Computer Science

Jerzy Świątek

Systems Modelling and Analysis

Choose yourself and new technologies

L.10 Multistage decision making

HUMAN CAPITAL
HUMAN – BEST INVESTMENT!
Wrocław University of Technology
EUROPEAN SOCIAL FUND
Project co-financed from the EU European Social Fund

MULTISTAGE DECISION MAKING

HUMAN CAPITAL
HUMAN – BEST INVESTMENT!
Wrocław University of Technology
EUROPEAN SOCIAL FUND
Project co-financed from the EU European Social Fund

$$x_1^*, x_2^*, \ldots, x_S^* \rightarrow F(x_1^*, x_2^*, \ldots, x_S^*) = \min_{x_1, x_2, \ldots, x_S \in \mathscr{D}_x} F(x_1, x_2, \ldots, x_S)$$

$$\mathscr{D}_x = (x_1, x_2, \cdots, x_S)$$

$$\left\{ [x_1\ x_2\ \cdots\ x_S]^T \in \mathscr{R}^S :\ \varphi_l(x_1, x_2, \cdots, x_S) = 0, l = 1, 2, \ldots, L, \right.$$
$$\left. \psi_m(x_1, x_2, \cdots, x_S) \leq 0, m = 1, 2, \ldots, M \right\}$$

The above task may be solved step by step, selecting a single decision variable to be optimized and relating with remaining decision variables.
Let us denote:

$$F \equiv F_S, \quad \varphi_l \equiv \varphi_{lS}, l = 1, 2, \ldots, L, \quad \psi_m = \psi_{mS}, m = 1, 2, \ldots, M \quad \mathscr{D}_x \equiv \mathscr{D}_{xS}$$

Multistage optimization

Step 1. $x_S^* = G_S(x_1, \ldots, x_{S-1}) \rightarrow F_S(x_1, x_2, \ldots, x_S^*) = \min_{x_S \in \mathcal{D}_{xS}} F_S(x_1, x_2, \ldots, x_S)$

The value of the goal function in the optimal solution:

$$F_{S-1}(x_1, x_2, \ldots, x_{S-1}) \stackrel{\Delta}{=} F_S(x_1, x_2, \ldots, x_S^*) = F_S(x_1, x_2, \ldots, G_S(x_1, \ldots, x_{S-1}))$$

Constraints in the optimal solution:

$$\mathcal{D}_{xS-1}(x_1, \ldots, x_{S-1}) \stackrel{\Delta}{=} \mathcal{D}_{xS}(x_1, \ldots, x_{S-1}, x_S^* = G_S(x_1, \ldots, x_{S-1})) =$$

$$\left\{ \begin{array}{l} [x_1 \ x_2 \ \cdots \ x_{S-1}]^T \in \mathcal{R}^{S-1} : \\ \varphi_{lS}(x_1, x_2, \cdots, G_S(x_1, \ldots, x_{S-1})) = \varphi_{lS-1}(x_1, x_2, \cdots, x_{S-1}) = 0, \ l = 1, 2, \ldots, L, \\ \psi_{mS}(x_1, x_2, \cdots, G_S(x_1, \ldots, x_{S-1})) = \psi_{mS-1}(x_1, x_2, \cdots, x_{S-1}) \le 0, \ m = 1, 2, \ldots, M \end{array} \right\}$$

Multistage optimization

Step 2. $x^*_{S-1} = G_{S-1}(x_1, \ldots, x_{S-2}) \rightarrow F_{S-1}(x_1, x_2, \ldots, x^*_{S-1}) = \min\limits_{x_{S-1} \in \mathscr{D}_{xS-1}} F_{S-1}(x_1, x_2, \ldots, x_{S-1})$

The value of the goal function in the optimal solution:

$$F_{S-2}(x_1, x_2, \ldots, x_{S-2}) \stackrel{\Delta}{=} F_{S-1}(x_1, x_2, \ldots, x^*_{S-1}) = F_{S-1}(x_1, x_2, \ldots, G_{S-1}(x_1, \ldots, x_{S-2}))$$

Constraints in the optimal solution:

$$\mathscr{D}_{xS-2}(x_1, \ldots, x_{S-2}) \stackrel{\Delta}{=} \mathscr{D}_{xS-1}(x_1, \ldots, x_{S-2}, x^*_{S-1} = G_{S-1}(x_1, \ldots, x_{S-2})) =$$

$$\left\{ \begin{array}{l} [x_1 \; x_2 \; \cdots \; x_{S-2}]^T \in \mathscr{R}^{S-2} : \\ \varphi_{lS-1}(x_1, x_2, \cdots, G_{S-1}(x_1, \ldots, x_{S-2})) = \varphi_{lS-2}(x_1, x_2, \cdots, x_{S-2}) = 0, \; l = 1, 2, \ldots, L, \\ \psi_{mS-1}(x_1, x_2, \cdots, G_{S-1}(x_1, \ldots, x_{S-2})) = \psi_{mS-2}(x_1, x_2, \cdots, x_{S-2}) \le 0, \; m = 1, 2, \ldots, M \end{array} \right\}$$

Multistage optimization

Step S-1. $x_2^* = G_2(x_1) \rightarrow F_2(x_1, x_2^*) = \min_{x_2 \in \mathcal{D}_{x2}} F_2(x_1, x_2)$

The value of the goal function in the optimal solution:

$$F_1(x_1) \stackrel{\Delta}{=} F_2(x_1\, x_2^*) = F_2(x_1, G_2(x_1))$$

Constraints in the optimal solution:

$$\mathcal{D}_{x1}(x_1) \stackrel{\Delta}{=} \mathcal{D}_{x2}(x_1, x_2^* = G_2(x_1)) = \left\{ \begin{array}{l} x_1 \in \mathcal{R} : \\ \varphi_{l2}(x_1, G_2(x_1)) = \varphi_{l1}(x_1) = 0,\ l = 1, 2, \ldots, L, \\ \psi_{m2}(x_1, G_2(x_1)) = \psi_{m1}(x_1) \le 0,\ m = 1, 2, \ldots, M \end{array} \right\}$$

Multistage optimization

Step S. $x_1^* \to F_1(x_1^*) = \min_{x_1 \in \mathscr{D}_{x1}} F_1(x_1)$

We may now return to expressions „*G*” determined in the previous steps

$$x_1^*$$

$$x_2^* = G_2(x_1^*)$$

$$\vdots$$

$$x_{S-1}^* = G_{S-1}(x_1^*, x_2^* \ldots, x_{S-2}^*)$$

$$x_S^* = G_S(x_1^*, x^*, \ldots, x_{S-1}^*)$$

Multistage optimization

Example: $$F(x_1, x_2) = x_1^2 + x_1 x_2 + x_2^2 = F_2(x_1, x_2)$$

Step 1. $$x_2^* = G_2(x_1) \to F_2(x_1, x_2^*) = \min_{x_2} \left\{ x_1^2 + x_1 x_2 + x_2^2 \right\}$$

$$\frac{\partial}{\partial x_2} F_2(x_1, x_2^*) = x_1 + 2x_2^* = 0 \Rightarrow x_2^* = -\frac{1}{2} x_1 = G_2(x_1)$$

$$F_1(x_1) \overset{\Delta}{=} F(x_1, x_2^*) = F(x_1, G_2(x_1)) = (x_1)^2 + x_1\left(-\frac{1}{2}x_1\right) + \left(-\frac{1}{2}x_1\right)^2$$

$$F_1(x_1) = \frac{3}{4} x_1^2$$

Multistage optimization

Step 2. $$x_1^* \Longrightarrow F_1(x_1) = \min_{x_1}\left\{\frac{3}{4}x_1^2\right\}$$

$$\frac{d}{dx_1}F_1(x_1) = 2\frac{3}{4}x_1 = 0 \Rightarrow x_1^* = 0$$

Now we may return to initial condition:

$$x_1^* = 0$$

$$x_2^* = G_2(x_1^*) = -\frac{1}{2}x_1^* = 0$$

DYNAMIC PROGRAMMING

HUMAN CAPITAL
HUMAN – BEST INVESTMENT!
Wrocław University of Technology
EUROPEAN SOCIAL FUND
Project co-financed from the EU European Social Fund

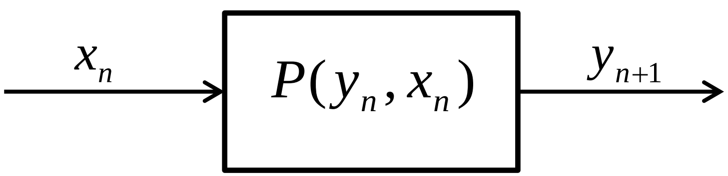
Dynamic programming

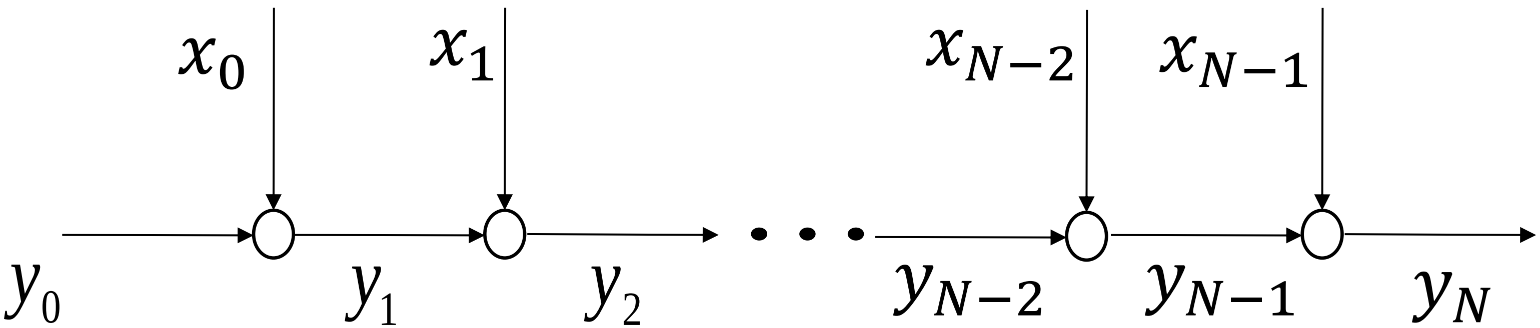
Dynamic process: $y_{n+1} = P(y_n, x_n),\ y_o$

n – the time step $n = 1, 2, \ldots, N$

x_n – decision made in the *n*–th step

y_n – state of the process in the *n*–th step





Decision making task: to determine a sequence of decisions: $x_0^*, x_1^*, \ldots, x_{N-1}^*$,

