water reactor.
- The network uses as input patterns the information provided by signals that are monitored in the plant and provides as output an estimate of the class assignment.
- When a transient belonging to a different class is fed to the network, it is capable of detecting that the pattern does not belong to any of the classes for which it was trained