Umělé neuronové sítě

Studijní materiál pro Letní školu učitelů informatiky

Lipnice, 18. - 29. 8. 1997

Mrázová Iveta

1 Úvod do problematiky umělých neuronových sítí

V posledním desetiletí jsme svědky rychlého rozvoje umělých neuronových sítí. Podobně jako biologické neuronové sítě představují velice silný nástroj, k jehož hlavním přednostem patří především robustnost ať už vzhledem ke ztrátě funkční jednotky, tak také vzhledem k drobným odchylkám předkládaných vzorů a schopnost zobecňovat zkušenosti získané v procesu učení. Tyto vlastnosti jsou spolu s paralelním způsobem zpracovávání informací nesmírně důležité především v takových oblastech, jako je zpracování mluvené řeči, obrazu a mnoha dalších. Za umělou neuronovou síť považujeme obecně takovou strukturu pro distribuované a paralelní zpracování informací, jejímiž základními funkčními jednotkami jsou tzv. formální neurony. Každý z těchto neuronů může současně přijímat libovolný konečný počet různých vstupních informací a tuto informaci předávat dalším neuronům prostřednictvím svého jediného, avšak rozvětveného výstupu.

Nejjednodušší model neuronu počítá vážený součet vstupů, který předává dále pomocí tzv. nelineární přenosové funkce. Formální neuron je tedy charakterizován svými vahami, které vedou informaci ze vstupů tohoto neuronu, prahem a typem použité přenosové funkce (tou může být např. tzv. skoková přenosová funkce - viz obr. 2, resp. tzv. sigmoidální přenosová funkce - viz obr. 3, ap.). Graf neuronové sítě pak odpovídá tzv. topologii neuronové sítě. Jednotlivé uzly tohoto grafu odpovídají neuronům a hrany grafu, které jsou obvykle ohodnoceny velikostí příslušných synaptických vah, reprezentují příslušné synaptické spoje mezi odpovídajícími neurony.

Jeden z nejstarších a dosud nejrozšířenějších modelů neuronu, často označovaný jako tzv. formální neuron, navrhli již v roce 1943 McCulloch a Pitts. Tento model zavedl myšlenku skokové přenosové funkce, chyběl mu však klíčový faktor - schopnost učení. Matematický pojem učení zavedl v roce 1949 D. Hebb. Podle Hebbovského pravidla učení se účinnost (t.j. váha) synapse zvětšuje, následuje-li bezprostředně po presynaptické aktivitě aktivita postsynaptická. Pozdější verze Hebbovského učení jsou popsány jako vzrůst synaptických spojů úměrný korelaci presynaptických a postsynaptických potenciálů. Rosenblatt v roce 1958 zobecnil pojem formálního neuronu rozšířením o fenomén učení a nazval ho perceptronem. Váhy mezi vstupní a výstupní vrstvou se adaptují v závislosti na odchylce mezi skutečnými a požadovanými výstupy.

V roce 1962 navrhli Widrow a Hoff svou neuronovou síť podobnou perceptronu a nazvali ji adaptivní lineární neuron neboli ADALINE. Klasický model perceptronu využívá

skokovou přenosovou funkci a jeho výstupy jsou binární hodnoty. Naproti tomu model ADALINE využívá sigmoidální přenosovou funkci a vytváří "spojité" výstupy. Widrow a Hoff ukázali, že chyba mezi skutečnou a požadovanou odezvou dosáhne za určitých podmínek (lineárně separabilní množina vstupních vzorů) svého globálního minima. V 60. letech vydali Minsky a Papert knihu "Perceptrony". Ukázali v ní, že perceptrony lze úspěšně použít jen pro úlohy s lineárně separabilními prostory řešení. Další rozvoj problematiky umělých neuronových sítí a neuropočítačů pak začal až ve druhé polovině 80. let.

V roce 1982 zavedl J. Hopfield model asociativní paměti používaný i při řešení optimalizačních úloh (např. problému obchodního cestujícího). T. Kohonen navrhl v roce 1984 adaptivní vektorový kvantizátor nazývaný také jako samoorganizující se příznaková mapa. V podstatě se jedná o autoasociativní klasifikátor podle nejbližšího souseda, který se pomocí korekce chyby učí klasifikovat vstupní vzory do předem daného počtu tříd. Tato obvykle dvouvrstvá síť pracuje v diskrétním čase. Výsledky procesu učení přitom mohou záviset na posloupnosti předkládaných vstupních vzorů - a to obzvlášť v případě, že je jejich počet malý.

V roce 1987 navrhl R. Hecht-Nielsen model sítí se vstřícným šířením. Tento typ sítí se skládá ze dvou navzájem symetrických částí a byl navržen k aproximaci spojitých zobrazení a zobrazení inverzních. V jistém smyslu představují sítě se vstřícným šířením zobecnění Kohonenových map. Od roku 1987 je ovšem jednou z nejpoužívanějších metod v oblasti umělých neuronových sítí algoritmus zpětného šíření. Tato metoda je založená na principu tzv. učení s učitelem a používá se k adaptaci vah ve vrstevnatých neuronových sítích.

2 Základní pojmy

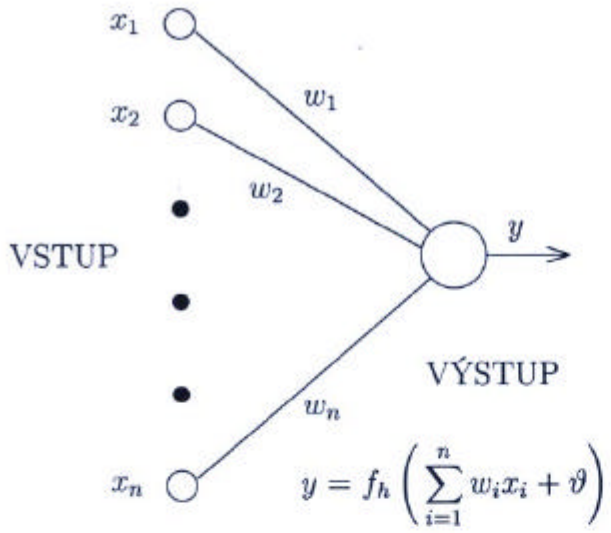
Každý neuron nejprve počítá hodnotu tzv. vnitřního potenciálu, která odpovídá váženému součtu jednotlivých složek vstupního vektoru a přičte k nim tzv. prahovou hodnotu, ϑ. Na výsledek potom aplikuje nelineární přenosovou funkci. V případě skokové přenosové funkce tedy bude výstup neuronu y roven 1 anebo -1. V situaci znázorněné na obrázku 1 pak bude rozhodovací pravidlo dávat odezvu odpovídající třídě A, jestliže bude výstup neuronu roven 1, a odezvu odpovídající třídě B, jestliže bude výstup neuronu roven -1.

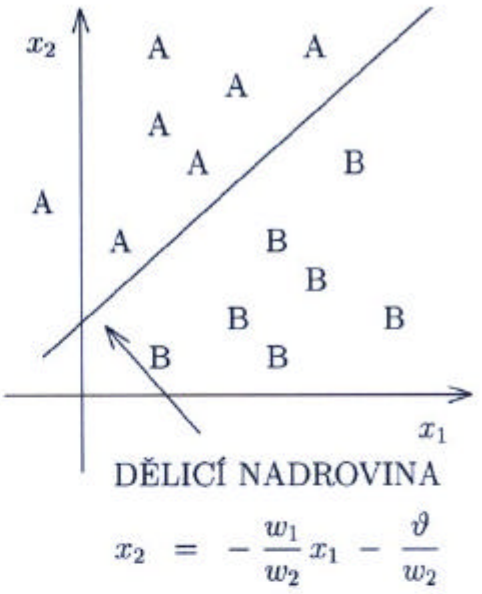
Formální neuron tedy definuje dělicí nadrovinu, která rozděluje příznakový prostor na dvě oblasti. Rovnice dělicí nadroviny přitom závisí na vahách a prahu daného neuronu. Ve zmíněném dvourozměrném případě tak dělicí nadrovina odpovídá přímce, přičemž vzory nad touto přímkou jsou označeny jako vzory patřící do třídy A. Vzory ležící pod touto přímkou jsou označeny jako náležející do třídy B. Přesněji lze formální neuron popsat takto:

Neuron s vahami $(w_1, \ldots, w_n) \in R^n$, prahem $\vartheta \in R$ a přenosovou funkcí f, kde $f : R^{n+1} \times R^n \longrightarrow R$, počítá pro libovolný vstup $\vec{z} \in R^n$ svůj výstup $y \in R$ jako hodnotu přenosové funkce v $\vec{z}$, $f[\vec{w}, \vartheta](\vec{z})$.