After N steps we want to achieve some goal, e.g.: $y_N = y^*$ – desired state,

The performance index $Q\left(x_0, \ldots, x_{N-1}, y_1, \ldots, y_N\right)$ must take minimal value

Dynamic programming

Let us evaluate the quality of the sequence of decisions and their effects

$$Q(x_0, x_1, \ldots, x_{N-1}, y_1, y_2, \ldots, y_N)$$

Examples.:

$$Q(x_0, x_1, \ldots, x_{N-1}, y_1, y_2, \ldots, y_N) = \sum_{n=0}^{N-1} x_n^2, \quad y_N = y^*$$

$$Q(x_0, x_1, \ldots, x_{N-1}, y_1, y_2, \ldots, y_N) = \sum_{n=0}^{N-1} x_n^2, \quad \underline{y} \le y_N \le \overline{y}$$

$$Q(x_0, x_1, \ldots, x_{N-1}, y_1, y_2, \ldots, y_N) = \sum_{n=1}^{N-1} (y_n - y_n^*), \quad 0 \le x_n \le \overline{x}$$

$$Q(x_0, x_1, \ldots, x_{N-1}, y_1, y_2, \ldots, y_N) = \sum_{n=0}^{N-1} A_{n+1}(x_n, y_{n+1}), \quad \text{with constraints } x_n \text{ and } y_n$$

HUMAN CAPITAL
HUMAN – BEST INVESTMENT!
Wrocław University of Technology
EUROPEAN SOCIAL FUND
Project co-financed from the EU European Social Fund

Dynamic programming

The following procedure may be applied:

$$y_0$$

$$y_1 = P(y_0, x_0)$$

$$y_2 = P(y_1, x_1) = P(P(y_0, x_0), x_1) \overset{\Delta}{=} P_1(y_0, x_0, x_1)$$

$$\vdots$$

$$y_N = P(y_{N-1}, x_{N-1}) = P(P(y_{N-2}, x_{N-2}), x_{N-1}) = \overset{y_{n+1}=P(y_n, x_n)}{\cdots} = P_{N-1}(y_0, x_0, x_1, \ldots, x_N)$$

After substitution to *Q(.)* we obtain:

$$Q(x_0, x_1, \ldots, x_{N-1}, y_1, y_2, \ldots, y_N) = \sum_{n=0}^{N-1} A_{n+1}(x_n, y_{n+1}) \overset{\Delta}{=} F(y_0, x_0, x_1, \ldots, x_{N-1})$$

Dynamic programming

Due to the fact, that:

$$Q(x_0, x_1, \ldots, x_{N-1}, y_1, y_2, \ldots, y_N) = \sum_{n=0}^{N-1} A_{n+1}(x_n, y_{n+1}) \stackrel{\Delta}{=} F(y_0, x_0, x_1, \ldots, x_{N-1})$$

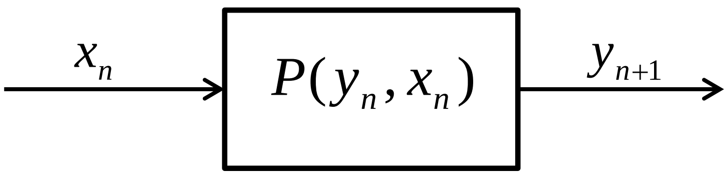
The optimization task:

$$x_0^*, x_1^*, \ldots, x_{N-1}^* \to Q(x_0^*, x_1^*, \ldots, x_{N-1}^*, y_1, y_2, \ldots, y_N) = \min_{x_0, x_1, \ldots, x_{N-1}} Q(x_0, x_1, \ldots, x_{N-1}, y_1, y_2, \ldots, y_N)$$

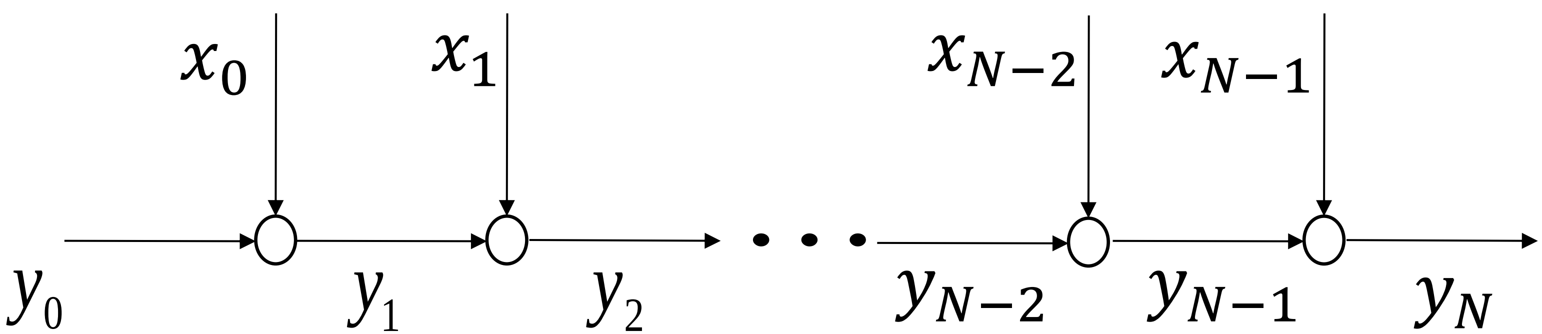
is equivalent to:

$$x_0^*, x_1^*, \ldots, x_{N-1}^* \to F(y_0, x_0^*, x_1^*, \ldots, x_{N-1}^*) = \min_{x_0, x_1, \ldots, x_{N-1}} F(y_0, x_0, x_1, \ldots, x_{N-1})$$

In order to solve the task above, the multi-stage approach may be applied.



$$y_{n+1} = P(y_n, x_n),\ y_o$$



$$Q(x_0, x_1, \ldots, x_{N-1}, y_1, y_2, \ldots, y_N) = \sum_{n=0}^{N-1} A_{n+1}(x_n, y_{n+1}) \overset{\Delta}{=} F(y_0, x_0, x_1, \ldots, x_{N-1})$$

Dynamic programming

$$Q(x_0, x_1, \ldots, x_{N-1}, y_1, y_2, \ldots, y_N) = \sum_{n=0}^{N-1} A_{n+1}(x_n, y_{n+1}) \overset{\Delta}{=} F(y_0, x_0, x_1, \ldots, x_{N-1})$$

Remark:

The final state y_N depends on decision x_{N-1} and the state y_{N-1}, which is dependent on previous decisions $x_0, x_1, \ldots, x_{N-2}$.

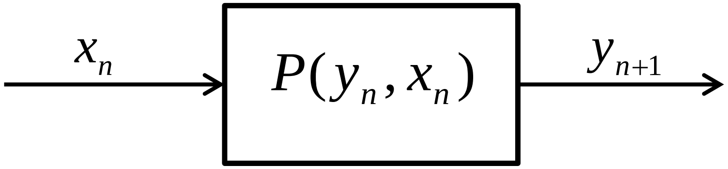
The previous state y_{N-1} depends on decision x_{N-2} and the state y_{N-2}, which is dependent on previous decisions $x_0, x_1, \ldots, x_{N-3}$.

⋮

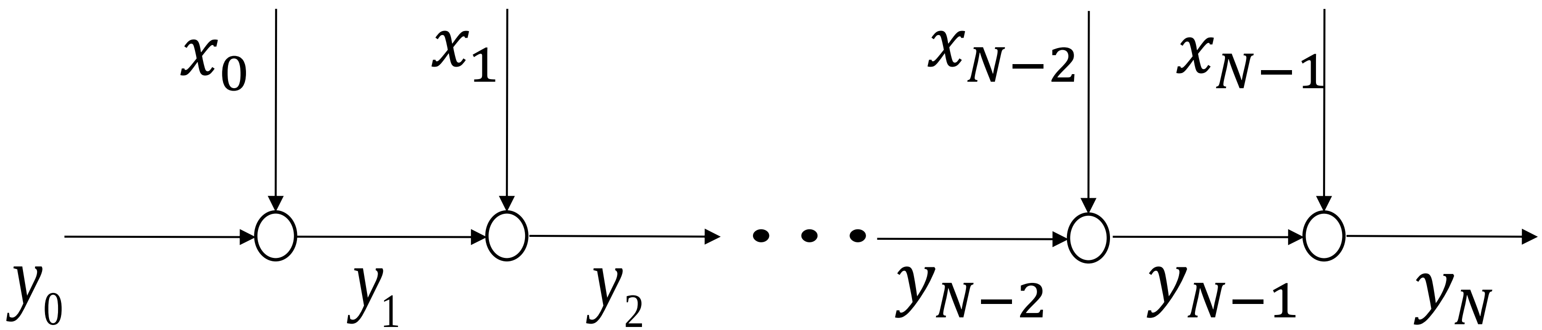
The state y_2 depends on decision x_1 and the state y_1, which is dependent on previous decisions x_0, x_1.

