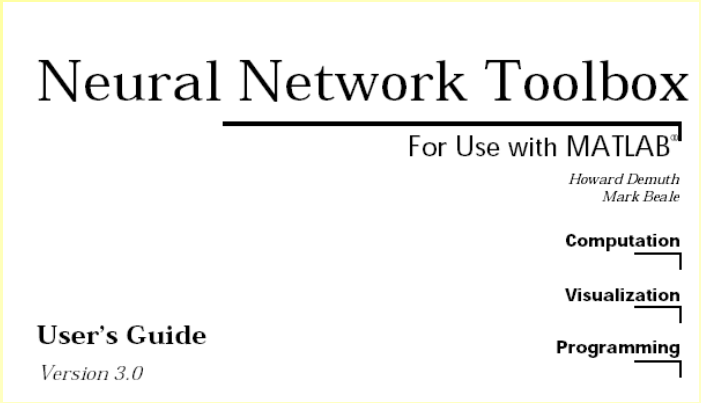
Neuronové sítě (Úvod a MLP sítě)

Neuronové sítě (Úvod a MLP sítě)

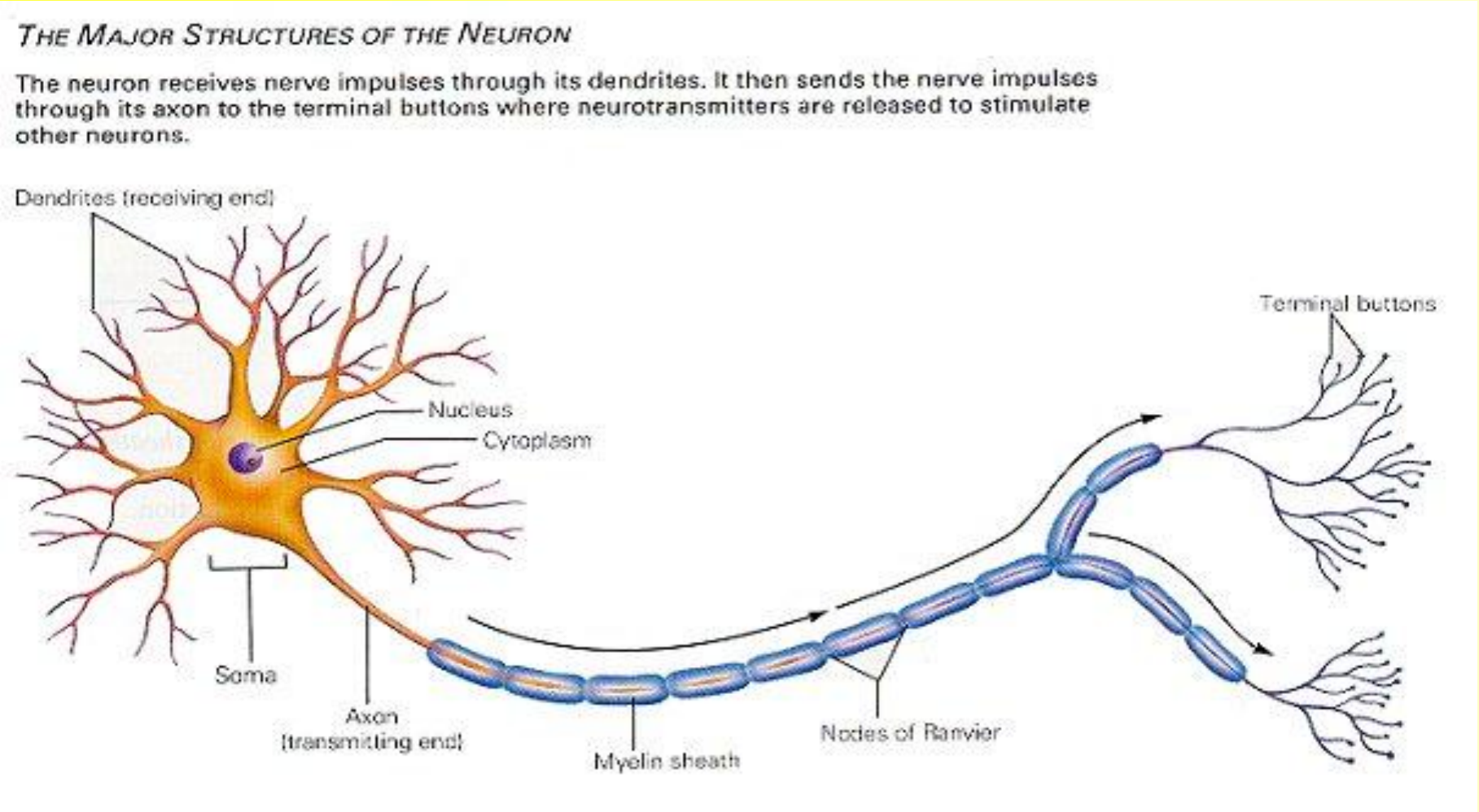
Použité zdroje

- **Prof. Ing. Jiří Bíla, DrSc.: Umělá inteligence a neuronové sítě v aplikacích, FS ČVUT,1998**
  - *Manuál k NN toolboxu pro Matlab v AJ*

http://www-ccs.ucsd.edu/matlab/pdf_doc/nnet/nnet.pdf

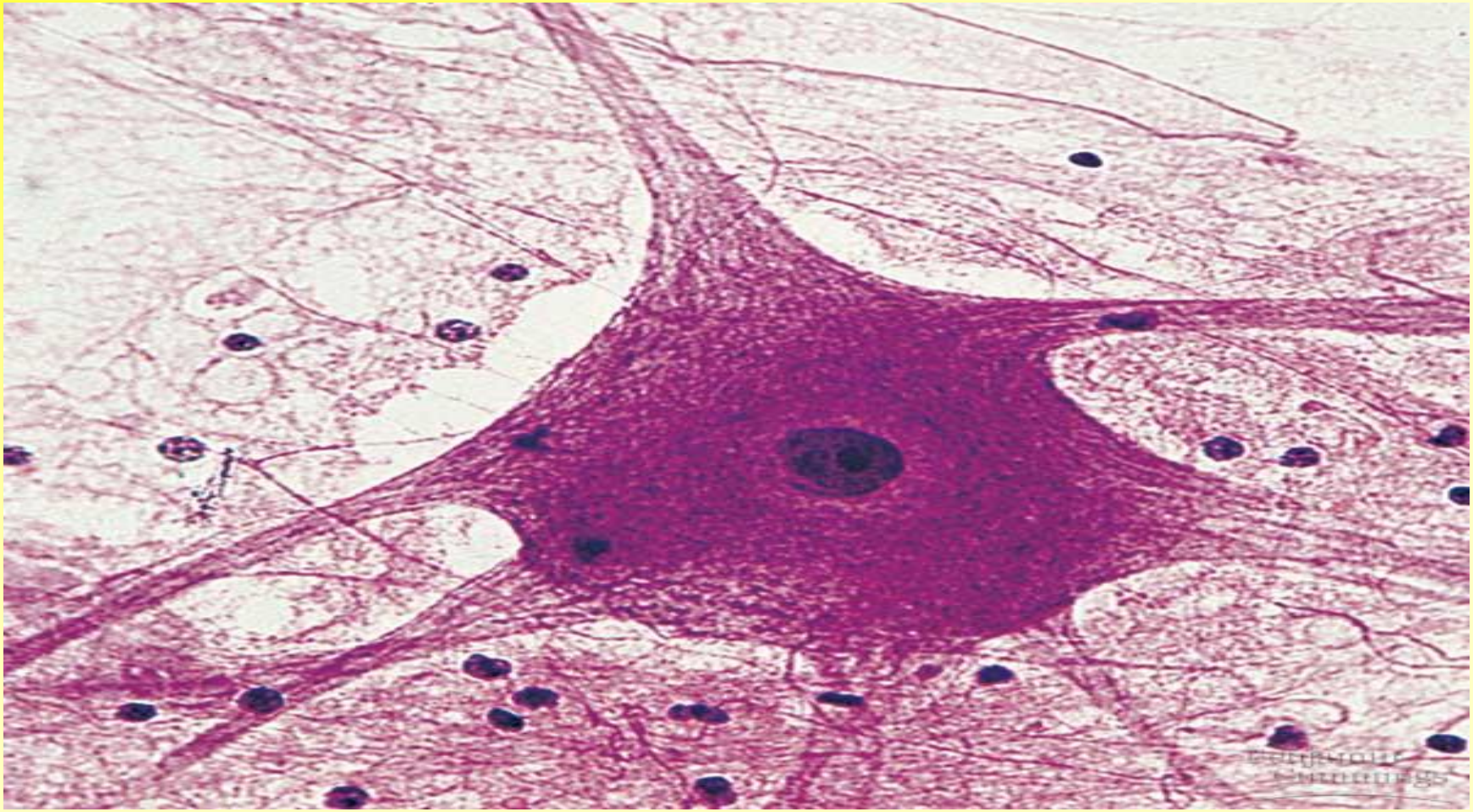


Biologický Neuron



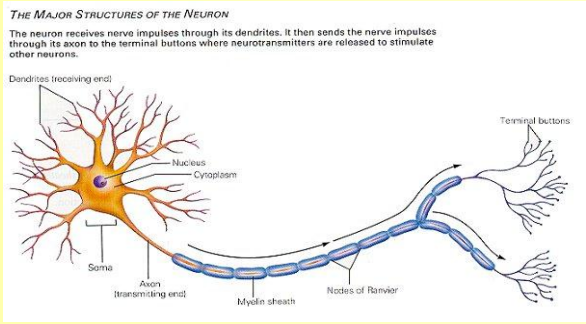
zdroj“: http://inside.salve.edu/walsh/neuron.jpg
(www.google.com) , 04.05.2005

