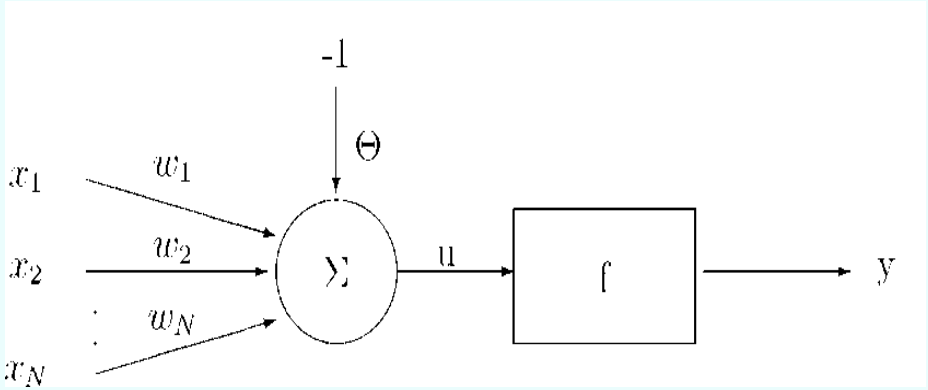
Neuronové sítě

27. 3. 2013

Model umělého neuronu



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

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 $f_s(u) = \mathbf{tgh}(u)$

znaménková funkce $f_s(u) = sgn(u)$

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

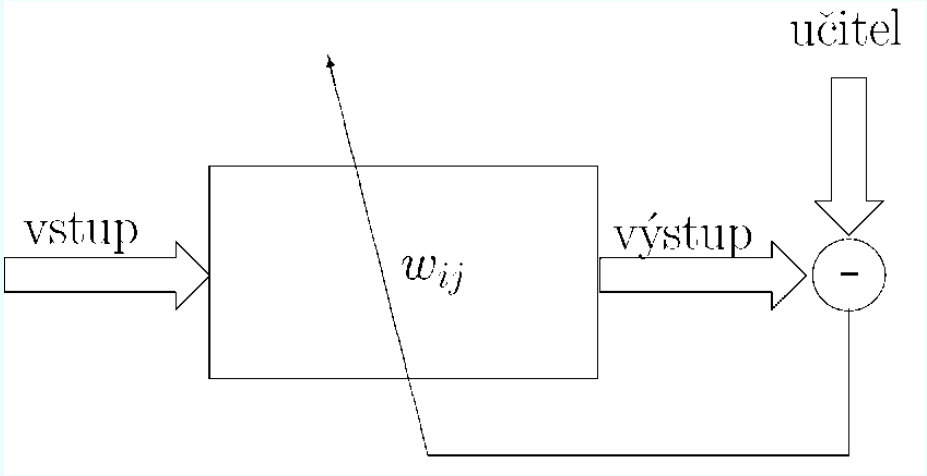
Učení neuronových sítí

Cílem učení je nastavit váhy spojení w_{ij} tak, aby síť vytvářela správnou odezvu na vstupní signál.

Základní způsoby učení:

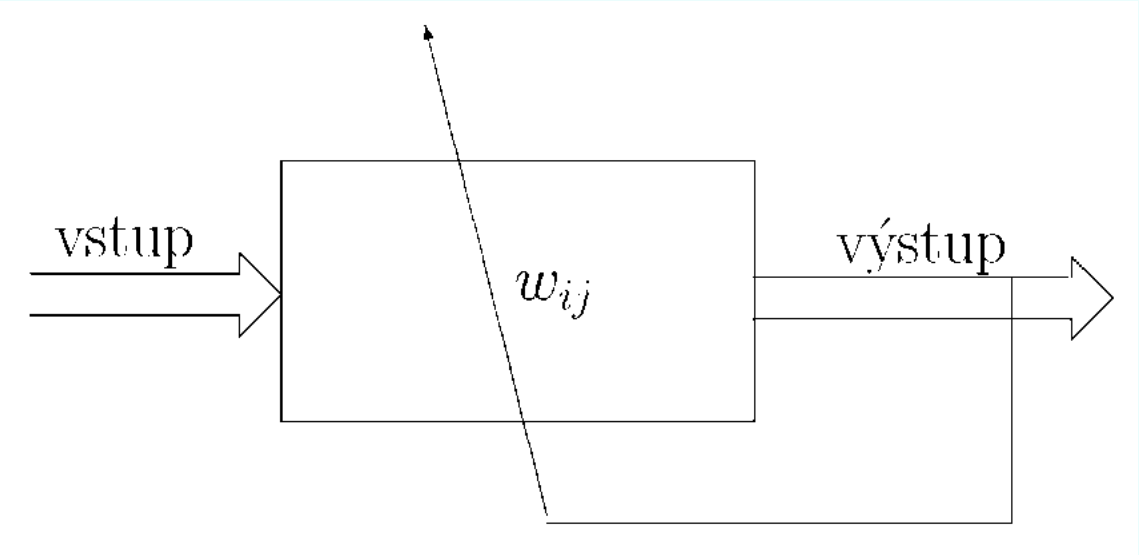
• učení s učitelem (supervised learning)

Neuronová síť se učí srovnáváním aktuálního výstupu s výstupem požadovaným (učitel) a nastavováním vah synapsí tak, aby se snížil rozdíl mezi skutečným a požadovaným výstupem.



• učení bez učitele (unsupervised learning)

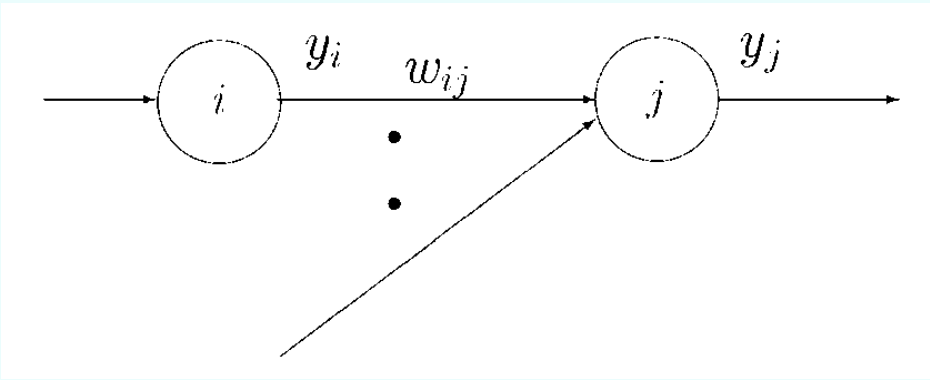
Váhy spojení se nastavují tak, aby výstup sítě byl konzistentní, tj. aby síť poskytovala stejnou odezvu při stejných, popř. podobných vstupních vektorech.



