Obsah

1. Neuronové sítě - Perceptrony

1.1.Základní informace

Následující text je součástí učebních textů předmětu Umělá inteligence a je určen hlavně pro studenty Matematické biologie. Kapitola se zabývá nejrozšířenějším konceptem dopředných umělých neuronových sítí, obvykle ne zcela přesně nazývaných jako jedno a vícevrstvé perceptrony. Kapitola popisuje organizační, adaptační i aktivní dynamiku těchto sítí, rozebírá a ilustruje jejich klasifikační možnosti. Podrobněji se zabývá základním adaptačním algoritmem pro vícevrstvé dopředné sítě, algoritmem zpětného šíření chyby.

1.2.Výstupy z učení

Zvládnutí učebního textu umožní studentům:

- Seznámit se se základními pojmy dopředných neuronových sítí, principy adaptační, aktivní a organizační dynamiky sítě.
- Definovat organizační dynamiku jedno a vícevrstvých perceptronů (MLP).
- Zobecnit poznatky z předchozí kapitoly na vyjádření klasifikačních možností jednovrstvého perceptronu.
- Odvodit klasifikační možnosti vícevrstvých perceptronů ve 2D prostoru a demonstrovat jejich klasifikační možnosti pomocí realizace XOR vícevrstvým perceptronem.
- Pochopit důsledek Kolmogorovy věty pro MLP.
- Porozumět adaptační dynamice MLP a vysvětlit formální matematické vyjádření algoritmu backpropagation, popsat význam jeho jednotlivých parametrů.
- Poznat aplikační oblasti pro MLP

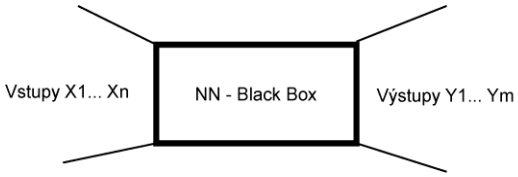
1.3.Dopředné neuronové sítě

1.1.1. Úvod

Umělá neuronová síť je matematickým modelem, který je složen z jednoduchých propojených elementárních výpočetních jednotek, neuronů.

Pro umělou neuronovou síť můžeme vysledovat jisté typické obecné rysy, zpravidla pro umělé neuronové sítě platí následující:

- Procesní elementy velmi jednoduché, pracuje jich však současně velký počet
- Činnost sítě je paralelní
- Jednotlivé procesní elementy pracují autonomně, lokálně a často asynchronně
- Neurony jsou vzájemně víceméně hustě propojeny orientovanými, číselnými vahami ohodnocenými spoji
- Váhy se upravují řízeným učením sítě a jejich nastavení představuje interní paměť znalostí sítě
    - Jsou velmi robustním řešením při částečném poškození sítě či neúplnosti vstupních dat



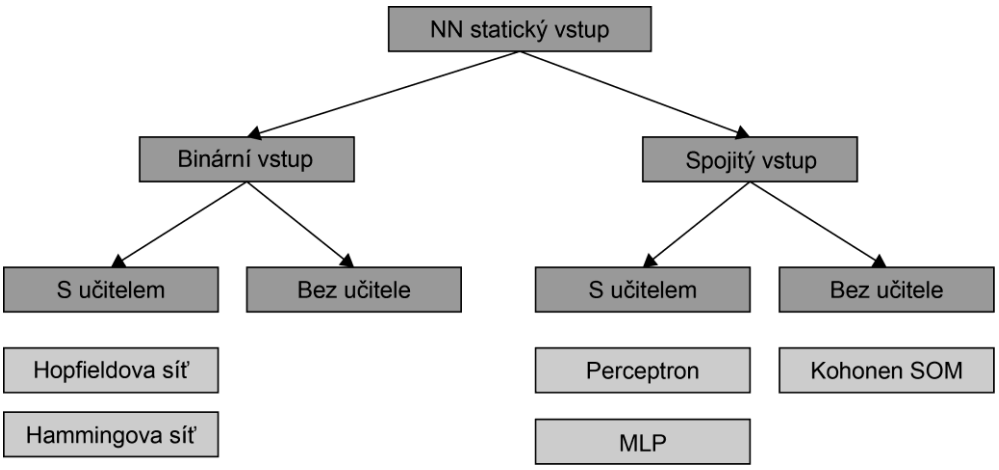
Obr. 1 *Obecné schéma neuronové sítě.*

Umělá neuronová síť realizuje zobrazení (transformaci) ze vstupního prostoru (vstupní vektory x) do prostoru výstupního (výstupní vektory y). Z vnějšího pohledu se tedy lze na umělou neuronovou síť dívat jako na „černou skříňku“, neboť nalezenou transformaci je až na elementární případy obtížné interpretovat, neexistuje formalizovaná analýza umělých neuronových sítí.

Umělé neuronové sítě můžeme dělit dle několika a kriterií:

- Dle druhu použitých výpočetních elementů (formálního neuronu)
- Dle topologie sítě, tedy uspořádání elementů (rekurentní, dopředné, vzájemné vazby)
- Dle trénovacího algoritmu
- Dle strukturovanosti
    - Nestrukturované – všechny neurony rovnocenné (Hopfield)
    - Strukturované – typicky obsahují uspořádání nestrukturovaných podsítí
        - Hierarchické
        - Soutěživé

Níže je uvedeno jedno z možných členění umělých neuronových sítí dle Lippmanna, včetně uvedení typických reprezentantů dané třídy.



Obr. 2 *Možné členění neuronových sítí dle Lippmanna.*

1.1.2. Dynamika neuronové sítě

Podobně jako v případě jednotlivého neuronu můžeme rozlišit tři fáze ustavení a provozu umělé neuronové sítě, tři dynamiky.

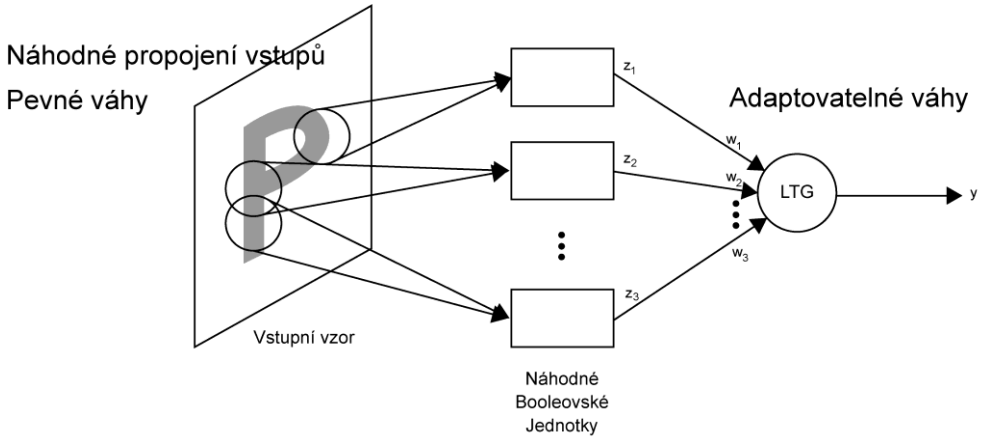
Organizační dynamika specifikuje topologii sítě, ustavování sítě a případnou změnu architektury. Většinou bývá konstantní, ale nemusí. Můžeme rozlišit dvě základní organizační dynamiky cyklickou a acyklickou. Acyklickou lze vždy rozdělit do jednotlivých vrstev