Biologický neuron

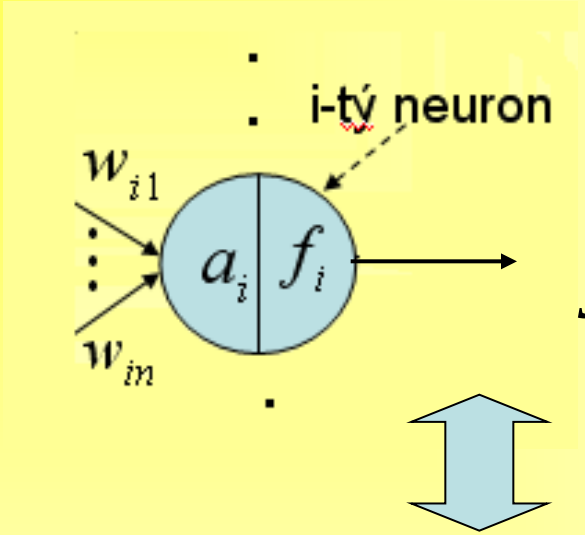


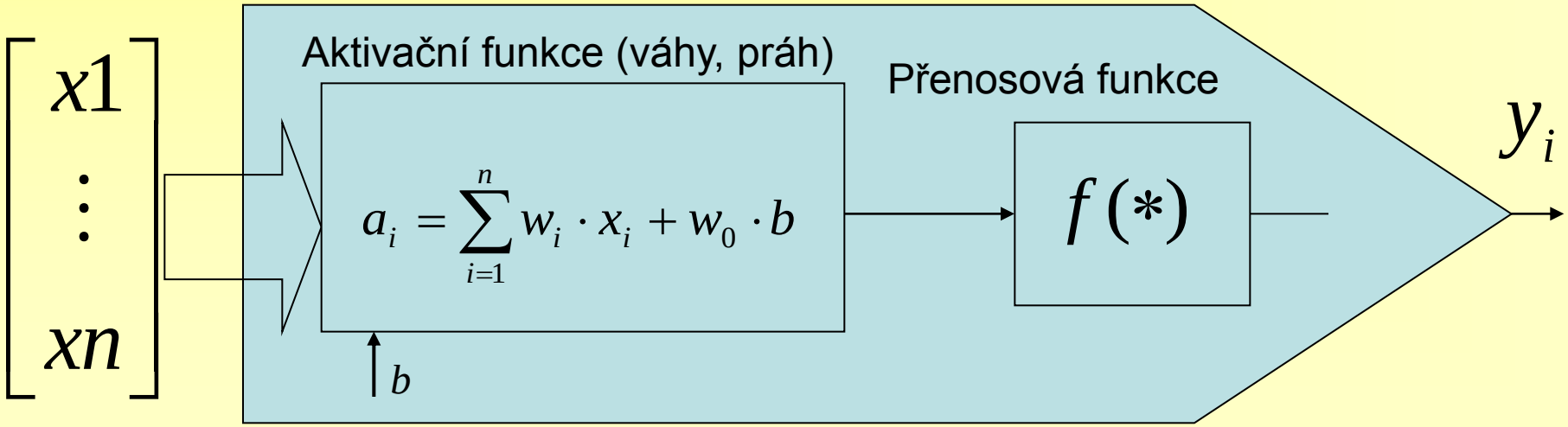
Zdroj: http://35.9.122.184/images/40-AnimalStructureAndFunction/40-03-Neuron.jpg (www.google.com) 04.05.2005

i-tý UMĚLÝ NEURON



$$\begin{bmatrix} x1 \\ \vdots \\ xn \end{bmatrix}$$

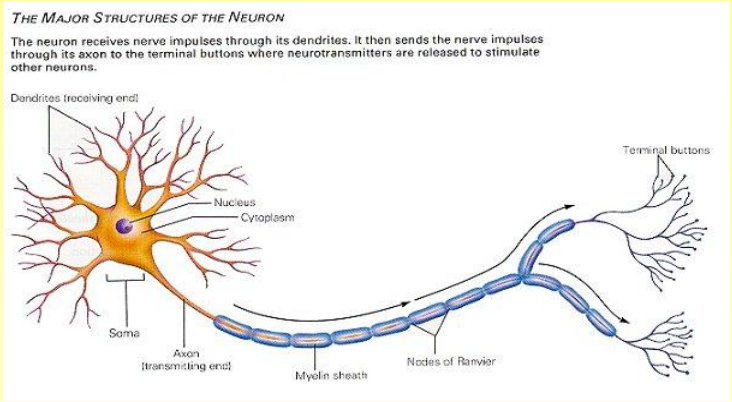


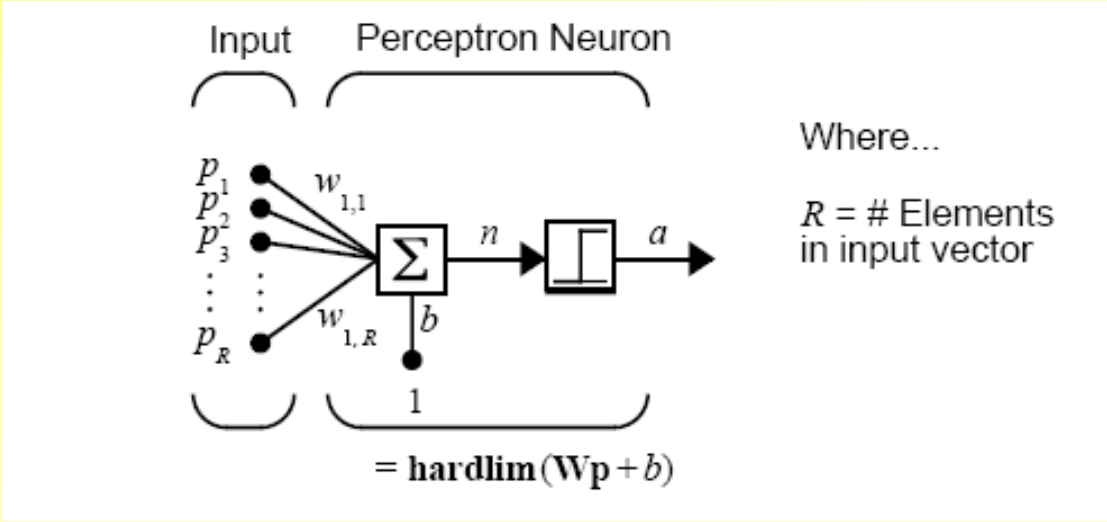


První modely neuronu

1943 McCullock a Pitts: první model

1958 Rosenblatt: Prahová logika, neuron typu „Perceptron“

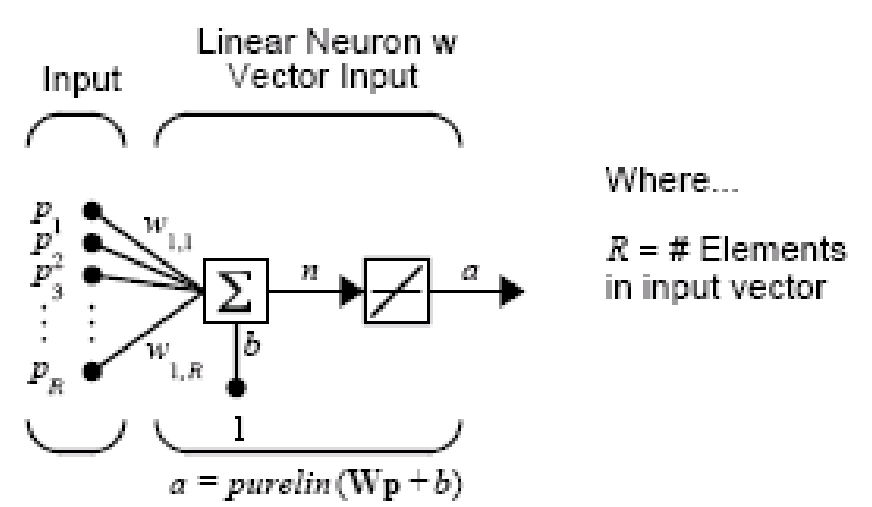




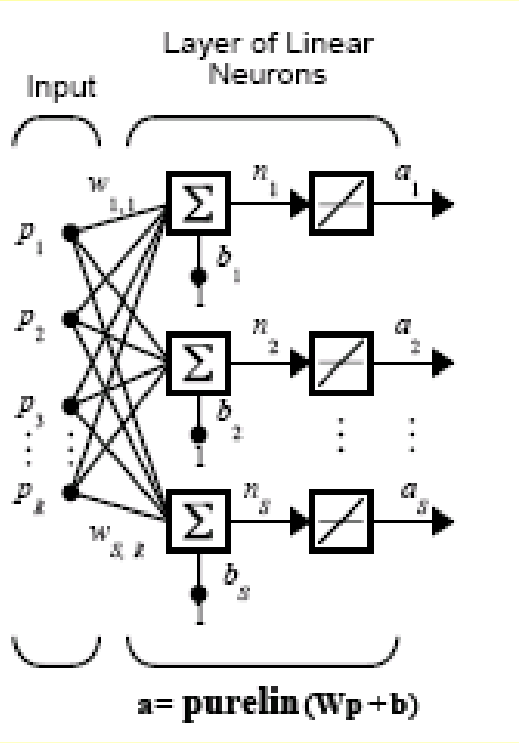
První modely neuronu

- 1960 ADALINE (Widrow, Hoff) , ADAptivní LIneární NEuron

ADALINE



MADALINE=Many ADALINE



Využití prvních modelů umělých neuronů

- Rozpoznávání příznaků (perceptron)
- Lineární klasifikátor ((M)ADALINE)

Diskriminant, diskriminační funkce

Typické úlohy pro neuronové sítě

- Modelování, Identifikace a Analýza systémů
- Predikce signálů
- Klasifikace
- Rozpoznávání obrazců
- Detekce poruch

Např.: diagnostika fraktury z rentgenového snímku, rozpoznávání písma, rukopisu, některé programy pro rozpoznávání řeči,…

Co je třeba (také) vědět pokud navrhujeme neuronovou síť

- Je třeba vědět jaký typ sítě je pro danou aplikaci vhodný, typ přenosové funkce *f* (*), zda se jedná o klasifikaci nebo identifikaci, adaptabilní řízení, a v neposlední řadě zda řešíme ‘lineární’ nebo nelineární problém, a další …