Jako přenosová funkce se nejčastěji uvažuje již zmíněná skoková přenosová funkce anebo sigmoidální přenosová funkce definovaná pomocí:

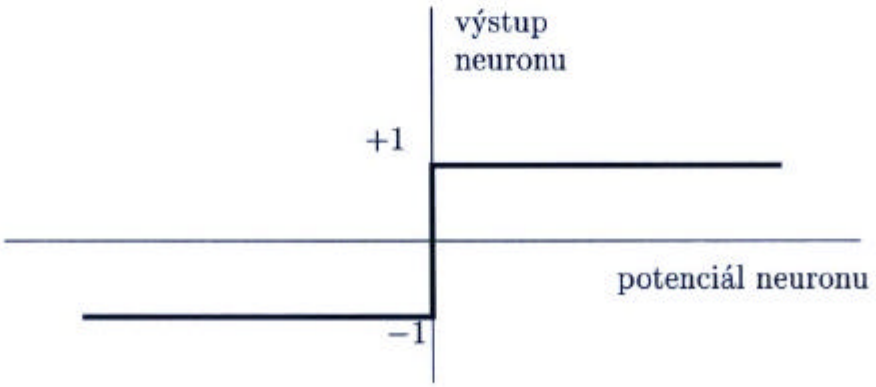
$$y \;=\; f[\vec{w}, \vartheta]\,(\vec{z}) \;=\; f(\xi) \;=\; \frac{1}{1 + e^{-\xi}} \qquad ;$$



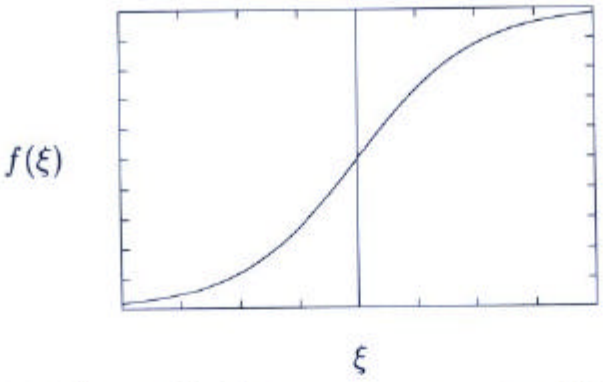


$$y = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} w_i x_i + \vartheta \geq 0 \implies \text{TŘÍDA A} \\ -1 & \text{if } \sum_{i=1}^{n} w_i x_i + \vartheta < 0 \implies \text{TŘÍDA B} \end{cases}$$

Obrázek 1: **Formální neuron:** Jednotlivé neurony počítají vážený součet n vstupů, který dále předávají pomocí tzv. nelineární přenosové funkce. Neuron s vahami $(w_1, \ldots, w_n)$, prahem, ϑ, a skokovou přenosovou funkcí, f_h, tak může vstupní vektory $(x_1, \ldots, x_n)$ klasifikovat podle výstupní hodnoty neuronu, y, do dvou tříd označených jako A a B. Příslušná dělicí nadrovina (anebo přímka v případě dvourozměrného vstupního prostoru) rozděluje vstupní prostor do dvou oblastí.



Obrázek 2: **Skoková přenosová funkce.**



Obrázek 3: **Sigmoida:** Výstupní hodnotu neuronu s potenciálem ξ lze určit pomocí sigmoidální přenosové funkce $f(\xi)$: $y = f(\xi) = 1/(1 + e^{-\xi})$.

kde $\xi = \sum_{i=1}^{n} z_i w_i + \vartheta$ označuje tzv. potenciál neuronu a R množinu reálných čísel.

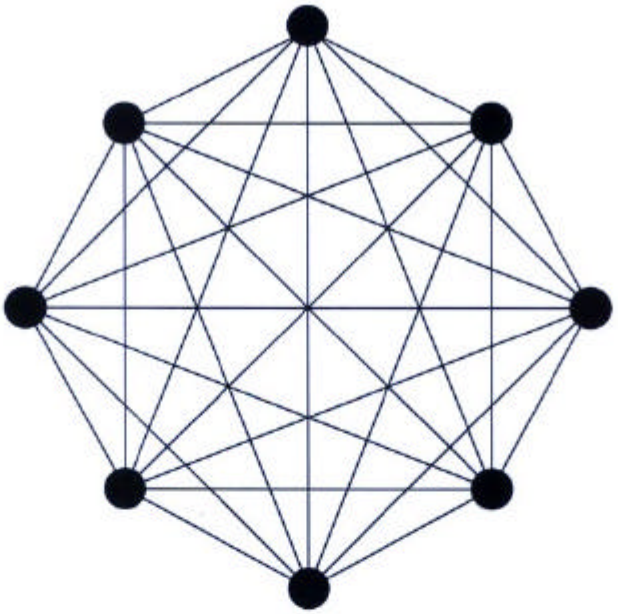
Umělá neuronová síť se skládá z konečné neprázdné množiny neuronů navzájem propojených synaptickými spoji. Přesněji ji lze popsat následujícím způsobem:

Umělá neuronová síť je uspořádaná 6-tice $M = (N, C, I, O, w, t)$, kde:

- N je konečná množina neuronů,
- $C \subseteq N \times N$ je množina orientovaných spojů mezi neurony,
- $I \subseteq N$ je množina vstupních neuronů,
- $O \subseteq N$ je množina výstupních neuronů,
- $w : C \longrightarrow R$ je váhová funkce; R označuje množinu všech reálných čísel,
- $t : N \longrightarrow R$ je prahová funkce.

Pro umělou neuronovou síť M s n vstupními a m výstupními neurony budeme dále používat tato označení:

- Vstupní vzor označuje vstupní vektor $\vec{x} \in R^n$ zpracovávaný sítí.
- Požadovaný výstup $\vec{d} = (d_1, \ldots, d_m)$ tvoří požadované výstupy neuronů výstupní vrstvy.
- Pro daný vstupní vzor představuje skutečný výstup B vektor $\vec{y} = (y_1, \ldots, y_m)$ tvořený skutečnými výstupy neuronů výstupní vrstvy.



Obrázek 4: **Model Hopfieldovy sítě.**

3 Hopfieldův model

Tento typ sítí se používá především jako asociativní paměť anebo při řešení optimalizačních úloh. V případě použití Hopfieldova modelu jako asociativní paměti většinou předpokládáme, že každý z n neuronů sítě je propojen se všemi ostatními neurony v síti. Každý neuron i se skokovou přenosovou funkcí a binárními vstupy kódovanými jako $+1$, resp. -1 tedy může předávat všem ostatním neuronům j ; $i \leq j$ informaci o svém stavu pomocí synaptického spoje s váhou w_{ij}. Váhy mezi jednotlivými neurony by přitom měly být symetrické - mělo by tedy platit, že $w_{ij} = w_{ji}$. Zároveň by mělo být omezeno ovlivňování neuronu sebe samým - to znamená, že by mělo platit: $w_{ii} = 0$; $\forall i$.

Vzory z trénovací množiny lze do "paměti sítě" uložit například pomocí níže uvedeného vztahu pro nastavení vah založeného na principu Hebbovského učení. Poté lze síti předložit nový neznámý vzor. V procesu rozpoznávání síť iterativně aktualizuje stav jednotlivých neuronů. Poté, co síť zkonverguje ke stabilnímu stavu, kdy se už výstupní hodnoty jednotlivých neuronů nemění, reprezentuje výstup sítě rozpoznaný vzor. Kromě toho lze ukázat, že Hopfieldova síť s postupně aktualizovanými stavy svých neuronů zkonverguje z libovolného počátečního stavu některého ze stabilních stavů.