The state y_1 depends on decision x_0 and the state y_0, which values are known.

In order to solve the task above, the multi-stage approach may be applied. Taking into account the form of performance index (sum of functions of decisions x_n resulting states y_{n+1}). Beginning from optimization of the last term we relate the solution from the previous state and previous decisions.



$$y_{n+1} = P(y_n, x_n),\ y_o$$



$$Q(x_0, x_1, \ldots, x_{N-1}, y_1, y_2, \ldots, y_N) = \sum_{n=0}^{N-1} A_{n+1}(x_n, y_{n+1}) \overset{\Delta}{=} F(y_0, x_0, x_1, \ldots, x_{N-1})$$

Dynamic programming

Step 1. $$x^*_{N-1} \rightarrow \min_{x_{N-1}} A_N\left(x_{N-1}, y_N\right)$$

We know, that: $$y_N = P\left(y_{N-1}, x_{N-1}\right)$$

$$x^*_{N-1} = G_{N-1}\left(y_{N-1}\right) \rightarrow \min_{x_{N-1}} A_N\left(x_{N-1}, P\left(y_{N-1}, x_{N-1}\right)\right)$$

$$V_{N-1}\left(y_{N-1}\right) \overset{\Delta}{=} \min_{x_{N-1}} A_N\left(x_{N-1}, P\left(y_{N-1}, x_{N-1}\right)\right) =$$

$$= A_N\left(x^*_{N-1}, P\left(y_{N-1}, x^*_{N-1}\right)\right) = A_N\left(G_{N-1}\left(y_{N-1}\right), P\left(y_{N-1}, G_{N-1}\left(y_{N-1}\right)\right)\right)$$

Dynamic programming

Step 2. $x^*_{N-2} \to \min_{x_{N-2}} \{A_{N-1}(x_{N-2}, y_{N-1}) + V_{N-1}(y_{N-1})\}$

We know, that $y_{N-1} = P(y_{N-2}, x_{N-2})$

$$x^*_{N-2} = G_{N-2}(y_{N-2}) \to \min_{x_{N-2}} \{A_{N-1}(x_{N-2}, P(y_{N-2}, x_{N-2})) + V_{N-1}(P(y_{N-2}, x_{N-2}))\}$$

$$V_{N-2}(y_{N-2}) \overset{\Delta}{=} \min_{x_{N-2}} \{A_{N-1}(x_{N-2}, P(y_{N-2}, x_{N-2})) + V_{N-1}(P(y_{N-2}, x_{N-2}))\} =$$

$$= \{A_{N-1}(x^*_{N-2}, P(y_{N-2}, x^*_{N-2})) + V_{N-1}(P(y_{N-2}, x^*_{N-2}))\} =$$

$$= A_{N-1}(G_{N-2}(y_{N-2}), P(y_{N-2}, G_{N-2}(y_{N-2}))) + V_{N-1}(P(y_{N-2}, G_{N-2}(y_{N-2})))$$

⋮

Dynamic programming

Step N-1. $x_1^* \to \min_{x_1} \left\{ A_2(x_1, y_2) + V_2(y_2) \right\}$

We know, that $y_2 = P(y_1, x_1)$

$$x_1^* = G_1(y_1) \to \min_{x_1} \left\{ A_2(x_1, P(y_1, x_1)) + V_2(P(y_1, x_1)) \right\}$$

$$V_1(y_1) \overset{\Delta}{=} \min_{x_1} \left\{ A_2(x_1, P(y_1, x_1)) + V_2(P(y_1, x_1)) \right\} =$$
$$= A_2(x_1^*, P(y_1, x_1^*)) + V_2(P(y_1, x_1^*))$$
$$= A_2(G_1(y_1), P(y_1, G_1(y_1))) + V_2(P(y_1, G_1(y_1)))$$

Dynamic programming

Step N. $x_0^* \rightarrow \min_{x_0}\{A_1(x_0, y_1) + V_1(y_1)\}$

We know, that $y_1 = P(y_0, x_0)$

$$x_0^* = G_0(y_0) \rightarrow \min_{x_0}\{A_1(x_0, P(y_0, x_0)) + V_1(P(y_0, x_0))\}$$

y_0 is known and from now on successive decisions may be determined

$x_0^*, x_1^*, \ldots, x_{N-1}^*,$

$$x_0^* = G_0(y_0) \rightarrow y_1 = P(y_0, x_0^*)$$
$$x_1^* = G_1(y_1) \rightarrow y_2 = P(y_2, x_2^*)$$
$$\vdots$$
$$x_{N-2}^* = G_{N-2}(y_{N-2}) \rightarrow y_{N-1} = P(y_{N-2}, x_{N-2}^*)$$
$$x_{N-1}^* = G_{N-1}(y_{N-1}) \rightarrow y_N = P(y_{N-1}, x_{N-1}^*)$$

Dynamic programming

Example: $y_{n+1} = P(y_n, x_n) = 2y_n + x_n, \quad y_o = 0$

$$Q(x_0, x_1, y_1, y_2) = \sum_{n=0}^{1} A_{n+1}(x_n, y_{n+1}) = \sum_{n=0}^{1} \left(x_n^2 + (y_{n+1} - 5)^2\right) =$$

$$= \left(x_0^2 + (y_1 - 5)^2\right) + \left(x_1^2 + (y_2 - 5)^2\right)$$

Step1.

$$x_1^* \to \min_{x_1}\left(x_1^2 + (y_2 - 5)^2\right)$$

$$y_2 = 2y_1 + x_1$$

$$x_1^* = G_1(y_1) \to \min_{x_1}\left(x_1^2 + (2y_1 + x_1 - 5)^2\right)$$

$$2x_1^* + 2\left(2y_1 + x_1^* - 5\right) = 0 \Rightarrow x_1^* = G_1(y_1) = \frac{5}{2} - y_1$$

$$V_1(y_1) = \left(\frac{5}{2} - y_1\right)^2 + \left(2y_1 + \frac{5}{2} - y_1 - 5\right)^2 = 2\left(y_1 - \frac{5}{2}\right)^2$$

Dynamic programming

Step 2.

$$x_0^* \rightarrow \min_{x_0}\left\{\left(x_0^2+(y_1-5)^2\right)+2\left(y_1-\frac{5}{2}\right)^2\right\}$$

$$y_1 = 2y_0 + x_0$$

$$x_0^* = G_0(y_0) \rightarrow \min_{x_0}\left\{\left(x_0^2+(2y_0+x_0-5)^2\right)+2\left(2y_0+x_0-\frac{5}{2}\right)^2\right\}$$

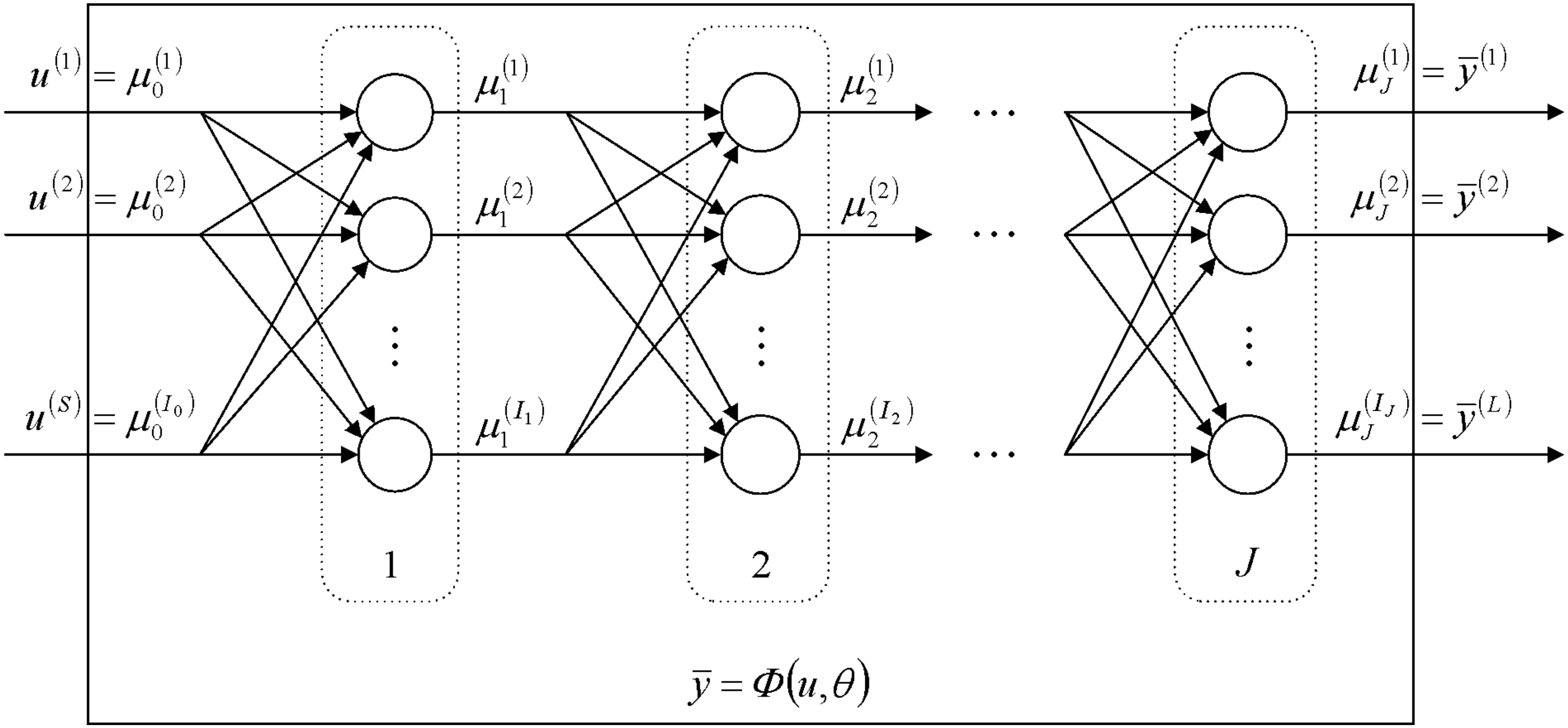
$$2x_0^*+2\left(2y_0+x_0^*-5\right)+4\left(2y_0+x_0-\frac{5}{2}\right)=0 \Rightarrow x_0^*=G_0(y_0)=\frac{15}{8}-\frac{3}{2}y_0=\frac{15}{8}$$