Rozdělení neuronových sítí

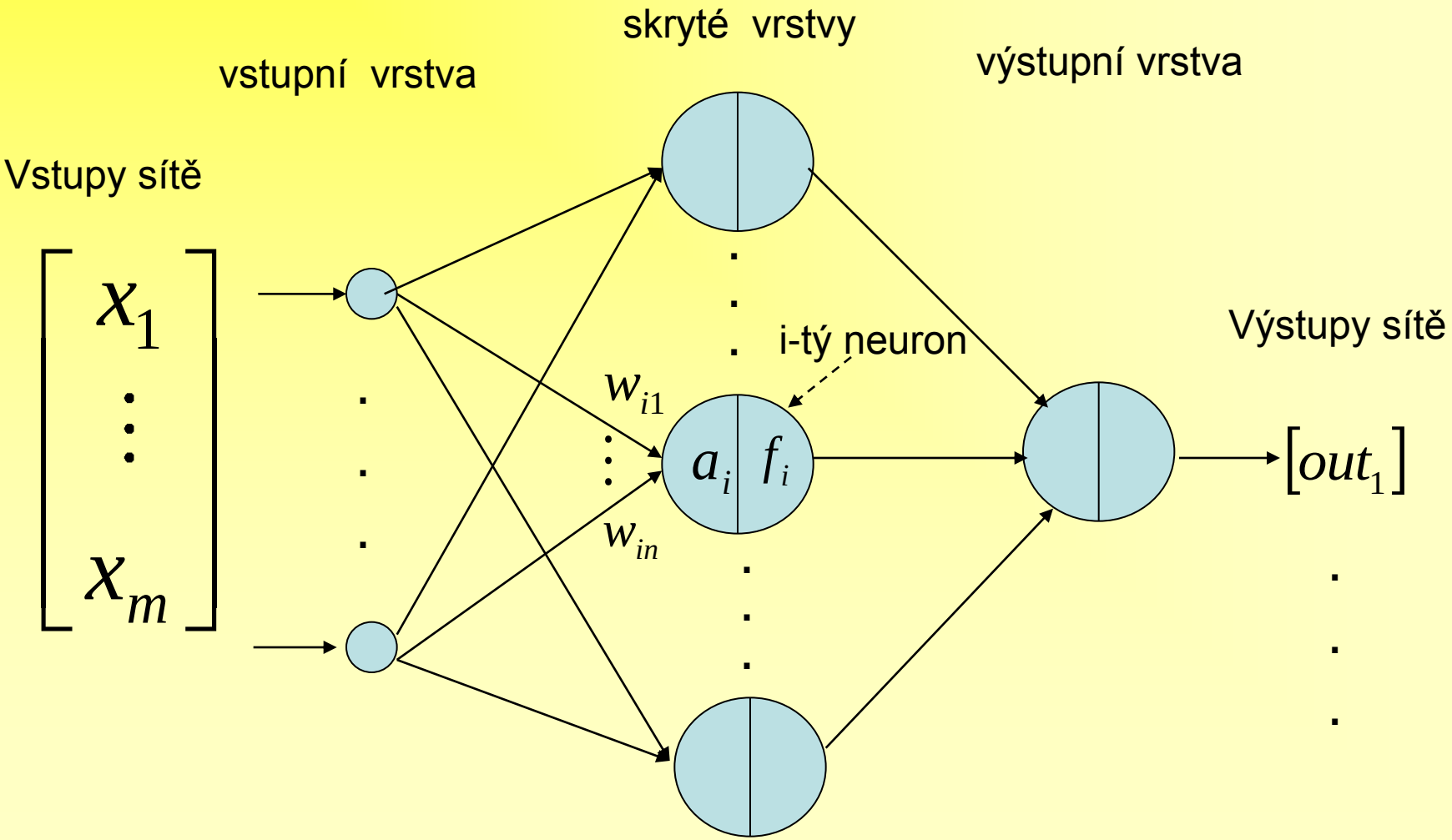
Vzhledem k architektuře sítě

- **VÍCEVRSTVÉ PERCEPTRONOVÉ SÍTĚ (SÍTĚ TYPU MLP)**
- **Sítě s radiální bází (sítě typu RBF)**
- Dále dynamické neuronové architektury,jako různé typy rekurentních neuronových sítí, Hoppfieldova síť, …mnohem více

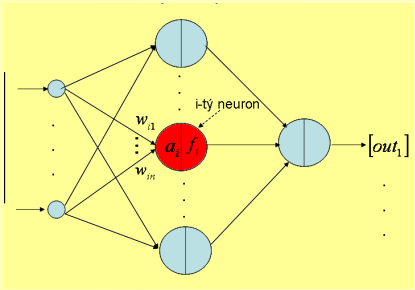
Jednotlivé architektury vyžadují příslušný algoritmus učení !

Neuronové sítě typu MLP:

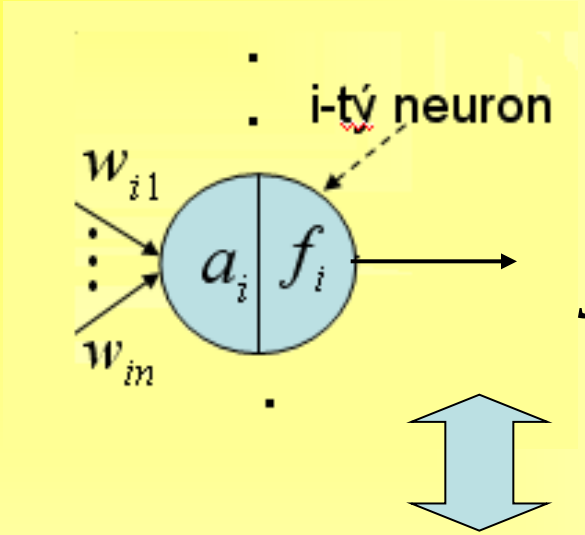
MULTILAYER FEED-FORWARD NN (FFNN)

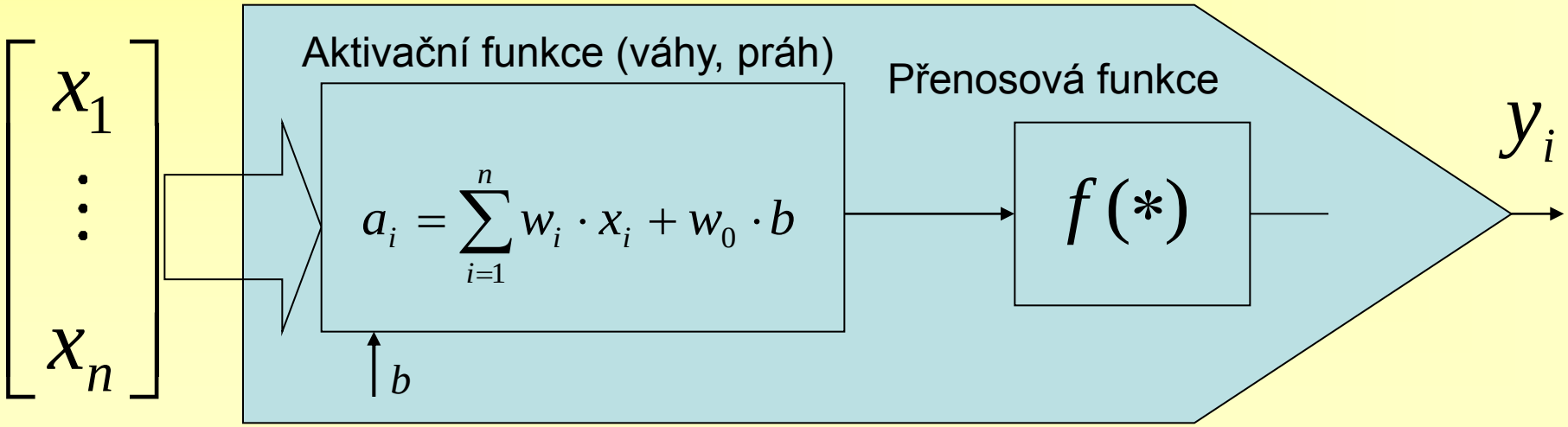


i-tý UMĚLÝ NEURON

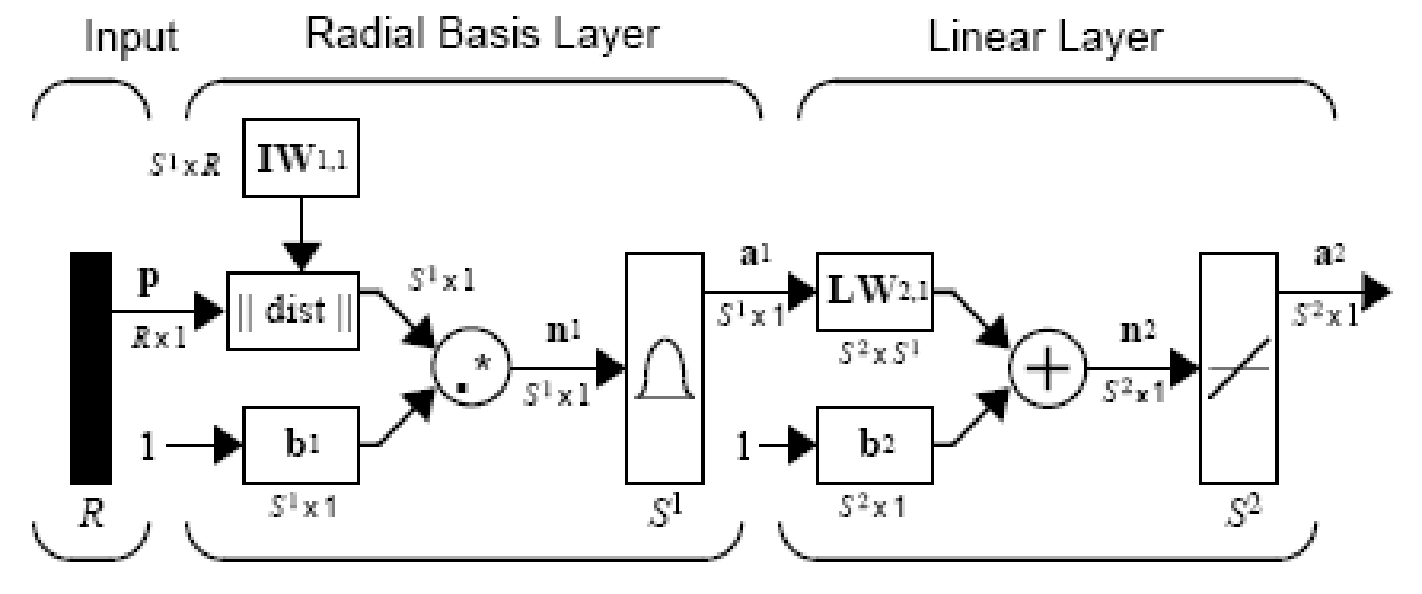


$$\begin{bmatrix} x1 \\ \vdots \\ xn \end{bmatrix}$$





Neuronové sítě s radiální bází



- Centra se zvolí nebo automaticky navrhnou
- Učení spočívá v interpolaci mulidimenzionální funkce S^i

