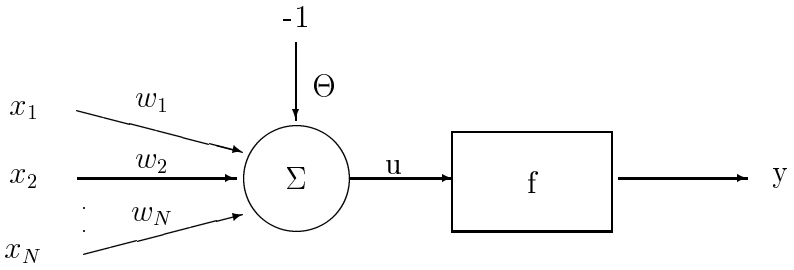
Model umělého neuronu



$$y = f(\sum_{i=1}^{N} w_i x_i - \Theta)$$

y – výstup neuronu

u – vnitřní potenciál neuronu

w_i – váhy neuronu

x_i – vstupy neuronu

Θ – práh neuronu

f – neuronová aktivační funkce

Neuronové aktivační funkce:

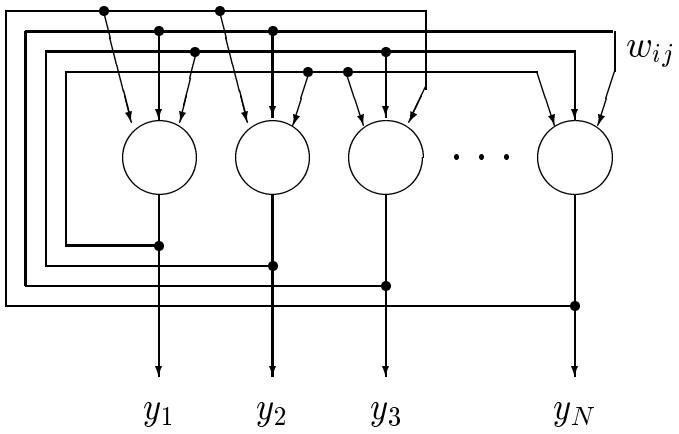
Sigmoidální funkce $f_s(u) = \frac{1}{1+e^{-u}}$

Hyperbolický tangens $f_h(u) = tgh\,(u)$

Znaménková funkce $f_z(u) = sgn\,(u)$

Heavisideova funkce $f_H(u) = \begin{cases} 1 \;\; pro\; u > 0 \\ 0 \;\; pro\; u \leq 0 \end{cases}$

Hopfieldova síť



– jednovrstvá rekurentní síť s pevnými vahami,
– vstupní vektory musí být binární (0/1), popř. bipolární (-1/1)

Neuronová aktivační funkce:

– funkce jednotkového skoku $$f_H(u) = \begin{cases} 1 & u > 0 \\ 0 & u \leq 0 \end{cases}$$

– znaménková funkce (pro bipolární vstupní vektory)

Použití:

– autoasociativní paměť

– rekonstrukce neúplných a šumem poškozených obrazů

– optimalizační problémy

Kapacita sítě (maximální počet uchovaných vzorů):

$$P \leq 0.15N,$$

kde P je počet vzorů uchovávaných v paměti, N je počet neuronů

Energie:

$$E(t) = -\frac{1}{2} \sum_i \sum_j w_{ij} a_i(t) a_j(t) - \sum_i \Theta_i a_i(t)$$

Hopfieldova síť (asynchronní model)

Krok 1: *Inicializace váhových koeficientů* w_{ij} *a prahů* Θ_i *podle vztahu*

$$w_{ij} = \begin{cases} \Sigma_{s=1}^{P}(2a_i^s - 1)(2a_j^s - 1)\,, & i \neq j, \quad 1 \leq i,j \leq N\,, \\ 0 & i = j \end{cases}$$

$$\Theta_i = \tfrac{1}{2}\,\Sigma_{j=1}^{N} w_{ij}\,, \qquad i = 1,...,N\,,$$

kde w_{ij} *jsou váhové koeficienty mezi i-tým a j-tým neuronem,* a_i^s *je i-tý prvek vzorového vektoru s-té třídy (může nabývat pouze hodnot* $\{0,1\}$*),* P *je počet vzorových vektorů.*

Krok 2: *Předložení neznámého vstupního vektoru* $\mathbf{X} = [x_1, ...x_N]^T$, $\mathbf{X} \in \{0,1\}^N$

$$y_i(0) = x_i\,, \qquad 1 \leq i \leq N\,,$$

kde $y_i(0)$ *je výstup i-tého neuronu v čase t=0,* x_i *je i-tý prvek vstupního vektoru.*