V aktivní dynamice se neurony v sítě nacházejí v definovaném stabilním stavu, dojde k nastavení vstupních uzlů sítě, průběh vlastního výpočtu. Obecně je možné uvažovat spojitou funkci sítě, prakticky jde však o taktovaný systém s diskrétním časem, kdy v jednotlivých diskrétních krocích dochází k výpočtu odezvy neuronů. Stav výstupních neuronů představuje výstup neuronový sítě, výsledek výpočtu. Po ustálení aktivní dynamiky sítě máme na daný vstup k dispozici ustálený výstup daný stavem výstupních neuronů představující funkci, zobrazení sítě

Adaptační dynamika představuje proces učení neuronové sítě, kdy dochází k nastavování vah jednotlivých neuronů sítě. Podobně jako výše můžeme uvažovat spojitou funkci, prakticky je nicméně tento proces opět rozdělen do jednotlivých diskrétních adaptačních kroků. Výsledkem procesu jsou nastavené váhy neuronové sítě, které vstupují do aktivní dynamiky sítě.

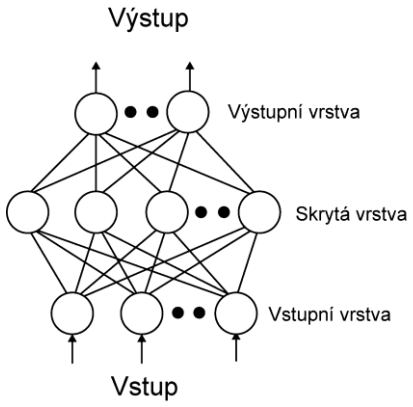
1.1.3. Historický Rosenblattův perceptron

Představuje klasický model neuronové sítě, která byla inspirována lidským okem. Modelovala percepci, odtud tedy pochází pojem perceptron. Jejím úkolem bylo pomocí optických snímačů uspořádaných do pole 20x20 elementů rozpoznávat jednotlivé zaznamenané znaky.



Obr. 3 *Rosenblattův perceptron.*

Pojem perceptron se umělé inteligenci ujal a nyní je bez ohledu na původní význam používán pro všechny dopředné neuronové sítě, tedy sítě s vrstevnatým uspořádáním neuronů a jednosměrným šířením signálu od vstupu na výstup. Neurony v jedné vrstvě jsou nezávislé a mohou pracovat paralelně, neurony další vrstvy jsou aktivovány po skončení výpočtu všech neuronů vrstvy předešlé.



Obr. 4 *Obecná struktura vícevrstvé dopředné sítě.*

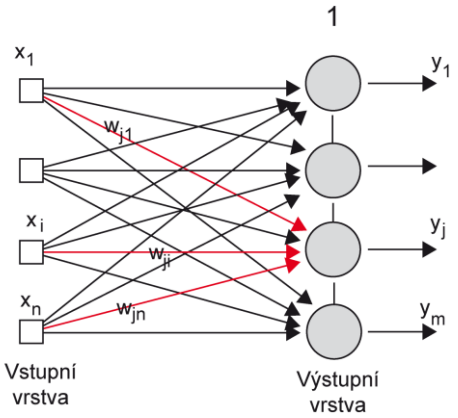
Dopředná neuronová síť je tedy představována orientovaným grafem, kde jednotlivé uzly jsou spojeny ohodnocenými orientovanými (obecně oboustraně) hranami. Ohodnocení těchto hran (váhy), jsou parametrem zpracování signálu. Signál se v dopředných sítích šíří po jednotlivých vrstvách od vstupu směrem k výstupu. Rozeznáváme uzly vstupní, výstupní a skryté, hrany reprezentují tok signálu. Spojení mezi dvěma neurony (uzly) je orientováno, je tedy dáno, ze kterého neuronu směřují informace a do kterého. Vlastností spojení je váha, která určuje schopnost a intenzitu přenášení informace. Váha může být jak kladně, tak záporně ohodnocena, neurony se tedy mohou navzájem povzbuzovat nebo potlačovat.

1.4.Jednovrstvý perceptron

1.1.4. Organizační, aktivní a adaptační dynamika

Jednovrstvý perceptron představuje koncepčně nejjednodušší vrstevnatou síť. Jedná se o M nezávislých, paralelně pracujících neuronů. Každý z těchto neuronů tedy realizuje transformaci vstupního vektoru na výstupní hodnotu nezávisle na neuronech ostatních.

Z pohledu organizační dynamiky se síť skládá z N neuronů vstupní vrstvy a vrstvy M výstupních neuronů. Vstupní vrstva není tvořena neurony ve smyslu zavedené definice, jedná se pouze o uzly realizující identickou kopii hodnot vstupního vektoru na vstupy všech neuronů. Obě vrstvy jsou vzájemně úplně propojeny, kdy každý j-tý výstupní neuron je propojen se všemi neurony vstupními.



Obr. 5 *Jednovrstvý perceptron.*

V průběhu aktivní dynamiky síť realizuje v obecném případě nezvažujícím charakteristiku výstupních funkcí (funkce identity) zobrazení z R^n -> R^m, které bylo nastaveno v průběhu dynamiky adaptační. Konkrétní obor výstupních hodnot je dán aktivačními přenosovými funkcemi výstupních neuronů, tedy například v případě sigmoidálních aktivačních funkcí aproximujících ostrou nelinearitu jde o realizaci zobrazení z R^n -> $(0,1)^m$.

Pokud zavedeme pro značení vah neuronů následující pravidlo

$W_{j,i} = W_{\text{kam, odkud}} = {}_{\text{kam}}W_{\text{odkud}} = {}_{\text{j-tého neuronu}}W_{\text{i-tá váha}}$

(1)

, můžeme pro odezvu j-tého neuronu při aktivní dynamice psát

$$y_j = \sigma(\xi) = \sigma(\sum_{i=0}^{n} w_{ji} x_i)$$

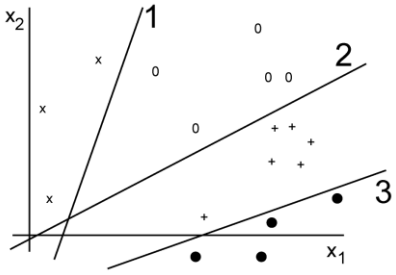
(2)

Z nezávislosti neuronů ve výstupní vrstvě vyplývá, že každý z nich se chová v aktivní i adaptační dynamice zcela nezávisle na neuronech ostatních. Aktivní i adaptační dynamika jednovrstvého perceptronu tak zcela odpovídá jednotlivému neuronu.

