Fakulta elektrotechniky a informatiky
VŠB - Technická univerzita Ostrava

Neuronové sítě

prof. Ing. Ivo Vondrák, CSc.

katedra informatiky

Tento text vznikl pro potřeby výuky předmětu Neuronové sítě z původních zdrojů skrip ***Umělá inteligence a neuronové sítě*** *(ISBN 80-7078-259-5, VŠB-TU Ostrava, 2001). Jedná se o zkrácenou verzi obsahující pouze kapitolu věnovanou neuronovým sítím. Z důvodů nekompatibility původních zdrojů došlo k technické (nikoliv faktické) revizi textu, zejména vložených obrázků. Vzhledem k této skutečnosti se mohlou v textu vyskytovat přehlédnuté chyby způsobené konverzí. Prosím proto čtenáře o shovívavost a doufám, že i přesto jim bude tento text užitečný.*

Ing. Dušan Fedorčák
V Ostravě, duben 2009

Obsah

OBSAH **3**

NEURONOVÉ SÍTĚ **4**

Model neuronu *5*
Neuron prvé generace (McCulloch) *6*
PERCEPTRON 7
Adaptace perceptronu (Hebb) *8*
SPOJITÝ PERCEPTRON 9
VÍCEVRTSVÉ SÍTĚ A METODA BACKPROPAGATION 10
Spojitý perceptron s intervalovým stavem excitace *16*
Zobecněná metoda Back Propagation *17*
REKURENTNÍ VÍCEVRSTVÉ NEURONOVÉ SÍTĚ 18
KOMPETICE, KOHONENOVO UČENÍ A SAMOORGANIZACE 24
Kohonenovy mapy *26*
Kvantování vektorů učením *28*
Counter-Propagation *30*
HOPFIELDOVA SÍŤ 32
Funkce energie *33*
Ukládání vzorů *34*
Proces vyvolání informace *35*
Boltzmannův stroj *37*
Řešení omezujících podmínek *39*
Dvousměrná asociativní paměť *41*
ADAPTIVNÍ REZONANČNÍ TEORIE 44
OBJEKTOVĚ-ORIENTOVANÝ PŘÍSTUP K NEURONOVÝM SÍTÍM 48
Analýza problému - OOA *48*
Návrh a implementace - OOD, OOP *49*
Hierarchie neuronů *50*
Hierarchie vazeb *51*
Hierarchie tříd Interconnections *52*
Hierarchie neuronových sítí *53*

POUŽITÁ LITERATURA **55**

Kapitola 10

Neuronové sítě

Specifika neuronových sítí lze v hrubých rysech vyjádřit v následujících několika bodech:

- *Neuronové sítě jsou inspirovány biologickými neuronovými sítěmi*. Tato vlastnost určitým způsobem předurčuje, že uměle vytvořené neuronové sítě by měly být schopny, z hlediska základních principů, se chovat stejně nebo alespoň podobně jako jejich biologické vzory. Je zřejmé, že vytvoření umělého lidského mozku se všemi jeho schopnostmi je věc jen velmi těžce řešitelná ať už z hlediska kvantity jeho neuronů či jejich způsobu propojení, chování jednotlivých typů neuronů apod. Nicméně skýtá se tu šance simulovat alespoň některé funkce lidského myšlení a tyto pak implementovat.

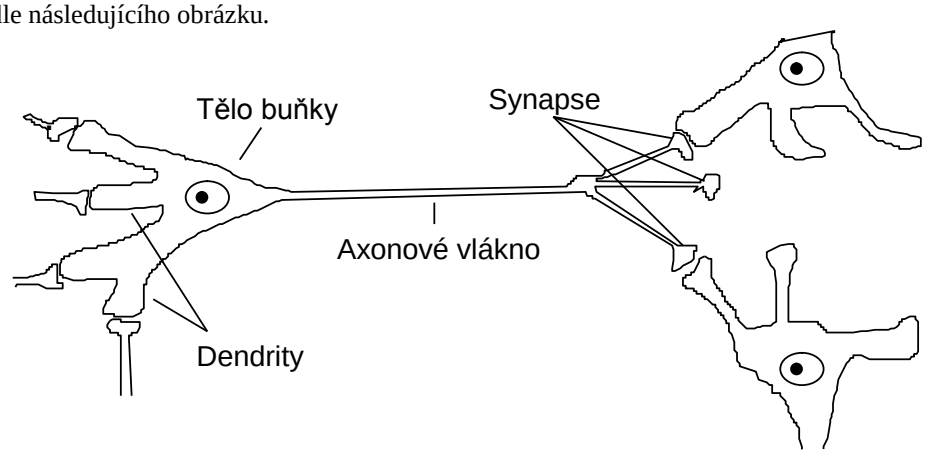
- *Neuronové sítě využívají distribuované, paralelní zpracování informace při provádění výpočtů*. Jinými slovy ukládání, zpracování a předávání informace probíhá prostřednictvím celé neuronové sítě spíše než pomocí určitých paměťových míst. Tedy paměť a zpracování informace v neuronové síti je ve své přirozené podstatě spíše *globální* než *lokální*.

- *Znalosti jsou ukládány především prostřednictvím síly vazeb mezi jednotlivými neurony*. Vazby mezi neurony vedoucí ke "správné odpovědi" jsou posilovány a naopak, vazby vedoucí k "špatné odpovědi" jsou oslabovány pomocí opakované expozice příkladů popisujících problémový prostor.

- *Učení je základní a podstatná vlastnost neuronových sítí*. Tento fakt zjevně vyjadřuje základní rozdíl mezi dosud běžným použitím počítačům a použitím prostředků na bázi neuronových sítí. Jestliže jsme doposud veškeré své usilí při tvorbě uživatelských programů soustředili na vytvoření algoritmů, které transformují vstupní množinu dat na množinu dat výstupních, pak neuronové sítě již tuto náročnou fázi nepotřebují. Jakým způsobem se budou vstupní data transformovat na data výstupní určuje právě fáze učení založená na již dříve uvedené expozici vzorků (příkladů) popisující řešenou problematiku - *trénovací množina*. Odpadá tedy nutnost algoritmizace úlohy, které je nahrazena předložením trénovací množiny neuronové síti a jejím učením.

Model neuronu

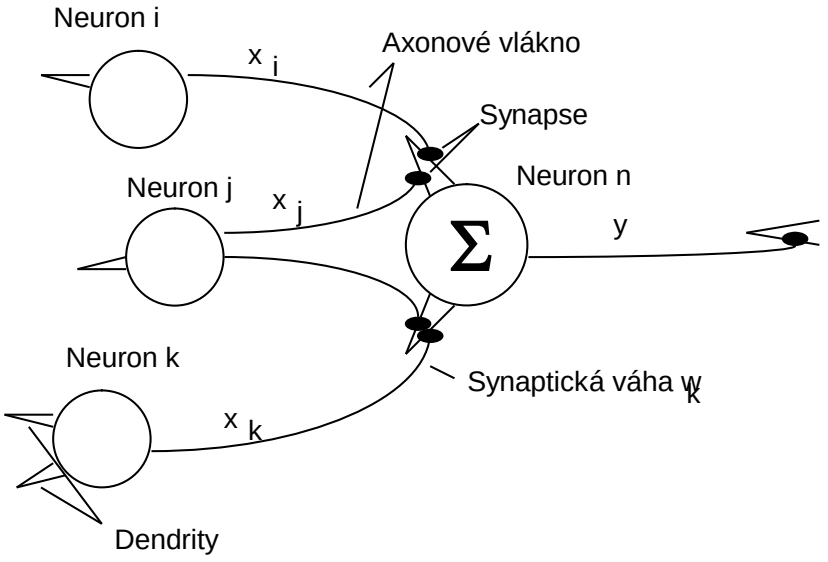
Uměle vytvořený neuron je dán svým biologickým vzorem a tvoří jakousi základní "výpočetní jednotku" složitějšího komplexu - neuronové sítě. Zjednodušeně lze biologický neuron znázornit a popsat dle následujícího obrázku.



Obr. 10-1. Biologický vzor

1. *Dendrity* reprezentují místo vstupu signálů do těla neuronu.
2. *Tělo buňky* sčítá signály dané okolními neurony. Takto stanovený vnitřní potenciál vede k excitaci (vybuzení) neuronu.
3. *Axonové vlákno* přenáší signál daný stupněm excitace k synapsím.
4. *Synapse* tvoří výstupní zařízení neuronů, které signál zesilují či zeslabují a předávají dalším neuronům.

Výše uvedený model pak lze vyjádřit schematicky asi takto

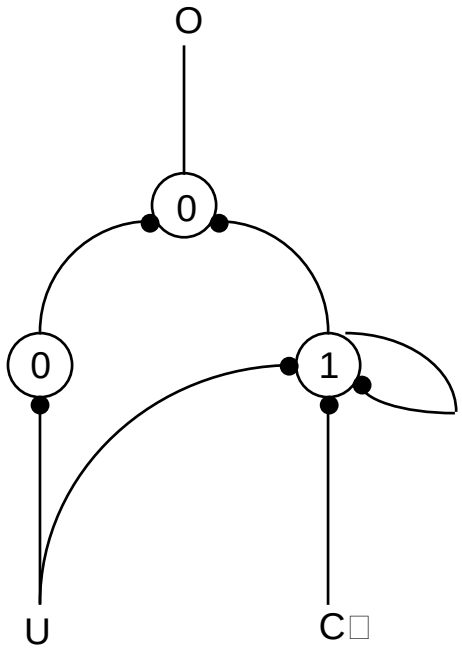


Obr. 10-2. Schematický model neuronu

kde $x_{i,j,k}$ výstupní signál neuronů i,j,k
$w_{i,j,k}$ synaptické váhy upravující výstupní signál neuronů i,j,k
Σ
y výstupní excitační signál neuronu n.

Neuron prvé generace (McCulloch)

Jeden z prvních modelů neuronu byl navržen McCullochem a jeho zjednodušení spočívalo v zavedení excitačních resp. inhibičních vazeb neuronu. První typ vazby je reprezentován synaptickou vahou rovné hodnotě +1 (označení plnou značkou) a druhý pak hodnotou 0 (označení prázdnou značkou). Každý neuron má definovanou svou vnitřní hodnotu prahu, která musí být překonána vnitřním potenciálem neuronu, aby došlo k jeho vybuzení (hodnota výstupního signálu 1). Tento jednoduchý způsob definice neuronu umožňuje modelovat různé procesy, jako například níže uvedený podmíněný reflex.



Obr. 10-3. Podmíněný reflex

Model se skládá z tří neuronů s definovanu hodnotou prahů, dvou vstupů (nepodmíněný *U-unconditioned* a podmíněný *C-conditioned*), jednoho výstupu (podmíněný reflex *O-output*) a pouze z excitačních vazeb. Pokud bychom pro přenos signálu v neuronové síti zavedli hodiny, pak jednotlivé takty reprezentované tabulkou dokumentují proces vyvolání podmíněného reflexu (takt hodin 6 a 7), kdy vstupní signál je přiveden pouze na vstup C a následuje odezva systému rovna hodnotě 1.

T	**U**	**C**	**L**	**P**	**O**
1	1	0	1	0	0
2	0	0	0	0	1
3	0	1	0	0	0
4	0	0	0	0	0
5	1	1	1	1	0
6	0	1	0	1	1
7	0	0	0	0	1

Podmíněný reflex: U,C - vstupy systému, L,P - levý a pravý neuron, O - výstupní signál třetího neuronu

Perceptron

Jeden z nejdůležitějších modelů dodnes používaných je tzv. *perceptron*, jehož potenciál je definovaný jako vážený součet vstupujících signálů. Pokud tento vnitřní potenciál neuronu překoná jeho prahovou hodnotu (definovánou v našem případě symbolem ϑ), dojde k excitaci neuronu na hodnotu 1. V opačném případě je neuron inhibitován, což je reprezentováno hodnotou 0. Matematicky lze tento postup vyjádřit pomocí funkce signum tímto způsobem:

$$y = Sgn\left(\sum_{i=1}^{n} w_i x_i - \vartheta\right),$$

$$Sgn(x) = 1 \quad x > 0,$$

$$Sgn(x) = 0 \quad x \le 0,$$

zavedením stálého neuronu na vstupu se stavem excitace $x_0 = 1$ a vazbou k našemu neuronu $w_0 = -\vartheta$ lze předchozí zjednodušit takto:

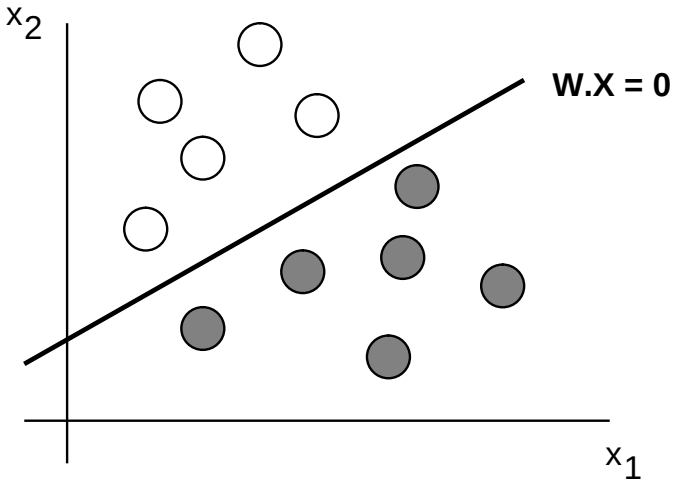
$$y = Sgn\left(\sum_{i=0}^{n} w_i x_i\right).$$

Pokud provedeme analýzu výrazu v závorce tak, že jej položíme roven nule získáme rovnici nadroviny (v dvourozměrném případě reprezentovanou přímkou). Tedy pomocí vektorového zápisu

$$\sum_{i=0}^{n} w_i x_i = \overline{W} \cdot \overline{X},$$

$$\overline{W} \cdot \overline{X} = 0.$$

Tato rovina rozděluje vstupní prostor na dva poloprostory (viz obrázek). Jinými slovy, jsme schopni prostřednictvím perceptronu rozlišit dvě třídy vstupů. Jedné z nich odpovídá excitace rovna hodnotě 1 a druhé inhibice neuronu daná hodnotou 0.



Obr.10-4. Rozpoznávání pomocí perceptronu (dvourozměrný případ)

Otázkou nyní je, jak stanovit hodnoty vah neuronu, aby byl schopen správně rozpoznávat (přiřazovat do tříd) předložené vstupy. K tomu je potřebné náš perceptron adaptovat na základě trénovací množiny prostřednictvím nějakého algoritmu. Jeden z nejznámějších principů je adaptace (učení) neuronu podle Hebbova pravidla, které je definováno následujícím způsobem:

Adaptace perceptronu (Hebb)

1. Inicializace vah a prahu náhodnými malými čísly

   $w_i(t), (0 \le i \le n)$ je váha vstupu i v čase t.

2. Předložení vstupu a požadovaného výstupu z trénovací množiny

   $\left(x_0 \quad x_1 \quad \ldots \quad x_n\right) \rightarrow d(t).$

3. Stanovení skutečné odezvy

   $$y(t) = Sgn\left(\sum_{i=0}^{n} w_i(t)x_i(t)\right).$$

4. Adaptace
   výstup je správný $w_i(t+1) = w_i(t),$
   výstup 0 a měl být 1 $w_i(t+1) = w_i(t) + x_i(t),$
   výstup 1 a měl být 0 $w_i(t+1) = w_i(t) - x_i(t).$

Tento poslední krok algoritmu lze modifikovat pomocí multiplikativního faktoru, který může pozměnit razanci změn hodnot adaptovaných vah následujícím způsobem:

4. Adaptace vah
   výstup je správný $w_i(t+1) = w_i(t),$
   výstup 0 a měl být 1 $w_i(t+1) = w_i(t) + \eta x_i(t),$
   výstup 1 a měl být 0 $w_i(t+1) = w_i(t) - \eta x_i(t),$

kde $0 \le \eta \le 1$ je koeficient učení (*learning rate*) ovlivňující proces adaptace.

Pokud v dalším zavede chybu odezvy Δ definovanou jako rozdíl požadované a skutečné odezvy:

$$\Delta = d(t) - y(t),$$

pak můžeme adaptaci vah zobecnit následujícím předpisem:

4. Adaptace vah

   $$\Delta = d(t) - y(t),$$
   $$w_i(t+1) = w_i(t) + \eta \Delta x_i(t).$$

Další algoritmus umožňující adaptaci vah neuronu je obdobný minulému a navrhli jej Widrow a Hoff. Jeho podstata spočívá ve snaze kvantifikovat chybu neuronu, jehož vstupem je vektor reálných hodnot a odezvou reálné číslo dané hodnotou potenciálu neuronu, tedy

$$y(t) = \sum_{i=0}^{n} w_i(t)x_i(t).$$

Tento lineární neuron (ADALINE - ADAptive LINear Element) umožňuje přijetí vcelku zřejmého principu adaptace, že bude nejlepší velkou měrou změnit váhy v případě, že jejich vážený součet je velmi vdálen od požadovaného výstupu neuronu. Tento přístup pak vedl k vytvoření tzv. *Widrow-Hoffova delta pravidla*, které vypočítává rozdíl mezi potenciálem neuronu a jeho požadovanou odezvou ve formě:

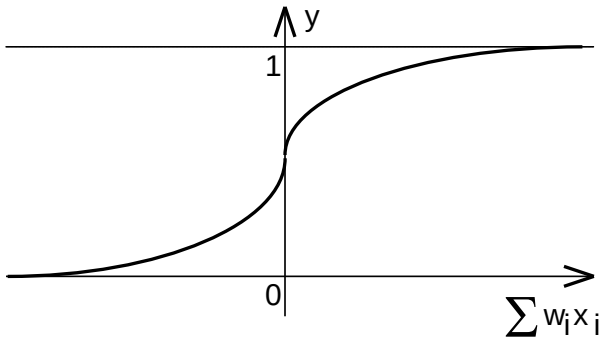
$$\Delta = d(t) - \sum_{i=0}^{n} w_i(t)x_i(t).$$

Vlastní adaptace vah pak probíhá podle již známého vztahu

$$w_i(t+1) = w_i(t) + \eta \Delta x_i(t).$$

Spojitý perceptron

Hodnota excitace *y*, jak již bylo dříve uvedeno, je dána tzv. aktivační funkcí neuronu. V tomto případě má tvar sigmoidy, jejíž průběh je zobrazen na následujícím obrázku.



Obr. 10-5. Aktivační funkce neuronu

Formálně lze tuto funkci výjadřit pomocí následujícího matematického vztahu:

$$y = S(z) = \frac{1}{1+e^{-\lambda z}},$$

$$z = \sum_{i=0}^{n} w_i x_i,$$

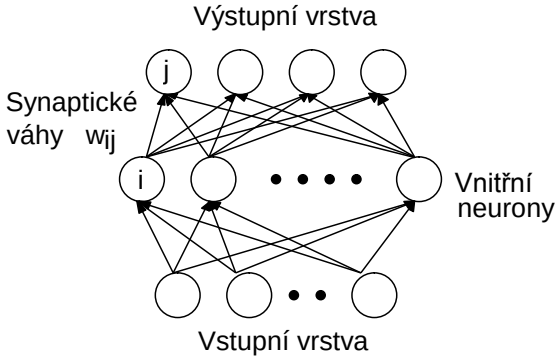
kde
- $S(z)$ aktivační funkce neuronu
- z vnitřní potenciál neuronu
- λ strmost sigmoidu.

Z výše uvedených faktů vyplývají následující závěry:

1. Excitace neuronu se pohybuje v rozmezí 0 a 1, kde hodnota 1 znamená plnou excitaci neuronu na rozdíl od hodnoty 0, která odpovídá stavu inhibice (utlumení).
2. V případě, že se vnitřní potenciál neuronu blíží hodnotě +∞, pak dochází k plné excitaci neuronu, tedy $x = S(+\infty) = 1$.
3. Naopak v případě, že se vnitřní potenciál blíží hodnotě -∞, pak dochází k úplné inhibici neuronu, tedy $x = S(-\infty) = 0$.

Vícevrtsvé sítě a metoda backpropagation

Pravděpodobně nejrozšířenější způsob propojení spojitých perceptronů jsou tzv. vícevrstvé sítě, jejichž topologie je následující



Obr.10-6. Vícevrstvá neuronová síť

Z uvedeného obrázku vyplývá, že neuronová síť je tvořena minimálně třemi vrstvami neuronů: vstupní, výstupní a alespoň jednou vnitřní vrstvou. Vždy mezi dvěmi sousedními vrstvami se pak nachází tzv. *úplné propojení neuronů*, tedy každý neuron nižší vrstvy je spojen se všemi neurony vrstvy vyšší. Jakým způsobem je informace zpracována v takové neuronové síti? Výklad započneme tím jednodušším, tedy dopředným šířením (*feedforward*) signálu:

1. Nejprve jsou excitovány na odpovídající úroveň (v rozmezí 0 až 1) neurony vstupní vrstvy.
2. Tyto excitace jsou pomocí vazeb přivedeny k následující vrstvě a upraveny (zesíleny či zeslabeny) pomocí synaptických vah.
3. Každý neuron této vyšší vrsvy provede sumaci upravených signálů od neuronů nižší vrstvy a je excitován na úroveň danou svou aktivační funkcí (vztah 1).
4. Tento proces probíhá přes všechny vnitřní vrstvy až k vrstvě výstupní, kde pak získáme excitační stavy všech jejích neuronů.

