

# 6 UMELE NEURÓNOVÉ SIETE

POTREBNÉ POJMY Z NEUROBIOLÓGIE •  
PERCEPTRÓN • VIACVRSTVOVÉ DOPREDNÉ SIETE •  
UČENIE SA Z PRÍKLADOV • REKURENTNÉ SIETE •  
RBF SIETE • SOM – SAMOORGANIZUJÚCA SA  
MAPA • HISTÓRIA UMELÝCH NEURÓNOVÝCH SIETÍ

*V tejto kapitole 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 v súčasnosti vrcholom evolúcie na našej Zemi a zatiaľ najdokonalejším „nástrojom“ na spracovanie informácií. Tak ako sa klasická umelá inteligencia snaží napodobovať ľudskú inteligenciu, prenesene povedané „mentálny softvér“, tak sa umelé neurónové siete snažia napodobovať „mozgový hardvér“, na ktorom tento „mentálny softvér“ beží.

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 multiagentových systémov sú založené na modelovaní princípov evolúcie života a spoločnosti.

UMELÉ  
NEURÓNOVÉ SIETE  
V KONTEXTE  
UMELEJ  
INTELIGENCIE

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 iba jedna z možností. Ďalšími alternatívami sú napríklad klasické štatistické metódy, multiagentové 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 vedú 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 hlavne takými modelmi umelých neurónových sietí, ktoré sa učia tak, že „učiteľ“ im v priebehu učenia hovorí, aké má byť správne riešenie problému. Predstavíme však aj také umelé neurónové siete, ktoré sa učia bez tejto informácie a majú vlastnosti samoorganizácie.

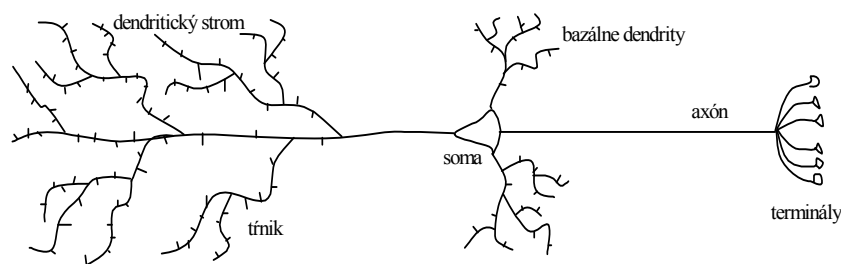
Základná terminológia umelých neurónových sietí je založená na neurobiologickej terminológii, preto našu kapitolu začíname vysvetlením potrebných neurobiologických pojmov. Historicky aj konceptuálne je dôležitá teória perceptrónov, na ktorú nadväzuje konštrukcia dopredných viacvrstvových neurónových 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. Spomenieme RBF siete s novým typom stavebných prvkov, ktoré podstatne zrýchľujú učenie. Na záver sa budeme venovať samoorganizujúcim sa neurónovým sieťam, ktorých učenie je silno biologicky motivované.

## 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, ako sú myslenie, vnímanie a pamäť. V neurónoch prebiehajú zložité biochemické deje, ktoré zabezpečujú to, že neuróny môžu spracúvať signály z iných neurónov a vysielat' 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**. 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). Tisíciky dendritov tvoria bohato rozvetvený strom, na ktorom sa nachádza väčšina synáps, a to najmä na tŕnikoch.

OBR. 6.1.  
NEURÓNEXCITÁCIA  
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  $10\text{--}10^2$  Hz. Frekvencia je úmerná celkovej stimulácii neurónu. Výstupné impulzy sa šíria k ostatným neurónom pozdĺž jediného výstupného výbežku neurónu, ktorý sa nazýva *axón*. Axón sa na svojom konci rozvetvuje na tisíce 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 „doladovať“ 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 (Jedlička a kol., 2002; Beňušková, 2002). 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, že 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 **distribúovaná reprezentácia**. Rozličné objekty sa reprezentujú rozličnými vzorcami resp. distribúciami aktivity v príslušných neurónových sieťach.

? Vstupnú aj výstupnú aktivitu umelých neurónových sietí interpretuje programátor. 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**<sup>1</sup> je model neurónu, ktorý prijíma vstupné signály  $\bar{x} = (x_1, x_2, \dots, x_{n+1})$  cez synaptické váhy tvoriace váhový vektor  $\bar{w} = (w_1, w_2, \dots, w_{n+1})$  (pozri obr. 6.2).

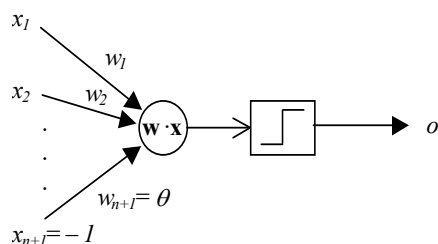
Vstupný vektor  $\bar{x}$  sa nazýva **vzor** alebo obrazec (angl. *pattern*). Zložky vstupného vektora môžu nadobúdať reálne alebo binárne hodnoty. Príkladom môže byť obrázok (písmeno, odtlačok prsta atď.) zakódovaný ako pole (vektor) hodnôt. Zložky váhového vektora sú reálne čísla.

Výstup perceptrónu  $o$  je daný vzťahom:

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

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 diskretný perceptrón s bipolárnou binárnou aktivačnou funkciou (funkcia signum, znamienko):

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

Rovnica

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

<sup>1</sup> Percept = vnem.