Now, we return back with calculations:

$$x_0^*=\frac{15}{8} \rightarrow y_1 = 2\times 0+\frac{15}{8}=\frac{15}{8}$$

$$x_1^*=\frac{5}{2}-\frac{15}{8}=\frac{5}{8} \rightarrow y_1 = 2\times 0+\frac{5}{8}=\frac{5}{8}$$

Neural networks



Linear case - recursive algorithm

$$Q_N(\theta) = \sum_{n=1}^{N} (y_n - \bar{y}_n)^2 = \sum_{n=1}^{N} \left(y_n - \theta^T \varphi(u_n)\right)^2$$

For N measurements $U_N = [u_1 \quad u_2 \quad \cdots \quad u_N], \quad Y_N = [y_1 \quad y_2 \quad \cdots \quad y_N]$

$$\theta_N^* = \left[\sum_{n=1}^{N} \varphi(u_n)\varphi^T(u_n)\right]^{-1} \times \sum_{n=1}^{N} y_n \varphi(u_n)$$

New *N+1* measurement point u_{N+1}, y_{N+1}

How to adopt vector of parameters using new measurement?

$$\theta_{N+1}^* = A\left(\theta_N^*, u_{N+1}, y_{N+1}\right)$$

$$Q_{N+1}(\theta) = \sum_{n=1}^{N+1} (y_n - \bar{y}_n)^2 = \sum_{n=1}^{N+1} \left(y_n - \theta^T \varphi(u_n)\right)^2$$

$$\theta_{N+1}^* = \left[\sum_{n=1}^{N+1} \varphi(u_n)\varphi^T(u_n)\right]^{-1} \times \sum_{n=1}^{N+1`} y_n \varphi(u_n) =$$

$$\left[\sum_{n=1}^{N} \varphi(u_n)\varphi^T(u_n) + \varphi(u_{N+1})\varphi^T(u_{N+1})\right]^{-1} \times \left[\sum_{n=1}^{N`} y_n \varphi(u_n) + y_{N+1}\varphi(u_{N+1})\right]$$

Linear case - recursive algorithm

$$\left(\mathbf{A}+\mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right)^{-1}=\mathbf{A}^{-1}-\mathbf{A}^{-1}\mathbf{B}\left(\mathbf{D}+\mathbf{C}\mathbf{A}^{-1}\mathbf{B}\right)^{-1}\mathbf{C}\mathbf{A}^{-1}$$

$\mathbf{B}=\varphi$ – wektor kolumnowy

$$\mathbf{D}^{-1}=1$$

$$\mathbf{C}=\varphi^{T}$$

$$\left(\mathbf{A}+\varphi\varphi^{T}\right)^{-1}=\mathbf{A}^{-1}-\mathbf{A}^{-1}\varphi\left(1+\varphi^{T}\mathbf{A}^{-1}\varphi\right)^{-1}\varphi^{T}\mathbf{A}^{-1}$$

$$\mathbf{D}^{-1}=-1$$

$$\left(\mathbf{A}-\varphi\varphi^{T}\right)^{-1}=\mathbf{A}^{-1}+\mathbf{A}^{-1}\varphi\left(1-\varphi^{T}\mathbf{A}^{-1}\varphi\right)^{-1}\varphi^{T}\mathbf{A}^{-1}$$

Linear case - recursive algorithm

Let: $$\varphi_n = \varphi(u_n),\ n = 1, 2, \ldots, N+1$$

$$\left(\sum_{n=1}^{N+1} \varphi_n \varphi_n^T\right)^{-1} = \left(\sum_{n=1}^{N} \varphi_n \varphi_n^T + \varphi_{N+1}\varphi_{N+1}^T\right)^{-1} =$$

$$= \left(\sum_{n=1}^{N} \varphi_n \varphi_n^T\right)^{-1} - \left(\sum_{n=1}^{N} \varphi_n \varphi_n^T\right)^{-1} \frac{\varphi_{N+1}\varphi_{N+1}^T}{1 + \varphi_{N+1}^T \left(\sum_{n=1}^{N} \varphi_n \varphi_n^T\right)^{-1} \varphi_{N+1}} \left(\sum_{n=1}^{N} \varphi_n \varphi_n^T\right)^{-1}$$

Let: $$P_N = \left(\sum_{n=1}^{N} \varphi_n \varphi_n^T\right)^{-1}$$

$$P_{N+1} = P_N - P_N \frac{\varphi_{N+1}\varphi_{N+1}^T}{1 + \varphi_{N+1}^T P_N \varphi_{N+1}} P_N$$

Linear case - recursive algorithm

$$\theta^*_{N+1} = \left(P_N - P_N \frac{\varphi_{N+1}\varphi^T_{N+1}}{1+\varphi^T_{N+1}P_N\varphi_{N+1}} P_N \right)\left(\sum_{n=1}^{N} \varphi_n y_n + \varphi_{N+1} y_{N+1} \right) =$$

$$= P_N \sum_{n=1}^{N} \varphi_n y_n + P_N \varphi_{N+1} y_{N+1} - \frac{P_N \varphi_{N+1}\varphi^T_{N+1} P_N}{1+\varphi^T_{N+1}P_N\varphi_{N+1}} \sum_{n=1}^{N} \varphi_n y_n - \frac{P_N \varphi_{N+1}\varphi^T_{N+1} P_N \varphi_{N+1} y_{N+1}}{1+\varphi^T_{N+1}P_N\varphi_{N+1}} =$$

$$= P_N \sum_{n=1}^{N} \varphi_n y_n + \frac{P_N \varphi_{N+1} y_{N+1} + P_N \varphi_{N+1} y_{N+1} \varphi^T_{N+1} P_N \varphi_{N+1} - P_N \varphi_{N+1} y_{N+1} \varphi^T_{N+1} P_N \varphi_{N+1} - P_N \varphi_{N+1}\varphi^T_{N+1} P_N \sum_{n=1}^{N} \varphi_n y_n}{1+\varphi^T_{N+1}P_N\kappa_{N+1}} =$$

$$= P_N \sum_{n=1}^{N} \varphi_n y_n + \frac{P_N \varphi_{N+1} y_{N+1} - P_N \varphi_{N+1}\varphi^T_{N+1} P_N \sum_{n=1}^{N} \varphi_n y_n}{1+\varphi^T_{N+1}P_N\varphi_{N+1}} y =$$

$$= P_N \sum_{n=1}^{N} \varphi_n y_n + \frac{P_N \varphi_{N+1} y_{N+1}}{1+\varphi^T_{N+1}P_N\varphi_{N+1}} \left(y_{N+1} + \varphi^T_{N+1} P_N \sum_{n=1}^{N} \varphi_n y_n \right)$$

$$= \theta^*_N + K_{N+1}\left(y_{N+1} - \varphi^T_{N+1}\theta_N \right)$$

where:

$$K_{N+1} = \frac{P_N \varphi_{N+1}}{1+\varphi^T_{N+1}P_N\varphi_{N+1}}$$

Linear case - recursive algorithm

Finaly:

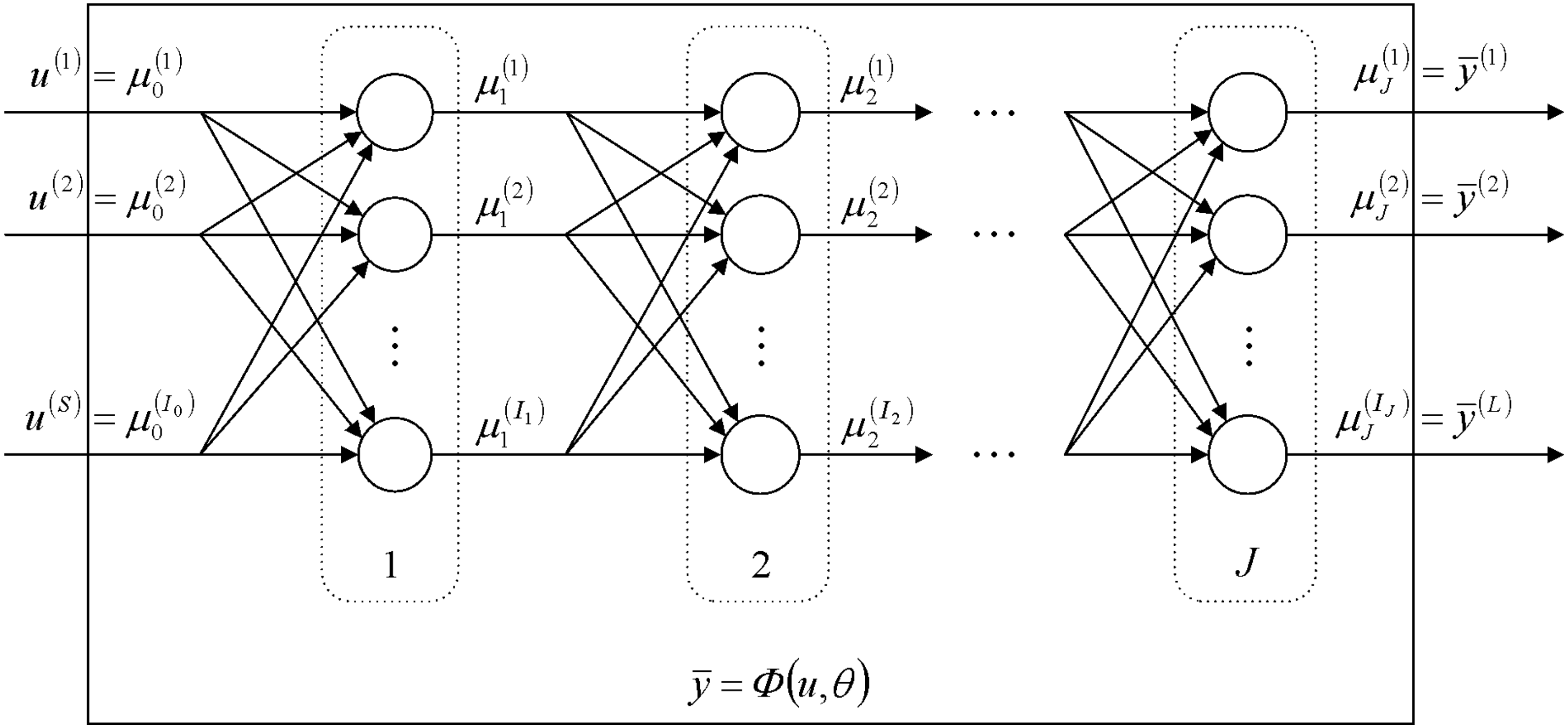
$$\theta_N^* = \left(\sum_{n=1}^{N} \varphi_n \varphi_n^T\right)^{-1} \sum_{n=1}^{N} \varphi_n y_n = P_N \sum_{n=1}^{N} \varphi_n y_n \text{, gdzie: } P_N = \left(\sum_{n=1}^{N} \varphi_n \varphi_n^T\right)^{-1}$$

$$\theta_{N+1}^* = A\left(\theta_N^*, u_{N+1}, y_{N+1}\right) = \theta_N^* + K_{N+1}\left[y_{N+1} - \varphi(u_{N+1})^T \theta_N^*\right]$$

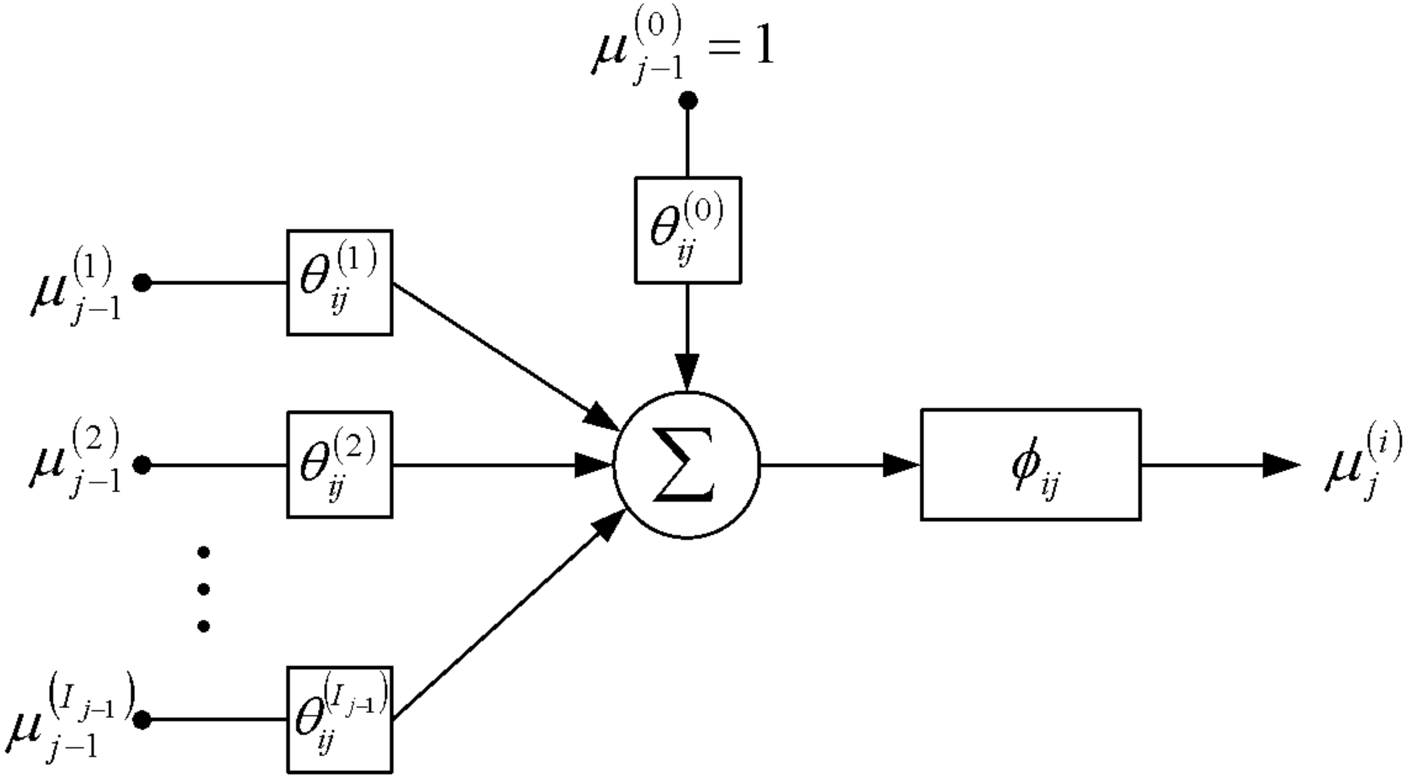
$$K_{N+1} = \frac{P_N \varphi(u_{N+1})}{1 + \varphi(u_{N+1})^T P_N \varphi(u_{N+1})}$$

$$P_{N+1} = P_N - \frac{P_N \varphi(u_{N+1}) \varphi(u_{N+1})^T P_N}{1 + \varphi(u_{N+1})^T P_N \varphi(u_{N+1})}$$

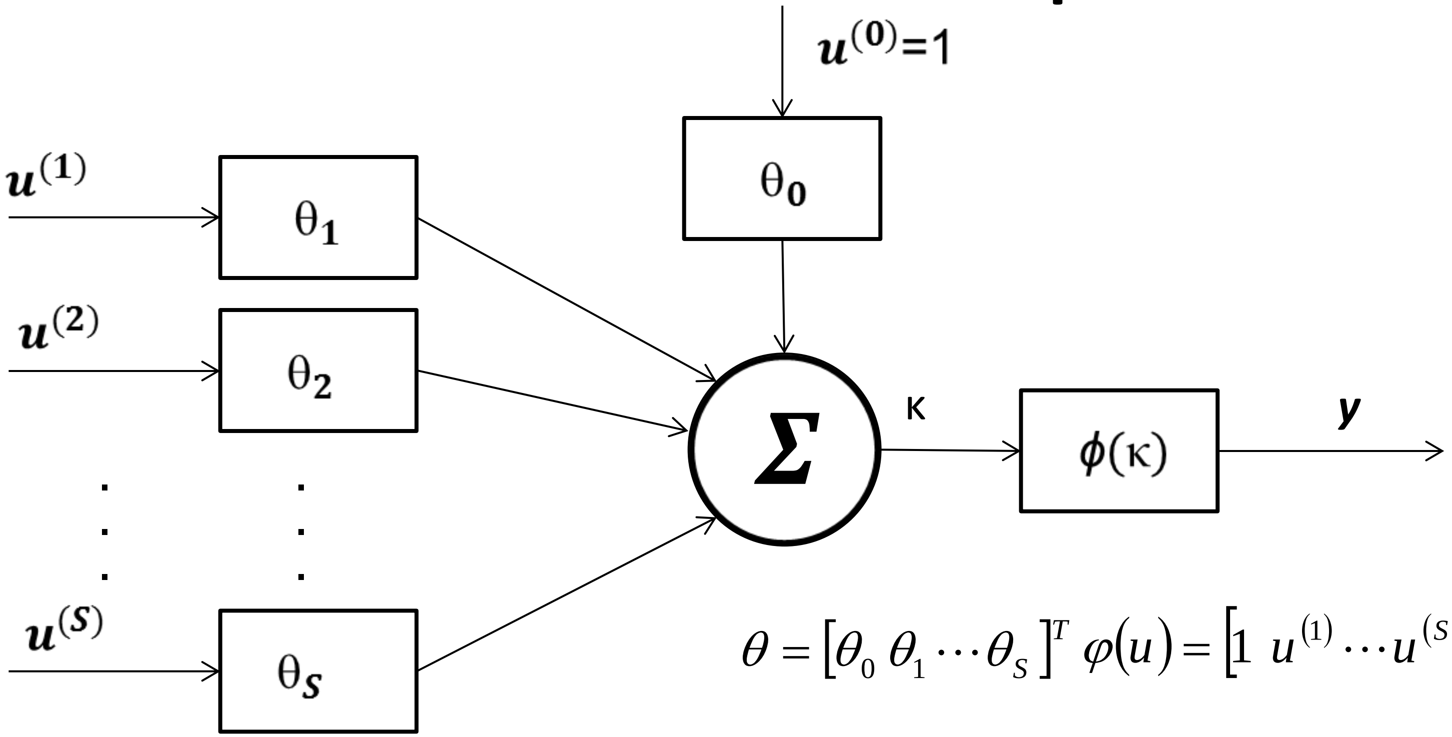
Neural networks



Neuron model



Neuron model simplification



$$\theta = [\theta_0\ \theta_1 \cdots \theta_S]^T \quad \varphi(u) = [1\ u^{(1)} \cdots u^{(S)}]^T$$