V podstatě jsme tímto způsobem získali odezvu neuronové sítě na vstupní podnět daný excitací neuronů vstupní vrstvy. Takovým způsobem vlastně probíhá šíření signálů i v biologickém systému, kde vstupní vrstva může být tvořena např. zrakovými buňkami a ve výstupní vrstvě mozku jsou pak identifikovány jednotlivé objekty sledování. Otázkou zůstává to nejdůležitější, jakým způsobem jsou stanoveny ony synaptické váhy vedoucí ke korektní odezvě na vstupní signál. Proces stanovení synaptických vah je opět spjat s pojmem učení - adaptace - neuronové sítě.

Další otázkou je i schopnost *generalizace* - *zobecnění* -nad naučeným materiálem, jinými slovy jak je neuronová síť schopna na základě naučeného usuzovat na jevy, které nebyly součástí učení, které však lze nějakým způsobem odvodit. I tady je cítit jakási analogie s lidským učením daná rozdílem mezi bezduchým biflováním a učením spjatým se schopností porozumět problematice tak, abych mohl nové odvodit z předchozího.

Co je nutné k naučení neuronové sítě? Je to jednak tzv. *trénovací množina* obsahující prvky popisující řešenou problematiku a dále pak metoda, která dokáže tyto vzorky zafixovat v neuronové síti formou hodnot synaptických vah pokud možno včetně již uvedené schopnosti generalizovat. Zastavme se nejdříve u trénovací množiny. Každý vzor trénovací množiny popisuje jakým způsobem jsou excitovány neurony vstupní a výstupní vrstvy.

Formálně můžeme trénovací množinu *T* definovat jako množinu prvků (vzorů), které jsou definovány jako uspořádané dvojice následujícím způsobem:

$$T = \Big\{ \{I_1, O_1\} \quad \{I_2, O_2\} \quad \dots \quad \{I_p, O_p\} \Big\},$$
$$I_i = \big[i_1 \quad i_2 \quad \dots \quad i_k \big], \qquad i_j \in \langle 0,1 \rangle,$$
$$O_i = \big[o_1 \quad o_2 \quad \dots \quad o_m \big], \qquad o_j \in \langle 0,1 \rangle,$$

kde p počet vzorů trénovací množiny
I_i vektor excitací vstupní vrstvy tvořené *k* neurony
O_i vektor excitací výstupní vrstvy tvořené *l* neurony
i_j, o_j excitace *j-tého* neuronu vstupní resp. výstupní vrstvy.

Metoda, která umožňuje adaptaci neuronové sítě nad danou trénovací množinou se nazývá *backpropagation*, což v překladu znamená metodu zpětného šíření. Na rozdíl od už popsaného dopředného chodu při šíření signálu neuronové sítě tato metoda adaptace spočívá v opačném šíření informace směrem od vrstev vyšších k vrstvám nižším. Pokusme se nejprve o verbální popis uvedené metody:

1. Vezmeme nejprve vektor I_i *i-tého* prvku trénovací množiny, kterým excitujeme neurony vstupní vrstvy na odpovídající úroveň.
2. Známým způsobem provedeme dopředné šíření tohoto signálu až k výstupní vrstvě neuronů.
3. Srovnáme požadovaný stav daný vektorem O_i *i-tého* prvku trénovací množiny se skutečnou odezvou neuronové sítě.
4. Rozdíl mezi skutečnou a požadovanou odezvou definuje chybu neuronové sítě. Tuto chybu pak v určitém poměru - *learning rate* - "vracíme zpět" do neuronové sítě formou úpravy synaptických vah mezi jednotlivými vrstvami směrem od horních vrstev k vrstvám nižším tak, aby chyba při následující odezvě byla menší.
5. Po vyčerpání celé trénovací množiny[1] se vyhodnotí celková chyba přes všechny vzory trénovací množiny a pokud je tato vyšší než požadovaná celý proces se opakuje znovu.

Pokusme se v dalším o exaktní definování metody backpropagation (BP). Podstata metody BP spočívá v hledání minima funkce chyby *E* definované např. tímto způsobem

$$E = \frac{1}{2} \sum_{i=1}^{p} \sum_{j=1}^{m} (y_j - o_j)_i^2,$$

kde y_j skutečná odezva j-tého neuronu výstupní vrstvy
o_j požadovaná odezva j-tého neuronu výstupní vrstvy daná vzorem trénovací množiny
p celkový počet vzorů trénovací množiny
m počet neuronů výstupní vrstvy.

Cesta, jakou lze tohoto cíle dosáhnout, je právě úpravou synaptických vah mezi neurony *i* a *j* dle následující formule[2]

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i} + \mu \Delta w_i',$$

kde η koeficient učení,
μ koeficient vlivu změny vah z předchozího kroku (z intervalu $\langle 0,1 \rangle$),
$\Delta w'_i$ změna synaptické váhy z předchozího kroku.

[1] Zde lze explicitně rozhodnout kolik iterací věnujeme každému vzoru, zda-li pouze jednu, či několik.
[2] Pro jednoduchost nebudeme v dalším používat druhý index j.

Rozbor tohoto výrazu započneme tím jednodušším, tedy druhou částí součtu, která vyjadřuje podíl vlivu na hodnotu w_i daný změnou synaptické váhy vypočítanou v předchozím kroku. Pokud se koeficient μ rovná nule znamená to pro naši neuronovou síť, že nás minulost nezajímá a hodnota změny synaptické váhy je dána pouze aktuálním krokem. Naopak jednotková hodnota koeficientu znamená velkou "setrvačnost" sítě respektující trend daný předchozími kroky. Jestliže výše uvedené má především význam doplňkového charakteru umožňující urychlit konvergenci sítě k naučenému stavu, pak první část vzorce vyjadřuje podstatu adaptace neuronové sítě. Pokusme se o osvětlení výrazu daného součinem koeficientu učení η a parciální derivace chyby E podle synaptické váhy w_i. Pokud hodnota této derivace je velká a kladná, znamená to, že byť i minimální nárůst hodnoty synaptické váhy vede k velké chybě odezvy neuronové sítě. Je proto nutné "ubrat" z aktuální hodnoty synaptické váhy, neboť tímto onu chybu zmenšíme. Analogicky platí pro velkou, ale zápornou hodnotu derivace, že je naopak nutné hodnotu synaptické váhy zvětšit, pokud by měla být chyba odezvy v následujícím kroku nižší. Velikosti úprav synaptických dat jsou logicky dány jednak hodnotami těchto derivací a již výše zmíněným koeficientem učení - learning rate. Čím větší bude tento koeficient, tím razantnější budou změny v neuronové síti a naopak.Pokud se bude jeho hodnota blížit k nule, pak změny budou jen velmi nepatrné. Na tomto místě se opět nabízí analogie s lidským chováním. Jestliže v prvém případě se jedná o člověka, který s každou novou informací výrazně přebuduje své názory a znalosti ("kam vítr tam plášť"), pak druhý případ vyžaduje dlouhé přesvědčování a působení, než dotyčný akceptuje něco nového. Již z této analogie je patrné, jak je tento koeficient důležitý pro efektivní adaptaci neuronové sítě, nicméně jeho stanovení je věc experimentu a hledání. Prakticky neexistuje exaktní pravidlo, které by tento problém mohlo vyřešit.

V dalším si ukážeme, jakým způsobem lze hodnotu derivace z výrazu $\Delta w'_i$ vypočítat. Počáteční stav neuronové sítě je dán náhodným vygenerováním malých kladných hodnot synaptických vah.

Výpočet pak povedeme následujícím směrem

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial y} \cdot \frac{dy}{dz} \cdot \frac{\partial z}{\partial w_i} ,$$

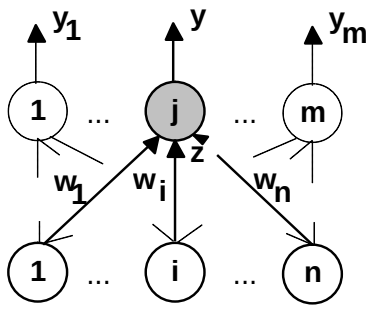
přičemž platí následující vztahy

$$z = \sum_{i=0}^{n} w_i x_i \quad \text{a} \quad y = \frac{1}{1+e^{-\lambda z}}$$

$$\frac{\partial z}{\partial w_i} = x_i$$

$$\frac{dy}{dz} = y \cdot (1-y) \cdot \lambda .$$

Posledním problémem je, jak určit hodnotu výrazu $\frac{\partial E}{\partial y}$. Nejprve uvažujme situaci, kdy daný neuron je součástí výstupní vrstvy neuronové sítě:

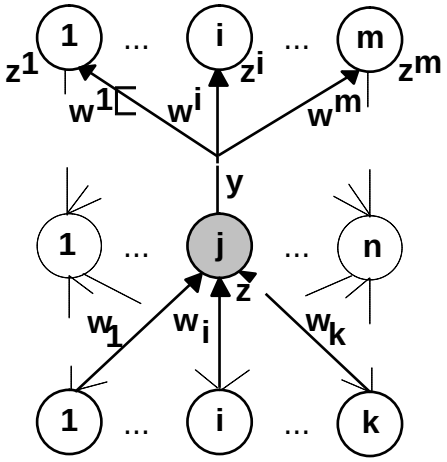


Obr. 10-7 Adaptace vah neuronu výstupní vrstvy

Pak lze odvodit, že pro hledaný výraz a vzor *k* platí

$$\frac{\partial E}{\partial y} = \left(y - o_j\right)_k .$$

Dále předpokládejme, že neuron se nachází v některé z vnitřních vrstev:



Obr. 10-8 Adaptace vah neuronu vnitřní vrstvy

Horním a dolním indexováním se pokusíme rozlišit, ke které vrstvě indexovaný výraz náleží. Odtud pak platí vztah

$$\frac{\partial E}{\partial y} = \sum_{i=1}^{m} \frac{\partial E}{\partial z^i} \cdot \frac{\partial z^i}{\partial y},$$

kde suma je provedena přes všechny neurony vrstvy nacházející se nad uvažovaným neuronem. Na základě platnosti

$$\frac{\partial z^i}{\partial y} = w^i,$$

lze provést následující substituci

$$\frac{\partial E}{\partial y} = \sum_{i} \frac{\partial E}{\partial z^i} \cdot w^i .$$

Výsledný vztah pro výpočet gradientu neuronu má po opětovném zavedení všech indexů tento tvar:

$$\frac{\partial E}{\partial w_{ij}} = \delta_{jk} \cdot y_j \cdot \left(1 - y_j\right) \cdot \lambda \cdot x_i ..$$

Rozdíl δ_{jk} mezi skutečnou a požadovanou odezvou neuronu *j* vzoru *k* vnější resp. vniřní vrstvy je definován následovně

$$\delta_{jk} = \left(y_j - o_j\right)_k \quad \text{resp.} \qquad \delta_{jk} = \sum_{i=1}^{m} \delta^{ik} \cdot y^i \cdot \left(1 - y^i\right) \cdot \lambda \cdot w^{ji},$$

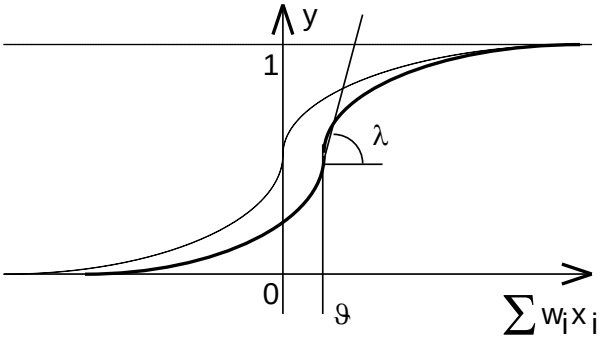
kde suma je provedena přes všechny neurony vrstvy vyšší k aktuální. Z výrazů je zřejmá nutnost nejprve stanovit chyby neuronů ve vrstvách vyšších, na základě kterých pak může být vypočtena chyba neuronů ve vrstvách nižších.

Doposud jsme se v našem výkladu zabývali pouze adaptací synaptických vah mezi neurony a uvažovali jsme neurony se stejnou aktivační funkcí, přesněji se stejnou strmostí sigmoidu λ. Adaptaci prahové hodnoty neuronu jsme zatím uvažovali jako součást adaptace vah s indexem rovným 0. Tedy

$w_0 = \vartheta \quad \text{pro} \quad x_0 = 1.$

Taková síť se nazývá *homogenní*. Nicméně nic nebrání tomu, abychom adaptaci podrobili u jednotlivých neuronů nejen synaptické váhy, ale i prahy a strmosti sigmoidů. Aktivační dynamika pak bude vyjádřena následujícím způsobem:

$$z = \sum_{i=1}^{n} w_i x_i \quad \text{a} \quad y = \frac{1}{1 + e^{-\lambda(z-\vartheta)}}.$$



Obr. 7 Parametrizovaná aktivační funkce neuronu

Tímto způsobem lze získat tzv. *heterogenní* síť, kde obecně každý neuron může mít svou aktivační dynamiku. Tato možnost ve většině případů zvyšuje schopnost sítě konvergovat k naučenému stavu. Metoda využívající této možnosti se nazývá *parametrická backpropagation* PAB. Podstata PAB spočívá v definování chybové funkce E závislé nejen na vektoru synaptických vah, ale i na vektoru strmostí sigmoidů a prahů. Formálně tedy

$E = E(w, \lambda, \vartheta).$

Úprava strmostí neuronů probíhá podle analogických pravidel jako v případě synaptických vah. V našem případě bude platit

$$\Delta\lambda_i = -\xi \frac{\partial E}{\partial \lambda_i} + \gamma \, \Delta\lambda_i',$$

$$\Delta\vartheta_i = -\psi \frac{\partial E}{\partial \vartheta_i} + \rho \, \Delta\vartheta_i',$$

kde $\zeta,, \psi$ koeficient učení pro strmosti a prahové hodnoty,
γ, ρ koeficient vlivu změny parametrů z předchozího kroku (z intervalu $\langle 0,1 \rangle$)
$\Delta\lambda_i', \Delta\vartheta_i'$ změna strmosti sigmoidu z předchozího kroku.

Pro stanovení hodnot derivací platí pro strmosti resp. prahové hodnoty vybraného neuronu následující výrazy[3]:

$$\frac{\partial E}{\partial \lambda} = \frac{\partial E}{\partial y} \cdot \frac{dy}{d\lambda}, \quad \text{kde} \quad \frac{dy}{d\lambda} = y \cdot (1-y) \cdot (z - \vartheta),$$

resp.

[3]Stejně jako v předchozím případě i zde pro jednoduchost nebudeme uvádět indexování

$$\frac{\partial E}{\partial \vartheta} = \frac{\partial E}{\partial y} \cdot \frac{dy}{d\vartheta}, \quad \text{kde} \quad \frac{dy}{d\vartheta} = y \cdot (1 - y) \cdot (-\lambda),$$

kde $\frac{\partial E}{\partial y}$ se vypočítá stejným způsobem jak bylo uvedeno v předchozím textu v závislosti na tom, zda-li se neuron nachází ve vnější nebo vnitřní vrstvě.

Spojitý perceptron s intervalovým stavem excitace

Tento přístup je zobecněním předchozího přístupu s tím rozdílem, že stav excitace neuronu je dán vnitřním potenciálem ve formě intervalu $\langle x, y \rangle$, kterému odpovídá vlastní excitace opět ve formě intervalu definovaného následujícím způsobem $\langle a, b \rangle$. Označíme-li množinu neuronů předcházející (nižší) vrstvy jako *low*, pak pro minimální resp. maximální vnitřní potenciál a pro kladnou hodnotu strmosti neuronu platí:

$$x = w_0 + \sum_{w_i>0, i\in low} w_i a_i + \sum_{w_i<0, i\in low} w_i b_i,$$

resp.

$$y = w_0 + \sum_{w_i>0, i\in low} w_i b_i + \sum_{w_i<0, i\in low} w_i a_i,$$

kde a_i, b_i vyjadřují minimální a maximální stav excitace neuronu *i* předchozí vrstvy.

Vlastní excitace je pak dána aplikací aktivační funkce, tedy

$$a = S(x), \quad b = S(y).$$

Použití metody *backpropagation* však vyžaduje diferencovatelné funkce stanovující vnitřní potenciál neuronu. K tomuto účelu nám poslouží spojitá signum funkce následujícího tvaru:

$$s(w) = \frac{1}{1+e^{-w}}, \quad \bar{s}(w) = \frac{1}{1+e^{w}}, \text{ pro které aproximativně platí}$$

$$s(w) = 1 \quad \text{pro} \quad w \to +\infty, \; s(w) = 0 \quad \text{pro} \quad w \to -\infty$$
$$\bar{s}(w) = 1 \quad \text{pro} \quad w \to -\infty, \; \bar{s}(w) = 0 \quad \text{pro} \quad w \to +\infty.$$

Použitím těchto funkcí lze přepsat vztahy pro vnitřní potenciál neuronu tímto způsobem:

$$x = w_0 + \sum_{i\in low} \left(s(w_i) w_i a_i + \bar{s}(w_i) w_i b_i \right) = w_0 + \sum_{i\in low} w_i \left(s(w_i) a_i + \bar{s}(w_i) b_i \right),$$
$$y = w_0 + \sum_{i\in low} \left(s(w_i) w_i b_i + \bar{s}(w_i) w_i a_i \right) = w_0 + \sum_{i\in low} w_i \left(s(w_i) b_i + \bar{s}(w_i) a_i \right)..$$

Tímto způsobem je zajištěna diferencovatelnost obou výrazů. Je zřejmé, že platí $s(w) + \bar{s}(w) = 1$ pro všechna *w*. To tedy znamená, že hodnota váhy w_i je rozdělena mezi a_i a b_i podle stupně její positivity, či negativity.

Poslední věc, kterou je nutné vyřešit je vliv znaménka strmosti neuronu. Zde je třeba si uvědomit, že platí $\bar{s}(w) = s(-w)$ a můžeme tedy považovat hodnotu strmosti neuronu λ za parametr funkce signum. Nový tvar funkce signum bude pak následující

$$s(w) = \frac{1}{1+e^{-\lambda w}}, \quad \bar{s}(w) = \frac{1}{1+e^{\lambda w}}.$$

Zobecněná metoda Back Propagation

Stav excitace neuronu ve formě intervalu umožňuje definovat také vzory trénovací množiny ve formě intervalu. Požadovaná odezva sítě tedy bude dána intervalem $\langle A_i, B_i \rangle$ a minimalizovaná chybová funkce *E* bude mít následující tvar:

$$E = \frac{1}{2}\sum_{i=1}^{n}\left(a_i - A_i\right)^2 + \frac{1}{2}\sum_{i=1}^{n}\left(b_i - B_i\right)^2.$$

Stejně jako v předchozím případě změna vah (analogicky i změna strmostí neuronů) bude dána vtahem:

$$\Delta w_i = -\eta\frac{\partial E}{\partial w_i} + \mu\,\Delta w_i' \qquad \text{resp. } \Delta\lambda_i = -\xi\frac{\partial E}{\partial \lambda_i} + \mu\,\Delta\lambda_i'$$

Výpočet pak povedeme následujícím směrem

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial a}\cdot\frac{da}{dx}\cdot\frac{\partial x}{\partial w_i} + \frac{\partial E}{\partial b}\cdot\frac{db}{dy}\cdot\frac{\partial y}{\partial w_i} \qquad \text{resp.} \qquad \frac{\partial E}{\partial \lambda} = \frac{\partial E}{\partial a}\cdot\frac{da}{d\lambda} + \frac{\partial E}{\partial b}\cdot\frac{db}{d\lambda}$$

kde pro jednotlivé členy platí

$$\frac{\partial x}{\partial w_i} = a_i s(w_i)\left(1+\lambda w_i \bar{s}(w_i)\right) + b_i \bar{s}(w_i)\left(1-\lambda w_i s(w_i)\right),$$

$$\frac{\partial y}{\partial w_i} = a_i \bar{s}(w_i)\left(1-\lambda w_i s(w_i)\right) + b_i s(w_i)\left(1+\lambda w_i \bar{s}(w_i)\right),$$

$$\frac{da}{dx} = a\cdot(1-a)\cdot\lambda \text{ a} \qquad \frac{db}{dy} = b\cdot(1-b)\cdot\lambda,$$

$$\frac{\partial a}{\partial \lambda} = \frac{\partial S(x)}{\partial \lambda} + \frac{\partial a}{\partial x}\cdot\frac{\partial x}{\partial \lambda} = a(1-a)\cdot\left(x - \lambda\cdot\sum_{i\in low} w_i^2 s(w_i)\bar{s}(w_i)\left(b_i - a_i\right)\right),$$

$$\frac{\partial b}{\partial \lambda} = \frac{\partial S(y)}{\partial \lambda} + \frac{\partial b}{\partial y}\cdot\frac{\partial y}{\partial \lambda} = b(1-b)\cdot\left(y + \lambda\cdot\sum_{i\in low} w_i^2 s(w_i)\bar{s}(w_i)\left(b_i - a_i\right)\right).$$

Analogicky se standartní *backpropagation* určíme hodnoty derivací chybové funkce podle stavu excitace pro výstupní resp. vnitřní vrstvu sítě následujícím způsobem:

$$\frac{\partial E}{\partial a} = a_i - A_i, \quad \frac{\partial E}{\partial b} = b_i - B_i,$$

resp.