Použití Hopfieldova modelu jako asociativní paměti má však dva zásadní nedostatky. Prvním z nich je relativně malý počet vzorů, které lze "bezpečně uložit" do paměti sítě. Překročení kapacity paměti pak může vést k rozpoznávání tzv. "falešných vzorů". Počet "bezpečně" uložených vzorů se přitom pohybuje kolem $0.15n$, kde n označuje počet neuronů sítě. Další nevýhodou je poměrně velká nestabilita těch vzorů, které jsou si navzájem hodně podobné (t.j. ty vzory, které mají větší počet shodných složek příznakového vektoru). V procesu rozpoznávání potom může Hopfieldova síť zkonvergovat k ne zcela odpovídajícímu vzoru i v případě trénovacích vzorů.

Hopfieldův model asociativní paměti

Krok 1: Nastavení vah sítě
Váhy sítě nastav pomocí:

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

V tomto vztahu představuje hodnota w_{ij} váhu spoje mezi neuronem i a j. x_i^s může nabývat hodnot $+1$ anebo -1 a představuje i-tou složku trénovacího vzoru ze třídy s.

Krok 2: Předlož síti nový vstupní vzor
Předlož síti nový vstupní vzor $\vec{x}$ ve tvaru $x_1, x_2, ..., x_n$ a nastav počáteční stav neuronů sítě pomocí:

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

kde $y_i(t)$ představuje výstup neuronu i v čase t a x_i označuje i-tou složku vstupního vzoru, která může nabývat hodnoty $+1$ anebo -1.

Krok 3: Opakuj, dokud síť nezkonverguje do stabilního stavu
Aktualizuj stav jednotlivých neuronů v síti pomocí:

$$y_j(t+1) = f_h \left[\sum_{i=1}^{n} w_{ij} \, y_i(t) \right] \quad , \quad 1 \leq j \leq n \; .$$

Funkce f_h přitom označuje skokovou přenosovou funkci. Aktualizace stavu neuronů se bude provádět tak dlouho, dokud se budou měnit výstupní hodnoty neuronů sítě. Při ukončení procesu vybavování (t.j. aktualizace stavu neuronů sítě) tedy bude platit: $y_j(t+1) = y_j(t) \; ; \; 1 \leq j \leq n$. Výstupy neuronů pak budou reprezentovat ten z p uložených vzorů, který nejlépe odpovídá předloženému vzoru $\vec{x}$.

Krok 4: Přejdi ke Kroku 2 a opakuj pro další vzor

4 Vrstevnaté neuronové sítě a algoritmus zpětného šíření

V oblasti umělých neuronových sítí patří v současné době k nejpoužívanějším model vrstevnatých neuronových sítí typu zpětného šíření. Někdy se tento model označuje také jako BP-sítě. Oblast využití tohoto modelu sahá od počítačového vidění přes robotiku, řízení, medicínu a ekonomii až po umělou inteligenci. Obvykle se tyto sítě používají ke klasifikaci vzorů s pevným počtem příznaků, které mohou nabývat i spojitých hodnot. Při učení je vždy síti předkládán vstupní vzor zároveň s požadovaným výstupem. Vrstevnaté neuronové sítě typu zpětného šíření se skládají ze vstupní vrstvy (obsahující všechny vstupní neurony), několika mezilehlých vrstev (tzv. skrytých vrstev obsahujících skryté neurony) a výstupní vrstvy (obsahující všechny výstupní neurony). Umělé neuronové sítě tohoto typu však neobsahují synaptické váhy spojující neurony uvnitř jednotlivých vrstev ani synaptické váhy jdoucí z vyšších vrstev do nižších nebo přeskakující jednu anebo více vrstev. Aktivita neuronů se v každé vrstvě určí podle:

$$\xi_j = \sum_i y_i w_{ij} \qquad \text{a} \qquad y_j = f(\xi_j) = \frac{1}{1 + e^{-\lambda \xi_j}}$$

Celkový vážený vstup ξ_j neuronu j (tzv. potenciál neuronu) je dán součtem výstupních aktivit y_i neuronů předchozí vrstvy, které jsou vynásobeny váhou příslušné synapse w_{ij} mezi neuronem i a j. Výstupní aktivita neuronu j, y_j, je určena nelineární přenosovou funkcí f potenciálu daného neuronu. Neurony uvnitř jedné vrstvy přitom mění svou aktivitu paralelně, a to postupně po jednotlivých vrstvách od vstupní vrstvy směrem k výstupní. Parametr λ se nazývá strmost přenosové funkce (a může mít případně různou hodnotu pro každý neuron).

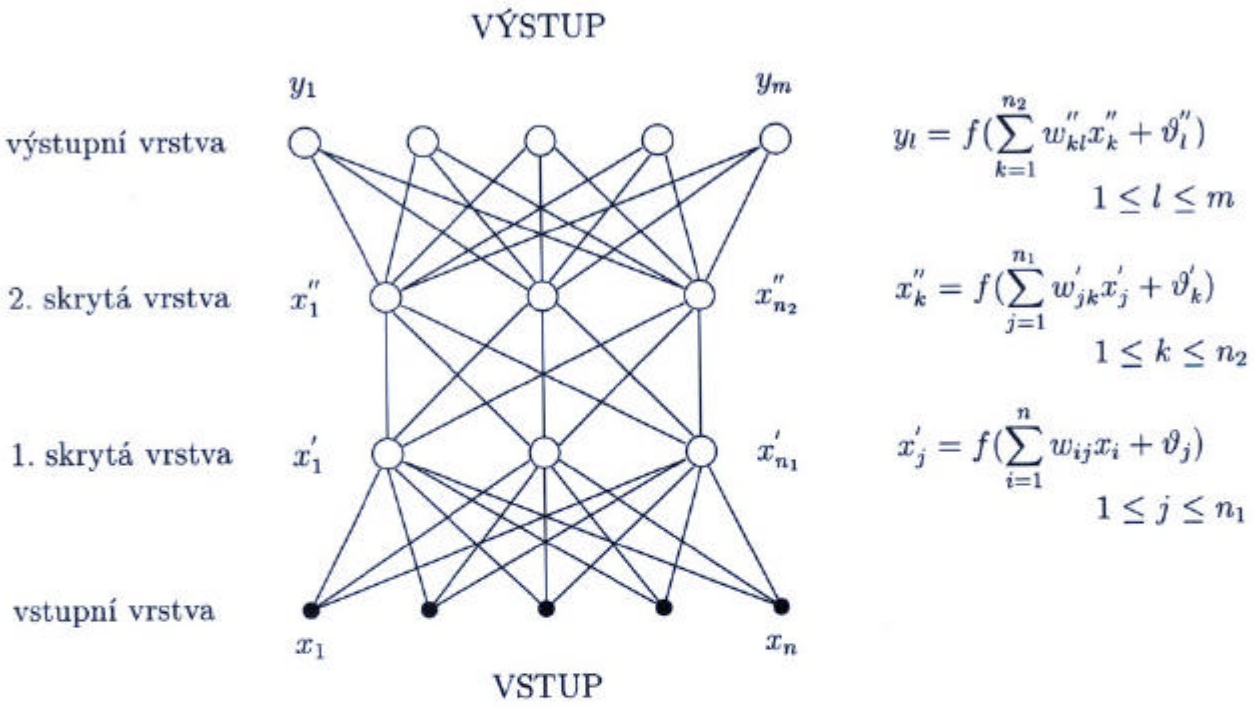
Vrstevnaté neuronové sítě se "učí správně reagovat na možné vstupy" postupným předkládáním vzorů z trénovací množiny (učení s učitelem). Přitom se pomocí algoritmu zpětného šíření adaptují váhy spojů mezi jednotlivými neurony sítě. Cílem algoritmu zpětného šíření je nalézt takovou množinu vah, která by zaručovala pro všechny vstupní vzory z trénovací množiny to, že skutečný výstup dané neuronové sítě bude stejný jako její požadovaný výstup. Úloha přitom nespecifikuje ani skutečnou ani požadovanou aktivitu skrytých neuronů. Teprve v procesu učení se musí síť "sama rozhodnout", za jakých okolností má být ten který skrytý neuron aktivní a přispět tak k dosažení požadovaného chování sítě.

