OBSAH

6 UMELÉ NEURÓNOVÉ SIETE 157

6.1 POTREBNÉ POJMY Z NEUROBIOLÓGIE 158
6.2 PERCEPTRÓN – MODEL NEURÓNU 160
6.3 VIACVRSTVOVÉ DOPREDNÉ SIETE A ICH UČENIE 162
6.4 REKURENTNÉ SIETE A ICH UČENIE 176
6.5 RBF SIETE A ICH UČENIE 186
6.6 SAMOORGANIZUJÚCA SA MAPA 187
6.7 HISTÓRIA UMELÝCH NEURÓNOVÝCH SIETÍ 194

6 UMELÉ NEURÓNOVÉ SIETE

POTREBNÉ POJMY Z NEUROBIOLÓGIE • PERCEPTRÓN • VIACVRSTVOVÉ DOPREDNÉ SIETE A ICH UČENIE • REKURENTNÉ SIETE A ICH UČENIE • RBF SIETE A ICH UČENIE • HISTÓRIA UMELÝCH NEURÓNOVÝCH SIETÍ

V tejto kapitole si predstavíme základné modely umelých neurónových sietí (angl. artificial neural networks), ako z teoretického, tak aj aplikačného hľadiska. Oboznámime sa so základnými pojmami týkajúcimi sa prvkov, architektúry a učenia neurónových sietí, ktoré boli pôvodne inšpirované neurobiológiou mozgu. Vysvetlíme si, ako umelé neurónové siete vykonávajú klasifikáciu vzorov, aproximáciu funkcií, ako vedia predpovedať budúci vývoj dát, či ako vedia hrať hry a navigovať robota.

HLAVNÁ MYŠLIENKA UMELÝCH NEURÓNOVÝCH SIETÍ

Ľudský mozog je vrcholom evolúcie na našej Zemi a zatiaľ najdokonalejším „nástrojom“ na spracovanie informácií. Tak ako sa klasická umelá inteligencia snaží napodobovať „softvér“, tak sa umelé neurónové siete snažia napodobovať „hardvér“, na ktorom sa tento „softvér“ v realite vykonáva. V priebehu tohoto napodobovania vznikli zaujímavé výpočtové systémy, ktoré vykonávajú podobné funkcie ako ich vzor, ale ním nie sú, tak isto ako v priebehu napodobovania lietania nebol stvorený vták, ale lietadlo.

EMERGENTNÉ SPRÁVANIE

Učenie sa z príkladov a paralelné spracovanie signálov mnohými prvkami vedú k takému makroskopickému správaniu neurónových sietí, ktoré nie je predpovedateľné na základe vlastností jednotlivých prvkov systému. Ide o tzv. **emergentné správanie** podľa latinského slova emergencia, ktoré vo všeobecnosti znamená vynorenie sa, objavenie sa. Kolektívne správanie umelých neurónových sietí sa vynára na základe modelovania princípov činnosti nervového systému. Na druhej strane emergentné vlastnosti viackonateľových systémov sú založené na modelovaní princípov evolúcie života a spoločnosti.

UMELÉ NEURÓNOVÉ SIETE V KONTEXTE UMELEJ

V tých prípadoch, keď nepoznáme pravidlá, podľa ktorých by sme modelovali riešenie danej situácie, alebo tieto pravidlá sú veľmi zložité, či neúplné, vtedy je jednou z možností použitie umelých neurónových sietí. Treba zdôrazniť, že je to

INTELIGENCIE

iba jedna z možností. Ďalšími alternatívami sú napríklad klasické štatistické metódy, viackonateľové systémy alebo iné adaptívne výpočtové systémy. Keď poznáme pravidlá, je vždy lepšie použiť prístupy klasickej umelej inteligencie. Umelé neurónové siete sú „dobré“ v rozpoznávaní a klasifikácii vzorov (obrazcov) do tried (angl. *pattern recognition and classification*).

Vzory treba chápať ako abstraktné entity, nemusia to byť iba vizuálne vzory. Vo všeobecnosti umelé neurónové siete vedia riešiť najrôznejšie asociačné úlohy. Aj rozpoznanie, či klasifikácia vzorov je vlastne asociácia, a to asociácia vzoru s jeho triedou. Ak vieme nejaký problém formulovať ako asociačnú úlohu, budeme môcť použiť umelé neurónové siete.

VIAC O OBSAHU KAPITOLY

V tejto kapitole sa budeme zaoberať takými modelmi umelých neurónových sietí, ktoré sa učia tak, že „učiteľ“ im v priebehu učenia hovorí, aké má byť správne riešenie problému. Dôležité sú však aj umelé neurónové siete, ktoré sa učia bez tejto informácie a majú vlastnosti samoorganizácie (Kvasnička et al., 1997; Oravec et al., 1998). Naša voľba je motivovaná tým, že prvý typ má v súčasnosti univerzálnejšie použitie ako druhý typ.

Základná terminológia umelých neurónových sietí je založená na neurobiologickej terminológii, preto našu kapitolu začíname vysvetlením potrebných základných neurobiologických pojmov. Historicky aj konceptuálne je dôležitá teória perceptrónov, na ktorú nadväzuje konštrukcia dopredných viacvrstvových neurónových sietí, ktoré sa používajú na aproximáciu zložitých nelineárnych funkcií a na asociačné úlohy. Pokračujeme predstavením rekurentných neurónových sietí, ktoré sa používajú na asociačné a predikčné úlohy s časovým kontextom. Na záver spomenieme RBF siete s novým typom stavebných prvkov, ktoré podstatne zrýchľujú učenie.

6.1 POTREBNÉ POJMY Z NEUROBIOLÓGIE

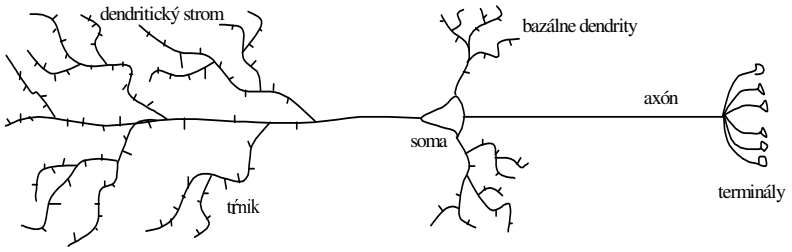
NEURÓN

Odhaduje sa, že v ľudskom mozgu sa nachádza rádovo 10^{11} nervových buniek (neurónov) (Maršala, 1985). Dve tretiny neurónov tvoria 4–6 mm hrubú mozgovú kôru, ktorá tvorí jeho silne zvrásnený povrch. Predpokladá sa, že mozgová kôra je sídlom poznávacích (kognitívnych) procesov. V neurónoch prebiehajú zložité biochemické deje, ktoré zabezpečujú to, že neuróny môžu spracúvať signály z iných neurónov a vysielať k nim svoje vlastné signály.

Signály môžeme reprezentovať ako reálne čísla vyjadrujúce intenzitu (veľkosť) prijímaných a vysielaných signálov, ktoré majú v skutočnosti elektrickú a chemickú povahu. Spoj medzi dvoma neurónmi (miesto prenosu signálu z jedného neurónu na druhý) sa nazýva **synapsa**. V synapse sa môže ten istý signál buď zosilniť alebo zoslabiť. Silu pôsobenia synapsy určuje **váha synapsy** (angl. *synaptic weight*). Jeden neurón môže mať na svojom povrchu rádovo 10^3 až 10^5 synáps. Vstupný povrch neurónu pozostáva z dendritov a tela (somy) neurónu (pozri obr. 6.1). Tisícky dendritov tvoria bohato rozvetvený strom, na ktorom sa

nachádza väčšina synáps, a to najmä na tŕnikoch.

OBR. 6.1. NEURÓN



EXCITÁCIA INHIBÍCIA

Signály od ostatných neurónov môžu mať buď kladné alebo záporné znamienko. V prvom prípade sú miestom prenosu tzv. **excitačné synapsy** a v druhom prípade tzv. **inhibičné synapsy**. Keď suma kladných a záporných príspevkov (signálov) od ostatných neurónov váhovaná váhami príslušných synáps prekročí istú hodnotu, nazývanú *prah excitácie* neurónu, neurón vygeneruje **výstupný impulz** (angl. *spike*). Zvyčajne neurón ako odpoveď na svoju stimuláciu vygeneruje celú sériu impulzov, ktoré majú nejakú priemernú frekvenciu, rádovo $10–10^2$ Hz. Frekvencia je úmerná celkovej stimulácii neurónu. Výstupné impulzy sa šíria k ostaným neurónom pozdĺž jediného výstupného výbežku neurónu, ktorý sa nazýva *axón*. Axón sa na svojom konci rozvetvuje na tisícky výbežkov. Zakončenia týchto axónových výbežkov tvoria synapsy na ostatných neurónoch v sieti.

UČENIE

V mozgovej kôre a vôbec v celom mozgu je to, ktoré neuróny a akým typom synáps budú komunikovať, úplne geneticky určené. Koľko synáps bude medzi danými neurónmi je tiež z podstatnej časti geneticky určené. Tento počet sa môže „dolaďovať“ v závislosti od konkrétnej skúsenosti jedinca, a to najmä v detstve.

V priebehu celého života sa v dôsledku individuálnej skúsenosti (učenia) menia váhy jednotlivých synáps (Kvasnička et al., 1997). V súčasnosti sa všeobecne akceptuje poznatok, že *učenie sprevádzajú zmeny váh synáps v mozgových neurónových sieťach*. Otázkou zostáva akým pravidlom, či pravidlami sa tieto zmeny riadia.

KÓDOVANIE A REPREZENTÁCIA

Prvým princípom kódovania a reprezentácie informácií v mozgu je **nadbytočnosť** (redundancia). Znamená to, že každá informácia (akokoľvek ju chápeme) sa prenáša, prijíma a spracúva nadbytočným počtom neurónov a synáps, aby sa v prípade poškodenia sietí nestratila úplne. Výsledkom je to, že keď sa poškodzujú neurónové siete, či už biologické alebo umelé, ich výkon upadá len veľmi pozvoľna (angl. *graceful degradation*).

Ďalej, súčasná predstava je taká, informácia (nie v shannonovskom zmysle, ale v zmysle obsahu, resp. významu) je zakódovaná v tom, ktoré neuróny s ktorými komunikujú. To je dané geneticky (evolučne) a v danom rámci sa dotvára učením. Každý objekt sa reprezentuje celou danou sieťou neurónov. V tejto sieti je dôležitá tak distribúcia aktívnych, ako aj neaktívnych neurónov. Nazýva sa to **distribuovaná reprezentácia**. Rozličné objekty sa reprezentujú rozličnými vzorcami resp. distribúciami aktivity v príslušných neurónových sieťach.

? V umelých neurónových sieťach sa vstupná i výstupná aktivita siete interpretuje programátorom. Čo plní túto úlohu v mozgu? Iné neurónové siete? Vedomie? Čo je to vedomie? To ostáva zatiaľ záhadou.

6.2 PERCEPTRÓN – MODEL NEURÓNU

VSTUPNÝ VEKTOR

VÁHOVÝ VEKTOR

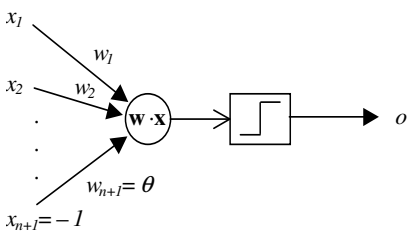
Perceptrón (percept = vnem) je model neurónu, ktorý prijíma vstupné signály $\bar{x} = (x_1, x_2, ..., x_{n+1})$ cez synaptické váhy tvoriace váhový vektor $\bar{w} = (w_1, w_2, ..., w_{n+1})$ (pozri obr. 6.2). Vstupný vektor $\bar{x}$ sa nazýva **vzor** alebo obrazec (angl. *pattern*). Zložky vstupného vektora môžu nadobúdať reálne alebo binárne hodnoty. Príkladom môže byť obrázok (písmeno, odtlačok prsta atď.) zakódovaný ako pole (vektor) hodnôt. Zložky váhového vektora sú reálne čísla.

Výstup perceptrónu *o* je daný vzťahom:

$$o = f(net) = f(\bar{w} \cdot \bar{x}) = f\left(\sum_{j=1}^{n+1} w_j x_j\right) = f\left(\sum_{j=1}^{n} w_j x_j - \theta\right) \tag{6.1}$$

kde premenná *net* označuje váhovanú sumu vstupov, t.j. skalárny (zložkový) súčin váhového a vstupného vektora. Funkcia *f* sa volá **aktivačná funkcia** perceptrónu.

OBR. 6.2. PERCEPTRÓN



V tejto notácii predpokladáme, že perceptrón má *n+1* vstupov. Hodnota (*n+1*)-vého vstupu je vždy –1 a $w_{n+1} = \theta$, čo je hodnota prahu excitácie perceptrónu (angl. *threshold*).

V roku 1958 Rosenblatt zaviedol diskrétny perceptrón s bipolárnou binárnou aktivačnou funkciou (funkcia signum, znamienko):

$$f(net) = sign(net) = \begin{cases} +1 & ak\ net \geq 0 \Leftrightarrow \sum_{j=1}^{n} w_j x_j \geq \theta \\ -1 & ak\ net < 0 \Leftrightarrow \sum_{j=1}^{n} w_j x_j < \theta \end{cases} \tag{6.2}$$

Rovnica

$$\sum_{j=1}^{n} w_j x_j - \theta = 0 \tag{6.3}$$

je rovnicou nadroviny v *n*-rozmernom priestore. Napr. v 2-rozmernom priestore

je to rovnica priamky: $w_1 x_1 + w_2 x_2 - \theta = 0$. Váhy perceptrónu predstavujú koeficienty deliacej priamky (nadroviny), ktorá rozdeľuje priestor vzorov na dva podpriestory. Perceptrón teda dokáže klasifikovať, t.j. zatrieďovať vzory, do dvoch tried, ktoré sú lineárne oddelené deliacou hranicou vyjadrenou rovnicou (6.3). Inými slovami, perceptrón dokáže riešiť iba tzv. **lineárne separovateľné problémy**.

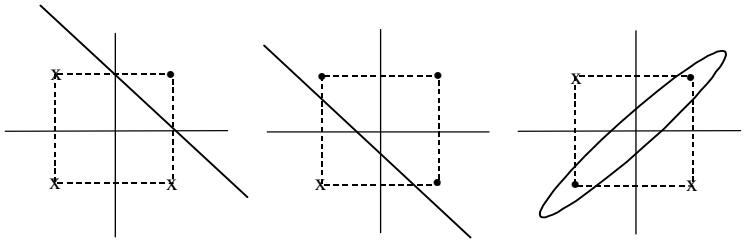
KLASIFIKÁCIA

LINEÁRNA SEPARÁCIA

Takýmito problémami sú napr. logické funkcie AND a OR. Funkcia XOR nie je lineárne separovateľná (obr. 6.3). Rosenblatt ukázal, že je možné natrénovať perceptrón pomocou vzorov tak, že jeho synaptické váhy budú zodpovedať koeficientom deliacej nadroviny, ak táto existuje (Rosenblatt, 1958).

OBR. 6.3.

AND, OR A XOR



UČENIE

Všeobecné pravidlo učenia pre umelé neurónové siete znie: Váhový vektor $\overline{w}$ sa zväčšuje priamo úmerne so súčinom vstupného vektora $\overline{x}$ a učiaceho signálu s. Učiaci signál s je funkciou $\overline{w}$, $\overline{x}$ a niekedy aj spätnej väzby od učiteľa d. Teda:

$$s = s(\overline{w}, \overline{x}, d) \quad alebo \quad s = s(\overline{w}, \overline{x}) \tag{6.4}$$

V prvom prípade ide o **učenie s učiteľom** (angl. *supervised learning*). V druhom prípade ide o **učenie bez učiteľa** (angl. *unsupervised learning*). Pri učení v umelých neurónových sieťach sa v diskrétnych časových krokoch mení j-ta váha takto:

$$w_j(t+1) = w_j(t) + \Delta w_j(t) = w_j(t) + \alpha \; s(t) \; x_j(t) \tag{6.5}$$

Konštanta $0 < \alpha \leq 1$ sa nazýva **rýchlosť učenia** (angl. *learning rate*).

δ–PRAVIDLO

Učiacim signálom pre binárny perceptrón je aritmetický rozdiel medzi požadovanou a skutočnou odpoveďou perceptrónu, t.j. $s = d - o = \delta$. Pravidlo učenia binárneho perceptrónu sa nazýva **pravidlo δ (delta)**. Pre j-tu váhu platí:

$$\Delta w_j = \alpha \, (d - o) \, x_j \tag{6.6}$$

Pre binárny bipolárny perceptrón je požadovaným výstupom pre jednu triedu $d = +1$, a pre druhú triedu $d = -1$. Pravidlo δ platí aj pre unipolárny binárny perceptrón, pre ktorý $d \in \{0, 1\}$ aj $o \in \{0, 1\}$. Keď sa pozrieme na pravidlo (6.6) vidíme, že keď chceme, aby daný vzor patril do triedy $d = 1$, ale momentálne dáva perceptrón výstup $o = -1$, potom $\delta = 2$ a j-ta váha sa zväčší, ak je j-ty vstup $x_j > 0$. Ak $x_j < 0$, j-ta váha sa zmenší. Takéto zmeny váh vedú k tomu, aby sa po k krokoch daný vzor klasifikoval do triedy $d = 1$. Podľa pravidla

(6.6) sa upravuje aj prah perceptrónu $w_{n+1} = \theta$.

TRÉNOVACIA MNOŽINA

$A_{train} = \{(\bar{x}^1, d^1)(\bar{x}^2, d^2)...(\bar{x}^p, d^p)....(\bar{x}^P, d^P)\}$ zložená z P dvojíc vstupných vektorov a k nim prislúchajúcich požadovaných výstupov je **trénovacia množina** (angl. *training set*). Potom je algoritmus trénovania perceptrónu takýto:

KROK 1: Zvolíme $\alpha \in \langle 0, 1\rangle$. Počiatočné váhy (vrátane prahu) inicializujeme ako náhodné čísla $\in (-1, 1)$. Počítadlá nastavíme takto: $k = 1$, $p = 1$, kde k je poradové číslo prechodu cez A_{train} a p je index vzoru. Chyba $E = 0$.

KROK 2: Na vstup dáme vzor $\bar{x}^p$ vypočítame výstup $o = sign(\sum_{j=1}^{n+1} w_j x_j^p)$.

KROK 3: Upravíme váhy tak, že $w_j \leftarrow w_j + \alpha\,(d^p - o^p)\,x_j^p$ pre $j = 1,\ldots,n+1$.

KROK 4: Vypočítame chybu $E \leftarrow E + \frac{1}{2}(d^p - o^p)^2$.

