Neuronové sítě pro modelování systémů

Ing. Michal Schmidt

E-545

E-mail: xschmi00@stud.feec.vutbr.cz
XMPP: michich@jabber.cz

Obsah

- Neuronové sítě (neural networks - NN)
  - Úvod do NN
  - Metody učení
  - Obecná doporučení pro aplikaci NN
- NN jako modely dynamických systémů
  - NARX model
  - Off-line a on-line učení modelu
  - Výběr trénovacích vzorků

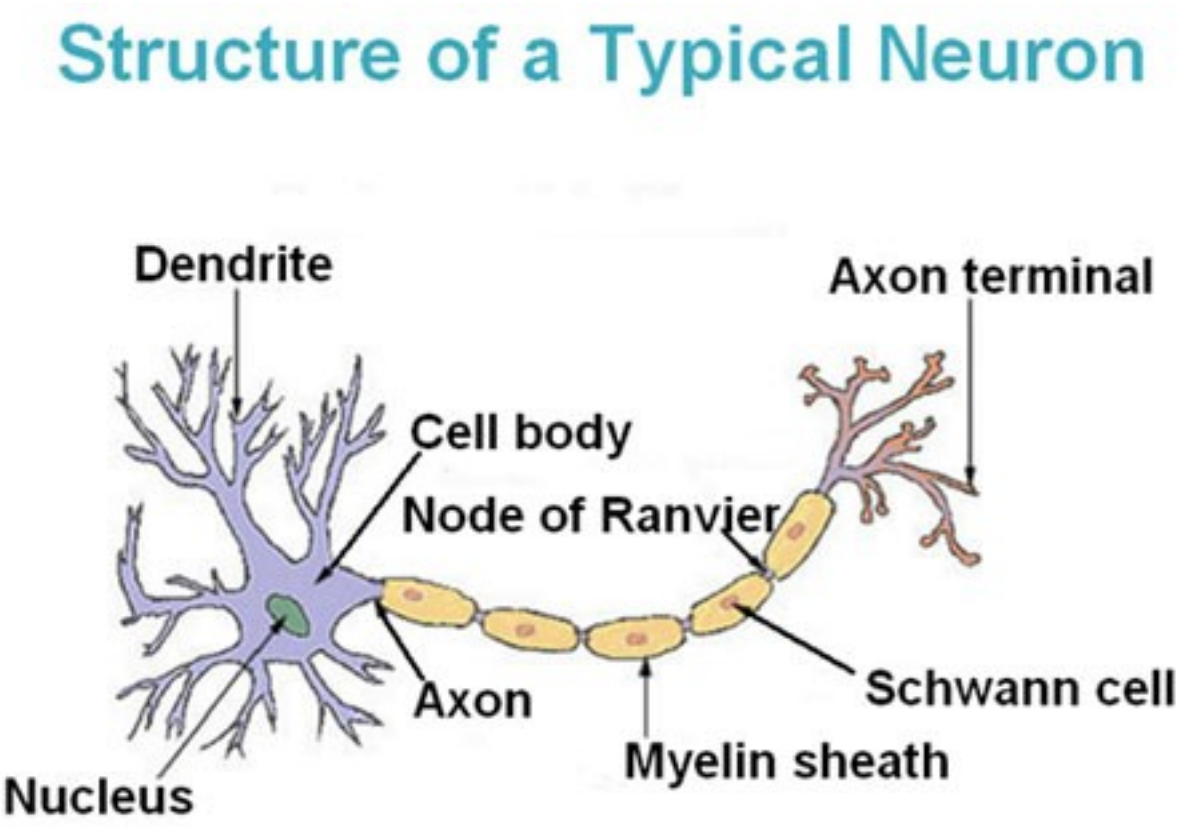
Lidský mozek jako inspirace

- masivně paralelní systém
- vysoký stupeň redundance
- úžasná schopnost adaptivity
- 100 miliard buněk – neuronů
  - každý z nich propojen s 10 000 jinými

Biologický neuron

Biologický neuron

- **soma** (tělo) 4-100 μm v průměru
- strom **dendrit**ů pro vstup informace
- dlouhý, rozvětvený **axon** jako výstup



Synapse

- Elektrická a chemická propojení neuronů
- Napětím řízené iontové kanály
- Synaptická plasticita - původce schopnosti učení a pamatování

Modelování neuronu

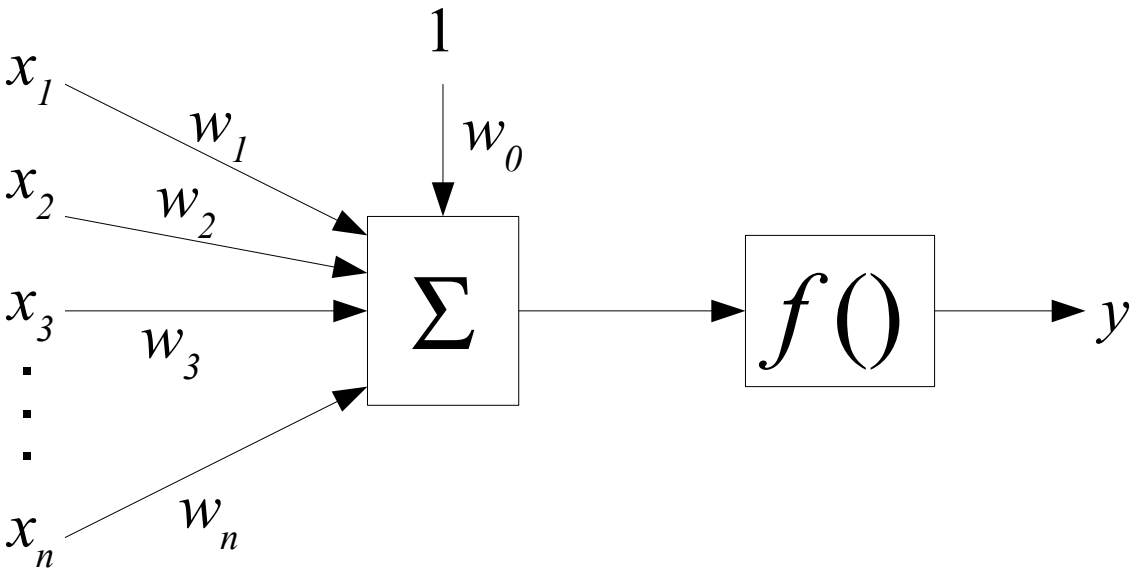
- Dnes už jsou vypracovány velmi detailní matematické modely biologických neuronů.
- Pro využití v oblastech umělé inteligence jsou však takové modely nepraktické.
- Používáme mnohem jednodušší modely.