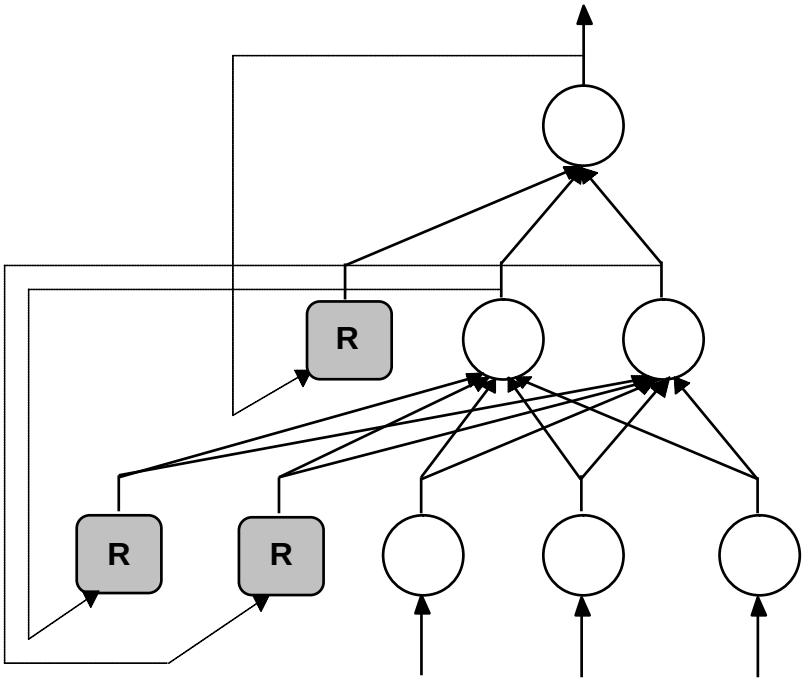
$$\frac{\partial E}{\partial a} = \sum_{i=1}^{m}\left(\frac{\partial E}{\partial x^i}\cdot\frac{\partial x^i}{\partial a} + \frac{\partial E}{\partial y^i}\cdot\frac{\partial y^i}{\partial a}\right) = \sum_{i=1}^{m}\left(\frac{\partial E}{\partial x^i}\cdot w^i s(w^i) + \frac{\partial E}{\partial y^i}\cdot w^i \bar{s}(w^i)\right),$$

$$\frac{\partial E}{\partial b} = \sum_{i=1}^{m}\left(\frac{\partial E}{\partial x^i}\cdot\frac{\partial x^i}{\partial b} + \frac{\partial E}{\partial y^i}\cdot\frac{\partial y^i}{\partial b}\right) = \sum_{i=1}^{m}\left(\frac{\partial E}{\partial x^i}\cdot w^i \bar{s}(w^i) + \frac{\partial E}{\partial y^i}\cdot w^i s(w^i)\right),$$

kde m je počet neuronů vrstvy nad vrstvou aktuální.

Rekurentní vícevrstvé neuronové sítě

Všechny doposud používané způsoby mapování vstupního vektoru na výstupní pomocí vícevrstvé neuronové sítě byly nezávislé na kontextu. To znamená, že stejný vstupní stimul vede vždy na stejnou odezvu sítě. Cílem této kapitole je ukázat, jakým způsobem lze implementovat do vícevrstvé neuronové sítě *časový kontext*. Jinými slovy řečeno, odezva sítě nebude dána pouze aktuálním vstupním stimulem, ale bude odrážet i vliv stimulů, které tento aktuální předcházely. Model vícevrtvé sítě umožňující tento princip realizovat je tzv. *rekurentní neuronová síť*, která bude opět využívat metodu *backpropagation* ke své adaptaci. Oproti předcházejícím modelům se v tomto případě signál nešíří pouze od vstupní vrstvy směrem k vrstvě výstupní, ale dochází i ke zpětnovazebnému přenosu informace od vrstev výšších zpět do vrstev nižších. Tato zpětná vazba je řešena prostřednictvím tzv. rekurentních neuronů a příkladem takové sítě nám může být následující obrázek.



Obr.10-9. Rekurentní vicevrstvá neuronová síť

Topologie rekurentní sítě se liší od klasické dalšími rekurentními neurony (označené písmenem R) ve vstupní a vnitřní vrstvě. Ve střední vrstvě je jeden neuron odpovídající jednomu neuronu výstupní vrtvy a ve vstupní vrstvě jsou dva rekurentí neurony odpovídající dvou neuronům vnitřní vrstvy. Každý z těchto neuronů přijímá jediný vstupní signál od jemu příslušejícímu regulérního neuronu z následující vrsvy (znázorněno čárkovanou vazbou). Pokusme se nyní o verbální popis šíření signálu při vyvolání informace. Zde je důležité si uvědomit, že vyvolání informace spočívá v postupném po sobě následujícím předložení vstupních simulů s následnou odezvou respektující kontext, ve kterém jsou vstupy předkládány.

1. Před předložením požadované sekvence stimulů jsou všechny neurony vnitřní a výstupní vrstvy inicializované na "malou excitaci". Předpokládejme tedy, že v čase *t=0* excitace těchto neuronů bude 0.25. Tato excitace neuronů současně reprezentuje aktivitu rekurentních neuronů v čase *t=1*.

2. Následuje vyslání signálu vstupní vrstvy prostřednictvím vazeb do vnitřní vrstvy (časový krok *t=1*). Současně je však do vnitřní vrsvy přenesena i inicializační aktivita neuronů z předchozího časového kroku prostřednictvím rekurentních neuronů. Tyto dva zdroje vstupní informace se odrazí ve vstupním potenciálu neuronu dle již známého výrazu

$$z = \sum_{i=0}^{n} w_i x_i ,$$

kde suma je provedena přes všechny neurony nižší vrsvy včetně rekurentních. Po aplikaci aktivační funkce (např. již zmíněné sigmoidy) je signál předán prostřednictvím vazeb až k výstupní vrstvě neuronů. Navíc tyto neurony také přijímají signál od rekurentních neuronů předcházející vnitřní vrstvy. Výstupy těchto rekurentní neuronů reprezentují odezvu výstupní vrstvy neuronů v předchozím časovém kroku. Proces stanovení potanciálu je shodný s předchozím. Po aplikaci aktivační funkce pak získáváme výstupní vektor.

3. Příchodem dalšího vstupního stimulu vnitřní vrstva neuronů přijímá nejen tento nový vstupní signál, ale prostřednictvím rekurentních neuronů také svou vlastní aktivitu odpovídající předcházejícímu vzoru. Jejich kombinace pak určuje aktivitu neuronů vnitřní vrstvy v čase t=2. Analogicky s předchozím bodem výstupní vrtsva neuronů odráží nejen aktivitu předcházející vnitřní vrtvy, ale také svou vlastní z předchozího časového kroku.

Celý postup se pak opakuje dokud nejsou vyčerpány všechny vzory dané sekvence.

Komplikovanější je také vlastní proces adaptace rekurentní neuronové sítě. Cílem je totiž adaptovat síť prostřednictvím sekvencí vzorů tak, aby odrážela nejen jednotlivé vzory, ale i jejich kontext. Trénovací množina je pak tvořena např. takovými sekvencemi vzorů:

Vstupní sekvence			Výstupní (cílová) sekvence		
Vzor 1	Vzor 2	Vzor 3	Vzor 1	Vzor 2	Vzor 3
1 0 0 ⇒	0 1 0 ⇒	0 0 1	0.25 ⇒	0.0 ⇒	1.0
0 0 1 ⇒	0 1 0 ⇒	1 0 0	0.5 ⇒	1.0 ⇒	0.75

Je vcelku zřejmé, že v případě, kdyby tyto vzory byly samostatné a použitá metoda adaptace by byla klasická *backpropagation*, pak by síť nemohla být nikdy adaptována. Vyplývá to z faktu, že vstupní vektory mají pokaždé jinou odezvu. Použití klasické metody by si tedy vyžádalo definovat topologii o devíti neuronech ve vstupní a třech neuronech ve výstupní vrstvě. Tímto bychom zavedli jakýsi *pseudokontext* do standardního modelu backpropagation. V řadě případů se tento způsob také používá, ale zřejmě rozsáhlejší úkoly naráží jednak na problém efektivní aplikace a dále pak na problém nutnosti použití sekvencí konstantní délky.

Vlastní algoritmus adaptace probíhá vždy v rámci dané sekvence dávkovým způsobem, což znamená, že všechny změny vah se akumulují pro všechny vzory sekvence a teprve potom jsou tyto změny provedeny. Konkrétně má algoritmus implementující tento způsob učení následující čtyř části:

1. Nejprve se provede dopředné šíření signálu pro všechny vzory sekvence počínaje časovým krokem 1 až *N*, kde *N* reprezentuje počet vzorů v dané sekvenci.

2. Vypočítají se chyby neuronů vnější a dále pak předcházejících nižších vrstev analogicky s metodou *backpropagation* s tím, že nejprve se tyto chyby stanoví pro poslední časový krok a následně pak pro kroky předcházející až po první.

3. Vypočítané chyby z předchozího kroku algoritmu se použijí ke stanovení změn vah, které se akumulují pro každou jednotlivou vazbu a každý jednotlivý časový krok počínaje prvním a konče *N-tým* v pořadí.

4. Váhy všech vazeb neuronové sítě se aktualizují prostřednictvím hodnot akumulovaných změn vah vypočítaných v bodě 3 algoritmu.

Poslední co zbývá, je stanovit hodnoty změn vah vazeb v jednotlivých krocích. Zde je nutné si nejprve uvědomit, že chybné nastavení vah vede nejen k chybné odezvě aktuálního časového kroku, ale také k chybné odezvě kroku následujícího. Odtud vyplývá, že se gradientní metoda *backpropagation* musí minimalizovat výslednou chybu E_v definovanou následujícím způsobem

$$E_v(t) = E(t) + E(t+1),$$

kde pro chybnou odezvu v aktuálním resp. následujícím kroku platí (analogicky s předchozími kapitolami)

$$E(t) = \frac{1}{2}\sum_{j=1}^{m}\left(y_j(t) - o_j(t)\right)^2 \text{ resp.} \qquad E(t+1) = \frac{1}{2}\sum_{j=1}^{m}\left(y_j(t+1) - o_j(t+1)\right)^2.$$

Zde je nutné si uvědomit, že výrazy $y_j(t)$ a $o_j(t)$ vyjadřují skutečný stav excitace *j-tého* neuronu výstupní vrstvy a požadovanou odezvu vzoru sekvence předloženého neuronové síti v daném časovém kroku *t*. Zatímco výrazy $y_j(t+1)$ a $o_j(t+1)$ vyjadřují skutečný stav excitace *j-tého* neuronu výstupní vrstvy a požadovanou odezvu dalšího vzoru sekvence předloženého neuronové síti v následujícím časovém kroku *t+1*. Vlastní změna váhy vazby pro daný krok je dána již známým vztahem

$$\Delta w_i(t) = -\eta\frac{\partial E_v(t)}{\partial w_i(t)}.$$

Pokusíme se nejprve vyjádřit vliv aktuální chyby resp. chyby vyplývajícího z následujícího kroku na změnu vah pomocí následujících operací analogických s klasickou *backpropagation*[4]

$$\frac{\partial E(t)}{\partial w_i(t)} = \frac{\partial E(t)}{\partial y(t)} \cdot \frac{dy(t)}{dz(t)} \cdot \frac{\partial z(t)}{\partial w_i(t)},$$

resp.

$$\frac{\partial E(t+1)}{\partial w_i(t)} = \frac{\partial E(t+1)}{\partial y(t)} \cdot \frac{dy(t)}{dz(t)} \cdot \frac{\partial z(t)}{\partial w_i(t)},$$

kde druhý člen je závislý na typu aktivační funkce a poslední člen je dán vztahem

$$\frac{\partial z(t)}{\partial w_i(t)} = x_i(t) \qquad \text{pro regulérní neuron a}$$

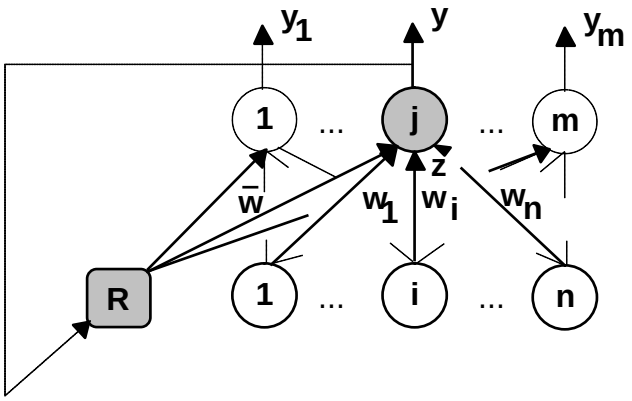
$$\frac{\partial z(t)}{\partial \overline{w_i}(t)} = y_i(t-1) \qquad \text{pro rekuretní neuron.}$$

Symbol váhy $\overline{w}$ budeme v dalším používat pro odlišení vazeb směřujících od rekurentního neuronu k aktuálnímu neuronu .

Zbývá tedy vyjádřit prvé členy obou výrazů reprezentované parciálními derivacemi $\frac{\partial E(t)}{\partial y(t)}$ a $\frac{\partial E(t+1)}{\partial y(t)}$ pro neurony výstupní a libovolné vnitřní vrstvy.

Započněme jednodušší situací neuronu výstupní vrstvy schematicky znázorněnou následujícím obrázkem:

[4]Opět pro jednoduchost vynecháme indexování vztahující se k aktuálnímu neuronu



Obr. 10-10 Adaptace vah neuronu výstupní vrstvy rekurentní sítě

Zřejmě platí

$$\frac{\partial E(t)}{\partial y(t)} = \delta(t),$$

kde
$\delta(t) = y(t) - o(t).$

Pro vliv chyby následujícího kroku pak platí

$$\frac{\partial E(t+1)}{\partial y(t)} = \sum_{j=1}^{m} \left(y_j(t+1) - o_j(t+1) \right) \cdot \frac{\partial y_j(t+1)}{\partial y(t)}.$$

Použitá suma vyjadřuje fakt, že výstup aktuálního neuronu ovlivňuje prostřednictvím rekurentního neuronu chybu všech neuronů výstupní vrstvy v následujícím časovém kroku. Dalšími úpravami získáme následující

$$\frac{\partial y_j(t+1)}{\partial y(t)} = \frac{dy_j(t+1)}{dz_j(t+1)} \cdot \frac{\partial z_j(t+1)}{\partial y(t)},$$
$$\frac{\partial z_j(t+1)}{\partial y(t)} = \overline{w}_j(t),$$

kde $\overline{w}_j(t)$ vyjadřuje váhu vazby mezi rekurentním neuronem a *j-tým* neuronem výstupní vrstvy, a která je stejná pro všechny vzory sekvence, tedy i pro všechny jim odpovídající časové kroky jejich postupného předkládání neuronové síti. Odtud

$$\frac{\partial E(t+1)}{\partial y(t)} = \sum_{j=1}^{m} \delta_j(t+1) \cdot \frac{dy_j(t+1)}{dz_j(t+1)} \cdot \overline{w}_j(t),$$

kde $\delta_j(t+1) = y_j(t+1) - o_j(t+1).$

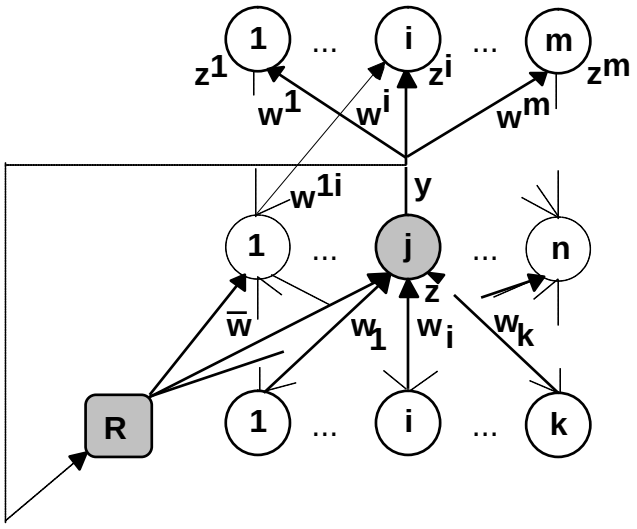
Výsledné výrazy pro stanovení hodnoty parciálních derivací nutných pro úpravu vah k regulérním resp. rekurentním neuronům od neuronu výstupní vrstvy mají tuto formu

$$\frac{\partial E_v(t)}{\partial w_i(t)} = \left[\delta(t) + \sum_{j=1}^{m} \delta_j(t+1) \cdot \frac{dy_j(t+1)}{dz_j(t+1)} \cdot \overline{w}_j(t) \right] \cdot \frac{dy(t)}{dz(t)} \cdot x_i(t),$$

resp.

$$\frac{\partial E_v(t)}{\partial \overline{w}_i(t)} = \left[\delta(t) + \sum_{j=1}^{m} \delta_j(t+1) \cdot \frac{dy_j(t+1)}{dz_j(t+1)} \cdot \overline{w}_j(t) \right] \cdot \frac{dy(t)}{dz(t)} \cdot y_i(t-1).$$

Na závěr provedeme odvození těchto vztahů také pro neurony vnitřní vrstvy dle následujícího obrázku:



Obr. 10-11 Adaptace vah neuronu vnitřní vrstvy rekurentní sítě

Zřejmě platí

$$\frac{\partial E(t)}{\partial y(t)} = \sum_{j=1}^{m} \frac{\partial E(t)}{\partial z^j(t)} \cdot \frac{\partial z^j(t)}{\partial y(t)},$$

$$\frac{\partial E(t)}{\partial y(t)} = \sum_{j=1}^{m} \frac{\partial E(t)}{\partial z^j(t)} \cdot w^j(t),$$

přičemž pro zjednodušení zavedeme následující substituci

$$\delta(t) = \sum_{j=1}^{m} \frac{\partial E(t)}{\partial z^j(t)} \cdot w^j(t) = \sum_{j=1}^{m} \delta^j(t) \cdot \frac{dy^j(t)}{dz^j(t)} \cdot w^j(t).$$

Komplikovanější je situace odrážející vliv chyby v následujícím časovém kroku

$$\frac{\partial E(t+1)}{\partial y(t)} = \sum_{j=1}^{m} \frac{\partial E(t+1)}{\partial z^j(t+1)} \cdot \frac{\partial z^j(t+1)}{\partial y(t)},,$$

kde dále platí

$$\frac{\partial z^j(t+1)}{\partial y(t)} = \frac{\partial}{\partial y(t)} \cdot \sum_{k=1}^{n} w^{kj}(t) \cdot y_k(t+1).$$

Suma na pravé straně nevyjadřuje nic jiného, než stanovení potenciálu i-tého neuronu vrstvy následující za vrstvou aktuální. Tento výraz lze dále upravit tímto způsobem

$$\frac{\partial z^j(t+1)}{\partial y(t)} = \sum_{k=1}^{n} w^{kj}(t) \cdot \frac{\partial y_k(t+1)}{\partial y(t)},$$

kde dále platí

$$\frac{\partial y_k(t+1)}{\partial y(t)} = \frac{dy_k(t+1)}{dz_k(t+1)} \cdot \frac{\partial z_k(t+1)}{\partial y(t)} = \frac{dy_k(t+1)}{dz_k(t+1)} \cdot \overline{w}_k(t).$$