KROK 5: Ak $p < P$, tak polož $p = p + 1$ a choď na krok 2. Inak choď na krok 6.

KROK 6: Ak $E = 0$, ukonči učenie. Inak polož $E = 0$, $p = 1$, $k = k + 1$ a choď na krok 2. Začína sa nový trénovací cyklus (epocha), t.j. nový prechod cez A_{train}.

Rosenblatt (1958) dokázal vetu o konvergencii binárneho perceptrónu, ktorá hovorí, že pre $\forall\alpha\,(\alpha > 0)$ perceptrón nájde koeficienty lineárnej deliacej hranice medzi dvoma triedami po konečnom počte iterácií (6.6), ak takáto hranica existuje. Minsky a Papert neskôr ukázali, že viacvrstvové dopredné siete zložené z takýchto binárnych perceptrónov dokážu riešiť iba lineárne separovateľné problémy (Minsky and Papert, 1969).

6.3 VIACVRSTVOVÉ DOPREDNÉ SIETE A ICH UČENIE

UČENIE METÓDOU SPÄTNÉHO ŠÍRENIA CHÝB

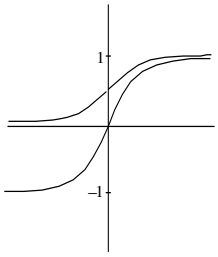
Väčšina reálnych problémov má nelineárny charakter. To znamená, že sa nedajú vyriešiť sčítaním lineárnych hraníc. Problém s obmedzenou výpočtovou schopnosťou perceptrónov sa vyriešil až v roku 1986, keď Rumelhart, Hinton a Williams (1986) zaviedli pravidlo trénovania nazvané **metóda spätného šírenia chýb** (angl. *error backpropagation*) pre dopredné neurónové siete so skrytými neurónmi.

Tzv. **viacvrstvové dopredné** umelé neurónové siete (angl. *multilayer feed-forward ANN*), ktoré sa trénujú týmto pravidlom sú schopné riešiť aj nelineárne problémy. Pri odvodzovaní tohoto algoritmu pre dvojvrstvovú doprednú sieť budeme postupovať podľa (Zurada, 1992). Dopredné neurónové siete sa vyznačujú tým, že v nich existujú iba dopredné spojenia medzi neurónmi. Každý *neurón jednej vrstvy vysiela signály na každý neurón nasledujúcej vrstvy*.

Spojenia do predchádzajúcej vrstvy ani v rámci jednej vrstvy neexistujú.

Najskôr si zovšeobecníme pravidlo δ pre jednovrstvovú neurónovú sieť zloženú zo spojitých perceptrónov. Výstup spojitého perceptrónu je daný vzťahom (6.1). Aktivačná funkcia spojitého perceptrónu *f*(*net*) môže byť ľubovoľná diferencovateľná funkcia. Najčastejšie sa volí výstup v tvare sigmoidy (obr. 6.4). Aj vstupno-výstupná charakteristika biologického neurónu má sigmoidálny tvar. Neskôr si povieme aj o iných spojitých aktivačných funkciách.

OBR.6.4. SIGMOIDA - UNIPOLÁRNA A BIPOLÁRNA

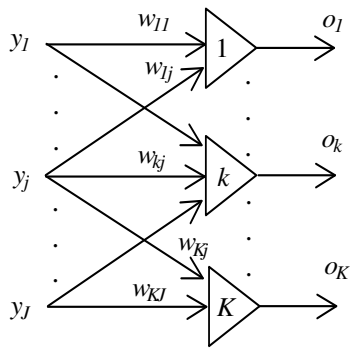


Unipolárna sigmoida: $f(net) = \dfrac{1}{1+\exp(-\lambda\ net)}$ (6.7)

Bipolárna sigmoida: $f(net) = \dfrac{2}{1+\exp(-\lambda\ net)} - 1$ (6.8)

Konštanta $\lambda > 0$ sa nazýva strmosť sigmoidy. Zvyčajne sa používa $\lambda = 1$. V limite pre $\lambda \to \infty$ bipolárna sigmoida prejde na funkciu signum a unipolárna sigmoida na krokovú funkciu.

OBR.6.5. JEDNOVRSTVOVÁ DOPREDNÁ UNS



Majme jednovrstvovú neurónovú sieť ilustrovanú na obr. 6.5. Vstupný vektor je $\bar{y} = (y_1,...,y_j,...,y_J)$. Výstupný vektor je $\bar{o} = (o_1,...,o_k,...,o_K)$, kde o_k = *f*(net_k) a

$$net_k = \sum_{j=1}^{J} w_{kj}\, y_j \qquad (6.9)$$

Nech vždy $y_J = -1$ a $w_{kJ} = \theta_k$, čo je modifikovateľný prah pre $k = 1,\ldots,K$ výstupných neurónov. Požadovaný výstup siete je $\bar{d} = (d_1,...,d_k,...,d_K)$.

CHYBOVÁ FUNKCIA

Chceme, aby sa po naučení skutočný výstup siete rovnal požadovanému výstupu, resp. aby sa mu priblížil čo najviac, a to pre všetky vzory $p = 1,\ldots, P$ z A_{train}. Definujeme účelovú funkciu, ktorá sa volá **chybová funkcia** (angl. *error function*) a má tvar:

$$E_p = \tfrac{1}{2} \sum_{k=1}^{K} (d_{pk} - o_{pk})^2 \qquad (6.10)$$

kde p je index vzoru. E_p je sumou štvorcov chýb na všetkých výstupných neurónoch. Učenie siete spočíva v modifikovaní váh tak, aby sa minimalizovala E_p. Na hľadanie minima E_p sa aplikuje inkrementová metóda negatívneho gradi-

entu[1] nazývaná aj **metóda najprudšieho spádu (najstrmšieho zostupu)** (angl. *steepest descent*).

Zmena váhy w_{kj} je nepriamo úmerná parciálnej derivácii E_p podľa w_{kj}[2]:

$$\Delta w_{kj} = -\alpha \frac{\partial E_p}{\partial w_{kj}} = -\alpha \frac{\partial E_p}{\partial (net_k)} \frac{\partial (net_k)}{\partial w_{kj}} = \alpha \, \delta_{ok} \, y_j \tag{6.11}$$

kde α je rýchlosť učenia. Záporná parciálna derivácia $-\partial E_p/\partial(net_k) = \delta_{ok}$, čo je zovšeobecnený učiaci signál produkovaný k-tym výstupným neurónom. Parciálna derivácia $\partial(net_k)/\partial w_{kj} = y_j$ (pozri rovnicu 6.9). Ďalej si odvodíme, čomu sa rovná δ_{ok}:

$$\delta_{ok} = -\frac{\partial E_p}{\partial (net_k)} = -\frac{\partial E_p}{\partial o_k} \frac{\partial o_k}{\partial (net_k)} = (d_{pk} - o_{pk}) \, f_k' \tag{6.12}$$

f'_k je derivácia aktivačnej funkcie neurónu (sigmoidy) podľa net_k. Pre unipolárnu sigmoidu (6.7) je $f'_k = o_k (1 - o_k)$. Pre bipolárnu sigmoidu (6.8) je $f'_k = (1/2)(1 - o_k^2)$. Pravidlo na zmenu j-tej váhy k-teho výstupného neurónu je potom

ZOVŠEOBECNÉ PRAVIDLO δ PRE VÁHY VÝSTUPNÝCH NEURÓNOV

$$\Delta w_{kj} = \alpha \, (d_{pk} - o_{pk}) \, f_k' \, y_j \tag{6.13}$$

kde $(d_{pk} - o_{pk}) f'_k = \delta_{ok}$, čo je zovšeobecnený chybový signál, ktorý sa spätne šíri na všetky váhy prichádzajúce na k-ty výstupný neurón. Ak by sme položili $f'_k = 1$, dostaneme učiace pravidlo pre perceptrón (6.6).

SKRYTÉ NEURÓNY

Teraz pridáme do našej siete ďalšiu vrstvu neurónov, ktorá sa bude nazývať **skrytá vrstva** alebo **vrstva skrytých neurónov** (obr. 6.6).

Vstup siete je totožný so vstupným vektorom pre skrytú vrstvu $\bar{x} = (x_1, ..., x_i, ..., x_I)$. Výstupné neuróny spracúvajú výstup skrytej vrstvy $\bar{y} = (y_1, ..., y_j, ..., y_J)$, kde $y_j = f(net_j)$ a

$$net_j = \sum_{i=1}^{I} v_{ji} x_i \tag{6.14}$$

I-ty vstup je vždy = −1 ako aj výstup J-teho skrytého neurónu[3]. Modifikovateľné prahy skrytých neurónov sú $v_{jI} = \theta_j$, pre $j = 1, \ldots, J$.

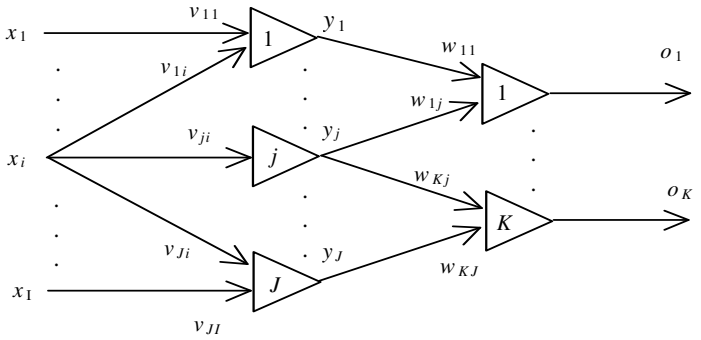
---

[1] Gradient skalárnej funkcie, napr. troch premenných g(x, y, z), udáva smer najprudšieho nárastu jej hodnoty. Vyjadruje to vektor $\overline{\nabla} g(x, y, z) = \frac{\partial g}{\partial x}\bar{i} + \frac{\partial g}{\partial y}\bar{j} + \frac{\partial g}{\partial z}\bar{k}$, kde $\bar{i}, \bar{j}, \bar{k}$ sú jednotkové vektory s v smere jednotlivých osí. Zložky gradientu sú parciálne derivácie podľa jednotlivých premenných. V našom prípade výpočtu gradientu E_p budú zložkami parciálne derivácie podľa jednotlivých váh a prahov.

[2] E_p derivujeme ako zloženú funkciu. Vo všeobecnosti ak máme $h(z(y(x)))$, tak $\partial h/\partial x = (\partial h/\partial z)(\partial z/\partial y)(\partial y/\partial x)$.

[3] Skutočných vstupov je teda len $I-1$ a počet skutočných skrytých neurónov je $J-1$. J-ty skrytý neurón je len akási „atrapa“, ktorej výstup slúži pre prahy výstupných neurónov.

OBR.6.6.
DVOJVRSTVOVÁ DOPREDNÁ UNS



Vzťahy (6.11) až (6.13) popisujú modifikáciu váh medzi skrytou vrstvou a výstupnou vrstvou. Teraz si odvodíme vzťahy pre modifikáciu váh medzi vstupom a skrytou vrstvou. Aj tieto váhy sa musia meniť tak, aby sa minimalizovala chyba E_p vyjadrená rovnicou (6.10) pomocou gradientovej metódy najprudšieho spádu.

Formula pre modifikáciu vstupných váh v_{ji} je takáto:

$$\Delta v_{ji} = -\alpha \frac{\partial E_p}{\partial v_{ji}} = -\alpha \frac{\partial E_p}{\partial (net_j)} \frac{\partial (net_j)}{\partial v_{ji}} = \alpha \, \delta_{yj} \, x_i \tag{6.15}$$

Záporná parciálna derivácia $-\partial E_p/\partial(net_j) = \delta_{yj}$, čo je zovšeobecnený učiaci signál produkovaný j-tym skrytým neurónom. δ_{yj} je zovšeobecnená chyba šírená na vstupné váhy. Parciálna derivácia $\partial(net_j)/\partial v_{ji} = x_i$ (pozri rovnicu 6.14). Ďalej si odvodíme, čomu sa rovná δ_{yj} :

$$\delta_{yj} = -\frac{\partial E_p}{\partial (net_j)} = -\frac{\partial E_p}{\partial y_j} \frac{\partial y_j}{\partial (net_j)} = -\frac{\partial E_p}{\partial y_j} f_j' \tag{6.16}$$

f'_j je derivácia výstupnej funkcie skrytého neurónu (sigmoidy) podľa net_j.

$$\frac{\partial E_p}{\partial y_j} = -\sum_{k=1}^{K} (d_{pk} - o_{pk}) \frac{\partial \{f(net_k)\}}{\partial y_j} = -\sum_{k=1}^{K} (d_{pk} - o_{pk}) \frac{\partial f(net_k)}{\partial (net_k)} \frac{\partial (net_k)}{\partial y_j} \tag{6.17}$$

Keďže f'_k je derivácia sigmoidy výstupného neurónu podľa net_k a $\partial(net_k)/\partial y_j = w_{kj}$ (pozri vzťah 6.9), potom

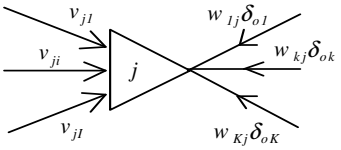
$$\frac{\partial E_p}{\partial y_j} = -\sum_{k=1}^{K} (d_{pk} - o_{pk}) \, f_k' \, w_{kj} = -\sum_{k=1}^{K} \delta_{ok} \, w_{kj} \tag{6.18}$$

ZOVŠEOBECNÉ PRAVIDLO δ PRE VÁHY SKRYTÝCH NEURÓNOV

Dosadením do (6.16) dostávame vzťah pre δ_{yj}:

$$\delta_{yj} = \left(\sum_{k=1}^{K} \delta_{ok} \, w_{kj} \right) f_j' \tag{6.19}$$

OBR.6.7.
SPÄTNÉ ŠÍRENIE CHÝB NA SKRYTÝ NEURÓN



A nakoniec vzťah pre učenie váh medzi vstupom a skrytými neurónmi:

$$\Delta v_{ji} = \alpha \left(\sum_{k=1}^{K} \delta_{ok}\ w_{kj} \right) f_j'\ x_i \qquad (6.20)$$

ZOVŠEOBECNENIE PRE *n* SKRYTÝCH VRSTIEV

Predstavme si, že máme n skrytých vrstiev. Vo všeobecnosti pre n-tú skrytú vrstvu platí:

$$\Delta v_{ji}^{n} = \alpha\ \delta_{yj}^{n}\ x_i^{n-1} \qquad (6.21)$$

kde

$$\delta_{yj}^{n} = \left(\sum_{k=1}^{K} \delta_{ok}^{n+1}\ w_{kj}^{n+1} \right) (f_j^{n})' \qquad (6.22)$$

$(f_j^n)'$ je derivácia aktivačnej funkcie skrytého neurónu n-tej vrstvy podľa net_j^n.

ZRÝCHLENIE UČENIA

Často používaná metóda na zrýchlenie konvergencie do minima chybovej funkcie je **momentum**:

$$\Delta w_{kj}(t) = \Delta w_{kj}(t) + \mu\ \Delta w_{kj}(t-1) \quad \wedge \quad \Delta v_{ji}(t) = \Delta v_{ji}(t) + \mu\ \Delta v_{ji}(t-1) \qquad (6.23)$$

VEĽKOSŤ α A μ

kde konštanta $\mu \in \langle 0, 1 \rangle$ sa nazýva momentum alebo momentový člen. Zvyčajne sa volí $\mu < \alpha$. Rádovo je zvyčajne veľkosť týchto koštánt 10^{-2} a 10^{-1}. Čo sa týka konkrétnych hodnôt, tie sa zisťujú experimenálne pre daný problém.

Majme $A_{train} = \{(\bar{x}^1, \bar{d}^1)(\bar{x}^2, \bar{d}^2)...(\bar{x}^p, \bar{d}^p)....(\bar{x}^P, \bar{d}^P)\}$. Potom je algoritmus trénovania doprednej neurónovej siete pomocou spätného šírenia chýb takýto:

KROK 1: Zvolíme $\alpha \in \langle 0, 1 \rangle$. Počiatočné váhy (vrátane prahov) inicializujeme ako malé náhodné čísla[4] napr. $\in (-0.5, 0.5)$. Počítadlá a chybu nastavíme takto: $k = 1$, $p = 1$, $E = 0$. Poradové číslo prechodu cez A_{train} je k, index vzoru je p a E je kumulovaná chyba:

KUMULOVANÁ CHYBA

$$E = \sum_{p=1}^{P} E_p \qquad (6.24)$$

kde E_p sa počíta podľa (6.10). Zvolíme malé kladné číslo ε, ktoré nám bude slúžiť na zastavenie učenia.

KROK 2: Na vstup dáme vzor $\bar{x}^p$ a vypočítame $\bar{y}^p$ a $\bar{o}^p$.

[4] Neexistuje predpis na výber počiatočných váh. Môže sa stať, že sieť je naštartovaná tak, že skončí v lokálnom minime chyby. Vtedy treba začať učenie odznova s novými počiatočnými váhami.

KROK 3: Pre každý výstupný neurón vypočítame δ_{ok} podľa (6.12) a pre každý skrytý neurón δ_{yj} podľa (6.19).

KROK 4: Modifikujeme váhy pre výstupné neuróny $w_{kj} \leftarrow w_{kj} + \alpha\, \delta_{ok}\ y_j$ a pre skryté neuróny $v_{ji} \leftarrow v_{ji} + \alpha\, \delta_{yj}\ x_i$.

KROK 5: Ak $p < P$, tak polož $p = p + 1$ a choď na krok 2. Inak choď na krok 6.

KROK 6: Bez modifikácie váh prejdeme cez A_{train} a vypočítame kumulovanú chybu E, ktorá nám povie aká je chyba po jednej epoche učenia. Ak $E < \varepsilon$, ukonči učenie. Inak: vzory v A_{train} premiešaj tak, aby v každej epoche učenia prichádzali v inom náhodnom poradí. Polož $E = 0$, $p = 1$, $k = k + 1$ a choď na krok 2. Začína sa nový trénovací cyklus, nová epocha trénovania, t.j. nový prechod cez A_{train}.[5]

Dopredná viacvrstvová sieť ako univerzálny aproximátor

V tejto časti si povieme aké úlohy rieši dopredná viacvrstvová neurónová sieť trénovaná pomocou spätného šírenia chýb. Tiež sa dozvieme praktické rady ohľadom výberu A_{train}, výberu počtu skrytých neurónov, ukončenia trénovania atď.

Majme doprednú NS s jedným výstupným neurónom a jednou skrytou vrstvou. Majme $A_{train} = \{\bar{x}^1, \bar{x}^2, ...\bar{x}^p, ...\bar{x}^P\}$ tvorenú vstupnými vektormi dimenzie I.