Požadované chování sítě lze formulovat pomocí tzv. cílové funkce, jejíž hodnota by se měla v procesu učení zmenšovat. Pro konečnou množinu trénovacích vzorů ve tvaru (vstupní_vzor/požadovaný_výstup) lze cílovou funkci sítě vyjádřit pomocí rozdílu mezi skutečným a požadovaným výstupem u každého předloženého vzoru. Cílová funkce E, označovaná někdy i jako chybová anebo účelová funkce, je pak definovaná jako:

$$E = \frac{1}{2} \sum_p \sum_j (y_{j,p} - d_{j,p})^2$$

kde p je index označující předkládané vzory z trénovací množiny, j indexuje výstupní neurony, y představuje skutečný výstup příslušného neuronu a d jeho požadovaný výstup.

K minimalizaci chybové funkce E se používá gradientní metoda. Přitom je nutné nastavit v průběhu učení hodnoty všech synaptických vah v síti tak, aby byla hodnota



Obrázek 5: **Čtyřvrstvá síť** s n vstupními neurony, m výstupními neurony a dvěma skrytými vrstvami s n_1, resp. n_2 skrytými neurony. Jako nelinearitu lze použít například sigmoidální přenosovou funkci znázorněnou na obrázku Figure 3. Hodnoty x'_j a x''_k zde představují výstupy neuronů z první, resp. druhé skryté vrstvy, ϑ'_k a ϑ''_l značí prahy těchto neuronů, w_{ij} označuje váhu synapse mezi vstupním neuronem a neuronem z první skryté vrstvy a w'_{ij}, resp. w''_{ij} jsou vahami mezi první a druhou skrytou vrstvou, resp. mezi druhou skrytou vrstvou a vrstvou výstupní.

funkce E co možná nejmenší. Hodnotu každé váhy w_{ij} je tedy třeba změnit o hodnotu odpovídající záporné parciální derivaci E podle této váhy, $-\partial E/\partial w_{ij}$. Označíle-li tento člen jako $\Delta_E w_{ij}$, platí:

$$\Delta_E w_{ij} \quad = \quad -\frac{\partial E}{\partial w_{ij}}$$

Prahy neuronů přitom lze reprezentovat pomocí vah vycházejících z fiktivního neuronu s konstantním výstupem rovným 1. Pro jednoduchost dále nebudeme uvažovat index p ani u požadovaných, ani u skutečných výstupních hodnot neuronů, d and y. Z podobných důvodů budeme dále označovat neurony z vrstvy pod příslušným neuronem j jako i a neurony z vrstvy nad neuronem j jako k. K výpočtu hodnot $\Delta_E w_{ij}$ lze použít standardní algoritmus zpětného šíření. Pro předložený trénovací vzor je nejprve třeba určit skutečný výstup sítě a ten potom porovnat s požadovaným výstupem. Na základě zjištěné odchylky už lze pro každou váhu v dané síti spočítat zápornou parciální derivaci E podle této váhy, $\Delta_E w_{ij}$:

– pro výstupní vrstvu

$$\begin{aligned}
\Delta_E w_{ij} \quad &= \quad -\frac{\partial E}{\partial w_{ij}} \quad = \quad -\frac{\partial E}{\partial \xi_j}\frac{\partial \xi_j}{\partial w_{ij}} \quad = \quad -\frac{\partial E}{\partial \xi_j}\frac{\partial}{\partial w_{ij}}\sum_{i'} w_{i'j}\, y_{i'} \quad = \\
&= \quad -\frac{\partial E}{\partial \xi_j}\, y_i \quad = \quad -\frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial \xi_j}\, y_i \quad = \quad (d_j - y_j)\,\lambda\, y_j\,(1 - y_j)\, y_i \quad = \\
&= \quad \delta_j\, y_i
\end{aligned}$$

– a pro skryté vrstvy

$$\begin{aligned}
\Delta_E w_{ij} \quad &= \quad -\sum_k \frac{\partial E}{\partial \xi_k}\frac{\partial \xi_k}{\partial y_j}\frac{\partial y_j}{\partial \xi_j}\, y_i \quad = \quad -\left(\sum_k \frac{\partial E}{\partial \xi_k}\frac{\partial}{\partial y_j}\sum_j w_{jk}\, y_j\right)\frac{\partial y_j}{\partial \xi_j}\, y_i \quad = \\
&= \quad -\left(\sum_k \frac{\partial E}{\partial \xi_k}\, w_{jk}\right)\frac{\partial y_j}{\partial \xi_j}\, y_i \quad = \quad \left(\sum_k \delta_k\, w_{jk}\right)\lambda\, y_j\,(1 - y_j)\, y_i \quad = \\
&= \quad \delta_j y_i
\end{aligned}$$

Výraz pro aktualizaci vah ve vrstevnatých neuronových sítích tedy bude mít tvar:

$$w_{ij}(t+1) \quad = \quad w_{ij}(t) \quad + \quad \alpha\, \Delta_E w_{ij} \quad = \quad w_{ij}(t) \quad + \quad \alpha\, \delta_j\, y_i$$

V tomto vztahu odpovídá člen δ_j:

$$\delta_j \quad = \quad \begin{cases} (\,d_j \,-\, y_j\,)\, y_j\,(\,1 \,-\, y_j\,)\,\lambda & \text{pro výstupní neuron} \\ \lambda\, y_j\,(\,1 \,-\, y_j\,)\,\sum_k \delta_k\, w_{jk} & \text{pro skrytý neuron} \end{cases}$$

w představuje jednotlivé váhy, resp. prahy sítě, i a i' označuje neurony propojené s neuronem j vahou w_{ij}, resp. $w_{i'j}$. k indexuje neurony ve vrstvě nad neuronem j. d_j je požadovaný a y_j skutečný výstup neuronu j. $\xi_j = \sum_{i'} w_{i'j} y_{i'}$ pak označuje potenciál tohoto neuronu. $t+1$ a t představují následující a aktuální cyklus učení. α je konstanta reprezentující parametr učení. Z praktických důvodů je vhodné volit hodnotu α co možná největší (na druhou stranu ovšem i s ohledem na případné oscilace v procesu učení). To obvykle umožňuje rychlejší učení, ačkoliv pro praktické účely je standardní algoritmus zpětného šíření přece jen poněkud pomalý.

Proces učení lze urychlit např. pomocí tzv. momentu. Tato metoda zpravidla vede i k omezení oscilací během učení a vyžaduje pouze drobnější modifikaci adaptačního pravidla, které nyní bere v úvahu i změny vah z předchozího cyklu:

$$w_{ij}(t+1) \quad = \quad w_{ij}(t) \quad + \quad \alpha\, \delta_j\, y_i \quad + \quad \alpha_m\, (\, w_{ij}(t) \, - \, w_{ij}(t-1)\,)$$

V této rovnici představuje w odpovídající váhy, resp. prahy. i a k indexují neurony spojené s neuronem j pomocí váhy w_{ij}, resp. w_{jk}. y_j je skutečný výstup neuronu j a δ_j odpovídá příslušnému chybovému členu. $t+1$, t a $t-1$ indexují následující, aktuální a předchozí váhy. α označuje parametr učení a α_m je konstanta mezi 0 a 1, která určuje velikost vlivu předchozí změny vah při jejich současné aktualizaci. Obvykle se tento člen označuje jako moment. Vliv momentu se projeví především v oblastech pomalé konvergence algoritmu s téměř konstantní hodnotou gradientu, kde vede ke zvětšování prováděných změn. Naproti tomu v oblastech s prudkými změnami hodnot gradientu (+/-) vede použití této modifikace standardního algoritmu zpětného šíření k omezení oscilací v procesu učení.