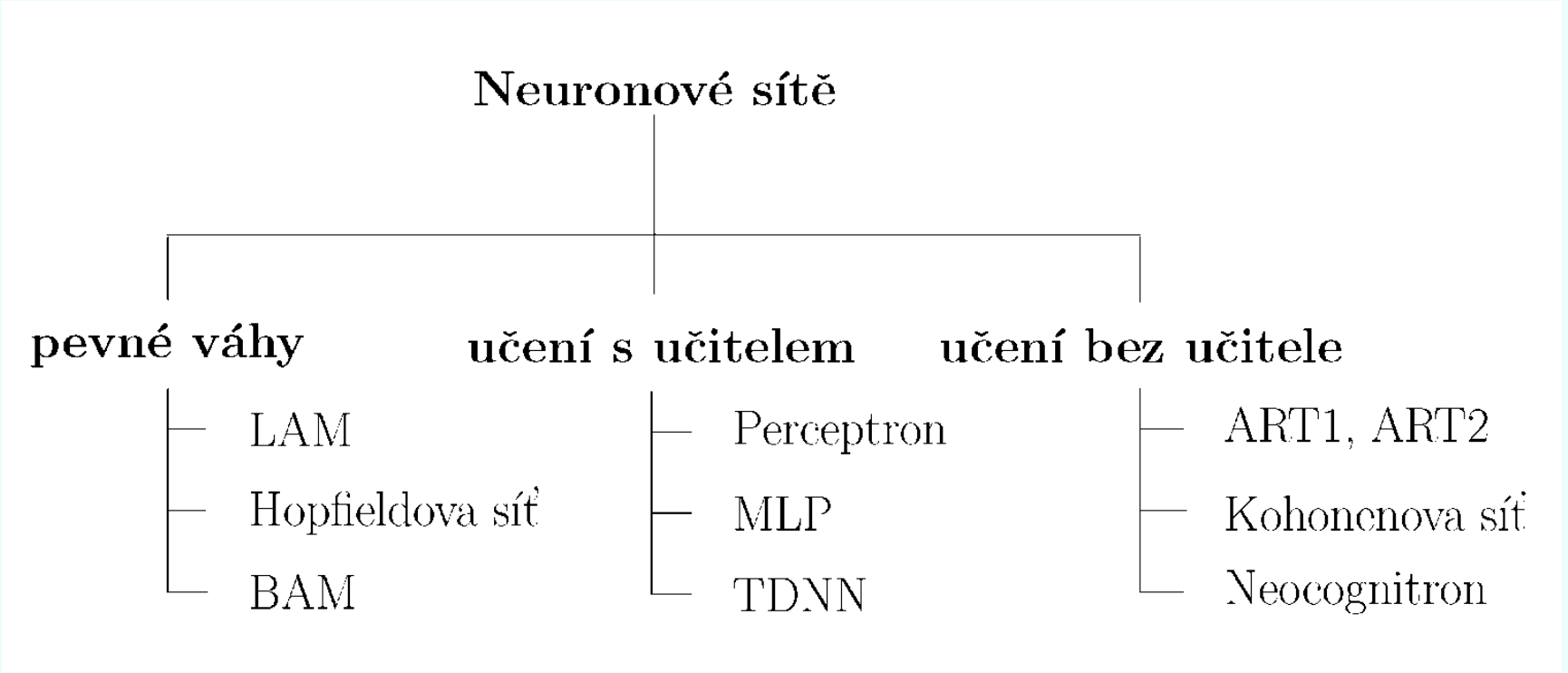
Základní model učení – D.O. Hebb (1949)

- učení bez učitele
- "síla" synaptického spojení mezi dvěma neurony se zvětší, pokud jsou oba tyto neurony aktivovány; změna hodnoty synaptického spojení je přímo úměrná součinu výstupních hodnot aktivovaných neuronů (α je tzv. koeficient učení)

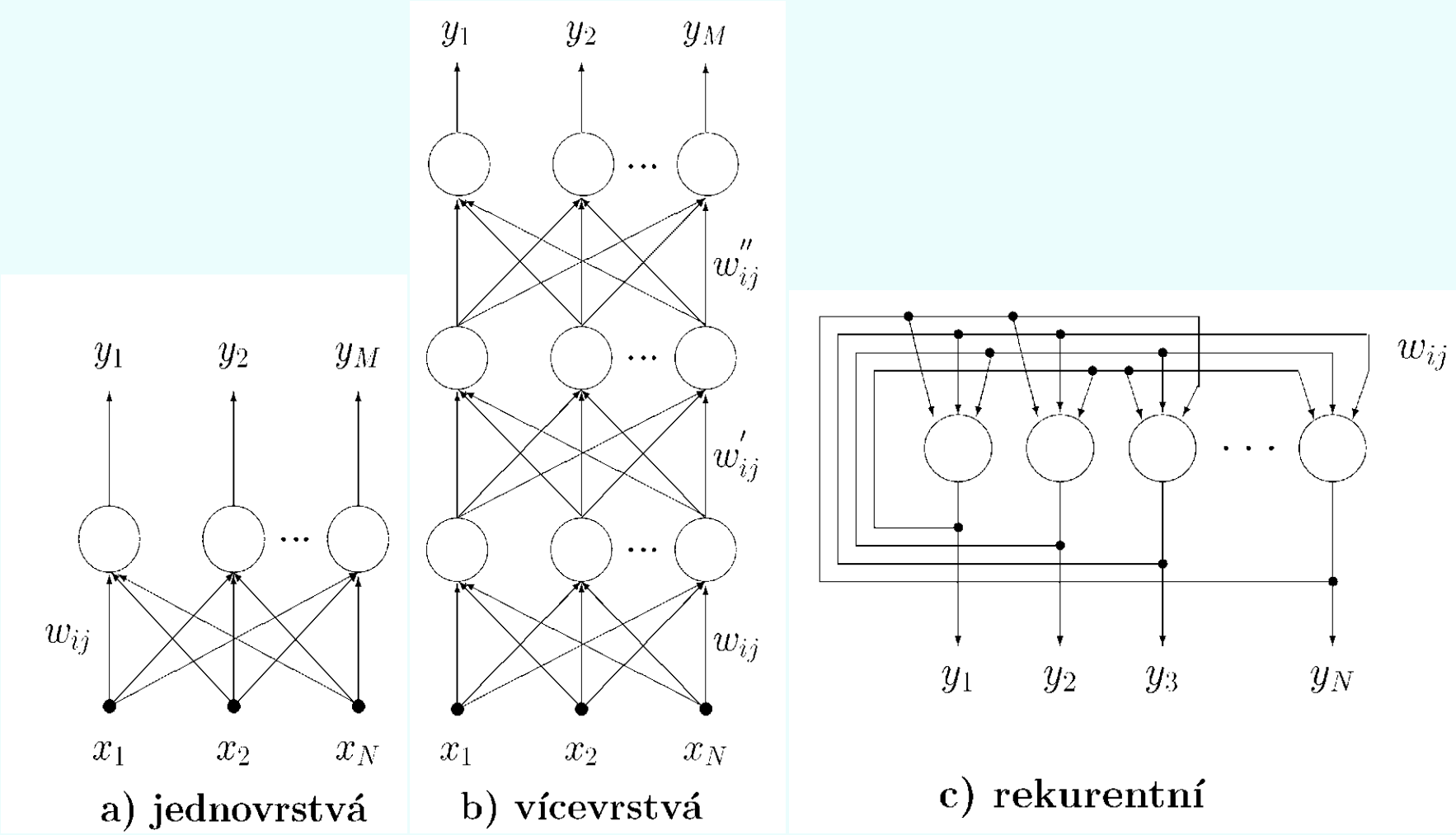
$$w_{ij}(t+1) = w_{ij}(t) + \alpha y_i y_j$$



Typy neuronových sítí

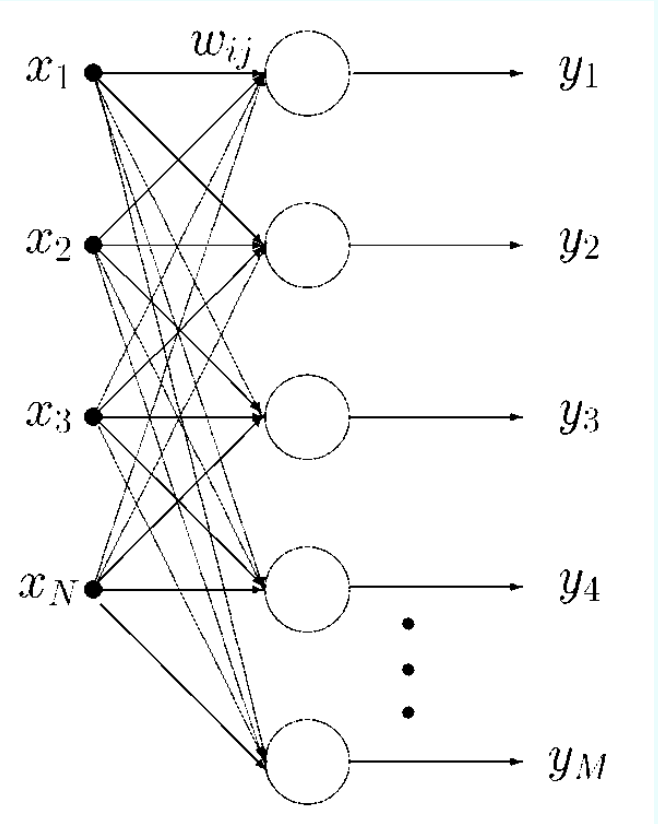


5. Neuronové sítě



Lineární asociativní paměť (LAM)

- jednovrstvá síť s dopředným šířením a pevnými vahami
- vstupní vektory mohou být binární (0/1), popř. reálné



Výstupní hodnoty neuronů:

- **pro reálné vstupy**

$$y_i = \sum_{j-1}^{N} w_{ij} x_j \,,\; pro\; i = 1, \ldots, M$$

- **pro binární vstupy**

$$y_i = f_H\left(\sum_{j-1}^{N} w_{ij} x_j - \Theta_i\right), \quad pro\; i = 1, \ldots, M$$

Použití:

- hetero-/autoasociativní paměť
- rekonstrukce neúplných a šumem poškozených obrazů

Reálné vstupy:

Nastavení váhových koeficientů pro asociační seznam L:

$$\boldsymbol{L} = \{(\boldsymbol{B^1}, \boldsymbol{A^1}), \dots, (\boldsymbol{B^P}, \boldsymbol{A^P})\}$$

vstup: $\boldsymbol{B^k} = [\boldsymbol{B_1^k}, \dots, \boldsymbol{B_N^k}]^{\boldsymbol{T}}$

výstup: $\boldsymbol{A^k} = [\boldsymbol{A_1^k}, \dots, \boldsymbol{A_N^k}]^{\boldsymbol{T}}$

$$\boldsymbol{w_{ij}} = \sum_{\boldsymbol{s=1}}^{\boldsymbol{P}} \boldsymbol{a_i^s b_i^s}, \qquad pro\ i = 1, \dots, N;\ j = 1, \dots, M$$

nebo maticově

$$\boldsymbol{W} = [\boldsymbol{w_{ij}}] = \boldsymbol{AB^T}$$

kde $\boldsymbol{A} = [\boldsymbol{M} \times \boldsymbol{P}] = [\boldsymbol{A^1}, \dots, \boldsymbol{A^P}]$, $\boldsymbol{B} = [\boldsymbol{NxP}] = [\boldsymbol{B^1}, \dots, \boldsymbol{B^P}]$

Odezva sítě na vstupní vektor $\boldsymbol{X} = [\boldsymbol{X^1}, \ldots, \boldsymbol{X^P}]$

$$\boldsymbol{Y} = \boldsymbol{WX} = \boldsymbol{AB^T X} = \sum_{\boldsymbol{s=1}}^{\boldsymbol{P}} (\boldsymbol{B^s X})\boldsymbol{A^s}$$

tj. výstup je lineární kombinace vektorů $\boldsymbol{A^s}$ s koeficienty určenými skalárním součinem $\boldsymbol{B^s X}$.

Jsou-li vstupní vektory ortonormální, je výstup shodný s obrazem uloženým v asociativní paměti.

Kapacita sítě (maximální počet uchovávaných vzorů):

$\boldsymbol{P} \leq \boldsymbol{N}$

kde $\boldsymbol{P}$ je počet vzorů uchovávaných v paměti, $\boldsymbol{N}$ je dimenze vstupního vektoru.

Binární vstupy:

Nastavení váhových koeficientů w_{ij} **a hodnot prahů** Θ_i pro asociační seznam L

$$w_{ij} = (2a_i^s - 1)(2b_j^s - 1), \quad pro\ i = 1, \dots, N;\ j = 1, \dots, M$$

$$\Theta_i = \frac{1}{2}\sum_{j=1}^{N} w_{ij}, \qquad pro\ i = 1, \dots, M$$

Odezva sítě

$$y_i = \begin{cases} 0 & \sum_{j=1}^{M}(w_{ij}x_i - \Theta_i) \leq 0 \\ 1 & \sum_{j=1}^{M}(w_{ij}x_i - \Theta_i) > 0 \end{cases}$$

Příklad:

Asociační seznam L = {([1011]^T, [1011]^T), ([0101]^T, [0101]^T)}

$$\boldsymbol{W} = (2\boldsymbol{A} - \mathbf{1})(2\boldsymbol{B} - \mathbf{1})^{\boldsymbol{T}} = \begin{bmatrix} \mathbf{1} & -\mathbf{1} \\ -\mathbf{1} & \mathbf{1} \\ \mathbf{1} & -\mathbf{1} \\ \mathbf{1} & \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{1} & -\mathbf{1} & \mathbf{1} & \mathbf{1} \\ -\mathbf{1} & \mathbf{1} & -\mathbf{1} & \mathbf{1} \end{bmatrix} = \begin{bmatrix} \mathbf{2} & -\mathbf{2} & \mathbf{2} & \mathbf{0} \\ -\mathbf{2} & \mathbf{2} & -\mathbf{2} & \mathbf{0} \\ \mathbf{2} & -\mathbf{2} & \mathbf{2} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{2} \end{bmatrix}$$

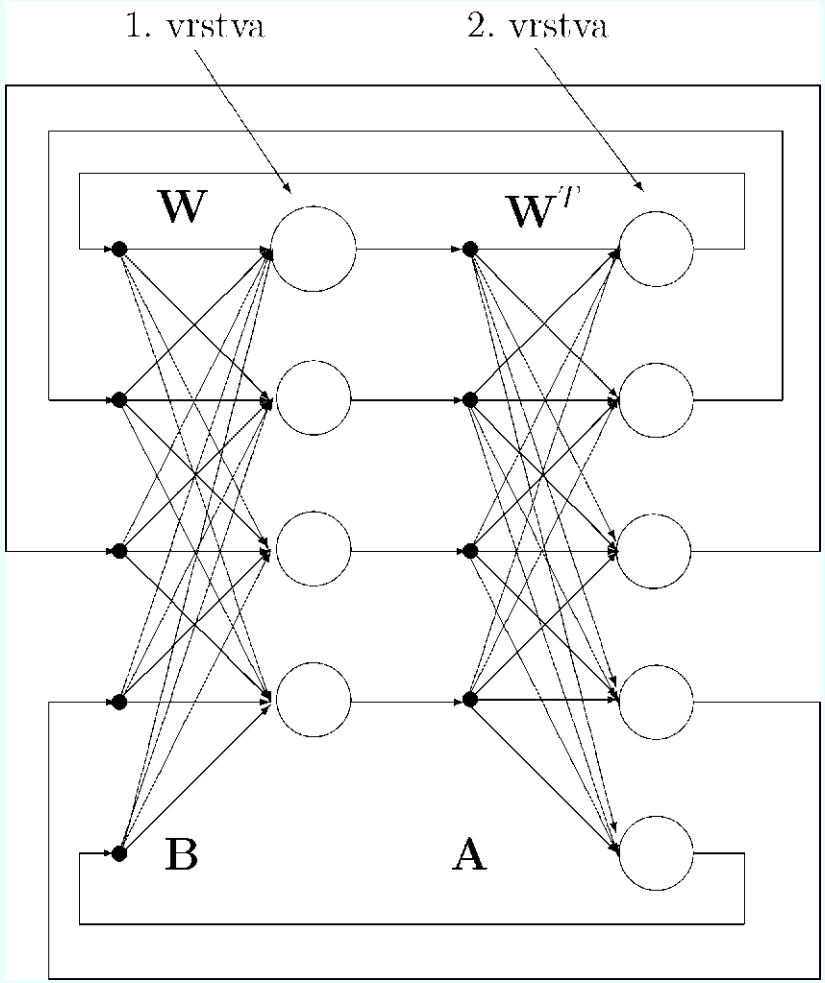
$$\boldsymbol{\Theta} = \frac{\mathbf{1}}{\mathbf{2}} \sum_{\boldsymbol{j}=\mathbf{1}}^{\boldsymbol{M}} \boldsymbol{w_{ij}} = [\mathbf{1} - \mathbf{1}\ \mathbf{1}\ \mathbf{1}]^{\boldsymbol{T}}$$

Testovací vektor $\boldsymbol{X} = [1101]^T$

$$\boldsymbol{Wx - \Theta} = \begin{bmatrix} \mathbf{2} & \mathbf{-2} & \mathbf{2} & \mathbf{0} \\ \mathbf{-2} & \mathbf{2} & \mathbf{-2} & \mathbf{0} \\ \mathbf{2} & \mathbf{-2} & \mathbf{2} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{2} \end{bmatrix} \begin{bmatrix} \mathbf{1} \\ \mathbf{1} \\ \mathbf{0} \\ \mathbf{1} \end{bmatrix} - \begin{bmatrix} \mathbf{1} \\ \mathbf{-1} \\ \mathbf{1} \\ \mathbf{1} \end{bmatrix} = \begin{bmatrix} \mathbf{-1} \\ \mathbf{1} \\ \mathbf{-1} \\ \mathbf{1} \end{bmatrix}$$

$$\boldsymbol{Y} = \boldsymbol{f_H}(\boldsymbol{Wx - \Theta}) = \begin{bmatrix} \mathbf{0} \\ \mathbf{1} \\ \mathbf{0} \\ \mathbf{1} \end{bmatrix}$$

Bidirectional Associative Memory (BAM)



BAM je

- dvouvrstvá rekurentní síť s pevnými vahami,
- vstupní vektory mohou být binární (0/1), popř. bipolární (-1/1)

Použití:

- heteroasociativní paměť
- rekonstrukce neúplných a šumem poškozených obrazů

Kapacita sítě (maximální počet uchovávaných vzorů):

Hrubý odhad: $P \leq n$, $n = \min\{N, M\}$,

kde N a M jsou počty neuronů v první a druhé vrstvě a P je maximální počet asociací.