Krok 3: *Iterace, dokud síť konverguje*
Během t-té iterace se nastavují výstupy neuronů v sekvenčním pořadí pro i=1,2,...,N

a) $$u_i(t+1) = \Sigma_{j=1}^{N} w_{ij} y_j(t)) - \Theta_i$$

b) $$y_i(t+1) = \begin{cases} 1 & pro \;\; u_i(t+1) > 0 \\ 0 & pro \;\; u_i(t+1) < 0 \\ y_i(t) & pro \;\; u_i(t+1) = 0 \end{cases}$$

Krok 4: *Opakování od kroku 2 pro nový vstupní vektor*

Hopfieldova síť (synchronní model)

Krok 1: *Inicializace váhových koeficientů w_{ij} a prahů Θ_i podle vztahu*

$$w_{ij} = \sum_{s=1}^{P} (2a_i^s - 1)(2a_j^s - 1)\,, \qquad 1 \le i, j \le N\,,$$

$$\Theta_i = \tfrac{1}{2} \Sigma_{j=1}^{N} w_{ij}\,, \qquad i = 1, ..., N\,,$$

kde w_{ij} jsou váhové koeficienty mezi i-tým a j-tým neuronem, a_i^s je i-tý prvek vzorového vektoru s-té třídy (může nabývat pouze hodnot {0,1}), P je počet vzorových vektorů.

Krok 2: *Předložení neznámého vstupního vektoru* $\mathbf{X} = [x_1, ...x_N]^T$, $\mathbf{X} \in \{0,1\}^N$

$$y_i(0) = x_i\,, \qquad 1 \le i \le N\,,$$

kde $y_i(0)$ je výstup i-tého neuronu v čase t=0, x_i je i-tý prvek vstupního vektoru.

Krok 3: *Iterace, dokud síť konverguje*
Během t-té iterace se:

1) vypočtou hodnoty $u_i(t+1)$ paralelně pro i=1,2,...N podle vztahu

$$u_i(t+1) = \Sigma_{j=1}^{N} w_{ij} y_j(t)) - \Theta_i$$

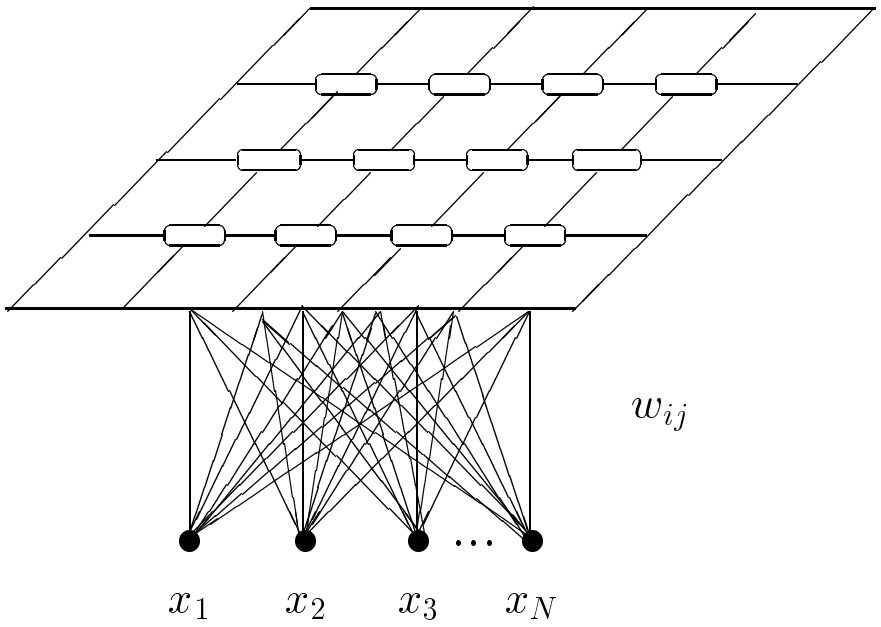
2) vypočtou výstupní hodnoty $y_i(t+1)$ pro i=1,2,...N podle vztahu

$$y_i(t+1) = \begin{cases} 1 & pro \;\; u_i(t+1) > 0 \\ 0 & pro \;\; u_i(t+1) < 0 \\ y_i(t) & pro \;\; u_i(t+1) = 0 \end{cases}$$

Tento proces se opakuje, dokud se výstup mění. Po ukončení iterací odpovídají výstupy vzorovému vektoru, který se nejvíce podobá vektoru vstupnímu.