Umělé neuronové sítě

- Umělé neurony propojené do většího celku pro zpracování informace
- Mnoho typů NN:
  - dopředné sítě
    - perceptron
    - **vícevrstvý perceptron**
    - ADALINE
    - Kohonenovy sítě
    - RBF
  - rekurentní sítě
    - simple recurrent network
    - Hopfieldovy sítě
  - stochastické sítě
    - Boltzmannův stroj
  - ...

Umělý neuron
(perceptronového typu)

- Vážený součet vstupů...
- ...přes obecně nelineární aktivační funkci



$$y = f\left(\mathbf{w}^{\mathrm{T}} \cdot \mathbf{x} + w_0\right)$$

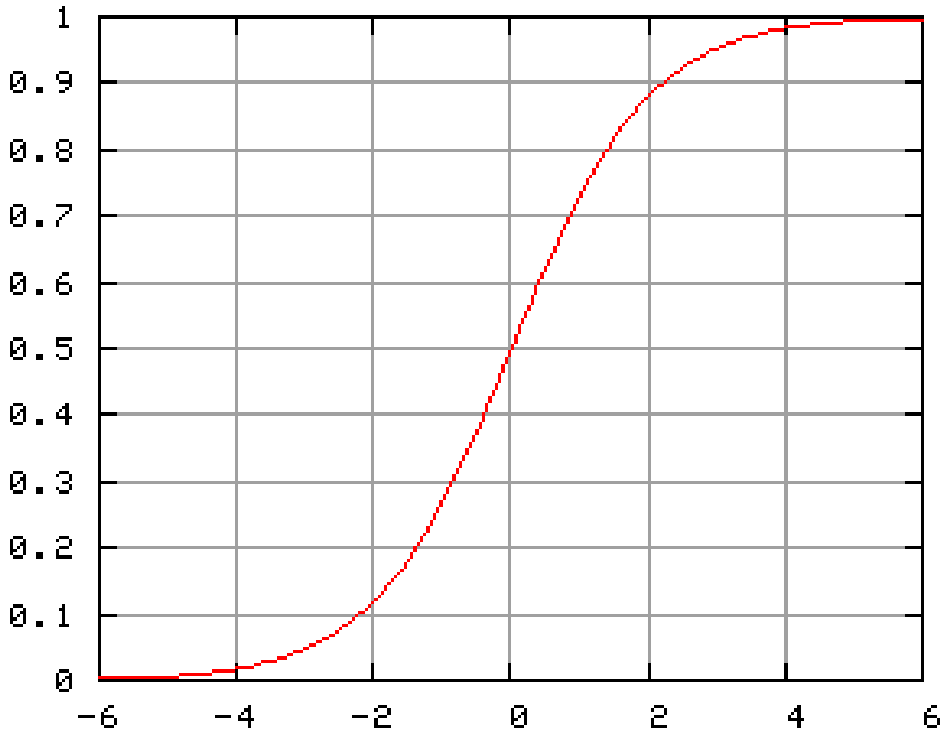
Aktivační funkce

- lineární funkce
- prahová funkce (step function)
- sigmoida (zvláštní případ logistické funkce)

$$f(\xi)=\frac{1}{1+e^{-\xi}}$$

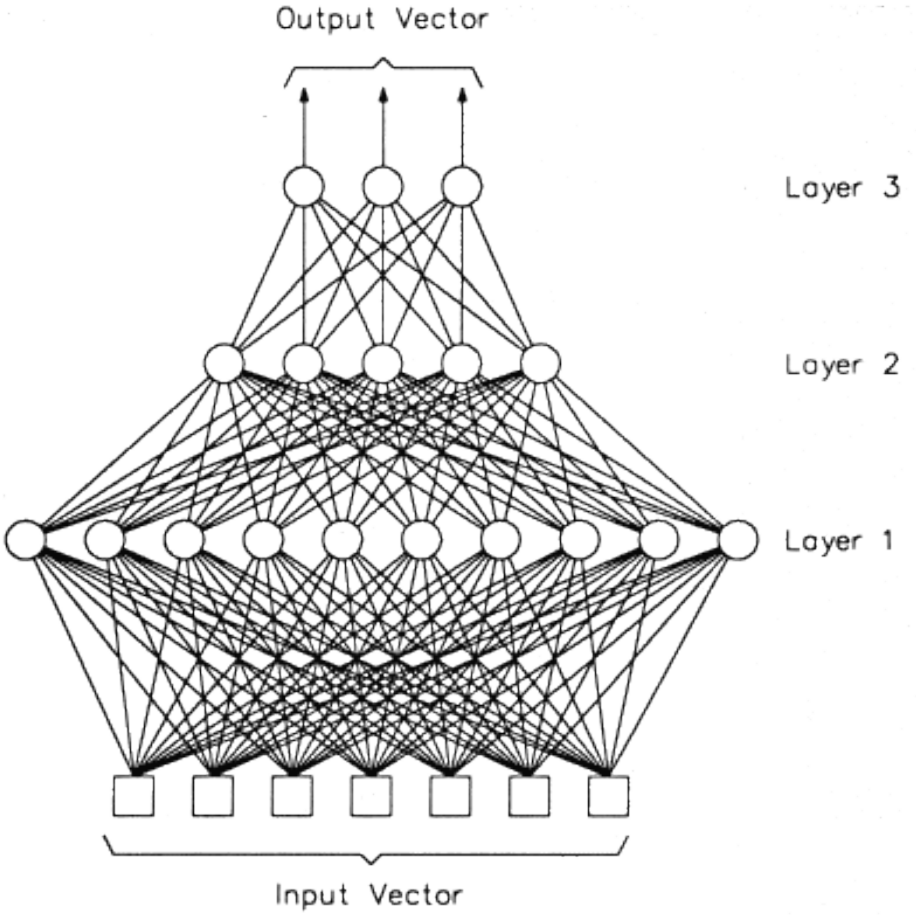
jednoduše diferencovatelná:

$$f'(\xi)=f(\xi)\left(1-f(\xi)\right)$$



Vícevrstvá dopředná síť
Multi-layered perceptron

- označovaná též jako „feed-forward net“, „back-propagation net“
- v pracovním režimu se informace šíří jedním směrem přes jednotlivé vrstvy



$$\mathbf{y}=\mathbf{F}(\mathbf{x})=\mathbf{f}_3(\mathbf{W}_3\cdot\mathbf{f}_2(\mathbf{W}_2\cdot\mathbf{f}_1(\mathbf{W}_1\cdot\mathbf{x}+\mathbf{w}_{0,1})+\mathbf{w}_{0,2})+\mathbf{w}_{0,3})$$

Vícevrstvá dopředná síť

- zřejmě nejpoužívanější umělá neuronová síť
- Má-li aspoň jednu skrytou vrstvu a dostatečný počet neuronů, dokáže aproximovat jakoukoliv vícerozměrnou funkci s libovolnou přesností.
  (vyplývá z Kolmogorovova teorému)
- Objevení efektivní metody učení v 80. letech vedlo k obnovení zájmu o neuronové sítě.

Učení sítě

- Víme, že síť dokáže aproximovat libovolnou funkci.
- Ale jak dosáhnout toho, aby realizovala funkci, kterou chceme?
- Znalost sítě je uchována v hodnotách vah.
- Musíme nějak rozumně váhy nastavit.

Trénovací množina

- Při učení síti předkládáme učící **vzory**.
- Každý vzor se skládá z ukázkového vstupu a odpovídajícího požadovaného výstupu.

- Takže trénovací množina je:

$$M = \{(\mathbf{x}_1, \mathbf{z}_1), (\mathbf{x}_2, \mathbf{z}_2), \ldots, (\mathbf{x}_m, \mathbf{z}_m)\}$$

Kritérium učení

- Výstup nenaučené sítě se od požadovaného výstupu liší – pro každý vzor dostaneme na výstupu nějaký chybový vektor.
- Chceme všechny chybové vektory co nejmenší.
- Minimalizujeme kritérium (chybovou funkci)

$$E = \frac{1}{2} \sum_{i=1}^{m} \|\mathbf{y}_i - \mathbf{z}_i\|^2$$

(součet kvadrátů velikostí chybových vektorů pro všechny vzory)

Učení sítě je vícerozměrný minimalizační problém

- Pro danou trénovací množinu a strukturu sítě minimalizujeme kritérium.

$$E = \frac{1}{2} \sum_{i=1}^{m} \|\mathbf{y}_i - \mathbf{z}_i\|^2$$

- Měníme váhy, těch je mnoho a každá ovlivňuje výsledek.
  Proto vícerozměrný problém.
- Pro netriviální sítě neexistuje analytické řešení.
- Používají se iterační metody, většinou využívají principu sestupu po gradientu.

Back-propagation (zpětné šíření chyby)

- Princip svedení viny na podřízené
  - Předložíme síti trénovací vzor.
  - Srovnáme skutečný výstup sítě s požadovaným.
  - Každý neuron si spočítá, o kolik menší nebo větší jeho výstup měl být, aby odpovídal požadovanému. Zná tak svou lokální chybu.
  - Neuron obviní za vznik své lokální chyby svoje kolegy z předchozí vrstvy. Přiřadí jim „vinu“ tak, že největší podíl na ní dostanou ti, kteří jsou s ním spojeni silnější vahou.
  - Obviněné neurony dál postupují úplně stejně. Jako svou lokální chybu vezmou vinu, která jim byla přiřazena.
  - Když už není koho obvinit, každý neuron upraví své vstupní váhy, aby snížil lokální chybu.

Back-propagation

- Chyba se tedy počítá opačným směrem, než jak prochází informace v pracovním režimu.
- Aktualizace vah:

$$w_{ij}(t+1) = w_{ij}(t) - \eta \delta_j f'(\xi_j) y_i$$

- η je **učící konstanta**
- Někdy se zavádí **momentum** (setrvačnost) pro zamezení oscilací v lokálních minimech.

$$w_{ij}(t+1) = w_{ij}(t) - \eta \delta_j f'(\xi_j) y_i + \mu (w_{ij}(t) - w_{ij}(t-1))$$

Metoda Levenberg-Marquardt

- Optimalizuje váhy pro všechny vzory najednou.
- Mějme vektor $\mathbf{w}$ složený ze všech vah v síti.
- Mějme $\mathbf{E}$ jako dlouhý chybový vektor slepený z jednotlivých chybových vektorů $\mathbf{y}-\mathbf{z}$ pro všechny vzory.
- Minimalizujeme kritérium: $E = \frac{1}{2}\|\mathbf{E}\|^2$

Metoda Levenberg-Marquardt

- Jacobiho matice:

$$\mathbf{J} = \frac{\mathrm{d}\,\mathbf{E}}{\mathrm{d}\,\mathbf{w}} = \begin{pmatrix} \frac{\partial y_1}{\partial w_1}\Big|_{x_1} & \cdots & \frac{\partial y_1}{\partial w_p}\Big|_{x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_k}{\partial w_1}\Big|_{x_1} & \cdots & \frac{\partial y_k}{\partial w_p}\Big|_{x_1} \\ \frac{\partial y_1}{\partial w_1}\Big|_{x_2} & \cdots & \frac{\partial y_1}{\partial w_p}\Big|_{x_2} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_k}{\partial w_1}\Big|_{x_m} & \cdots & \frac{\partial y_k}{\partial w_p}\Big|_{x_m} \end{pmatrix}$$