Přesnější odhad: Pokud vybereme L vektorů tak, že

$$L < \frac{0,68n^2}{[log_2(n) + 4]^2}$$

a každý vektor má 4 + $\log_2$(n) prvků rovných 1 a zbytek 0, pak lze vytvořit BAM, pro kterou je 98% stavů stabilních.

Algoritmus trénování a činnosti BAM

Krok 1: Inicializace váhových koeficientů $\boldsymbol{w_{ij}}$ a prahové hodnoty $\boldsymbol{\Theta}$:

1. vrstva $\quad \boldsymbol{W} = [\boldsymbol{w_{ij}}] = \sum_{\boldsymbol{s=1}}^{\boldsymbol{P}} (\boldsymbol{2a_i^s - 1})(\boldsymbol{2b_j^s - 1})$

$$\boldsymbol{\Theta_i'} = \frac{\boldsymbol{1}}{\boldsymbol{2}} \sum_{\boldsymbol{j=1}}^{\boldsymbol{M}} \boldsymbol{w_{ij}} \ , \qquad i = 1, \dots, N$$

2. vrstva $\quad \boldsymbol{W}^{\boldsymbol{T}}$

$$\boldsymbol{\Theta_i''} = \frac{\boldsymbol{1}}{\boldsymbol{2}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} \boldsymbol{w_{ij}} \ , \qquad j = 1, \dots, M$$

kde $\boldsymbol{w_{ij}}$ jsou váhové koeficienty mezi i-tým neuronem první vrstvy a j-tým neuronem druhé vrstvy, $\boldsymbol{a_i^s, b_i^s}$ jsou i-té prvky s-tých vektorů asociačního seznamu (mohou nabývat pouze hodnot {0,1}), $\boldsymbol{P}$ je počet prvků asociačního seznamu, $\boldsymbol{\Theta'}$ a $\boldsymbol{\Theta''}$ jsou vektory prahových hodnot neuronů první a druhé vrstvy (pro homogenní BAM jsou tyto vektory nulové).

Krok 2: Předložení neznámého vstupního vektoru $\boldsymbol{B} = [b_1,\ldots b_n]^{\mathrm{T}}$, $\boldsymbol{B} \in \{0,1\}^N$

Krok 3: Iterace, dokud síť konverguje:

Během t-té iterace se pro

1. vrstvu:

1) vypočtou hodnoty $u'_j(t+1)$ paralelně pro $i = 1,2,...,M$ podle vztahu

$$\boldsymbol{u'_j(t+1) = \sum_{i=1}^{N} w_{ij} b_i(t) - \Theta'_j}$$

2) vypočtou výstupní hodnoty $a_j(t+1)$ pro $j = 1,2,...,M$ podle vztahu

$$\boldsymbol{a_j(t+1) = \begin{cases} 1 & pro\ u'_j(t+1) > 0 \\ 0 & pro\ u'_j(t+1) < 0 \\ a_j(t) & pro\ u'_j(t+1) = 0 \end{cases}}$$

2. vrstvu:

1) vypočtou hodnoty $u_j''(t+1)$ paralelně pro $j = 1,2,...,N$ podle vztahu

$$\boldsymbol{u_i''(t+1) = \sum_{j=1}^{M} w_{ij} b_j(t) - \Theta_i''}$$

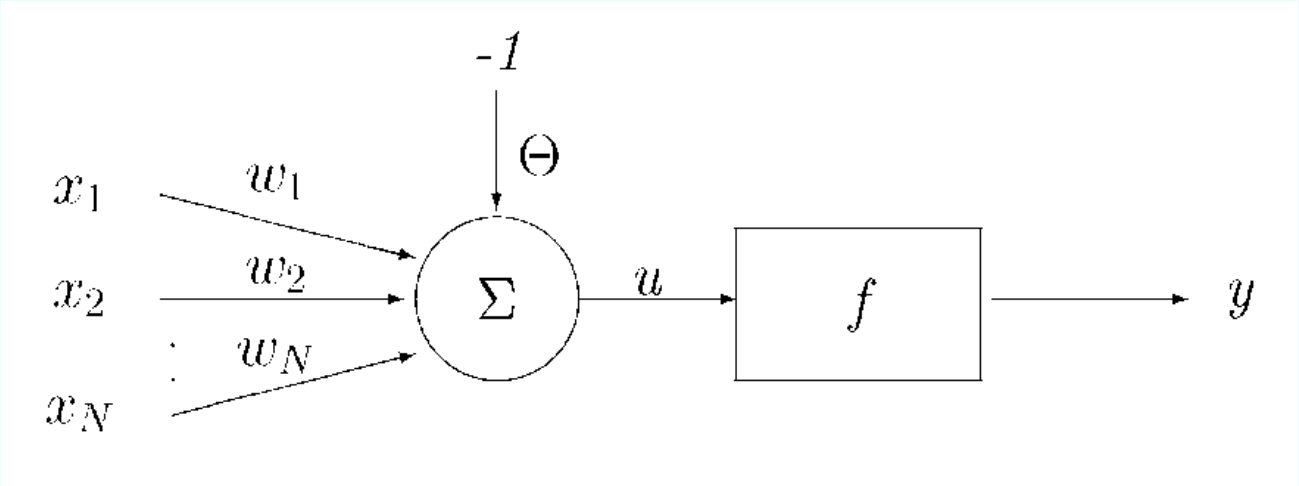
2) vypočtou výstupní hodnoty $b_i(t+1)$ pro $j = 1,2,...,N$ podle vztahu

$$\boldsymbol{b_i(t+1) = \begin{cases} 1 & pro\ u_i''(t+1) > 0 \\ 0 & pro\ u_i''(t+1) < 0 \\ b_i(t) & pro\ u_i''(t+1) = 0 \end{cases}}$$

Tento proces se opakuje, dokud se výstup první, popř. druhé vrstvy mění. Po ukončení iterací odpovídá vstup první vrstvy BAM vzorovému vektoru z asociačního seznamu, ktery se nejvíce podobá neznámému vektoru předloženému na vstup, výstup pak odpovídá asociaci uložené v BAM během trénování.