VETA O UNIVERZÁLNEJ APROXIMÁCII

Pre $\forall \varepsilon > 0\ \exists$ taká funkcia $G(\bar{x}) = f\left(\sum_{j=1}^{J} w_j\ f(\sum_{i=1}^{I} v_{ji} x_i)\right)$, kde w_j je váha synapsy medzi j-tym skrytým neurónom a výstupným neurónom, v_{ji} je váha synapsy medzi i-tym vstupom a j-tym skrytým neurónom a $f(z)$ je diferencovateľná aktivačná funkcia. Pre ľubovoľnú spojitú funkciu F: $\Re^n \rightarrow (0, 1)$, ktorá je definovaná nad konečnou množinou A_{train} platí, že $\sum_{p=1}^{P} \left| F(\bar{x}^p) - G(\bar{x}^p) \right| < \varepsilon$.

Hovoríme, že funkcia G aproximuje funkciu F nad trénovacou množinou A_{train} s presnosťou ε (Hornik et al., 1989).

Táto veta hovorí, že existuje taká dvojvrstvová UNS, ktorá aproximuje F pre $A_{train.}$ Rovnica (6.10) vyjadruje sumu štvorcov chýb, ktorú učením minimalizujeme. Trénovaním siete upravujeme váhy, ktoré sú vlastne koeficientami polynómu G, ktorý sieť prekladá cez body dané trénovacou množinou (pozri obr. 6.8). Cieľom je extrapolácia funkčných hodnôt mimo A_{train}

[5] Tento algoritmus zodpovedá tzv. inkrementovému učeniu, keď upravujeme váhy po prezentácii každého jedného vstupu. Môže sa použiť aj tzv. "batch" učenie, keď váhy zmeníme až po prechode celou trénovacou množinou. Výsledná zmena váhy je súčtom zmien vyvolaných všetkými vstupnými vzormi.

GENERALIZÁCIA

čiže predikcia (predpoveď) *aké budú funkčné hodnoty pre vstupné vektory, ktoré sieť nikdy nevidela*. Táto schopnosť UNS sa volá **zovšeobecňovanie** alebo **generalizácia** (angl. *generalisation*).

TESTOVANIE

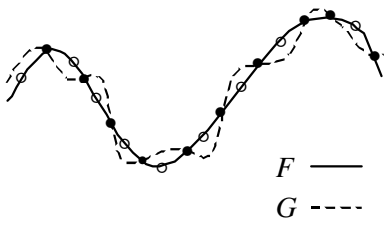
VALIDÁCIA

Aby sme túto schopnosť siete mohli testovať, nepoužívame všetky dáta, ktoré máme k dispozícii na trénovanie, ale rozdelíme si ich na *trénovaciu* a *testovaciu* množinu A_{train} a A_{test} (plné a prázdne krúžky na obr. 6.8). Testovacia množina sa nazýva aj *validačná* množina. Pomocou A_{train} modifikujeme váhy a pomocou A_{test} zisťujeme chybu zovšeobecňovania. Pracujeme s kumulovanou chybou E (6.24) pre A_{train} aj $A_{test.}$

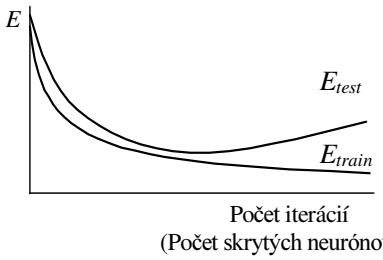
PREUČENIE

Je dôležité, aby sme dáta správne rozdelili. A_{train} musí rovnomerne pokrývať daný interval. Zistilo sa, že v priebehu trénovania kumulovaná E_{train} stále klesá, zatiaľ čo E_{test} začne od určitého kroku rásť (obr. 6.9). Dochádza k javu, ktorý sa nazýva **preučenie** siete (angl. *overfitting*) alebo pretrénovanie, či premodelovanie dát.

OBR.6.8. APROXIMÁCIA FUNKCIE POMOCOU UNS



OBR.6.9. VÝVOJ CHYBY POČAS TRÉNOVANIA



KLASIFIKÁCIA DO K TRIED

Doprednú neurónovú sieť možno trénovať pomocou metódy spätného šírenia chýb aj na klasifikáciu vzorov do K tried, ktoré majú nelineárne deliace hranice. Vtedy má sieť K výstupných neurónov. Napr. pre 3 triedy, požadované výstupné vektory navrhneme takto: $\bar{d}^1 = (1,0,0)$, $\bar{d}^2 = (0,1,0)$ a $\bar{d}^3 = (0,0,1)$, ak je aktivačná funkcia výstupných neurónov unipolárna sigmoida.

Skutočný výstup musíme nejako interpretovať. Zvyčajne považujeme skutočný výstup za = 1, keď $o \geq 0{,}9$ a za = 0, keď $o \leq 0{,}1$. Takúto interpretáciu používame pre výpočet počtu chybných výsledkov na A_{train} a A_{test}, t.j. N_{train}^{error} a N_{test}^{error}. Pod chybným výsledkom rozumieme nesprávne klasifikovaný vzor $\bar{x}^p$.

Pri trénovaní na klasifikáciu sledujeme vývoj počtu chybných výsledkov, ale je užitočné sledovať aj vývoj kumulovanej chyby (6.24), ktorú počítame pomocou nezaokrúhleného výstupu. Na zastavenie učenia môžeme použiť kritérium $N_{train}^{error} \leq N_{max}$.

„DOÚČANIE“

Ak chceme sieť doučiť novú triedu, musíme trénovanie začať odznova so všetkými triedami. Ak chceme sieť doučiť iba zopár nových vzorov, na ktoré už máme triedu, ale sieť ich nie je schopná správne zovšeobecniť, môžeme ju skúsiť doučiť. Ak sa to nepodarí, musíme začať trénovanie odznova.

ASOCIÁCIA

Pre každý typ úlohy možno vo všeobecnosti povedať, že viacvrstvová dopredná neurónová sieť sa učí asociovať vstupné vektory $\bar{x}^p$ s výstupnými vektormi $\bar{d}^p$.

EXTRAKCIA PRÍZNAKOV

Skryté neuróny pritom vykonávajú extrakciu príznakov, na základe ktorých je možno objekty rozdeliť do tried. Nevýhodou neurónových sietí ako detektorov príznakov je, že obvykle vieme len veľmi ťažko na základe aktivít skrytých neurónov určiť, aké príznaky sieť vlastne extrahovala.

SKORÉ ZASTAVENIE UČENIA

Užitočnou metódou na zastavenie učenia, keď nevieme dopredu ε či N_{max}, alebo sa tieto ukazovatele po mnoho iterácií výrazne nemenia, je metóda nazvaná **skoré** (**optimálne**) **zastavenie** (angl. *early* (*optimal*) *stopping*) (Bishop, 1995):

1. Rozdeľ dáta na 2 neprekrývajúce sa podmnožiny A_{train} a A_{test} (napr. náhodne vyber 75% vzorov do jednej a zvyšok do druhej).
2. Po každom trénovacom cykle vypočítaj kumulovanú chybu na trénovacej a testovacej množine E_{train} a E_{test} (respektíve N_{train}^{error} a N_{test}^{error}, keď učíme sieť klasifikovať).
3. Zastav učenie, keď E_{test} (resp. N_{test}^{error}) začne rásť.

OPTIMÁLNY POČET SKRYTÝCH NEURÓNOV

Selekcia modelu (angl. *model selection*) znamená výber optimálneho počtu skrytých neurónov[6] (Bishop, 1995). Neplatí, že čím viac skrytých neurónov, tým lepšie (pozri obr. 6.9). Zvyčajným postupom je, že určíme chybu zovšeobecňovania E_{test} pre každý model (počet skrytých neurónov) a vyberieme ten model, pre ktorý je E_{test} minimálna. Korektnou štatistickou metódou je **vrstvová *k*-násobná prekrížená validácia** (angl. *stratified k-fold cross-validation*):

1. Množinu všetkých dát A rozdelíme na k podmnožín, $A_{test}^1, A_{test}^2, ..., A_{test}^k$, ktoré majú nulový prekryv. Zvyčajne $k = 10$. Je dôležité jednotlivé podmnožiny vyberať tak, aby obsahovali približne také isté kvantitatívne zastúpenie tried ako pôvodná množina A.
2. Každý model (počet skrytých neurónov) sa trénuje k-krát, a to na k trénovacích množinách $A_{train}^i = A \setminus A_{test}^i$ pre $i = 1,\ldots, k$. Učenie možno zastaviť podľa metódy skorého zastavenia, keď E_{test}^i začína rásť.
3. Po skončení všetkých trénovaní, pre každý model vypočítame „cross“-validačný odhad presnosti klasifikácie tak, že

$$CV = \frac{1}{k}\sum_{i=1}^{k} V^i \tag{6.25}$$

pričom pre klasifikačnú úlohu je $V^i = N_{test}^{error}$, t.j. počet chybných klasifikácií pre i-tu testovaciu množinu A_{test}^i. Pre aproximačnú úlohu je $V^i = E_{test}^i$.

4. Vyberieme ten model (počet skrytých neurónov), pre ktorý je CV minimálny.

[6] Ukázalo sa, že pridávanie skrytých vrstiev výrazne nezlepšuje výkon doprednej siete. Stačí jedna-dve skryté vrstvy.

Pre každý model môžeme vypočítať strednú kvadratickú odchýlku pre *CV* a interval spoľahlivosti.

SPRIEMERŇOVANIE

V kontexte aproximácie funkcií i klasifikácie sa ako optimalizovaná aproximácia berie priemer predikcií členov víťazného modelu. Inými slovami, pri zovšeobecňovaní na úplne nové príklady sa nespoliehame iba na jednu sieť, ale berieme do úvahy aj „mienku“ ostatných členiek víťazného modelu (angl. *bagging*).

SVOJPOMOCNÝ VÝBER

Svojpomocný výber (angl. *bootstrap*) je populárna metóda na konštrukciu A^i_{test} (Efron and Tibshirani, 1993). Každú A^i_{test} zostrojíme tak, že z A vyberieme náhodne (rovnomerná náhodnosť) $(1/k)$ % prvkov so zámenou (angl. *with replacement*). Náhodný výber so zámenou znamená, že po každom náhodnom výbere vrátime prvok späť, aby mohol poslúžiť pre ďalší výber. Potom sa pozrieme na každú A^i_{test} a prvky, ktoré obsahuje vylúčime z príslušnej A^i_{train} tak, že $A^i_{train} = A \setminus A^i_{test}$ (množinový rozdiel).

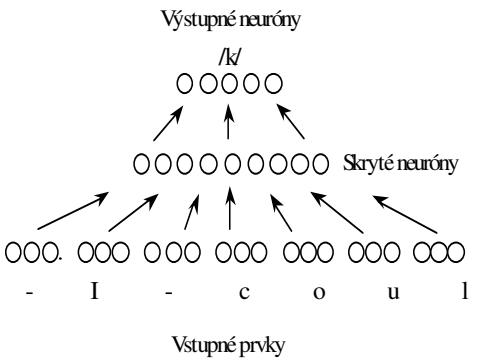
Aplikácie dopredných neurónových sietí trénovaných spätným šírením chýb

V tejto časti uvádzame najznámejšie prvé aplikácie viacvrstvových dopredných neurónových sietí trénovaných pomocou metódy spätného šírenia chýb, ktoré slúžia ako vzor im podobných aplikácií.

NETTALK – ČÍTANIE ANGLICKÉHO TEXTU NAHLAS

Sejnowski a Rosenberg (1987) vytvorili neurónovú sieť NETTalk, ktorá bola schopná čítať písaný anglický text nahlas, za použitia príslušných vstupných a výstupných zariadení. Pritom angličtina patrí medzi tie jazyky, v ktorých je výslovnosť hlásky často viazaná na kontext susedných hlások v slove. Pomerne malá neurónová sieť sa bola schopná naučiť väčšinu pravidelností a mnoho výnimiek z pravidiel výslovnosti. Na obr. 6.10 ilustrujeme použitú architektúru. Šipky indikujú dopredné spojenia každého prvku s každým prvkom v nasledujúcej vrstve. Rýchlosť učenia bola 0,2 a počiatočné váhy z intervalu ⟨-0,3; 0,3⟩.

OBR.6.10. ARCHITEKTÚRA NETTALKU



Vstup pozostával zo siedmych skupín, pričom každá skupina kódovala jedno písmeno zo vstupného textu. Každá skupina obsahovala 26 prvkov, ktoré kódovali 26 písmen anglickej abecedy systémom „*one-hot-encoding*“, teda pre príslušné písmeno len jeden prvok má hodnotu 1, ostatné majú hodnotu 0. V každej skupine boli ešte tri ďalšie

prvky, kódujúce diakritiku a hranice slova.

V skrytej vrstve sa použilo 80 neurónov. Požadovaným výstupom siete bola správna fonéma (zvuková podoba) asociovaná so znakom v strede, t.j. so štvrtým znakom zľava. Vstup teda tvorilo časové okno pozostávajúce zo siedmych znakov, pričom 3 predchádzajúce a 3 nasledujúce znaky predstavovali kontext pre výslovnosť stredného znaku. Text sa posúval cez vstupné okno znak po znaku. V každom kroku, sieť vypočítala príslušnú fonému a váhy sa upravili po každom slove, pričom výsledná zmena váh bola súčtom zmien po každej hláske v slove. Chybový signál sa mohol šíriť spätne len vtedy, keď rozdiel medzi požadovanou a skutočnou hodnotou na výstupe neurónu bol väčší ako 0,1. Vo výstupnej vrstve bolo 23 neurónov, ktoré reprezentovali 23 artikulačných príznakov (znelosť, nazálnosť, frikatívnosť, atď.) a 3 dodatočné neuróny, ktoré reprezentovali hranice hlásky a dôraz. Čiže kódovanie vo výstupnej vrstve bolo distribuované a nie lokálne („one-hot“) ako vo vstupnej vrstve.

Autori použili takú interpretáciu výstupu na priradenie do vzorovej fonémy, ktorú nazvali najlepší odhad (angl. *best guess*). Určujúci bol najmenší uhol medzi skutočným a požadovaným výstupným vektorom. Prvá trénovacia množina pozostávala z 1024 slov extrahovaných z detskej reči, kontinuáneho rečového prejavu. Keď bola sieť trénovaná na detskej reči, po 50 epochách dosiahla 95%-nú úspešnosť na trénovacej množine a 78%-nú úspešnosť na testovacej množine pozostávajúcej zo 439 nových slov. Autori uvádzajú, že bolo fascinujúce počúvať výstup siete počas učenia, ktorý pripomínal zrýchlené učenie sa hovoriť u dieťaťa. Ako prvé sa sieť naučila diskriminovať medzi samohláskami a spoluhláskami. Avšak, sieť substituovala rovnakú samohlásku namiesto všetkých samohlások a rovnako aj so spoluhláskami, takže spočiatku len „bľabotala“. Potom sa naučila rozpoznávať hranice slov a produkovala akési pseudoslová. Po 10 epochách sa jej už dalo rozumieť. Najčastejšie chyby boli v rozdieli vo výslovnosti „th“ v takých slovách ako „thesis“ a „these“.

Druhú trénovaciu množinu tvorilo 1000 najčastejšie používaných slov. Po 30 epochách trénovania, sieť dosiahla úspešnosť 98% na trénovacej a 90% na testovacej množine (náhodne vybraté iné slová), pri použití 120 neurónov v skrytej vrstve.

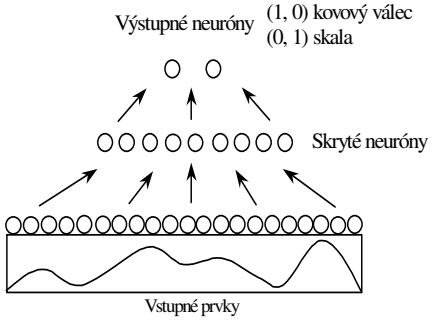
V oboch prípadoch skúsili autori poškodiť synaptické váhy pridaním malého náhodného čísla z intervalu $\langle -0{,}5;\ 0{,}5\rangle$. Nemalo to vplyv na výkon siete. Keď zväčšili poškodenie, sieť sa veľmi rýchlo doučila to, čo vedela. V tomto projekte boli skombinované dva rozdielne vývinové procesy: učenie sa hovoriť a učenie sa čítať. Avšak, keď sa dieťa učí čítať, už má úplne naučené fonetické reprezentácie hlások, takže úplne to nie je model ani jedného z týchto procesov. Autori poukazujú, že aj takýto jednoduchý model môže slúžiť ako východisko pre zložitejšie modely, pomocou ktorých by sa dali skúmať dyslexie, poruchy čítania. Umelá neurónová sieť sa totižto dá rozličným spôsobom učiť a poškodzovať, pričom možno pozorovať paralely medzi chybami, ktoré vznikajú v sieti a chybami, ktoré produkujú dyslektici.

Ďalším rozšírením modelu by mohlo byť trénovanie siete na odšumenie hovorenej reči. Trénovaciu množinu by tvorili slová, slabiky alebo hlásky, hovorené rozličnými ľuďmi za rozličných podmienok. Požadovanými výstupmi by mohli byť jasne artikulované slová (slabiky, hlásky). Vďaka schopnosti zovšeobecňovať by takýto systém našiel uplatnenie napríklad v telekomunikácii.

KLASIFIKÁCIA PODMORSKÝCH SONAROVÝCH SIGNÁLOV

Príkladom na použitie umelej neurónovej siete na klasifikáciu signálov je práca Gormana a Sejnowského (1988), ktorí trénovali doprednú neurónovú sieť na klasifikáciu podmorských sonarových signálov. Úlohou bolo naučiť sa rozlišovať medzi kovovým objektom a prírodným útvarom (skalou). Na obr. 6.11 ilustrujeme použitú architektúru. Rýchlosť učenia bola 0,2 a počiatočné váhy $\in \langle -0{,}3;\ 0{,}3\rangle$.

OBR.6.11. ARCHITEKTÚRA SONAROVÉHO KLASIFIKÁTORA



Sonarové signály boli namerané z rôznych uhlov na skutočných objektoch zhruba rovnakej veľkosti. Tieto časové signály sa predspracovali pomocou Fourierovej transformácie, výsledkom ktorej bolo spektrum frekvencií, každá s inou intenzitou.

Spektrum frekvencií sa rozdelilo na 60 rovnakých frekvenčných intervalov a intenzity sa normalizovali do intervalu $\langle 0;\ 1\rangle$, keďže neuróny mali unipolárnu sigmoidálnu aktivačnú funkciu. Čiže vstup pozostával zo 60 reálnych čísiel, výstup mal 2 neuróny kódujúce 2 triedy (pozri obr. 6.11) a sieť mala 24 skrytých neurónov. Množina signálov pozostávala zo 111 príkladov pre válec a 97 príkladov pre skalu. Z týchto sa pre každú sieť vybralo náhodne 16 príkladov na testovanie a zvyšok tvoril trénovaciu množinu. Výsledky sa prezentujú ako priemer z 10 sietí, pričom každá bola inicializovaná inými náhodnými váhami.