Dosazením do původního výrazu a dalšími úpravami získáme následující

$$\frac{\partial E(t+1)}{\partial y(t)} = \sum_{j=1}^{m} \frac{\partial E(t+1)}{\partial z^j(t+1)} \cdot \sum_{k=1}^{n} w^{kj}(t) \cdot \frac{dy_k(t+1)}{dz_k(t+1)} \cdot \overline{w}_k(t),$$

$$\frac{\partial E(t+1)}{\partial y(t)} = \sum_{k=1}^{n} \overline{w}_k(t) \cdot \sum_{j=1}^{m} \frac{\partial E(t+1)}{\partial z^j(t+1)} \cdot w^{kj}(t) \cdot \frac{dy_k(t+1)}{dz_k(t+1)}.$$

Zavedením již známe substituce zjednodušíme poslední výraz do této formy

$$\frac{\partial E(t+1)}{\partial y(t)} = \sum_{k=1}^{n} \delta_k(t+1) \cdot \frac{dy_k(t+1)}{dz_k(t+1)} \cdot \overline{w}_k(t).$$

Výsledné výrazy pro stanovení hodnoty parciálních derivací nutných pro úpravu vah k regulérním resp. rekurentním neuronům od neuronu vnitřní vrstvy mají tuto formu

$$\frac{\partial E_v(t)}{\partial w_i(t)} = \left[\sum_{j=1}^{m} \delta^j(t) \cdot \frac{dy^j(t)}{dz^j(t)} \cdot w^j(t) + \sum_{k=1}^{n} \delta_k(t+1) \cdot \frac{dy_k(t+1)}{dz_k(t+1)} \cdot \overline{w}_k(t) \right] \cdot \frac{dy(t)}{dz(t)} \cdot x_i(t),$$

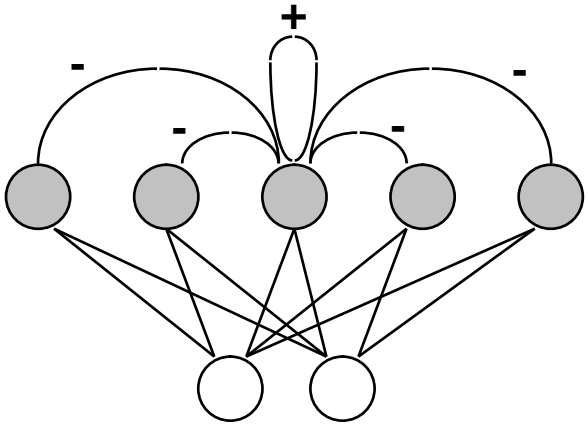
resp.

$$\frac{\partial E_v(t)}{\partial \overline{w}_i(t)} = \left[\sum_{j=1}^{m} \delta^j(t) \cdot \frac{dy^j(t)}{dz^j(t)} \cdot w^j(t) + \sum_{k=1}^{n} \delta_k(t+1) \cdot \frac{dy_k(t+1)}{dz_k(t+1)} \cdot \overline{w}_k(t) \right] \cdot \frac{dy(t)}{dz(t)} \cdot y(t-1).$$

Z uvedeného je vcelku zřejmé, že pro výpočet chyby v daném kroku musíme znát chybu z kroku následujícího vyjma posledního *N-tého* vzoru sekvence, kde již další časový krok neexistuje. Proto se v procesu adaptace začíná s vyhodnocením chyby v tomto posledním kroku (pomocí uvedených vztahů s vypuštěním členů s argumentem *t+1*) a postupuje se zpět k prvému vzoru sekvence. Teprve nakonec lze prostřednictvím akumulovaných změn vah provést aktualizaci všech vazeb.

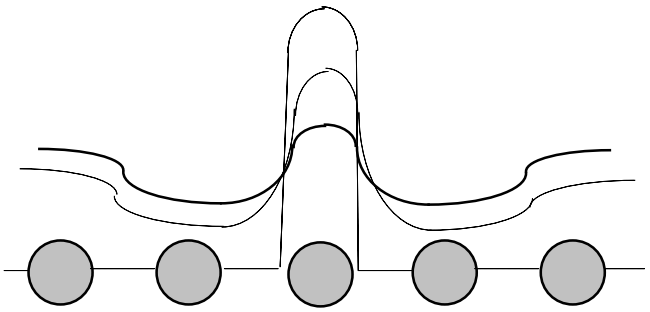
Kompetice, Kohonenovo učení a samoorganizace

V následujícím se budeme zabývat topologií sítě tvořenou dvěmi vrstvami neuronů, přičemž spodní reprezentuje vstupní jednotky, které jsou propojeny se všemi neurony vrsvy výstupní. V této vrstvě jsou neurony opět mezi sebou propojeny. Schématický obrázek je následující:



Obr.10-12. Kompetitivní síť

Propojení ve výstupní vrstvě je typické sebeexcitující vazbou ("+") a inhibičními vazbami vzhledem k ostatním neuronům. Tento způsob propojení vede k postupnému posilování toho neuronu, který byl na počátku excitován nejvíce. Výsledkem je pak situace, kdy tento neuron je vyexcitován na maximum, zatímco ostatní jsou uplně potlačeny (laterární, postranní inhibice).



Obr. 10-13. Vítězný neuron kompetice

Tento postup je obdobný postupu excitace a inhibice neuronů ve skutečném mozku. Každý neuron pak reprezentuje nějaký objekt, či třídu objektů ze vstupního prostoru. Takový neuron budeme nazývat jako *Grandmother cell* (GC), čili babiččina buňka, neboť právě ten je schopen rozpoznat "babičku vystupující ze tmy do místnosti". V našem případě onu temnou část reprezentuje spodní vrstva neuronů ve formě své excitace reprezentovanou vektorem ***x***. Ten neuron horní vrstvy, jehož potenciál $\Sigma w.x$ je maximální se pak stává GC odpovídající vektoru ***x***. Tedy *GC reprezentuje **x**, **x** patří k GC*. Tento neuron GC je však schopen rozpoznat celou třídu takových, si podobných, vektorů, což si také ukážeme v následujícím.

Předpokládejme, že síť je tvořena *m* vstupními a *k* výstupními neurony, přičemž vstupní vektor i vektor vah je normalizován. Pro *j-tý* neuron tedy platí:

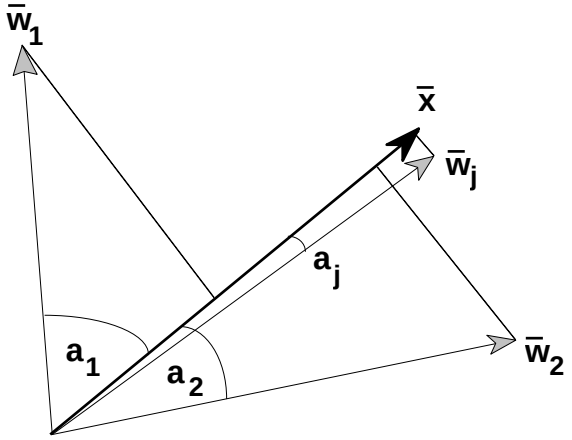
$$z_j = \sum_{i=1}^{m} w_{ji} \cdot x_i = \overline{w}_j \cdot \overline{x}\text{, kde}$$

$$\left|\overline{w}_j\right| = |\overline{x}| = 1$$

Vítězem soutěže se v důsledku laterální inhibice stává ten neuron, jehož vstupní potenciál je největší a tedy předložený vstup spadá do kategorie vstupních vektorů reprezentované tímto neuronem GC. Pokud se pokusíme vyjádřit tuto situaci prostřednictvím vektorového počtu, pak jednotlivé potenciály neuronů vyjadřují skalární součiny vektorů vah jednotlivých neuronů a vstupního vektoru reprezentovanými vztahem

$\overline{w}_j \cdot \overline{x} = \left|\overline{w}_j\right| \cdot |\overline{x}| \cdot \cos\alpha_j$ kde α_j vyjadřuje úhel sevřený oběma vektory,

a tudíž tyto součiny můžeme chápat jako projekce vektorů vah na vstupní vektor. Čím menší je sevřený úhel mezi oběma vektory, tím delší je i projekce váhy na vstupní vektor (viz. obrázek)



Obr. 10-14. Skalární součiny vektorů vah a vstupu

Zřejmě maximální potenciál neuronu rovný 1 je dosažitelný v případě že $\overline{w}_j = \overline{x}$. Zda-li tento neuron bude zastupovat i ostatní vstupní vektory, to závisí na tom, jak podobné jsou tyto vektoru $\overline{x}$. Ty vstupy $\overline{x}^i$, pro které platí

$\overline{w}_j \cdot \overline{x} \cong \overline{w}_j \cdot \overline{x}^i$

mohou být neuronem GC rozpoznány.

Princip adaptace vah takového neuronu pak spočívá v tom, že jeho váhy se přibližují novým vstupům dle následujícího předpisu (Kohonenovo pravidlo)[5]:

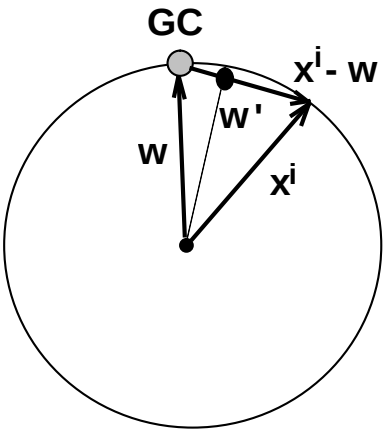
$\overline{w}' = \overline{w} + \eta \cdot \left(\overline{x}^i - \overline{w}\right)$ pro $\eta \in (0,1)$

kde η koeficient učení (plasticity) sítě.

Váhy ostatních neuronů, které prohrály kompetici zůstávají při adaptaci neměnné. Samotný koeficient učení se postupně snižuje až na hodnotu nula, což znamená že naše "babička" GC se stává stále usedlejší, až nakonec zůstává doma obklopena svou "rodinou" - množinou vstupů náležejícím neuronu GC.

Pro případ, kdy *m* = 2 a η = 0.3 lze proces adaptace zobrazit následujícím obrázkem:

[5]Index j vztahující se k danému neuronu GC pro jednoduchost zápisu vypustíme.



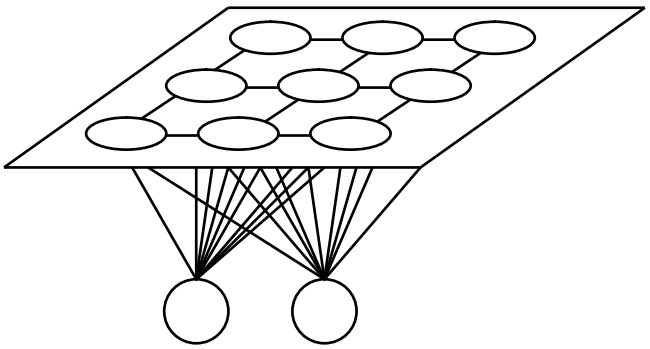
Obr. 10-15. Proces adaptace dle Kohonena

Z obrázku je patrné, že vektory blízké původnímu téměř zachovávají i normalitu nových vah. Porušení normality nastane v případě, že nový vektor je příliš vzdálen původnímu. Tento jev by ale měl vést ke stavu, kdy tento vektor bude excitovat jiný neuron GC, kolem kterého by se měla vytvořit další třída, či populace vstupních vektorů. Postupně se tak vytvoří shluky vstupních neuronů odpovídajících svému neuronu GC. Velmi podstatný je i fakt, že k vytvoření těchto shluků a jejich GC došlo prostřednictvím *adaptace bez učitele* a síť je tak schopna *samoorganizace*.

Problémem může být nevhodná náhodná inicializace vah, která vede k blízkým počátečním neuronům GC a tudíž pouze jeden z nich vyhrává kompetici zatímco ostatní zůstávají nevyužity. Jedna z možností jak tuto situaci pomoci vyřešit, je princip založený na "svědomí" každého z neuronů tak, že v případě příliš častých vítězství jednoho z nich, je tento z procesu soutěže na chvíli vyjmut, aby dostali šanci i ostatní neurony kohonenovy vrstvy.

Kohonenovy mapy

Uvedené principy adaptace a samoorganizace jsou aplikovány v sítích nazývaných jako Kohonenovy mapy. Hlavní ideou těchto neuronových sítí je nalézt prostorovou reprezentaci složitých datových struktur. Tedy, aby třídy si podobných vektorů, byly reprezentovány neurony blízkými si v dané topologii. Tato vlastnost je typická i pro skutečný mozek, kde například jeden konec sluchové části mozkové kůry reaguje na nízké frekvence, zatímco opačný konec reaguje na frekvence vysoké. Tímto způsobem je možné mnohodimenzionální data zobrazit v jednodušším prostoru. Prvotní a nejčastější použití Kohonenova algoritmu je ve formě dvoudimenzionální implementace. Topologie takové sítě může být např. následující:



Obr. 10-16. Kohonenova mapa

Kohonenův algoritmus je definován následujícím způsobem:

1. Inicializace sítě
   Definuj $w_{ij}(t)$ $(0 \le i \le n-1)$ jako váhu mezi vstupem *i* a neuronem *j* v čase *t*. Inicializuj tyto váhy malými náhodnými čísly. Nastav počáteční průměr sousedství kolem neuronu *j*, $N_j(0)$ na maximum (obvykle tak, aby pokrýval celou výstupní vrstvu).

2. Předložení vstupního vektoru
   Předlož vstup ve formě $x_o(t), x_1(t), x_2(t), \ldots, x_{n-1}(t)$, kde $x_i(t)$ je vstup uzlu *i* v čase *t*.

3. Výpočet vzdáleností (stanovení vítěze kompetice)
   Vypočti vzdálenosti d_j mezi vstupním vektorem a každým neuronem *j* následovně
   $d_j = \sum_{i=0}^{n-1} \left(x_i(t) - w_{ij}(t) \right)^2$.

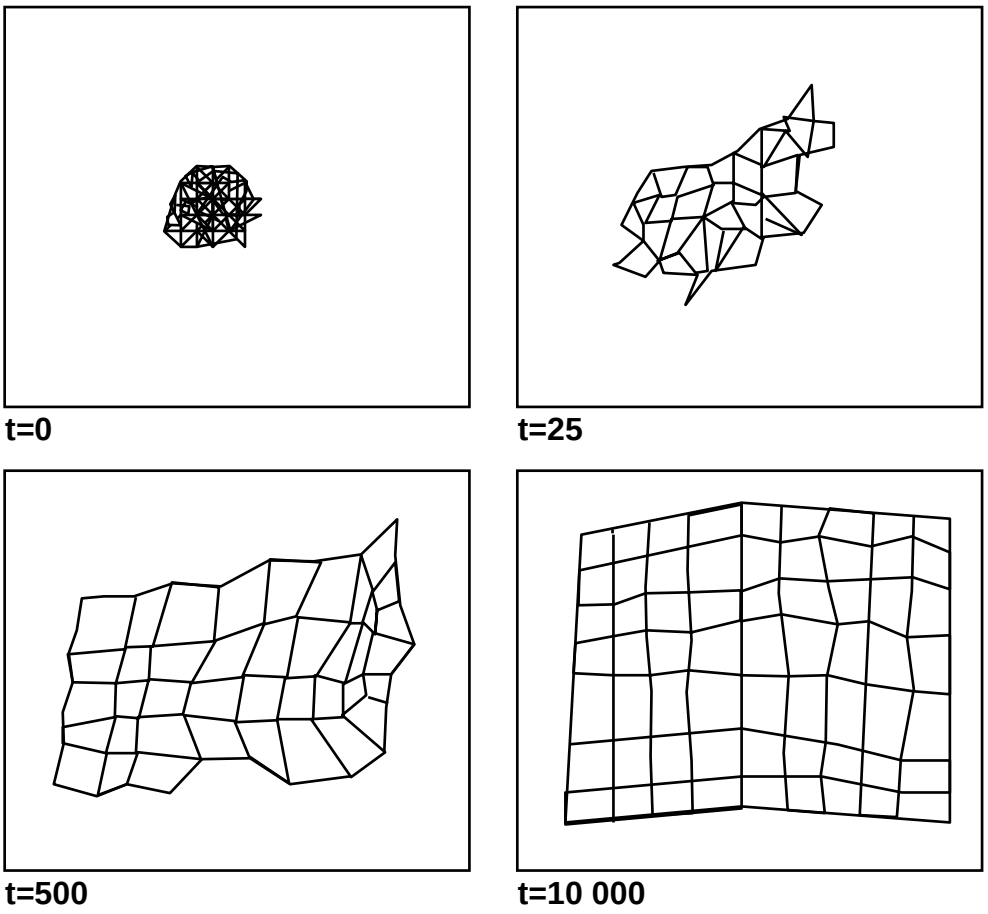
4. Výběr minimální vzdálenosti
   Určení vítězného neuronu prostřednictvím minima d_j jako *j**.

5. Úprava vah
   Úprava vah pro neuron *j** a jeho sousedy definované $N_{j*}(t)$. Nové váhy jsou dány vztahem:
   $w_{ij}(t+1) = w_{ij}(t) + \eta(t)\left(x_i(t) - w_{ij}(t) \right)$,
   pro $j \in N_{j*}(t)$, $0 \le i \le n-1$, kde $0 < \eta(t) < 1$.
   Přičemž hodnoty $\eta(t)$ a $N_{j*}(t)$ se postupně snižují v čase s cílem stabilizovat nastavené váhy a lokalizovat místa maximální aktivity.

6. Přestup k bodu 2.

Proces shlukování lze vysvětlit prostřednictvím funkce hustoty pravděpodobnosti. Tato funkce reprezentuje statistický nástroj popisující rozložení dat v prostoru. Pro daný bod prostoru lze tedy stanovit pravděpodobnost, že vektor bude v daném bodu nalezen. Je-li dán vstupní prostor a funkce hustoty pravděpodobnosti, pak je možné dosáhnout takové organizace mapy, která se této funkci přibližuje (za předpokladu, že je k dispozici reprezentativní vzorek dat). Jinými slovy řečeno, jestli jsou vzory ve vstupním prostoru rozloženy podle nějaké distribuční funkce, budou v prostoru vah rozloženy vektory analogicky.

Pokusme se výše uvedené demonstrovat na příkladu, kdy vstupní data jsou rovnoměrně rozložena v dvoudimenzionálním prostoru, konkrétně ve čtvercové oblasti. Váhové vektory budou tedy také dvoudimenzionální a budou zobrazovány formou bodu v prostoru vah. Dále budou v témže prostoru vykreslovány přímky spojující body (váhy) sousedících neuronů. Toto zobrazení pak vyjadřuje prostorové vztahy mezi neurony v prostoru vah. Vývoj prostorového uspořádání váhových vektorů lze demonstrovat na následujících diagramech.



Obr. 10-17. Proces adaptace mapy

Z obrázku je patrné, že neurony byly optimálně rozloženy tak, aby pokryly vstupní datový prostor.