Krok 4: *Opakování od kroku 2 pro nový vstupní vektor*

Kohonenova síť



– jednovrstvá síť s dopředným šířením

– učení: bez učitele

– vstupní vektor: $X = [x_1, ..., x_N]^T, \quad x_i \in \mathbf{R}$

– výstupní hodnota neuronů je definována jako vzdálenost mezi vstupním a váhovým vektorem, t.j.

$$y_i(t) = \sum_{j=1}^{N} (x_j(t) - w_{ij}(t))^2$$

Použití: – shlukování
– analýza dat
– vytváření sémantických map

Algoritmus (Kohonenova síť)

Krok 1: *Počáteční inicializace vah w_{ij} a množiny $Ne_i(t)$, která definuje sousednost okolo každého uzlu i=1,2,...,N.*

Krok 2: *Přivedení vektoru $X = [x_1, ..., x_N]^T$ na vstup sítě a výpočet vzdálenosti mezi vstupním vektorem a váhovým vektorem každého neuronu podle vztahu*

$$d_i = \sum_{j=1}^{N} (x_j(t) - w_{ij}(t))^2$$

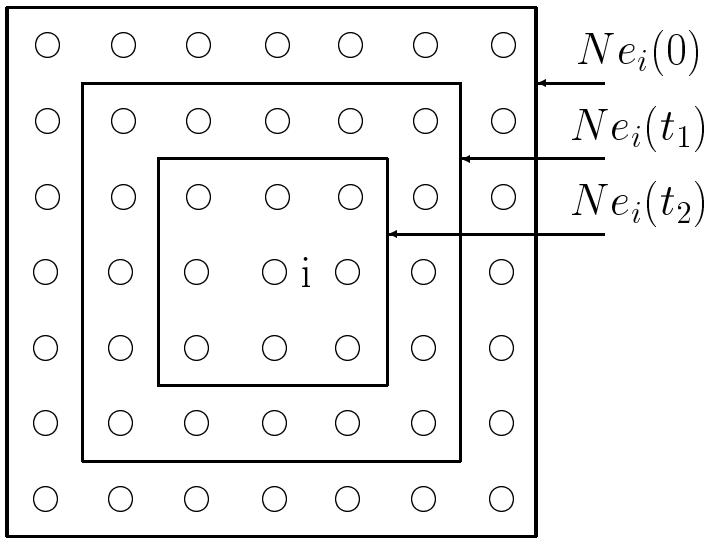
Krok 3: *Výběr i-tého neuronu (s nejmenší vzdáleností d_i) (tzv. vítězného neuronu)*

Krok 4: *Adaptace vah vítězného neuronu a všech neuronů* **k**, *pro které platí* $\mathbf{k} \in Ne_i(t)$ *(sousedí s i-tým neuronem) podle vztahu*

$$w_{ki}(t+1) = w_{ki}(t) + \eta(t)(x_i(t) - w_{ki}(t)),\ \forall\, k \in Ne_i(t)\ ,\ i = 1, 2, ..., N\ ,$$

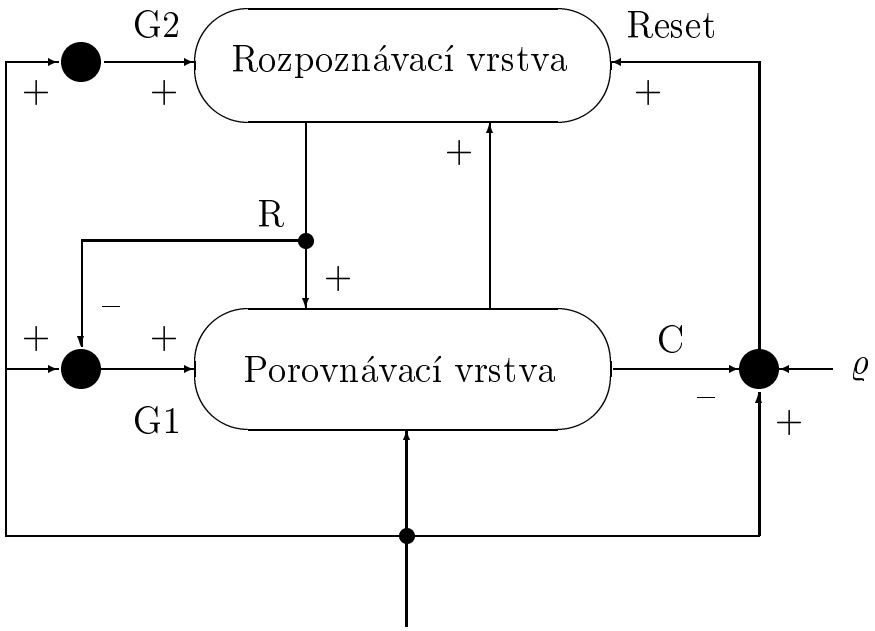
kde $\eta(t)$ $(0 < \eta(t) < 1)$ je koeficient učení, jehož hodnota s rostoucím časem klesá