Po 30 epochách trénovania dosiahla sieť 100%-nú úspešnosť na trénovacej množine a 89%-nú úspešnosť na testovacej množine. Výsledky sa porovnali s klasifikáciou na základe K najbližších susedov (angl. *K-nearest neighbor*) a s výkonom ľudských expertov. Klasifikácia pomocou K najbližších susedov je založená na spočítaní euklidovskej vzdialenosti medzi testovacím signálom a všetkými signálmi z trénovacej množiny. Hoci obvykle sa používa nepárne K, napr $K = 3$ alebo 5, autori použili $K = 2$. Ak dvaja najbližší susedia patria do tej istej triedy, priradí sa tam aj nový signál. Ak každý patrí do inej triedy, rozhodneme sa na základe hodu mincou. Touto metódou sa získala 82,7%-ná úspešnosť na testovacej množine. Človek dosiahol úspešnosť tiež iba 82%. V tomto experimente teda neurónové siete predčili ľudského experta, nebýva to však pravidlom.

ALVINN - AUTO-

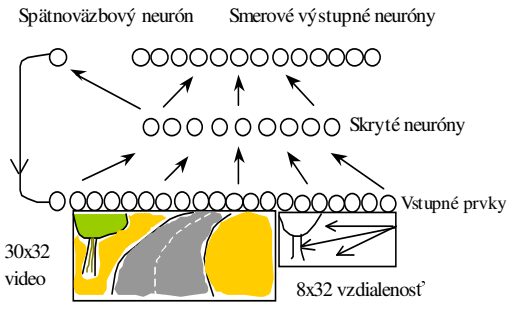
Opis projektu nazvaného ALVINN (Autonomous Land Vehicle In a Neural

NÓMNE POZEMNÉ VOZIDLO

Network) Pomerleau (1989) nám poskytne predstavu ako neurónovo riadiť pohybujúce sa vozidlá, roboty a pod.

Vstupom do doprednej siete boli kamerou nasnímané obrázky cesty pred vozidlom a signál z laserového detektora vzdialenosti. Výstupom siete bol smer, ktorým sa malo vozidlo vydať, aby sledovalo stredovú čiaru cesty. Vozidlo sa pohybovalo konštantnou rýchlosťou asi 5 km/h. Vstup z kamery predstavoval 960 = 30 x 32 segmentov z nasnímanej cesty v modrej časti svetelného spektra kvôli najväčšiemu kontrastu medzi cestou a „necestou“. Každý vstup číselne vyjadroval intenzitu modrého svetla v danom segmente nasnímaného obrazu. Druhou časťou zloženého vstupu bolo 8 x 32 segmentov vzdialenostného obrazu nasnímaného laserovým detektorom vzdialenosti. Vo vstupe sa nachádzal aj prvok, ktorý indikoval, či je cesta v danom momente svetlejšia alebo tmavšia ako necesta v predchádzajúcom kroku. Architektúra siete je znázornená na obr. 6.12.

OBR.6.12. ARCHITEKTÚRA „NEURÓNOVÉHO ŠOFÉRA“



Teda, 1217 vstupov prichádza na 29 skrytých neurónov. Skryté neuróny sú spojené doprednými väzbami so 46 výstupnými neurónmi. Jeden výstupný neurón zabezpečuje spätnoväzbovú informáciu o svetelnej intenzite cesty vzhľadom k neceste a zvyšných 45 neurónov kóduje smer pohybu.

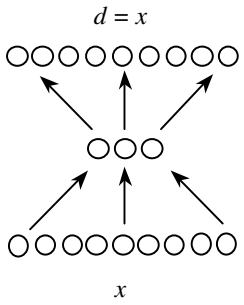
Najvyššia hodnota výstupu stredného neurónu indikuje priamy smer. Neuróny najviac vľavo a vpravo indikujú maximálne zatočenie doľava respektíve doprava. Ostatné neuróny reprezentujú diskrétny uhol otočenia medzi nulovým a maximálnym uhlom. Pri trénovaní siete, treba samozrejme sieti poskytnúť informáciu o správnom uhle otočenia vozidla. Trénovacia množina pozostávala z 1200 rozličných snímok cesty, nasnímaných z rozličných uhlov a za rozličných svetelných podmienok (tie boli nasimulované počítačovo). Autor použil skutočné upravené vozidlo so snímačmi na streche a počítačom namiesto šoféra. Keď bolo všetko odladené, stačil polhodinový tréning, aby vozidlo spoľahlivo prešlo aj po nových cestičkách v miestnom parku. Sieti veľmi pomáhalo to, že keď spravila chybu, oprava chyby sa zaradila do jej trénovacej množiny.

ROZPOZNÁVANIE PÍSMEN A KOMPRESIA

Jeden z neurónových projektov v AT&T Bell Laboratories bol zameraný na rozpoznávanie rukou písaných PSČ (LeCun et al. 1991). Lokalizácia číslic na obálke aj ich odlíšenie od susedov sa urobilo „ručne“, aj lineárna transformácia zo 40 x 60 pixelov na 16 x 16. Táto mapa rozličných úrovní šede predstavovala vstup do doprednej siete s tromi skrytými vrstvami. Prvé dve skryté vrstvy obsahovali 12 skupín po 8 x 8 neurónov, ktoré extrahovali príznaky z rozličných častí predchádzajich máp. Tretia skrytá vrstva mala 30 neurónov a výstupná vrstva mala 10 neurónov, ktoré kódovali 10 číslic systémom „one-hot“. Celkovo

mala sieť 1256 neurónov a 64 660 spojení. Po 23 prezentáciách množiny 7291 skutočných ľuďmi napísaných číslic, dosiahla sieť úspešnosť 98,6% na trénovacej množine a 95% na testovacej množine pozostávajúcej z 2007 nových číslic. Keď ten istý tréning vykonali so sieťou majúcou iba jednu obyčajnú skrytú vrstvu so 40 neurónmi, dosiahli úspešnosť 92% na testovacej množine, čiže zhoršenie iba o 3%. Môže sa to zdať málo, ale v takých prípadoch ako medicínska diagnostika alebo predikcia, môžu aj 3% znamenať veľa. V každom prípade, aj táto štúdia prakticky ukázala, že veľmi záleží na predspracovaní vstupu pre neurónovú sieť. V súčasnosti sa s väčším úspechom skúšajú rozličné metódy identifikácie nosných príznakov (pozri kapitolu o vnímaní) a vstupom do siete nie je samotný objekt, ale súbor hodnôt relevantných príznakov.

OBR.6.13. ARCHITEKTÚRA „ÚZKE HRDLO "



Jeden zo spôsobov ako extrahovať zo vstupu príznaky (aj keď nevieme aké) a zároveň zredukovať dimenziu vstupu pri zachovaní skoro všetkej informácie, je použiť na predspracovanie neurónovú sieť s architektúrou „úzke hrdlo“ (obr. 6.13). Požadovaným výstupným vektorom je vstupný vektor samotný. V skrytej vrstve použijeme malý počet neurónov a sieť trénujeme. Po skončení trénovania, máme na skrytej vrstve zakódované a skomprimované vstupy. Cottrell et al. (1989) takúto architektúru použili na stratovú kompresiu súborov.

Výhodou umelých neurónových sietí je to, že netreba poznať formálny model riešeného problému. Namiesto toho stačí vhodne vybratá trénovacia množina a vhodná architektúra siete. Trénovanie pomocou spätného šírenia chýb nastaví parametre (váhy a prahy) siete tak, aby sme dostali akceptovateľné riešenie. K riešeniu sa dá dopracovať simuláciami a experimentovaním namiesto rigorózneho a formálneho prístupu k problému.

Hranie hier a učenie s odmenou a trestom

UČENIE S ODMENOU A TRESTOM

Pomocou viacvrstvových dopredných sietí je možné hrať hry. Alebo navigovať robota v teréne. Ide o úlohy takého typu, keď ku každému vstupnému vektoru môže existovať viacero vhodných požadovaných výstupov. Na takéto úlohy sa používa tzv. **učenie s odmenou a trestom**[7] (angl. *reinforcement learning*). Princípom je, že sieti povieme, či jej výstup bol dobrý alebo zlý, ale nepovieme jej aký presne mal byť. Vysvetlíme si účinný **algoritmus TD** a **TD(λ)** (TD z anglického *temporal difference*) (Sutton, 1988; Tesauro, 1990).

Nech vstupný vektor $\vec{x}^{(t)} = (x_1,...,x_i,...,x_I)$ kóduje pozíciu P na hracom poli v diskrétnom časovom kroku t. Pri kódovaní pozície môžme postupovať napr.

[7] Niekedy sa používa aj termín známkované učenie (Oravec et al., 1998). V psychológii sa „reinforcement learning“ prekladá ako posilňovanie.

takto:

$$x_i = \begin{cases} 1 & ak\ i-te\ pole\ obsahuje\ \ vlastnú\ \ figúrku \\ 0 & ak\ i-te\ pole\ neobsahuje\ \ nijakú\ \ figúrku \\ -1 & ak\ i-te\ pole\ obsahuje\ \ súperovu\ \ figúrku \end{cases} \tag{6.26}$$

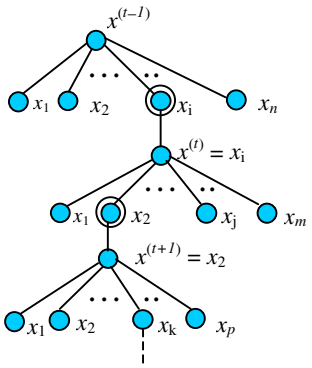
Majme dvojvrstvovú doprednú sieť s jedným výstupným neurónom. Majme sekvenciu pozícií (partiu) $\overline{x}^{(1)},...,\overline{x}^{(t)},...\overline{x}^{(T)}$ a tri požadované výstupy siete, ktoré sa nazývajú **odmena** (angl. *reward*):

ODMENA (TREST)

$$r = \begin{cases} 1 & ak\ sekvencia\ pozícií\ vedie\ \ k\ výhre \\ 0.5 & ak\ sekvencia\ pozícií\ vedie\ \ k\ remíze \\ 0 & ak\ sekvencia\ pozícií\ vedie\ \ k\ prehre \end{cases} \tag{6.27}$$

OBR.6.14.

VÝBER ŤAHU - ILUSTRÁCIA



Na vstup siete dáme postupne všetky možné ťahy (pozície) a nasledujúci ťah v čase t vyberieme podľa najväčšej hodnoty skutočného výstupu siete $o^{(t)}$ (na obr. 6.14 zakrúžkované pozície). Počas učenia sieti predstavujeme stovky (až tisícky) partií. Časť partií vedie k výhre, časť k prehre, respektíve k remíze. Po skončení trénovania vie sieť ohodnotiť každú pozíciu v čase t číslom $o^{(t)}$, ktoré je úmerné pravdepodobnosti výhry. Podľa najväčšieho $o^{(t)}$ sa vyberie ťah v čase t. Možnosti ťahov na ohodnotenie musíme sieti predkladať my, respektíve príslušný algoritmus, sieť ich len ohodnotí.

Pomocou gradientovej metódy najprudšieho spádu hľadáme optimálne váhy siete, a teda minimum chybovej funkcie:

$$E = \tfrac{1}{2}\sum_{t=1}^{T}(r - o^{(t)})^2 \tag{1.28}$$

UČENIE

Symbol w bude označovať ktorúkoľvek váhu v sieti, aby sme sa vyhli nadbytočnému indexovaniu. Čiže, nech w označuje aj w_{kj} aj v_{ji}. Potom:

$$\Delta w = -\alpha\frac{\partial E}{\partial w} = \alpha\sum_{t=1}^{T}(r - o^{(t)})\frac{\partial o^{(t)}}{\partial w} \tag{1.29}$$

Nech $r = o^{(T+1)}$. Potom

$$o^{(T+1)} - o^{(t)} = (o^{(T+1)} - o^{(T)}) + (o^{(T)} - o^{(T-1)}) + ... + (o^{(t+1)} - o^{(t)}) \tag{1.30}$$

Po dosadení do (1.29) dostaneme

$$\Delta w = \alpha \sum_{t=1}^{T} \left[\sum_{n=t}^{T} (o^{(n+1)} - o^{(n)}) \right] \frac{\partial o^{(t)}}{\partial w} \tag{1.31}$$

Preskupením pravej strany získame vzťah

$$\Delta w = \alpha \sum_{t=1}^{T} (o^{(t+1)} - o^{(t)}) \sum_{n=1}^{t} \frac{\partial o^{(n)}}{\partial w} \tag{1.32}$$

TD pravidlo hovorí, že pre úpravu ľubovoľnej váhy v sieti, keď má na vstupe pozíciu $\bar{x}^{(t)}$ platí

TD PRAVIDLO

$$\Delta w^{(t)} = \alpha \ (o^{(t+1)} - o^{(t)}) \sum_{n=1}^{t} \frac{\partial o^{(n)}}{\partial w} \tag{1.33}$$

TD(λ) je modifikáciou predchádzajúceho vzťahu tak, že

TD(λ) PRAVIDLO

$$\Delta w^{(t)} = \alpha \ (o^{(t+1)} - o^{(t)}) \sum_{n=1}^{t} \lambda^{t-n} \frac{\partial o^{(n)}}{\partial w} \tag{1.34}$$

kde $0 < \lambda \leq 1$ je konštanta, ktorej hodnota sa nastavuje experimentovaním. Pre proces učenia a na výber počtu skrytých neurónov platí všetko, čo sme povedali pre UNS trénované na aproximáciu funkcií a klasifikáciu objektov.

Pomocu TD(λ) sa dá hrať backgammon na veľmajstrovskej úrovni (Tesauro, 1990) i dáma (Bahna, 1998). Existuje aj program na šach založený na kombinácii TD(λ) a algoritmu MINIMAX (Baxter et al., 1997). Učenie s odmenou a trestom sa často aplikuje na riadenie robotov, pretože pri trénovaní im netreba presne hovoriť, čo majú robiť, ale len sa známkujú sekvencie ich akcií.

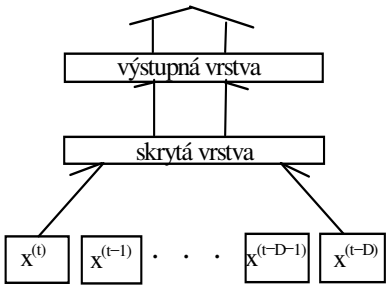
6.4 REKURENTNÉ SIETE A ICH UČENIE

ČASOVÁ ŠTRUKTÚRA DÁT

Predstavme si, že trénovacia množina by obsahovala nasledujúce dvojice: $a \rightarrow \alpha$, $b \rightarrow \beta$, $b \rightarrow \alpha$, $b \rightarrow \gamma$, $c \rightarrow \alpha$, $c \rightarrow \gamma$, $d \rightarrow \alpha$, atď, kde jednotlivé znaky predstavujú vstupné a požadované výstupné vektory. K jednému vstupu môže prislúchať viacero výstupov, a to v závislosti na **časovom kontexte**. Inými slovami, o výstupe siete rozhoduje nielen momentálny vstup siete, ale aj doterajšia história predkladaných vzorov. Vrstvová sieť by mala byť rozšírená o možnosť reprezentovať časový kontext. Architektonicky najjednoduchšie riešenie ponúka **neurónová sieť s oknom do minulosti** (angl. *time delay neural network*, TDNN) (obr. 6.15, vrstvy neurónov sú reprezentované obdĺžnikmi). Okno do minulosti má konečnú nemennú dĺžku D.

OBR.6.15.

NS S OKNOM DO MINULOSTI



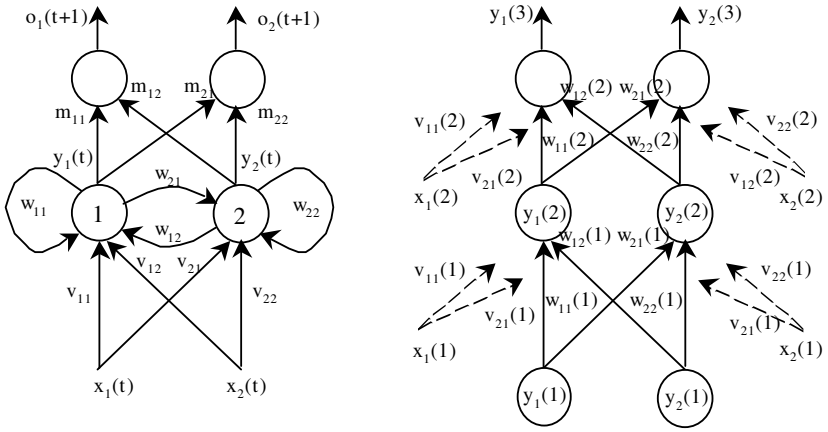
TDNN umožňuje viacvrstvovej doprednej sieti „okno do minulosti“, t.j. okrem momentálneho vstupu (v čase t) „vidí“ sieť ešte aj vstupy z minulých *D* krokov (v časoch *t*–1, *t*–2, ..., *t*–*D*). Takúto sieť je možné trénovať klasickou procedúrou spätného šírenia chýb, pričom je dôležité *zachovať poradie trénovacích vzoriek* v trénovacej množine.

Ak máme šťastie, aj takéto jednoduché rozšírenie viacvrstvovej architektúry môže postihnúť časovú štruktúru skrytú v trénovacích dátach. Výhodou architektúry TDNN je možnosť trénovania klasickou procedúrou spätného šírenia chýb. Nevýhodou tejto architektúry je, že spôsob reprezentácie časového kontextu pomocou okna do minulosti konečnej nemennej dĺžky nemusí byť dostatočne silný na zvládnutie časovej štruktúry trénovacích dát.

Treba podotknúť, že aj v prípade, keď TDNN je schopná reprezentovať časovo-priestorovú štruktúru dát, nie je jednoduché len na základe trénovacej množiny správne odhadnúť dĺžku *D* okna do minulosti. Napriek tomu architektúra TDNN našla uplatnenie v mnohých oblastiach pracujúcich s časovo-priestorovými štruktúrami, napríklad v robotike, rozpoznávaní reči, atď. (Sejnowski a Rosenberg 1987; Weibel, 1989).