1.1.5. Klasifikační možnosti

Uvažujme nyní pro jednoduchost jednovrstvý perceptron realizující zobrazení z R^2 -> $\{0,1\}^m$, tedy klasifikaci prvků dvojdimenzionálního prostoru do dvou tříd. Připomeňme, že jednotlivý neuron s binárním výstupem dokáže rozlišit v prostoru dvě lineárně separovatelné třídy, kde dělící linii představuje přímka s normálovým vektorem odpovídajícím vektoru vah neuronu.

V případě jednovrstvého perceptronu takových dělících linií můžeme prostorem vést m, jednu pro každý neuron výstupní vrstvy. Samotné zmnožení neuronů ve výstupní vrstvě však nepřináší oproti jednotlivému neuronu žádný posun v klasifikačních možnostech perceptronu, protože neuronové síti chybí možnost výstupy jednotlivých neuronů dále kombinovat a umožnit tak klasifikaci do více tříd.



	x	0	+	●
y_1	1	-1	-1	-1
y_2	1	1	-1	-1
y_3	1	1	1	-1

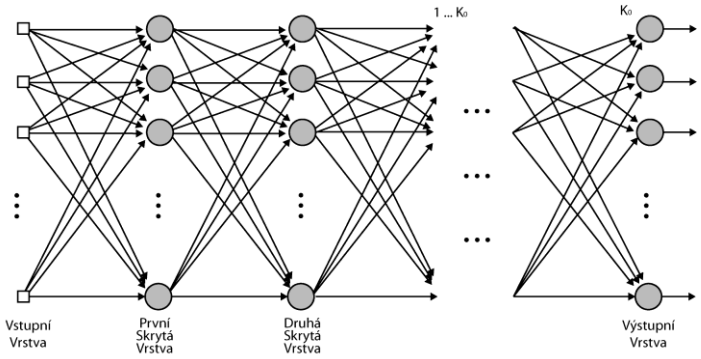
Obr. 6 *Rozdělení Euklidovského prostoru jednovrstvým perceptronem se třemi neurony*

Východisko z této situace představuje přidání další vrstvy neuronů a zavedení konceptu vícevrstvých perceptronů.

1.5.Vícevrstvý perceptron

1.1.6. Organizační a aktivní dynamika

Obecná organizační dynamika vícevrstvého perceptronu je uvedena na následujícím obrázku.



Obr. 7 *Organizační dynamika vícevrstvého perceptronu.*

Oproti perceptronu jednovrstvému byla topologie sítě rozšířena mimo vrstvu vstupní a výstupní o minimálně jednu, tzv. skrytou vrstvu. Platí, že jednotlivé neurony sousedních vrstev mezi sebou mají úplné propojení, neurony v rámci jedné vrstvy propojeny nejsou. Počet neuronů ve skrytých vrstvách může být různý, volí se dle charakteru řešené úlohy obvykle v rozmezí mezi počtem vstupních a výstupních neuronů.

Vstupní i výstupní vrstva je definována shodně, jako v případě jednovrstvého perceptronu, proto v režimu aktivní dynamiky realizuje síť v závislosti na charakteru výstupních aktivačních funkcí neuronů výstupní vrstvy opět zobrazení z R^n -> R^m. Na výpočtu výstupních hodnot se ale podílejí i neurony skrytých vrstev.

Aktivní dynamika vícevrstvého perceptronu probíhá v jednotlivých taktovaných diskrétních časových krocích (k), kdy jsou vždy paralelně a lokálně počítány výstupy j-tých neuronů ve vrstvě (k) opět dle vztahu

$$^{k}y_j = \sigma(\xi) = \sigma(\sum_{i=0}^{n} {}^{k}w_{ji}x_i)$$

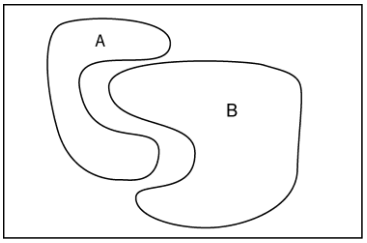
(3)

Pouze s tím rozdílem, že n zde má význam nejen rozměru vstupního vektoru (počtu neuronů vstupní vrstvy), jako v případě jednovrstvého perceptronu, ale obecněji pro k-tou vrstvu vyjadřuje počet neuronů v předcházející, k-1 vrstvě.

1.1.7. Ilustrace klasifikačních možnosti vícevrstvého perceptronu

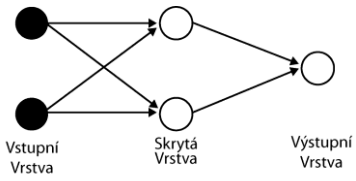
Předpokládejme pro jednoduchost opět klasický euklidovský dvojrozměrný vstupní prostor, ve kterém chceme klasifikovat jednotlivé body do dvou tříd. Vstupní vrstva bude obsahovat stejný počet neuronů, který je roven rozměru vstupního vektoru, tedy dva neurony. Výstupní vrstva pak bude v případě klasifikace do dvou tříd tvořena jediným neuronem s výstupní funkcí ve tvaru ostré nelinearity, tedy poskytující výstupy {0,1}. Konkrétní počet neuronů ve skrytých vrstvách nebude pro provedenou ilustraci podstatný.

Vstupní prostor a klasifikované množiny jsou znázorněny schematicky na následujícím obrázku.



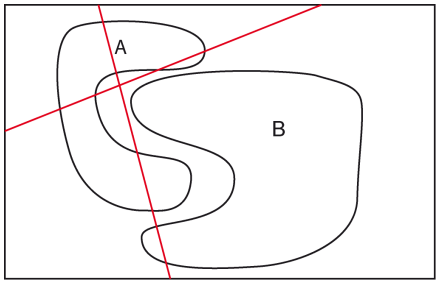
Obr.8 *Schematické znázornění klasifikovaných množin.*

Již jsme konstatovali, že jednovrstvý perceptron povede prostorem linie, ale nemá schopnost jednotlivé podporostory kombinovat. Tu přináší až další vrstva ve dvouvrstvém perceptronu.



Obr. 9 *Skrytá vrstva ve vícevrstvém perceptronu.*

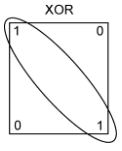
Dvouvrstvý perceptron se dvěma neurony skryté vrstvy by tedy mohl vstupní prostor rozdělit na 4 oblasti například takto:



Obr. 10 *Ilustrace Rozdělení prostoru vícevrstvým perceptronem.*

Klasifikace do dvou tříd A a B by zde nebyla ještě zcela dokonalá, neboť v oblasti vlevo dole se objevují vzory převážně ze třídy A, ale současně i některé ze třídy B. Představa aktivní dynamiky dvouvrstvého perceptronu je taková, že první vrstva rozdělí prostor na poloroviny a druhá vrstva provede jejich logickou konjunkci. Umožňuje tak vytvoření libovolné konvexní oblasti. Samozřejmě síť se může naučit zcela jinak, zde je jen uvedena ilustrace, že toto lze.