Krok 5: *Změna sousednosti $Ne_i(t)$ pro i=1,2,...,N*



Krok 6: *Opakování kroků 2-5 pro všechny vstupní vektory, dokud dochází ke změně vah.*

ART (**A**daptive **R**esonance **T**heory)



– dvouvrstvá rekurentní síť
– učení : bez učitele
– vstupní vektor **X**: binární pro model ART-1
reálný pro model ART-2

Použití:

– shlukování
– rozpoznávání znaků, řečových segmentů atd.

Řídicí signály

G2 – G2=1, je-li alespoň jeden prvek vstupního vektoru $x_i = 1$.

G1 – G1=1, je-li alespoň jeden prvek $x_i = 1$. Pokud je však zároveň alespoň jeden prvek $r_i = 1$, je G1=0.

Reset – je generován, je-li podobnost mezi vektorem **C** a **X** menší než předem zvolená hodnota prahu "ostražitosti" ϱ. Podobnost mezi vektory se počítá jako skalární součin **X** a **T** dělený počtem jednotkových bitů vektoru **X**.

Porovnávací vrstva

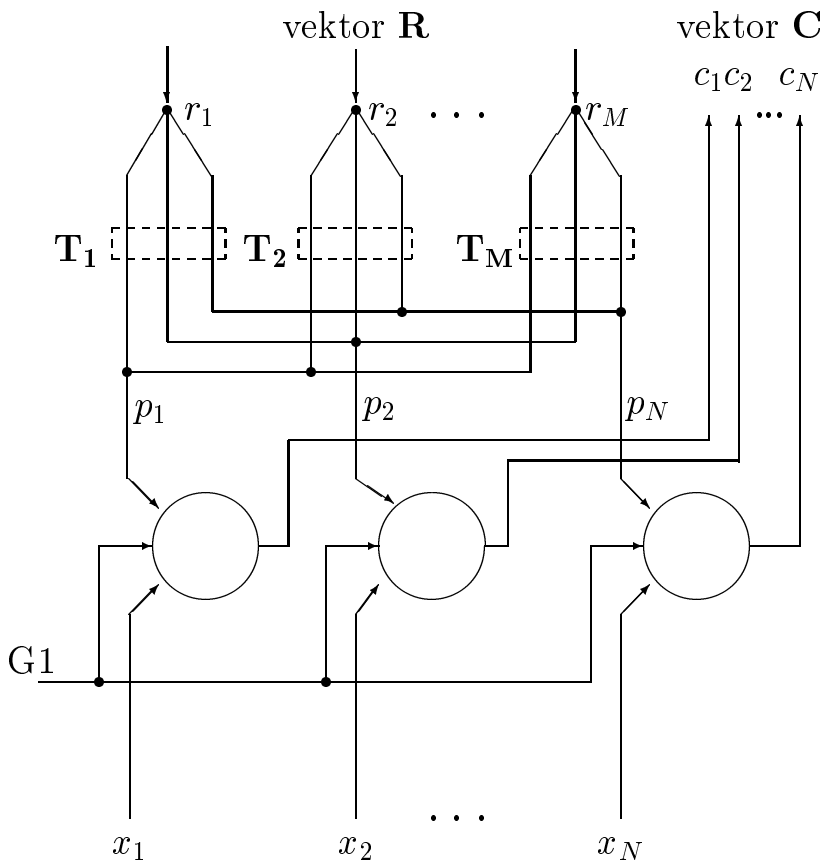
Každý neuron v této vrstvě přijímá tři binární signály:

(1) složky x_i vstupního vektoru $X = [x_1, x_2, ..., x_N]^T$,

(2) zpětnovazební signál p_j, který je vytvořen jako vážený součet výstupů neuronů z rozpoznávací vrstvy,