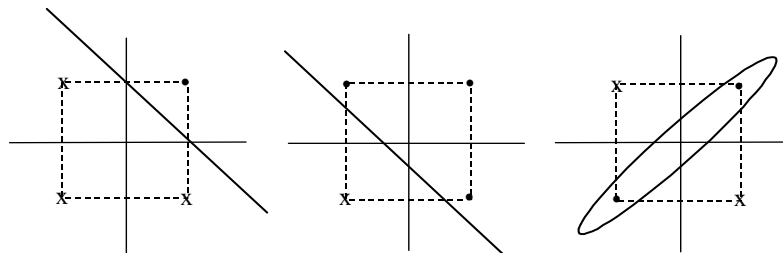
je rovnicou nadroviny v  $n$ -rozmernom priestore. Napr. v 2-rozmernom priestore je to rovnica priamky:  $w_1x_1 + w_2x_2 - \theta = 0$ . Váhy perceptrónu predstavujú koeficienty deliacej priamky (nadroviny), ktorá rozdeľuje priestor vzorov na dva podpriestory. Perceptrón teda dokáže klasifikovať, t.j. zatriediť 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 neuronové siete znie: Váhový vektor  $\bar{w}$  sa zväčšuje priamo úmerne so súčinom vstupného vektora  $\bar{x}$  a učiaceho signálu  $s$ . Učiaci signál  $s$  je funkciou  $\bar{w}$ ,  $\bar{x}$  a niekedy aj spätnej väzby od učiteľa  $d$ . Teda:

$$s = s(\bar{w}, \bar{x}, d) \quad \text{alebo} \quad s = s(\bar{w}, \bar{x}) \quad (6.4)$$

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 neuronových sieťach sa v diskretných časových krokoch mení  $j$ -ta váha takto:

$$w_j(t+1) = w_j(t) + \Delta w_j(t) = w_j(t) + \alpha s(t) x_j(t) \quad (6.5)$$

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

$\delta$ -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  $\delta$  (delta)**. Pre  $j$ -tu váhu platí:

$$\Delta w_j = \alpha (d - o) x_j \quad (6.6)$$

Pre binárny bipolárny perceptrón je požadovaným výstupom pre jednu triedu  $d = +1$ , a pre druhú triedu  $d = -1$ . Pravidlo  $\delta$  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

TRÉNOVACIA  
MNOŽINA

tomu, aby sa po  $k$  krokoch daný vzor klasifikoval do triedy  $d = 1$ . Podľa pravidla (6.6) sa upravuje aj prah perceptrónu  $w_{n+1} = \theta$ .

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

**Krok 1:** Zvolíme  $\alpha \in (0, 1)$ . 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  $o = \text{sign}(\sum_{j=1}^{n+1} w_j x_j^p)$ .

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

**Krok 4:** Ak  $p < P$ , tak polož  $p = p + 1$  a choď na krok 2. Inak choď na krok 5.

**Krok 5:** Bez úpravy váh ešte raz prejdeme cez  $A_{train}$  a vypočítame kumulovanú chybu  $E \leftarrow E + \frac{1}{2}(d^p - o^p)^2$  pre  $p = 1, 2, \dots, P$ .

**Krok 6:** Ak  $E = 0$ , ukonči učenie. Inak polož  $E = 0$ ,  $p = 1$ ,  $k = k + 1$  a choď na krok 2. Začína sa nový tréovací 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 a 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 tréovania nazvané **metóda spätného šírenia chýb** (angl. error backpropagation) pre dopredné neurónové siete so skrytými neurónmi.

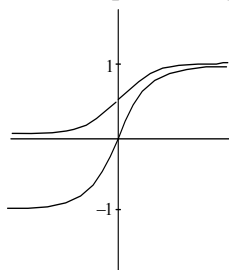


Tzv. **viacvrstvové dopredné** umelé neurónové siete (angl. multilayer feed-forward ANN), ktoré sa tréujú týmto pravidlom sú schopné riešiť aj nelineárne problémy. Pri odvodzovaní tohoto algoritmu pre dvojvrstvovú doprednú sieť budeme postupovať podľa (Zurada, 1992). Dopredné neurónové siete sa vyznačujú 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  $\delta$  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

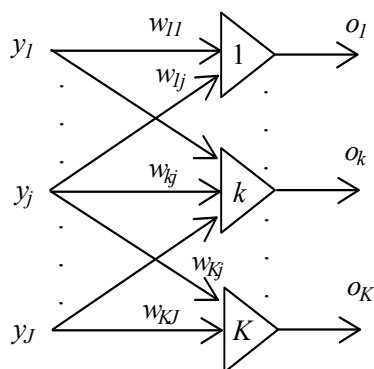


$$\text{Unipolárna sigmoida: } f(net) = \frac{1}{1 + \exp(-\lambda net)} \quad (6.7)$$

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

Konštanta  $\lambda > 0$  sa nazýva strmosť sigmoidy. Zvyčajne sa používa  $\lambda = 1$ . V limite pre  $\lambda \rightarrow \infty$  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  $\bar{y} = (y_1, \dots, y_j, \dots, y_J)$ . Výstupný vektor je  $\bar{o} = (o_1, \dots, o_k, \dots, o_K)$ , kde  $o_k = f(net_k)$  a

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

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

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  $p = 1, \dots, P$  z  $A_{train}$ . Definujeme účelovú funkciu, ktorá sa volá **chybová funkcia** a má tvar:

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

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<sup>2</sup> nazývaná aj **metóda najprudšieho spádu (najstrmšieho zostupu)** (angl. steepest descent).

Zmena váhy  $w_{kj}$  je nepriamo úmerná parciálnej derivácii  $E_p$  podľa  $w_{kj}$ <sup>3</sup>:

$$\Delta w_{kj} = -\alpha \frac{\partial E_p}{\partial w_{kj}} = -\alpha \frac{\partial E_p}{\partial(\text{net}_k)} \frac{\partial(\text{net}_k)}{\partial w_{kj}} = \alpha \delta_{ok} y_j \quad (6.11)$$

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

$$\delta_{ok} = -\frac{\partial E_p}{\partial(\text{net}_k)} = -\frac{\partial E_p}{\partial o_k} \frac{\partial o_k}{\partial(\text{net}_k)} = (d_{pk} - o_{pk}) f'_k \quad (6.12)$$

$f'_k$  je derivácia aktivačnej funkcie neurónu (sigmoidy) podľa  $\text{net}_k$ . 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  $\delta$  PRE  
VÁHY VÝSTUPNÝCH  
NEURÓNŮV

$$\Delta w_{kj} = \alpha (d_{pk} - o_{pk}) f'_k y_j \quad (6.13)$$

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  $\bar{x} = (x_1, \dots, x_i, \dots, x_J)$ . Výstupné neuróny spracúvajú výstup skrytej vrstvy  $\bar{y} = (y_1, \dots, y_j, \dots, y_J)$ , kde  $y_j = f(\text{net}_j)$  a

$$\text{net}_j = \sum_{i=1}^I v_{ji} x_i \quad (6.14)$$

$I$ -ty vstup je vždy  $= -1$  ako aj výstup  $J$ -teho skrytého neurónu<sup>4</sup>. Modifikovateľné prahy skrytých neurónov sú  $v_{jI} = \theta_j$ , pre  $j = 1, \dots, J$ .