SPÄTNÉ ŠÍRENIE CHÝB V ČASE

Na to, aby sme mohli rozšírili svoje možnosti pri spracovaní časových postupností dát na okno do minulosti premenlivej konečnej dĺžky, musíme zaviesť novú architektúru NS a nový pohľad na jej trénovanie. Novou architektúrou je tzv. **rekurentná neurónová sieť** (RNS) (pozri nasledujúci obrázok vľavo). Novým algoritmom trénovania RNS je **spätné šírenie chýb v čase** (angl. *back-propagation through time*, BPTT). Úlohou je natrénovať RNS na klasifikáciu postupností, či patria alebo nepatria do danej množiny postupností. Príkladom môže byť postupnosť signálov z nejakého snímača, ktorá vedie alebo nevedie k poruche, ktorej treba zabrániť. Požadovaný výstup sa sieti poskytne až na konci každej postupnosti (napríklad zapni-nezapni alarm). Jednotlivé postupnosti môžu mať premenlivú dĺžku *T*.



Pri BPTT sa RNS rozvíja v čase, t.j. má toľko skrytých vrstiev koľko je vstupov T v jednej postupnosti. Obrázok vpravo ukazuje rozvinutú sieť pre $T = 2$. Nech prvý vstup príde v čase $t = 1$ a posledný v čase $t = T$. Aktivity skrytých neurónov sa na začiatku každej postupnosti v čase $t = 1$ nastavia na hodnotu 0,5. Takáto rozvinutá RNS sa potom trénuje ako obyčajná dopredná UNS s T skrytými vrstvami (Rumelhart et al., 1986). V čase T+1 poznáme požadovaný výstup $d(T+1) \in \{(1,0),(0,1)\}$, t.j. postupnosť patrí do danej množiny alebo nepatrí. Pri výpočte skutočného výstupu siete používame vzorce

$$o_k^{(T+1)} = f\left(\sum_{j=1}^{J} m_{kj}^{(T+1)} y_j^{(T+1)}\right) \quad \text{kde} \quad y_j^{(t+1)} = f\left(\sum_{i=1}^{J} w_{ji}^{(t)} y_i^{(t)} + \sum_{i=1}^{I} v_{ji}^{(t)} x_i^{(t)}\right)$$

Váhy siete upravujeme až v kroku T+1 a to tak, aby sme minimalizovali chybovú funkciu

$$E(T+1) = \tfrac{1}{2}\sum_{k=1}^{K} (d_k^{(T+1)} - o_k^{(T+1)})^2$$

Teda, váhy medzi skrytou a výstupnou vrstvou sa menia podľa vzťahov

$$\Delta m_{kj}^{(T+1)} = -\alpha \frac{\partial E(T+1)}{\partial m_{kj}} = \alpha \delta_k^{(T+1)} y_j^{(T+1)} \quad \text{kde} \quad \delta_k^{(T+1)} = \left(d_k^{(T+1)} - o_k^{(T+1)}\right) f'(net_k^{(T+1)})$$

Váhy medzi jednotlivými rozvinutými skrytými vrstvami a medzi vstupom a rozvinutými skrytými vrstvami sa upravujú nasledovne:

$$\Delta w_{hj}^{(T)} = -\alpha \frac{\partial E(T+1)}{\partial w_{hj}} = \alpha \delta_h^{(T)} y_j^{(T)} \quad \text{kde} \quad \delta_h^{(T)} = \left(\sum_{k=1}^{K} \delta_k^{(T+1)} m_{kh}^{(T+1)} \right) f'(net_h^{(T)})$$

$$\Delta v_{ji}^{(T)} = -\alpha \frac{\partial E(T+1)}{\partial v_{ji}} = \alpha \delta_j^{(T)} x_i^{(T)} \quad \text{kde} \quad \delta_j^{(T)} = \left(\sum_{k=1}^{K} \delta_k^{(T+1)} m_{kj}^{(T+1)} \right) f'(net_j^{(T)})$$

$$\Delta w_{hj}^{(T-1)} = -\alpha \frac{\partial E(T+1)}{\partial w_{hj}} = \alpha \delta_h^{(T-1)} y_j^{(T-1)} \quad \text{kde} \quad \delta_h^{(T-1)} = \left(\sum_{j=1}^{J} \delta_j^{(T)} w_{jh}^{(T)} \right) f'(net_h^{(T-1)})$$

$$\Delta v_{ji}^{(T-1)} = -\alpha \frac{\partial E(T+1)}{\partial v_{ji}} = \alpha \delta_j^{(T-1)} x_i^{(T-1)} \quad \text{kde} \quad \delta_j^{(T-1)} = \left(\sum_{h=1}^{J} \delta_h^{(T)} w_{hj}^{(T)} \right) f'(net_j^{(T-1)})$$

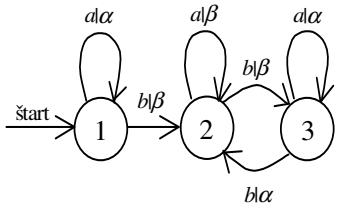
atď. Výsledné zmeny váh na konci postupnosti sú rovné

$$\Delta w_{hj}^{(T+1)} = \frac{\sum_{t=1}^{T} \Delta w_{hj}^{(t)}}{T} \quad \text{a} \quad \Delta w_{ji}^{(T+1)} = \frac{\sum_{t=1}^{T} \Delta w_{ji}^{(t)}}{T}$$

Pre každú novú postupnosť s inou dĺžkou T, začíname proces rozvíjania a trénovania siete odznova. Neuróny môžu mať aj adaptovateľné prahy.

OBR.6.16.

MEALYHO AUTOMAT

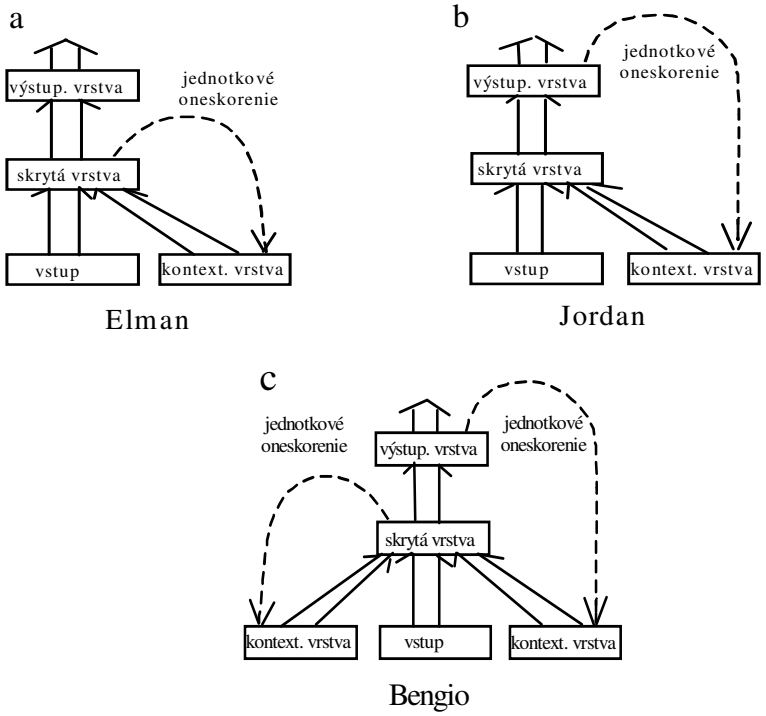


Príkladom, kedy nemusí ani TDNN ani BPTT stačiť na postihnutie časovej štruktúry dát sú postupnosti generované automatmi (obr. 6.16). Stavy automatu kódujú históriu vstupných vektorov tak, aby sme mohli vždy bez váhania odpovedať aký bude asociovaný výstup k danému vstupu pri danej histórii predkladaných vstupov.

STAVOVÁ REPREZENTÁCIA

Takáto **stavová reprezentácia časového kontextu** predkladaných vzoriek môže byť omnoho úspornejšia ako reprezentácia časového kontextu pomocou okna do minulosti a niekedy aj nevyhnutná. Môže sa totiž stať, že by sme potrebovali neobmedzene dlhé okno do minulosti. Ak by sme boli v stave 1 automatu na obr. 6.15, môže prísť ľubovoľný počet vstupov a a asociovaný výstup je α. To isté patrí aj o stave 3. Podstatný rozdiel je však vo výstupe asociovanom so vstupom b. Ten je β, v prípade stavu 1 a α v prípade stavu 3. Nie je možné zvoliť žiadne konečné D alebo T tak, aby za každých okolností bolo možné na základe minulých vstupov rozhodnúť o výstupe asociovanom k vstupu b. Zrejme pre úplné zovšeobecnenie trénovacej množiny reprezentujúcej časovo-priestorovú štruktúru popísanú automatom na obr. 6.16 bude architektúra TDNN nevyhovujúca, takisto ako aj RNS trénovaná pomocou BPTT.

OBR.6.17. ARCHITEKTÚRY REKURENTNÝCH SIETÍ



REKURENTNÉ NEURÓNOVÉ SIETE

Ukázalo sa, že isté typy RNS (obr. 6.17) sú schopné vytvoriť si stavovú reprezentáciu časového kontextu v dátach (Tiňo and Šajda, 1995). Vo všeobecnosti možno za rekurentnú sieť považovať akúkoľvek NS, v ktorej si istá podmnožina neurónov (**rekurentné neuróny**) uchováva informáciu o svojich aktiváciách v predošlých časoch. Hodnoty, ktoré boli na výstupe rekurentných neurónov v čase t+1 sa prekopírujú na **kontextové neuróny** a pripoja sa k vstupnému vektoru v čase t (pozri obr. 6.17). Toto kopírovanie sa deje s tzv. jednotkovým oneskorením. neurónové siete sa takto rozširuje o *vnútornú pamäť*.

Podobne ako v tradičných dopredných neurónových sieťach, v rámci jednej vrstvy nie sú neuróny navzájom prepojené. Dvojité šípky reprezentujú spojenia z každého neurónu spodnej vrstvy do každého neurónu hornej vrstvy. Tieto spojenia majú váhy, ktoré sa modifikujú počas trénovacieho procesu. Jednoduché šípky predstavujú **rekurentné spojenia** medzi zodpovedajúcimi neurónmi východiskovej a cieľovej vrstvy. Rekurentné spojenia majú nemennú váhu 1. Existujú len medzi i-tym neurónom rekurentnej a i-tym neurónom kontextovej vrstvy. Funkcia rekurentných spojení je odpamätanie aktivácií rekurentných neurónov a ich zavedenie do kontextových neurónov s jednotkovým časovým oneskorením. Architektúru na obr. 6.17a navrhol Elman (1990). Kontextová vrstva obsahuje kópie aktivácií skrytých neurónov z predošlého kroku. Autorom siete 6.17b je Jordan (1989). Obr. 6.17c predstavuje kombináciu predchádzajúcich modelov (Bengio et al., 1990).

REKURENTNÉ UČENIE V REÁLNOM ČASE

RNS sa trénujú na predikciu nasledujúceho požadovaného vektora pomocou metódy známej ako **rekurentné učenie v reálnom čase** (angl. *real time recurrent learning*, RTRL) (Williams and Zipser, 1989). Uvedieme hlavné rovnice tohto postupu, a to pre Elmanovu rekurentnú sieť ilustrovanú na obr. 6.17a. Táto architektúra sa volá jednoduchá rekurentná sieť (angl. *simple recurrent network*, SRN).

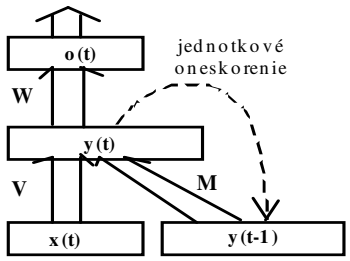
V čase *t* je na vstupe vektor $\bar{x}^{(t)} = (x_1^{(t)}, x_2^{(t)}, ..., x_I^{(t)})$, skutočný výstup siete je vektor aktivácií výstupných neurónov $\bar{o}^{(t)} = (o_1^{(t)}, o_2^{(t)}, ..., o_K^{(t)})$ a požadovaný výstup je $\bar{d}^{(t)} = (d_1^{(t)}, d_2^{(t)}, ..., d_K^{(t)})$. Definujeme chybovú funkciu ako

$$E^{(t)} = \frac{1}{2}\sum_{k=1}^{K}(d_k^{(t)} - o_k^{(t)}) \tag{6.35}$$

Jednotlivé váhy upravujeme „on-line“, t.j. v každom kroku *t*, proporcionálne k negatívnemu gradientu $E^{(t)}$:

$$\Delta w_{kj}^{(t)} = -\alpha \frac{\partial E^{(t)}}{\partial w_{kj}^{(t)}} \qquad \Delta v_{ji}^{(t)} = -\alpha \frac{\partial E^{(t)}}{\partial v_{ji}^{(t)}} \qquad \Delta m_{jl}^{(t)} = -\alpha \frac{\partial E^{(t)}}{\partial m_{jl}^{(t)}} \tag{6.36}$$

Pri výpočte týchto derivácií treba brať do úvahy časové vzťahy vo vnútri siete (Kvasnička et al., 1997). Ako vidíme, podstata učenia je opäť spätné šírenie chýb, v ktorom sa teraz odráža nielen priestorová štruktúra dát, ale aj časová.



Elman

Výstupy neurónov sú

$$o_k^{(t)} = f\left(net_k^{(t)}\right) = f\left(\sum_{j=1}^{J} w_{kj}\, y_j^{(t)}\right) \text{ a}$$

$$y_j^{(t)} = f\left(net_j^{(t)}\right), \text{ kde}$$

$$net_j^{(t)} = \sum_{i=1}^{I} v_{ji} x_i^{(t)} + \sum_{i=1}^{J} m_{ji}\, y_i^{(t-1)} .$$

Váhy medzi skrytou (rekurentnou) a výstupnou vrstvou sa upravujú ako pre doprednú NS (rovnica 6.13), teda

$$\Delta w_{kj}^{(t)} = \alpha \delta_k^{(t)}\, y_j^{(t)} = \alpha\, (d_k^{(t)} - o_k^{(t)})\; f_k'(net_k^{(t)})\; y_j^{(t)}$$

Pre časovú zmenu váh medzi vstupom a skrytou (rekurentnou) vrstvou, a medzi kontextovou a rekurentnou vrstvou platí

$$\Delta v_{ji}^{(t)} = \alpha \sum_{k=1}^{K}\left[\delta_k^{(t)} \sum_{h=1}^{J} w_{kh} \frac{\partial y_h^{(t)}}{\partial v_{ji}}\right] \quad \text{resp.} \quad \Delta m_{ji}^{(t)} = \alpha \sum_{k=1}^{K}\left[\delta_k^{(t)} \sum_{h=1}^{J} w_{kh} \frac{\partial y_h^{(t)}}{\partial m_{ji}}\right]$$

kde

$$\frac{\partial y_h^{(t)}}{\partial v_{ji}} = f'(net_h^{(t)})\left[x_i^{(t)}\delta_{jh}^{Kron.} + \sum_{l=1}^{J} m_{hl}\frac{\partial y_l^{(t-1)}}{\partial v_{ji}} \right]$$

$$\frac{\partial y_h^{(t)}}{\partial m_{ji}} = f'(net_h^{(t)})\left[x_i^{(t)}\delta_{jh}^{Kron.} + \sum_{l=1}^{J} m_{hl}\frac{\partial y_l^{(t-1)}}{\partial m_{ji}} \right]$$

kde $\delta_{jh}^{Kron.}$ je Kroneckerova delta, pre ktorú platí $\delta_{jh}^{Kron.} = 1$, ak $j=h$, a inak, $\delta_{jh}^{Kron.} \neq 1$. Počas trénovania je užitočné sledovať veľkosť šírených chýb, pretože tieto smerom do minulosti klesajú k nule. Vtedy treba ich šírenie zastaviť.

Rekurentné neurónové siete (RNS) sa používajú na tri typy úloh:

KLASIFIKÁCIA S ČASOVÝM KONTEXTOM

I. RNS má rozhodnúť, či práve ukončená postupnosť vstupov patrí do nejakej triedy, alebo nie, poprípade do ktorej z možných tried patrí. Sem možno zaradiť napríklad klasifikáciu postupností symbolov z nejakej konečnej abecedy. Požadovanými výstupmi by boli výstupné symboly automatu. Ďalším príkladom môže byť, či postupnosť nejakých signálov vedie k poruche zariadenia, alebo nie, respektíve k akej poruche. Požadovanými výstupmi pri trénovaní by boli zakódované indikátory stavu zariadenia.

PREDIKCIA

II. Úlohou je na základe časovej štruktúry v postupnosti dát pred časom t predpovedať dáta po čase t. Napríklad pri trénovaní na predpoveď (predikciu) nasledujúceho člena postupnosti je trénovacia množina $A_{train} = \{(\bar{x}^{(t)}, \bar{x}^{(t+1)})\}_{t=0}^{T}$. Vo všeobecnosti, na predikciu údaju v čase $t+\tau$ je $A_{train} = \{(\bar{x}^{(t)}, \bar{x}^{(t+\tau)})\}_{t=0}^{T}$. Na trénovanie predikcie na netriviálnych postupnostiach potrebujeme postupnosti, ktoré majú tisícky až desaťtisícky údajov. Tiež sa oplatí nepracovať priamo s reálnymi hodnotami, ale zakódovať ich na symboly, ktoré odrážajú relatívne kvantitatívne zmeny nahor a nadol.

GENEROVANIE POSTUPNOSTI

III. Tretí typ úloh je komplikovanejšia verzia predikčných úloh. Tentoraz nejde len o predikovanie hodnoty dát v niektorom budúcom čase. Na základe pozorovania určitého úseku vývoja dát je úlohou *pokračovať* v časovom rade dát zohľadňujúc základnú tendenciu dát skrytú v dostupnom úseku. Napríklad, ak by sme pozorovali úsek dát: 23123123123123123... zrejmé pokračovanie by bolo 123123123... V reálnych úlohách však časová štruktúra dát môže byť omnoho zložitejšia než prísna periodicita časového radu. Samotné generovanie pokračovania úseku časovej rady sa môže realizovať napríklad týmto spôsobom: Po predložení dostupného úseku dát (do času t), sieť vygeneruje predikciu možnej hodnoty dát v nasledujúcom čase t+1. Táto predikcia sa priradí k pôvodnému úseku a na základe takto vytvoreného nového úseku dát vygenerujeme predikciu pre čas t+2, atď.

RNS a iteračné funkčné systémy

V tejto časti si vysvetlíme, ako je možné, že RNS sú schopné využívať históriu vstupov na predikciu budúceho vývoja postupnosti. Sústredíme sa na analýzu aktivít rekurentných neurónov, napríklad v architektúre RNS Elmanovho typu (obr. 6.17a). Vektor aktivít rekurentných neurónov si môžeme predstaviť ako bod v J-rozmernom priestore, tzv. stavovom priestore RNS. Ukážeme si, že RNS je schopná previesť časovú štruktúru v dátach na priestorovú fraktálovú štruktúru vo svojom stavovom priestore. Na to využijeme tzv. teóriu funkčných systémov (angl. *iterated function systems*, IFS) (Barnsley 1988).