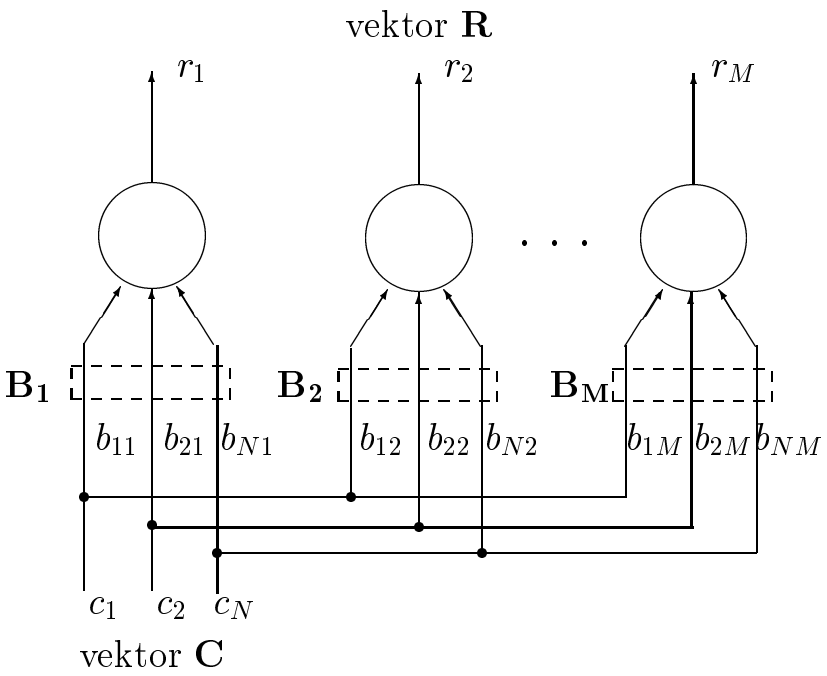
(3) signál G1.

Výstup neuronů $\mathbf{c_i}$ je roven jedné, pokud jsou nejméně dva ze tří signálů rovny jedné.
Na počátku jsou všechny složky vektoru **R** a signál **G1** nastaveny na hodnotu 0 $\Rightarrow$ **C**=**X**.



Rozpoznávací vrstva

Rozpoznávací vrstva klasifikuje vstupní vektor. Skládá se z M neuronů (M je počet klasifikačních tříd), vstupy neuronů jsou spojeny s výstupem porovnávací vrstvy. Spojením jsou přiřazeny vektory váhových koeficientů $\mathbf{B_j}$ $(b_{ji} \in \mathbf{R})$.
Po předložení vstupního vektoru je v rozpoznávací vrstvě aktivován pouze neuron s vektorem váhových koeficientů nejvíce podobným vstupu. Ostatní neurony budou mít výstup nulový.



Algoritmus (ART síť)

Krok 1: *Počáteční inicializace vah t_{ij}, b_{ij} a nastavení prahu ostražitosti ϱ*

$$t_{ij}(0) = 1 \ ,$$

$$b_{ij}(0) = \frac{1}{1+N} \ ,$$

$$i = 1, 2, \ldots, N \ ,$$

$$j = 1, 2, \ldots, M \ ,$$

$$0 \leq \varrho \leq 1$$

Krok 2: *Přivedení vektoru $X = [x_1, ..., x_N]^T$ na vstup sítě.*

Krok 3: *Výpočet odezvy neuronů v rozpoznávací vrstvě podle vztahu*

$$y_j = \sum_{i=1}^{N} b_{ij}(t) x_i \, , \qquad j = 1, 2, \ldots, M \, ,$$

kde y_j je odezva j-tého neuronu v rozpoznávací vrstvě, x_i je i-tý prvek vstupního vektoru.

Krok 4: *Výběr neuronu s největší odezvou $y_k = max(y_j)$.*

Krok 5: *Výpočet podobnosti μ podle vztahu*

$$\mu = \frac{\sum_{i=1}^{N} t_{ik} x_i}{\sum_{i=1}^{N} x_i}$$

Je-li $\mu \geq \varrho$, pokračování krokem 7, jinak provedení kroku 6.