$\frac{\partial y_i}{\partial w_j}\Big|_{x_r}$ je derivace i-tého výstupu podle j-té váhy pro r-tý vzor

Metoda Levenberg-Marquardt

- Můžeme odhadnout, jak se bude vyvíjet chybový vektor, když trošku změníme váhy:

$$\mathbf{E}(i+1)=\mathbf{E}(i)+\mathbf{J}(\mathbf{w}(i+1)-\mathbf{w}(i))$$

Takže i novou hodnotu kritéria:

$$E(i+1)=\frac{1}{2}\|\mathbf{E}(i)+\mathbf{J}(\mathbf{w}(i+1)-\mathbf{w}(i))\|^2$$

- Analytickým řešením nalezneme novou hodnotu vah, pro kterou očekáváme minimum kriteriální funkce...

Metoda Levenberg-Marquardt

- Dá se odvodit, že odhadované kritérium bude minimalizovat tato hodnota vah:

$$\mathbf{w}(i+1)=\mathbf{w}(i)-(\mathbf{J}^{\mathrm{T}}\mathbf{J})^{-1}\mathbf{J}^{\mathrm{T}}\mathbf{E}$$

  To odpovídá Gauss-Newtonově metodě.

- L-M zavádí jednu modifikaci:

$$\mathbf{w}(i+1)=\mathbf{w}(i)-(\lambda\mathbf{I}+\mathbf{J}^{\mathrm{T}}\mathbf{J})^{-1}\mathbf{J}^{\mathrm{T}}\mathbf{E}$$

  tlumící faktor je proměnný a přibližuje metodu buď ke G-N nebo k sestupu po gradientu.

Metoda Levenberg-Marquardt

- Je robustnější než Gauss-Newtonova.
- Ve srovnání s back-propagation
  - pracuje s velkými maticemi – paměťově náročná
  - jedna iterace trvá déle
  - ale konverguje velmi rychle, obyčejně v jednotkách iterací.

Obecná doporučení pro aplikaci NN

- Struktura sítě
- Dávkové učení
- Normalizace
- Problém přetrénování

Struktura sítě

- Obecně platí, že méně je více.
- Počet vrstev
  - aspoň jednu skrytou, jestli chceme zachovat univerzální aproximační vlastnost
  - pokud to na problém stačí, nepřidáváme zbytečně další vrstvy navíc.
- Počet neuronů ve vrstvách
  - vstupní a výstupní jsou celkem jasné
  - ve skrytých vrstvách máme možnost volby
  - dáme příliš málo => nepůjde naučit
  - dáme moc => nebude umět generalizovat

Dávkové učení

- Má smysl o něm uvažovat při použití back-propagation.
- Inkrementální učení:
  - Váhy měníme po každém předložení vzoru.
- Dávkové učení
  - Změny vah střádáme a aplikujeme je najednou až po předložení všech vzorů.
  - Ukazuje se jako lepší strategie.

Normalizace

- Neurony jsou nelineární prvky.
- Neuron se sigmoidální aktivační funkcí nikdy nedá na výstupu hodnotu mimo interval (0;1)
- Měli bychom je provozovat v rozumných pracovních bodech (okolí nuly).
- Proto hodnoty fyzikálních veličin na vstupu do sítě normalizujeme a na výstupu zase denormalizujeme.

Problém přetrénování

- Problém:
  - Síť se perfektně nabifluje vzory z trénovací množiny.
  - Přitom žalostně selhává pro jiné vstupy – nemá schopnost generalizace.
  - Znamená to, že došlo k přetrénování.
- Obrana:
  - Ubrat neurony
  - Včas zastavit učení
  - Přidávat šum do trénovacích vzorů
  - Použití testovací množiny (cross-validation)

Neuronové sítě - shrnutí

- Univerzální aproximátory
- Schopnost generalizace
- Učení a adaptace
- Použitelné i tam, kde závislost mezi vstupy a výstupy není algoritmicky známá.
- Aplikace NN: klasifikace, rozpoznávání vzorů a sekvencí, sekvenční rozhodování, filtrace, shluková analýza, komprese, lékařská diagnostika, regresní analýza, časová predikce, **modelování**, **identifikace**, **řízení**, ...

Modelování systémů

- Síť typu vícevrstvý perceptron je univerzální aproximátor vícerozměrné funkce.
- Jenže výsledek funkce vždy závisí pouze na hodnotách vstupů – neuronová síť není dynamický člen.
- Může být vůbec užitečná pro modelování dynamických systémů?