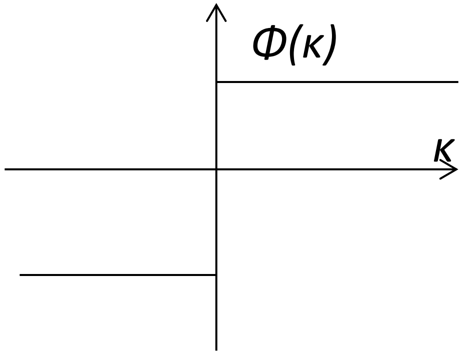
$$y = \phi\left(\sum_{s=1}^{S} \theta_s u^{(s)} + \theta_0\right) = \phi\left(\theta^T \varphi(u)\right)$$

Φ – activation function

Activation function

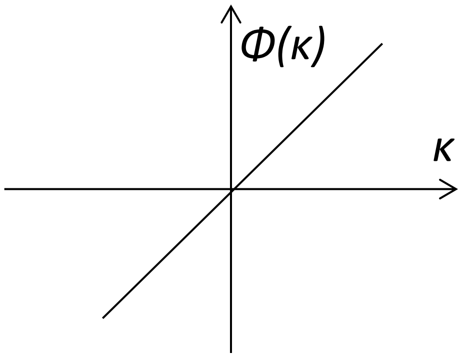
$$\phi(\kappa) = \begin{cases} 1 \; dla \;\; \kappa > 0 \\ -1 \; dla \;\; \kappa \le 0 \end{cases}$$

Perceptron



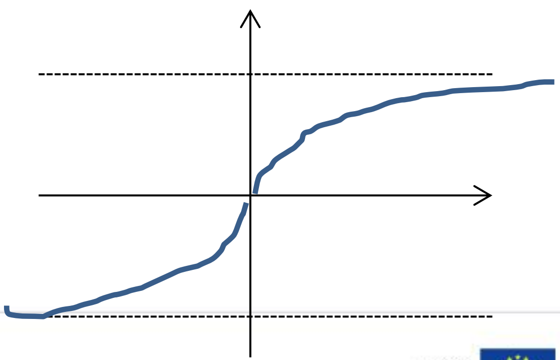
$$\phi(\kappa) = \kappa$$

Adaline
Adaptive Linear Neuron



$$\phi(\kappa) = \frac{1}{1 + e^{-\beta\kappa}}$$

$$\phi(\kappa) = \tanh(\beta\kappa) = \frac{1 - e^{-\beta\kappa}}{1 + e^{-\beta\kappa}}$$

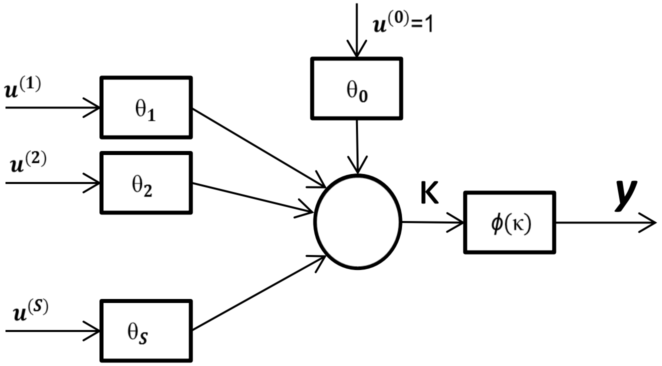


Sigmoidal Neuron

Neuron learning

Adaline
Adaptive Linear Neuron

$$\phi(\kappa) = \kappa$$



$$y = \phi\left(\sum_{s=1}^{S} \theta_s u^{(s)} + \theta_0\right) = \phi\left(\theta^T \varphi(u)\right) = \theta^T \varphi(u)$$

$$\theta = [\theta_0\ \theta_1 \cdots \theta_S]^T \qquad \varphi(u) = [1\ u^{(1)} \cdots u^{(S)}]^T$$

$$[u_1\ u_2 \cdots u_N] = U_N \qquad [y_1\ y_2 \cdots y_N] = Y_N$$

$$Q_N(\theta) = \sum_{n=1}^{N} (y_n - \bar{y}_n)^2 = \sum_{n=1}^{N} \left(y_n - \theta^T \varphi(u_n)\right)^2$$

$$\theta_N^* = \left[\sum_{n=1}^{N} \varphi(u_n)\varphi^T(u_n)\right]^{-1} \times \sum_{n=1}^{N} y_n \varphi(u_n)$$

Or recursive algorithm

Recursive algorithm

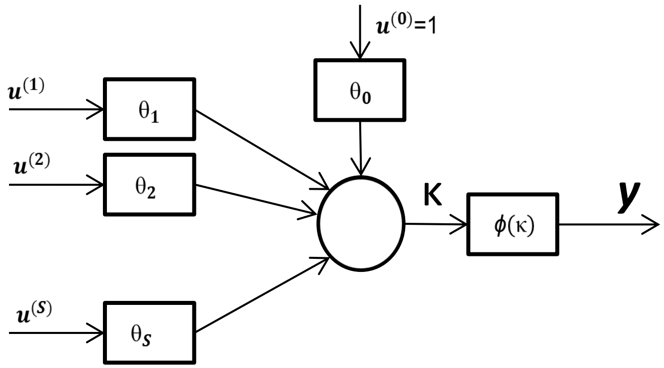
$$\theta_{N+1}^{*} = A\left(\theta_{N}^{*}, u_{N+1}, y_{N+1}\right) = \theta_{N}^{*} + K_{N+1}\left[y_{N+1} - \varphi\left(u_{N+1}\right)^{T}\theta_{N}^{*}\right]$$

$$K_{N+1} = \frac{P_{N}\varphi\left(u_{N+1}\right)}{1 + \varphi\left(u_{N+1}\right)^{T} P_{N}\varphi\left(u_{N+1}\right)}$$

$$P_{N+1} = P_{N} - \frac{P_{N}\varphi\left(u_{N+1}\right)\varphi\left(u_{N+1}\right)^{T} P_{N}}{1 + \varphi\left(u_{N+1}\right)^{T} P_{N}\varphi\left(u_{N+1}\right)}$$

Neuron learning

Like neuron Adaline (Adaptive Linear Neuron) $\phi(\kappa)$ - One to one mapping



$$y=\phi\left(\sum_{s=1}^{S}\theta_s u^{(s)}+\theta_0\right)=\phi\left(\theta^T\varphi(u)\right)$$

$$\theta=\left[\theta_0\ \theta_1\cdots\theta_S\right]^T \qquad \varphi(u)=\left[1\ u^{(1)}\cdots u^{(S)}\right]^T$$

Uczenie: $\left[u_1\ u_2\cdots u_N\right]=U_N \qquad \left[y_1\ y_2\cdots y_N\right]=Y_N$

$$Q_N(\theta)=\sum_{n=1}^{N}\left(\kappa_n-\bar{\kappa}_n\right)^2=\sum_{n=1}^{N}\left(\phi^{-1}(y_n)-\theta^T\varphi(u_n)\right)^2$$

Like before

$$\theta_N^*=\left[\sum_{n=1}^{N}\varphi(u_n)\varphi^T(u_n)\right]^{-1}\times\sum_{n=1}^{N}\phi^{-1}(y_n)\varphi(u_n)$$

Or recursive algorithm

Recursive algorithm

$$\theta_{N+1}^{*} = A\left(\theta_N^{*}, u_{N+1}, y_{N+1}\right) = \theta_N^{*} + K_{N+1}\left[\phi^{-1}\left(y_{N+1}\right) - \varphi\left(u_{N+1}\right)^T \theta_N^{*}\right]$$

$$K_{N+1} = \frac{P_N \varphi\left(u_{N+1}\right)}{1 + \varphi\left(u_{N+1}\right)^T P_N \varphi\left(u_{N+1}\right)}$$

$$P_{N+1} = P_N - \frac{P_N \varphi\left(u_{N+1}\right) \varphi\left(u_{N+1}\right)^T P_N}{1 + \varphi\left(u_{N+1}\right)^T P_N \varphi\left(u_{N+1}\right)}$$

Neuron learning – delta roll

$$Q(\theta) = \frac{1}{2}\left(y - \theta^T \varphi(u)\right)^2, \quad \theta_0$$

Numerical optimization method (it will be)

$$\theta^* \to Q(\theta^*) = \min_\theta Q(\theta)$$

$$\theta^*_{n+1} = \theta^*_n - \eta \nabla_\theta Q(\theta^*_n), \quad \theta^*_0$$

η – learning factor

$$\nabla Q(\theta) = -\left(y - \theta^T \varphi(u)\right)\varphi(u), \quad \theta_0$$

$$\theta^*_{n+1} = \theta^*_n + \eta\left(y_{n+1} - \theta^{*T}_n \varphi(u_{n+1})\right)\varphi(u_{n+1}), \quad \theta^*_0$$

$$\Delta_n = \left(y_{n+1} - \theta^{*T}_n \varphi(u_{n+1})\right)$$

$$\theta^*_{n+1} = \theta^*_n + \eta \Delta_n \varphi(u_{n+1})$$

Neuron learning – delta roll

$$Q(\theta)=\frac{1}{2}\left(y-\phi\left(\theta^T\varphi(u)\right)\right)^2,\quad \theta_0$$

