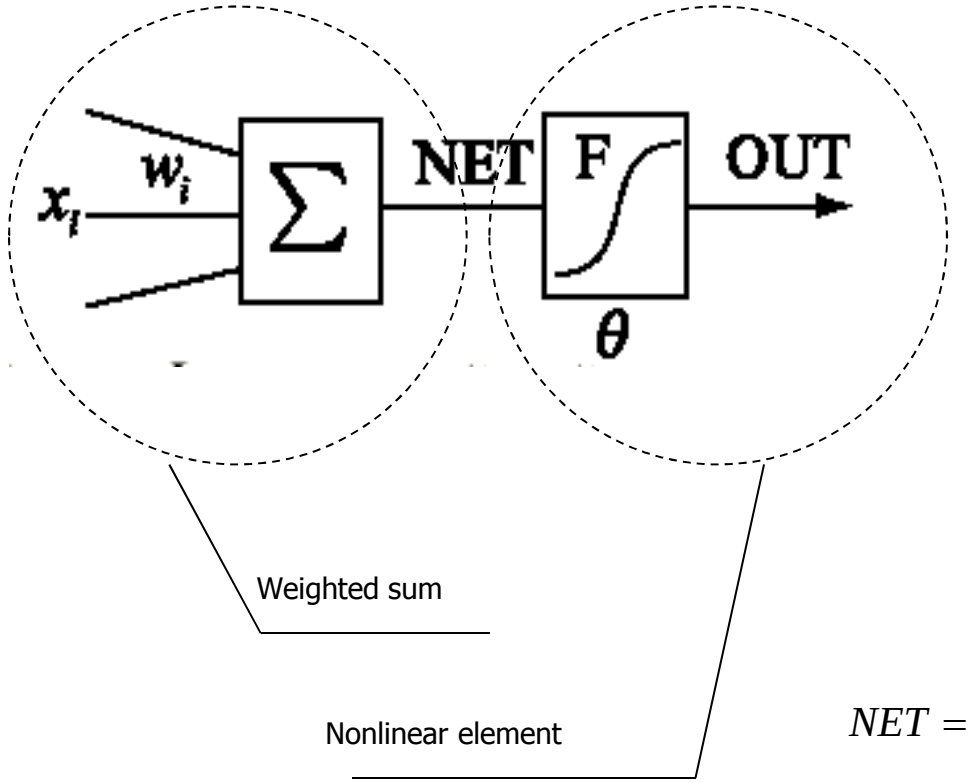
1918
TALLINNA
TEHNIKAÜLIKOOL

Identification with artificial neural networks

Eduard Petlenkov,

Artificial neuron



Input vector: $X = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$

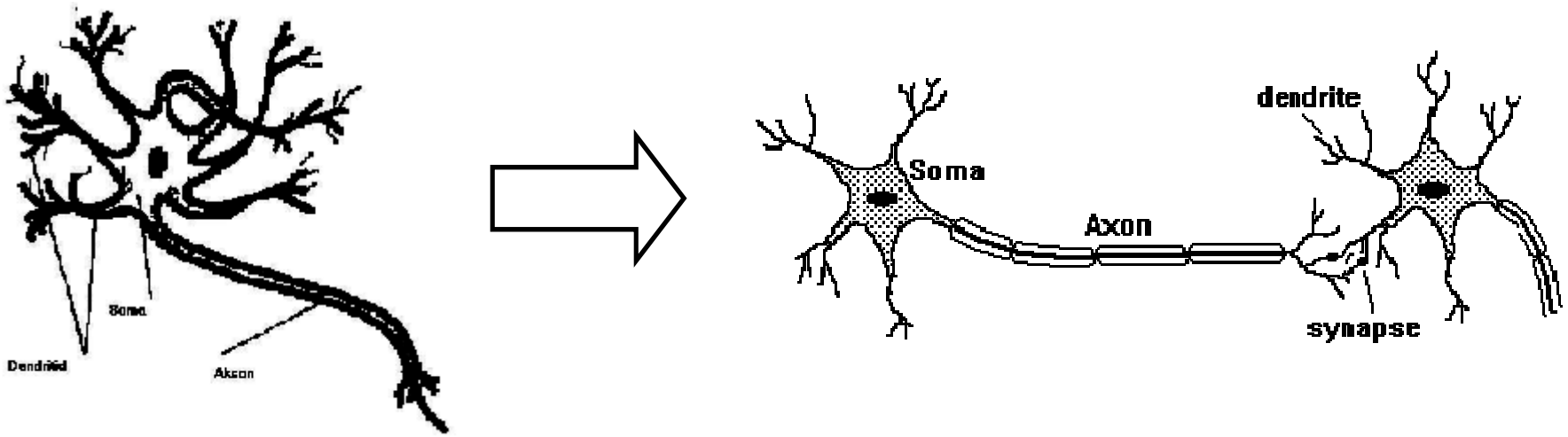
Vector of weight coeffitients:

$$W = [w_1 \dots w_n]$$

Weighted sum:

$$NET = W \cdot X = [w_1 \dots w_n] \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = w_1 x_1 + \dots + w_n x_n$$

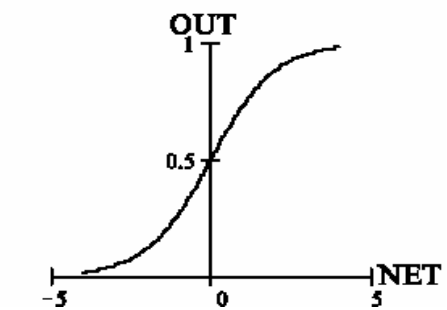
Biological neuron and biological neural networks



Activation functions (1)

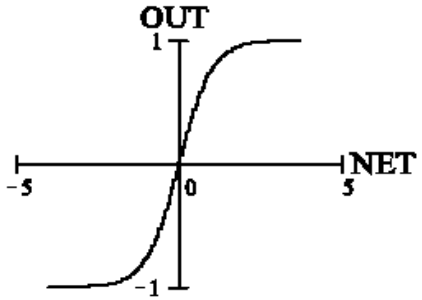
OUT=f(NET)