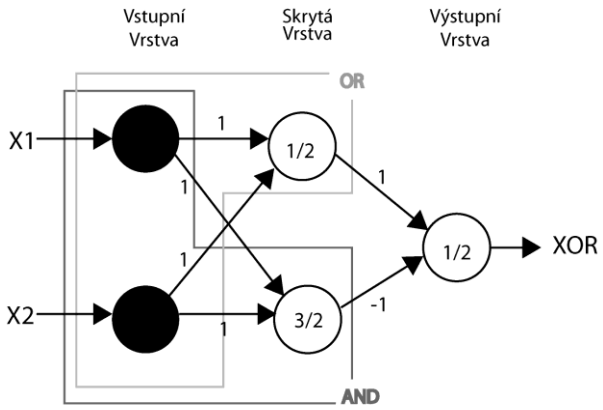
Vraťme se nyní k logické funkci XOR, kde nemožnost jejího vyčíslení jednotlivým neuronem a neexistence adaptačního algoritmu pro vícevrstvé sítě vedla ke stagnaci jejich rozvoje. Bude dvouvrstvý perceptron schopen logickou funkci XOR realizovat?



x_1	x_2	x_1 **XOR** x_2
0	0	0
0	1	1
1	0	1
1	1	0

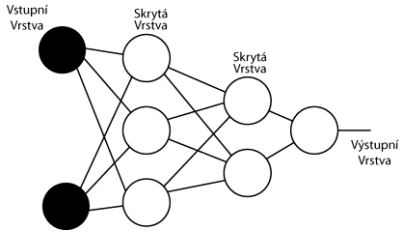
Obr. 11 *Logická funkce XOR.*

Funkci XOR můžeme vyjádřit pomocí elementárních logických funkcí jako XOR (X1,X2) = (X1 OR X2) NOT (X1 AND X2). Realizace neuronovou sítí může vypadat například jako na následujícím obrázku.



Obr. 12 *Možná realizace logické funkce XOR vícevrstvým perceptronem.*

Stručně ještě uvažujme model třívrstvého perceptronu se třemi neurony v první skryté vrstvě a dvěma neurony v další skryté vrstvě.



Obr. 13 *Třívrstvý perceptron.*

Představa aktivní dynamiky je taková, že první vrstva rozdělí prostor na poloroviny a druhá vrstva rozdělí prostor na konvexní podoblasti a třetí provede sjednocení těchto oblastí. Samozřejmě síť se může naučit zcela jinak, zde je jen uvedena ilustrace, že toto lze. Vícevrstvý perceptron s dvěma skrytými vrstvami a binárními neurony je tedy univerzálním aproximátorem jakékoli boolovské funkce N proměnných.

1.1.8. Klasifikační možnosti vícevrstvého perceptronu - Kolmogorova věta

Při vyjádření klasifikačních schopností a výpočetní síly vícevrstvých perceptronů můžeme vyjít z Kolmogorovy věty, která říká:

„Každou vícerozměrnou reálnou spojitou funkci N-proměnných lze přesně vyjádřit jako lineární kombinaci konečného počtu spojitých nelineárních funkcí jedné proměnné.“

$$f(x_1,\ldots,x_n)=\sum_{q=1}^{2n+1}\psi_q\left(\sum_{p=1}^{n}\phi_{pq}(x_p)\right),$$

(4)

Pokud budeme uvedenou větu vnímat v oblasti realizace umělých neuronových sítí a budeme hledat analogii, zjistíme, že vícevrstvý perceptron poskytuje pro realizaci uvedené věty dostatek prostředků. Jednotlivé nelineární funkce požadované Kolmogorovou větou mohou být realizovány pomocí nelineárních, například sigmoidálních funkcí jednotlivých neuronů. Lineární kombinace a tedy sumační vztahy Kolmogorovy věty pak odpovídají výpočtu aktivační funkce jednotlivých neuronů skrytých vrstev.

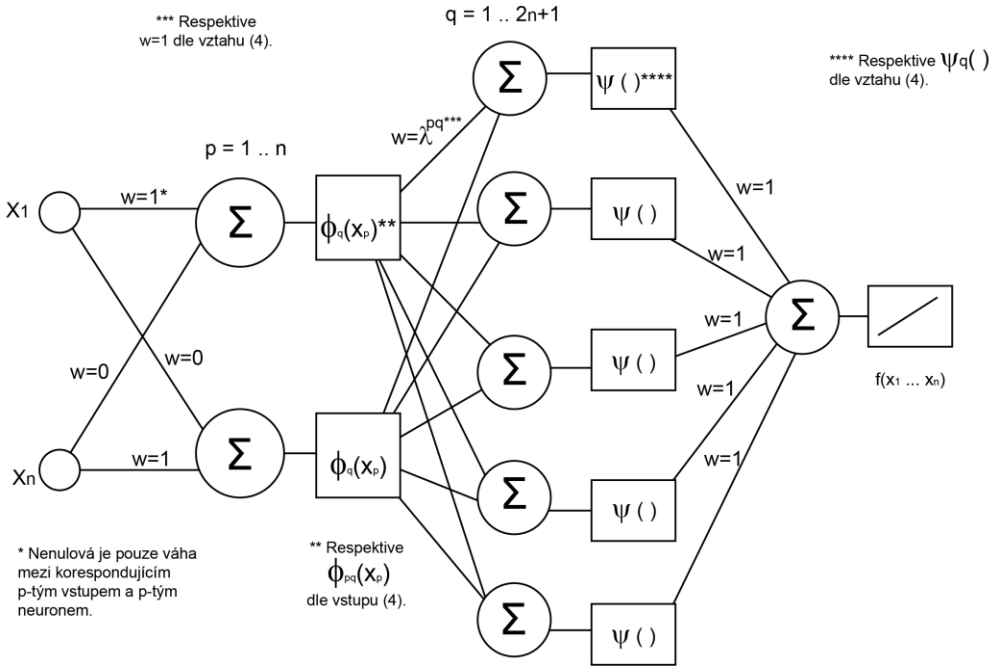
Uvedený vztah tedy vlastně odpovídá výpočtu, který realizuje vícevrstvý perceptron s n vstupními neurony, n neurony první skryté vrstvy s funkcemi Φ_{pq} (4), 2n+1 neurony druhé skryté vrstvy s funkcemi Ψ_q a jedním neuronem výstupním. Podmínkou je použití neuronů se spojitými, monotónně rostoucími aktivačními funkcemi ve skrytých vrstvách.