Lineární ARX model

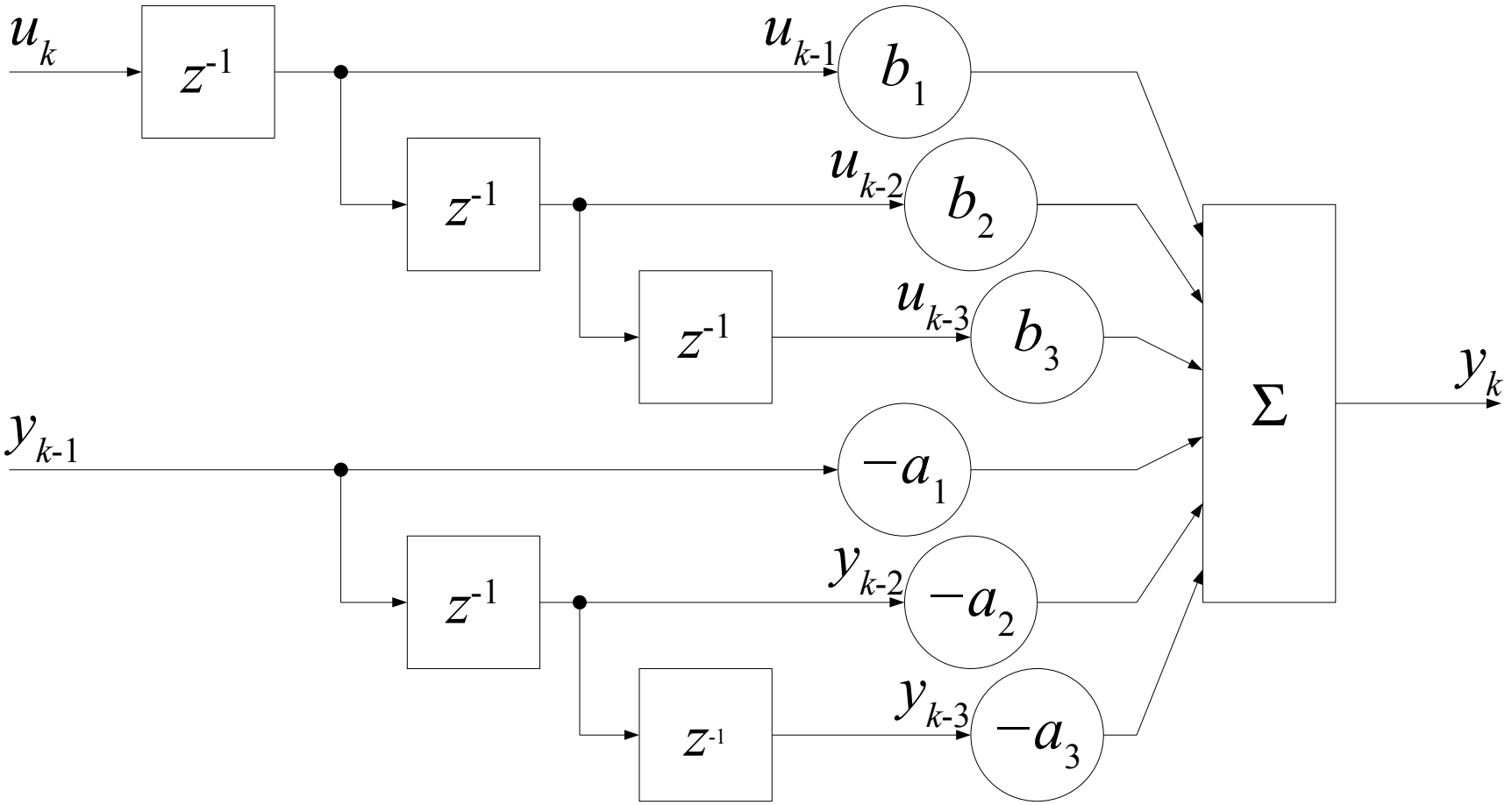
- je dán rovnicí $A(z^{-1})\,y_k = B(z^{-1})\,u_k + e_k$
- např. pro 3.řád:

$$y_k = -a_1 y_{k-1} - a_2 y_{k-2} - a_3 y_{k-3} + b_1 u_{k-1} + b_2 u_{k-2} + b_3 u_{k-3} + e_k$$

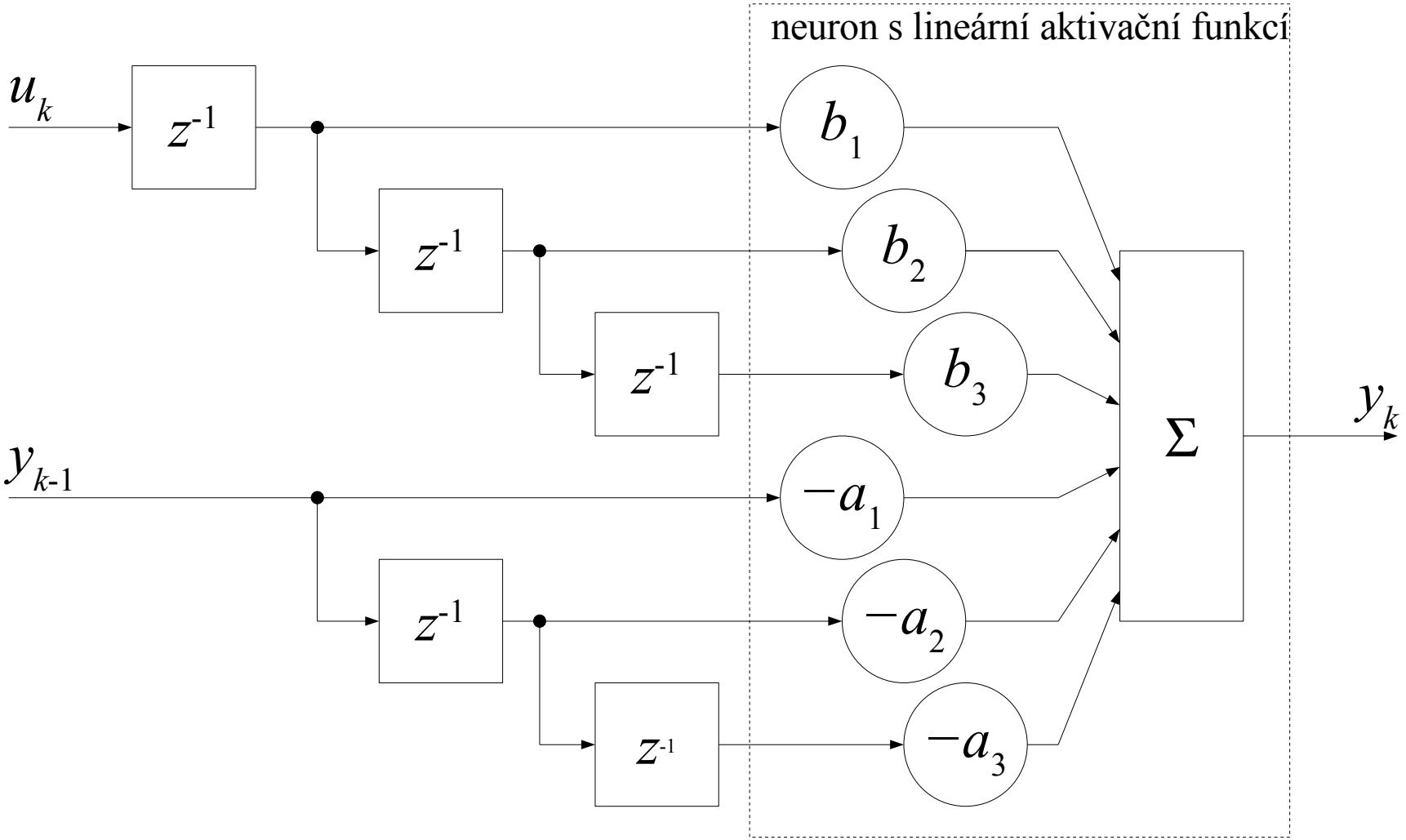
- přenos modelu:

$$F_m(z) = \frac{b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3}}{1 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3}}$$

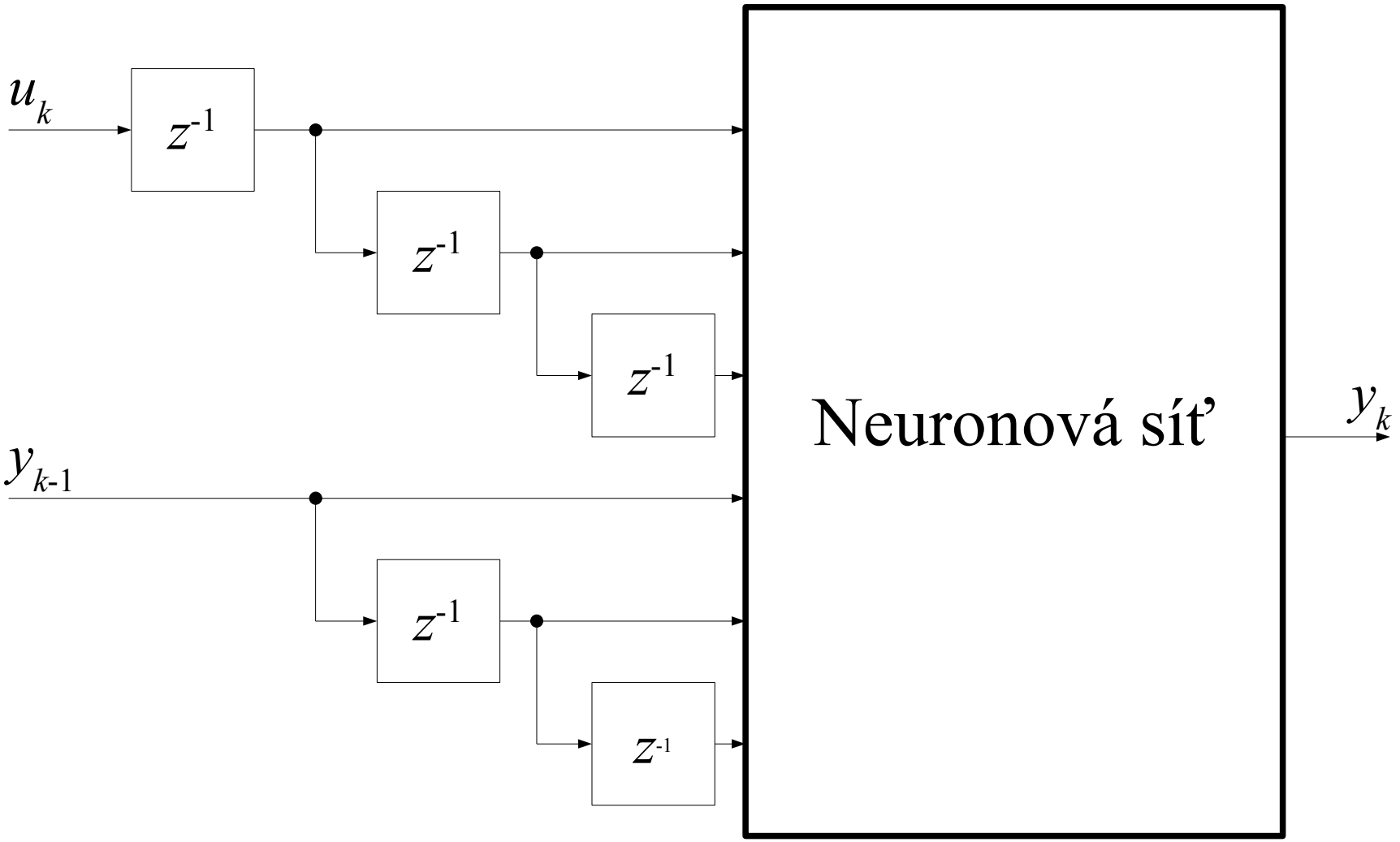
Lineární ARX model



Totéž s neuronovou „sítí“



Nelineární model NARX



Nelineární model NARX

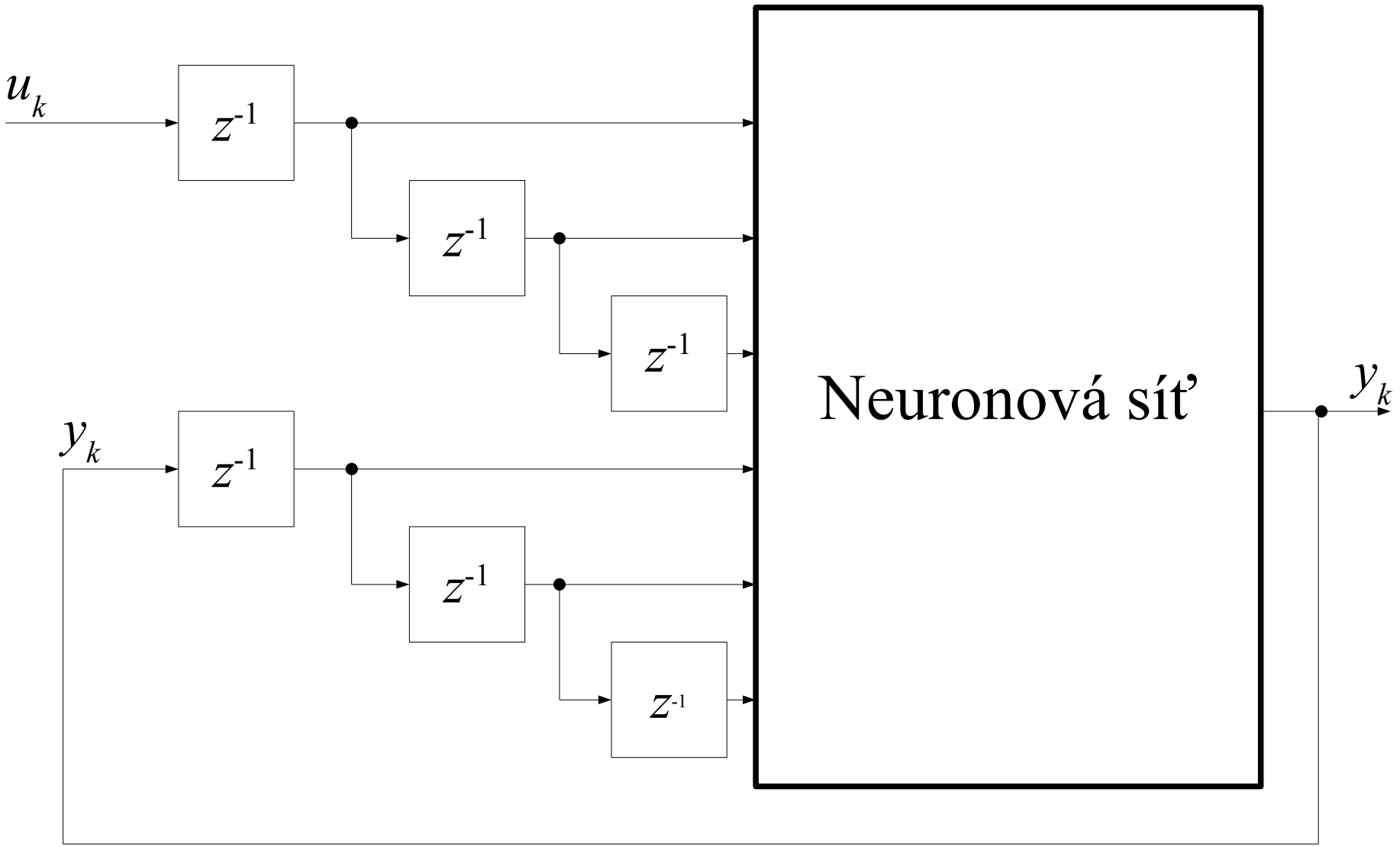
- např. pro 3.řád:

$$y_k = f_{NN}(y_{k-1}, y_{k-2}, y_{k-3}, u_{k-1}, u_{k-2}, u_{k-3})$$