Φ – activation function, differentiable

Numerical optimization method (it will be)

$$\theta^* \to Q(\theta^*)=\min_\theta Q(\theta)$$

$$\theta^*_{n+1}=\theta^*_n-\eta\nabla_\theta Q(\theta^*_n),\ \ \theta^*_0$$

η - learning factor

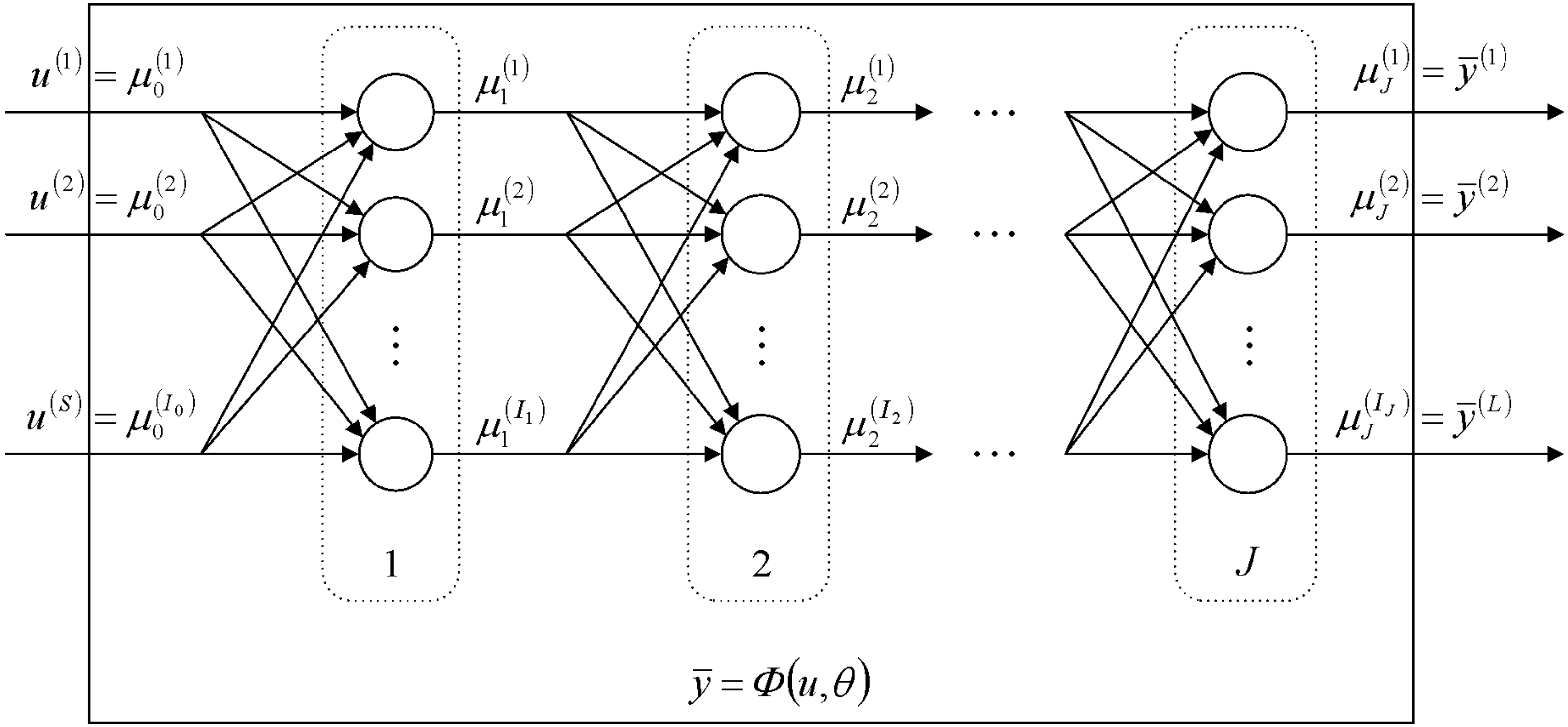
$$\nabla Q(\theta)=-\left(y-\phi\left(\theta^T\varphi(u)\right)\right)\frac{\partial\phi(\kappa)}{\partial(\kappa)}\varphi(u),\quad \theta_0$$

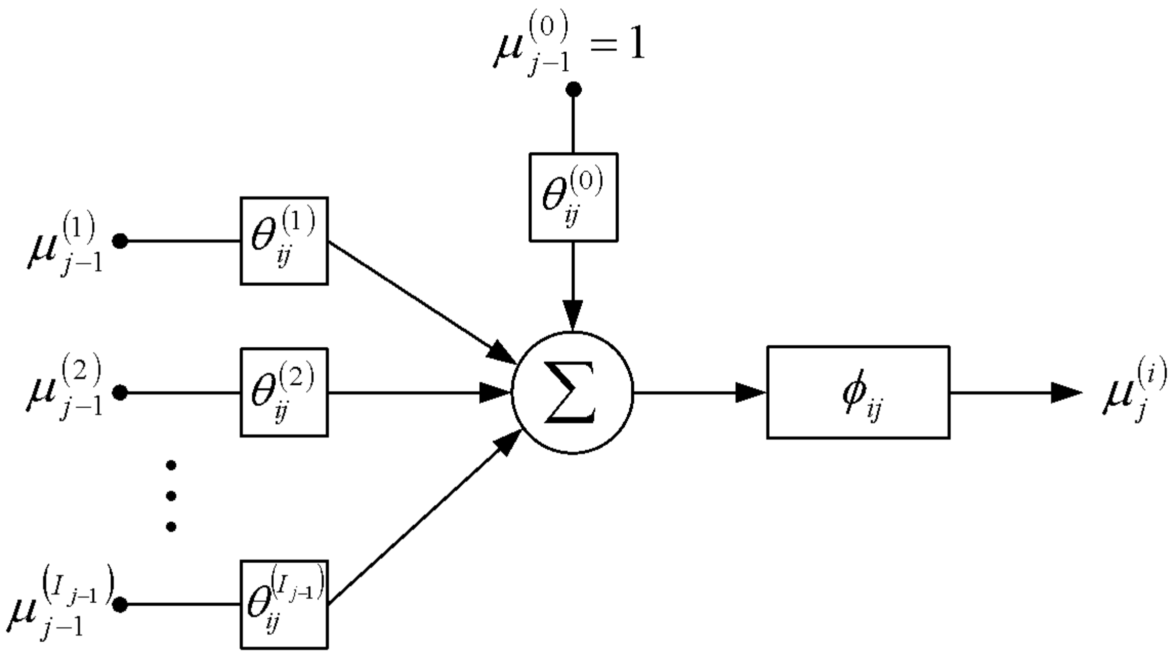
$$\theta^*_{n+1}=\theta^*_n+\eta\left(y_{n+1}-\phi\left(\theta^{*T}_n\varphi(u_{n+1})\right)\right)\frac{\partial\phi(\kappa)}{\partial\kappa}\varphi(u_{n+1}),\ \ \theta^*_0$$

$$\Delta_n=\left(y_{n+1}-\phi\left(\theta^{*T}_n\varphi(u_{n+1})\right)\right)\frac{\partial\phi(\kappa)}{\partial\kappa}$$

$$\theta^*_{n+1}=\theta^*_n+\eta\Delta_n\varphi(u_{n+1})$$

Multilayer network





$$\mu_j^{(i)} = \phi_{ij}\left(\bar{\mu}_{j-1}^T \theta_{ij}\right).$$

$$\mu_j^{(i)} = \phi_{ij}\left(\sum_{s=1}^{I_{j-1}} \mu_{j-1}^{(s)}\theta_{ij}^{(s)} + \theta_{ij}^{(0)}\right), \quad \mu_{j-1} = \begin{bmatrix} \mu_{j-1}^{(1)} \\ \mu_{j-1}^{(2)} \\ \vdots \\ \mu_{j-1}^{(I_{j-1})} \end{bmatrix}, \quad \bar{\mu}_{j-1} \stackrel{\text{df}}{=} \begin{bmatrix} \mu_{j-1}^{(0)} \\ \mu_{j-1} \end{bmatrix} = \begin{bmatrix} \mu_{j-1}^{(0)} \\ \mu_{j-1}^{(1)} \\ \mu_{j-1}^{(2)} \\ \vdots \\ \mu_{j-1}^{(I_{j-1})} \end{bmatrix}, \quad \theta_{ij} \stackrel{\text{df}}{=} \begin{bmatrix} \theta_{ij}^{(0)} \\ \theta_{ij}^{(1)} \\ \vdots \\ \theta_{ij}^{(I_{j-1})} \end{bmatrix},$$

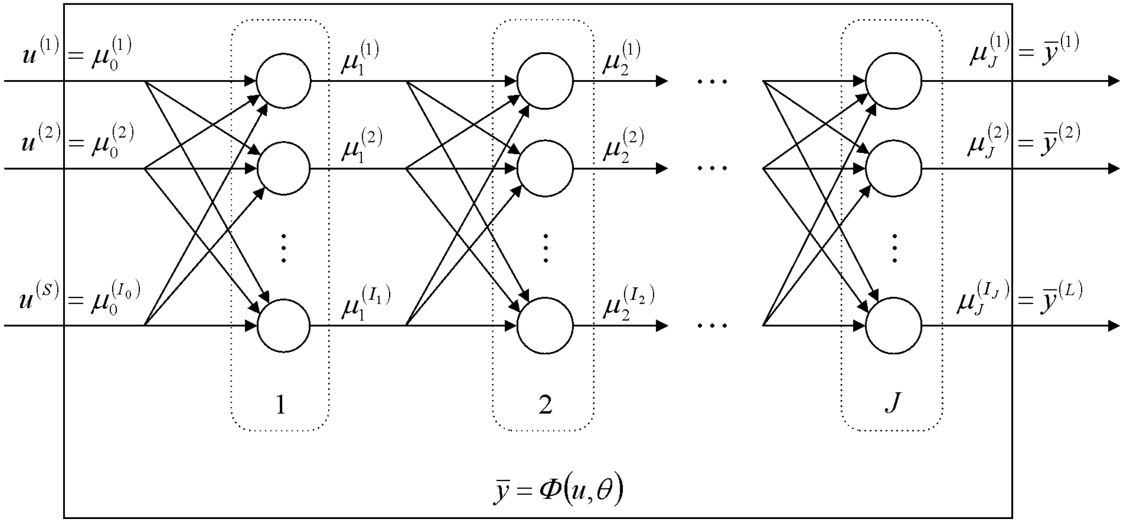
HUMAN CAPITAL
HUMAN – BEST INVESTMENT!