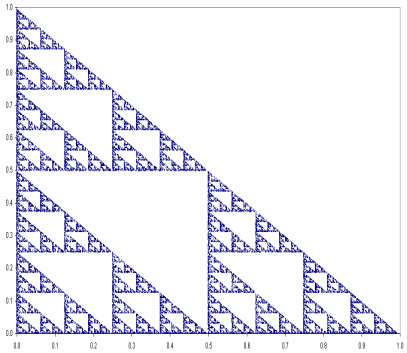
IFS

Definícia: **Iteračný funkčný systém** (IFS) je konečná množina kontraktívnych transformácií $\Omega = \{\omega_i \mid \omega_i : X \to X, i \le n\}$.

Nech $X = \langle 0,1 \rangle$. Majme jednu jedinú kontraktívnu transformáciu ω, napr. $\omega(x) = 0{,}5x + 0{,}5$. Limitné správanie jednej kontraktívnej transformácie, teda jej aplikácia $n \to \infty$ krát na ľubovoľný počiatočný bod, vedie k jednému jedinému bodu v X, ktorý nazývame atraktor typu fixný bod. (V našom malom príklade by to bol bod 1.) Limitná množina bodov zjednotenia viacerých rôznych transformácií môže predstavovať zložitú priestorovú štruktúru. Limitná množina zloženého IFS sa nazýva **IFS atraktor**. Pre každý bod IFS atraktora definujeme tzv. IFS adresu. Táto adresa zodpovedá nekonečnej postupnosti transformácií, ktorá nás k nemu doviedla.

IFS ATRAKTOR

Sierpinského trojuholník

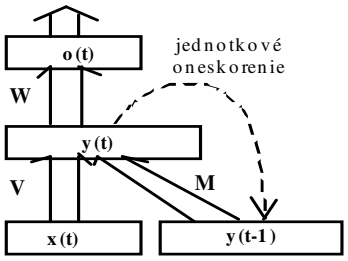


Príkladom IFS je sústava týchto troch transformácií v priestore $X = \langle 0,1 \rangle^2$:

$$\omega_1(x, y) = (0.5x + 0.5, 0.5y)$$
$$\omega_2(x, y) = (0.5x, 0.5y + 0.5)$$
$$\omega_3(x, y) = (0.5x, 0.5y)$$

Limitná množina každej tejto transformácie aplikovanej jednotlivo je bod v rohu stavového priestoru X (t.j. $\omega_1 \to (1,0)$, $\omega_2 \to (0,1)$, $\omega_3 \to (0,0)$). Limitná množina kompozície všetkých troch zobrazení je **samopodobná štruktúra – fraktál** známy ako Sierpinského trojuholník.

RNS AKO IFS



Elman

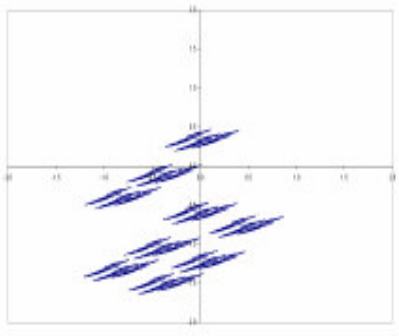
Správanie RNS pri spracovaní postupnosti symbolov sa dá vysvetliť pomocou IFS. Vnútorná dynamika (= časový vývoj aktivít rekurentných neurónov) sa riadia podľa rovnice

$$\bar{y}^{(t)} = f(\mathbf{M} \cdot \bar{y}^{(t-1)} + \mathbf{V} \cdot \bar{x}^{(t)})$$

Vstupný vektor je $\bar{x}$, stavový vektor je $\bar{y}$. **M** a **V** sú matice váh a f je aktivačná funkcia. Výstupná vrstva nás zatiaľ nezaujíma.

Majme konečnú vstupnú abecedu, napr. A = {a,b,c}. Každý symbol je zakódovaný pomocou „one-hot" kódovania, teda a = (1,0,0), b = (0,1,0), c = (0,0,1). Stavovú rovnicu RNS môžme prepísať ako $\bar{y}^{(t)} = f(\mathbf{M}_x \cdot \bar{y}^{(t-1)})$, kde matica váh $\mathbf{M}_x$ je transformačná matica aplikovaná na predchádzajúci stav siete po príchode konkrétneho symbolu $\bar{x}$. Inými slovami, IFS transformácie sú reprezentované váhovými maticami $\mathbf{M}_x$ (x = a,b,c) a špecifický vstupný vektor vyberie, ktorá transformácia sa aplikuje na predchádzajúci stav. Vstupný symbol predstavuje index transformácie.

FRAKTÁL



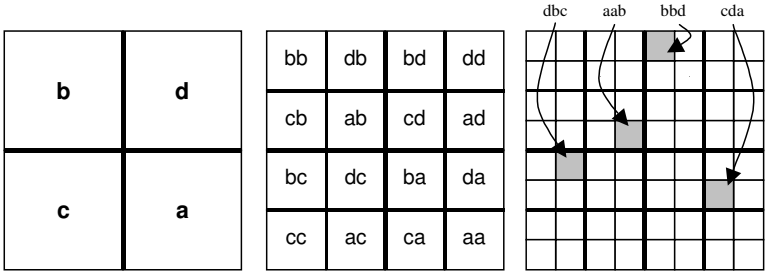
Z povedaného vyplýva, že aj nenatrénované RNS inicializované malými náhodnými váhami sa správajú ako IFS (Kolen 1994). Tento jav sa nazýva architekturálny bias RNS (Tiňo et al. 2002). Môžeme očakávať, že v stavovom priestore sa vytvorí zaujímavá priestorová štruktúra odrážajúca časovú štruktúru konkrétnej postupnosti symbolov, aj keď je sieť nenaučená (pozri obrázok vľavo, prevzatý z Čerňanský 2001).

Vráťme sa teraz k pojmu IFS adresy. Uvažujme nasledujúcu sústavu IFS transformácií nad priestorom $X = \langle 0,1 \rangle^2$

$$\omega_a(x,y) = (0.5x + 0.5, 0.5y)$$
$$\omega_b(x,y) = (0.5x, 0.5y + 0.5)$$
$$\omega_c(x,y) = (0.5x, 0.5y)$$
$$\omega_d(x,y) = (0.5x + 0.5, 0.5y + 0.5)$$

Atraktorom štvrtej transformácie je bod (1,1). Každá z týchto transformácií kontrahuje priestor X na štvrtinovú kópiu originálu vo svojej štvrtine štvorca (pozri nasledujúci obrázok prevzatý z Čerňanský 2001).

IFS ADRESA



Ak by sme mali nekonečnú presnosť, pozícia každého bodu v stavovom priestore by bola jednoznačne určená postupnosťou transformácií, ktorá k nemu viedla (v opačnom poradí smerom do minulosti). IFS adresa bodu v IFS atraktore je nekonečná postupnosť indexov transformácií, ktoré mapujú priestor X práve do tohto bodu. Prvý index IFS adresy zodpovedá poslednému symbolu, ktorý prišiel na vstup siete, druhý index predposlednému symbolu, atď.

HISTÓRIA SYMBOLOV

Podobné podpostupnosti trénovacej (resp. testovacej postupnosti) symbolov vedú k priestorovo blízkym bodom v IFS atraktore. Čím je dlhší spoločný sufix, tým budú ich body v stavovom priestore bližšie. Často sa vyskytujúce podpostupnosti so spoločnými sufixami v stavovom priestore vytvárajú klastre. Klastrovaním stavového priestoru RNS dostávame klastre, v ktorých sú body odrážajúce podobnú históriu symbolov. Podobná história symbolov (časový kontext) vedie k podobnému pokračovaniu postupnosti. Táto skutočnosť sa využíva na budovanie predikčných modelov. Keďže však nemáme k dispozícii nekonečnú presnosť, môžeme budovať iba predikčné modely s konečnou pamäťou.

TRÉNOVANIE RNS

Na tomto mieste právom vzniká otázka, aký zmysel má potom trénovanie RNS, keď sa vďaka architekturálnemu biasu časová štruktúra trénovacej postupnosti premietne do priestorovej štruktúry IFS atraktora aj v nenaučenej sieti. Po prvé, trénovaním sa výstupná vrstva učí asociovať históriu symbolov (klaster blízkych bodov v stavovom priestore) s nasledujúcim symbolom (ak používame RTRL). Po druhé, čo je dôležitejšie, učením sa lepšie zorganizuje stavový priestor, resp. body v IFS atraktore, čím sa umožní lepšia predikcia pokračovania postupnosti dát v budúcnosti. V prípade trénovania RNS na slovách generovaných konečno-stavovými automatmi sa zistilo, že v stavovom priestore RNS z pôvodného veľkého počtu klastrov vznikne práve toľko klastrov, koľko je stavov automatu (Tiňo and Šajda 1995). Keďže je ľahšie spracovávať symbolové postupnosti, reálne postupnosti dát sa obvykle kvantizujú na postupnosti niekoľkých symbolov, ktoré vyjadrujú relatívny pokles alebo nárast hodnoty sledovanej veličiny.

6.5 RBF SIETE A ICH UČENIE

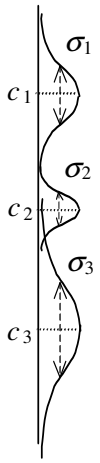
RADIÁLNA BÁZOVÁ FUNKCIA

V tejto časti si predstavíme tzv. RBF siete, ktoré majú všetky výpočtové schopnosti klasických dopredných vrstvových UNS trénovaných s učiteľom, ale ich trénovanie je podstatne rýchlejšie (Bishop, 1995). RBF siete sú také, v ktorých je aktivačnou funkciou skrytých neurónov tzv. **radiálna bázová funkcia** (angl. *radial basis function*, **RBF**) (Bishop, 1995; Oravec et al., 1998). Najčastejšie sa používa gaussovská funkcia. Výstup *n*-tého výstupného neurónu je lineárny, t.j.

$$o_n = \sum_{j=1}^{k} w_{nj}\phi_j \tag{6.35}$$

OBR.6.18.

ILUSTRÁCIA RBF NA SKRYTEJ VRSTVE



Čiže celkovo máme k skrytých neurónov. Výstup j-teho skrytého neurónu je gaussovská funkcia

$$\phi_j(\bar{x}) = \exp\left(-\frac{\left\|\bar{x}-\bar{c}_j\right\|^2}{\sigma_j^2}\right) \tag{6.36}$$

kde σ_j je šírka gaussiánu prislúchajúceho j-temu neurónu, $\left\|\bar{x}-\bar{c}_j\right\|$ vyjadruje euklidovskú vzdialenosť medzi vstupným vektorom $\bar{x}$ a centrom gaussiánu $\bar{c}_j$. Obr. 6.18 ilustruje tri gaussiány s rôznymi centrami a šírkami. Trénovanie RBF sietí prebieha v troch krokoch:

I. Určenie centier $\bar{c}_j$ skrytých neurónov (RBF prvkov). Je dôležité, aby centrá vystihovali štruktúru vstupných dát. Určujeme ich na základe všetkých $\bar{x}^p \in A_{train}$, pričom $A_{train} = \{(\bar{x}^1,\bar{d}^1)(\bar{x}^2,\bar{d}^2)...(\bar{x}^p,\bar{d}^p)....(\bar{x}^P,\bar{d}^P)\}$. Používajú sa na to klastrovacie metódy, napr. *k*-priemerovací klastrovací algoritmus (angl. *k-means clustering algorithm*), ktorý hľadá minimum celkovej sumy vzdialeností medzi centrami $\bar{c}_j$ a bodmi $\bar{x}^p$, ktoré patria do ich klastrov. Samotný algoritmus je takýto:

1) Najprv zvolíme *k*, t.j. počet centier a zároveň i počet RBF prvkov. Zvyčajne *k* = 40 ÷ 60. Optimálny počet sa musí nájsť experimentovaním.

2) Náhodne vyberieme *k* vstupných vektorov $\bar{x}^p \in A_{train}$, ktoré budú našimi štartovacími centrami.

3) V čase *t* urobíme toto:

a) Nájdeme centrum $\bar{c}_j(t)$, ktoré je najbližšie k vstupu $\bar{x}^p(t)$.

b) Posunieme centrum $\bar{c}_j(t)$ ku $\bar{x}^p(t)$ podľa vzťahu: $\bar{c}_j(t+1) = \bar{c}_j(t) + \rho(t)\left\|\bar{x}^p(t) - \bar{c}_j(t)\right\|$, kde $0 \leq \rho(t) \leq 1$. Zvyčajne má na začiatku veľkú hodnotu a postupne klesá k nule. Alebo môžeme hľadanie centier zastaviť tak, že v nasledujúcich krokoch po sebe sa ich poloha mení menej ako nejaké malé $\varepsilon = 10^{-3}$. Cez A_{train} prechádzame dovtedy, pokiaľ nie je táto podmienka splnená.

II. Určenie šírok RBF funkcií. Šírky σ_j funkcií ϕ_j ovplyvňujú generalizačné schopnosti siete. Čím sú menšie, tým horšie zovšeobecňovanie sa dá očakávať. Avšak ani príliš veľký prekryv medzi susednými gaussiánmi nie je dobrý. Môžeme ich nastaviť pre všetky RBF prvky rovnaké. Alebo ich môžeme vypočítať podľa: $\sigma_j = \sqrt{(1/L)\sum_{l=1}^{L}\left\|\bar{c}_j - \bar{c}_l\right\|^2}$, kde L je počet najbližších susedov.

III. Učenie váh výstupných neurónov. Tieto váhy sa učia pomocou spätného šírenia chyby, teda $\Delta w_{nj} = -\alpha(\partial E / \partial w_{nj})$, kde E je chybová funkcia (6.10). Váhy meníme až po nájdení $\bar{c}_j$ a σ_j.

VÝHODY RBF SIETÍ

Dopredné vrstvové RBF siete dokážu rovnako dobre vykonávať všetky typy úloh ako neurónové siete, v ktorých majú neuróny sigmoidálnu aktivačnú funkciu. Sú tiež univerzálnymi aproximátormi funkcií. Centrá a rozptyly radiálnych bázových funkcií sa určujú principiálne odlišne ako váhové koeficienty výstupných neurónov. Trénovanie RBF sietí je rýchlejšie ako pri klasických sigmoidálnych dopredných neurónových sieťach a nie je citlivé na poradie vstupných vzorov. RBF prvky sa dajú použiť aj v rekurentných neurónových sieťach.

6.6 SOM – SAMOORGANIZUJÚCA SA MAPA

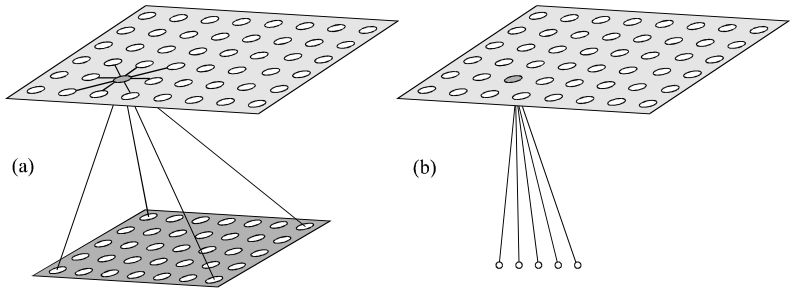
SamoOrganizujúca sa Mapa (SOM) je názov neurónovej siete, v ktorej prebieha **učenie bez učiteľa** (angl *unsupervised learning*). To znamená, že jej algoritmus učenia nemá informáciu o požadovaných aktivitách výstupných neurónov v priebehu trénovania (Kohonen 1995). Adaptácia váh prebieha len na základe trénovacej množiny, ktorá je sieti prezentovaná vo forme vstupných vzorov (vektorov), s binárnymi alebo reálnymi zložkami. Architektúra[8], ale hlavne spôsob učenia SOM sú v súlade s neurobiologickými poznatkami týkajúcimi sa mozgovej kôry živočíchov.

[8] Aj predchádzajúce architektúry NS sa nachádzajú v nervovom systéme živočíchov.

Na obr. 6.19a je znázornená jedna z principiálnych organizácií biologických NS. Spodná mriežka, tzv. vrstva receptorov, reprezentuje vstup. Každá zložka vstupu, každý receptor, vysiela dopredné spojenia na všetky neuróny, ktoré sú usporiadané len v jednej vrstve, ktorá reprezentuje napríklad mozgovú kôru. Aktivity neurónov predstavujú výstup siete. Neuróny sú navzájom pospájané tzv. laterálnymi spojeniami, ktoré môžu byť buď excitačné a/alebo inhibičné. Vpravo, obr. 6.19b, je ešte viac zjednodušená architektúra, ktorá sa používa vo výpočtoch. Receptorová vrstva je nahradená vstupným vektorom a laterálne väzby neurónov tzv. funkciou okolia neuróna, súťažením a kooperáciou neurónov, ako uvidíme ďalej. (Obrázok je prevzatý z Farkaš 1997).

OBR.6.19.

ARCHITEKTÚRA SOM



ZACHOVANIE TOPOLÓGIE VSTUPU

Špecifickou črtou SOM je to, že po natrénovaní umožňuje realizovať ***zobrazenie zachovávajúce topológiu*** trénovacej množiny dát. To znamená, že ľubovoľné dva vzory blízke vo vstupnom priestore evokujú v sieti odozvy na neurónoch, ktoré sú tiež fyzicky blízke (vo výstupnom priestore). Vzdialenosť dvoch neurónov sa rovná ich euklidovskej vzdialenosti na mriežke.

Fenomén topologického zobrazenia príznakov (angl. *feature mapping)* má výrazné zastúpenie v biologických neurónových sieťach, konkrétne v mozgoch vyšších cicavcov i človeka (Kohonen 1995, Maršala 1985). Existencia *topografických máp* bola zistená v rôznych častiach mozgu, hlavne v mozgovej kôre. Jedná sa o mapy, ktoré buď priamo reprezentujú senzorický (zmyslový) priestor (napr. mapa povrchu tela), alebo reprezentujú príznaky vstupu, ktoré sú nejakým spôsobom vypočítavané. Ako príklady možno uviesť vizuálne mapy (napr. mapa orientácií svetelných kontrastov) a sluchové mapy (napr. mapa frekvencií a amplitúd akustických stimulov). Mozgová kôra je priestorovo organizovaná a je pre ňu charakteristická *lokálnosť odoziev* na vstupné podnety. Topografické mapy nie sú pri narodení úplne vyvinuté, ale formujú sa v počiatočných štádiách vývoja v dôsledku zmyslovej skúsenosti. Hoci usporiadanie neurónových sietí mozgu a ich funkcie sú dané geneticky, je tu priestor pre *modifikovateľnosť* týchto štruktúr v dôsledku aktuálnej skúsenosti. Mechanizmy tejto *plasticity* sú tiež pripravené geneticky, aby sa organizmus mohol prispôsobiť zmenému prostrediu. Proces modifikácie prebieha na základe podnetov prichádzajúcich z okolitého prostredia a je potrebný na vývoj normálnych topografických máp.