Kvantování vektorů učením

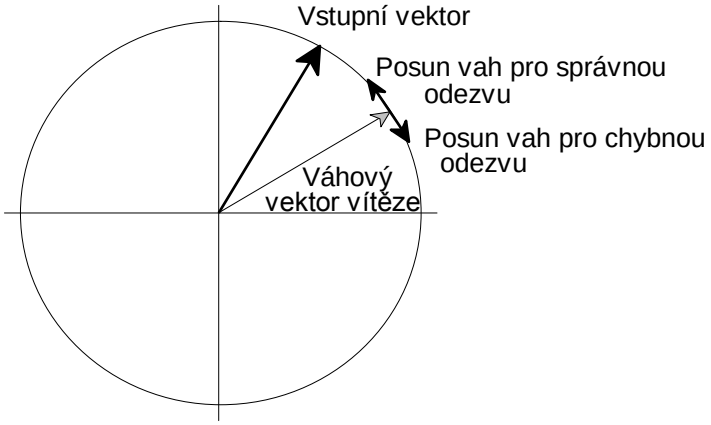
Kvantování vektorů učením (LVQ - Learning Vector Quantization) vychází z uvedených principů Kohonenova učení s jediným rozdílem, že místo již výše zmíněné samoorganizace chceme zajistit aby pro každou kategorii si podobných vektorů existoval jí odpovídající, námi definovaný neuron. Nejprve tedy musíme určit kolik takových kategorií či tříd budeme požadovat. Každé této třídě zřejmě přiřadíme jeden neuron Kohonenovy vrstvy. Následuje proces postupného předkládání vektorů vstupního prostoru a adaptace sítě, tentokráte s učitelem, který rozhoduje o správnosti odezvy. Vlastní odezva je realizována stejným způsobem jako v předchozím případě Kohonenových map postavená na základě kompetice. Klíčový rozdíl je ve způsobu úpravy vah neuronové sítě. V případě, že se jedná o správnou odezvu, adaptace propíhá podle známého vztahu

$$\overline{w}' = \overline{w} + \eta \cdot \left(\overline{x}^i - \overline{w} \right),$$

tentokráte však pouze pro daný vítězný neuron. Sousedství v tomto případě ztrácí smysl, neboť o rozložení kategorií rozhoduje učitel. Zřejmě tímto dochází k přiblížení vah neuronu směrem ke vstupnímu vektoru. V případě chybné odezvy bude našim cílem váhy chybného vítěze spíše oddálit od vstupu, což vede k následujícímu předpisu adaptace jeho vah

$$\overline{w}' = \overline{w} - \eta \cdot \left(\overline{x}^i - \overline{w} \right).$$

Celá situace je znázorněna na následujícím obrázku



Obr. 10-18. Adaptace vah pro neuronovou síť LVQ

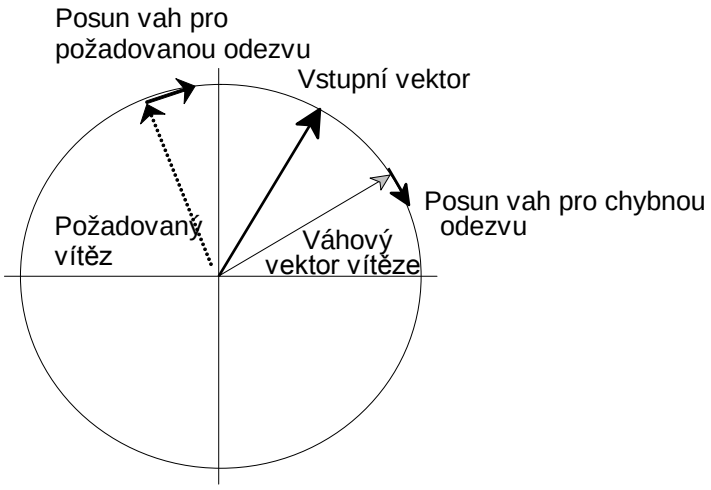
Tento přístup však lze ještě dále zdokonalit. Předpokládejme, že pro předložený vstup se stal vítězem opět *j-tý* neuron namísto *k-tého*. Až doposud jsme zatím uvažovali adaptaci vah pouze u tohoto vítěze. V tomto případě chybné odezvy by tedy došlo k jeho odsunutí od vstupu, zatímco váhy požadovaného *k-tého* neuronu zůstaly nezměněny. Proč tedy v rámci této adaptace neadaptovat váhy požadovaného vítěze tak, aby se přiblížil vstupnímu vektoru? Znamená to tedy, že budeme pro všechny vstupní vektory adaptovat váhy požadovaných neuronů nezávisle na tom, zda jsou či nejsou vítězy soutěže podle vztahu

$\overline{w}_k' = \overline{w}_k + \eta \cdot \left(\overline{x}^i - \overline{w}_k \right)$, kde k vyjadřuje index požadovaného vítěze .

V případě, že vítězem se stal nežádoucí neuron, bude tento odsunut od vstupu dle následujícího pravidla

$\overline{w}_j' = \overline{w}_j + \eta \cdot \left(\overline{x}^i - \overline{w}_j \right)$, kde j vyjadřuje index vítěze kompetice.

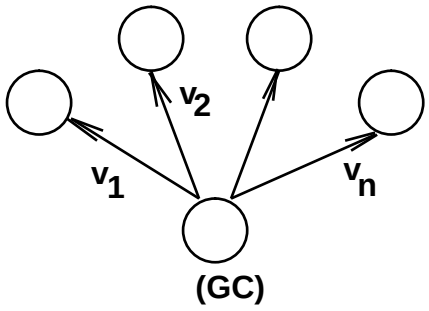
Celá situace je pak ještě dále zobrazena schematicky tímto způsobem



Obr. 10-19. Zdokonalená adaptace vah pro neuronovou síť LVQ

Counter-Propagation

Dříve než přistoupíme k popisu tohoto modelu neuronové sítě, pokusme se nejprve o definování tzv. *Grossbergovy hvězdy*. Topologii této sítě si můžeme představit jako vrstvu neuronů obklopující GC ve středu (viz. obrázek):



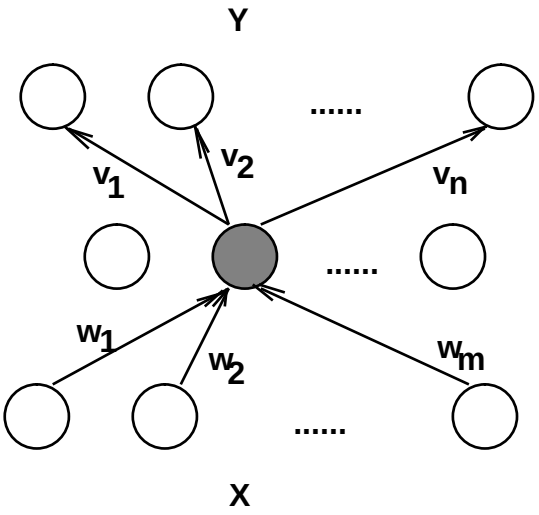
Obr. 10-20. Grossbergova hvězda

V případě, že vstupní neuron je plně excitovaný, pak výstupem je vektor $\bar{v} = \left[v_1, v_2, ..., v_n\right]$. Jestliže se tento liší od požadovaného výstupu $\bar{y} = \left[y_1, y_2, ..., y_n\right]$ dodaného učitelem, pak následuje adaptace vah prostřednictvím Grossbergova pravidla definovaného následujícím způsobem:

$$\bar{v}' = \bar{v} + \mu \cdot \left(\bar{y} - \bar{v}\right).$$

Obdobně jako v případě Kohonenova algoritmu se koeficient učení μ mění v čase, obvykle jeho počáteční hodnota je 0.2 a postupně se při adpataci dosáhne nulové hodnoty. Zřejmě takto vytvářené výstupní shluky sdružují vektory, které jsou si velmi podobné.

V dalším se pokusíme o spojení dvou výše uvedených modelů neuronové sítě. Předpokládejme vytvoření sítě o třech vrstvách neuronů tvořených Kohonenovou mapou a Grossbergovou vrstvou. Každý vstupní vektor tak bude mít ve vnitřní vrstvě svou reprezentaci ve formě GC. Tento vítězný neuron pak tvoří vstup do Grossbergovy hvězdy tak, jak je to znázorněno na obrázku:



Obr. 10-21 Schematické vyjádření Counter-Propagation

Proces adaptace takové sítě podléhá již zvýše zmíněným pravidlům, tedy:

$\overline{w}' = \overline{w} + \eta \cdot (\overline{x} - \overline{w})$ pro Kohonenovu spodní část

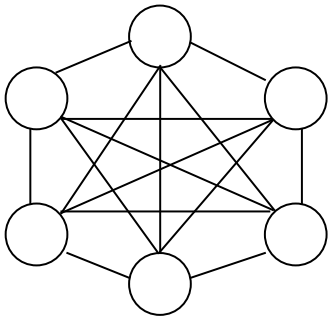
a

$\overline{v}' = \overline{v} + \mu \cdot (\overline{y} - \overline{v})$ pro Grossbergovu výstupní vrstvu.

Nespornou výhodou uvedeného modelu je rychlost jeho adaptace, nevýhodou pak menší přesnost odezvy ve srovnáním s metodou Back-Propagation.

Hopfieldova síť

Autorem této neuronové sítě je John Hopfield, který se zabýval studiem neuronů podobných již dříve uvedeným perceptronům, ale přece jenom s některými podstatnými odlišnostmi. Podstatou problému bylo použití energetické funkce svázané s neuronovou sítí tak, jak je to běžné i u jiných fyzikálních systémů. Hopfieldova síť se skládá z množiny neuronů navzájem úplně v obou směrech propojených (viz. obrázek):



Obr. 10-22. Hopfieldova síť

Váhy sítě jsou symetrické ve smyslu rovnosti $w_{ij} = w_{ji}$. Každý neuron má stejně jako perceptron svůj práh a skokovou funkci aktivační dynamiky, jejíž vstupem je opět vnitřní potenciál daný váhovanou sumou výstupů okolních neuronů. Stav neuronů tedy může být buďto standartní {0, 1} nebo bipolární {-1,+1}. Vzhledem k tomu, že druhý případ je běžnější a také má jasnější matematický aparát budeme se mu také v dalším věnovat.

Hlavní rozdíl Hopfieldova modelu spočívá v tom, že vstup je aplikován na všechny neurony sítě ve formě hodnot -1 a +1, načež následuje cyklus postupných změn excitací neuronů, až do okamžiku dosažení stabilní stavu. Jinými slovy výstupy předchozího kroku se staly novými vstupy současného kroku. Tento proces je vysvětlitelný následujícím schématem. Inicializační stav reprezentuje různorodost exitací neuronů, které vzhledem k tomu, že jsou všechny propojeny, se začnou navzájem ovlivňovat. To může znament, že jeden neuron se snaží neurony excitovat na rozdíl od jiného, který se snaží o opačné. Výsledkem je nalezení kompromisu - síť relaxovala do stabilního stavu.

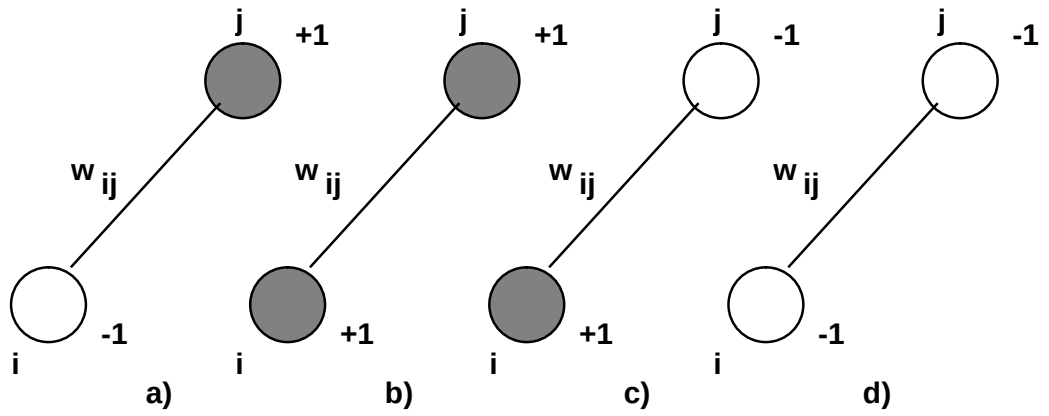
Pokusme se o vyjádření algoritmu Hopfieldovy sítě v následující několika bodech:

1. Přiřazení vah propojením

$$w_{ij} = \begin{cases} \sum_{s=1}^{M} x_i^s \cdot x_j^s & i \neq j \\ 0 & i = j, 0 \leq i, j \leq M \end{cases},$$

kde w_{ij} váha vazby mezi neuron *i* a *j*
x_i^s *i-tý* element vzoru *s* trénovací množiny daný hodnotami {-1,+1}
M počet prvků trénovací množiny.

Smysl této adaptace vah si ozřejmíme na následujícím obrázku:



Obr. 10-23. Adaptace vah Hopfieldovy sítě

V případě varianty b) a d) je stav excitace obou neuronů totožný. Dle výše uvedeného to znamená, že nová hodnota váhy vazby bude dána vztahem:

$$w_{ij}(t+1) = w_{ij}(t) + x_i \cdot x_j = w_{ij}(t) + 1.$$

Znamená to tudíž "posílení" propojení mezi těmito neurony a v případě relaxace sítě pak oba neurony budou mít snahu dosáhnout stejného stavu. Čím více bude vzorů s tímto stavem obou neuronů, tím větší bude snaha o dosažení totožného stavu obou těchto neuronů při relaxaci.

V případech a) a c) pak bude postup obrácený. Tedy nová váha propojení bude mít následující hodnotu:

$$w_{ij}(t+1) = w_{ij}(t) + x_i \cdot x_j = w_{ij}(t) - 1$$

a vazba bude směřovat k takovému stavu, aby při relaxaci sítě stavy obou neuronů byly různé.

2. Inicializace sítě neznámým vzorem
   $\mu_i(0) = x_i, \quad 1 \le i \le N,$

   kde $\mu_i(t)$ stav excitace neuronu i v čase t
   N počet neuronů sítě

3. Iterace až do okamžiku dosažení stabilního stavu

   Relaxuj prostřednictvím vztahu
   $$\mu_i(t+1) = Sgn\left(\sum_{j=1}^{N} w_{ij}\mu_j(t)\right), \quad 1 \le i \le N,$$

   až do okamžiku, kdy
   $\mu_i(t+1) = \mu_i(t), \quad 1 \le i \le N.$

Funkce energie

Princip kódování a následného vyvolání vzorů lze nejlépe vysvětlit prostřednictvím funkce energie Hopfieldovy sítě. Představme si tedy, že tato energetická plocha má svá lokální minima reprezentující předložené vzory ve fázi adaptace sítě, následným vstupem pak definujeme bod na této energetické ploše. Iterační proces relaxace sítě vyjadřuje proces pohybu tohoto bodu po energetické ploše až k některému stabilnímu bodu - atraktoru - reprezentovanému některým jejím lokálním minimem.

Výše uvedeným požadavkům reprezentance vzorů trénovací nožiny ve formě lokálních minim energie a jejich hledání při procesu relaxace odpovídá funkce následujícího tvaru (prahové hodnoty neuronů budeme pro jednoduchost považovat v dalším za nulové):

$$E = -\sum_i \sum_{j \neq i} w_{ij} x_j x_i$$

kde w_{ij} váha propojení mezi uzlem *i* a *j*

x_i stav neuronu *i*.

Podrobnějším zkoumáním uvedeného vztahu pro funkci energie sítě vyplynou následující závěry. Vnitřní suma vlastně reprezentuje vnitřní potenciál neuronu *i* a v případě, že jeho znaménko je odlišné od stavu excitace neuronu (což je ve skutečnosti chybový stav), je hodnota energie vyšší než v korektní situaci, kdy jsou potenciál a excitace neuronu stejného znaménka. Čím více je takových "nesrovnalostí" v neuronové síti, tím větší je tedy i její energie.

Ukládání vzorů

Pokusme si v této části dokázat, že bod 1) Hopfieldova algoritmu skutečně vede k postupnému ukládání vzorů ve formě minimalizace energetické funkce, která tak může vytvářet atraktory ve formě svých lokálních minim.

Nejprve předpokládejme vzor *s* trénovací množiny ve formě vektoru následujícího tvaru:

$$\left(x_1^s, x_2^s, \ldots, x_n^s\right).$$

Hodnoty vah w_{ij} obsahují informaci danou všemi naučenými vzory. Tyto váhy lze tedy rozdělit na dvě části tímto způsobem:

$$w_{ij} = w'_{ij} + w_{ij}^s$$

kde w'_{ij} reprezentuje podíl všech vzorů, kromě vzoru *s*

w_{ij}^s podíl vzoru *s* na váze propojení.

Energetická funkce pak pro vzor *s* nabyde následující podoby:

$$E = -\sum_i \sum_{j \neq i} w'_{ij} x_j^s x_i^s - \sum_i \sum_{j \neq i} w_{ij}^s x_j^s x_i^s,$$

$$E = E_{all-s} + E_s.$$

Tímto způsobem se můžeme dívat na energii jako na součet *šumu* daného ostatními vzory a našeho *signálu*. Uložení vzoru pak znamená minimalizovat podíl vzoru na celkové energii tak, aby tato byla minimální. Zřejmě ona část šumu daná vahami w'_{ij} je našim vzorem *s* neovlivnitelná, takže zbývá minimalizovat druhou část výrazu na pokud možno co nejmenší hodnotu energie

$$E = -\sum_i \sum_{j \neq i} w_{ij}^s x_j^s x_i^s.$$

To znamená maximalizovat hodnotu výrazu

$$\sum_i \sum_{j \neq i} w_{ij}^s x_j^s x_i^s.$$

Vzhledem k tomu, že hodnoty jednotlivých prvků našeho vektoru vzoru *s* jsou rovny -1 či +1, pak jejich čtverec je *vždy* kladný, tedy $x_i^2 \geq 0$ (v dalším pro jednoduchost výrazů vynecháme index s). To znamená, že pokud položíme členy maximalizovaného výrazu úměrné $x_j^2 x_i^2$, budou tyto vždy kladné a tedy i jejich suma bude maximální. Nejjednodušším řešením tedy bude provést následující:

$$\sum_i \sum_{j \neq i} w_{ij}^s x_j x_i = \sum_i \sum_{j \neq i} x_j^2 x_i^2 ,$$

porovnáním pak pro hledané hodnoty vah platí

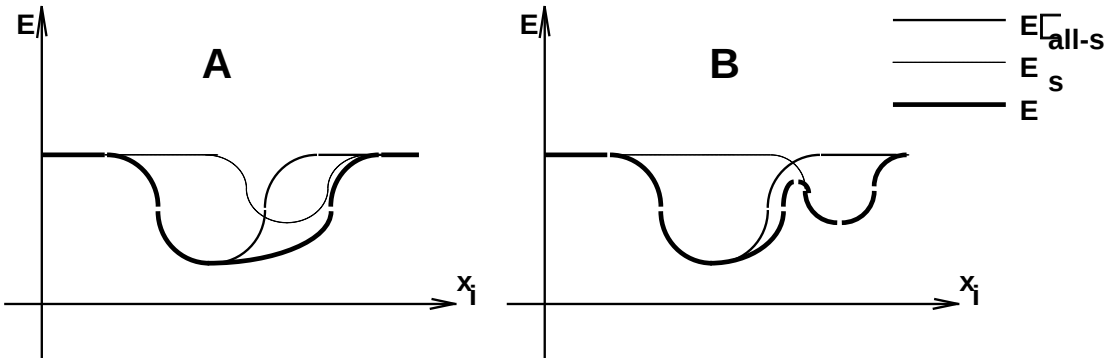
$$w_{ij}^s = x_j x_i .$$

To znamená, že toto nastavení vah vede k minimalizaci hodnoty energie pro vzor *s*. Chceme-li tedy stanovit hodnoty vah pro všechny vzory trénovací množiny, je nutné provést postupnou sumaci hodnot vah odpovídajících každému vzoru následujícím způsobem:

$$w_{ij} = \sum_s w_{ij}^s = \sum_s x_j^s x_i^s ,$$

což je výraz, který jsme měli dokázat.

Je si však třeba uvědomit, že proces minimalizace energetické funkce neznamená nutně vytvoření lokálního minima pro vzor *s*. To je totiž závislé na tvaru energetické funkce dané již výše zmíněným šumem. Tento proces si můžeme znázornit na řezu energetickou plochou, kde pouze varianta B vede k reprezentaci vzoru *s* ve formě lokálního minima energetické funkce:



Obr. 10-24. Minimalizace energetické funkce vzorem *s*

Proces vyvolání informace

Jestliže jsme v předchozí kapitole dosáhli reprezentace vzorů trénovací množiny ve formě lokálních minim energetické funkce (v ideálním případě), pak v této části se pokusíme najít metodu jejich nalezení v případě zadání nějakého vstupu.

Nejprve vyjádříme funkci energie jejím rozložení do několika částí následujícím způsobem:

$$E = -\sum_{i \neq k} \sum_{j \neq k} w_{ij} x_j x_i - x_k \sum_j w_{kj} x_j - x_k \sum_i w_{ik} x_i .$$

Nechť se stav neuronu *k* změní z hodnoty x_{k1} an x_{k2}. Změna energie $\Delta E_k = E_2 - E_1$ způsobená změnou stavu neuronu *k* je tedy dána tímto způsobem:

$$\Delta E_k = -\left[(x_{k2} - x_{k1}) \cdot \sum_j w_{kj} x_j + (x_{k2} - x_{k1}) \cdot \sum_i w_{ik} x_i \right].$$

Uvážíme-li, že vazby mezi neurony jsou symetrické, pak můžeme tento vztah přepsat do tvaru:

$$\Delta E_k = -2 \cdot (x_{k2} - x_{k1}) \cdot \sum_j w_{kj} x_j ,$$

a následně

$$\Delta E_k = -2 \cdot \left[x_{k2} \sum_j w_{kj} x_j - x_{k1} \sum_j w_{kj} x_j \right].$$

Snižování energie sítě vyžaduje splnění podmínky $\Delta E_k \leq 0$ a tedy musí platit:

$$x_{k2} \sum_j w_{kj} x_j - x_{k1} \sum_j w_{kj} x_j \geq 0,$$
$$x_{k2} \sum_j w_{kj} x_j \geq x_{k1} \sum_j w_{kj} x_j . \qquad (*)$$

Stav excitace neuronu je dán již známým způsobem následovně:

$$\sum_{j \neq k} w_{kj} x_j \begin{cases} > 0 & x_{k2} \to +1 \\ = 0 & x_{k2} \to x_{k1}, \\ < 0 & x_{k2} \to -1 \end{cases}$$

a tudíž pro všechny možnosti

$$\sum_j w_{kj} x_j = 0 \quad \vee \quad \sum_j w_{kj} x_j > 0 \quad \vee \quad \sum_j w_{kj} x_j < 0,$$

je nerovnost (*) splněna a tím je splněn i požadavek relaxace sítě až do stabilního stavu daného lokálním minimem energetické funkce sítě. V podstatě je takto dokázán bod 3) Hopfieldova algoritmu.