Standardní algoritmus zpětného šíření má mnoho předností. Jeho aproximační schopnosti a schopnost vytvořit (alespoň v některých případech) vhodnou interní reprezentaci znalostí patří k nejdůležitějším vlastnostem vrstevnatých neuronových sítí typu zpětného šíření a přispívají k jejich dobrým generalizačním schopnostem. Obvykle je však velice obtížné odhadnout význam jednotlivých skrytých neuronů. Mezi velké nedostatky této metody naopak patří mezi jinými výskyt tzv. lokálních minim a pomalá konvergence obzvlášť při řešení náročných úloh pomocí sítí s téměř optimální architekturou. Obecně přitom nelze konvergenci algoritmu zaručit. Kritéria pro hodnocení vlastností a funkce vrstevnatých neuronových sítí dále zahrnují robustnost sítě vzhledem k malým odchylkám předkládaných vstupních vzorů a možnost opětovného využití již natrénovaných sítí za změněných podmínek. Změněné podmínky tu často představuje modifikovaná trénovací množina a/nebo pozměněné požadavky na funkci sítě.

Algoritmus zpětného šíření

Algoritmus zpětného šíření je v podstatě iterativní gradientní metoda navržená k minimalizaci střední kvadratické odchylky mezi skutečným a požadovaným výstupem BP-sítě. Jeho použití tedy předpokládá hladkou nelineární přenosovou funkci - pro naše účely použijeme sigmoidální přenosovou funkci $f(\xi)$ se strmostí 1. Graf této funkce je znázorněn na obrázku 3:

$$f(\xi) = \frac{1}{1+e^{-\xi}} \quad \text{, kde} \quad \xi = \sum_k w_k x_k$$

Krok 1: Inicializace vah a prahů
Zvol hodnoty vah a prahů v síti jako malé náhodné hodnoty.

Krok 2: Předlož nový trénovací vzor
Předlož síti vstupní vzor ve tvaru $x_1, x_2, ..., x_n$ a specifikuj požadované výstupy $d_1, d_2, ..., d_m$.

Krok 3: Spočítej skutečné výstupy
Skutečné výstupy $y_1, y_2, ..., y_m$ se určí pomocí sigmoidální přenosové funkce a vzorců uvedených v obrázku 5. Dále spočítej hodnotu chybové funkce E a **Skonči**, jestliže je tato hodnota dostatečně malá anebo byl proveden dostatečný počet cyklů

Krok 4: Aktualizace vah a prahů
Při aktualizaci synaptických vah postupuj od výstupní vrstvy směrem ke vstupní. Váhy se adaptují podle

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

V tomto vztahu představuje $w_{ij}(t)$ váhu ze skrytého anebo vstupního neuronu i v čase t, y_i označuje skutečný výstup neuronu i, α je parametr učení a δ_j je chybový člen odpovídající neuronu j. Pokud leží neuron j ve výstupní vrstvě,

$$\delta_j = y_j(1-y_j)(d_j - y_j) \ ,$$

kde d_j je požadovaný výstup neuronu j a y_j je jeho skutečný výstup.
Jestliže leží neuron j ve skryté vrstvě,

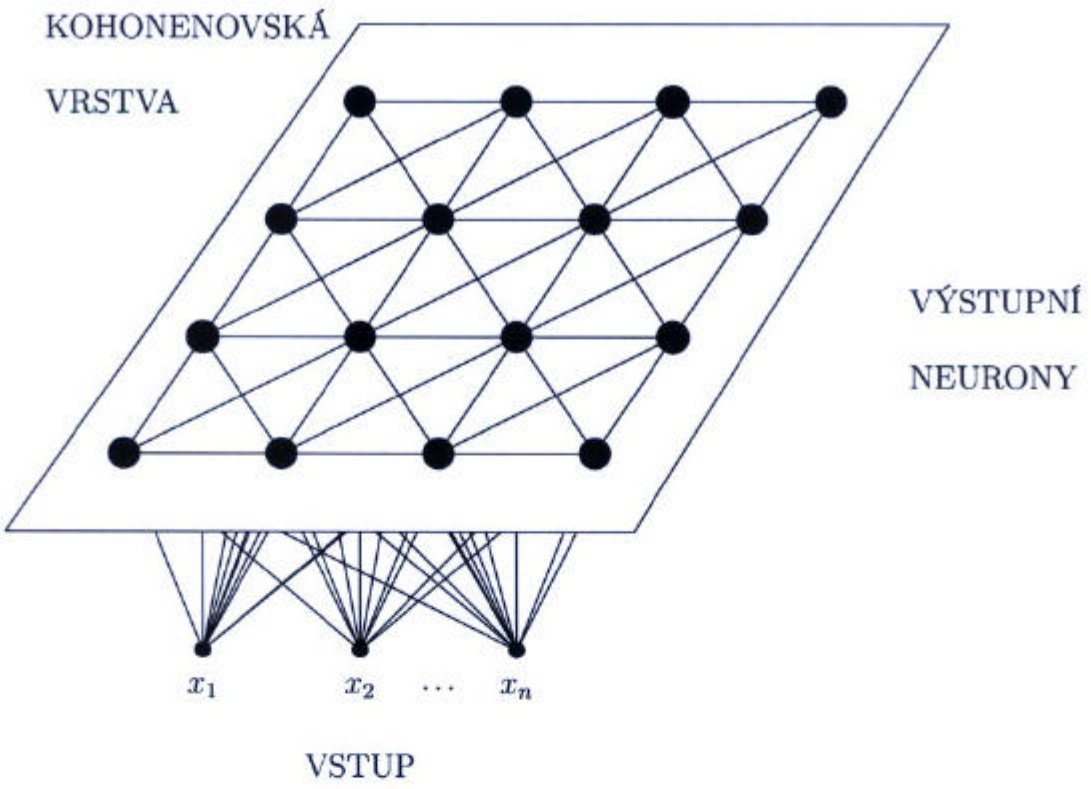
$$\delta_j = y_j(1-y_j) \sum_k \delta_k w_{jk} \ ,$$

kde k je index pro neurony z vrstvy nad neuronem j. Prahy neuronů se adaptují podobným způsobem, vezmeme-li v úvahu, že se jedná v podstatě o váhy s konstantními vstupy. Konvergence bývá ovšem často rychlejší, použijeme-li modifikované adaptační pravidlo s momentem:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha \delta_j y_i + \alpha_m (w_{ij}(t) - w_{ij}(t-1)) \ ,$$

přičemž $0 < \alpha_m < 1$.

Krok 5: Přejdi ke Kroku 2 a opakuj cyklus



Obrázek 6: **Kohonenova mapa** s výstupními neurony uspořádanými do dvourozměrné mřížky. Každý vstupní neuron je přitom propojen se všemi výstupními neurony.