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

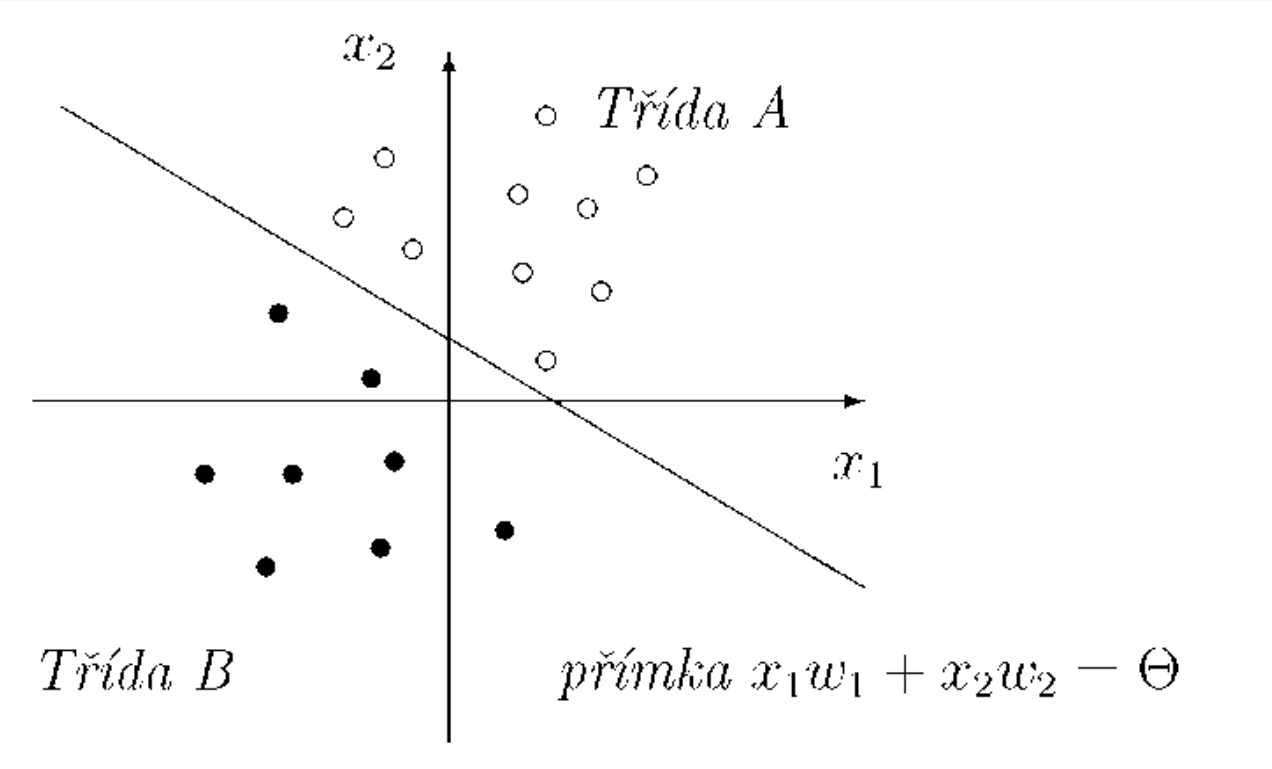
Jednoduchý perceptron



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

Příklad 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

$$y(t) = f_z\left(\sum_{i=1}^{N} w_i(t)x_i(t) - \Theta\right), \quad 1 \leq i \leq N$$

Krok 4: Adaptace vah

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

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

kde $\boldsymbol{d(t)}$ je požadovaný výstup, $\boldsymbol{\eta}$ 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 $\boldsymbol{w_i}$ a prahu $\boldsymbol{\Theta}$

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

Krok 3: Výpočet vnitřního potenciálu perceptronu podle vztahu

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

Krok 4: Výpočet chyby prezentace podle vztahu

$$\boldsymbol{E(t) = d(t) - u(t)}$$

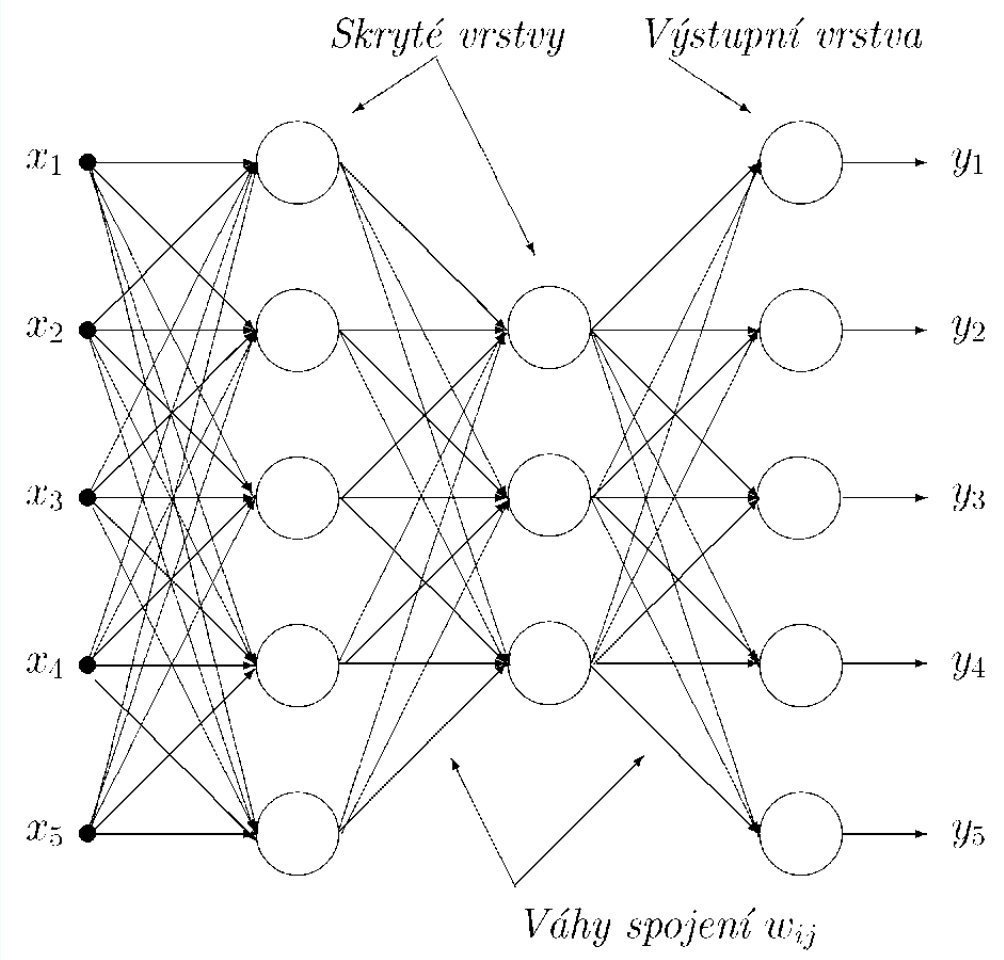
Krok 5: Adaptace vah

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

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

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

Vícevrstvý perceptron



Vícevrstvý perceptron je

- vícevrstvá síť s dopředným šířením
- učení: s učitelem
- neuronová aktivační funkce: sigmoidálná 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 alg.)

Krok 1: Počáteční inicializace vah $\boldsymbol{w_{ij}}$ a prahů $\boldsymbol{\Theta_i}$ jednotlivých neuronů.

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

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

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

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

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

Krok 4: Adaptace vah a prahů podle vztahu

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

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

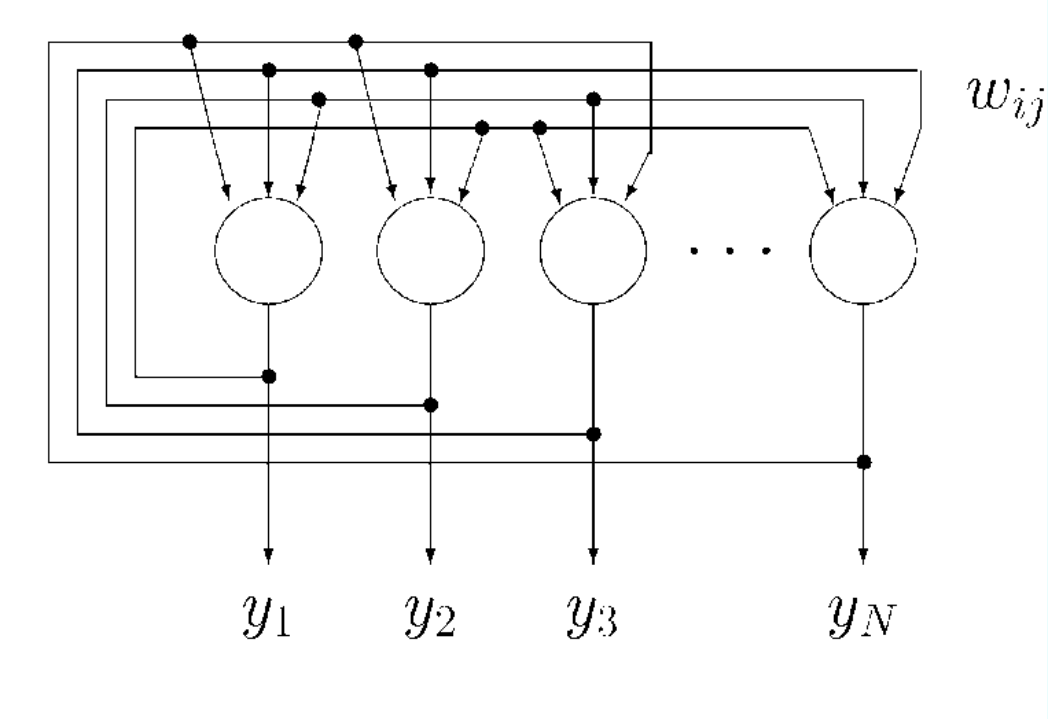
Nastavení vah začíná u výstupních uzlů a postupuje zpětně směrem ke vstupům. V uvedených vztazích jsou w_{ij} váhy mezi i-tým skrytým uzlem (popř. vstupním uzlem) a uzlem j-tým v čase *t*, x_i' je výstup *i*-tého, popř. vstupního uzlu, η je koeficient učení (zisk), α je tzv. momentový koeficient a δ_j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

$$\delta_j = \begin{cases} y_j(1-y_j)(d_j-y_j) & pro\ výstupní\ uzly \\ x_j'(1-x_i)\left(\sum_k \delta_k w_{jk}\right) & pro\ vnitřní\ skryté\ uzly \end{cases}$$

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.