Boltzmannův stroj

Hlavním problémem uvedeného algoritmu Hopfieldovy sítě je nebezpečí uvíznutí v některém z nežádoucích lokálních minim energie během relaxace sítě. Jako jedno z možných řešení tohoto problému se ukázal model, jehož autorem je S.Kirpatrick. Základní myšlenkou je umožnit systému i dočasná zvýšení energie, která vedou právě k opuštění nežádoucích lokálních minim energie. Na proces relaxace sítě se pak můžeme dívat jako na proces přesunu kuličky po energetické ploše tím způsobem, že s celým systémem je "natřásáno" a kulička má tak možnost vyskočit i ze stabilních míst. Čím větší bude natřásání tím hlubší energetická minima kulička může opustit. Budeme-li však proces "natřásání" postupně snižovat, má kulička menší šanci opustit hluboká minima, zatímco minima mělká jsou ještě překonatelná. To znamená, že přechod z mělčích míst bude mnohem častější než přestup z míst hlubších do mělčích. Kirpatrickův nápad tedy spočívá v následujícím: začít s velkým natřásáním, to však zvolna zmírňovat až do úplného zastavení. To je v podstatě princip žíhání, kdy se materiáln po zahřátí postupně ochlazuje, aby se jeho struktura dostala do energetického minima a tím se odtranilo jeho vnitřní pnutí. Matoda založená na tomto principu se nazývá simulované žíhání.

Hinton, Sejnovski a Ackley rozšířili Hopfieldův model použitím metody simulovaného žíhání a nazvali svůj model na počest zakladatele statistické fyziky *Boltzmannův stroj*. Princip natřásání je zde realizován stochastickým charakterem stanovení stavu excitace neuronu dle následujícího pravidla:

$$p\{x_i(t+1)=1\} = \frac{1}{1+e^{-\Delta E_j/T}},$$

$$\Delta E_j = \sum_j w_{ij} x_j(t) \qquad (*)$$

kde

p	pravděpodobnost, že neuron nabyde stavu excitace jedna
T	teplota systému.

Z uvedeného zřejmě vyplývá, že čím větší bude teplota T, tím je pravděpodobnější, že excitace neuronu bude opačná k jeho potenciálu a tedy dojde k přechodnému zvýšení energie systému. Naopak pro $T \to 0$ přejde vtah v obvyklé pravidlo pro aktivitu Hopfieldova modelu v němž energie systému monotónně klesá až ke stabilnímu energetickému stavu.

Systém jehož chování se řídí tímto pravidlem se po nějakém čase dostane do stavu tepelné rovnováhy, v níž se pravděpodobnost jednotlivých energetických stavů systému nemění a je definována Boltzmannovým rozdělením následujícího tvaru:

$$P_\alpha = k \cdot e^{-E_\alpha/T},$$

kde P_α vyjadřuje pravděpodobnost, že systém dosáhne energetického stavu E_α. Odtud pro porovnání dvou pravděpodobností výskytu stavu α a β platí:

$$\frac{P_\alpha}{P_\beta} = \frac{e^{-E_\alpha/T}}{e^{-E_\beta/T}}.$$

Předpokládejme, že energie stavu α je nižší než stavu β, tedy:

$$E_\alpha < E_\beta,$$

$$e^{-\left(E_\alpha - E_\beta\right)/T} > 1,$$

$$P_\alpha / P_\beta > 1,$$

$$P_\alpha > P_\beta.$$

To znamená, že jakmile systém dosáhne tepelné rovnováhy je dosažení nižšího energetického stavu pravděpodobnější než dosažení stavu o vyšší energii.

Algoritmus učení Boltzmannova stroje

Adaptace vah v tomto případě je založena na pravděpodobnostním přístupu. Nejprve si síť neuronů rozdělíme na tři části. První část bude reprezentovat vtupní neurony, druhá část neurony výstupní a zbylá pak jakoby skrytou vrstvu neuronů. Učení sítě rozdělíme na dvě fáze:

1. Vázaný režim
   a) Navázání a "zajištění" vstupních a výstupních neuronů na vektor trénovací množiny.
   b) Při postupném snižování teploty systému nalezení stabilního stavu.
   c) Zaznamenání excitačních stavů všech neuronů.
   d) Opakování postupu pro všechny prvky trénovací množiny.
   e) Výpočet pravděpodobnosti P_{ij}^{+} přes všechny prvky trénovací množiny, kdy neurony i a j jsou oba excitovány na hodnotu jedna.

2. Volnoběžný režim
   a) Pro náhodnou vstupní excitaci volný běh při postupném snižování teploty až ke stabilnímu stavu.
   b) Velký počet opakování kroku 2a při současném zaznamenání výsledných excitací jednotlivých běhů.
   c) Výpočet pravděpodobnosti P_{ij}^{-}, kdy neurony i a j jsou oba excitovány na hodnotu jedna.

3. Adaptace vah dle vztahu:
   $\Delta w_{ij} = \eta \cdot \left[P_{ij}^{+} - P_{ij}^{-} \right],$
   kde η vyjadřuje koeficient učení.

4. Opakování postupu až po dosažení stabilních vah.

První část algoritmu je v podstatě pravděpodobnostím vyjádřením Hebbova adaptačního pravidla, zatímco druhá část má vlastně stabilizační účinek. V případě, že pravěpodobnost excitace dvou neuronů při volnoběžném chodu je vyšší než při vázaném, mě-li bychom snížit váhu jejich propojení, aby nedocházelo k jejímu předimnenzování vlivem trénovací množiny. Síť relaxuje do požadovaného stavu sama a není třeba ji v tom ještě podporovat. Naopak, jestliže pravděpodobnost excitace dvou neuronů vázaného běhu je vyšší než u volného běhu, pak musíme tuto váhu posílit a vynutit si excitaci obou neuronů. Váhy se zřejmě nemění v případě, že obě četnosti jsou stejné. Jinými slovy, jestliže volnoběžný chod dovede síť k jednotlivým stavům se stejnou pravděpodobností, která je obsažena ve stavech vzorů trénovací množiny, považujeme neuronovou síť za adaptovanou.

Řešení omezujících podmínek

Princip hledání lokálního minima energetické funkce Hopfieldovy sítě je využitelný i jiným způsobem než bylo zatím uvedeno. Dokážeme-li formulovat omezení nějaké optimalizační úlohy ve formě energetické funkce neuronové sítě, pak proces její relaxace povede k nalezení některého z optimálních, či alespoň suboptimálních řešení. Ve srovnání z předchozím tedy nebudeme síť adaptovat na základě prvků trénovací množiny, ale pokusíme se stanovit váhy mezi jednotlivými neurony na základě porovnání obecně definované funkce energie Hopfieldovy sítě a energetické funkce vyjadřující naše omezující podmínky.

Problém obchodního cestujícího

Problém obchodního cestujícího (Travelling Salesman Problem - TSP) je klasickou úlohou, kterou lze uvedeným postupem úspěšně řešit. Cílem úlohy je navštívit všechna města oblasti tak, aby žádné z nich nebylo navštíveno dvakrát a přitom, aby délka trasy byla co nejmenší. Nejlepší řešení pro TSP je velmi složité najít, neboť čas potřebný pro jeho nalezení roste exponenciálně s počtem měst. Proto každé "dostatečně dobré" řešení bude pro nás zajímavým.

Omezeními pro nás bude, že *každé město může být navštíveno pouze jednou a trasa musí být co nejkratší*. Pokud se nám podaří sestavit energetickou fuknci, která bude tato omezení odrážet, pak její minimalizace povede k řešení optimalizující tato omezení. Poněvadž výsledkem je seznam měst navštívených v určitém pořadí, budeme potřebovat nějakým způsobem tento fakt vyjádřit. Jestliže budeme chtít navštívit *n* měst pak každé z nich se bude nacházat v seznamu na některé z *n* pozic. Pro potřeby řešení úlohy TSP tedy použijeme čtvercovou matici *n* x *n* neuronů (všech vzájemně propojených), kde města jsou reprezentována řádky a pořadí sloupci naší matice:

Město/Pořadí	1	2	3	...	n
A	1	0	0	...	0
B	0	0	1	...	0
C	0	0	0	...	1
...	...	...	...	...	...
N	0	1	0	...	0

Z příkladu vyplývá následující pořadí navštívených měst $A \to N \to B \to ... \to C$. Pokusme se nyní formulovat jednotlivá omezení:

1. Podmínka, že každé město je navštíveno maximálně jedenkrát znamená, že v každém řádku je excitován maxímálně jeden neuron.
2. Podmínka, že v daném kroku může být navštíveno maximálně jedno město, je vyjádřena v excitaci maximálně jednoho neuronu v každém sloupci matice.
3. Další podmínka je, že každé město musí být navštíveno právě jedenkrát a matice tedy musí obsahovat právě *n* neuronů excitovaných na hodnotu jedna.

   Realizace těchto tří podmínek vede k následujícímu vyjádření energetické fuknce:

   $$E = A\sum_{M}\sum_{i}\sum_{j\neq i} x_{Mi}x_{Mj} + B\sum_{i}\sum_{M}\sum_{M\neq N} x_{Mi}x_{Ni} + C\left(\sum_{M}\sum_{i} x_{Mi} - n\right)^2,$$

   kde

   M index vyjadřující jméno navštíveného města
   i index vyjadřující pořadí města.

4. Poslední podmínka vydřuje proces minimalizade délky trasy ve formě přidaného čtvrtého členu ve vztahu pro energii systému následujícího tvaru:

$$0.5D\sum_{M}\sum_{N\neq M}\sum_{i} d_{MN}x_{Mi}\left(x_{N,i+1}+x_{N,i-1}\right),$$

kde d_{MN} vzdálenost mezi městy M a N a index i je definován modulo n, tedy platí, že $x_{n+i} = x_i$.

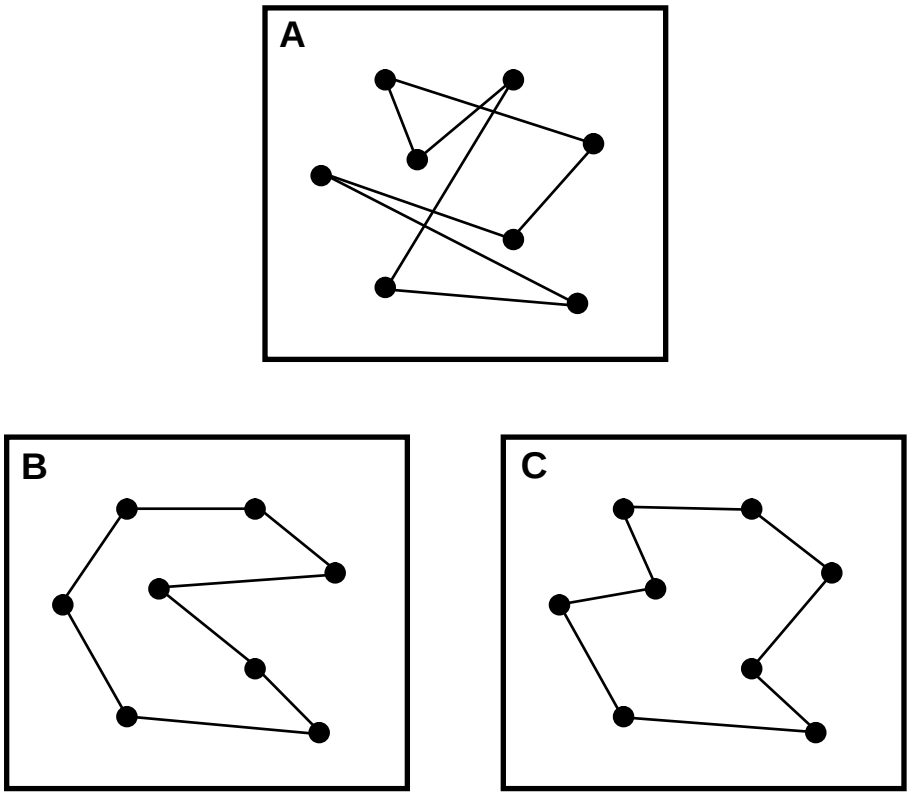
Člen za vzdáleností d je různý od nuly pouze v tom případě, že se obě města vzájemně následují na trase. Zřejmě pak celý tento člen vyjadřuje vzdálenost mezi těmito městy.

Porovnáním se standardní definicí energie získáme vztah pro hodnotu váhy mezi dvěma neurony v této formě:

$$\begin{aligned} w_{Mi,Nj} = & -2A\cdot\delta_{MN}\left(1-\delta_{ij}\right) \\ & -2B\cdot\delta_{ij}\left(1-\delta_{MN}\right) \\ & -2C \\ & -D\cdot d_{MN}\left(\delta_{j,i+1}+\delta_{j,i-1}\right), \end{aligned}$$

kde $\delta_{ij} = 1$pro $i = j$, v ostatních případech 0.

Hopfield s Tankem (1985) realizovali experiment úlohy TSP pro deset měst, přičemž 16 případů z 20 konvergovalo ke správnémů řešení. Navíc pak 50% z nich patřilo k nejkratším trasám nalezených úplným řešením úlohy (viz následující obrázek).



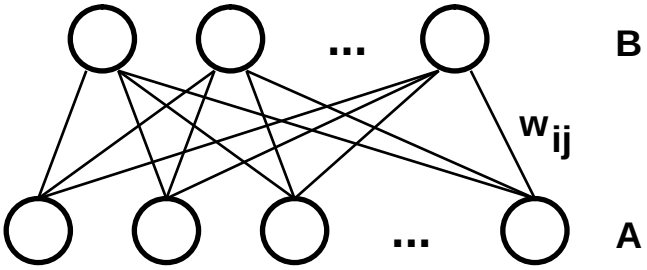
Obr. 10-25. Problém obchodního cestujícího: varianta B je suboptimální, C je optimální

Dvousměrná asociativní paměť

V předcházející kapitole jsme se zabývali tzv. autoasociativní pamětí, což znamená, že informace je vybavena či opravena pouze v rámci téže paměti. To je dáno jedinou vrstvou neuronů vyžadující, aby se výstupní vektor objevil na stejných neuronech, na kterých byl aplikován vstupní vektor.

Dvousměrná asociativní paměť (Bidirectional Associative Memory - BAM) je heteroasociativní, tedy přijímá vstupní vektor na jedné množině neuronů a tento vede k výstupnímu vektoru jiné množiny neuronů. Autorem řady publikací na téma tohoto modelu je pak především B.Kosko a C.Guest.

Struktura BAM je dána dvěmi vrstvami neuronů A a B, přičemž tyto vrstvy jsou vzájemně úplně propojeny obousměrnými vazbami (viz obrázek):



Obr. 10-26. Dvousměrná asociativní paměť

Postup vyvolání informace spočívá v přivedení vstupního vektoru **A**, který prostřednictvím vah **W** vede k vektoru **B** druhé vrstvy. Na tento jsou pak opět aplikovány transponované váhy $\mathbf{W^T}$odukující nový výstup **A.** Tento proces se opakuje až do okamžiku dosažení stabilního stavu, kdy vektory **A** a **B** jsou neměnné. Formálně vyjádřeno tedy platí:

$$b_i = S\left(\sum_j w_{ij} a_j\right),$$

nebo vektorově

$$\overline{B} = S\left(\overline{W} \cdot \overline{A}\right),$$

kde

$\overline{B}$	sloupcový vektor výstupu vrstvy B
$\overline{A}$	sloupcový vektor výstupu vrstvy A
$\overline{W}$	matice vah mezi vrstvou A a B
S	aktivační funkce neuronu.

Obdobně pak platí

$$\overline{A} = S\left(\overline{W}^T \cdot \overline{B}\right),$$

kde

$\overline{W}^T$	transponovaná matice vah.

Postup adaptace vah vychází z Hebbova pravidla dle následujícího vztahu:

$$\overline{W} = \sum_k \overline{A}_k \cdot \overline{B}_k^T,$$

kde *k* vyjadřuje index pořadí vzoru v trénovací množině.

Na následujícím příkladu si ukažme výše uvedené:

Vstupní vektor *Asociovaný vektor*

$A_1 = (1,0,0)$ $B_1 = (0,0,1)$

$A_2 = (0,1,0)$ $B_2 = (0,1,0)$

$A_3 = (0,0,1)$ $B_3 = (1,0,0)$

Bipolární tvar

$A'_1 = (1,-1,-1)$ $B'_1 = (-1,-1,1)$

$A'_2 = (-1,1,-1)$ $B'_2 = (-1,1,-1)$

$A'_3 = (-1,-1,1)$ $B'_3 = (1,-1,-1)$

V dalším budou tyto vektory považovány za sloupcové.
Výpočet vah

$$W = A'_1 B'^T_1 + A'_2 B'^T_2 + A'_3 B'^T_3,$$

tedy

$$W = \begin{bmatrix} -1 & -1 & 3 \\ -1 & 3 & -1 \\ 3 & -1 & -1 \end{bmatrix}.$$

Aplikací vstupního vektoru $A_1 = (1,0,0)$

$$O = W \cdot A = \begin{bmatrix} -1 & -1 & 3 \\ -1 & 3 & -1 \\ 3 & -1 & -1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ 3 \end{bmatrix}.$$

a aplikací pravidla o prahu

$$b_i = \begin{cases} 1 & \text{pro } o_i > 0 \\ 0 & \text{pro } o_i < 0 \\ \textit{stejné} & \text{pro } o_i = 0 \end{cases}$$

obdržíme

$B'_1 = (0,0,1)$.

Nyní je tento vektor transformován do vrstvy A podle vztahu

$$W^T \cdot B'_1 = \begin{bmatrix} -1 & -1 & 3 \\ -1 & 3 & -1 \\ 3 & -1 & -1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ -1 \\ -1 \end{bmatrix},$$

což po aplikaci aktivační funkce vede opět k původnímu vektoru $A_1 = (1,0,0)$.

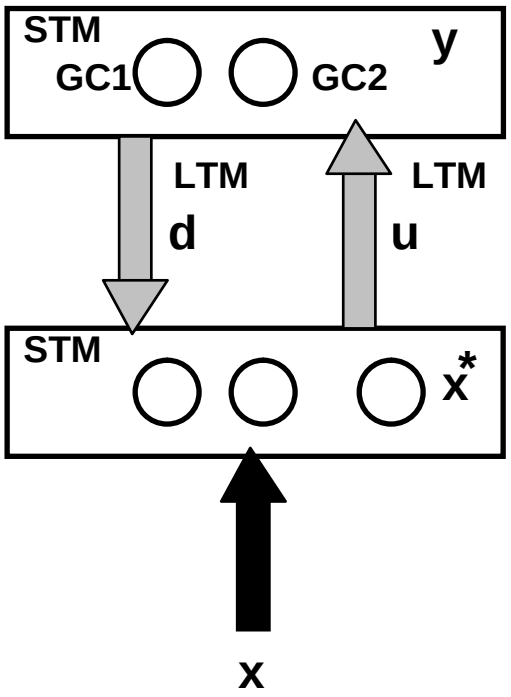
Tímto příkladem byla dokumentována situace, kdy vstup **A** vede přes váhy **W** k vektoru **B**. Tento pak zpět přes transponované váhy $\mathbf{W}^{\mathbf{T}}$ opět produkuje vektor **A**. Takto vlastně dochází k vytvoření rezonance ve smyčce relaxace sítě.

Bohužel snadnost a rychlost uvedeného mechanismu je zatížena omezenou kapacitou takovéto sítě. Kosko odhadl její maximální kapacitu reprezentovanou počtem vzorů zakódovatelných do sítě na hodnotu danou počtem neuronů v menší z obou vrstev. Snahou současného výzkumu a vývoje je zdokonalit BAM tak, aby toto omezení bylo redukováno na maximální možnou míru a využít tak plně možností tohoto modelu neuronové sítě.

Adaptivní rezonanční teorie

Ve všech zatím uvedených modelech neuronových sítí byla zatím přísně oddělena fáze učení a vyvolání informace. Tento fakt výrazně odlišuje uměle vytvořené neuronové sítě od skutečných biologických, kde tyto procesy jsou vzájemně se prolínající. Navíc se tu objevuje velmi nepříjemná situace, kdy naši simulovanou neuronovou síť chceme po čase přizpůsobit novým vzorům. Ty totiž mohou úplně zničit vše, co bylo vytvořeno v minulé etapě adaptace a nezbývá, než celý proces učení započít znovu od začátku, pro všechny prvky trénovací množiny.