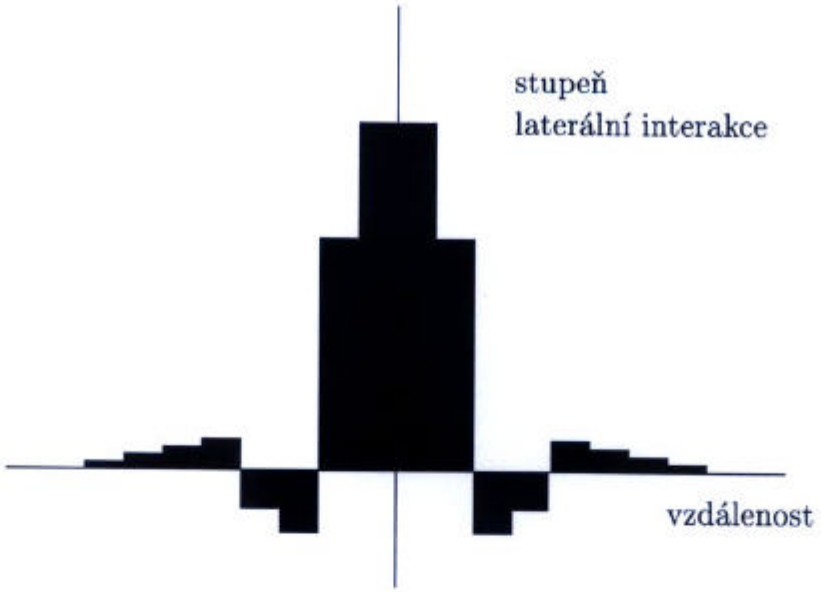
5 Samoorganizace a Kohonenovy mapy

Při myšlení i podvědomém zpracování informací je patrná obecná tendence komprimovat zpracovávané informace vytvářením redukovaných reprezentací nejrelevantnějších faktů pokud možno beze ztráty znalosti jejich vzájemných vztahů. Existují zobrazení - např. Voronoiova teselace, která jsou schopna zachovat topologické vztahy vstupních signálů a přitom provést redukci dimenze příznakového prostoru. Voronoiova teselace rozděluje příznakový prostor do oblastí, v nichž mají všechny vzory $\vec{x}$ stejný obraz $\vec{m}_i$ a zároveň je jejich vzdálenost k $\vec{m}_i$ menší než jejich vzdálenost k libovolnému jinému $\vec{m}_j; j \neq i$. Tyto oblasti jsou ohraničeny nadrovinami, které jsou kolmé na spojnice navzájem sousedících vektorů $\vec{m}_i$ a $\vec{m}_j$. Problémem je však rozložení vektorů $\vec{m}_i$ v procesu učení bez učitele - tzv. samoorganizaci, má-li rozložení vstupních vzorů v příznakovém prostoru $p(\vec{x})$ obecný tvar.

T. Kohonen navrhl strategii pro vytváření tzv. samoorganizujících se příznakových map podobných těm, které fungují v mozku. Samoorganizující se příznaková mapa rozprostře neurony ve vstupním prostoru tak, že jsou soustředěny v oblastech s nejvyšší hustotou vstupních vzorů. Přitom zachovávají prostorové uspořádání mezi vstupními vzory.

Kohonenův algoritmus adaptuje váhy z obecných vstupních neuronů do výstupních neuronů uspořádaných např. ve dvourozměrné mřížce. Vstupní vrstva je plně propojena s výstupní, tzv. Kohonenovskou vrstvou. Neurony v Kohonenovské vrstvě jsou propojeny s ostatními neurony téže vrstvy v "okolích". Definice sousedů každého neuronu tak určuje topologii Kohonenovy vrstvy. Jsou-li neurony uspořádané obecně do n-rozměrné mřížky

Cílem přitom je zachovat



Obrázek 7: **Funkce laterální interakce** tvaru "mexického klobouku".

a je-li neuron s maximální odezvou na daný vstupní vzor jeho obrazem, označujeme toto zobrazení za uspořádané, jestliže jsou topologické vztahy obrazů a vzorů podobné. V procesu učení jsou síti postupně předkládány jednotlivé vstupní vzory, a to bez specifikace požadovaného výstupu - jedná se tedy o učení bez učitele. Poté, co bylo síti předloženo dostatečné množství vstupních vzorů, by se měly váhy uspořádány tak, že topologicky blízké neurony budou citlivé na "podobné" vstupní vzory.

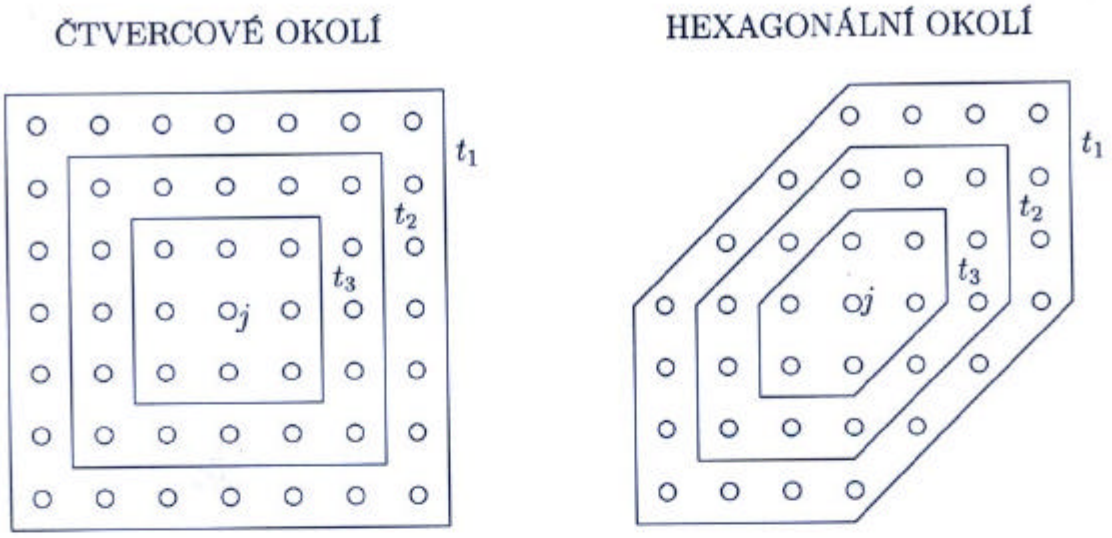
Při rozpoznávání dostává každý neuron vstupní vrstvy odpovídající vstupní hodnotu $x_i; 1 \leq i \leq n$ a $x_1, x_2, ..., x_n \in R$ (tzn., že je přenosová funkce zanedbána). Neuron j v Kohonenově vrstvě z těchto vstupních hodnot spočítá svůj potenciál: $\xi_j = \sum_{i=1}^{n} w_{ij} x_i$; kde $w_{ij} \in R$ jsou váhy synapsí jdoucích ze vstupního neuronu i k výstupnímu neuronu j. Cílem procesu učení je přitom stav, kdy synaptické váhy zkonvergovaly k takovým hodnotám, že je každý neuron citlivý na signály z určité oblasti vstupního prostoru. Maximální odezva neuronu pak odpovídá největší podobnosti mezi vektory $\vec{x} = [x_1, ..., x_n]^T$ a $\vec{w_i} = [w_{i1}, ..., w_{in}]^T$. Kritériem podobnosti může být např. i Euklidovská vzdálenost mezi těmito dvěma vektory. V takovém případě lze neuron c vykazující největší podobnost určit pomocí:

$$|| \vec{x} - \vec{w_c} || \quad = \quad \min_j \; || \vec{x} - \vec{w_j} || \; .$$

Neuron s maximální odezvou tak reprezentuje střed oblasti vykazující vysokou míru vzájemné podobnosti - tzv. "okolí". Mezi neurony v takovém "okolí" pak mohou existovat následující typy laterální interakce (tzn. stupně vzájemného ovlivnění):

- oblast laterální excitace s malým rozsahem
- excitační oblast je obklopena oblastí inhibiční akce
- oblast inhibiční akce obklopuje oblast menší excitační akce

Stupeň laterální interakce se obvykle popisuje pomocí funkce tvaru "mexického klobouku".



Obrázek 8: **Příklady topologického okolí** ($t_1 < t_2 < t_3$). Okolí je na počátku velké a v čase se postupně zmenšuje.