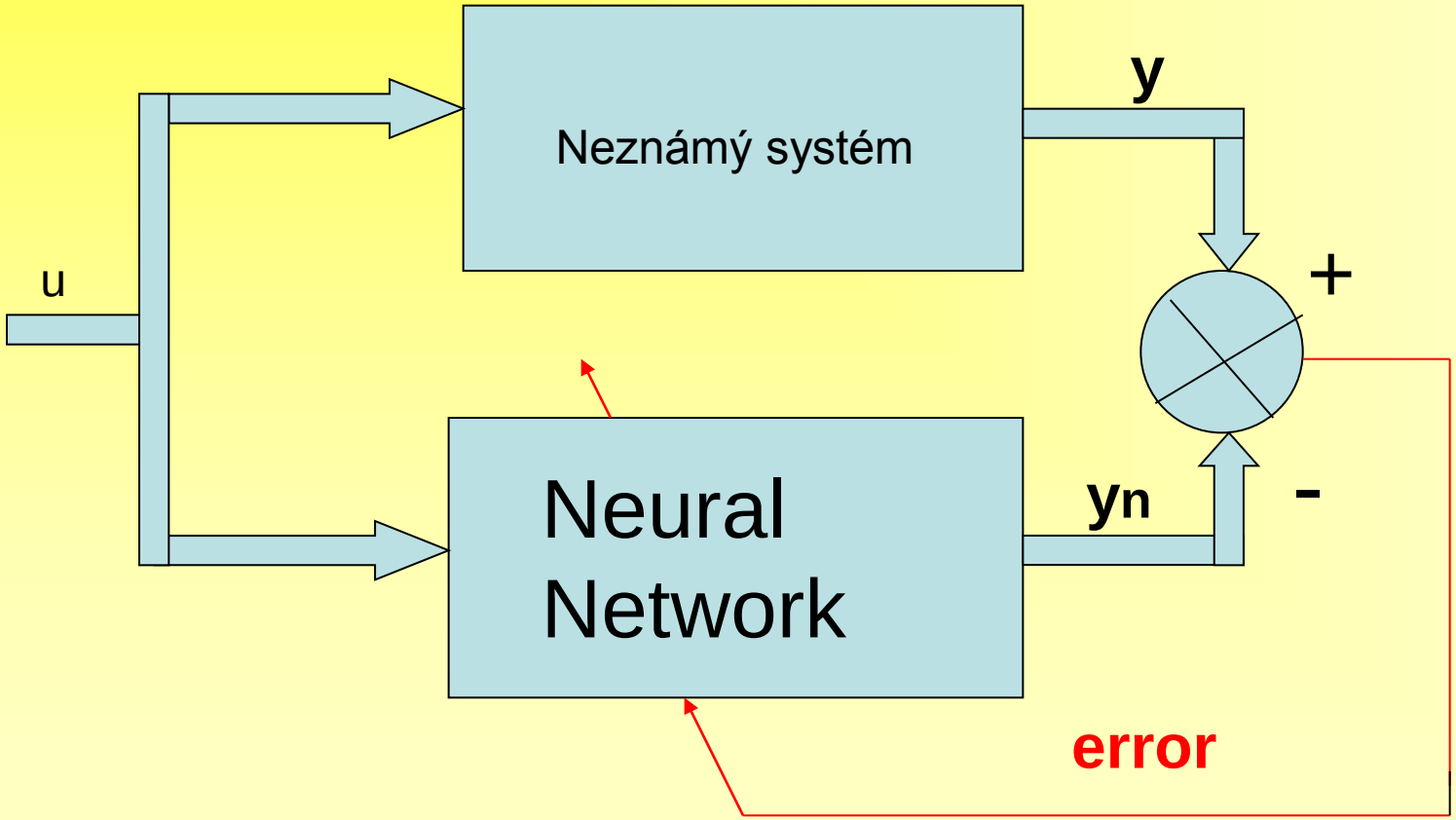
Jak pracuje neuronová síť pokr.

- Neuronová síť pracuje v režimech:
  - **režim učení**, tj. ladění vah neuronů pro dosažení minimální chyby : a) adaptace
    b) trénování
  - **testování sítě**
  - **režim provozu**: a) síť je už naučená a může se použít s pevně nastavenými váhami(např. rozpoznávání písma).
    b) síť je už naučená, může se používat ale váhy se stále adaptují (např. adaptabilní regulátor)

Jak se učí neuronová síť s učitelem

(supervised learning)

Příklad identifikace neznámého systému



Váhy neuronů v síti se ladí podle zvoleného pravidla učení (např. **Back Propagation**,…)

Jak se učí neuronová síť bez učitele
(unsupervised learning)

- Například v klasifikačních úlohách, síť nastavuje váhy aktivačních funkcí neuronů implementovaným pravidlem učení, např. **Hebbovo** nebo **Delta pravidlo**, pro jednotlivý daný vstupní obrazec tak dlouho, dokud nedosáhne konzistentního stavu (tj. vhodného energetického minima). Potom následuje další vstupní obrazec, atd.

Algoritmy učení sítě

- Hebbovo pravidlo $\Delta w_{ij}(k+1) = \eta \; a_i(k) \; y_j(k)$
- Delta pravidlo $\Delta w_{ij}(k+1) = \eta \; (p_i(k) - a_i(k)) \, y_j(k)$
- Back-Propagation

Poznámka: pro sítě typu RBF se používají metody interpolace multidimenzionální funkce (interpolační matice) pro nalezení neurálních vah

Pravidlo učení B-P

- Je jedno z nejpoužívanějších pravidel (rozšířené pravidlo Widrow-Hoff) kde přírůstek váhy je úměrný velikosti chyby a působí v opačném směru gradientu jejího přírůstku

$$např: \quad \Delta w_{ij} = -\mu \cdot \frac{\partial J}{\partial w_{ij}} = -\mu\, 2e \frac{\partial e}{\partial w_{ij}}$$

$$kde \quad \mu ... rychlost \;\; učení$$

$$J = \int e^2(t)\, dt \;\; ... \;\; Energetická \;\; funkce$$

$$(zde\; integrální \;\; kvadratické \;\; kriterium \;\; chyby)$$

$$e = (y_{target} - y_{neuron}) \; ... \; chyba$$

Modelování, identifikace a analýza signálů pomocí neuronových sítí.

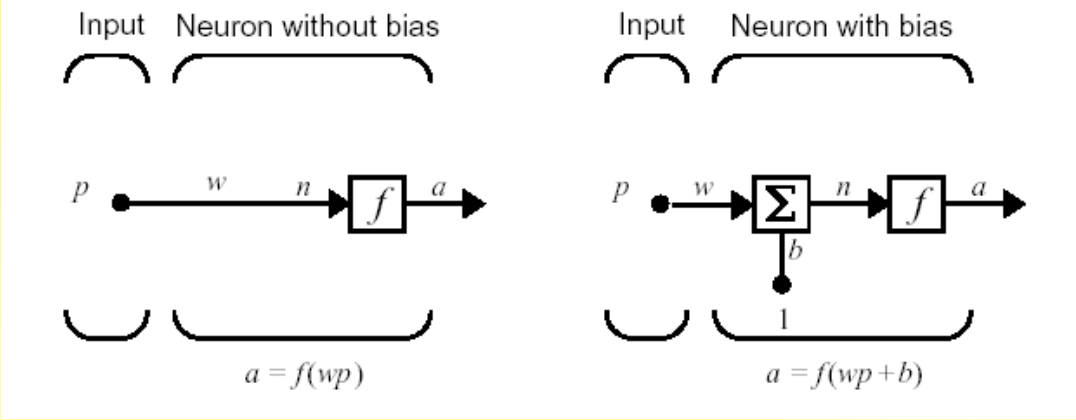
(Predikce signálů)

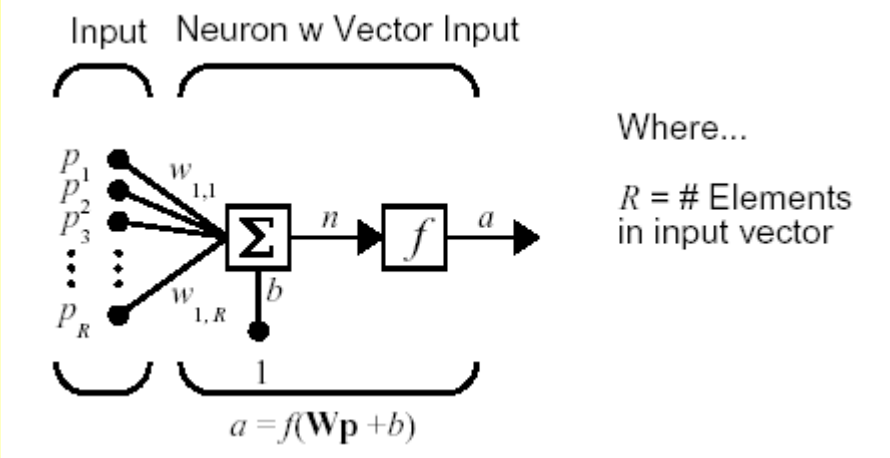
- Sítě MLP
- Rekurentní neuronové sítě
- Nové neurální statické a dynamické architektury HONNU (výzkum od r. 2003)
- A další