Paradigma adaptivní rezonanční teorie (Adaptive Resonance Theory - ART) nevržené Carpenterem a Grossbergem se snaží tento problém odstranit sice poněkud komplikovanějším, nicméně realitě bližším mechanismem. Model takové sítě je tvořen dvěmi vrstvami neuronů *m*-*k* s laterální inhibicí v horní vrstvě. Obě vrstvy jsou úplně propojeny v obou směrech, tentokráte jsou ale váhy vazeb v různých směrech různé. Označíme si pomocí ***u*** [bottom-Up] váhy od vrstvy nižší k vrsvě vyšší (u_{ij} je váha propojení mezi neuronem *j* nižší vrstvy a neuronem *i* vrstvy vyšší) a prostřednictvím ***d*** [top-Down] váhy propojení od vrstvy vyšší směrem k vrstvě nižší (analogicky d_{ji}). Ačkoliv obě v závorkách uvedené vazby propojují tytéž neurony obecně jsou jejich hodnoty různé. Neurony vyšší vrstvy pak splňují role GC, tak jak byly popsány v kapitole o kompetici neuronů. Vektory $\boldsymbol{u_j}$ a $\boldsymbol{d_j}$ pak budou označovat propojení neuronu GCj s neurony nižší vrstvy. Na následujícím obrázku je pak znázorněn schematický model sítě ART.



Obr. 10-27. Obecné schéma modelu ART

Podle toho, k čemu jednotlivé vrstvy neuronů slouží, získaly tyto následující označení. Spodní vrstva se nazývá *srovnávací vrstva* (comparison layer), zatímco horní se nazývá jako *rozpoznávací vrstva* (recognition layer).

Pokusme se nyní o hrubý popis mechanismu modelu ART:

- Předložení vstupního vektoru ***x*** srovnávací vrstvě. Ten aktivuje vzor v podobě krátkodobé paměti (STM).
- Prostřednictvím vazeb reprezentované ***u*** a pomocí laterální inhibice je vybrán neuron s největší excitací jako GC.
- Vítězný neuron pak vyšle signál ***d*** směrem k nižší vrstvě. Tento se nazývá *očekávání* a je odvozen z předchozí zkušenosti. Tento signál pak aktivuje ve srovnávací vrstvě nový vzor ***x****.
- Původní vektor ***x*** je porovnán se zpětnovazebným signálem.

- Jestliže podobnost mezi oběma vektory (vstupním a zpětnovazebným), tedy mezi realitou a očekáváním, je menší než předdefinovaná hodnota ρ (*vigilance* - ostražitost), pak vybraný neuron GC nereprezentuje správnou třídu, do které patří náš vstup a je tedy odstraněn z množiny možných vítězů.
- Pokud existuje další možný kandidát mezi zbývajícími GC jsou tyto otestovány stejným způsobem. Pakliže žádný vhodný GC neexistuje, je vtažen do procesu další zatím neangažovaný neuron, který bude garantovat správné rozpoznání našeho vstupu.
- V případě, že rozdíl mezi vstupem a zpětnovazebním signálem je dostatečně malý, tedy GC reprezentuje vektor x, dochází k jevu r*ezonance*, k souladu mezi vstupem a očekáváním.
- Dojde-li k rezonanci, pak následuje proces adaptace vah sítě prostřednictvím *Weberova zákona*, který bude popsán níže.

V dalším se budeme zabývat modelem ART1, který akceptuje a je schopen rozpoznávat pouze binární vektory s prvky {0,1}. Modely ART2 a ART3 jsou pak schopny pracovat se vstupy dané intervalem 0 a 1. Omezení binárních vstupů se pak odráží i v hodnotách d_{ij}, kterou jsou také binární, zatímco váhy u_{ij} jsou reprezentovány reálnými čísly.

Pokud dva binární vektory ***a*** a ***b*** jsou stejné dimenze, pak vektor ***a&b*** bude reprezentovat průnik obou vektorů daný aplikací logické funkce AND na jejich jednotlivé komponenty. Pomocí $|\boldsymbol{a}|$ označíme "velikost" vektoru ***a***, tedy počet jedniček v tomto vektoru.

Pokusme se v dalším o detailnější specifikaci mechanismu modelu ART:

1. Inicializace vigilance ρ (v rozmezí 0,1) a vah následujícím způsobem:
   $u_{ij} < L/(L-1+m)$

   kde

m	počet neuronů ve vstupní vrstvě (počet komponent vstupního vektoru)
L	konstanta ≥1.

   $d_{ij} = 1$
   pro všechny i a j.

2. Předložení vstupního vektoru ***x*** a nalezení neuronu s největší excitací (proces rozpoznávání) podle vztahu:

   $$\sum_i u_{ji} x_i .$$

   Prostřednictvím laterální inhibice bude tento po čase excitován na hodnotu 1 a ostatní budou mít hodnotu 0. Označme jej indexem *j*. V případě více stejně excitovaných neuronů vezmeme první z nich.

3. Tento neuron GCj vyšle zpětnovazební signál $\boldsymbol{d_j}$ (očekávání) zpět do srovnávací vrstvy. V této vrstvě dojde k porovnání binárních vektorů vstupu ***x*** a očekávání $\boldsymbol{d_j}$ prostřednictvím vypočtení průniku ***x**** obou vektorů. Tento pak stanoví jak podobný je vstup očekávání dle následujícího vztahu:

   $$\sigma = \frac{|x^*|}{|x|}.$$

   Zřejmě dokonalé porovnání vede k hodnotě podobnosti 1, zatímco nejhorší případ je roven 0.

4. V případě, že podobnost je dostatečná $\sigma > \rho$ (případ rezonance) provede se adaptace vah následujícím způsobem:

$d_j = x^*$ (proces abstrakce)

$u_j = L \cdot d_j \Big/ \left(L - 1 + \left| d_j \right| \right)$,

kde význam jmenovatele spočívá v "normalizaci" nově definovaných vah, pro L=1 platí

$u_j = d_j \Big/ \left| d_j \right|$.

Pokud ale platí $\sigma \leq \rho$ je neuron GCj vyloučen z další kompetice neboť nevyhovuje našemu vstupu ***x*** (nulování) a hledá se další neuron GCs vyhovující podmínce podobnosti. Pokud však žádný z existujících GC nevyhovuje, je do procesu vtažen další zatím neangažovaný neuron rozpoznávací vrstvy, jehož váhy jsou inicializovány dle bodu 1. Vzhledem k tomu, že vektor ***d*** tohoto neuronu má všechny komponenty jednotkové, průnik se vstupním vektorem vede k výsledné hodnotě podobnosti rovné jedné a dojde tedy k potřebnému jevu rezonance.

Pokusme se nyní celý mechanismus demonstrovat na malém příkladu, kde m=5, L=2 a ρ=1/2. Vstupní prostor je dán vektory

$x^1 = [10000], \quad x^2 = [11100], \quad x^3 = [11000]$.

Váhy $\boldsymbol{u_j}$ jsou inicializovány na hodnoty

$u_j = \left[1/6 \; 1/6 \; 1/6 \; 1/6 \; 1/6 \right]$.

V následující tabulce je prostřednictvím tučného písma označen reprezentant vstupu, ξj vyjadřuje vstupní potenciál, ∅ označuje, že daný GC nevyhrál kompetici a znak ⇐ odpovídá nutnosti nulování původního vítěze kompetice:

x/j	původní $\boldsymbol{u_j}$	ξj/původní $\boldsymbol{d_j}$	***x****=nový $\boldsymbol{d_j}$	σ	nový $\boldsymbol{u_j}$
10000 **1**	1/6 1/6 1/6 1/6 1/6	1/6 11111	10000	1	10000
11100 1	10000	1 10000	10000	1/3	⇐
2	1/6 1/6 1/6 1/6 1/6	1/2 11111	11100	1	1/2 1/2 1/2 00
11000 1	10000	1 10000	10000	1/2	⇐
2	1/2 1/2 1/2 00	1 11100	11000	1	2/3 2/3 000
10000 **1**	10000	1 10000	10000	1	10000
2	2/3 2/3 000	2/3∅			
11100 1	10000	1∅			
2	2/3 2/3 000	4/3 11000	11000	2/3	2/3 2/3 000
11000 1	10000	1∅			
2	2/3 2/3 000	4/3 11000	11000	1	2/3 2/3 000

Pokusme se ještě jednou o vysvělení procesu "normalizace" adaptovaných vah. Z čitatele vyplývá fakt, že velké vektory (obsahující velký počet jednotkových komponent) vedou k menším hodnotám vah, než vektory malé. Tato vlastnost umožňuje od sebe oddělit dva vektory, i když jeden z nich je podmnožinou druhého. Mějme například dva vektory následujícího tvaru:

$x^1 = [10000]$,

$x^2 = [11100]$.

Vektor x^1 je ve skutečnosti podmnožinou x^2, přičemž ale požadujeme, aby tyto byly reprezentovány rozdílnými neurony GC. Bez uvažované normalizace by vektory vah obou GC měly následující tvar:

$u_1 = d_1 = [10000]$,

$u_2 = d_2 = [11100]$.

Opětovným předložením prvého vektoru může nastat situace, že vítězem se stane druhý neuron GC (oba mají stejný vstupní potenciál), podobnost je rovna jedné a následnou adaptací dojde ke stanovení nových vah totožných s vahami prvého neuronu GC: došlo ke ztrátě druhého vzoru. Při použití "normalizace" s hodnotou L=2, budou hodnoty vah GC neuronů následující:

$u_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix}$,

$u_2 = \begin{bmatrix} 1/2 & 1/2 & 1/2 & 0 & 0 \end{bmatrix}$.

Opětovné předložení prvého vektoru vede k potenciálu 1 prvého neuronu GC, zatímco potenciál druhého z nich je roven pouze 1/2. Stejně tak druhý vstupní vektor vede k potenciálu prvého neuronu GC o hodnotě 1, kdežto potenciál druhého je roven 3/2. V obou případech tak byly zachovány oba vzory a nedošlo k jejich ztrátě jako v předchozím případě.

Obdobný význam má i inicializace vah na hodnoty

$u_{ij} < L/(L-1+m)$,

která zabezpečí, že již naučené vzory nebudou excitovat zatím neangažované neurony z rozpoznávací vrstvy. Z uvedeného příkladu vyplývá, že inicializační hodnoty vah by tedy měly menší než 1/3. Uvažujeme-li jejich hodnoty rovny 1/6, pak první vstupní vektor vede k hodnotě potenciálu neangažovaného neuronu 1/6 a druhý vstupní vektor vede k hodnotě 1/2. V obou případech je to hodnota nižší než potenciály neuronů pro které byly trénovány.

Jak je patrné z výše uvedeného ART skutečně řeší v úvodu definované problémy adaptace neuronové sítě na nové vzory bez nebezpečí destrukce již dříve naučeného. Navíc architektura ART je velmi věrohodná i z biologického pohledu, neboť mechanismus ART je konzistentní se skutečnými procesy probíhajícími v mozku.

Objektově-orientovaný přístup k neuronovým sítím

Pokusme se v rámci této kapitoly o implementaci většiny uvedených modelů neuronových sítí prostřednictvím objektově-orientované technologie, která dnes tvoří základ moderního softwarového inženýrství.

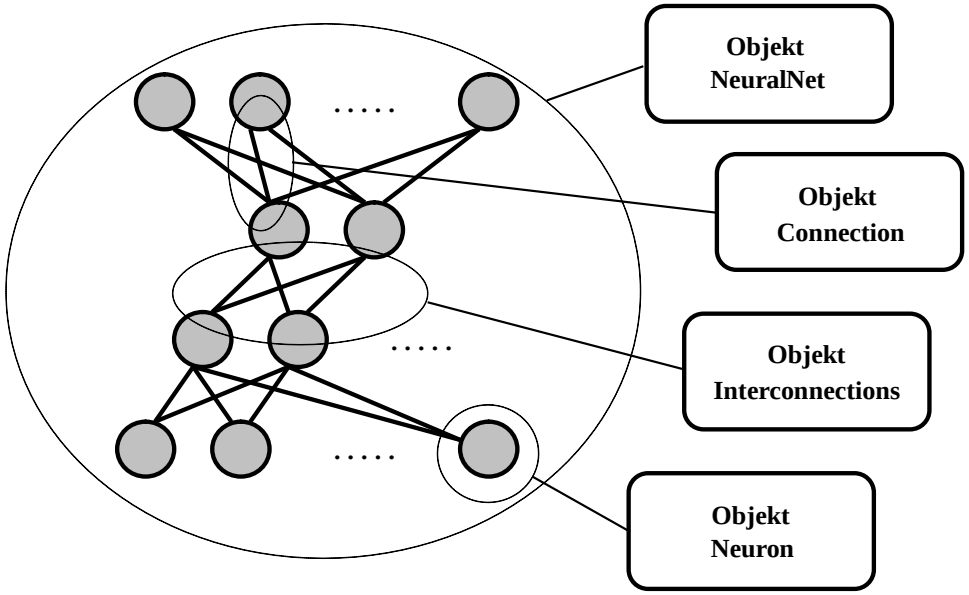
Objektově orientovaný systém lze reprezentovat jako množinu objektů komunikujících navzájem, kde cílem těchto interakcí je dosažení specifikovaného cíle. Každý takový objekt může být chápán jako malý virtuální počítač daný svým stavem (pamětí) a vlastní množinou operací (instrukční sada). Výpočet je prováděn již výše zmíněnou komunikací - zasíláním zpráv mezi objekty. Tento koncept je velmi blízký paradigma neuronových sítí, které jsou tvořeny množinou samostatně fungujících neuronů charakteristických svým stavem a chováním, přičež i zde proces výpočtu je reprezentován cestou předávání informace mezi jednotlivými neurony po jejich axonových vláknech.

Pokusme se nyní o komplexní přístup k implementaci neuronových sítí cestou objektově-orientované technologie tvořenou následujícími třemi etapami:

1. Objektově orientovaná analýza - OOA (Object-Oriented Analysis)
2. Objektově orientovaný návrh - OOD (Object-Oriented Design)
3. Objektově orientované programování (kódování a vlastní implementace)
   - OOP (Object-Oriented Programming)

Analýza problému - OOA

Cílem této části je identifikovat, které objekty tvoří modelovaný systém. V našem případě se jedná o neurony, vazby mezi neurony, množiny těchto vazeb reprezentující vrstvy a konečně vlastní neuronové sítě.



Obr. 10-28. Dekompozice problému

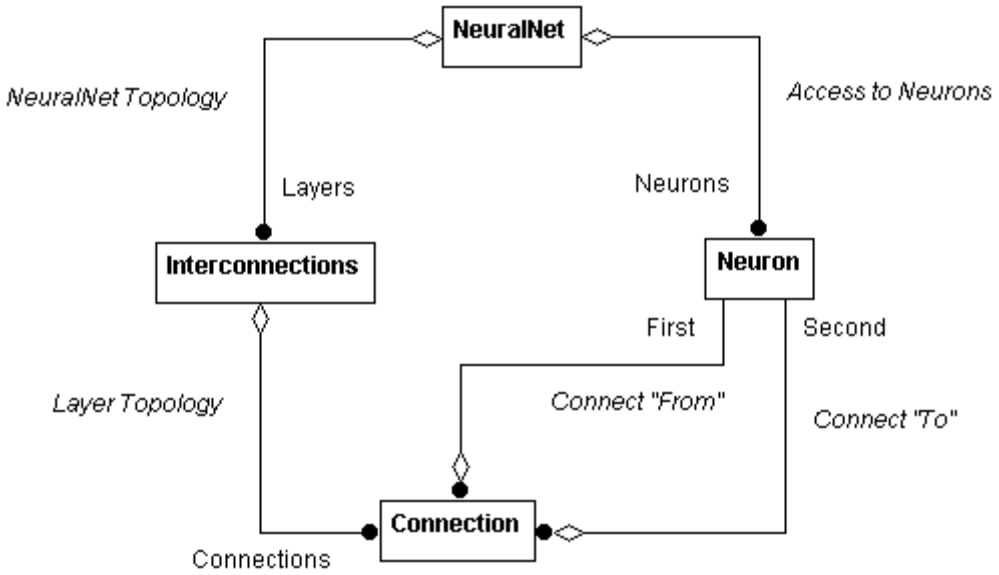
Výsledkem tohoto procesu je nalezení čtyř základních abstraktních datových typů (tříd) obsažených v naší úloze:

- třída ***Neuron***, jejíž instance budou tvořit neurony sítě
- třída ***Connection***, jejíž instance reprezentuje vazbu mezi dvěma neurony
- třída ***Interconnections*** tvořící abstraktní popis pro konkrétní množinu vazeb, obvykle reprezentovanou vrstvou
- třída ***NeuralNet***, ze které budou vytvářeny jednotlivé neuronové sítě konkrétních topologií a chování.

Návrh a implementace - OOD, OOP

Jestliže jsme v předchozí části identifikovali třídy a objekty nutné pro modelování neuronových sítí, pak nyní můžeme přistoupit k jejich návrhu a následně i k implementaci. Cílem dříve uvedeného je definice relací mezi třídami včetně definice základních metod popisujících chování jednotlivých objektů. V rámci implementace uvedeneme na tomto místě pouze některé z algoritmů. Detailní řešení nechť je záležitostí čtenáře, který má tak možnost teoretického ověření znalostí získaných v předchozích kapitolách.

Základní relace mezi třídami lze shrnout do dvou kategorií - asociace a dědičnosti. První typ relace vyjadřuje fzyické, či koncepční propojení mezi instancemi daných tříd. V případě neuronových sítí se bude jednat o relace následného schematického znázornění (Boochova notace):



Obr.10-29. Relace asociace mezi třídami

Z obrázku vyplývá že neuronová síť je jednak tvořena vrstvami vazeb definujících její topologii a dále pak množinou neuronů, k jejichž stavovým veličinám má přístup. Vlastní neurony pak vystupují v rolích prvého (First) resp. druhého neuronu (Second) ve vazbách zajišťujících jejich propojení. Tímto pořadí je taktéž definován i směr šíření signálu. Jednotlivé vazby pak definují topologii množiny propojení, obvykle reprezentující vrstvu ve vícevrstvých neuronových sítích.

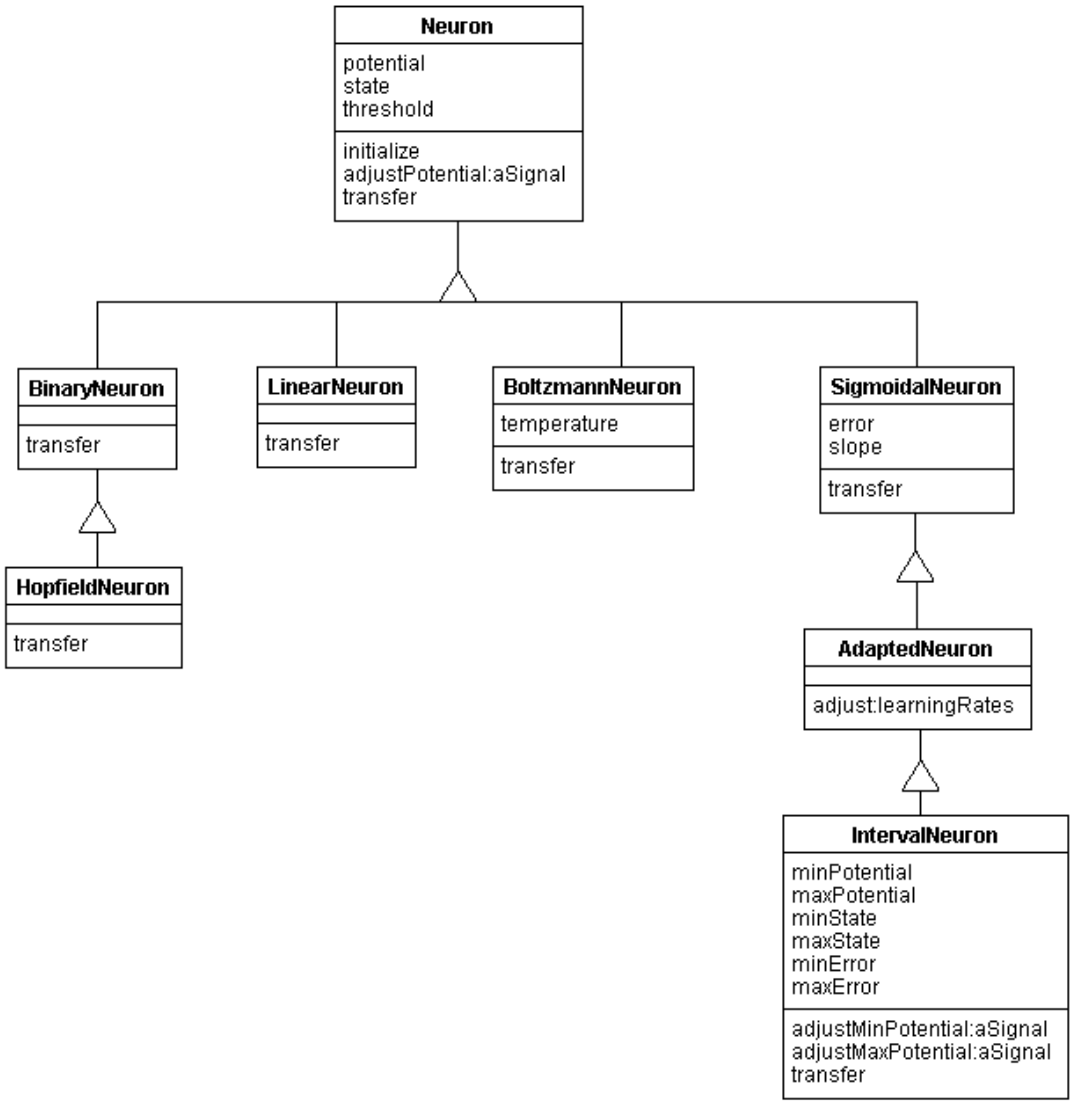
Zřejmě se jednotlivé modely neuronových sítí liší v typech neuronů, vazeb, ve způsobu jejich propojení i adaptace. Tato vlastnost je pak podchycena v hierarchiích jednotlivých stavebních prvků i neuronových sítích samotných.