Sigmoid functions are having an "S" shape (**sigmoid curve**)



1 Logistic function

$$OUT = \frac{1}{1+e^{-NET}}$$



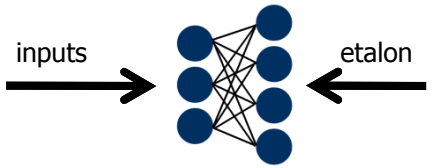
2 Hyperbolic tangent

$$OUT = \frac{e^{NET} - e^{-NET}}{e^{NET} + e^{-NET}}$$

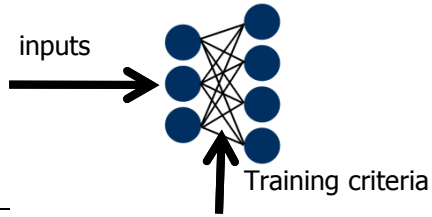
Types of artificial neural networks

Structure:
- Feedforward
- Feedback
  - internal feedback
  - external feedback

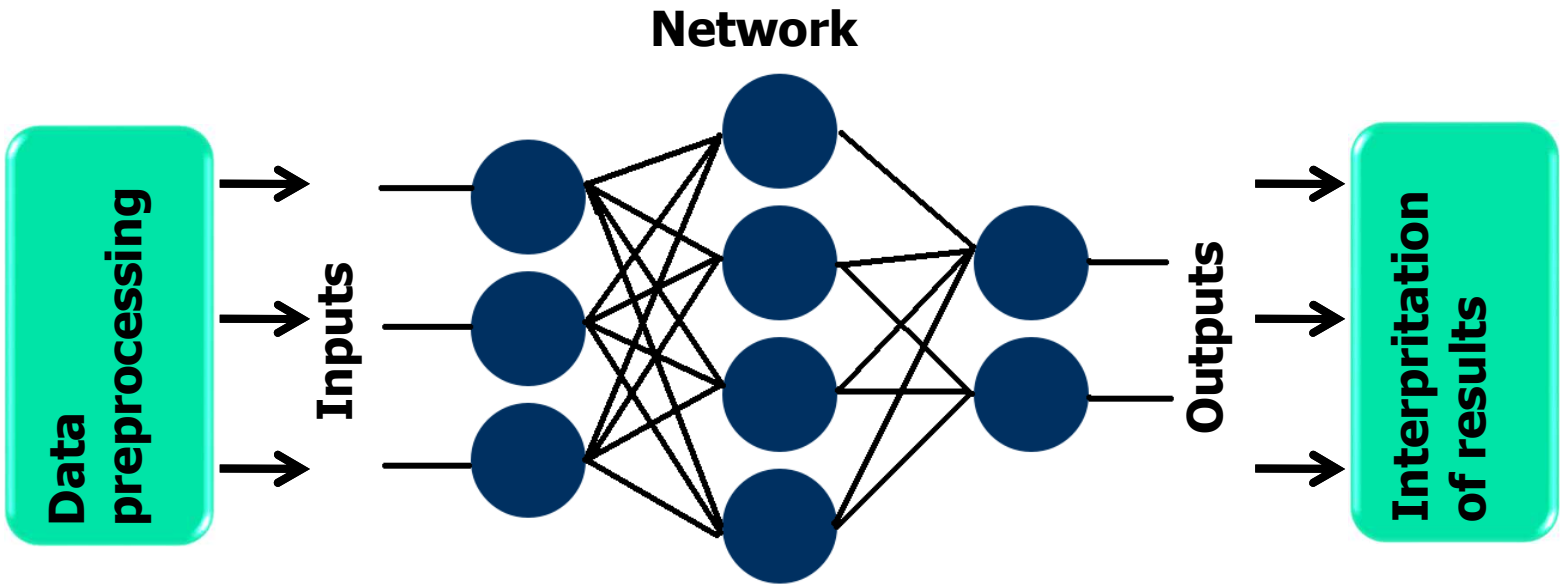
Training/ learning:
- Supervised learning



- Selflearning (unsupervised)

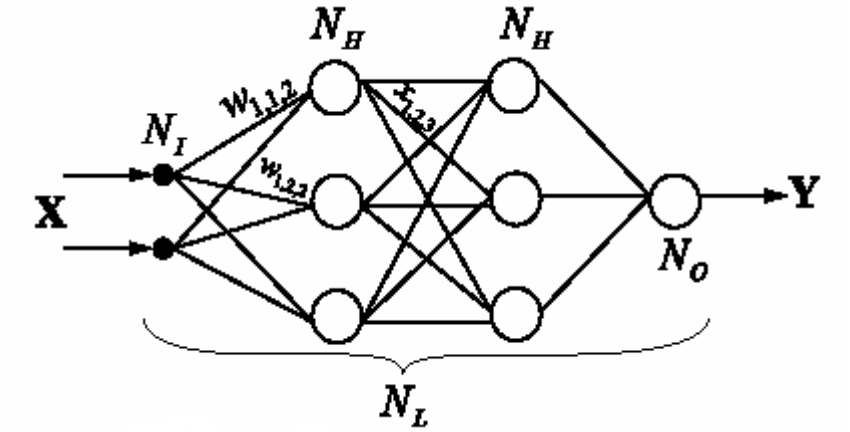


How to use artificial neural networks



Feedforward neural networks and multilayer perceptron

Feedforward networks are networks in which an output of a neuron can be connected only with an input of a next layer neuron.