HEBBOVO PRAVIDLO UČENIA

Samoorganizácia, respektíve učenie v SOM je založené na tzv. Hebbovom pravidle učenia. V r. 1949 vyšla kniha „The Organization of Behavior“, jedna z najcitovanejších prác v oblasti NS (Hebb, 1949). **Hebb** postuloval predpoklad, že váhy spojení medzi neurónmi v mozgu sa permanentne menia ako sa jedinec adaptuje a učí nové veci, a to podľa takéhoto pravidla: *"When an axon of cell A ... excite(s) cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells so that A´s efficiency as one of the cells firing B is increased"* (Keď má axón bunky A excitačný účinok na bunku B, a opakovane alebo vytrvalo sa zúčastňuje na jej aktivácii, v jednej alebo v oboch bunkách prebehne nejaký rastový proces alebo metabolická zmena, takže účinnosť bunky A ako jednej z buniek, ktoré aktivujú B, vzrastie).

KOHONENOV ALGORITMUS UČENIA

Fínsky teoretik Teuvo **Kohonen** navrhol jednu z možných formalizácií tohoto pravidla, ktorá sa používa na trénovanie SOM. Preto sa SOM niekedy nazýva aj Kohonenova sieť. Váhy medzi vstupom a neurónmi v mriežke inicializujeme na malé náhodné čísla, napr. z intervalu (-0,5; 0,5). Majme trénovaciu množinu vzorov $A_{train} = \{\bar{x}_p\}_{p=1}^{P}$. Neuróny v sieti sú lineárne s nulovými prahmi, t.j.

$$o_i = \sum_{j=1}^{m} w_{ij} x_j = \bar{w}_i \; \bar{x} \tag{6.37}$$

pričom m je dimenzia vstupu. Nech $i = 1,\ldots, n$ je počet neurónov v mriežke. V náhodnom poradí dávame na vstup siete jednotlivé vzory. Pre každý vzor, nájdeme víťazný neurón. Zisťovanie pozície (indexu) víťazného neuróna sa nazýva **súťaženie**. Výsledkom súťaženia v každom kroku (po predložení konkrétneho vstupu) je **víťazný neurón**, ktorý najviac reaguje na daný vstup $\bar{x}$. Jednou možnosťou je hľadať maximum výstupu lineárneho neurónu, t.j. $i^* = \text{argmax}(\bar{w}_i \; \bar{x})$, kde i^* je index víťazného neurónu (DP, *dot product* verzia). V základnej, tzv. ED (angl. *Euclidean Distance*) verzii algoritmu SOM figuruje iná miera podobnosti: nákde sa neurón, ktorého váhový vektor je najbližšie k aktuálnemu vstupu v zmysle euklidovskej vzdialenosti:

NÁJDENIE VÍŤAZA

$$i^* = \underset{i}{\text{argmin}} \left\| \bar{x} - \bar{w} \right\| \tag{6.38}$$

Po nájdení víťaza nasleduje adaptácia váh – **učenie**. To zabezpečí, že váhové vektory víťazného neurónu a jeho topologických susedov sa posunú smerom k aktuálnemu vstupu podľa vzťahu

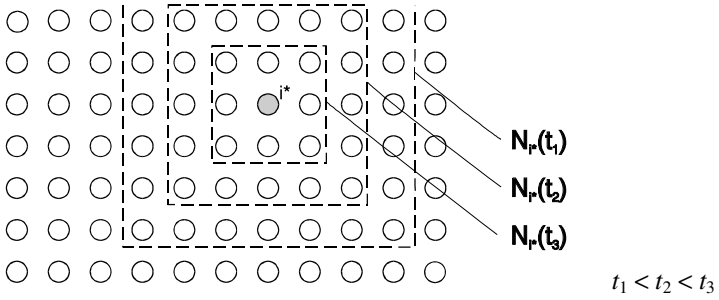
ADAPTÁCIA VÁH

$$\bar{w}_i(t+1) = \bar{w}_i(t) + \alpha(t) \cdot h(i^*, i) \cdot [\bar{x}(t) - \bar{w}_i(t)] \tag{6.39}$$

Funkcia $\alpha(t) \in (0,1)$ predstavuje premenlivú rýchlosť učenia, ktorá s časom klesá k nule (napr. podľa vzťahu $1/t$, resp. $\exp(-kt)$), čím sa zabezpečí ukončenie procesu učenia. Funkcia okolia $h(i^*,i)$ (obr. 6.20) definuje rozsah *kooperácie* medzi neurónmi, t.j. koľko váhových vektorov prislúchajúcich neurónom v okolí víťaza bude adaptovaných, a do akej miery.

OBR.6.20.

ZMENŠUJÚCE SA OKOLIE VÍŤAZNÉHO NEURÓNA



Najjednoduchšou používanou funkciou je pravouhlé okolie

$$h(i^*,i)=\begin{cases}1 & \text{ak } d_M(i^*,i)\le\lambda(t)\\ 0 & \text{inak}\end{cases} \tag{6.40}$$

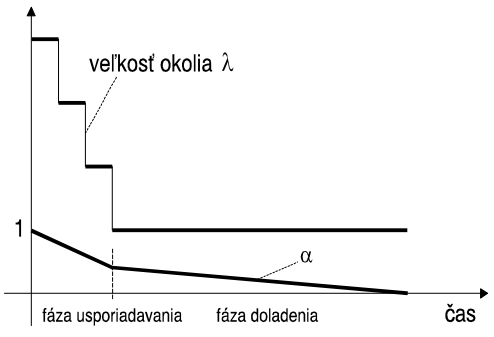
pričom $d_M(i^*,i)$ predstavuje vzdialenosť typu „Manhattan" medzi neurónmi i^* a i v mriežke mapy (t.j. sumu absolútnych hodnôt rozdielov ich súradníc). Na základe numerických simulácií dospel Kohonen k záveru, že najlepšie výsledky sa dosiahnu vtedy, ak sa veľkosť okolia s časom diskrétne zmenšuje (priemer okolia odpovedá hodnote $2\lambda(t)$) (obr. 6.20, prevzatý z Farkaš 1997). V blízkosti okrajov mapy okolie nie je symetrické (týka sa to najmä počiatočných fáz algoritmu, keď je polomer okolia veľký). Druhou často používanou voľbou je gaussovské okolie, ktoré možno popísať rovnicou

$$h(i^*,i)=\exp\left(-\frac{d_E^2(i^*,i)}{\lambda^2(t)}\right) \tag{6.41}$$

kde $d_E(i^*,i)$ predstavuje euklidovskú vzdialenosť neurónov i^* a i v mriežke, t.j. $d_E(i^*,i)=\|\mathbf{r}_{i^*}-\mathbf{r}_i\|$, kde $\mathbf{r}_i$ označuje vektor súradníc i-teho neurónu v SOM. Parameter $\lambda(t)$ klesá s časom k nule, čím sa zabezpečuje zmenšovanie okolia počas učenia. Na obr. 6.21 je spôsob ako upravovať veľkosť okolia a α.

OBR.6.21.

AKTUALIZÁCIA PARAMETROV UČENIA



V procese učenia sa rozlišujú dve fázy. V prvej, nazývanej *fáza usporiadavania*, klesá veľkosť okolia diskrétne s časom. Počas druhej fázy – *fázy doladenia* – možno ponechať najbližších susedov súčasťou okolia, až kým učenie neskončí. Na funkcii poklesu parametra učenia α v praxi až tak veľmi nezáleží.

Dôležité je, aby α bola monotónne klesajúca funkcia z nejakej hodnoty blízkej 1, s malými hodnotami (rádovo 0,1–0,01) počas fázy doladenia. Možnou voľbou je napr. lineárna lomená funkcia, exponenciálna funkcia atď. Na presnom počte iterácií takisto nezáleží. Kohonen uvádza empiricky získanú pomôcku, podľa ktorej počet iterácií má byť minimálne 500-násobok počtu neurónov v sieti. Bežne sa počet iterácií pohybuje v rozmedzí rádovo 10000–100000. Na základe simulácií sa takisto ukázalo, že je vhodné rozdeliť celkovú dobu trénovania tak, že na fázu doladenia sa ponechá viac času ako na prvú fázu.

KROK 1: Zvolíme α_0, λ_0 a t_{max} (maximálny počet iterácií). Počiatočné váhy inicializujeme ako náhodné čísla $\in$ (–0.5, 0.5). Počítadlá nastavíme takto: $t = 0$, $p = 1$, kde t je iterácia (čas) a p je index vzoru.

KROK 2: Na vstup dáme vzor $\bar{x}^p$ a nájdeme víťazný neurón (rovnica 6.38).
KROK 3: Upravíme váhy víťaza a jeho topologických susedov (rovnica 6.39).

KROK 4: Aktualizujeme hodnoty α a λ (obr. 6.21).

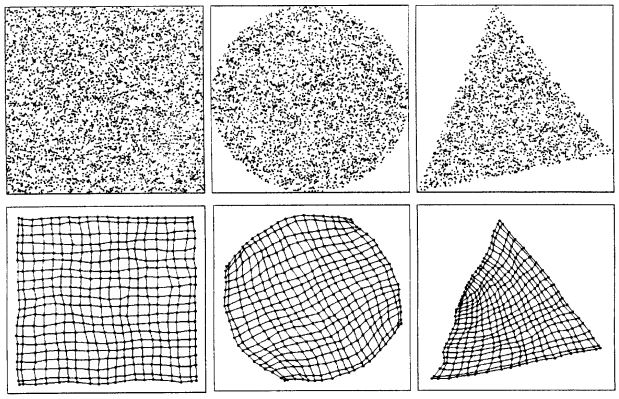
KROK 5: Ak $p < P$, tak polož $p = p + 1$ a choď na krok 2. Inak choď na krok 6.

KROK 6: Ak $t = t_{max}$, ukonči učenie. Inak polož $p = 1$ a choď na krok 2. Začína sa nový trénovací cyklus (epocha), t.j. nový prechod cez A_{train}.

Príklady topologických zobrazení pomocou SOM[9]

OBR.6.22.

APROXIMÁCIA DÁT S ROVNOMERNÝM ROZDELENÍM HUSTOTY

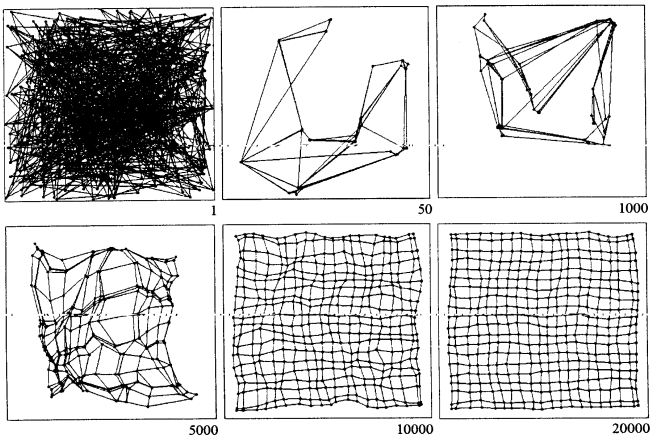


Horná trojica obrázkov znázorňuje tri rôzne množiny vstupných dát. Trénovacie vektory tvoria súradnice bodov v rovine, t.j. (x, y). Spodná trojica obrázkov znázorňuje váhy, $\bar{w} = (w_x, w_y)$ všetkých 20x20 neurónov SOM po natrénovaní (20000 iterácií). Váhy susedných neurónov sú spojené čiarou.

[9] Všetky príklady sú prevzaté z Farkaš 1997.

OBR.6.23.

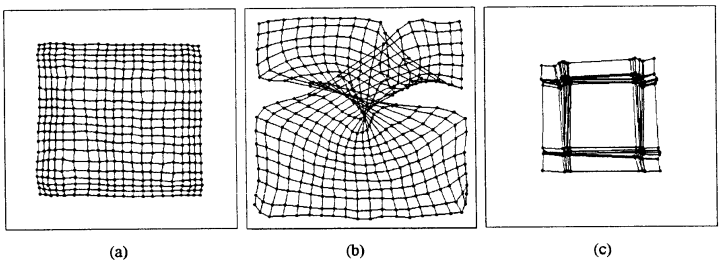
USPORIADAVANIE VÁHOVÝCH VEKTOROV



Obr. 6.23 ilustruje proces usporiadavania váhových vektorov SOM počas trénovania na dátach s rovnomerným rozdelením v tvare štvorca. Číslo pri pravom dolnom rohu vyjadruje počet vykonaných iterácií učenia. Horné tri obrázky zľava doprava ilustrujú fázu usporiadavania, spodná trojica obrázkov ilustruje fázu dolaďovania. Váhy susedných neurónov sú spojené čiarou, takže na obrázku v ľavom hornom rohu vidno, že po inicializácii malými náhodnými váhami, majú susedné neuróny v SOM úplne nepodobné váhové vektory.

OBR.6.24.

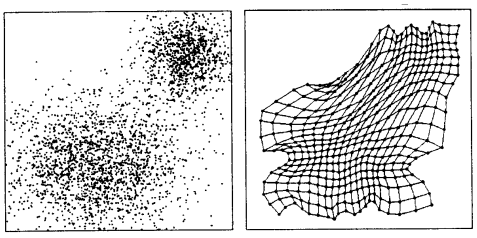
ZLÉ NATRÉNOVANIE SOM



Obr. 6.24 ilustruje niektoré špeciálne efekty, ktoré môžu vzniknúť pri zlom natrénovaní siete. (a) Neúplné rozvinutie siete v dôsledku príliš rýchleho poklesu rýchlosti učenia α v porovnaní so zmenšovaním okolia λ. (b) Tzv. motýlí efekt spôsobený príliš rýchlym zmenšením okolia λ v porovnaní s poklesom α (c) Tzv. „pinch“ efekt vznikajúci pri príliš pomalom zmenšovaní λ.

OBR.6.25.

APROXIMÁCIA DÁT S NEROVNOMERNÝM ROZDELENÍM HUSTOTY



Vstupné dáta tvoria dva gaussovské 2D „obláčiky“ bodov. Tam, kde sú dáta hustejšie, sú váhové vektory susedných neurónov v SOM k sebe bližšie. (20x20 neurónov, 20000 iterácií.)

SOM sa dá využiť na redukciu dimenzie dát a na odhalenie vnútornej štruktúry ich distribúcie. Toto príde vhod najmä pri viacrozmerných dátach.

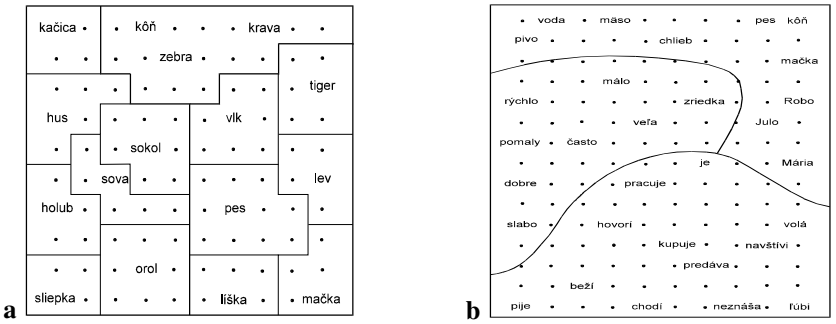
VEKTOROVÁ KVANTIZÁCIA

Na natrénovanú SOM sa môžeme pozerať ako na **vektorový kvantifikátor**. Pri vektorovej kvantifikácii je úlohou nahradiť danú množinu (distribúciu) vstupných dát početne menšou množinou referenčných vektorov, nazývaných **prototypy**. V SOM hrajú úlohu prototypov váhové vektory. Takáto náhrada sa môže uplatniť napr. pri prenose údajov v telekomunikáciách, či v kompressi dát (obrazových, rečových), keď sa dosiahne výrazné zmenšenie objemu dát pri minimálnom poklese kvality. Vďaka vektorovej kvantizácii stačí preniesť (či uchovať) len množinu prototypov spolu s informáciou (index prototypu) o príslušnosti každého vstupného vektora ku konkrétnemu prototypu (na základe ich vzájomnej euklidovskej vzdialenosti). Okolo každého prototypu (centra) možno vymedziť oblasť, ktorej množina bodov má k danému centru menšiu euklidovskú vzdialenosť ako ku akémukoľvek inému centru. Vektorovou kvantizáciou sa tak priestor X rozdelí na disjunktné oblasti, ktoré tvoria tzv. **Voronoiho mozaiku**.

TOPOGRAFICKÉ MAPY ABSTRAKTNÝCH DÁT (SYMBOLOV)

Vzťahy medzi symbolmi nie sú obvykle zistiteľné z ich kódových reprezentácií. SOM umožňuje využiť **kontext**, v ktorom sa symboly opakovane vyskytujú na vytvorenie topografických sémantických máp. Uvedieme dva príklady.

OBR.6.26. SÉMANTICKÉ MAPY



USPORIADANIE SYMBOLOV PODĽA ICH ATRIBÚTOV

V prvom príklade bol ku každému symbolu (meno zvieraťa) priradený vektor binárnych atribútov. Prítomnosť atribútu označená jednotkou, absencia nulou, ako veľkosť zvieraťa (malé, stredné, veľké), vonkajší popis tela (má 2 nohy, 4 nohy, srsť, kopytá, hrivu, perie) a čo rado robí (loví, behá, lieta, pláva). Takýto 13-rozmerný vektor atribútov $\mathbf{x}_a$ bol vygenerovaný pre každé zviera, pričom kódy zvierat $\mathbf{x}_s$ boli zámerne vytvorené tak, aby neniesli žiadnu informáciu o vzájomnej podobnosti medzi zvieratami: každý vektor $\mathbf{x}_s$ obsahoval samé nuly, až na jednu hodnotu z, ktorá figurovala na pozícii udávajúcej poradové číslo zvieraťa (1 až 16). Oba vektory boli zlúčené v jeden 29-rozmerný vektor $\mathbf{x} = [\mathbf{x}_s, \mathbf{x}_a]$ charakterizujúci každé zviera, pričom hodnota a bola stanovená na z = 0.2, aby vplyv atribútovej časti vektora $\boldsymbol{x}$ bol väčší ako vplyv symbolovej časti. Vektory $\boldsymbol{x}$ boli napokon normované kvôli lepšej stabilizácii učenia. Počas trénovania bolo SOM prezentovaných 2000 náhodne vyberaných vzorov $\boldsymbol{x}$ z 16-

prvkovej množiny. Proces určovania prototypov (návestí tried resp. indexov neurónov s maximálnou odozvou na daný symbol) bol však realizovaný na základe vektorov $\mathbf{x} = [\mathbf{x}_s, \mathbf{0}]$, čoho výsledkom je mapa na obr. 6.26a. Z toho vyplýva, že hoci reprezentácia vzájomných vzťahov podobnosti bola získaná vďaka prítomnosti atribútových častí počas trénovania, správna odozva SOM v testovacej fáze sa generuje i pri absencii $\mathbf{x}_a$, t.j. len na základe symbolovej časti. Kvalitatívne rovnakú reprezentáciu by sme dostali aj pri použití $\mathbf{x} = [\mathbf{x}_s, \mathbf{x}_a]$.