Hierarchie neuronů

Popis této hierarchie započneme popisem abstraktního neuronu, který reprezentuje nejobecnější strukturu neuronu (popisovaný kód bude využívat syntaxe jazyka Smalltalk pro definici jednotlivých tříd a jejich metod, metody umožňující prostý přístup k datovým elementům zde nebudeme uvádět):

class: *Neuron*
superclass: *Object*
data elements:
potential "vnitřní potenciál"
state "stav excitace"
threshold "prahová hodnota"
message protocol:
initialize "inicializace neuronu
adjustPotential: *aSignal* "přidej přijatý signál k potenciálu
transfer "abstraktní aktivační funkce

Je vcelku zřejmé, že právě metoda transfer bude tou metodou, jejíž definice se bude měnit podle typu neuronu. Následující obrázek pak popisuje hierarchii všech základních typů neuronů včetně jednotlivých instančních proměnných a metod spjatých s konkrétní třídou:



Obr. 10-30. Hierarchie neuronů

Binární neuron je reprezentován dvěma excitačními stavy 0,1 a jeho aktivační funkce je dána tímto předpisem:

transfer
potential > threshold
ifTrue: *[state := 1]* ***ifFalse:*** *[state := 0]*

Ostatní neurony v hierarchii odráží jejich vlastnosti. Např. lineární neuron implementuje lineární aktivační funkci, zatímco neuron se spojitou sigmoidální aktivací, používaný především v sítích s adaptací na bázi metody backpropagation, musí kromě jiné metody ***transfer*** implementovat také veličinu *error* reprezentující chybu na neuronu a hodnotu *slope* vyjadřující strmost neuronu.

Hierarchie vazeb

Vazba mezi neurony je důležitá nejen z hlediska zajištění propojení dvou neuronů, ale také z důvodu definice topologie neuronové sítě. Jedná se v podstatě o prvou úroveň definice této topologie. Nejobecnější popis vazby je dán následujícím způsobem:

class: *Connection*
superclass: *Object*
data elements:

first	"prvý neuron z propojované dvojice"
second	"druhý neuron"
weight	"váha, síla propojení"

message protocol:

initialize	"inicializace vazby"
adjust: *aValue*	"nastavení váhy vazby"
passSignal	"předání signálu od prvého k směrem druhému neuronu"

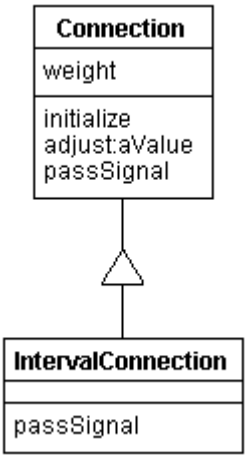
Metoda ***adjust*** jednoduše upravuje váhu vazby hodnotou *aValue* předanou jako argument ve zprávě zaslané konkrétní instance třídy *Connection*:

adjust: *aValue*
weight := weight + aValue

Signál je předán od prvého neuronu směrem k druhému prostřednictvím metody ***passSignal***, která je definována následujícím způsobem:

passSignal
second ***adjustPotential:*** *(weight * (first* ***state****))*

Celá hierarchie je velmi jednoduchá a má pouze dvě třídy:



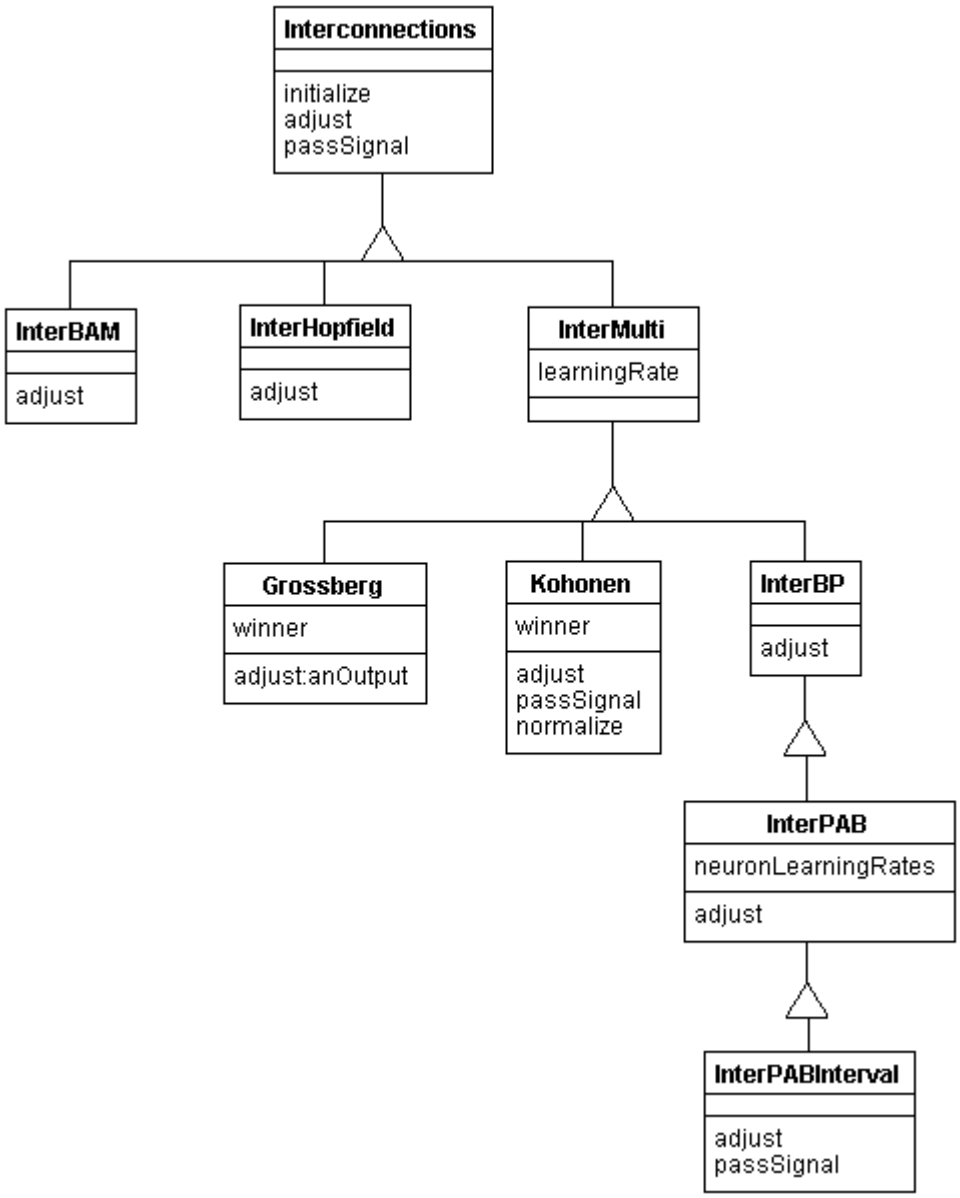
Obr.10-31. Hierarchie vazeb

Hierarchie tříd Interconnections

Množina vazeb tvořících "vrstvu" propojení tvoří druhou vyšší úroveň definice topologie neuronové sítě. Stejně jako v předchozích případech, i zde je každý typ propojení reprezentován svou třídou a celá hierarchie má jeden kořen - abstraktní třídu následující definice:

class: *Interconnections*
superclass: *Object*
data elements:
connections "dynamické pole vazeb"
message protocol:
initialize "inicializace vrstvy propojení"
adjust "nastavení vah vazeb propojení"
passSignal "předání signálu od vstupujích neuronů k vystupujícím"

Celá hierarchie odrážející různé typy propojení neuronu je následující:



Obr.10-32. Hierarchie vrstev propojení

Jak již bylo výše uvedeno, základ hierarchie tvoří třída Interconnections, která implementuje protokol základních zpráv celé hierarchie. Tyto jsou následně modifikovány nebo doplněny o další podle potřeby konkrétního typu propojení. Metoda passSignal je podobná pro všechny a je definována takto:

```
passSignal
        "Pass signal through interconnections."
        | neuron |
        connections do: [ :con |                    "projdi všechny bazby"
                neuron := con second.               "urči druhý neuron ve vazbě"
                neuron initialize.                  "inicializuj jeho potenciál"
                connections do: [ :c |              "nastav jeho potenciál signálem z
                                                     předcházejících neuronů"
                        (c second) = neuron
                        ifTrue: [
                                c passSignal
                        ]
                ].
                neuron transfer                     "nastav excitaci neuronu"
        ].
```

Tento algoritmus je shodný pro většinu tříd z dané hierarchie. Oproti tomu, algoritmus adaptace reprezentovaný metodou ***adjust*** se liší podle konkrétního typu vrstvy propojení. Ukážeme si alespoň jeden příklad za všechny popisující již dobře známou metodu *backpropagation*:

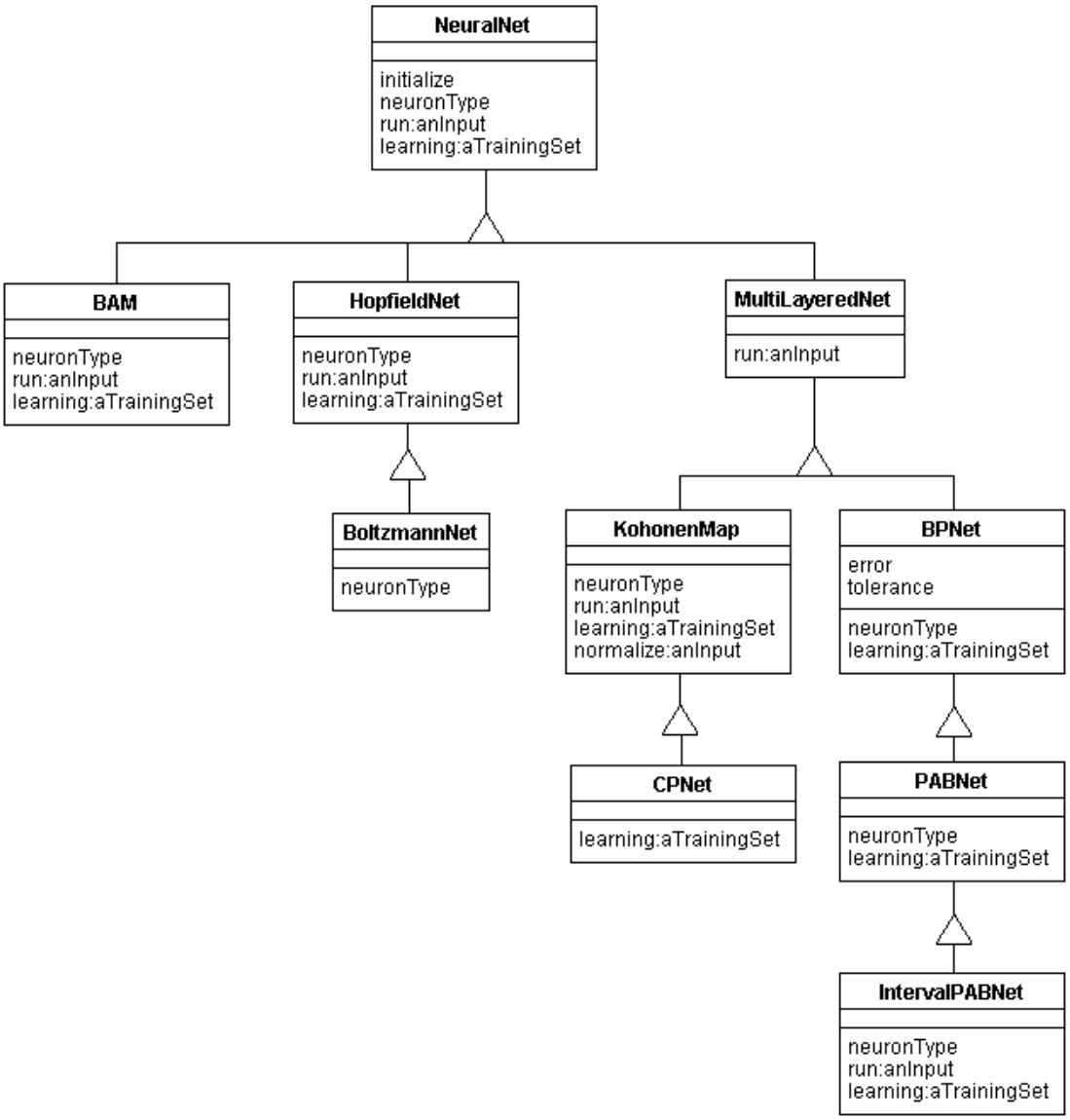
```
adjust
        "Uprav vazby dle BP."
        |  x xi d delta lambda|
        connections do: [ :con |                            "projdi všechny vazby"
                x  := con second state.
                xi := con first state.
                d  := con second error.
                lambda := con second slope.
                delta     :=  -1 * learningRate * d * (x * (1 - x)) * lambda * xi.
                con adjust: delta.                          "nastav váhy"
        ]
```

Hierarchie neuronových sítí

Hierarchie neuronových sítí je založena na třídách definující jednotlivé typy propojení. V podstatě to znamená, že vrstvy jsou skládány dohromady s cílem vytvořit požadovaný model neuronové sítě. Základní, abstraktní třídu hierarchie neuronových sítí tvoří třída NeuralNet, která je definována následujícím způsobem:

```
class:              NeuralNet
superclass:         Object
data elements:
        inter                           "dynamické pole vrstev propojení"
message protocol:
        initialize              "inicializace neuronové sítě"
        neuronType                      "vrací typ neuronu používaného v dané síti"
        learning: aTrainingSet          "adaptace sítě"
        run: anInput                    "vyvolání informace na základě předloženého stimulu"
```

Hierarchie tříd reprezentující všechny základní modely neuronových sítí má pak tuto strukturu:



Obr.10-33. Hierarchie neuronových sítí

Principy použití jsou analogické s předchozí hierarchií vrstev propojení s jedinou výjimkou, kterou je metoda ***neuronType***. Tato je předefinována ve všech třídách hierarchie s cílem vrátit instanci třídy neuronu používaného v daném konkrétním modelu neuronové sítě. Odtud tedy bude platit, že metoda ***neuronType*** implementovaná v rámci třídy *BPNet* vrácí instanci třídy *SigmoidalNeuron*, zatímco stejná metoda, tentokráte z třídy KohonenMap, bude vracet instanci třídy *LinearNeuron*.

Přínos předvedeného přístupu nespočívá pouze v efektivním softwarovém řešení uvedené problematiky, ale také v možnosti unifikovat (na bázi objektového modelu) jednotlivé modely neuronových sítí.

Použitá literatura

Beale,R.-Jackson,T.: Neural Computing: An Introduction, IOP Publishing, Bristol and Philadelphia 1990

Bratko,L.: Prolog Programming for Artificial Intelligence, Addison-Wesley, Reading, Mass. 1986

Caudill,M.: A Little Knowledge is a Dangerous Thing, AI Expert, June 1993, Miller Freeman Publications, San Francisco, CA, 1993

Caudill,M.: Putting Time in a Bottle, Neural Network Special Report '93, AI Expert, Miller Freeman Publications, San Francisco, CA, 1993

Clocksin,W.F -Mellish,C.S.: Programming in Prolog, Springer-Verlag, Berlin. 1981

Csonto,J.: Prolog v príkladoch, Elektrotechnická fakulta VŠT Košice, 1988

Feigenbaum,E.A.-Barr,A.: The Handbook of Artificial Intelligence, Vol I-IV, Addison-Wesley, Reading, Mass. 1989

Firebaugh,M.W.: Artificial Intelligence, A Knowledge-Based Approach, PWS-Kent Publishing Company, Boston, Mass. 1989

Goldberg,A.-Robson,D.: Smalltalk-80 The Language, Addison-Wesley, Reading, Mass. 1989

Goldberg,D.E.: Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading, Mass. 1989

Havel,I.M.: Robotika. Úvod do teorie kognitiních robotů, SNTL Praha, 1981

Hecht-Nielsen,R.: Neurocomputing, Addison-Wesley, Reading, Mass. 1990

Hořejš,J.: Tutorials: A View on Neural Networks Paradigms Development, Part 1.-9, IDG Czechoslovakia, 1991-92

Hu, D.: C/C++ for Expert Systems, MIS Press, Portland, Oregon, 1989

Jelínek,J. (sestavení): Metody umělé inteligence, sborník přednášek, FEL ČVUT, Praha, 1984

Kufudaki O., Hořejš J.: PAB: Parameters Adapting Back Propagation, Neural Network World, num.5 vol.1, IDG Czechoslovakia, 1991

Mařík,V.(sestavení): Aplikace umělé inteligence '87, sborník přednášek, DT ČSVTS, Praha, 1987

Mařík,V.-Zdráhal,Z.(sestavení): Metody umělé inteligence a expertní systémy, sborník přednášek, DT ČSVTS, Praha, 1985

Popper,M.-Kelemen,J.: Expertné systémy, Alfa Bratislava, 1988

Rumbaugh,J.-Blaha,M.-Premerlani,W.-Eddy,F.-Lorensen,W.: Object-Oriented Modeling and Design Prentice-Hall, New Jersey, 1991

Starling,L.-Shapiro,E.: The Art of Prolog, The MIT Press, Mass. 1987

Šíma,J.: Multi-Layered Neural Network as an Adaptive Expert System With the Ability to Work with Incomplete Information and to Provide Justification of Inference, Neural Network World, num.1 vol.2, IDG Czechoslovakia, 1992

Šíma,J.: Generalized Back Propagation for Interval Training Patterns, Neural Network World, num.2 vol.2, IDG Czechoslovakia, 1992

Virius,M.: Objektově orientované programování, FJFI ČVUT Praha, 1992

Vondrák,I.: Umělá inteligence, Přírodovědecká fakulta, Univerzita Palackého Olomouc, 1991

Vondrák,I.: Prázdný expertní systém IVEXPERT, Automatizace č.4, SNTL, Praha, 1990

Vondrák,I.: Object-Oriented Neural Networks, AI Expert, June 1994, Miller Freeman Publications, San Francisco, CA, 1994

Vondrák,I.: Neuronový Neurex. CHIP magazine, č.9, Praha 1992

Vondrák,I.: Neuronové sítě a expertní systémy. Automatizace č.12, roč.35, SNTL Praha 1992

Vondrák,I.: Neural Network and Reliablity of Knowledge Base. ESREL '93. European Safety and Reliability Conference, Elsevier Science Publishers, Munich 1993, Germany

Vondrák,I.: Neurex. Expertní systém na bázi neuronových sítí. PC WORLD, 5/93, IDG Czechoslovakia 1993

Vondrák,I.: Object-Oriented Design of Artificial Neural Networks. NEURONET '93. International Scientific Conference, Prague 1993

Vondrák,I.: Genetic Algorithm and Neural Network Adaptation, Neural Network World, num.2 vol.4, IDG Czechoslovakia, VSP International Science Publishers, The Netherlands, 1994

Vondrák,I.: Object-Oriented Design of Artificial Neural Networks, Neural Network World, num.4 vol.4, IDG Czechoslovakia, VSP International Science Publishers, The Netherlands, 1994

Wasserman, P.D.: Neural Computing, Theory and Practice. Van Nostrand Reinhold, NY, 1989

Waterman,D.E.: A Guide to Expert Systems, Addison-Wesley, Reading, Mass. 1986

Zadeh,L.A.: The Role of Fuzzy Logic in Management of Uncertainty in Expert Systems, Fuzzy Sets and Systems, 11, 1983