"from each to each"

N_I - Input layer

N_O - Output layer

N_H - Hidden layer

w_{ijl} - Weighting coefficients, where

i is the number of the neuron's input

j - neuron's number in the layer

l - number of the layer

Mathematical function of a two-layer perceptron

$$X = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}; \quad Y = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}; \quad W_1 = \begin{bmatrix} w_{111} & \cdots & w_{n11} \\ \vdots & \ddots & \vdots \\ w_{1k1} & \cdots & w_{nk1} \end{bmatrix}; \quad \Theta_1 = \begin{bmatrix} \theta_{11} \\ \vdots \\ \theta_{k1} \end{bmatrix}; \quad W_2 = \begin{bmatrix} w_{112} & \cdots & w_{k12} \\ \vdots & \ddots & \vdots \\ w_{1m2} & \cdots & w_{km2} \end{bmatrix}; \quad \Theta_2 = \begin{bmatrix} \theta_{12} \\ \vdots \\ \theta_{m2} \end{bmatrix};$$

F_1 - Activation function of the hiddent layer neurons;

F_2 - Activation function of the output layer neurons.

$$Y = F_2(W_2(\underbrace{F_1(W_1 X - \Theta_1)}_{\substack{output\ of\ the \\ first\ layer}}) - \Theta_2)$$

$output\ of\ the\ network$

Supervised learning

Supervised learning is a training algorithm based on known input-output data.

$$\left|Y_p - NN(X)\right| = \left|Y_p - Y\right| \to 0$$

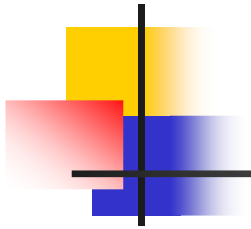
X is an input vector;

Yp is a vector of reference outputs corresponding to input vector X

Y is a vector of NN's outputs corresponding to input vector X

NN is a mathematica function of the network

(Y=NN(X))



Õpetamine (2)

Network training procedure consists of the following 3 steps:

1. Computation of NN's outputs corresponding to the current NN's paraneters;
2. Computation of NN's error;
3. Updating NN's parameters by using selected training algorithm in order to minimize the error.

Gradient descent error backpropagation (1)

Error function: $$J(W,\Theta) = \sum_k (Y_p - Y_p^d)^2$$

Y_p - NN's output

$$Y_p = NN(X_p, W, \Theta)$$

X_p - inputs used for training;

NN - NN's function;

Y_p^d - etalons (references) for outputs=desired outputs.

function to be minimized:

$$\min J(W,\Theta)$$

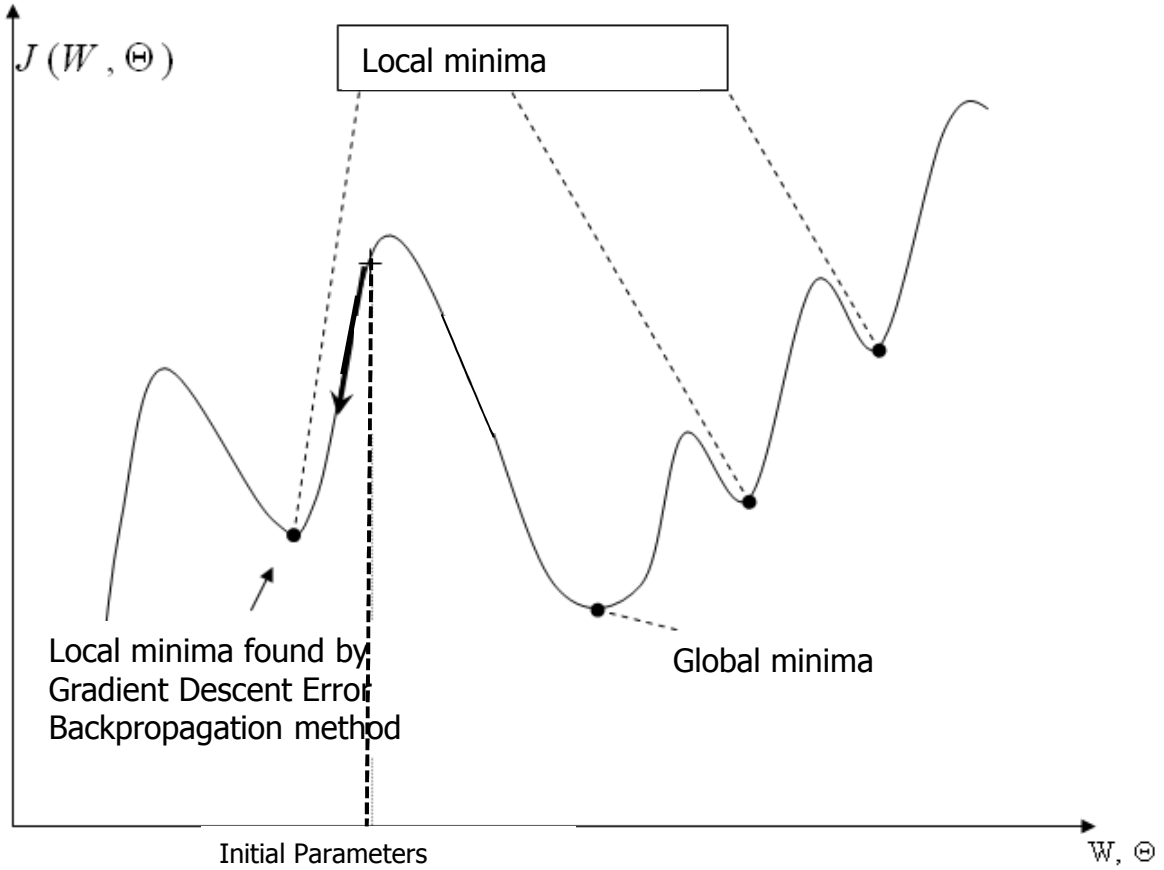
Gradient descent error backpropagation (2)

$$F(x_1,\dots x_n) \Rightarrow \nabla F(x_1,\dots x_n) = \left(\frac{\partial F}{\partial x_1} \dots \frac{\partial F}{\partial x_2} \right)$$

$$-\nabla F(x_1,\dots x_n) = \left(-\frac{\partial F}{\partial x_1} \dots -\frac{\partial F}{\partial x_2} \right)$$

$$J(W,\Theta) \Rightarrow -\nabla J(W,\Theta) = \left(-\frac{\partial J}{\partial W} \quad -\frac{\partial J}{\partial \Theta} \right)$$