Obecně je δ-okolím prvku p z množiny X množina všech takových prvků množiny X, které mají od prvku p vzdálenost menší než δ. V případě Kohonenových map je vhodné definovat topologické okolí neuronu c jako funkci indexu neuronu v diskrétním čase t, $N_c = N_c(t)$. Dobrých výsledků lze dosáhnout při velkém okolí na začátku adaptace, které se postupně zmenšuje. Adaptační pravidlo pro Kohonenovy mapy lze přitom formulovat ve tvaru:

$$\Delta w_{ik}(t) \quad = \quad \alpha(t)\,(\,x_i(t) \,-\, w_{ik}(t)\,) \qquad \text{pro} \quad k \in N_c$$

$$\Delta w_{ik}(t) \quad = \quad 0 \qquad\qquad\qquad\qquad\qquad\qquad \text{pro} \quad k \notin N_c \ ,$$

kde $\alpha(t)$ označuje hodnotu parametru učení v kroku t a posloupost $\{\alpha(t); t = 0, 1, ...; 0 < \alpha(t) < 1\}$ je obvykle funkce pomalu klesající v čase. Koeficienty $\alpha(t)$, které se někdy označují i jako koeficienty bdělosti, by navíc měly splňovat následující podmínky:

$$\sum_{s=0}^{\infty} \alpha(s) \quad = \quad \infty \qquad\qquad \text{a} \qquad\qquad \sum_{s=0}^{\infty} \alpha(s)^2 \quad < \quad \infty \ ;$$

Proces adaptace se automaticky zastaví při $\alpha(t) = 0$. Obecně pak mají váhové vektory $\vec{w}_i$ tendenci uspořádat se podle své vzájemné podobnosti a aproximovat tak hustotu rozložení vzorů v příznakovém prostoru.

V procesu samoorganizace přitom probíhají dvě protikladné tendence. Množina vah má jednak tendenci popsat hustotu vstupních vzorů v příznakovém prostoru. Na druhé straně má však laterální interakce mezi jednotlivými neurony tendenci zachovat kontinuitu v posloupnosti váhových vektorů. Výsledkem je, že rozdělení váhových vektorů bude aproximovat tvar podobný ploše nad příznakovým prostorem, ale bude také hledat optimální orientaci a tvar v příznakovém prostoru, který nejlépe odpovídá struktuře vstupních vektorů. Rozdělení váhových vektorů pak tedy může pomoci detekovat takové příznaky, ve kterých mají vstupní vzory vysoký rozptyl a které by tudíž měly být v mapě popsány.

Tento základní model lze dále modifikovat - např. lze modelovat efekty "únavy" pomocí postupného zeslabování vah spojů při vytrvalé aktivitě příslušného neuronu. Shluky neuronů se mohou po nějaké době také rozpadnout a síť může akceptovat nový typ excitace. Jednotlivé modifikace tohoto základního modelu by pak měly brát v úvahu i různé typy mezních efektů. Z aplikačního hlediska jsou Kohonenovy samoorganizující se příznakové mapy vhodné zejména pro předzpracování dat. Při analýze velkého množství dat pak mohou pomoci při výběru signifikantních příznaků zpracovávaných vstupních vzorů.

Adaptační algoritmus pro Kohonenovy mapy

Krok 1: Inicializace vah sítě
Zvol hodnoty vah mezi n vstupními a m výstupními neurony jako malé náhodné hodnoty. Zvol počáteční poloměr okolí.

Krok 2: Předlož nový trénovací vzor
Předlož síti vstupní vzor ve tvaru $x_1, x_2, ..., x_n$.

Krok 3: Spočítej vzdálenost mezi vstupním vzorem a váhovým vektorem
Spočítej vzdálenost e_j mezi vstupním vzorem a váhovým vektorem pro každý výstupní neuron j pomocí:

$$e_j = \left(\sum_{i=1}^{n} (x_i(t) - w_{ij}(t))^2 \right)^{1/2} ,$$

kde $x_i(t)$ je vstupem neuronu i v čase t a $w_{ij}(t)$ je váhou synapse ze vstupního neuronu i k výstupnímu neuronu j v čase t. Tuto vzdálenost lze případně i upravit pomocí váhových koeficientů a předat kompetiční (Kohonenově) vrstvě.

Krok 4: Vyber "vítězný neuron"
Vyber (např. pomocí laterální inhibice) takový výstupní neuron c, který má minimální vzdálenost e_j od předloženého vstupního vzoru, a označ ho jako "vítěze":

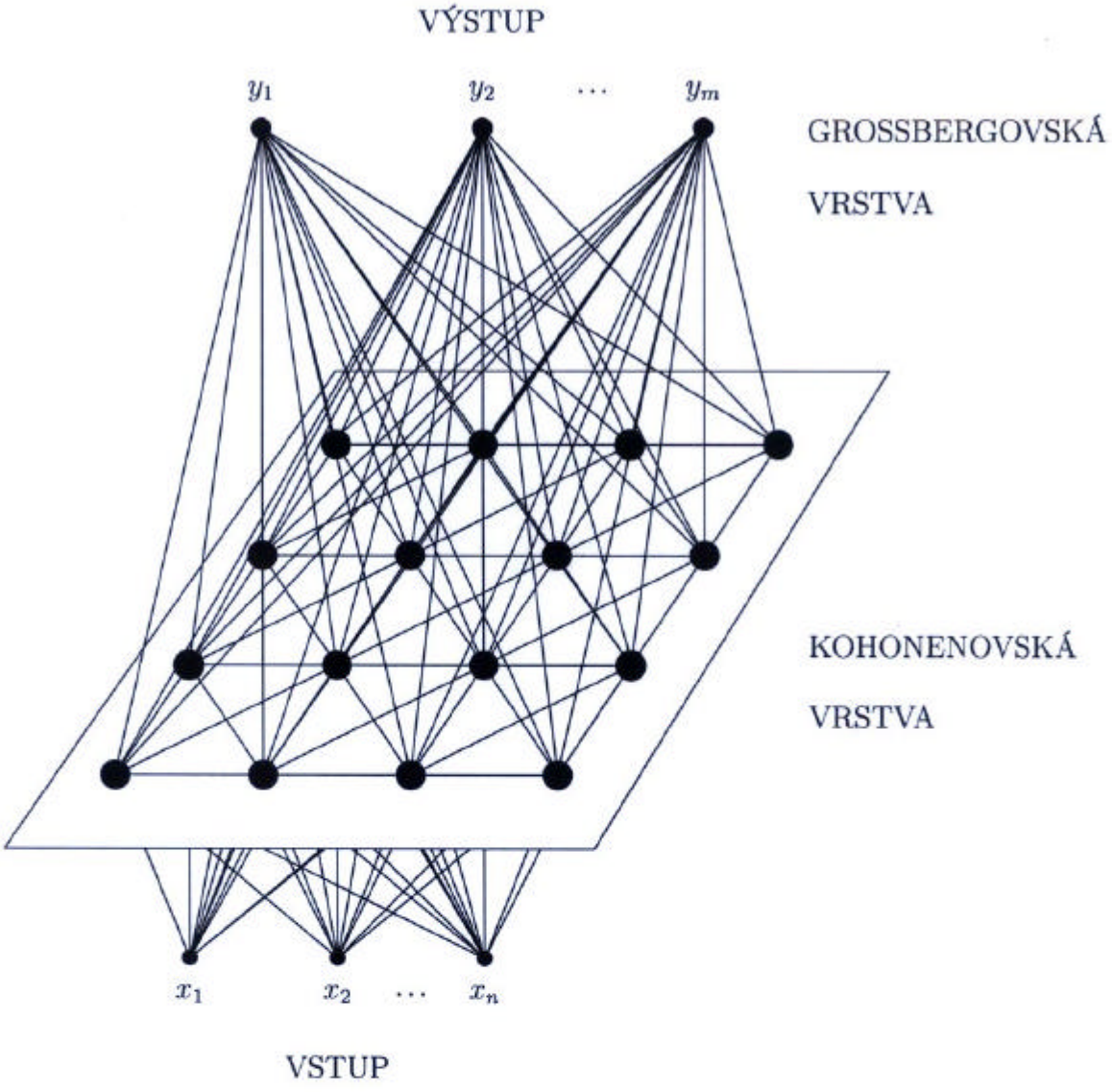
$$e_c = \min_j e_j = \min_j \left(\sum_{i=1}^{n} (x_i(t) - w_{ij}(t))^2 \right)^{1/2} .$$

Krok 5: Aktualizace vah
Váhy se aktualizují pro neuron c a všechny neurony k v jeho okolí, $k \in N_c$. Nové váhy se určí pomocí:

$$w_{ik}(t+1) = w_{ik}(t) + \alpha(t) (x_i(t) - w_{ik}(t)) .$$

Člen $\alpha(t)$ je koeficientem bdělosti (vigilance) $(0 < \alpha(t) < 1)$, který klesá v čase. Při procesu učení tak vítězný neuron upraví svůj váhový vektor směrem k předloženému vstupnímu vzoru. Totéž platí pro neurony v okolí vítězného neuronu. Koeficient bdělosti přitom klesá zároveň s rostoucí vzdáleností neuronů od středu okolí N_c.