<sup>2</sup> Gradient skalárnej funkcie, napr. troch premenných  $g(x, y, z)$ , udáva smer najprudšieho nárastu jej hodnoty. Vyjadruje to vektor  $\bar{\nabla}g(x, y, z) = \frac{\partial g}{\partial x} \bar{i} + \frac{\partial g}{\partial y} \bar{j} + \frac{\partial g}{\partial z} \bar{k}$ , kde  $\bar{i}, \bar{j}, \bar{k}$  sú jednotkové vektory s v smere jednotlivých osí.

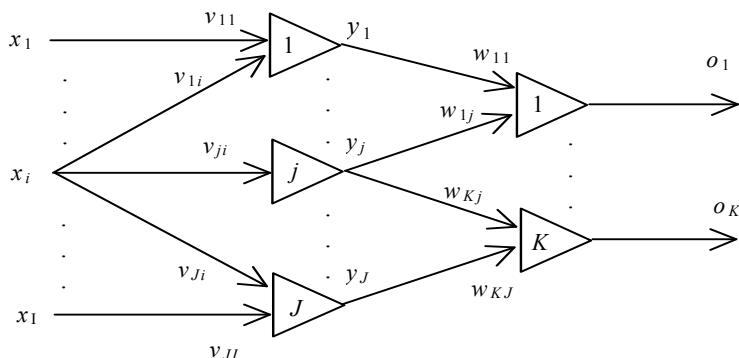
Zložky gradientu sú parciálne derivácie podľa jednotlivých premenných. V našom prípade výpočtu gradientu  $E_p$  budú zložkami parciálne derivácie podľa jednotlivých váh a prahov.

<sup>3</sup>  $E_p$  derivujeme ako zloženú funkciu. Vo všeobecnosti ak máme  $h(z(y(x)))$ , tak  $\partial h / \partial x = (\partial h / \partial z) (\partial z / \partial y) (\partial y / \partial x)$ .

<sup>4</sup> 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:

$$\Delta v_{ji} = -\alpha \frac{\partial E_p}{\partial v_{ji}} = -\alpha \frac{\partial E_p}{\partial(\text{net}_j)} \frac{\partial(\text{net}_j)}{\partial v_{ji}} = \alpha \delta_{yj} x_i \quad (6.15)$$



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

$$\delta_{yj} = -\frac{\partial E_p}{\partial(\text{net}_j)} = -\frac{\partial E_p}{\partial y_j} \frac{\partial y_j}{\partial(\text{net}_j)} = -\frac{\partial E_p}{\partial y_j} f'_j \quad (6.16)$$

$f'_j$  je derivácia výstupnej funkcie skrytého neurónu (sigmoidy) podľa  $\text{net}_j$ .

$$\frac{\partial E_p}{\partial y_j} = -\sum_{k=1}^K (d_{pk} - o_{pk}) \frac{\partial \{f(\text{net}_k)\}}{\partial y_j} = -\sum_{k=1}^K (d_{pk} - o_{pk}) \frac{\partial f(\text{net}_k)}{\partial(\text{net}_k)} \frac{\partial(\text{net}_k)}{\partial y_j} \quad (6.17)$$

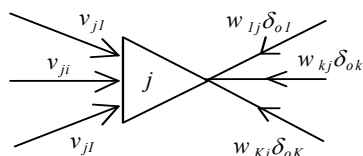
Keďže  $f'_k$  je derivácia sigmoidy výstupného neurónu podľa  $\text{net}_k$  a  $\partial(\text{net}_k) / \partial y_j = w_{kj}$  (pozri vzťah 6.9), potom

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

ZOVŠEOBECNÉ  
PRAVIDLO  $\delta$  PRE  
VÁHY SKRYTÝCH  
NEURÓNŮV

Dosadením do (6.16) dostávame vzťah pre  $\delta_{yj}$ :

$$\delta_{yj} = \left( \sum_{k=1}^K \delta_{ok} w_{kj} \right) f'_j \quad (6.19)$$

**OBR.6.7.**SPÄTNÉ ŠÍRENIE  
CHÝB NA SKRYTÝ  
NEURÓNA nakoniec vzťah pre učenie váh  
medzi vstupom a skrytými neurónmi:

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

ZOVŠEOBECNENIE  
PRE  $n$  SKRYTÝCH  
VRSTVIEVPredstavme si, že máme  $n$  skrytých vrstiev. Vo všeobecnosti pre  $n$ -tú skrytú vrstvu platí:

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

kde

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

 $(f_j^n)'$  je derivácia aktivačnej funkcie skrytého neurónu  $n$ -tej vrstvy podľa  $net_j^n$ .ZRÝCHLENIE  
UČENIAČasto používaná metóda na zrýchlenie konvergencie do minima chybovej funkcie je **momentum**:

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

VEĽKOSŤ  $\alpha$  A  $\mu$ kde konštanta  $\mu \in (0, 1)$  sa nazýva momentum alebo momentový člen. Zvyčajne sa volí  $\mu \approx \alpha$ . Rádovo je zvyčajne veľkosť týchto konštánt  $10^{-2}$  až  $10^{-1}$ . Čo sa týka konkrétnych hodnôt, tie sa zisťujú experimentálne pre daný problém.Majme  $A_{train} = \{(\bar{x}^1, \bar{d}^1)(\bar{x}^2, \bar{d}^2) \dots (\bar{x}^p, \bar{d}^p) \dots (\bar{x}^P, \bar{d}^P)\}$ . Potom je algoritmus tréningu doprednej neuronovej siete pomocou spätného šírenia chýb takýto:KUMULOVANÁ  
CHYBA**Krok 1:** Zvolíme  $\alpha \in (0, 1)$ . Počiatočné váhy (vrátane prahov) inicializujeme ako malé náhodné čísla<sup>5</sup> 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:

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

kde  $E_p$  sa počíta podľa (6.10). Zvolíme malé kladné číslo  $\varepsilon$ , ktoré nám bude slúžiť na zastavenie učenia.**Krok 2:** Na vstup dáme vzor  $\bar{x}^p$  a vypočítame  $\bar{y}^p$  a  $\bar{o}^p$ .**Krok 3:** Pre každý výstupný neurón vypočítame  $\delta_{ok}$  podľa (6.12) a pre každý<sup>5</sup> 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.

skrytý neurón  $\delta_{yj}$  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 tréningový cyklus, nová epocha tréningovania, t.j. nový prechod cez  $A_{train}$ .<sup>6</sup>

### Dopredná viacvrstvá sieť ako univerzálny aproximátor



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

Majme doprednú NS s jedným výstupným neurónom a jednou skrytou vrstvou.

Majme  $A_{train} = \{\bar{x}^1, \bar{x}^2, \dots, \bar{x}^p, \dots, \bar{x}^P\}$  tvorenú vstupnými vektormi dimenzie  $I$ .

VETA O  
UNIVERZÁLNEJ  
APROXIMÁCII

Pre  $\forall \varepsilon > 0 \exists$  taká funkcia  $G(\bar{x}) = f\left(\sum_{j=1}^J w_j f\left(\sum_{i=1}^I v_{ji} x_i\right)\right)$ , kde  $w_j$  je váha synapsy

medzi  $j$ -tým skrytým neurónom a výstupným neurónom,  $v_{ji}$  je váha synapsy medzi  $i$ -tým vstupom a  $j$ -tým skrytým neurónom a  $f(z)$  je diferencovateľná aktivačná funkcia. Pre ľubovoľnú spojitú funkciu  $F: \mathcal{R}^I \rightarrow (0, 1)$ , ktorá je definovaná nad konečnou množinou  $A_{train}$  platí, že  $\sum_{p=1}^P |F(\bar{x}^p) - G(\bar{x}^p)| < \varepsilon$ .

Hovoríme, že funkcia  $G$  aproximuje funkciu  $F$  nad tréningovou množinou  $A_{train}$  s presnosťou  $\varepsilon$  (Hornik a kol., 1989).

GENERALIZÁCIA

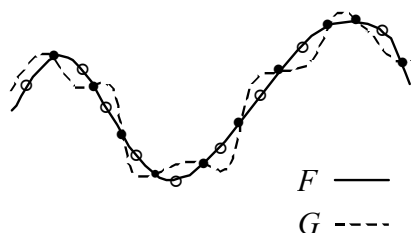
Táto veta hovorí, že existuje taká dvojvrstvá sieť umelá sieť, ktorá aproximuje  $F$  pre  $A_{train}$ . Rovnica (6.10) vyjadruje sumu štvorcov chýb, ktorú učenie minimalizujeme. Tréningom siete upravujeme váhy, ktoré sú vlastne koeficientami polynómu  $G$ , ktorý sieť prekladá cez body dané tréningovou množinou (pozri obr. 6.8). Cieľom je extrapolácia funkčných hodnôt mimo  $A_{train}$  čiže predikcia (predpoveď) aké budú funkčné hodnoty pre vstupné vektory, ktoré sieť nikdy nevidela. Táto schopnosť UNS sa volá **zovšeobecňovanie** alebo **generalizácia**.

<sup>6</sup> 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 tréningovou množinou. Výsledná zmena váhy je súčtom zmien vyvolaných všetkými vstupnými vzormi.

TESTOVANIE	Aby sme túto schopnosť siete mohli testovať, nepoužívame všetky dáta, ktoré máme k dispozícii na tréning, ale rozdelíme si ich na <i>trénovaciu</i> a <i>testovaciu</i> množinu $A_{train}$ a $A_{test}$ (plné a prázdne krúžky na obr. 6.8). Testovacia množina sa nazýva aj <i>validačná</i> množina. Pomocou $A_{train}$ modifikujeme váhy a pomocou $A_{test}$ zisťujeme chybu zovšeobecňovania. Pracujeme s kumulovanou chybou $E$ (6.24) pre $A_{train}$ aj $A_{test}$ .
VALIDÁCIA	
ROZDELENIE DÁT	Je dôležité, aby sme dáta správne rozdelili. $A_{train}$ musí rovnomerne pokrývať daný interval. Zistilo sa, že v priebehu tréningu kumulovaná $E_{train}$ stále klesá, zatiaľ čo $E_{test}$ začne od určitého kroku rásť (obr. 6.9). Dochádza k javu, ktorý sa nazýva <b>preučenie</b> siete (angl. overfitting) alebo pretrénovanie, či premodelovanie dát.
PREUČENIE	

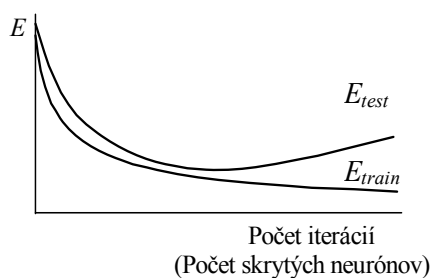
OBR.6.8.

APROXIMÁCIA  
FUNKCIE  
POMOCOU UNS



OBR.6.9.