- přenosová funkce:

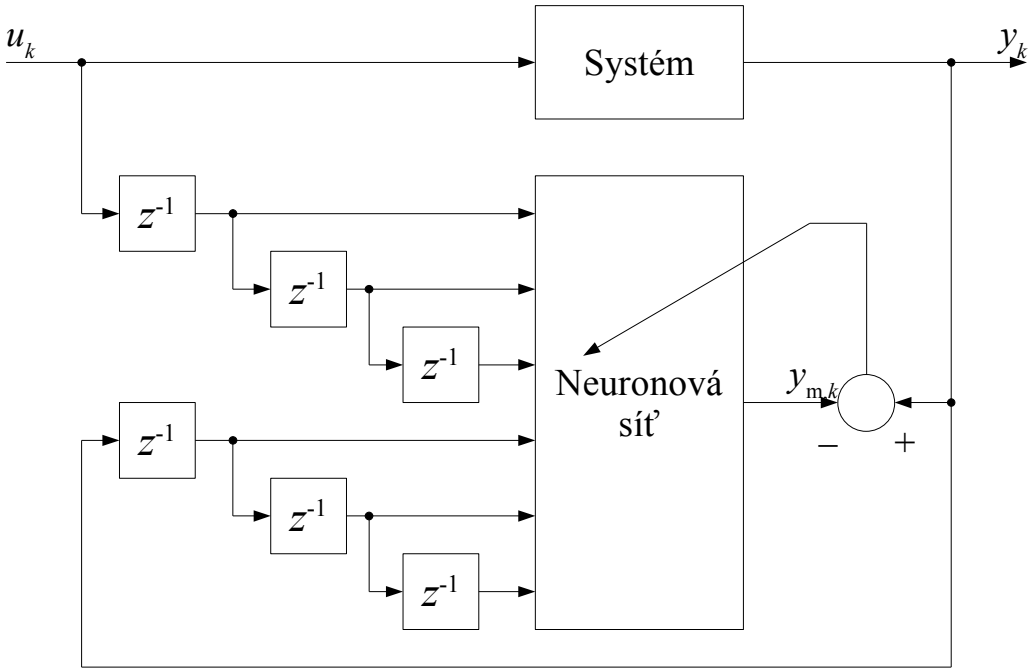
?

Vícekrokový prediktor



Identifikace soustavy

- off-line
  - naměříme data, později je zpracujeme
- on-line
  - identifikujeme průběžně i během regulace



On-line identifikace

- Výrazně obtížnější
- Probíhá současně s regulací, takže systém přitom nelze budit pestrým signálem.
- Trénovací množina se plní nezajímavými vzorky...