Notace v MATLAB NN-Toolboxu

a … výstup neuronu b … práh f…přenosová funkce w…váhy neuronu

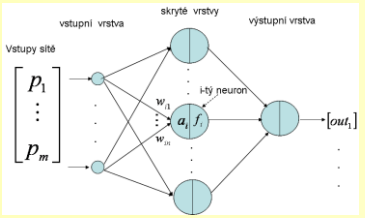
Neuron s jedním vstupem

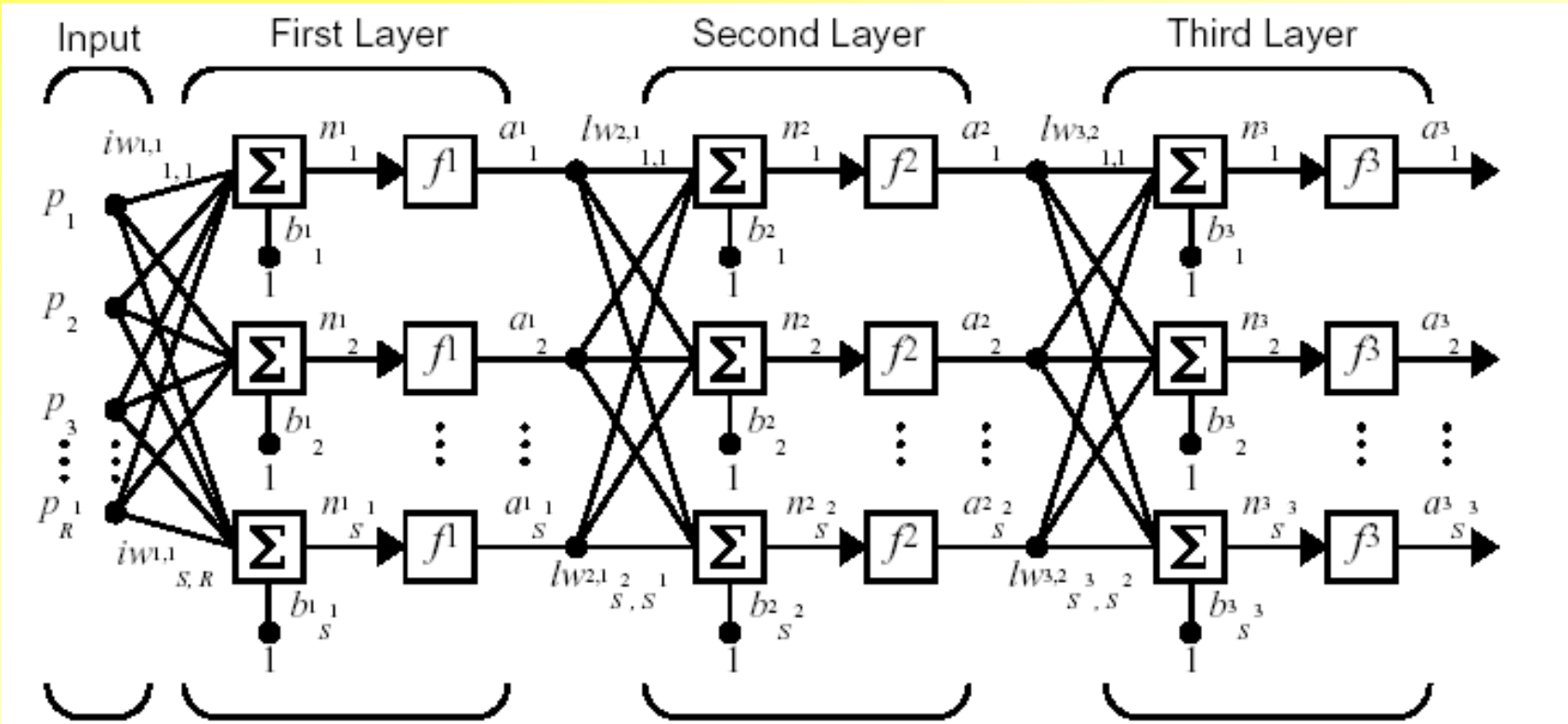




Notace v MATLAB NN-Toolboxu pokr.

Více-vrstvá dopředná neuronová síť (Feed-Forward NN)



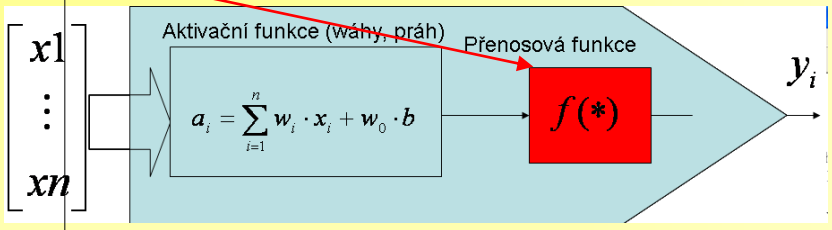


$$\mathbf{a}^1 = \mathbf{f}^1(\mathbf{IW}^{1,1}\mathbf{p} + \mathbf{b}^1) \qquad \mathbf{a}^2 = \mathbf{f}^2(\mathbf{LW}^{2,1}\mathbf{a}^1 + \mathbf{b}^2) \qquad \mathbf{a}^3 = \mathbf{f}^3(\mathbf{LW}^{3,2}\mathbf{a}^2 + \mathbf{b}^3)$$

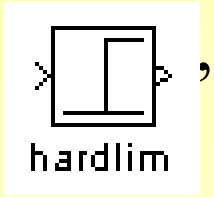
$$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{LW}^{3,2}\mathbf{f}^2(\mathbf{LW}^{2,1}\mathbf{f}^1(\mathbf{IW}^{1,1}\mathbf{p} + \mathbf{b}^1) + \mathbf{b}^2) + \mathbf{b}^3)$$

Typy FFNN v NN-Toolboxu

- Důležitými hledisky je příhodnost daného typu NN sítě pro danou aplikaci a typ implementované přenosové funkce *f* (*).



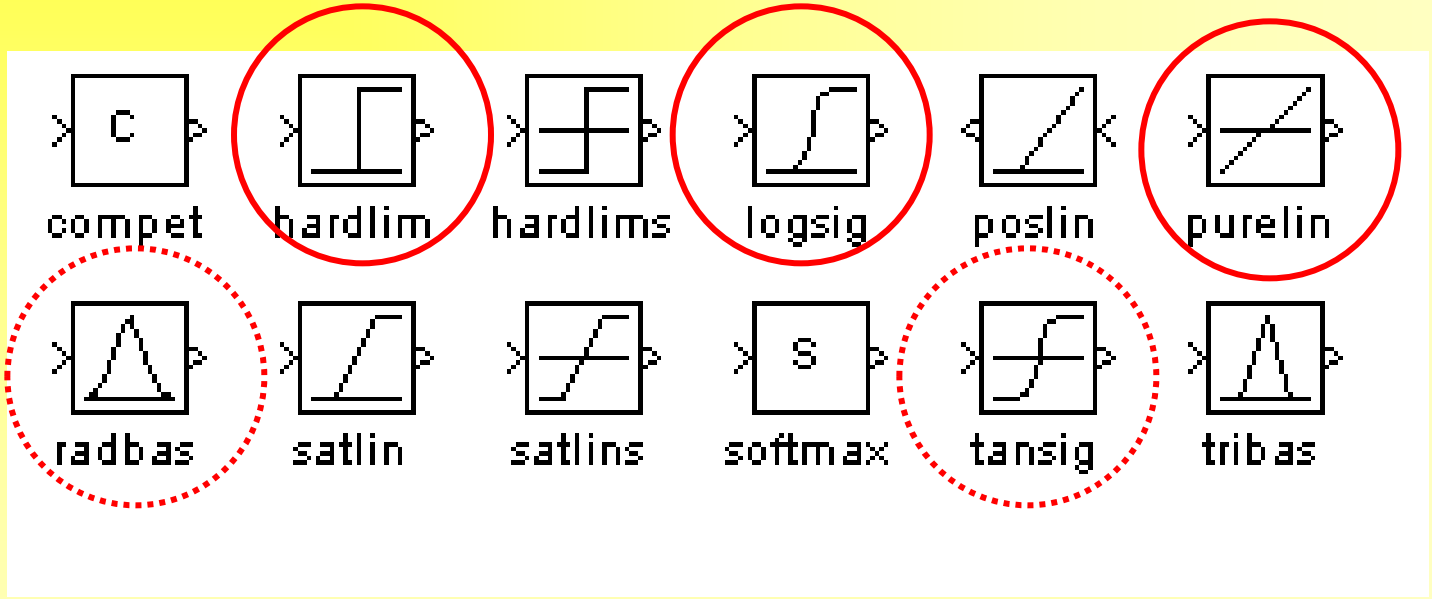
např:

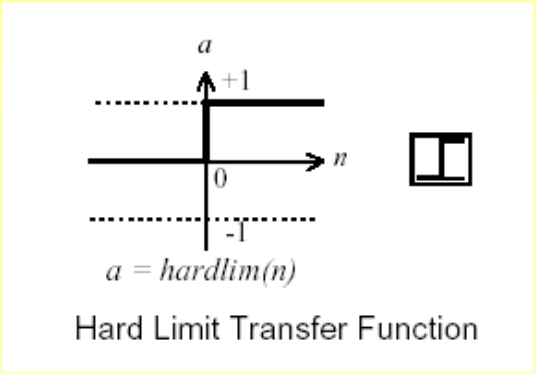
- Síť 1-Vrstvý perceptron: jednovrstvá síť s výstupní funkcí typu 

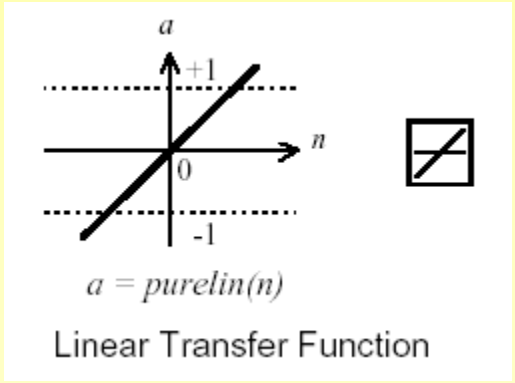
, vhodná pro klasifikační problémy.