Krok 6: *Zablokování neuronu s největší odezvou a opakování od kroku 3. Výstup k-tého neuronu je dočasně nastaven na nulovou hodnotu a neuron nebude testován při dalším výběru maxima.*

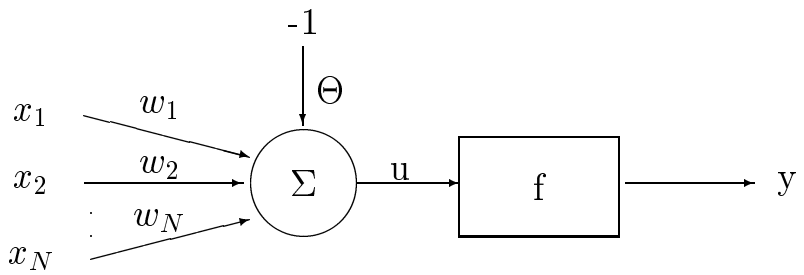
Krok 7: *Adaptace vah u neuronu s největší odezvou. V tomto kroku se modifikují váhy spojení t_{ik} a b_{ik} podle vztahů*

$$t_{ik}(t+1) = t_{ik}(t) x_i \, ,$$

$$b_{ik}(t+1) = \frac{t_{ik}(t) x_i}{0.5 + \sum_{l=1}^{N} t_{lk}(t) x_l}$$

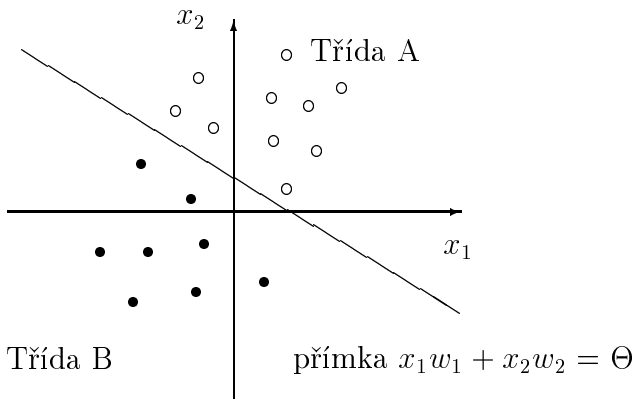
Krok 8: *Odblokování jednotek zablokovaných v kroku 6 a opakování od kroku 2*

Jednoduchý perceptron



- jednoduchý neuron, popř. jednovrstvá síť s dopředným šířením
- učení: s učitelem
- neuronová aktivační funkce: znaménková funkce, popř. jednotkový skok

Použití: Klasifikátor pro lineárně separovatelné obrazy



Trénování jednoduchého perceptronu

Krok 1: *Počáteční inicializace vah* w_i, $1 \leq i \leq N$ *a prahu* Θ *(náhodně).*

Krok 2: *Přivedení vstupního vektoru a definice požadované výstupní odezvy.*

Krok 3: *Výpočet odezvy perceptronu podle vztahu*

$$y(t) = f_z(\sum_{i=1}^{N} w_i(t)x_i(t) - \Theta)$$

Krok 4: *Adaptace vah*

$$w_i(t+1) = w_i(t) + \eta[d(t) - y(t)]x_i(t)\,, \quad 1 \leq i \leq N$$

$$d(t) = \begin{cases} +1\,, & patří - li\ vstup\ do\ třídy\ A \\ -1\,, & patří - li\ vstup\ do\ třídy\ B \end{cases}$$

kde d(t) je požadovaný výstup, η *je tzv. zisk (koeficient učení), který nabývá hodnot z intervalu (0,1).*

Krok 5. *Opakování od kroku 2 pro všechny trénovací dvojice.*

Pro lineárně neseparovatelné obrazy — modifikované algoritmy:

- Kapsový algoritmus (Gallant)
- LMS algoritmus

Trénování perceptronu (Widrow – Hoff LMS)

Krok 1: *Počáteční inicializace vah* w_i *a prahu* Θ

Krok 2: *Přivedení vstupního vektoru* $X = [x_1, ..., x_N]^T$ *a definice požadované výstupní odezvy* $\mathbf{d}$.