Kolmogorovu větu lze formulovat ještě obecněji, dle[1].

$$f(x_1 \dots x_n) = \sum_{q=1}^{2n+1} \Psi\Big(\sum_{p=1}^{n} \lambda^{pq} \Phi_q(x_p)\Big)$$

(5)

, kde došlo ke zmírnění požadavků na obecně různé aktivační přenosové funkce neuronů v jednotlivých vrstvách.



Obr. 14 *Kolmogorova věta pro vícevrstvý perceptron.*

Poněkud problematická může být dle [1] „volba funkcí Ψ a Φ, neboť pro přesnou aproximaci funkce f(**x**) mohou mít tyto hledané funkce vlastnosti, které neumožňují jejich použití jako přechodové výstupní funkce neuronu, protože funkce Ψ nejsou obecně hladké, mohou mít fraktální grafy a funkce Φ je konstruována v závislosti na funkci f(**x**), není tedy reprezentovatelná v parametrické podobě a není ji možné použít jako přechodovou výstupní funkci neuronů.“

I přes relativizaci platnosti výše uvedeného závěru Kolmogorovy věty pro vícevrstvý perceptron (nepoužitelnost obecných funkcí jako přenosových funkcí neuronu), bylo dokázáno [3], že zmnožením počtu neuronů ve skrytých vrstvách je možné dosáhnut univerzální aproximační schopnosti vícevrstvého perceptronu, pokud se spokojíme pouze s libovolně přesnou aproximací funkce f(**x**), nikoli jejím naprosto přesným vyjádřením.

Důsledek Kolmogorovy věty pro vícevrstvý perceptron můžeme shrnout tak, že vícevrstvý perceptron se dvěma skrytými vrstvami, dostatečným počtem neuronů se spojitými monotónně rostoucími aktivačními funkcemi může být univerzálním aproximátorem libovolné funkce N proměnných. To představuje značnou výpočetní sílu. Dokonce tak velkou, že pro řadu reálných problémů při jejich přibližném řešení vystačíme s klasifikací do lineárně reparabilních množin.

V případě perceptronu dvouvrstvého, tedy perceptronu s jednou skrytou vrstvou, pak můžeme vyjít z výsledků Hornika a Leshna, publikovaných v [1].

Z jejich závěrů plyne, že vícevrstvý perceptron s jednou skrytou vrstvou, dostatečným počtem neuronů (zdá se N*(2N+1)) a nepolynomiálními (sigmoidálními) aktivačními funkcemi neuronů skryté vrstvy je univerzálním aproximátorem libovolné funkce N proměnných.

1.1.9. Adaptační dynamika

Připomeňme, že adaptační dynamika jednotlivého neuronu i jednovrstvého perceptronu je založena na učení s učitelem na základě trénovací a testovací množiny postupně předkládaných vektorů. Předpokládejme trénovací množinu o p prvcích v podobě

$$T = \{[\mathbf{x}_1,\mathbf{y}_{d1}], [\mathbf{x}_2,\mathbf{y}_{d2}] \ldots [\mathbf{x}_p,\mathbf{y}_{dp}]\}$$

(6)

Algoritmus adaptační dynamiky spočívá v postupném předkládání trénovacích vzorů sítě a iterativní úpravě vah na základě odchylky zjištěné na výstupu sítě. Tato odchylka je tedy pro p-tý vzor trénovací množiny určena chybovým vektorem e_p.

$$\mathbf{e}_p = \mathbf{y}_{dp} - \mathbf{y}_p$$

(7)

Jeden krok učení (krok t) může být představován následující posloupností úkonů:

1. Předložení vstupu $\boldsymbol{x_p}$ síti.
2. Zjištění odezvy sítě $\boldsymbol{y_p}$
3. Výpočet odchylky $\boldsymbol{e}_p$ pro poslední vrstvu sítě.
4. Rozpočítání odchylky na neurony předchozích vrstev.
5. Lokální úprava vah neuronů sítě na základě rozpočítané odchylky pro každý neuron.

Tyto kroky se opakují, dokud nejsou splněny podmínky pro zastavení adaptace.

Zatímco v případě jednotlivého neuronu a jednovrstvého perceptronu je určení odchylky a následná úprava vah snadná, u vícevrstvých perceptronů je situace složitější. Adaptační algoritmus je totiž v případě vícevrstvého perceptronu schopen přímo určit jen odchylku na výstupní vrstvě sítě a tedy

dosud zavedenými algoritmy by bylo možné adaptovat pouze váhy poslední, výstupní vrstvy. Základní otázkou tedy je, jakým způsobem určit chybu neuronů ve skrytých vrstvách vícevrstvého perceptronu. Tuto situaci řeší tzv. algoritmus zpětného šíření chyby, kdy jeho odvození vedlo po déle trvajícím útlumu k bouřlivému rozvoji zájmu o neuronové sítě.

1.1.10. Algoritmus zpětného šíření chyby (BP algoritmus)

Algoritmus zpětného šíření chyby je základním algoritmem pro učení vícevrstvých perceptronů. V následujících odstavcích si jej odvodíme.

Učením sítě můžeme rozumět optimalizační proces, kdy síť nastavuje své parametry tak, aby minimalizovala chybovou funkci vznikající jako rozdíl mezi skutečným a požadovaným výstupem. Pro jeden vstupní vektor platí, že síť realizuje zobrazení

$$\bar{y}_p = \varphi(\bar{x}_p)$$

(8)

Chybový vektor získaný po předložení p-tého vektoru trénovací množiny lze zapsat jako

$$\bar{e}_p = \bar{y}_{dp} - \bar{y}_p$$

(9)

Okamžitá kvadratická odchylka klasifikace sítě pro p-tý vektor trénovací množiny je

$$\varepsilon_p = \sum_{j=1}^{N} ({}_j e_p)^2$$

, kde N značí počet výstupních vektorů, respektive rozměr výstupního vektoru.

(10)

Uvedená kvadratická odchylka je však pouze odchylkou vypočtenou na základě předložení jediného vstupu. Optimální by samozřejmě bylo, kdybychom byli schopni vyjádřit střední kvadratickou odchylku přes všechny možné vstupy sítě. Tyto vstupy však nemáme k dispozici, k dispozici máme pouze trénovací množinu obsahující *p* trénovacích dvojic. Pak můžeme střední kvadratickou odchylku přes všechny vstupy obsažené v trénovací množině vyjádřit jako

$$\varepsilon = \mathrm{E}\{\varepsilon_p\} \; pro \; \forall \varepsilon_p$$

(11)

Minimalizací této funkce gradientní metodou nejstrmějšího sestupu bychom dosáhli nejlepší možné klasifikace sítě ve smyslu střední kvadratické odchylky přes celou trénovací množinu. Střední kvadratická odchylka je evidentně funkcí vah ***w*** sítě. Jejich změnou je tedy realizován postup k minimalizaci střední kvadratické odchylky. Pro změnu vah ***w*** neuronu můžeme psát

$$\overline{w}(t+1) = \overline{w}(t) - \mu \nabla \varepsilon$$

, kde *t* označuje krok učení a μ je konstanta určující rychlost adaptace.