VÝVOJ CHYBY  
POČAS  
TRÉNOVANIA



KLASIFIKÁCIA DO  $K$   
TRIED

Doprednú neurónovú sieť možno tréning 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 navrhne takto:  $\vec{d}^1 = (1,0,0)$ ,  $\vec{d}^2 = (0,1,0)$  a  $\vec{d}^3 = (0,0,1)$ , 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  $\vec{x}^p$ .

Pri tréningu 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  $N_{train}^{error} \leq N_{max}$ .

„DOUČANIE“

Ak chceme sieť doučiť novú triedu, musíme tréning začať odznova so všetkými triedami a vzormi. 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 tréning odznova.

ASOCIÁCIA

Pre každý typ úlohy možno vo všeobecnosti povedať, že viacvrstvová dopredná neurónová sieť sa učí asociovať vstupné vektory  $\vec{x}^p$  s výstupnými vektormi  $\vec{d}^p$ . Skryté neuróny pritom vykonávajú extrakciu príznakov, na základe ktorých je

EXTRAKCIA

PRÍZNAKOV	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	<p>Užitočnou metódou na zastavenie učenia, keď nevieme dopredu <math>\varepsilon</math> či <math>N_{max}</math>, alebo sa tieto ukazovatele po mnoho iterácií výrazne nemenia, je metóda nazvaná <b>skoré (optimálne) zastavenie</b> (Bishop, 1995):</p> <ol style="list-style-type: none"> <li>1. Rozdeľ dáta na 2 neprekrývajúce sa podmnožiny <math>A_{train}</math> a <math>A_{test}</math> (napr. náhodne vyber 75% vzorov do jednej a zvyšok do druhej).</li> <li>2. Po každom tréningovom cykle vypočítaj kumulovanú chybu na tréningovej a testovacej množine <math>E_{train}</math> a <math>E_{test}</math> (respektíve <math>N_{train}^{error}</math> a <math>N_{test}^{error}</math>, keď učíme sieť klasifikovať).</li> <li>3. Zastav učenie, keď <math>E_{test}</math> (resp. <math>N_{test}^{error}</math>) začne rásť.</li> </ol>
OPTIMÁLNY POČET SKRYTÝCH NEURÓNOV	<p><b>Selekcia modelu</b> znamená výber optimálneho počtu skrytých neurónov<sup>7</sup> (Bishop, 1995). Neplatí, že čím viac skrytých neurónov, tým lepšie (pozri obr. 6.9). Zvyčajným postupom je, že určíme chybu zovšeobecňovania <math>E_{test}</math> pre každý model (počet skrytých neurónov) a vyberieme ten model, pre ktorý je <math>E_{test}</math> minimálna. Korektnou štatistickou metódou je <b>vrstvová <math>k</math>-násobná prekrížená validácia</b> (angl. stratified <math>k</math>-fold cross-validation):</p> <ol style="list-style-type: none"> <li>1. Množinu všetkých dát <math>A</math> rozdelíme na <math>k</math> podmnožín, <math>A_{test}^1, A_{test}^2, \dots, A_{test}^k</math>, ktoré majú nulový prekrýv. Zvyčajne <math>k = 10</math>. Je dôležité jednotlivé podmnožiny vybrať tak, aby obsahovali približne také isté kvantitatívne zastúpenie tried ako pôvodná množina <math>A</math>.</li> <li>2. Každý model (počet skrytých neurónov) sa trénuje <math>k</math>-krát, a to na <math>k</math> tréningových množinách <math>A_{train}^i = A \setminus A_{test}^i</math> pre <math>i = 1, \dots, k</math>. Učenie možno zastaviť podľa metódy skorého zastavenia, keď <math>E_{test}^i</math> začína rásť.</li> <li>3. Po skončení všetkých tréningov, pre každý model vypočítame „cross“-validačný odhad presnosti klasifikácie tak, že           <math display="block">CV = \frac{1}{k} \sum_{i=1}^k V^i \quad (6.25)</math>           pričom pre klasifikačnú úlohu je <math>V^i = N_{test}^{error}</math>, t.j. počet chybných klasifikácií pre <math>i</math>-tu testovaciu množinu <math>A_{test}^i</math>. Pre aproximačnú úlohu je <math>V^i = E_{test}^i</math>.</li> <li>4. Vyberieme ten model (počet skrytých neurónov), pre ktorý je <math>CV</math> minimálny. Pre každý model môžeme vypočítať strednú kvadratickú odchýlku pre <math>CV</math> a interval spoľahlivosti.</li> </ol>
SPRIEMERŇOVA-	V kontexte aproximácie funkcií i klasifikácie sa ako optimalizovaná aproximácia

<sup>7</sup> Ukázalo sa, že pridávanie skrytých vrstiev výrazne nezlepšuje výkon doprednej siete. Stačí jedna skrytá vrstva. Treba však nájsť optimálny počet skrytých neurónov.

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

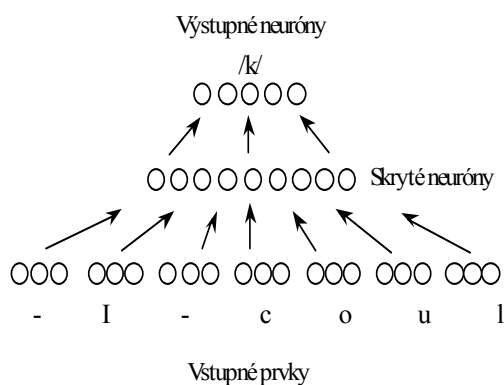
### Aplikácie dopredných neuronových sietí trébovaných spätným šírením chýb

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

NETTALK –  
ČÍTANIE  
ANGLICKÉHO  
TEXTU NAHLAS

Sejnowski a Rosenberg (1987) vytvorili neuronovú 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 neuronová sieť sa bola schopná naučiť väčšinu pravidiel a mnoho výnimiek z pravidiel výslovnosti. Na obr. 6.10 ilustrujeme použitú architektúru. Šípky 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. Tri ďalšie prvky kódovali diakritiku a hranice slova.

TRÉNOVANIE SIETE  
NETTALK

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 späťne 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.