Krok 3: *Výpočet vnitřního potenciálu perceptronu podle vztahu*

$$u(t) = \sum_{i=1}^{N} w_i(t)x_i(t) - \Theta$$

Krok 4: *Výpočet a chyby prezentace podle vztahu*

$$E(t) = d(t) - u(t)$$

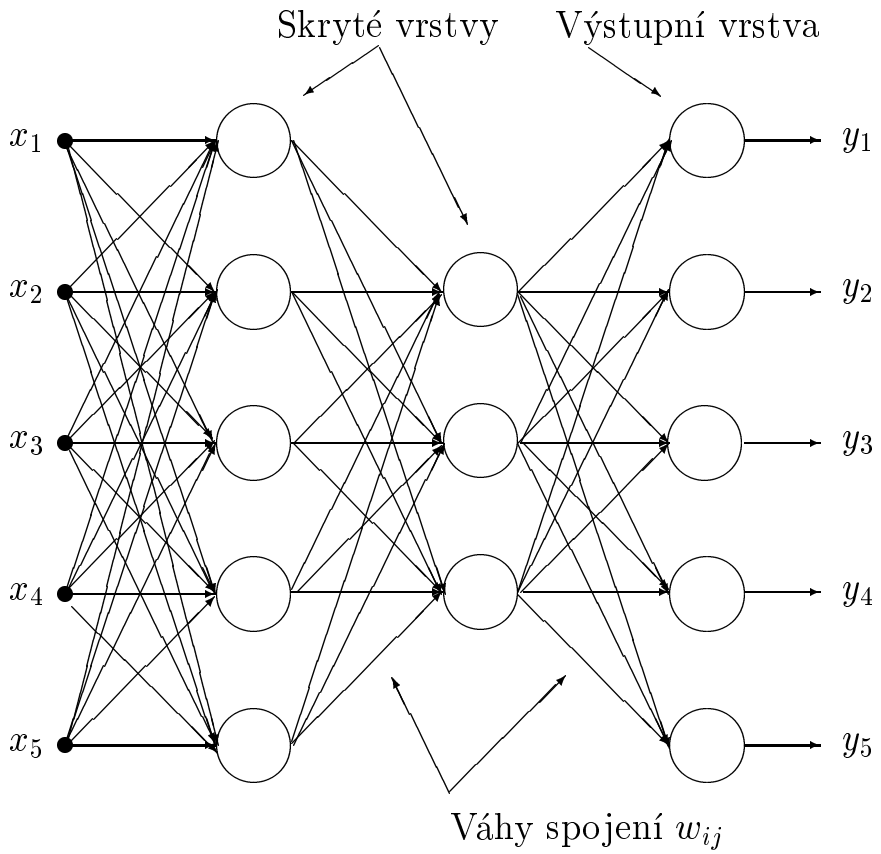
Krok 5: *Adaptace vah*

$$w_i(t+1) = w_i(t) + \frac{\alpha E(t)}{\|X\|^2} x_i \ ,$$

kde $0 < \alpha < 1$ *je tzv. koeficient učení*.

Krok 6: *Opakování kroků 3-5, dokud není změna vah menší než požadovaná přesnost.*

Vícevrstvý perceptron



– vícevrstvá síť s dopředným šířením

– **učení :** s učitelem

– **neuronová aktivační funkce:** sigmoidální funkce, hyperbolický tangens, popř. jiná nelineární funkce spojitá a spojitě diferencovatelná v celém definičním oboru

Použití : – klasifikace obrazů
– aproximace funkcí
– predikce časových řad
– řízení

Trénování vícevrstvého perceptronu (Backpropagation)

Krok 1: *Počáteční inicializace vah* w_{ij} *a prahů* Θ_i *jednotlivých neuronů.*

Krok 2: *Přivedení vstupního vektoru* $\mathbf{X} = [x_1, ..., x_N]^T$ *a definice požadované výstupní odezvy* $\mathbf{D} = [d_1, ..., d_M]^T$.

Krok 3: *Výpočet aktuálního výstupu podle následujících vztahů (platí pro třívrstvý perceptron)*

$$y_l(t) = f_s(\sum_{k=1}^{N_2} w''_{kl}(t)x''_k(t) - \Theta''_l)\,, \quad 1 \le l \le M \quad \textit{výstupní vrstva}$$

$$x''_k(t) = f_s(\sum_{j=1}^{N_1} w'_{jk}(t)x'_j(t) - \Theta'_k)\,, \quad 1 \le k \le N_2 \quad \textit{2. skrytá vrstva}$$

$$x'_j(t) = f_s(\sum_{i=1}^{N} w_{ij}(t)x_i(t) - \Theta_j)\,, \quad 1 \le j \le N_1 \quad \textit{1. skrytá vrstva}$$