Global minimum problem



Training of a double-layer perceptron using Gradient descent error backpropagation (1)

$$yh_j(t) = F_1(\sum_i Wh_{ji}(t)x_i(t) + Bh_j(t)) \qquad y_k(t) = F_2(\sum_i Wo_{kj}(t)yh_i(t) + Bo_k(t))$$

i - number of input;

j – number of neuron in the hidden layer;

k – number of output neuron;

$x_i(t)$ NN's input at time instance *t*;

$Wh_{ji}(t)$ - Values oh the hidden layer weighting coefficients at time instance *t*;

$Bh_j(t)$ - Values oh the hidden layer biases at time instance *t*;

F_1 - Activation function of the hidden layer neurons

$yh_j(t)$ - Outputs of the output layer neurons at time instance *t*;

$Wo_{kj}(t)$ - Values oh the output layer weighting coefficients at time instance *t*;

$Bo_k(t)$ - Values oh the output layer biases at time instance *t*;

F_2 - Activation function of the output layer neurons;

$y_k(t)$ - Outputs of the network at time instance *t*;

Training of a double-layer perceptron using Gradient descent error backpropagation (2)

error: $e(t) = y(t) - y_{etalon}(t)$

gradients:

$$\frac{\partial J}{\partial W_{ij}} = \delta_j(t) x_i(t)$$

$$\frac{\partial J}{\partial W_{kj}} = \delta_k(t) yh_j(t)$$

$\delta_j(t),\ \delta_k(t)$ Signals distributing information about error from output layer to previous layers (that is why the method is called error backpropagation):

$$\delta_k(t) = (y_k(t) - y_k^d(t)) \cdot F_2'$$

$$\delta_j(t) = F_1' \cdot \sum_k \delta_k(t) \cdot Wo_{kj}(t)$$

$F_1',\ F_2'$ Are derivatives of the corresponding layers activation functions.

Training of a double-layer perceptron using Gradient descent error backpropagation (3)

Updated weighting coefficients:

$$Wo_{kj}(t+1) = Wo_{kj}(t) - \eta \cdot \frac{\partial J}{\partial Wo_{kj}} = Wo_{kj}(t) - \eta \delta_k(t) yh_j(t)$$

$$Wh_{ji}(t+1) = Wh_{ji}(t) - \eta \cdot \frac{\partial J}{\partial Wh_{ji}} = Wh_{ji}(t) - \eta \delta_j(t) x_i(t)$$

η learning rate

$$f(x) = \frac{1}{1+e^{-x}} \Rightarrow f'(x) = \frac{e^{-x}}{\left(1+e^{-x}\right)^2} = \frac{1}{1+e^{-x}} - \frac{1}{\left(1+e^{-x}\right)^2} = f(x) - f^2(x)$$

Gradient Descent (GD) method vs Levenberg-Marquardt (LM) method

$$Z_n = \{[u(t), y(t)], t = 1, \ldots, N\}$$

$$J(\theta, Z_N) = \frac{1}{2N}\sum_{t=1}^{N}[y(t) - \hat{y}(t)]^2 = \frac{1}{2N}\sum_{t=1}^{N} e(t,\theta)^2$$

$$G(\theta) = \frac{\partial J(\theta, Z_N)}{\partial \theta} = \frac{1}{N}\sum_{t=1}^{N}\frac{\partial e(t,\theta)}{\partial \theta} e(t,\theta)$$ - Gradient

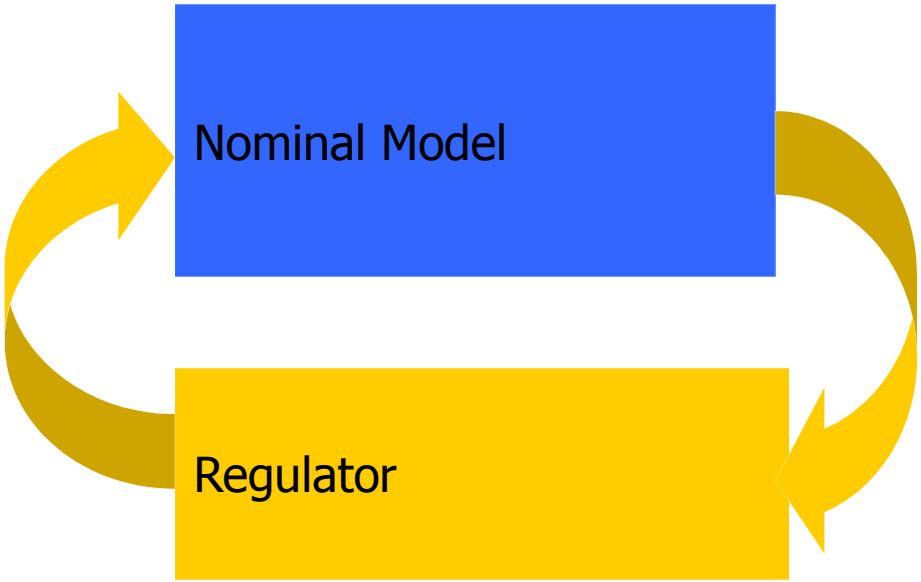
$$H(\theta) = \frac{\partial^2 J(\theta, Z_N)}{\partial \theta \partial \theta^T} = \underbrace{\frac{1}{N}\sum_{t=1}^{N}\frac{\partial e(t,\theta)}{\partial \theta}\left[\frac{\partial e(t,\theta)}{\partial \theta}\right]^T}_{R(\theta)} - \frac{1}{N}\sum_{t=1}^{N}\frac{\partial^2 e(t,\theta)}{\partial \theta \partial \theta^T} e(t,\theta)$$ - Hessian