#### INTERPRETÁCIA VÝSTUPU

Autori použili takú interpretáciu výstupu na priradenie do vzorovej fonémy, ktorú nazvali najlepší odhad. Určujúci bol najmenší uhol medzi skutočným a požadovaným výstupným vektorom. Prvá tréningová množina pozostávala z 1024 slov extrahovaných z detskej reči, kontinuálneho rečového prejavu. Keď bola sieť tréningovaná na detskej reči, po 50 epochách dosiahla 95%-nú úspešnosť na tréningovej množine a 78%-nú úspešnosť na testovacej množine pozostávajúcej zo 439 nových slov. Autori uvádzajú, že bolo fascinujúce počuť 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 (asi 10000 slov) sa jej už dalo rozumieť. Najčastejšie chyby boli v rozdieli vo výslovnosti „th“ v takých slovách ako „thesis“ a „these“.

#### VÝSLEDKY NETTALK

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

#### POŠKODENIE SYNAPTICKÝCH VÁH

V oboch prípadoch skúsili autori poškodiť synaptické váhy pridaním malého náhodného čísla z intervalu  $(-0,5; 0,5)$ . 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.

#### ODŠUMENIE

Ďalším rozšírením modelu by mohlo byť tréningovanie siete na odšumenie hovorenej reči. Tréningovú 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.

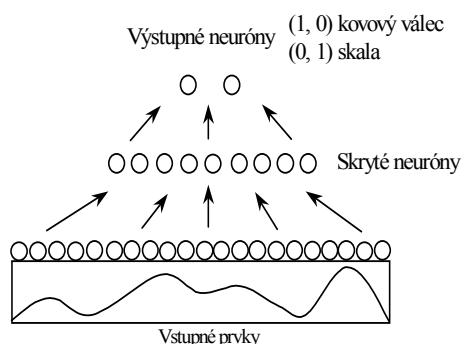
#### KLASIFIKÁCIA

Príkladom na použitie umelej neurónovej siete na klasifikáciu signálov je práca

PODMORSKÝCH  
SONAROVÝCH  
SIGNÁLOV

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  $\in (-0,3; 0,3)$ .

**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.

TRÉNOVANIE SIETE

Spektrum frekvencií sa rozdelilo na 60 rovnakých frekvenčných intervalov a intenzity sa normalizovali do intervalu  $(0; 1)$ , keďže neuróny mali unipolárnu sigmoidálnu aktivačnú funkciu. Číže 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 valec a 97 príkladov pre skalú. Z týchto sa pre každú sieť vybralo náhodne 16 príkladov na testovanie a zvyšok tvoril tréningovú množinu. Výsledky sa prezentujú ako priemer z 10 sietí, pričom každá bola inicializovaná inými náhodnými váhami.

VÝSLEDKY

Po 30 epochách tréningovania dosiahla sieť 100%-nú úspešnosť na tréningovej 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 tréningovej 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, nebyva to však pravidlom.

ALVINN - AUTO-  
NÓMNE POZEMNÉ  
VOZIDLO

Opis projektu nazvaného ALVINN (Autonomous Land Vehicle In a Neural Network) Pomerleau (1989) nám poskytne predstavu ako neurónovo riadiť pohybujúce sa vozidlá, roboty a pod.

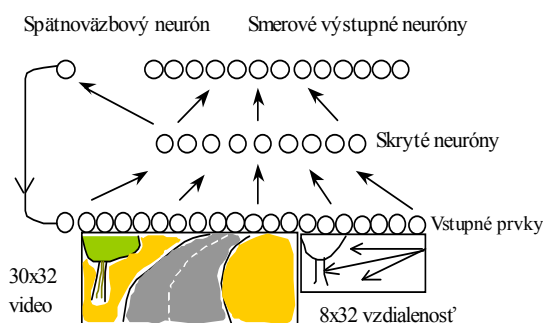
TRÉNOVANIE  
ALVINNU

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  $960 = 30 \times 32$  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“



Celkovo 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 necesta a zvyšných 45 neurónov kóduje smer pohybu.

#### ŠOFÉROVANIE

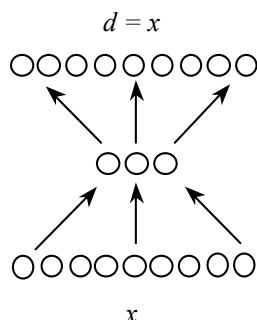
Najvyššia hodnota výstupu stredného neurónu indikuje priamy smer. Neuróny najviac vľavo a vpravo indikujú maximálne zatočenie doľava respektíve doprava. Ostatné neuróny reprezentujú diskretný uhol otočenia medzi nulovým a maximálnym uhlom. Pri trénovaní siete, treba samozrejme sieti poskytnúť informáciu 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

Jeden z neuronových projektov v AT&T Bell Laboratories bol zameraný na rozpoznávanie rukou písaných PSČ (LeCun a kol., 1991). Lokalizácia číslíc 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 predchádzajúcich máp. Tretia skrytá vrstva mala 30 neurónov a výstupná vrstva mala 10 neurónov, ktoré kódovali 10 číslíc systémom „one-hot“. Celkovo mala sieť 1256 neurónov a 64 660 spojení. Po 23 prezentáciách množiny 7291 skutočných ľuďmi napísaných číslíc, dosiahla sieť úspešnosť 98,6% na trénovacej množine a 95% na testovacej množine pozostávajúcej z 2007 nových číslíc. 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“ A  
KOMPRESIA DÁT



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 tréningu, máme na skrytej vrstve zakódované a skomprimované vstupy. Cottrell a kol. (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 vybrať tréningovú množinu a vhodnú architektúru siete. Tréning 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**<sup>8</sup> (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( $\lambda$ )** (TD z anglického *temporal difference*) (Sutton, 1988; Tesauro, 1990).