Wrocław University of Technology

EUROPEAN SOCIAL FUND

Project co-financed from the EU European Social Fund

Multilayer network



$$\mu_j = \begin{bmatrix} \mu_j^{(1)} \\ \mu_j^{(2)} \\ \vdots \\ \mu_j^{(I_j)} \end{bmatrix} = \begin{bmatrix} \phi_{1j}\left(\bar{\mu}_{j-1}^T \theta_{1j}\right) \\ \phi_{2j}\left(\bar{\mu}_{j-1}^T \theta_{2j}\right) \\ \vdots \\ \phi_{I_j j}\left(\bar{\mu}_{j-1}^T \theta_{I_j j}\right) \end{bmatrix} \overset{\text{df}}{=} \phi_j\left(\bar{\mu}_{j-1}, \theta_j\right),$$

$$\theta_j \overset{\text{df}}{=} \begin{bmatrix} \theta_{1j} & \theta_{2j} & \cdots & \theta_{I_j j} \end{bmatrix} = \begin{bmatrix} \theta_{1j}^{(0)} & \theta_{2j}^{(0)} & \cdots & \theta_{I_j j}^{(0)} \\ \theta_{1j}^{(1)} & \theta_{2j}^{(1)} & \cdots & \theta_{I_j j}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ \theta_{1j}^{(I_{j-1})} & \theta_{2j}^{(I_{j-1})} & \cdots & \theta_{I_j j}^{(I_{j-1})} \end{bmatrix}.$$

$$\mu_j = \phi_j\left(\bar{\mu}_{j-1}, \theta_j\right),\ j = 1, 2,\ \ldots,\ J.$$

$$\mu_1 = \phi_1\left(\bar{\mu}_0, \theta_1\right) = \phi_1\left(u, \theta_1\right).$$

$$\mu_j = \phi_j\left(\bar{\mu}_{j-1}, \theta_j\right),\ j = 2, 3,\ \ldots,\ J-1.$$

$$\bar{y} = \mu_J = \phi_J\left(\bar{\mu}_{J-1}, \theta_J\right).$$

$$\bar{y} = \begin{bmatrix} \bar{y}^{(1)} \\ \bar{y}^{(2)} \\ \vdots \\ \bar{y}^{(L)} \end{bmatrix} = \begin{bmatrix} \mu_J^{(1)} \\ \mu_J^{(2)} \\ \vdots \\ \mu_J^{(I_J)} \end{bmatrix} = \phi_J\left(\phi_{J-1}\left(\ldots \phi_1(u,\theta_1), \ldots, \theta_{J-1}\right), \theta_J\right) \overset{\text{df}}{=} \begin{bmatrix} \Phi^{(1)}(u,\theta) \\ \Phi^{(2)}(u,\theta) \\ \vdots \\ \Phi^{(L)}(u,\theta) \end{bmatrix} \overset{\text{df}}{=} \Phi(u,\theta),$$

$$\theta \overset{\text{df}}{=} \{\theta_1, \theta_2, \ldots, \theta_J\}.$$

$$Q(\theta) = \sum_{n=1}^{N} Q_n(\theta) = \sum_{n=1}^{N} [y_n - \Phi(u_n,\theta)]^T [y_n - \Phi(u_n,\theta)] = \sum_{n=1}^{N}\sum_{l=1}^{L}\left(y_n^{(l)} - \Phi^{(l)}(u_n,\theta)\right)^2$$

$$Q_n(\theta) = [y_n - \Phi(u_n,\theta)]^T [y_n - \Phi(u_n,\theta)] = \sum_{l=1}^{L}\left(y_n^{(l)} - \Phi^{(l)}(u_n,\theta)\right)^2 = \sum_{l=1}^{L} \varepsilon_n^{(l)\,2} \qquad \varepsilon_n^{(l)} \overset{\text{df}}{=} y_n^{(l)} - \bar{y}_n^{(l)} = y_n^{(l)} - \Phi^{(l)}(u_n,\theta),\ l = 1, 2, \ldots, L$$

$$\left.\frac{\partial Q_n(\theta)}{\partial \theta_{ij}^{(s)}}\right|_{\theta_{ij}^{(s)} = \tilde{\tilde{\theta}}_{ij,n}^{(s)}} = -2\delta_{jn}^{(i)}\mu_{j-1,n}^{(s)}, \qquad \tilde{\tilde{\theta}}_{ij,n+1}^{(s)} = \tilde{\tilde{\theta}}_{ij,n}^{(s)} - \eta_n \left.\frac{\partial Q_n(\theta)}{\partial \theta_{ij}^{(s)}}\right|_{\theta_{ij}^{(s)} = \tilde{\tilde{\theta}}_{ij,n}^{(s)}},\ s = 0, 1, \ldots, I_{J-1},\ i = 1, 2, \ldots, I_J,\ j = 1, 2, \ldots, J,$$

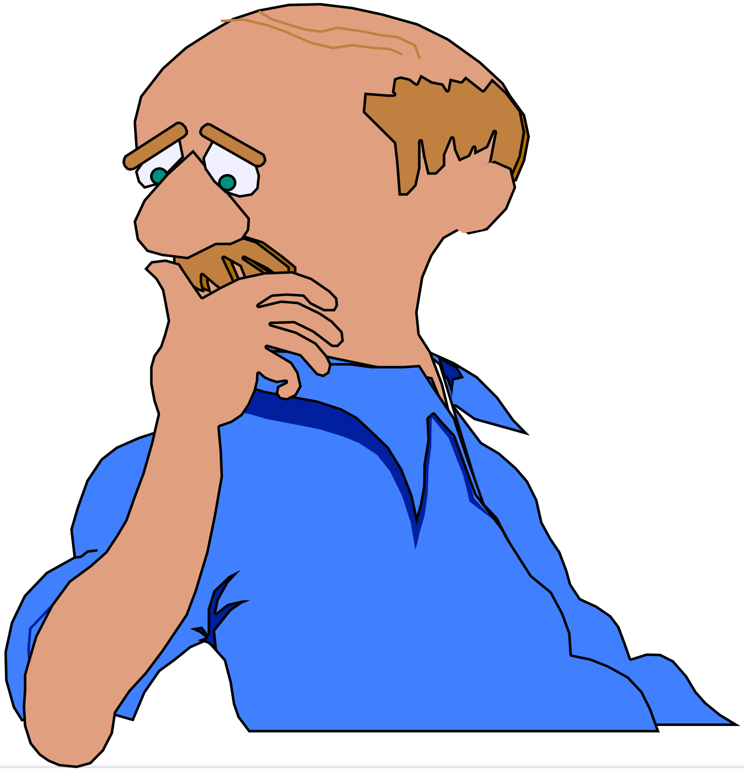
$$\varepsilon_{jn}^{(i)} \overset{\text{df}}{=} \begin{cases} \varepsilon_n^{(i)} & \text{dla } j = J \\ \sum_{p=1}^{I_{j+1}} \delta_{j+1,n}^{(p)} \tilde{\tilde{\theta}}_{pj+1,n}^{(i)} & \text{dla } j = J-1, J-2, \ldots, 1 \end{cases}$$

$$\delta_{jn}^{(i)} \overset{\text{df}}{=} \varepsilon_{jn}^{(i)} \left.\frac{d\phi_{ij}(\kappa_j^{(i)})}{d\kappa_j^{(i)}}\right|_{\theta_{ij}^{(s)} = \tilde{\tilde{\theta}}_{ij,n}^{(s)}}$$

$$\kappa_j^{(i)} \overset{\text{df}}{=} \sum_{s=1}^{I_{j-1}} \mu_{j-1}^{(s)}\theta_{ij}^{(s)} + \theta_{ij}^{(0)}.$$

$$\tilde{\tilde{\theta}}_{ij,n+1}^{(s)} = \tilde{\tilde{\theta}}_{ij,n}^{(s)} + 2\eta_n \delta_{jn}^{(i)}\mu_{j-1,n}^{(s)},\ s = 0, 1, \ldots, I_{J-1},\ i = 1, 2, \ldots, I_J,\ j = 1, 2, \ldots, J.$$

Thank you for attention



HUMAN CAPITAL
HUMAN – BEST INVESTMENT!
Wrocław University of Technology
EUROPEAN SOCIAL FUND
Project co-financed from the EU European Social Fund