GD: $\theta(k+1) = \theta(k) - \lambda G(\theta)$

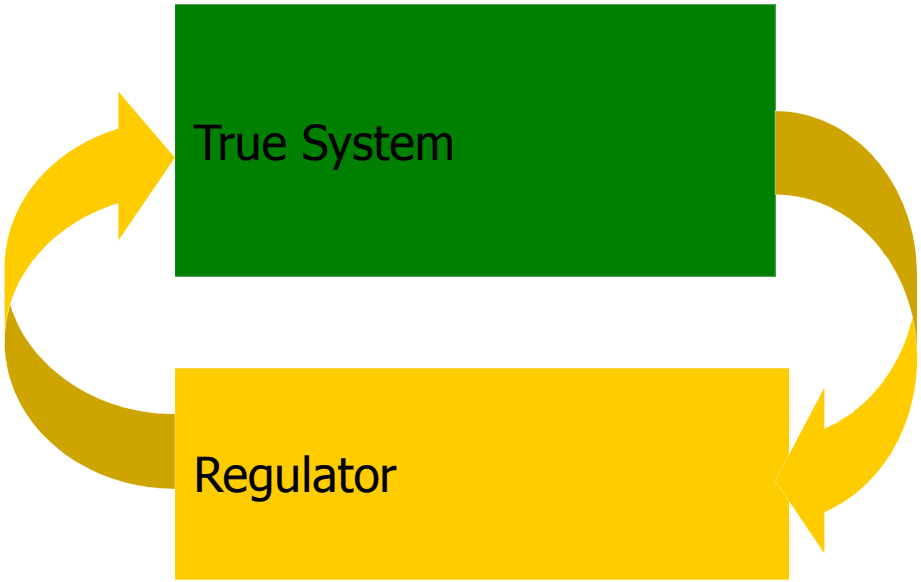
LM: $\theta(k+1) = \theta(k) + \Delta\theta(k)$

$$[R(\theta(k)) + \lambda I]\Delta\theta(k) = -G(\theta(k)) \quad \Rightarrow \Delta\theta(k)$$