(12)

Gradient $\nabla\varepsilon$ můžeme rozepsat jako

$$\nabla\varepsilon = \frac{\partial\varepsilon}{\partial w}(t)$$

(13)

Výpočet tohoto gradientu je však v praxi bohužel velmi obtížný i pro nepříliš rozsáhlé sítě. Proto jej nahrazuje výpočtem posloupnosti dílčích gradientů, kdy každý dílčí gradient získaný v jednom kroku učení sítě aproximuje hodnotu gradientu přes celou trénovací množinu.

$$\frac{\partial\varepsilon}{\partial w}(t) \approx \frac{\partial\varepsilon_p}{\partial w}(t)$$

(14)

Uvědomme si nyní, že výstupní vrstva perceptronu se skládá se samostatných neuronů, kde každý z nich přispívá k celkové chybě perceptronu. Uvažujme nyní jeden samostatný neuron výstupní vrstvy.

Hodnotu gradientu pro výstupní, k_0-tou vrstvu sítě můžeme pro *i*-tou váhu w_i *j*- tého neuronu této vrstvy vyjádřit dle pravidla o složené derivaci

$$\frac{\partial\varepsilon_p}{\partial {}^{k_0}_{j}w_i} = \frac{\partial\varepsilon_p}{\partial\sigma({}^{k_0}_{j}\xi_p)} \frac{\partial\sigma({}^{k_0}_{j}\xi_p)}{\partial {}^{k_0}_{j}\xi_p} \frac{\partial {}^{k_0}_{j}\xi_p}{\partial {}^{k_0}_{j}w_i}$$

(15)

Pokud vzorec rozebereme po jednotlivých částech, pak třetí část vzorce představuje derivaci vnitřního potenciálu j-tého neuronu podle i-té váhy. Vnitřní potenciál j-tého neuronu může být vyjádřen sumou vstupních hodnot násobených vahami w. Vstupní hodnoty neuronu jsou ale v tomto případě představovány výstupy y_p z předchozí k_0-1 té vrstvy, pro výsledek derivace tedy můžeme psát

$$\frac{\partial {}^{k_0}_{j}\xi_p}{\partial {}^{k_0}_{j}w_i} = \frac{\partial \sum_{\forall i} {}^{k_0-1}_{i}y_p \, {}^{k_0}_{j}w_i}{\partial {}^{k_0}_{j}w_i} = {}^{k_0-1}_{i}y_p$$

(16)

Střední část vzorce představuje derivaci aktivační výstupní funkce podle vnitřního potenciálu neuronu. Konkrétní aktivační přenosovou funkci nemáme v tuto chvíli definovanou, ponechme tedy vztah bez úprav, jen s přepsáním derivace

$$\frac{\partial\sigma({}^{k_0}_{j}\xi_p)}{\partial {}^{k_0}_{j}\xi_p} = \sigma'({}^{k_0}_{j}\xi_p)$$

(17)

Úvodní část vzorce (16) vyjadřující derivaci chyby podle výstupu neuronu y můžeme po zvážení výrazů (9,10) psát také jako

$$\frac{\partial \varepsilon_p}{\partial \sigma({}^{k_0}_{j}\xi_p)} = \frac{\partial ({}_{j}e_p)^2}{\partial \sigma({}^{k_0}_{j}\xi_p)} = \frac{\partial ({}_{j}y_{dp} - {}_{j}y_p)^2}{\partial {}_{j}y_p} = -2({}_{j}y_{dp} - \sigma({}^{k_0}_{j}\xi_p)) = -2({}^{k_0}_{j}e_p)$$

(18)

Po úpravě tedy dostáváme vztah

$$\frac{\partial \varepsilon_p}{\partial {}^{k_0}_{j}w_i} = \frac{\partial \varepsilon_p}{\partial \sigma({}^{k_0}_{j}\xi_p)} \frac{\partial \sigma({}^{k_0}_{j}\xi_p)}{\partial {}^{k_0}_{j}\xi_p} \frac{\partial {}^{k_0}_{j}\xi_p}{\partial {}^{k_0}_{j}w_i} = -2({}_{j}y_{dp} - \sigma({}^{k_0}_{j}\xi_p))\sigma'({}^{k_0}_{j}\xi_p)^{k_0-1}_{i}y_p = -2({}^{k_0}_{j}e_p)\sigma'({}^{k_0}_{j}\xi_p)^{k_0-1}_{i}y_p$$

(19)

Dosazením do vztahu (12) a vektorovým vyjádřením dostáváme pro úpravu vah jednoho neuronu poslední vrstvy

$${}^{k_0}_{j}\overline{w}(t+1) = {}^{k_0}_{j}\overline{w}(t) + 2\mu({}^{k_0}_{j}e_p)\sigma'({}^{k_0}_{j}\xi_p)^{k_0-1}\overline{y}_p$$

(20)

Ve vztahu zbývá určit hodnotu chyby e_p a případně derivaci σ'. Chyba e_p je pro neurony výstupní vrstvy zřejmá již ze vtahu (9) a je rovna

$${}^{k}_{j}e_p = {}_{j}y_{dp} - {}^{k}_{j}y_p$$

(21)

Derivace σ' samozřejmě závisí na konkrétní aktivační výstupní funkci neuron. Oblíbené jsou sigmoidální aktivační funkce, které jsou hladké a monotónní a navíc mají jednoduché vyjádření derivace.

$$y = \frac{1}{1+e^{-\alpha}}$$

a

$$y' = \frac{1}{(1+e^{-\alpha})^2} e^{-\alpha} = \frac{1}{1+e^{-\alpha}} \frac{1+e^{-\alpha}-1}{1+e^{-\alpha}} = \frac{1}{1+e^{-\alpha}} (1 - \frac{1}{1+e^{-\alpha}}) = y(1-y)$$

(22,23)

Volba monotónních funkcí je vhodnější vzhledem k omezení vzniku lokálních minim chybové funkce. Jako aktivační přenosové funkce ale mohou být ale použity libovolné diferencovatelné funkce.

V tuto chvíli je tedy schopni určit chybu a provést adaptaci neuronů výstupní vrstvy formálně odvozeným vztahem.