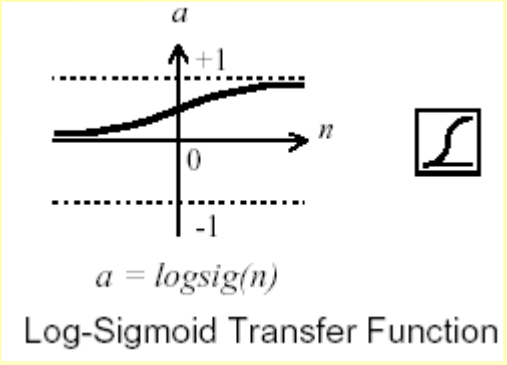
Typy FFNN v NN-Toolboxu pokr.

- Typy přenosových funkcí *f* (*)



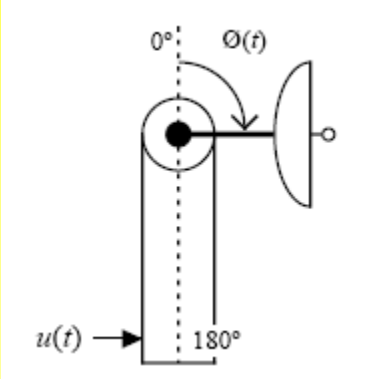






Příklad identifikace a řízení

>Appcs1 > Appcs2

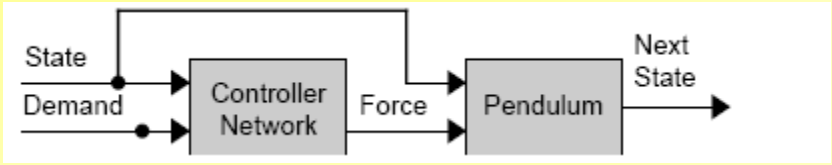


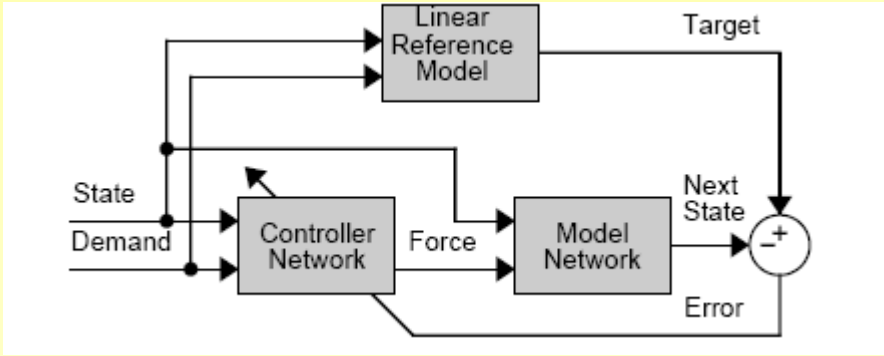
soustava

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ 9.81\sin x_1 - 2x_2 + u \end{bmatrix}$$

Lineární referenční model – požadované chování

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -9x_1 - 6x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 9r \end{bmatrix}$$





Práce v NN-Toolboxu **3.0.1**.

- NN síť v NN-Toolboxu se vytváří na příkazovém řádku Matlab commanderu, tj. nejlépe zápisem do *.m souboru a pak jeho spuštěním.
- Síť se z Matlab commanderu musí i učit.
- Naučenou síť lze vygenerovat i jako blok do Simulinku a tam dále používat.

Práce v NN-Toolboxu **3.0.1** pokr.

Vytvoření obecné prázdné struktury NN sítě námi nazvanou „net“:

net = NETWORK(…)

Příkazy které vytvoří před-definovanou prázdnou strukturu NN sítě námi nazvanou „net“:

net = NEWP([min max; …;min max],n);

net = NEWLIND(P,T);

net = NEWLIN([min max; …;min max],n);

net = NEWFF ([min max; …;min max],[n1 n2],{'tansig','purelin'},'trainlm');

Práce v NN-Toolboxu **3.0.1** pokr.

Některé atributy sítě které můžete chtít nastavit:

- net.layers{i}.initFcn = 'initwb';
- net.inputWeights{i,j}.initFcn='rands';
- net.layerWeights{i,j}.initFcn='rands';
- net.biases{i}.initFcn='rands';
- net.trainParam.epochs = 1000;
- net.trainParam.goal = .001^2;
- net.trainParam.show = 40;

Práce v NN-Toolboxu **3.0.1** pokr.

- Inicializace sítě:

  net = INIT(net);

- Trénování sítě:

  [net , e] = TRAIN(net,P,T); (vrací vektor výstupu a chybu)

- Spuštění provozu sítě v Matlab commanderu:

  y=SIM(net,P); (vrací vektor výstupu)

- Generování sítě do Simulinku:

  GENSIM(net,dt);

Práce v NN-Toolboxu **3.0.1** pokr.

- Další šikovné příkazy Matlabu:

- combvec
- minmax
- zeros, ones
- length
- save, load
- help
- Např.: >>: help combvec
  >>: help newlind

Příkazy pro učení sítě

Trénování (Batch Training)

Váhy a prahy neuronů se nastavují po proběhnutí celé sekvence všech vektorů vstupů

net = TRAIN(net,P,T,Pi,Ai).

net – Network, P - Network inputs, T - Network targets.

Větší možnosti metod učení

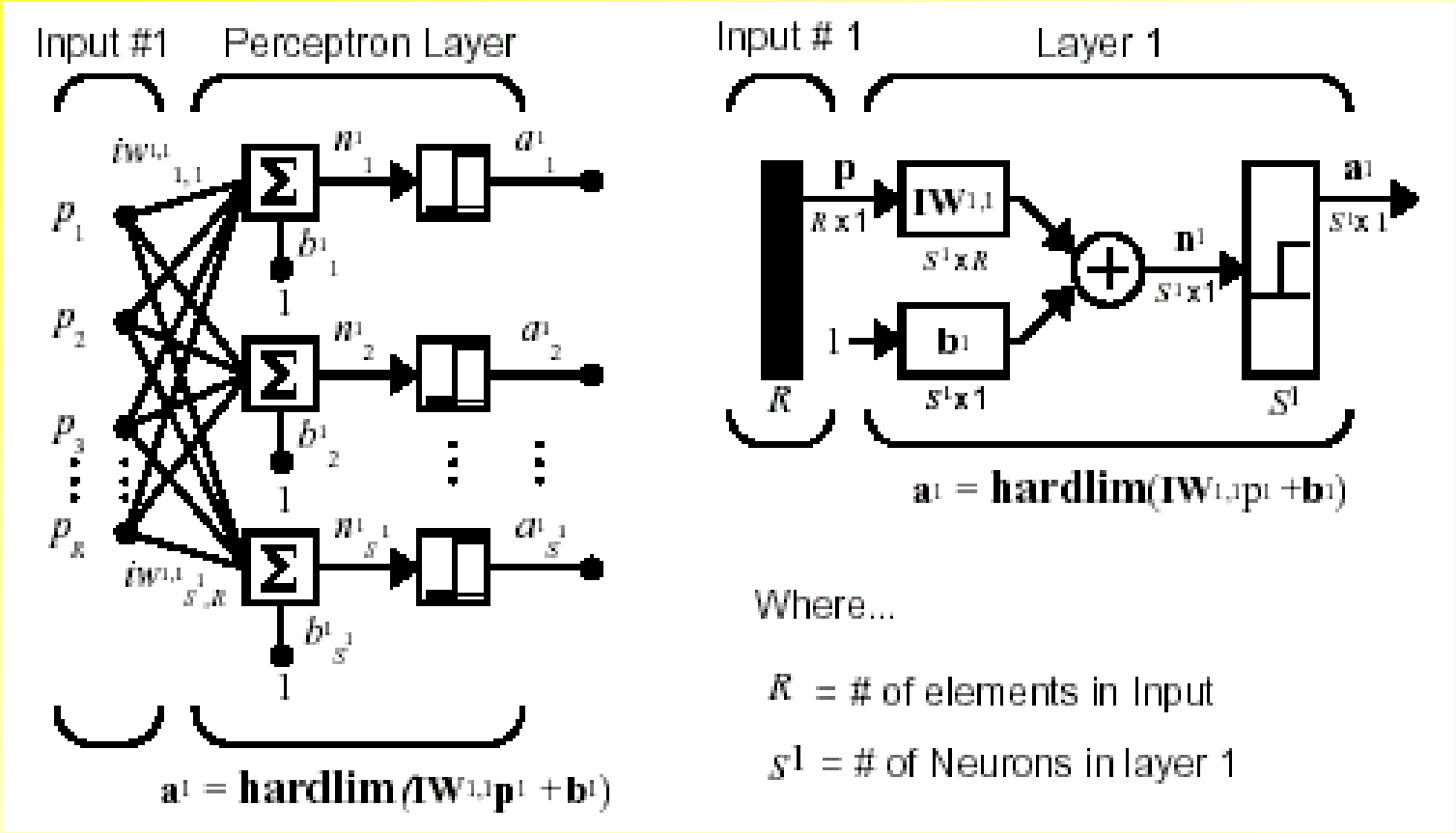
Adaptace (Incremental Training)

Váhy a prahy neuronů se nastavují po každé jednotlivé prezentaci vektoru vstupu

[net,Y,…] = ADAPT(net,P,T,…)