Krok 4: *Adaptace vah a prahů podle vztahu*

$$w_{ij}(t+1) = w_{ij}(t) + \eta\delta_j x_i\,, \quad \textit{popř.}$$

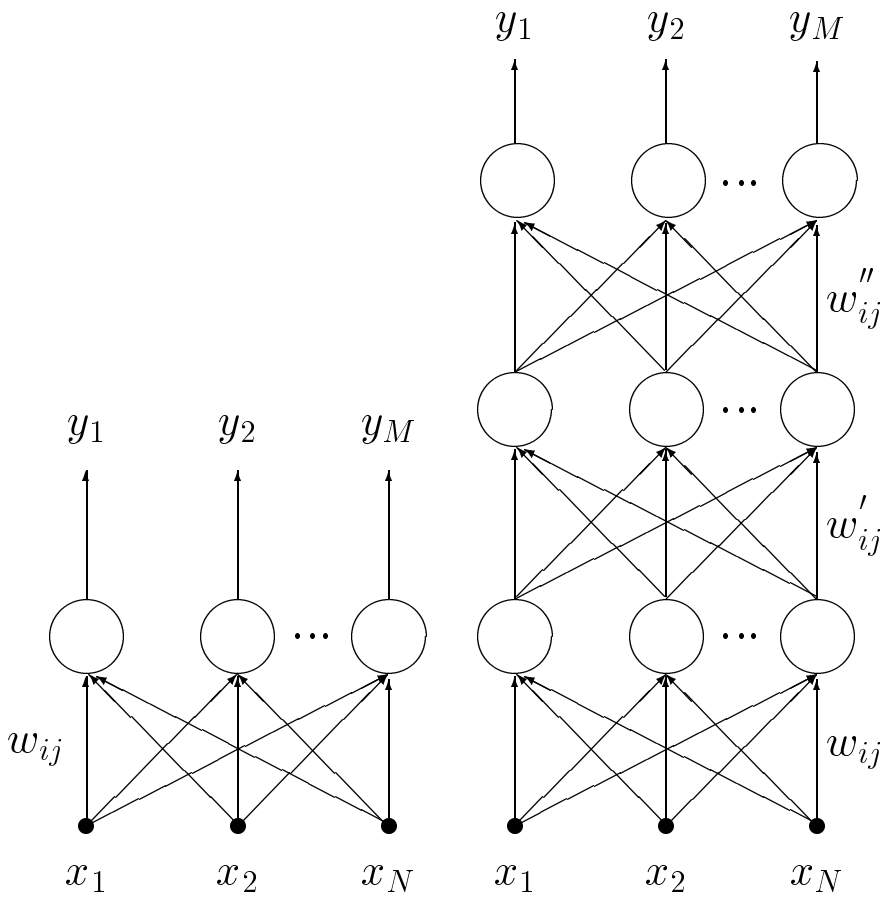
$$w_{ij}(t+1) = w_{ij}(t) + \eta\delta_j x_i + \alpha(w_{ij}(t) - w_{ij}(t-1))\,.$$

Nastavení vah začíná u výstupních uzlů a postupuje rekurzivně směrem ke vstupům. V uvedených vztazích jsou w_{ij} *váhy mezi i-tým skrytým uzlem (popř. vstupním uzlem) a uzlem j-tým v čase t,* x'_i *je výstup i-tého, popř. vstupního uzlu,* η *je koeficient učení (zisk),* α *je tzv. momentový koeficient a* δ_j *je chyba, pro kterou platí následující vztahy:*

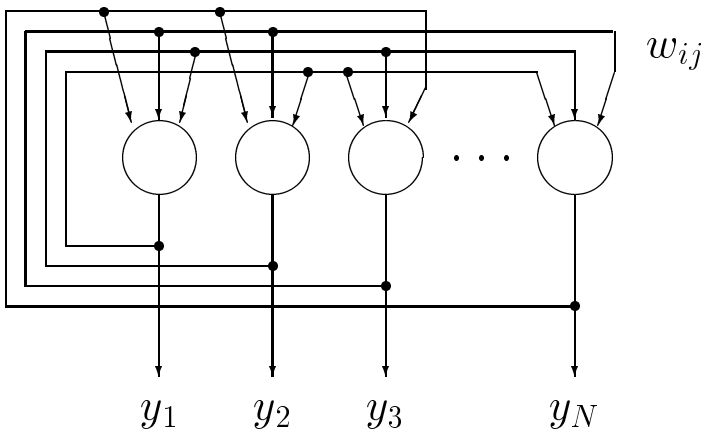
$$\delta_j = \begin{cases} y_j(1-y_j)(d_j - y_j) & \textit{pro výstupní uzly}\,, \\ x'_j(1-x'_j)(\Sigma_k\,\delta_k w_{jk}) & \textit{pro vnitřní skryté uzly}\,, \end{cases}$$

kde k se mění přes všechny uzly vrstvy, která následuje za uzlem j.

Krok 5 : *Opakování kroků 3 - 5, dokud chyba není menší než předem stanovená hodnota.*



a) jednovrstvá **b) vícevrstvá**



c) rekurentní