Nech vstupný vektor  $\bar{x}^{(t)} = (x_1, \dots, x_i, \dots, x_l)$  kóduje pozíciu  $P$  na hracom poli v diskretnom časovom kroku  $t$ . Pri kódovaní pozície môžeme postupovať napr. takto:

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

Majme dvojvrstvovú doprednú sieť s jedným výstupným neurónom. Majme sekvenciu pozícií (partiu)  $\bar{x}^{(1)}, \dots, \bar{x}^{(t)}, \dots, \bar{x}^{(T)}$  a tri požadované výstupy siete, ktoré

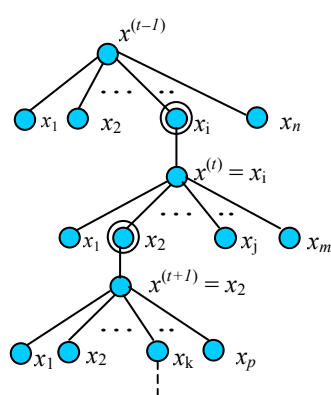
<sup>8</sup> Niekedy sa používa aj termín známokované učenie (Oravec a kol., 1998). V psychológii sa „reinforcement learning“ prekladá ako posilňovanie.

sa nazývajú **odmena** (angl. *reward*):

ODMENA (TREST)

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

**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 siete predstavujeme stovky (až tisícky) partií. Časť partií vedie k výhre, časť k prehre, respektíve k remíze. Po skončení tréningovania 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:

$$E = \frac{1}{2} \sum_{t=1}^T (r - o^{(t)})^2 \quad (6.28)$$

ZMENA VÁH

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:

$$\Delta w = -\alpha \frac{\partial E}{\partial w} = \alpha \sum_{t=1}^T (r - o^{(t)}) \frac{\partial o^{(t)}}{\partial w} \quad (6.29)$$

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

$$o^{(T+1)} - o^{(t)} = (o^{(T+1)} - o^{(T)}) + (o^{(T)} - o^{(T-1)}) + \dots + (o^{(t+1)} - o^{(t)}) \quad (6.30)$$

Po dosadení do (1.29) dostaneme

$$\Delta w = \alpha \sum_{t=1}^T \left[ \sum_{n=t}^T (o^{(n+1)} - o^{(n)}) \right] \frac{\partial o^{(t)}}{\partial w} \quad (6.31)$$

Preskupením pravej strany získame vzťah

$$\Delta w = \alpha \sum_{t=1}^T (o^{(t+1)} - o^{(t)}) \sum_{n=1}^t \frac{\partial o^{(n)}}{\partial w} \quad (6.32)$$

TD PRAVIDLO

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

$$\Delta w^{(t)} = \alpha (o^{(t+1)} - o^{(t)}) \sum_{n=1}^t \frac{\partial o^{(n)}}{\partial w} \quad (6.33)$$

TD( $\lambda$ ) PRAVIDLOTD( $\lambda$ ) je modifikáciou predchádzajúceho vzťahu tak, že

$$\Delta w^{(t)} = \alpha (o^{(t+1)} - o^{(t)}) \sum_{n=1}^t \lambda^{t-n} \frac{\partial o^{(n)}}{\partial w} \quad (6.34)$$

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 umelé neurónové siete tréované na aproximáciu funkcií a klasifikáciu objektov.

Pomocou TD( $\lambda$ ) sa dá hrať backgammon na veľmajstrovskej úrovni (Tesauro, 1990) alebo zahrať si s neurónovou sieťou dámu (Bahna, 1998). Existuje aj program na šach založený na kombinácii TD( $\lambda$ ) a algoritmu MINIMAX (Baxter a kol., 1997). Učenie s odmenou a trestom sa často aplikuje na riadenie robotov, pretože pri tréovaní im netreba presne hovoriť, čo majú robiť, ale len sa známujú sekvencie ich akcií.

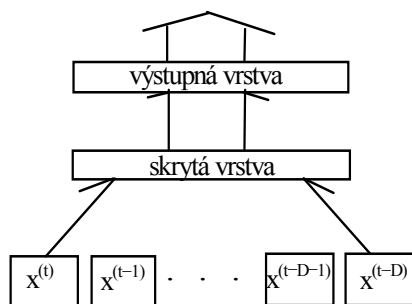
## 6.4 REKURENTNÉ SIETE A ICH UČENIE

ČASOVÁ  
ŠTRUKTÚRA DÁT

Predstavme si, že tréovacia 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  $D$ .

Ak máme šťastie, aj takéto jednoduché rozšírenie viacvrstvovej architektúry môže postihnúť časovú štruktúru skrytú v tréovacích dátach. Výhodou architektúry TDNN je možnosť tréovania 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 tréovacích dát.

**OBR.6.15.**  
NEURÓNOVÁ SIEŤ  
S OKNOM DO  
MINULOSTI

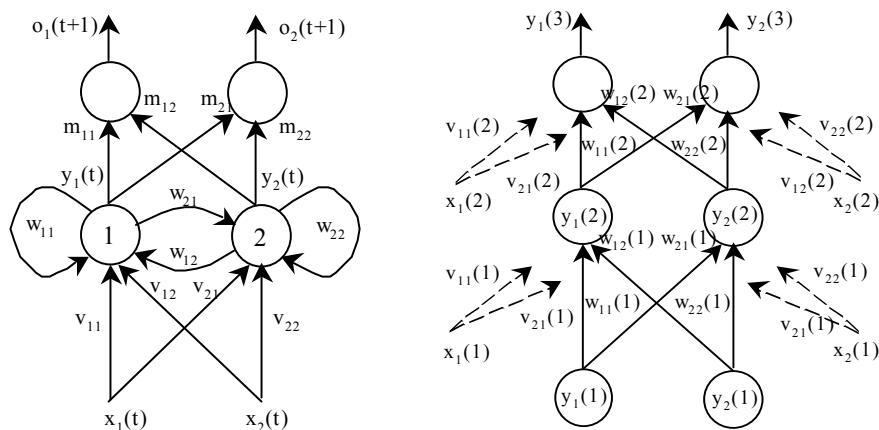


TDNN umožňuje viacvrstvovej doprednej sieti „okno do minulosti“, t.j. okrem momentálneho vstupu (v čase  $t$ ) „vidí“ sieť ešte aj vstupy z minulých  $D$  krokov (v časoch  $t-1, t-2, \dots, t-D$ ). Takúto sieť je možné trénovať klasickou procedúrou spätného šírenia chýb, pričom je dôležité zachovať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

Treba podotknúť, že aj v prípade, keď TDNN je schopná reprezentovať časovo-priestorovú štruktúru dát, nie je jednoduché len na základe tréningovej 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).