Vhodné pro průběžnou identifikaci, adaptabilní regulátory,…

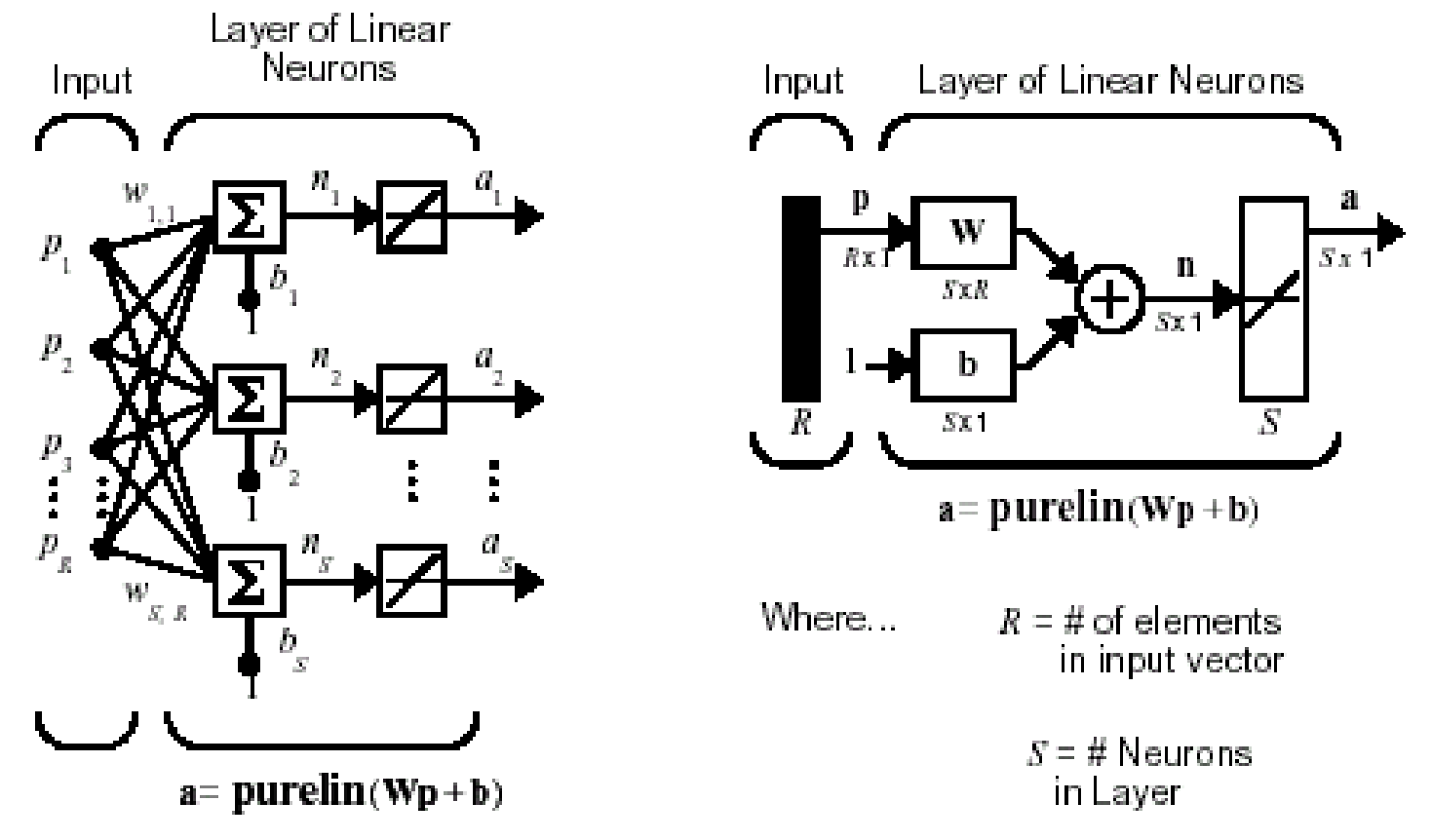
Příkaz NEWP: 1-Layer Perceptron NN



net = newp(PR, S)

kde PR=minmax(P) a P je vektor vstupů.

Příkaz NEWLIN: 1-Linear Layer NN



net = NEWLIN(PR,S)

kde PR=minmax(P) a P je vektor vstupů

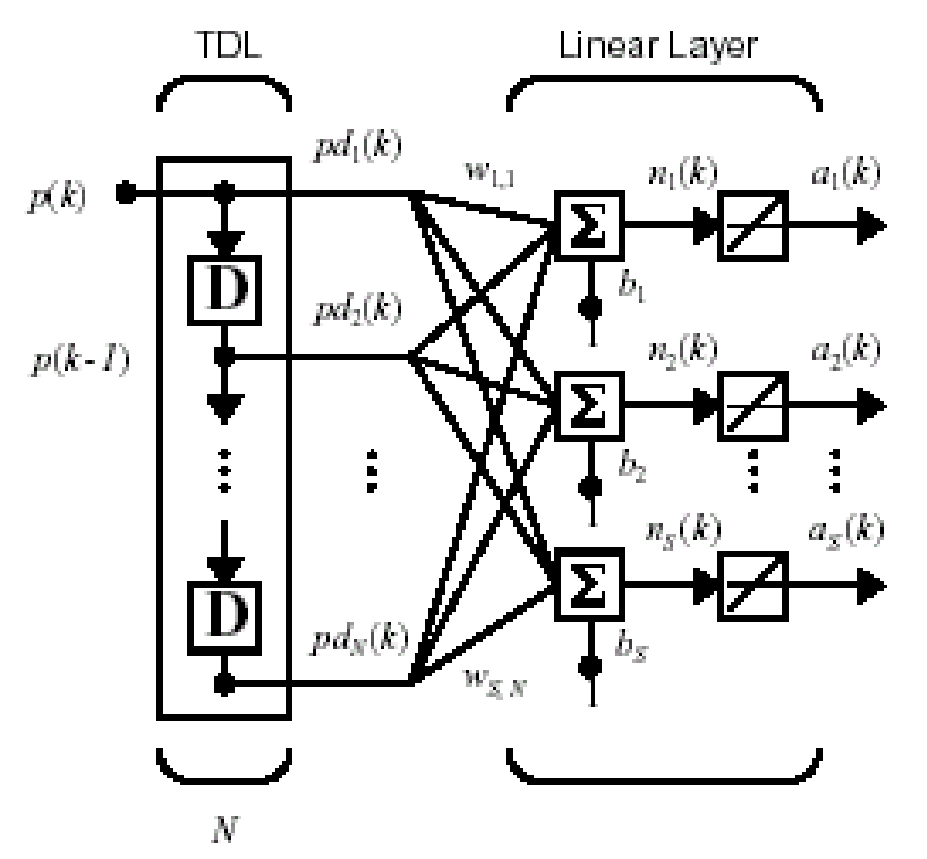
Příkaz NEWLIN: 1-Linear Layer NN

pokr.

Using Tapped Delays

net=NEWLIN(PR,S,ID)

ID =[0 1 2 …n] … Input delay vector



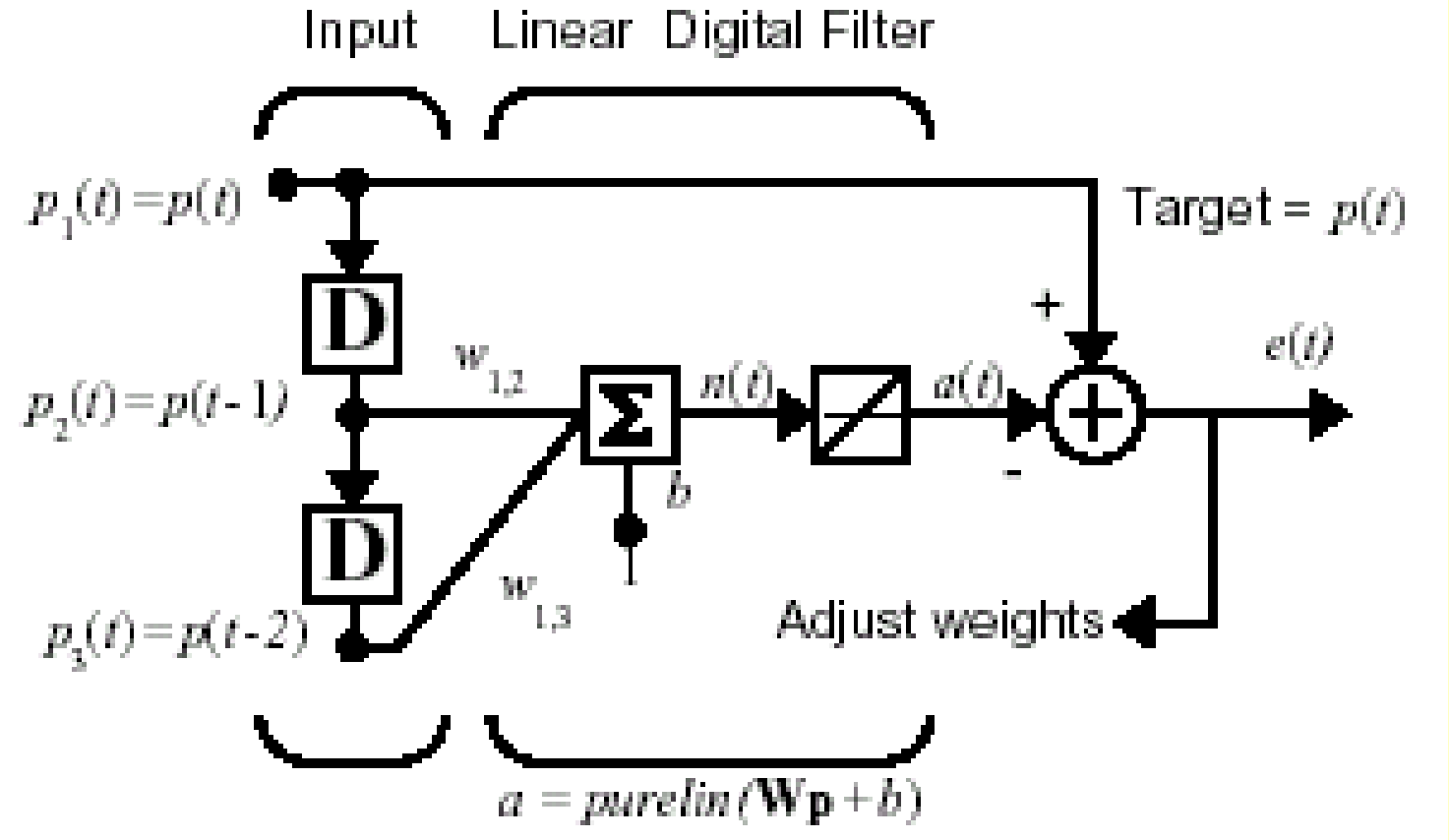
$$a(k) = purelin(\mathbf{W}\mathbf{p} + b)$$

$$= \sum_{i=1}^{R} w_{1,i} a(k - i + 1) + b$$

Příkaz NEWLIN: 1-Linear Layer NN

pokr.