Krok 6: Přejdi ke Kroku 2
Pokud nebyl proveden dostatečný počet cyklů, přejdi ke Kroku 2 a opakuj cyklus. V opačném případě skonči.



Obrázek 9: **Model sítě se vstřícným šířením.**

6 Sítě se vstřícným šířením

Sítě typu vstřícného šíření mají v zásadě 3 vrstvy neuronů: vstupní vrstvu, Kohonenovskou vrstvu a vrstvu výstupní, která se označuje jako Grossbergovská. Vstupní vrstva je plně propojena s Kohonenovskou a ta je opět plně propojena s Grossbergovskou vrstvou. Zobrazení ze vstupní do Kohonenovské vrstvy je přitom stejné jako v případě Kohonenových samoorganizujících se příznakových map. Na rozdíl od Kohonenových map je však třeba specifikovat v procesu učení i požadovaný výstup sítě - jedná se tedy o učení s učitelem.

Umělé neuronové sítě se vstřícným šířením vytvářejí v procesu učení jakousi "tabulku", která napomáhá k realizaci požadovaného zobrazení vstupních vzorů na výstup. Tato "tabulka" je přitom vytvářena tak, aby "rozložení záznamů v této tabulce" aproximovalo rozložení předkládaných vzorů z trénovací množiny v příznakovém prostoru. Při vybavování

se pro každý neuron Kohonenovské vrstvy určí míra podobnosti vstupního a váhového vektoru. Jako měřítko podobnosti se obvykle používá Euklidovská vzdálenost mezi vstupním a váhovým vektorem. Neuron, jehož váhový vektor nejlépe odpovídá vstupnímu vektoru, se označí jako "vítěz". Výstupní aktivita tohoto neuronu bude rovna 1. Výstupní aktivita všech ostatních neuronů Kohonenovské vrstvy bude rovna 0.

Výstupní vektor sítě tvoří v podstatě hodnoty synaptických vah neuronů Grossbergovské vrstvy vycházející z "vítězného" neuronu Kohonenovské vrstvy. Následky kvantizace výstupu Kohonenovské vrstvy lze ovšem při vybavování poněkud zmírnit použitím interpolačního mechanismu. Uvažování většího počtu "vítězů" v Kohonenovské vrstvě totiž umožňuje lepší aproximaci daného zobrazení. V průběhu učení sítí se vstřícným šířením pak probíhají v zásadě dva různé adaptační procesy. Během prvního se aktualizují váhy synaptických spojů mezi neurony vstupní a Kohonenovské vrstvy. Cílem tohoto procesu je optimálně pokrýt vstupní prostor reprezentovaný trénovacími vzory. Adaptace těchto vah probíhá stejně jako při učení Kohonenových samoorganizujících se příznakových map. Poněkud rovnoměrnějšího rozprostření vah lze dosáhnout např. pomocí tzv. "mechanismu svědomí".

V následující fázi učení se aktualizují synaptické váhy mezi neurony Kohonenovské a Grossbergovské vrstvy tak, aby odezva na daný vstupní vektor co nejlépe odpovídala požadovanému výstupnímu vektoru. To ovšem vede k průměrování. Váhy vycházející z "vítězného" neuronu, který reprezentuje danou oblast vstupního prostoru, pak odpovídají průměrným hodnotám příslušných složek požadovaných výstupních vektorů všech uvažovaných vzorů. Je tedy zřejmé, že váhy mezi neurony Kohonenovské a Grossbergovské vrstvy se mohou správně nastavit teprve po ukončení adaptace synaptických vah mezi neurony vstupní a Kohonenovské vrstvy.

Mezi výhody sítí se vstřícným šířením patří bezesporu jejich schopnost klasifikovat podle principu nejbližšího souseda libovolnou množinu vzorů. Klasifikace sama navíc probíhá podle určité "tabulky", kterou si je síť sama schopna "naprogramovat", a představuje tak dobrý statistický model vstupního prostoru. Mezi nedostatky tohoto modelu umělých neuronových sítí naproti tomu patří nezbytnost začlenění jednoho neuronu v Kohonenovské vrstvě pro každý výstupní vektor z Grossbergovské vrstvy v případě, že uvažujeme jediného "vítěze", a špatná srozumitelnost i kontrola v případě, že budeme uvažovat větší počet "vítězů". Sítě se vstřícným šířením lze m.j. použít například ke kompresi dat, k rozpoznávání vzorů, k aproximaci funkcí anebo např. i pro lepší pochopení funkce skrytých neuronů v sítích typu zpětného šíření.

Algoritmus učení pro sítě se vstřícným šířením

Krok 1: Inicializace vah sítě
Zvol hodnoty synaptických vah sítě jako malé náhodné hodnoty.

Krok 2: Předlož nový trénovací vzor
Předlož síti vstupní vzor ve tvaru $x_1, x_2, ..., x_n$ a specifikuj požadované výstupy $d_1, d_2, ..., d_m$.

Krok 3: V Kohonenovské vrstvě vyber "vítězný neuron"
Vyber v Kohonenovské vrstvě neuron c, jehož synaptické váhy nejlépe odpovídají předloženému vzoru $\vec{x}(t)$. Pro tento neuron tedy bude platit, že Euklidovská vzdálenost e_k mezi příslušným váhovým vektorem $\vec{w}_k(t)$ a předloženým vzorem $\vec{x}(t)$ je minimální:

$$e_c = \min_k e_k = \min_k \left(\sum_i (x_i(t) - w_{ik}(t))^2 \right)^{1/2}$$

kde $x_i(t)$ je vstupem neuronu i v čase t a $w_{ik}(t)$ je váhou synapse jdoucí ze vstupního neuronu i k neuronu k v Kohonenovské vrstvě.

Krok 4: Aktualizace vah neuronů z Kohonenovské vrstvy
Aktualizuj váhy w_{ik} mezi vstupním neuronem i a neurony Kohonenovské vrstvy, které se nacházejí v okolí neuronu c, N_c, tak, aby lépe odpovídaly předloženému vzoru $\vec{x}(t)$:

$$w_{ik}(t+1) = w_{ik}(t) + \alpha(t) (x_i(t) - w_{ik}(t))$$

Koeficient $\alpha(t)$ (kde $0 < \alpha(t) < 1$) označuje parametr učení vah mezi vstupní a Kohonenovskou vrstvou, který klesá v čase a se vzdáleností od středu okolí N_c. t představuje současný a $t+1$ následující krok učení.

Krok 5: Aktualizace vah neuronů z Grossbergovské vrstvy
Aktualizuj váhy v_{cj} mezi "vítězným" neuronem c z Kohonenovské vrstvy a neurony Grossbergovské vrstvy tak, aby skutečný výstupní vektor sítě, $\vec{y}$, lépe odpovídal požadované odezně $\vec{d}$:

$$v_{cj}(t+1) = (1 - \beta) v_{cj}(t) + \gamma z_c d_j$$

v_{cj} je váha synaptického spoje mezi c-tým neuronem Kohoneovské vrstvy a j-tým neuronem Grossbergovské vrstvy v čase t, $v_{cj}(t+1)$ označuje hodnotu této synaptické váhy v čase $t+1$. Parametr β (kde $0 < \beta < 1$) je konstanta ovlivňující závislost nové hodnoty synaptické váhy, v_{cj} na její hodnotě v předchozím kroku učení. Kladná konstanta γ představuje parametr učení vah mezi Kohonenovskou a Grossbergovskou vrstvou, z_c označuje aktivitu "vítězného" neuronu Kohonenovské vrstvy.

Krok 6: Přejdi ke Kroku 2
Pokud nebyl proveden dostatečný počet cyklů, přejdi ke Kroku 2 a opakuj cyklus. V opačném případě skonči.