Podobně, jako jsme odvodili vzorce pro vrstvu výstupní, můžeme postupovat i v případě neuronů skrytých k-tých vrstev, tedy

$$\frac{\partial \varepsilon_p}{\partial {}^{k}_{j}w_i} = \frac{\partial \varepsilon_p}{\partial \sigma({}^{k}_{j}\xi_p)} \frac{\partial \sigma({}^{k}_{j}\xi_p)}{\partial {}^{k}_{j}\xi_p} \frac{\partial {}^{k}_{j}\xi_p}{\partial {}^{k}_{j}w_i} = \frac{\partial \varepsilon_p}{\partial \sigma({}^{k}_{j}\xi_p)} \sigma'({}^{k}_{j}\xi_p)^{k-1}_{i}y_p$$

(24)

, kde

$$\frac{\partial \varepsilon_p}{\partial \sigma({}^k_j \xi_p)} = \frac{\partial \varepsilon_p}{\partial {}^k_j y_p} = ({}^k_j e_p) = \sum_{i=1}^{M_{(k+1)}} \frac{\partial \varepsilon_p}{\partial {}^{k+1}_i y_p} \frac{\partial {}^{k+1}_i y_p}{\partial {}^{k+1}_i \xi_p} \frac{\partial {}^{k+1}_i \xi_p}{\partial {}^k_j y_p} = -2 \sum_{i=1}^{M_{(k+1)}} {}^{k+1}_i e_p \sigma'({}^{k+1}_i \xi_p) {}^{k+1}_i w_j$$

(25)

Z uvedených vztahů (18,24) je zřejmé, že výpočet gradientu chybové funkce pro skryté neurony se od definice pro neurony výstupní vrstvy liší pouze částí definice pro výpočet chyby e_p na výstupu neuronu.

Úpravu vah pro libovolný neuron sítě můžeme tedy vyjádřit jako

$${}^k_j \overline{w}(t+1) = {}^k_j \overline{w}(t) - \mu({}^k_j e_p) \sigma'({}^k_j \xi_p) {}^{k-1} \overline{y}_p$$

(26)

, kde pro neurony výstupní vrstvy platí

$${}^k_j e_p = -2({}_j y_{dp} - \sigma({}^k_j \xi_p))$$

(27)

A pro neurony vnitřních skrytých vrstev platí

$${}^k_j e_p = -2 \sum_{i=1}^{M_{(k+1)}} \sigma'({}^{k+1}_i \xi_p) {}^{k+1}_i e_p \; {}^{k+1}_i w_j$$

(28)

Uvedené odvozené výrazy (26, 27, 28) se velmi často upravují na ne zcela přesné vyjádření ve formě

$${}^k_j \overline{w}(t+1) = {}^k_j \overline{w}(t) + 2\mu({}^k_j e_p) \sigma'({}^k_j \xi_p) {}^{k-1} \overline{y}_p$$

(29)

$${}^k_j e_p = {}_j y_{dp} - \sigma({}^k_j \xi_p)$$

(30)

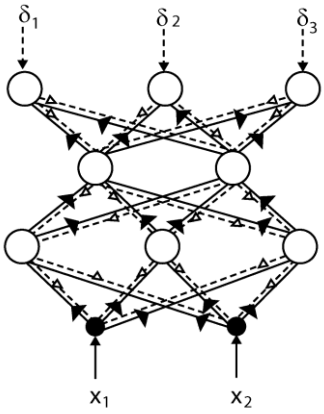
$${}^k_j e_p = \sum_{i=1}^{M_{(k+1)}} \sigma'({}^{k+1}_i \xi_p) {}^{k+1}_i e_p \; {}^{k+1}_i w_j$$

(31)

Souvislost s heuristicky odvozeným pravidlem δ-rule pro jednotlivý neuron je zde přímočará.

Pokud si všimneme vzorce (31) , je zřejmé, že chyba ${}_j^k e_p$ na výstupu j-tého neuronu k-té skryté vrstvy je získána jako součet chyb ${}_i^{k+1} e_p$ všech neuronů na výstupech následující, k+1 vrstvy, kde tato chyba ${}_i^{k+1} e_p$ je násobena derivací přenosové výstupní funkce σ' $({}_i^{k+1}\xi_p)$ a dále hodnotou váhy w synapse mezi neurony obou vrstev.

Princip algoritmu zpětného šíření chyby tedy spočívá v určení chyby na výstupech všech neuronů výstupní vrstvy, tu určíme lehce dle vztahu (30). Následně je tato chyba použita i k adaptaci skryté vrstvy a to tak, že se chyba z výstupní vrstvy se vlastně šíří strukturou sítě zpět od výstupní vrstvy na vrstvu předcházející, kde je chyba na výstupu modifikována derivací přenosové funkce neuronu výstupní vrstvy a následně ještě vahou, která k neuronu výstupní vrstvy vede z neuronu vrstvy předcházející. Každému neuronu skryté vrstvy je tedy dopravena chyba, která odpovídá jeho příspěvku k celkové chybě na výstupní vrstvě. Takto je chyba postupně šířena zpět strukturou sítě od výstupu na všechny vrstvy předcházející. Proto je algoritmus nazýván algoritmem zpětného šíření chyby.



Obr. 15 *Ilustrace algoritmu zpětného šíření chyby.*

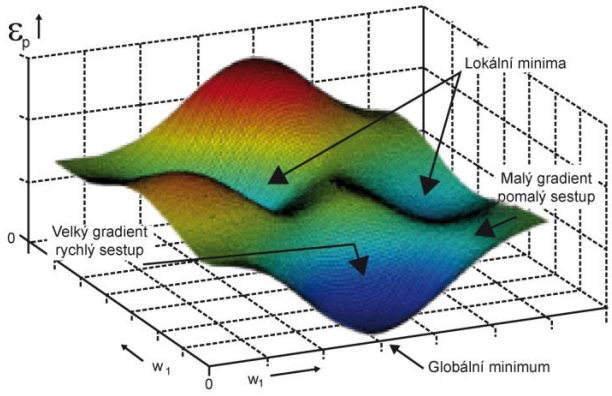
V bodech lze algoritmus zpětného šíření chyby pro vytvořený vícevrstvý perceptron shrnout jako:

1. Inicializuj váhy sítě
2. Vyber p-tý vzor z trénovací množiny
3. Předlož ténovací vzor $\bar{x}_p$ síti
4. Urči odezvu sítě $\bar{y}_p$ na $\bar{x}_p$
5. Urči chybu na výstupu sítě dle (30)
6. Urči chybu na předcházejících vrstvách pomocí zpětného šíření chyby na vrstvy předchozí
7. Lokálně uprav váhy dle (29)
8. Opakuj od bodu 2, dokud není výkonnost sítě dostatečná

1.1.11. Minimalizace chybové funkce adaptačním algoritmem

Adaptační algoritmus, například zde uvedený BP algoritmus, provádí minimalizaci chybové funkce, kde hledá gradientní metodou minimum na chybové hyperploše. Na každou váhu sítě můžeme v optimalizačním procesu pohlížet jako na jednu dimenzi n-dimenzionálního chybového prostoru. Pro

dvě váhy může mít tato hyperplocha podobu jako na obrázku, kde na vodorovných osách jsou hodnoty vah a na ose svislé pak velikost chybové funkce, která je funkcí prvků trénovací množiny. Úkolem adaptačního procesu je sejít na této hyperploše co nejníže, ideálně do globálního minima. Nicméně existuje zde i riziko uváznutí v minimech lokálních.