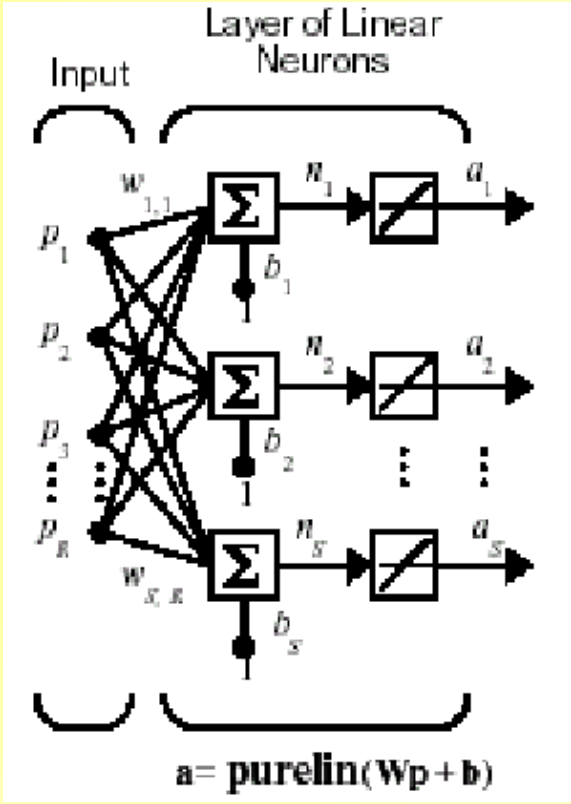
Linear Prediction



Příkaz NEWLIND:Linear System Design

net=NEWLIND(P,T)

Sám navrhne a naučí neuronovou síť o jedné lineární vrstvě neuronů (f(*)…purelin)



"… single-layer linear networks are just as capable as multi-layer linear networks.
For every multi-layer linear network,there is an equivalent single-layer linear network…"

pp.105

Příkaz NEWLIND:Linear System Design pokr.

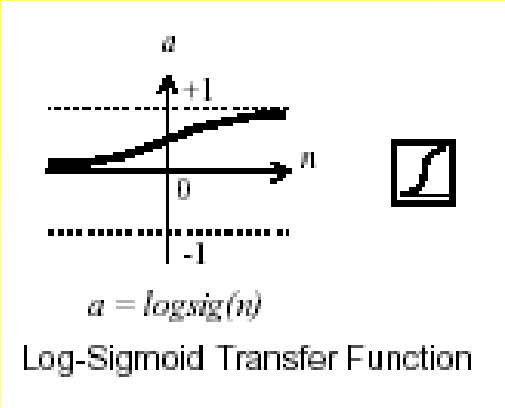
```
P = [1 2 3];
T= [2.0 4.1 5.9];

net = newlind(P,T);

Y = sim(net,P)
Y =
    2.0500    4.0000    5.9500
```

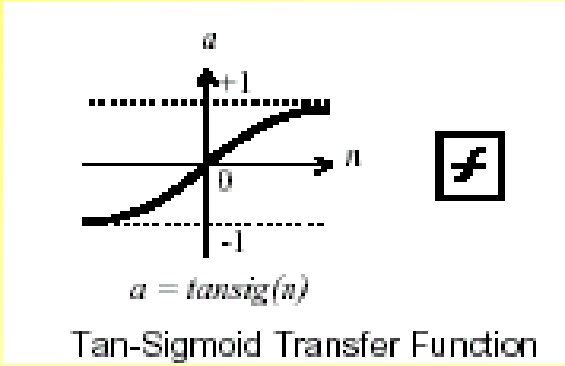
NN sítě s neurony s lineární přenosovou funkcí f(*) jsou vhodné hlavně pro lineární či lineárně separovatelné problémy.

Pro nelineární problémy alespoň minimalizují střední kvadratickou chybu (MSE), ale při učení nekonvergují k přesnému řešení, jinými slovy, síť musí obsahovat dané řešení jako podmnožinu své struktury, aby k ní vůbec mohla konvergovat…

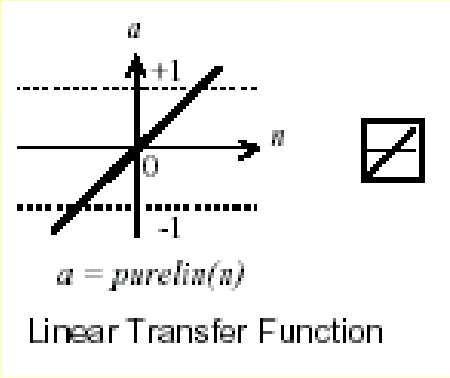
Multilayer Feed Forward NN

- Nejvíce používané sítě
- Přenosové funkce ve skrytých vrstvách:





- Přenosové funkce ve výstupní vrstvě jsou nejčastěji lineární:



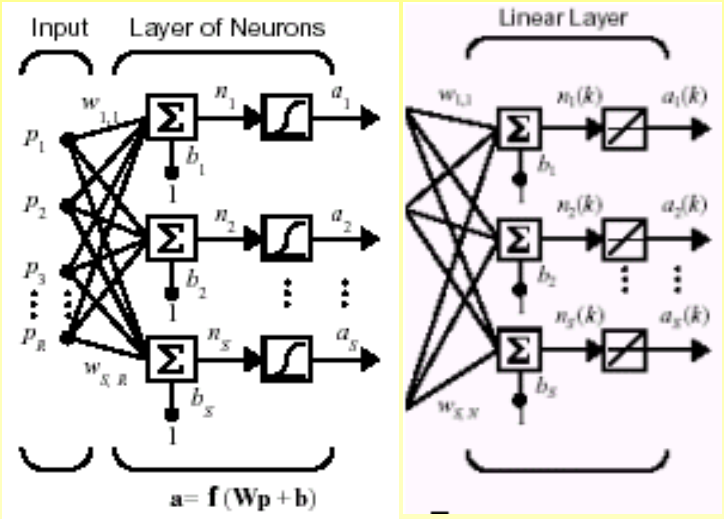
Příkaz NEWFF:

Multilayer Feed Forward NN pokr.

net=NEWFF(minmax(P),[n1,...,nn,],{'tansig',…'purelin};

Skryté vrstvy

Výstupní vrstva

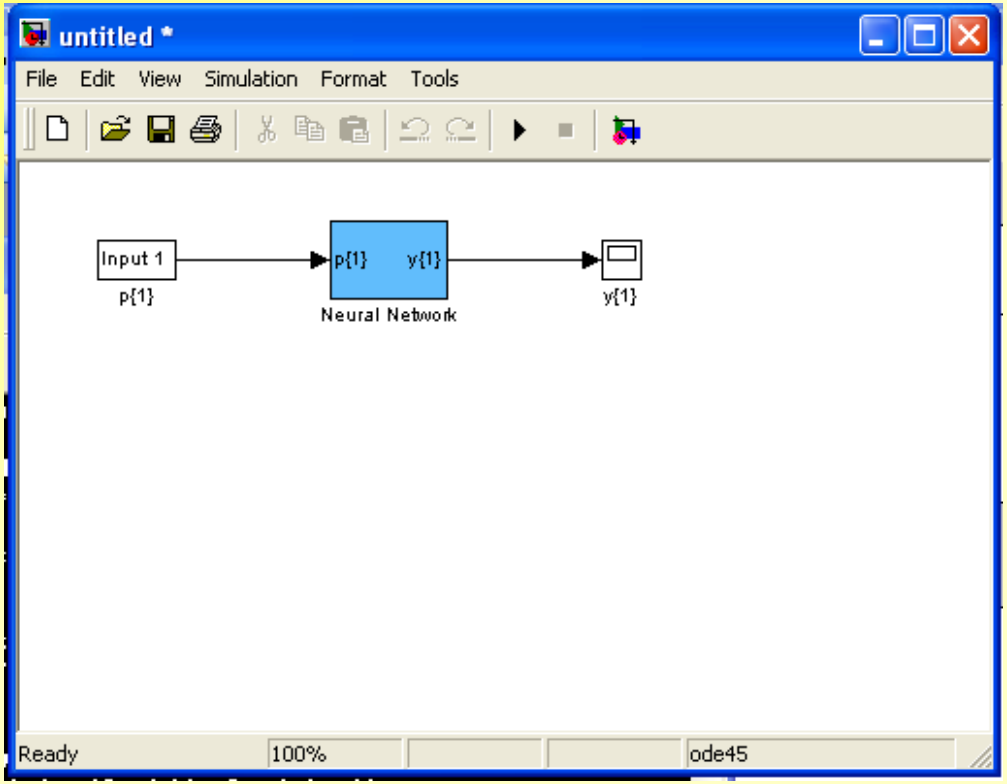


Např:

net=newff([-1 2; 0 5],[3,1],{'tansig','purelin'},'traingd');

Příkaz GENSIM

GENSIM(net,Ts) generuje navrženou NN síť „net“ do Simulinku



e-zdroje

Například

- *Manuál k NN toolboxu pro Matlab v AJ*
  http://www-ccs.ucsd.edu/matlab/pdf_doc/nnet/nnet.pdf
- http://www.automa.cz/automa/2005/au010520.htm česky
- http://www.stech.cz/articles.asp?ida=132&idk=235 česky

A mnoho dalších…