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 $$f_H(u) = \begin{cases} 1 & \text{pro } u > 0 \\ 0 & \text{pro } u < 0 \end{cases}$$
- znaménková funkce (pro bipolární vstupní vektory)

Použití:

- autoasociativní paměť
- rekonstrukce neúplných a šumem poškozených obrazů
- optimalizační problémy

Kapacita sítě (maximální počet uchovaných vzorů):

$$P \leq 0,15\,N,$$

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

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 (synchronní model)

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

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

$$\Theta_i = \frac{1}{2} \sum_{j=1}^{N} w_{ij}, \quad i = 1, \ldots, N,$$

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 $X = [x_1, \ldots x_N]^T$, $x_i \in \{0,1\}$

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

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

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

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

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

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

Příklad: Hopfieldova síť (synchronní model)

Dány trénovací vektory $\alpha^1 = [1110]^T$, $\alpha^2 = [1100]^T$

$$w_{ij} = \sum_{s=1}^{P}(2a_i^s - 1)\left(2a_j^s - 1\right), \quad i \neq j\,, \quad 1 \leq i,j \leq N, \quad \text{tj.}$$

$$\boldsymbol{W} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & -1 \\ -1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & -1 \\ 1 & 1 & -1 & -1 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 0 & -2 \\ 2 & 2 & 0 & -2 \\ 0 & 0 & 2 & 0 \\ -2 & -2 & 0 & 2 \end{bmatrix}$$

$$\boldsymbol{\theta} = \frac{1}{2}\sum_{j=1}^{M} w_{ij} = [1\ 1\ 1\ -1]^T$$

Pro testovací vektor $\boldsymbol{y}(0) = \boldsymbol{x} = [0100]^T$ dostaneme:

a) v 1. iteraci:

$$\boldsymbol{Wx} - \boldsymbol{\Theta} = \begin{bmatrix} 2 & 2 & 0 & -2 \\ 2 & 2 & 0 & -2 \\ 0 & 0 & 2 & 0 \\ -2 & -2 & 0 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix}$$

$$\boldsymbol{y}(1) = f_H(\boldsymbol{Wx} - \boldsymbol{\Theta}) = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

b) v 2. iteraci:

$$\boldsymbol{Wx} - \boldsymbol{\Theta} = \begin{bmatrix} 2 & 2 & 0 & -2 \\ 2 & 2 & 0 & -2 \\ 0 & 0 & 2 & 0 \\ -2 & -2 & 0 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \\ -1 \\ -3 \end{bmatrix}$$

$$\boldsymbol{y}(2) = f_H(\boldsymbol{Wx} - \boldsymbol{\Theta}) = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Hopfieldova síť (asynchronní model)

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

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

$$\Theta_i = \frac{1}{2} \sum_{j=1}^{N} w_{ij}, \qquad i = 1, \ldots, N,$$

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 $X = [x_1, ..., x_N]^T$, $x_i \in \{0,1\}$

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

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*

1)

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

2)

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

Krok 4: Opakování od kroku 2 pro nový vstupní vektor

Příklad: Hopfieldova síť (asynchronní model)

Dány trénovací vektory $\alpha^1 = [1110]^T$, $\alpha^2 = [1100]^T$

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

$$\boldsymbol{W} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & -1 \\ -1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & -1 \\ 1 & 1 & -1 & -1 \end{bmatrix} = \begin{bmatrix} \mathbf{0} & 2 & 0 & -2 \\ 2 & \mathbf{0} & 0 & -2 \\ 0 & 0 & \mathbf{0} & 0 \\ -2 & -2 & 0 & \mathbf{0} \end{bmatrix}$$

$$\boldsymbol{\Theta} = \frac{1}{2} \sum_{j=1}^{M} w_{ij} = [0\ 0\ 0\ -2]^T$$

Pro testovací vektor $\boldsymbol{y}(0) = \boldsymbol{x} = [0100]^T$ dostaneme:

a) v 1. iteraci:

$$\boldsymbol{y}(1) = f_H(\boldsymbol{W}\boldsymbol{x} - \boldsymbol{\Theta}) = \begin{bmatrix} 0 & 2 & 0 & 2 \\ 2 & 0 & 0 & -2 \\ 0 & 0 & 0 & 0 \\ -2 & -2 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ -2 \end{bmatrix}$$ postupně pro

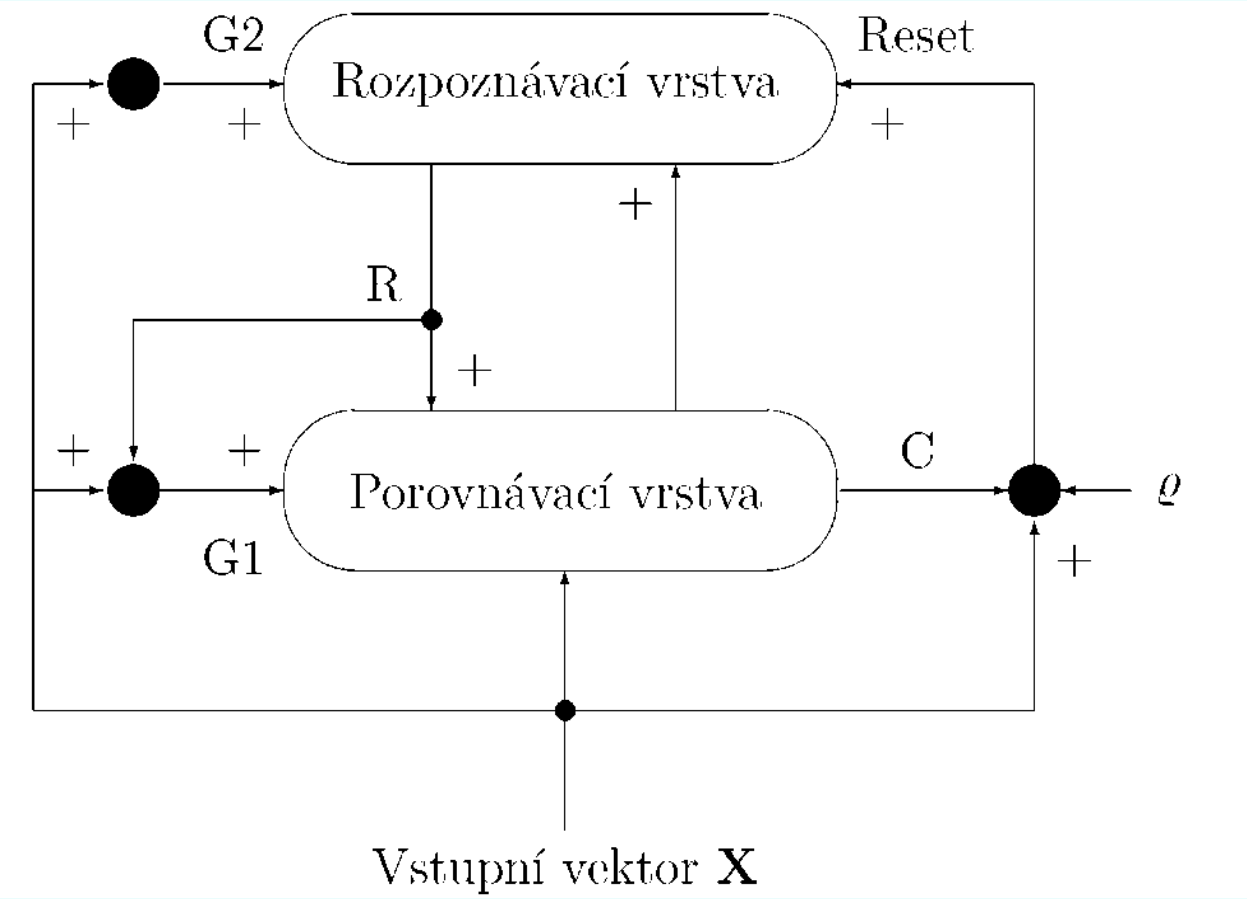
1. bit: $[1100]^T$
2. bit: $[1100]^T$
3. bit: $[1100]^T$
4. bit: $[1100]^T$