Výběr trénovacích vzorků

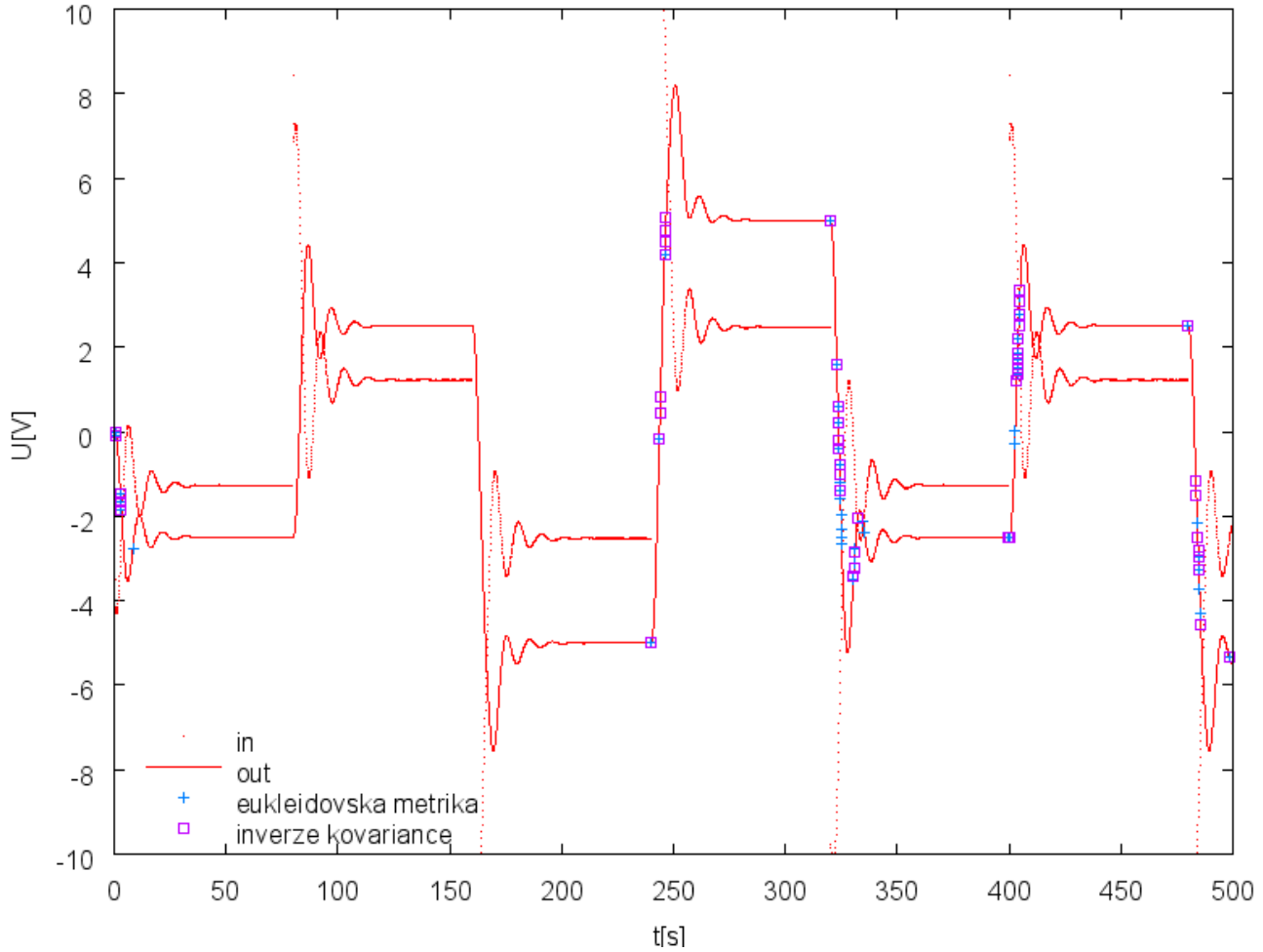
- Nejjednodušší přístup – posledních N vzorků
  - V ustáleném stavu se brzy úplně vytratí informace o dynamice soustavy.
- Detekce ustálených stavů a zastavení plnění
- Použití kritéria „zajímavosti“ vzorku

Výběr trénovacích vzorků

- vzorek chování systému: $\mathbf{v}_k = (\boldsymbol{\varphi}_{k-1}, y_k)$
- lze ukládat jen konečný počet vzorků, proto:
  - nalezneme dva sobě „nejpodobnější“ vzorky
  - odstraníme starší z nich
  - zařadíme nový vzorek
- podobnost definována eukleidovskou normou v transformovaném prostoru: $\boldsymbol{\varphi}' = \mathbf{R}\boldsymbol{\varphi}$

  $\mathbf{Q} = \mathbf{R}^{\mathrm{T}}\mathbf{R}$
- vzdálenost dvou vzorků: $r_{\mathbf{Q}}(\mathbf{v}_i, \mathbf{v}_j) = \|\boldsymbol{\varphi}_i - \boldsymbol{\varphi}_j\|_{\mathbf{Q}}$

Výběr trénovacích vzorků - graf



VYCHODIL, H. *Geometrické vlastnosti on-line identifikace*. Disertační práce. Brno: ÚAMT FEKT VUT, 2005.

Využití získaného modelu

- Model s on-line identifikací můžeme použít pro syntézu adaptivního regulátoru.
  - model se specifickou strukturou – jednotlivé váhy sítě mají předem daný fyzikální význam => přímá syntéza parametrů regulátoru
    - např. z modelu zjistíme kritické parametry a nastavíme PSD regulátor Ziegler-Nichols metodou
  - model typu black-box =>
    - lze linearizovat a použít pro prediktivní regulátor, nebo
    - nelineární optimalizace akčního zásahu

Závěr

- Výhody neuronových sítí v modelech
  - možnost on-line identifikace s výběrem vzorků
  - umí zachytit nelinearity soustavy
  - výzkumy naznačují možnost zkrácení periody vzorkování ve srovnání s klasickými přístupy