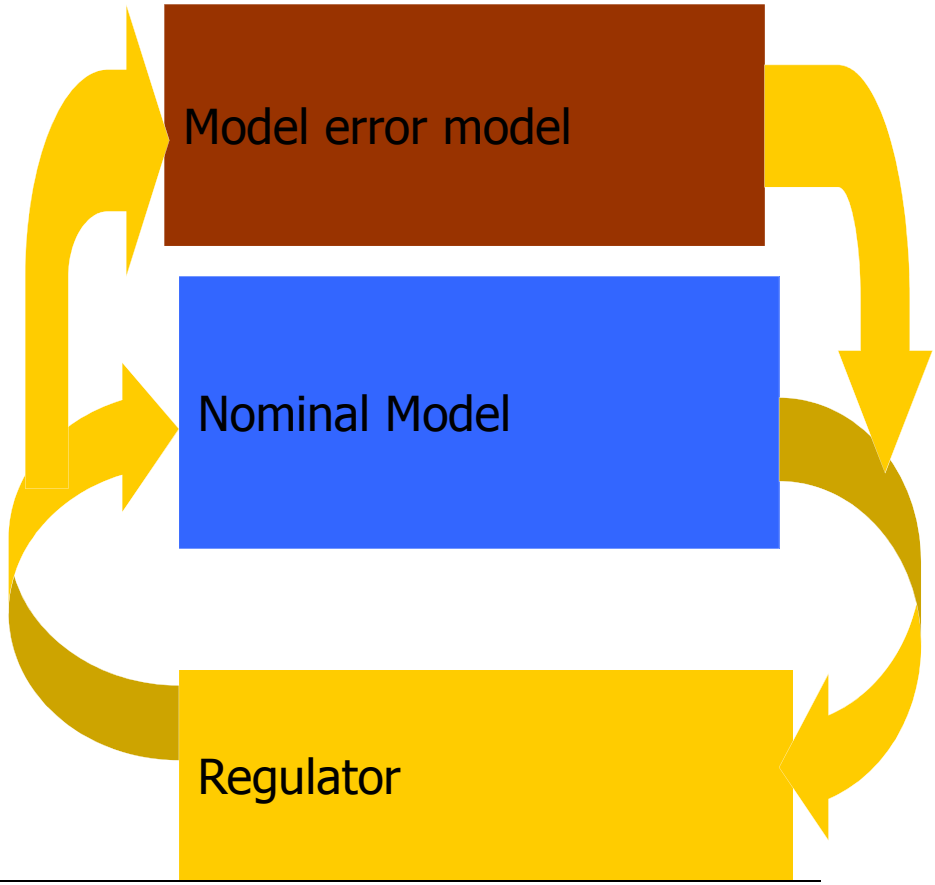
Control Design



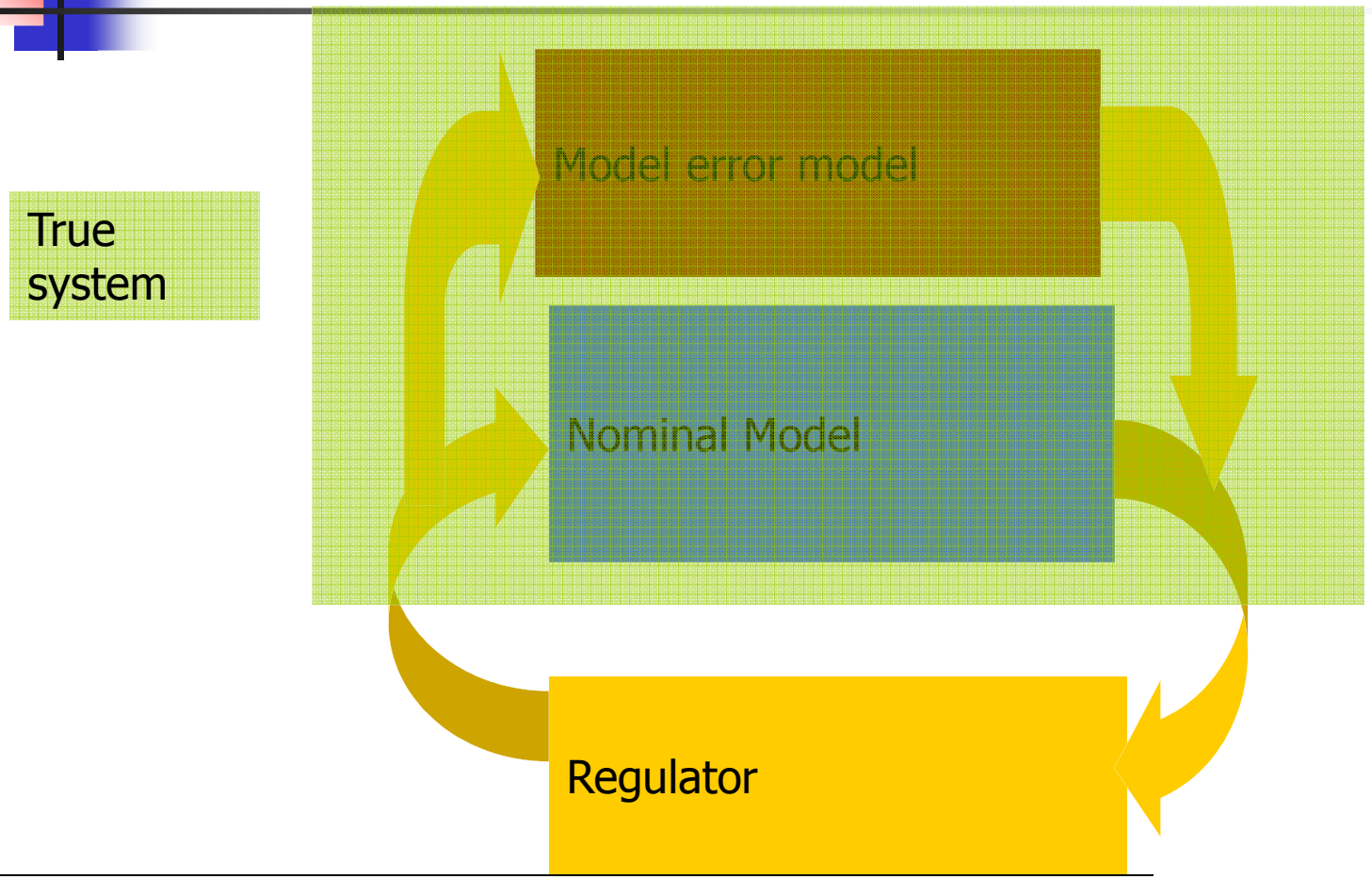
Control Design



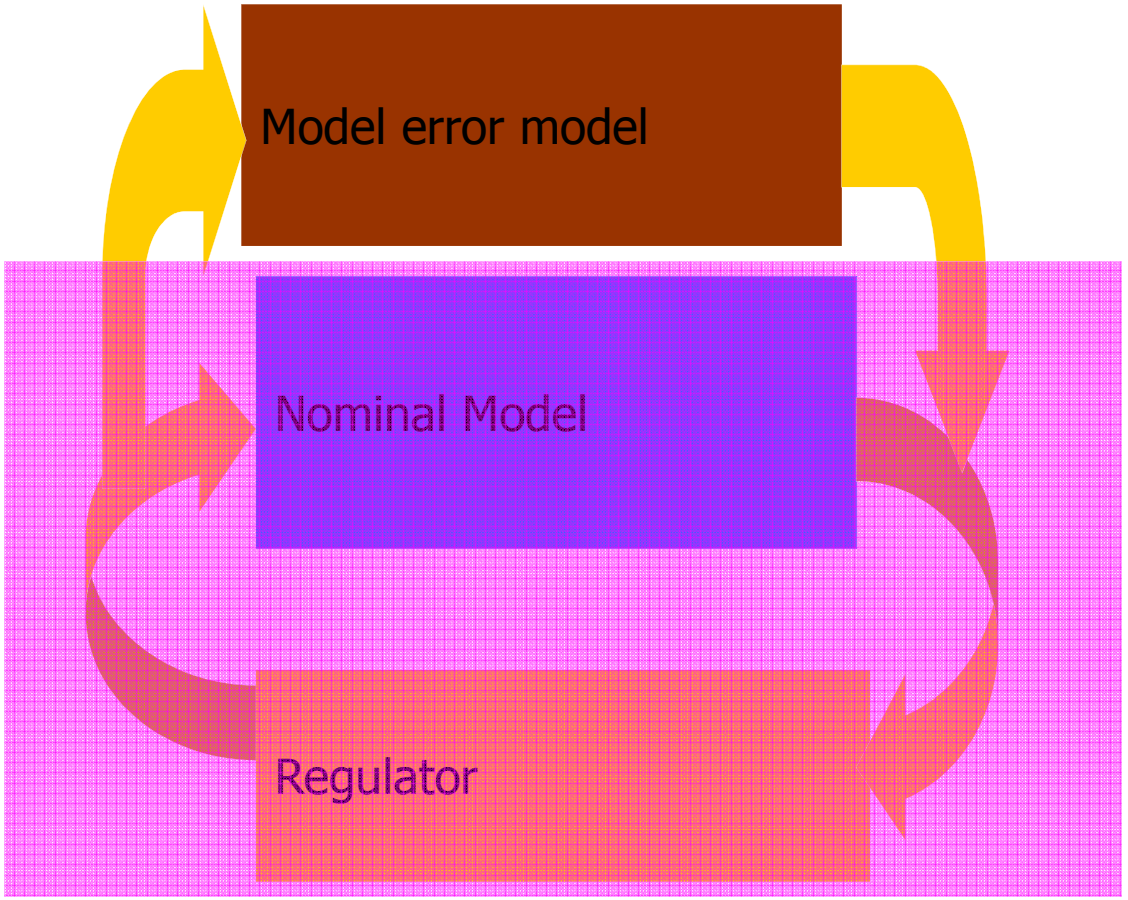
Control Design



Control Design

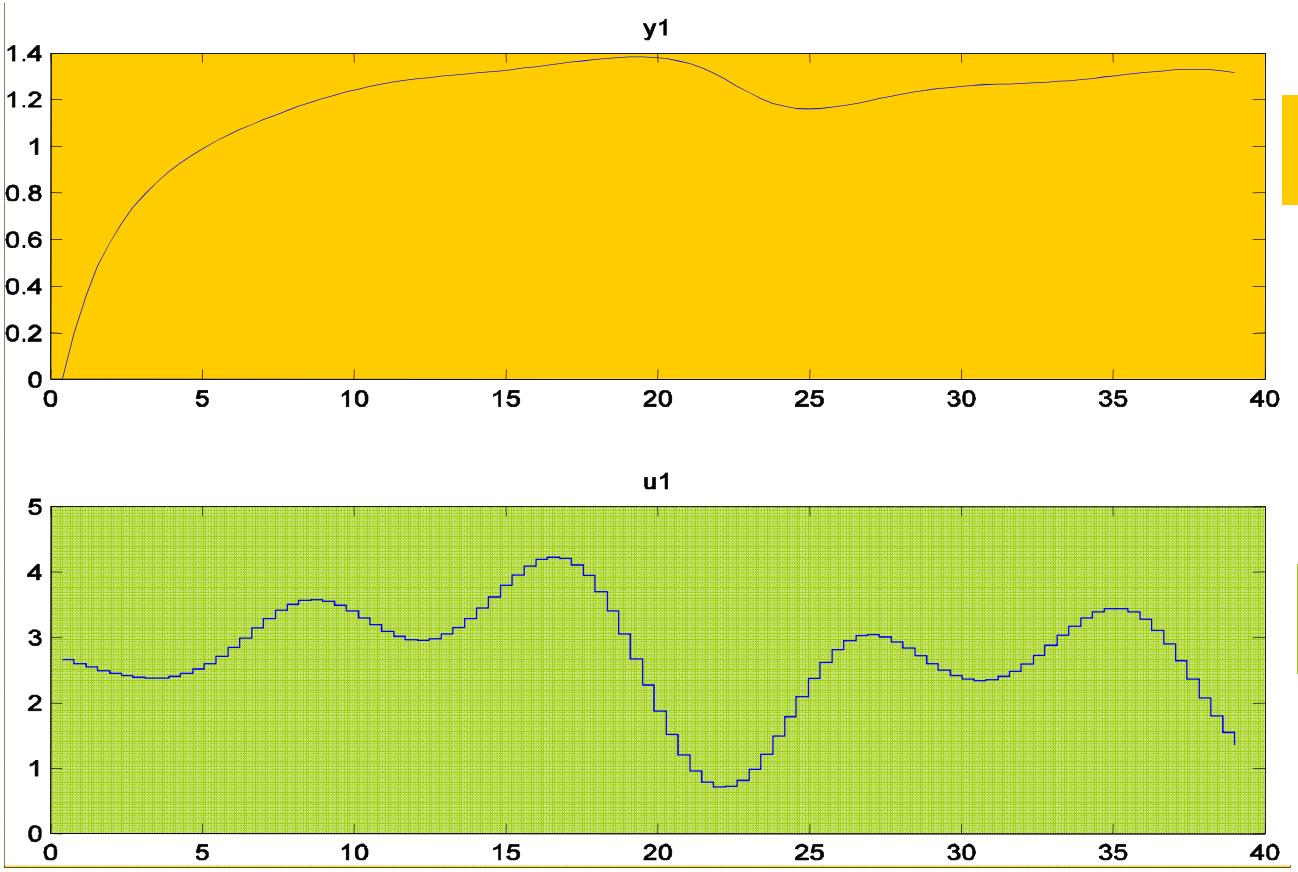