Obr. 16 *Minimalizace chybové funkce při adaptaci.*

Riziko uváznutí v lokálním minimu lze omezit volbou počátečních parametrů a trénovací množiny a samozřejmě i modifikací adaptačního algoritmu.

Počáteční inicializace vah neuronů

Počáteční inicializace vah definuje bod na chybové ploše, odkud při hledání minima algoritmus vychází. Pokud by byl výchozí bod počátečním nastavením vah definován na svážnici směřující k lokálnímu minimu, ze zřejmé, že globální minimum chybové funkce nalezeno nebude a minimalizace skončí v minimu lokálním. Správnou počáteční inicializaci bohužel nejsme prakticky schopni odhadnout, proto se pro počáteční inicializaci vah sítě používají náhodná čísla a v případě neúspěchu se inicializace opakuje.

Volba rychlostní konstanty μ

Rychlostní konstanta μ definuje velikost kroku, se kterým se algoritmus pohybuje po chybové ploše. Jeho výběr rovněž není jednoznačný, neboť příliš velký krok může vést k nekontrolovanému pohybu po ploše a v důsledku k nenalezení minima, malý krok naopak sice bezpečně směřuje k nejbližšímu minimu, ale konvergence algoritmu může být velmi pomalá. Kompromisní řešení představuje postupné snižování velikosti kroku během minimalizačního procesu.

Složitost chybové funkce

Tvar plochy chybové funkce závisí samozřejmě nejen na parametrech sítě (váhy, přenosové funkce) ale i na vstupních vektorech. Pokud je plocha příliš členitá s velkým množstvím lokálních minim je činnost minimalizačního algoritmu velmi ztížena. Proto jsou vstupy sítě často předzpracovávány. Cílem předzpracování je zmenšit vstupní dimenzi vektorů a zejména pak potlačit nepodstatné vlastnosti vstupních vektorů a zdůraznit vlastnosti podstatné. Eliminace nepodstatných vlastností vstupů omezuje výskyt lokálních minim v ploše chybové funkce a zjednodušuje její tvar. Bohužel o

podstatnosti či nepodstatnosti některých vlastností vstupů často nejsme schopni rozhodnout jinak než intuitivně.

Způsob předkládání vstupů trénovací množiny

Je výhodné předkládat vstupy z trénovací množiny síti náhodně. Pak je i pohyb po chybové ploše náhodný a v důsledku umožňuje prověření širší oblasti chybové plochy.

Modifikace algoritmu BP

Modifikace algoritmu BP má za úkol omezit riziko uváznutí v lokálním minimu. Modifikace algoritmu BP spočívá v zavedení setrvačnosti η do adaptačního procesu, kde zpočátku volíme η blízké 1 a jeho hodnotu v průběhu adaptace zmenšujeme k nule. V případě zvýšení chyby je doporučeno setrvačnost vypustit.

Vztah (12) zjednodušíme na

$$\overline{w}(t+1) = \overline{w}(t) + \Delta\overline{w}(t)$$

(32)

Zavedení setrvačnosti η do vztahu znamená

$$\overline{w}(t+1) = \overline{w}(t) + (1-\eta)\Delta\overline{w}(t) + \eta\Delta\overline{w}(t-1)$$

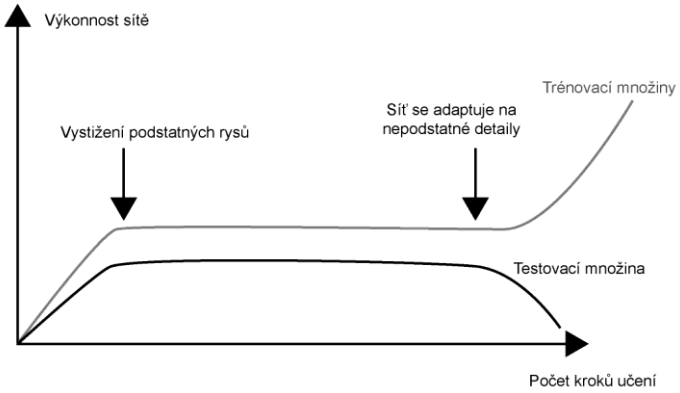
(33)

Pro $\eta = 0$ se pak algoritmus chová jako čistý BP algoritmus.

1.1.12. Vícevrstvý perceptron a syndrom přeučení

Vícevrstvý perceptron je univerzálním robustním aproximátorem, který představuje vzhledem k platnosti Kolmogorova teorému pro vícevrstvý perceptron velmi univerzální řešení. Praktické je zejména jeho použití v případech, kdy máme k dispozici jen příklady vstupů a výstupů nějakého procesu. Velká vyjadřovací síla vícevrstvého perceptronu je však vykoupena poměrně výpočetně náročnou optimalizací sítě pomocí BP algoritmu. Mezi nevýhody také patří, že nalezená transformace je ve struktuře sítě ukryta, nelze ji přímo interpretovat.

Nalezení globálního minima chybové funkce přes trénovací množinu není také vždy účelné. Může dojít k tzv. efektu přeučení sítě, kdy síť sice efektivně minimalizuje chybovou funkci přes trénovací množinu, nicméně na úkor obecnosti nalezené transformace pro vzory mimo trénovací množinu. Síť se adaptovala na nepodstatné detaily obsažené v trénovací množině. Taková přeučená síť pak není schopna zobecňovat, jak je vidět na poklesu výkonnosti sítě vzhledem k testovací množině.



Obr. 17 *Syndrom přeučení.*

Přeučení sítě lze bránit několika postupy. Je výhodné nesnažit se dosáhnout absolutního minima chybové funkce přes celou trénovací množinu, ale učení sítě zastavit o jistý okamžik dříve. Tento okamžik lze velmi často odhadnout jako náhlý pokles hodnoty chybové funkce nad trénovací množinou. Ještě efektivnějším postupem bránícím přeučení bývá zavedení nejen trénovací a testovací množiny, ale i množiny validační, která adaptaci včas zastaví.

Dále se při návrhu perceptronu snažíme volit pokud možno malý počet neuronů ve skrytých vrstvách, to povede síť ke hledání jednodušší a obecnější transformace. Aby se síť nesoustředila na vystižení nepodstatných detailů trénovací množiny, je možné předkládané vstupy vždy nepatrně zkreslit, například náhodným šumem.

1.6.Seznam použité literatury

[1] Šíma, J., Neruda, R.: Teoretické otázky neuronových sítí, MATFYZPRESS, 1996, ISBN 80-85863-18-9.

[2] Jan, J.: Číslicová filtrace, analýza a restaurace signálů. VUT v Brně, VUTIUM, 2002. ISBN 80-214-1558-4.

[3] Kůrková V.: Kolmogorov's theorem and multilayer neural network. Neural Networks, Volume 5, Issue 3, Pages 501-506, 1992