b) v 2. iteraci:

$$\boldsymbol{y}(2) = f_H(\boldsymbol{W}\boldsymbol{x} - \boldsymbol{\Theta}) = \begin{bmatrix} 0 & 2 & 0 & 2 \\ 2 & 0 & 0 & -2 \\ 0 & 0 & 0 & 0 \\ -2 & -2 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ -2 \end{bmatrix}$$ postupně pro

1. bit: $[1100]^T$
2. bit: $[1100]^T$
3. bit: $[1100]^T$
4. bit: $[1100]^T$

ART (Adaptive Resonance Theory)



ART je:

- 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 apod.

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**.

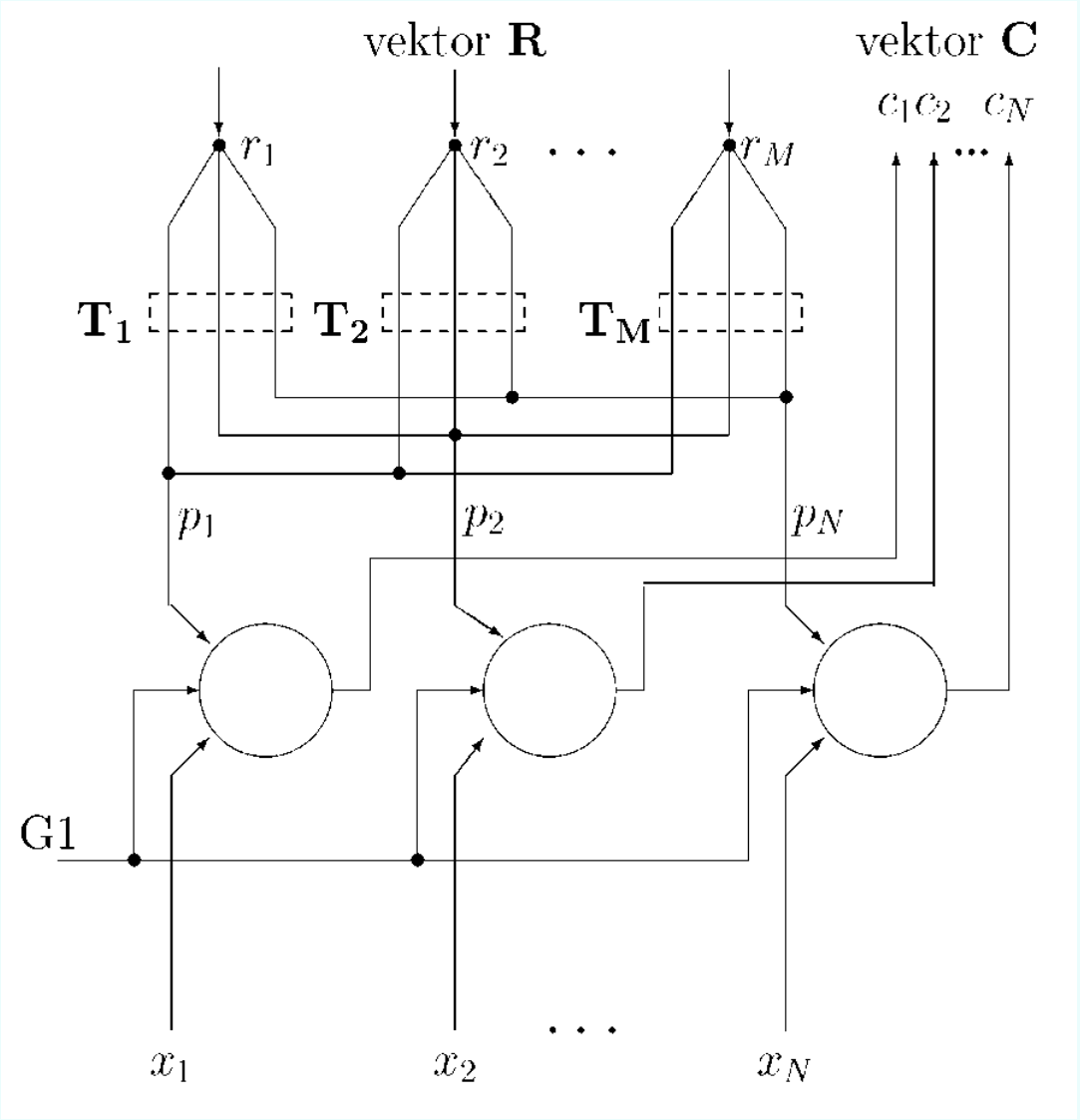
Porovnávací vrstva

Každý neuron v této vrstvě přijímá tři binární signály:

1) složky x_i vstupního vektoru $\mathbf{X} = [x_1, x_2, ..., x_N]^T$,
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.

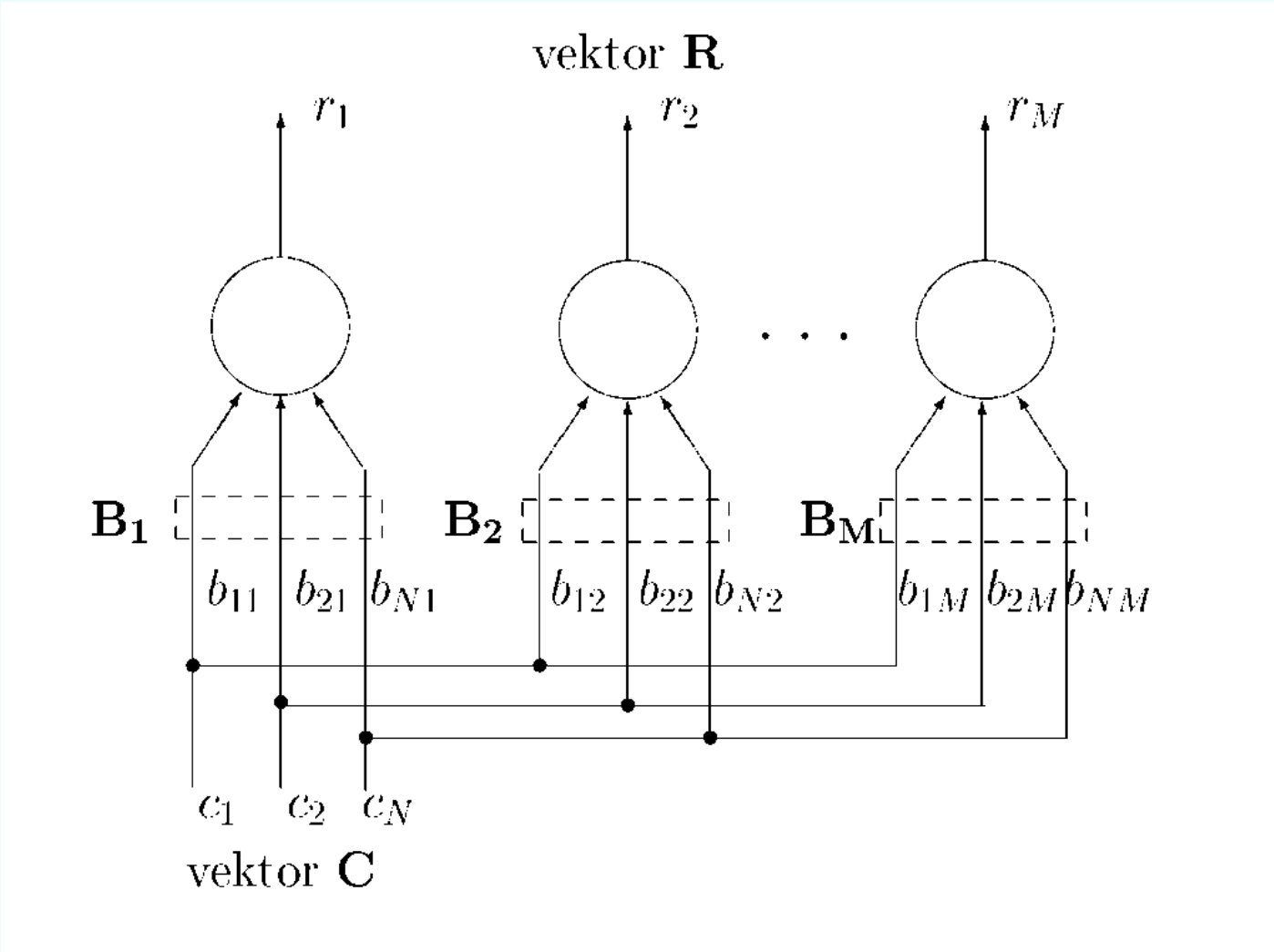
Na počátku jsou všechny složky vektoru **R** a signál **G1** nastaveny na hodnotu 0 => **C = X**.