Control Design



Nominal closed loop system

A Data Set/ nonlinear process

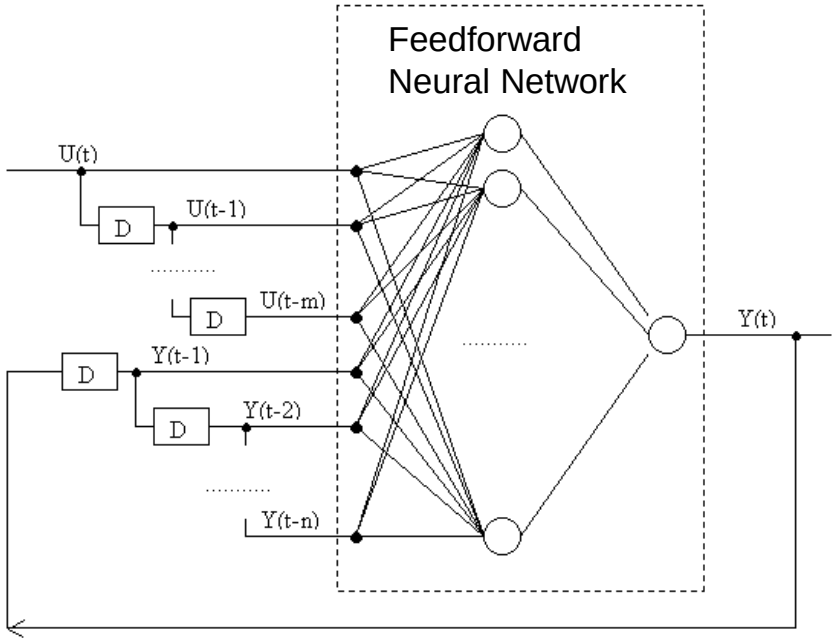


Output

Input

Feedback Neural Networks. Identification of dynamic systems (1)