SÉMANTICKÁ MAPA NA ZÁKLADE KONTEXTU

V druhom príklade je kontext symbolu reprezentovaný pomocou iných symbolov, ako to možno pozorovať v prirodzenom jazyku. Uvažovaná množina 30 symbolov zahrňovala podstatné mená, slovesá a príslovky. Generované trénovacie vzory pozostávali zo zmysluplných trojslovných viet (napr. Robo pomaly beží, lev je mäso, atď.), teda symbol+dvojslovný kontext. Každý z troch symbolov bol nejako kódovaný ako 7-rozmerný vektor bez toho, aby sa v kódoch skrývala nejaká podoba. Opäť, aby sa zvýraznil vplyv kontextu, bol parameter z v symbolovej časti stanovený na $z = 0.2$. Po natrénovaní na 2000 (21-rozmerných) vstupných vzoroch tvaru $\mathbf{x} = [\mathbf{x}_s, \mathbf{x}_a]$ boli prototypy určené len na základe symbolovej časti a výsledkom je mapa na obr. 6.26b. Separované oblasti označujú jednotlivé slovné druhy, v rámci ktorých možno vidieť aj usporiadanosť podľa významu zastúpených symbolov.

6.7 HISTÓRIA UMELÝCH NEURÓNOVÝCH SIETÍ

VÝPOČTOVÁ NEUROVEDA

Jedna z tém v histórii neurónových sietí súvisí s realistickým modelovaním nervových buniek a mozgu (Bower and Beeman, 1995). V súčasnosti sa pre túto oblasť používa názov **výpočtová neuroveda** (angl. *computational neuroscience*). V súčasnosti sa vo výpočtovej neurovede široko používajú a skúmajú tzv. **impulzné neuróny** (angl. *spiking neurons*) (Maass and Bishop, 1999).

MCCULLOCH A PITTS

V r. 1943 McCulloch a Pitts predstavili prvú umelú neurónovú sieť (McCulloch and Pitts, 1943). Ich **formálne neuróny** boli vlastne iba jednoduché **logické prepínače**. Hodnoty synaptických váh a prahov boli fixné. McCulloch a Pitts ukázali, že všetky procesy, ktoré sa dajú opísať konečným počtom symbolických výrazov, teda jednoduchá aritmetika, klasifikácia, záznam konečnej množiny dát, rekurzívna aplikácia logických pravidiel a pod., sa dajú realizovať sieťou zloženou z takýchto elementov. Ak je táto sieť nekonečne veľká, tak je výpočtovo ekvivalentná univerzálnemu Turingovmu stroju.

PERCEPTRÓNY

V roku 1958 Frank Rosenblatt ukázal, že McCullochove-Pittsove siete s modifikovateľnými synaptickými váhami sa dajú natrénovať tak, aby vedeli rozpoznávať a klasifikovať objekty (Rosenblatt, 1958). Vymyslel pre ne názov „perceptróny“. V r. 1969 Minsky a Papert ukázali, že tieto siete vôbec nie sú výpočtovo univerzálne a dokážu riešiť iba **lineárne separovateľné problémy** (Minsky and Papert, 1969). Tento **problém sa vyriešil** až skoro o 20 rokov neskôr, v r. 1986, keď Rumelhart, Hinton a Williams zaviedli pravidlo učenia **metódou spätného šírenia chýb** (angl. *error back-propagation learning*) pre

UČENIE METÓDOU SPÄTNÉHO ŠÍRENIA CHÝB

viacvrstvové dopredné neurónové siete (Rumelhart et al., 1986).

HOPFIELDOV MODEL

Špeciálnu kapitolu tvoria tzv. **Hopfieldove** alebo **atraktorové neurónové siete** (Amit, 1989; Kvasnička et al., 1997). Sú založené na analógii s fyzikálnymi systémami. Tieto siete sa neučia, ale ich **pamäťové stavy sú predprogramované** v matici synaptických spojení, ktorá je skonštruovaná tak, aby pamäťové stavy tvorili atraktory v stavovom priestore.

SAMOORGANIZÁCIA

Značná časť výskumu v neurónových sieťach sa venuje aj algoritmom **učenia bez učiteľa** (angl. *unsupervised learning*) a na systémy, ktoré sú **schopné samoorganizácie**. Vyvinuli sa neurónové siete, ktoré sú schopné naučiť sa klasifikovať vzory bez explicitnej informácie o tom, ktoré vzory do ktorej triedy patria. Takéto neurónové siete sú schopné samy objaviť **štatistické pravidelnosti** vo vstupných dátach a zakódovať ich na svojom výstupe. (Kohonen, 1995; Farkaš, 1997).

ZHRNUTIE

V tejto kapitole sme uviedli čitateľa do problematiky umelých neurónových sietí s dôrazom na **algoritmy učenia s učiteľom** (angl. *supervised learning*) • Na doplnenie vedomostí o trénovaní umelých neurónových sietí bez spätnej väzby od učiteľa z domácej literatúry odporúčame (Kvasnička et al., 1997; Oravec et al., 1998).

NEUROBIOLÓGIA

Umelé neurónové siete sú inšpirované poznatkami o mozgu • Objekty sú v mozgu reprezentované redundantne a distribuovane • Učenie sprevádzajú **zmeny váh synáps** v mozgových neurónových sieťach • Umelé neurónové siete majú emergentné správanie • „Vidia štatistiku“ svojich stimulov

PERCEPTRÓN

Perceptrón má binárny výstup • Svojim učením **koriguje chybu na výstupe** • Rieši iba lineárne separovateľné problémy

DOPREDNÉ UNS

RBF

SPÄTNÉ ŠÍRENIE CHÝB

Prvky dopredných neurónových sietí majú sigmoidálnu vstupno-výstupnú funkciu • Môžu mať aj inú prechodovú charakteristiku, napr. v tvare radiálnej bázovej funkcie (RBF) • Jednotlivé vrstvy vysielajú spojenia iba dopredu • V priebehu učenia tieto spojenia modifikujú svoju váhu tak, aby sa minimalizovala suma štvorcov chýb medzi požadovaným a skutočným výstupom siete • Umelá neurónová sieť so skrytou neurónovou vrstvou funguje ako **univerzálny aproximátor** mnohorozmerných funkcií • Vie riešiť aj **nelineárne problémy klasifikácie** • Po natrénovaní je schopná zovšeobecňovať na príklady, ktoré nikdy nevidela • Je citlivá na preučenie • Môžeme skoro zastaviť jej učenie a vybrať optimálny počet skrytých neurónov

HRANIE HIER

Namiesto toho, aby sme sieti povedali zakaždým správne riešenie, iba jej **výkon oznámkujeme** • Aj tak sa naučí zovšeobecňovať • Je dobré kombinovať umelé neurónové siete s inými prístupmi UI

PREDIKCIA

Dopredná neurónová sieť s **oknom do minulosti** a rekurentné neurónové siete s

BUDÚCEHO VÝVOJA DÁT

vnútornou pamäťou dokážu reprezentovať časový kontext údajov • Učia sa veľmi pomaly • Na ich dobré natrénovanie potrebujeme tisícky opakovaní

HISTÓRIA UMELÝCH NEURÓNOVÝCH SIETÍ

História umelých neurónových sietí sa datuje od roku 1943 • Zaviedlo sa mnoho algoritmov učenia a rôznych architektúr • V 90-tych rokoch 20. storočia sa podarilo dokázať **výpočtovú univerzálnosť** umelých neurónových sietí.

KĽÚČOVÉ POJMY

asociácia
neurón
excitácia a inhibícia
učenie
kódovanie a reprezentácia
perceptrón
lineárna separácia
pravidlo učenia δ
trénovacia množina
spätné šírenie chýb
spätné šírenie chýb v čase
okno do minulosti
sigmoida
viacvrstvové dopredné neurónové siete
chybová funkcia
univerzálna aproximácia
klasifikácia
preučenie, pretrénovanie
testovanie a validácia
skoré zastavenie učenia
počet skrytých neurónov
selekcia modelu
učenie s odmenou a trestom
hranie hier
časová štruktúra v dátach
neurónové siete s oknom do minulosti
rekurentné neurónové siete
predikcia údajov
rekurentné učenie v reálnom čase
RBF siete
Hebbovo pravidlo učenia
samoorganizácia
vektorová kvantizácia
generalizácia

CVIČENIA

1. 1. Naprogramujte binárny perceptrón s troma vstupmi. Pomocou delta pravidla trénujte perceptrón na vykonávanie logických funkcií AND a OR. Zaznamenajte evolúciu stavov perceptrónu t.j. hodnoty na vstupoch a na výstupe v každom trénovacom kroku.

1. 2. Skúmanie použitia binárneho perceptrónu na klasifikáciu objektov (číslic) prichádzajúcich zo „ sietnice oka “ rozmerov 5 x 5. Naprogramujte binárny perceptrón s 25 vstupmi, ktorý klasifikuje tieto číslice na párne a nepárne. Zopakujte cvičenie pre klasifikáciu číslic ako ≤ 3 a > 3. Po natrénovaní skúste číslice poškodiť (nahradiť niektoré 1 nulami a naopak) a pozorujte, či ich perceptrón aj tak správne zatriedi.

```
01110     00100     01110     01110     01000
10001     01100     10010     00001     01000
10001     00100     00100     01110     01010
10001     00100     01000     00001     01111
```

```
01110     01110     11111     01110     00010
```

1. 3. Navrhnite a natrénujte doprednú dvojvrstvovú neurónovú sieť tak, aby vykonávala binárne násobenie 3 bitov 3 bitmi. Jeden príklad trénovacieho páru je vstup=011011 a požadovaný výstup=001001 (t.j. 3x3=9). Celkovo je 64 trénovacích párov.

1. 4. Implementujte doprednú dvojvrstvovú neurónovú sieť, ktorá generuje hodnoty funkcie $f(x, y) = 0{,}8*\sin(x+y)$, kde suma $x+y$ nepresahuje $\pm\pi$. Sieť bude mať 2 vstupy, x a y , a 1 výstup. Na trénovanie použite asi 20 vzorov. Testujte sieť na zovšeobecňovanie pomocou ďalších 20 hodnôt, na ktoré nebola natrénovaná.

1. 5. V predchádzajúcich dvoch úlohách experimentujte s momentom a počtom skrytých neurónov. Po naučení môžete skúsiť vymazať nejaké spojenia a pozorovať, či a ako sa výkon siete zhoršuje.

1. 6. Opakujte úlohu 1.4 s RBF sieťou. Experimentujte s počtom skrytých neurónov. Po naučení môžete skúsiť vymazať nejaké spojenia a pozorovať, či a ako sa výkon siete zhoršuje.

1. 7. Navrhnite si nejaký konečno-stavový automat. (a) Vytvorte si trénovaciu množinu zloženú z pozitívnych (patriacich do jazyka) a negatívnych (nepatriacich do jazyka) slov napr. maximálnej dĺžky 10. Natrénujte RNS pomocou BPTT na zovšeobecnenie klasifikácie postupností symbolov, na ktoré nebola natrénovaná. (b) Trénujte RNS pomocou RTRL na tzv. „next-symbol prediction“ na slovách z automatu. Nechajte ju generovať pokračovanie testovacích slov.

LITERATÚRA

AMIT, D. J.: *Modeling Brain Function, The World of Attractor Neural Networks,* Cambridge: Cambridge University Press, 1989.

BAHNA, R.: *Využitie neurónových sietí na riešenie logických problémov*. Bratislava: FEI STU, 1998. Diplomová práca.

BARNSLEY, M.: *Fractals Everywhere*. San Diego: Academic Press, 1988.

BAXTER, J. - TRIDGELL, A. - WEAVER, L.: *KnightCap: a chess program that learns by combining TD(λ) with MINIMAX search.* Canberra: Australian National Univ., 1997.

BEŇUŠKOVÁ, Ľ. - KVASNIČKA, V. - POSPÍCHAL, J. (eds): *Hľadanie spoločného jazyka v kognitívnych vedách.* Bratislava: Iris, 2000. (http://www.ii.fmph.uniba.sk/~benus)

BENGIO, Y. - CARDIN, R. - DeMORI, R.: Speaker independent speech recognition with neural networks and speech knowledge. In: *Advances in Neural Information Processing Systems II*, D.S. Touretzky (ed), San Mateo, CA: Morgan Kaufmann, 1990, pp. 218-225.

BISHOP, C. M.: Neural Networks for Pattern Recognition. Oxford: Clarendon Press, 1995.

BOWER, J. M. - BEEMAN, D.: *The Book of GENESIS: Exploring Realistic Neural Models with the General NEural Simulation System*. New York: Springer Verlag, 1995. (http://www.bbb.caltech.edu/GENESIS)

ČERŇANSKÝ, M.: *Recurrent neural networks.* Písomná práca k rigoróznej skúške, FEI STU, 2001. (http://www.dcs.elf.stuba.sk/~cernans)
COTTRELL, G. W. - MUNRO, P. - ZIPSER, D.: Image compression by back propagation: an example of extensional programming. In: *Models of Cognition: A Review of Cognition Science.* N. E. Sharkey (ed), Norwood, NJ, 1989.
EFRON, B. - TIBSHIRANI, R. J.: *An Introduction to the Bootstrap*. London: Chapman & Hall, 1993.
ELMAN, J. L.: Finding structure in time. *Cognitive Science*, vol. 14, 1990, pp. 179-211.
FARKAŠ, I.: Samoorganiyujúce sa mapy. In: *Úvod do teórie neurónových sietí.* V. Kvasnička et al., Bratislava: Iris, 1997, pp. 142-189.
GORMAN, R. - SEJNOWSKI, T.: Learned classification of sonar targets using a massively parallel network. *IEEE Trans. Acoustics, Speech and Signal Proc.*, vol. 36, 1988, pp. 1135-1140.
HEBB, D.: *The Organization of Behavior*. New York: John Wiley and Sons, 1949.
HORNIK, K. - STINCHCOMBE, M. - WHITE, H.: Multilayer feedforward networks are universal approximators. *Neural Networks*, vol. 2, 1989, pp. 359-366.
JORDAN, M. I.: Serial order: a parallel distributed processing approach. In: *Advances in Connectionist Theory*. J.L. Elman and D. E. Rumelhart (eds), Hillsdale: Erlbaum, 1989.
KOHONEN, T.: *Self-Organizing Maps*. Berlin: Springer Verlag, 1995.
KOLEN, J.F.: The origin of clusters in recurrent network state space. In: *Proc. 16th Annual Conf. Cognitive Science Society,* Atlanta, GA, Aug. 13-16, 1994.
KVASNIČKA, V. - BEŇUŠKOVÁ, Ľ. - POSPÍCHAL, J. - FARKAŠ, I. - TIŇO, P. - KRÁĽ, A.: *Úvod do teórie neurónových sietí*. Bratislava: Iris, 1997.
(ftp://math.chtf.stuba.sk/pub/VLADO/NN_books_texts/UvodDoTeorieNS.pdf.zip)
KVASNIČKA, V. - POSPÍCHAL, J. - TIŇO, P.: *Evolučné algoritmy*. Bratislava: STU, 2000.
LeCUN, Y. - BOSER, B. - DENKER, J. S. et al.: Backpropagation applied to handwritten Zip code recognition. *Neural Computation*, vol. 1, 1989, pp. 541-551.
MARŠALA, J.: *Systematická a funkčná neuroanatómia*. Martin: Osveta, 1985.
MAAS, W. - BISHOP, C. M.: *Pulsed Neural Networks.* Cambridge, MA: MIT Press, 1999.
McCULLOCH, W. S. - PITTS, W.: A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, vol. 5, 1943, pp. 115-137.
MINSKY, M. - PAPERT, S: *Perceptrons: An Introduction to Computational Geometry*. Cambridge, MA: MIT Press, 1969.
ORAVEC, M. - POLEC, J. - MARCHEVSKÝ, S. et al.: *Neurónové siete pre číslicové spracovanie signálov*. Bratislava: Faber, 1998.
POMERLEAU, D. A.: ALVINN: An autonomous land vehicle in a neural network. In: *Advances in Neural Information Processing Systems I*, D.S. Touretzky (ed), San Mateo, CA: Morgan Kaufmann, 1989, pp. 305-313.
ROSENBLATT, F.: The Perceptron, a probabilistic model for information storage and organization in the brain. *Psychological Review*, vol. 62, 1958, pp. 386-408.
RUMELHART, D. E. - HINTON, G. E. - WILLIAMS, R. J.: Learning internal representations by error propagation. In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1., Foundations*. D. E. Rumelhart and J. L. McClelland (eds), Cambridge, MA: MIT Press / Bradford Books, 1986, pp. 318-363.
SEJNOWSKI, T. - ROSENBERG, C.: Parallel networks that learn to pronounce English text. *Complex Systems*, vol. 1, 1987, pp. 145-168.
SINČÁK, P. - ANDREJKOVÁ, G.: *Neurónové siete I, II.* Košice: Elfa, 1996. (http://neuron-ai.tuke.sk/cig/source/publications/books/NS1/html/index.html a …/NS2/html/index.html)

SUTTON, R. S.: Learning to predict by the methods of temporal differences. *Machine Learning*, vol. 3, 1988, pp. 9-44.
TESAURO, G.: Neurogammon wins computer olympiad. *Neural Computation*, vol. 1, 1990, pp. 321-323.
TIŇO, P. - ŠAJDA, J.: Learning and extracting initial Mealy machines with a modular neural network model. *Neural Computation*, vol. 4, 1995, pp. 822-844. (http://www.dcs.elf.stuba.sk/~tino)
TIŇO, P. – ČERŇANSKÝ, M. – BEŇUŠKOVÁ, Ľ.: Markovian architectural bias of recurrent neural networks. In: *Intelligent Technologies – Theory and Applications,* P. Sinčák, J. Vaščák, V. Kvasnička, J. Pospíchal (eds), Amsterdam: IOS Press, 2002, pp. 17-23.
WEIBEL, A.: Modular construction of time-delay neural networks for speech recognition. *Neural Computation*, vol. 1, 1989, 39-46.
WILLIAMS, R. J. - ZIPSER, D.: A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, vol. 1, 1989, pp. 270-280.
ZURADA, J.M.: *Introduction to Artificial Neural Systems*. St. Paul, MN: West Publ. Comp. 1992.