vektor R
vektor C
r_1
r_2
r_M
c_1 c_2 c_N
$\mathbf{T_1}$
$\mathbf{T_2}$
$\mathbf{T_M}$
p_1
p_2
p_N
G1
x_1
x_2
x_N

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 $\boldsymbol{B_j}$ $\left(b_{ji} \in \boldsymbol{R}\right)$.

Po předložení vstupního vektoru je v rozpoznávací vrstvě aktivován pouze neuron s vektorem váhových koeficientů nejvíce podobným vstupu. Ostatní neurony budou mít výstup nulový.


vektor R
r_1
r_2
r_M
$\mathbf{B_1}$
$\mathbf{B_2}$
$\mathbf{B_M}$
b_{11} b_{21} b_{N1}
b_{12} b_{22} b_{N2}
b_{1M} b_{2M} b_{NM}
c_1 c_2 c_N
vektor C

Algoritmus použití ART sítě

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

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

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

$$\boldsymbol{i = 1,2,\ldots,N,}$$

$$\boldsymbol{j = 1,2,\ldots,M,}$$

$$\boldsymbol{0 \leq \rho \leq 1}$$

Krok 2: Přivedení vektoru $\boldsymbol{X} = [x_1, ..., x_N]^T$ na vstup sítě

Krok 3: Výpočet odezvy neuronu v rozpoznávací vrstvě podle vztahu

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

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 $\mu \geq \rho$, 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

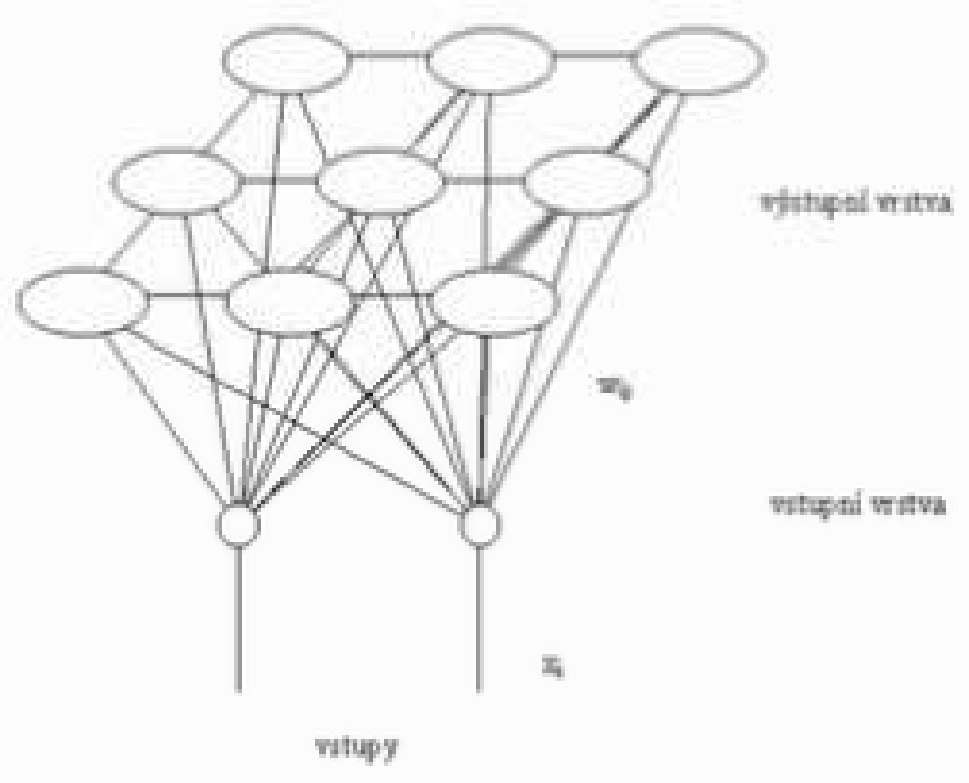
Krok 7: Adaptace vah u neuronu s největší odezvou. V tomto kroku se modifikují váhy spojení t_{ik} a b_{ik} podle vztahů

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

$$b_{ik}(t+1) = \frac{t_{ik}(t)x_i}{0,5 + \sum_{i=1}^{N} t_{ik}(t)x_i}$$

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

Kohonenova síť (Self Organizing Map – SOM)



Kohonenova síť je

- jednovrstvá síť s dopředným šířením
- učení: bez učitele
- vstupní vektor: $\boldsymbol{X} = [x_i, ...,x_N]^T$, $x_i \in \boldsymbol{R}$
- výstupní hodnota neuronu je definována jako vzdálenost mezi vstupním a váhovým vektorem, t.j.

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

Použití pro:

- shlukování
- analýzu dat
- vytváření sémantických map aj.

Algoritmus (Kohonenova síť)

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

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

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

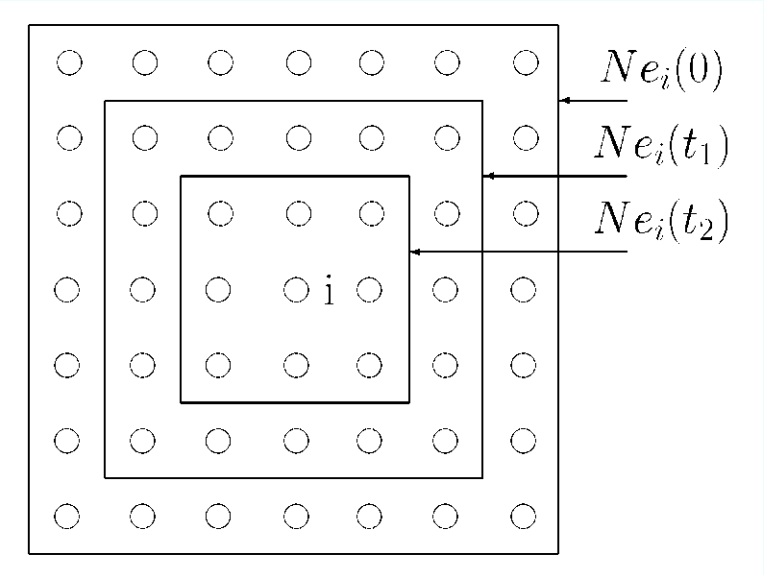
Krok 3: Výběr i-tého neuronu s nejmenší vzdáleností d_i (tzv. vítězného neuronu)

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

$$\boldsymbol{w_{ki}(t+1) = w_{ki}(t) + \eta(t)\left(x_i(t) - w_{ki}(t)\right)}, \quad \forall k \in Ne_i(t)\,,\ i = 1,2\ldots, N,$$

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.

Neocognitron

- vícevrstvá hierarchická neuronová síť

Použití: např. rozpoznávání rukou psaných znaků

Výhoda: schopnost správně rozpoznávat nejen naučené obrazy, ale i obrazy, které vzniknou částečným posunutím, otočením nebo jinou deformací

5. Neuronové sítě