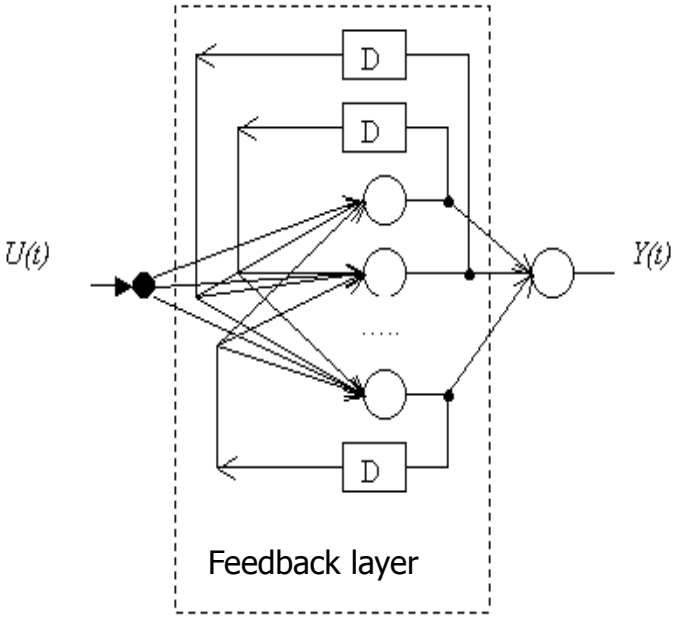
External feedback:



$$y(t) = F_2 \cdot (W_2 \cdot (F_1 \cdot (W_1 \cdot \begin{bmatrix} u(t) \\ \vdots \\ u(t-m) \\ y(t-1) \\ \vdots \\ y(t-n) \end{bmatrix} + \Theta_1) + \Theta_2) =$$

$$= f_{nn}(u(t), \ldots, u(t-m), y(t-1), \ldots, y(t-n))$$

Feedback Neural Networks. Identification of dynamic systems (2)



Elman network

Internal feedback:

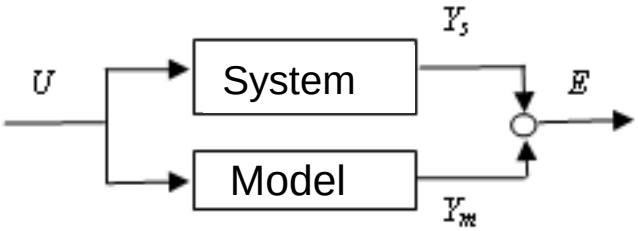
$$W_r = \begin{bmatrix} w_{11} & \cdots & w_{1r} \\ \vdots & \ddots & \vdots \\ w_{r1} & \cdots & w_{rr} \end{bmatrix}$$

here

s the number of neurons on the recurrent layer;

$$\begin{cases} Y(t) = f_{nn}(U(t), X(t), W, B) \\ X(t) = Y_h(t-1) \end{cases}$$

Identification with ANNs (1)



$$E = Y_s - Y_m$$

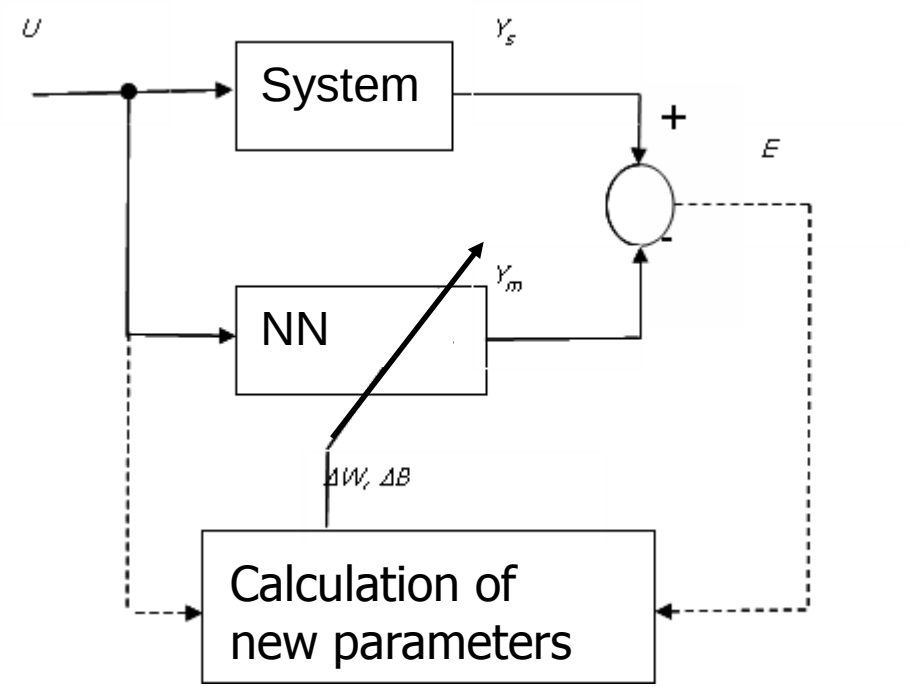
$$E \to 0$$

U is the input of the system

Y_s is the output of the system

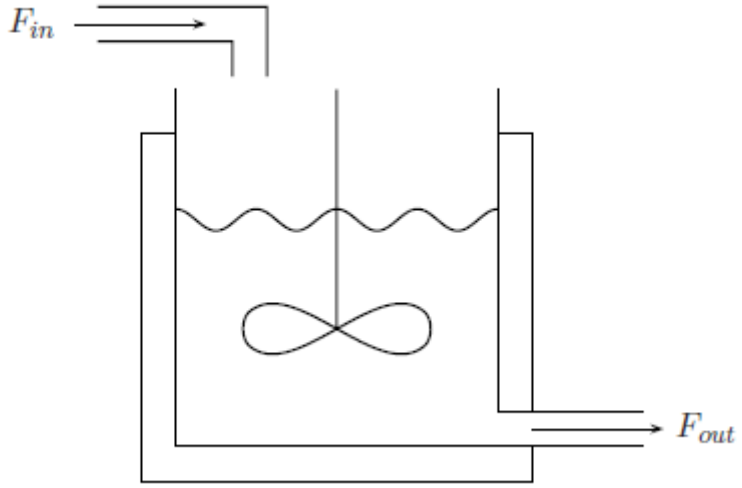
Y_m is the output of the model

Identification with ANNs (2)



Example: Jacketed CSTR (Continuous Stirred Tank Reactor)

Input-Output equation:



$$y(t+2) = 0.7653y(t+1) - 0.231y(t) +$$
$$+ 0.4801u(t+1) - 0.6407y^2(t+1) +$$
$$+ 1.014y(t)y(t+1) - 0.3921y^2(t+1) +$$
$$+ 0.592y(t+1)u(t+1) - 0.5611y(t)u(t+1)$$