Na to, aby sme rozšírili možnosti pri spracovaní časových postupností dát na okno do minulosti premenlivej konečnej dĺžky, musíme zaviesť novú architektúru siete a nový pohľad na jej tréning. Novou architektúrou je tzv. **rekurentná neurónová sieť** (RNS) (pozri obr. 6.16 vľavo).

**OBR.6.16.**  
REKURENTNÁ  
NEURÓNOVÁ SIEŤ:  
VĽAVO  
NEROZVINUTÁ,  
VPRAVO  
ROZVINUTÁ V ČASE



SPÄTNÉ ŠÍRENIE  
CHÝB V ČASE

Novým algoritmom tréningu RNS je **spätné šírenie chýb v čase** (angl. Back-Propagation Through Time, BPTT). Úlohou je natréningovať 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. Požadovaný výstup sa sieti poskytne na konci každej postupnosti (napríklad zapni/nezapni alarm). Jednotlivé postupnosti môžu mať premenlivú dĺžku  $T$ .

## BPTT

Pri BPTT sa RNS rozvíja v čase, t.j. má toľko skrytých vrstiev koľko je vstupov  $T$  v jednej postupnosti. Obr. 6.16 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  $t = 1$  nastavujú na hodnotu 0,5. Takáto rozvinutá RNS sa potom trénuje ako obyčajná dopredná sieť s  $T$  skrytými vrstvami (Rumelhart a kol., 1986). Skutočný výstup siete počítame takto

$$o_k^{(T+1)} = f\left(\sum_{j=1}^J m_{kj}^{(T+1)} y_j^{(T+1)}\right) \quad \text{kde} \quad y_j^{(t+1)} = f\left(\sum_{i=1}^J w_{ji}^{(t)} y_i^{(t)} + \sum_{i=1}^I v_{ji}^{(t)} x_i^{(t)}\right) \quad (6.35)$$

V čase  $T+1$  poznáme aj požadovaný výstup  $d^{(T+1)} \in \{(1,0), (0,1)\}$ . Požadovaný výstup znamená, či postupnosť patrí do danej množiny alebo nepatrí. Môžeme použiť kódovanie „one-hot“. Váhy siete upravujeme až v kroku  $T+1$  a to tak, aby sme minimalizovali chybovú funkciu

## CHYBOVÁ FUNKCIA

$$E(T+1) = \frac{1}{2} \sum_{k=1}^K (d_k^{(T+1)} - o_k^{(T+1)})^2 \quad (6.36)$$

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

$$\Delta m_{kj}^{(T+1)} = -\alpha \frac{\partial E(T+1)}{\partial m_{kj}} = \alpha \delta_k^{(T+1)} y_j^{(T+1)} \quad (6.37)$$

kde

$$\delta_k^{(T+1)} = (d_k^{(T+1)} - o_k^{(T+1)}) f'(net_k^{(T+1)}) \quad (6.38)$$

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

$$\Delta w_{hj}^{(T)} = -\alpha \frac{\partial E(T+1)}{\partial w_{hj}} = \alpha \delta_h^{(T)} y_j^{(T)}; \delta_h^{(T)} = \left(\sum_{k=1}^K \delta_k^{(T+1)} m_{kh}^{(T+1)}\right) f'(net_h^{(T)}) \quad (6.39)$$

$$\Delta v_{ji}^{(T)} = -\alpha \frac{\partial E(T+1)}{\partial v_{ji}} = \alpha \delta_j^{(T)} x_i^{(T)}; \delta_j^{(T)} = \left(\sum_{k=1}^K \delta_k^{(T+1)} m_{kj}^{(T+1)}\right) f'(net_j^{(T)}) \quad (6.40)$$

$$\Delta w_{hj}^{(T-1)} = -\alpha \frac{\partial E(T+1)}{\partial w_{hj}} = \alpha \delta_h^{(T-1)} y_j^{(T-1)}; \delta_h^{(T-1)} = \left(\sum_{j=1}^J \delta_j^{(T)} w_{jh}^{(T)}\right) f'(net_h^{(T-1)}) \quad (6.41)$$

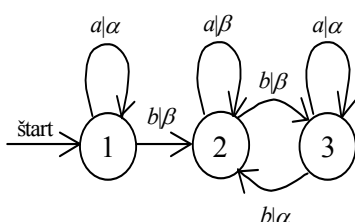
$$\Delta v_{ji}^{(T-1)} = -\alpha \frac{\partial E(T+1)}{\partial v_{ji}} = \alpha \delta_j^{(T-1)} x_i^{(T-1)}; \delta_j^{(T-1)} = \left(\sum_{h=1}^J \delta_h^{(T)} w_{hj}^{(T)}\right) f'(net_j^{(T-1)}) \quad (6.42)$$

A tak ďalej pre všetky časy. Výsledné zmeny váh na konci postupnosti sú rovné

$$\Delta w_{hj}^{(T+1)} = \frac{\sum_{t=1}^T \Delta w_{hj}^{(t)}}{T} \quad \text{a} \quad \Delta v_{ji}^{(T+1)} = \frac{\sum_{t=1}^T \Delta v_{ji}^{(t)}}{T} \quad (6.43)$$

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

OBR.6.17.

MEALYHO  
AUTOMAT

Príkladom, kedy nemusí ani TDNN ani BPTT stačiť na postihnutie časovej štruktúry dát sú postupnosti generované automatmi (obr. 6.17). 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

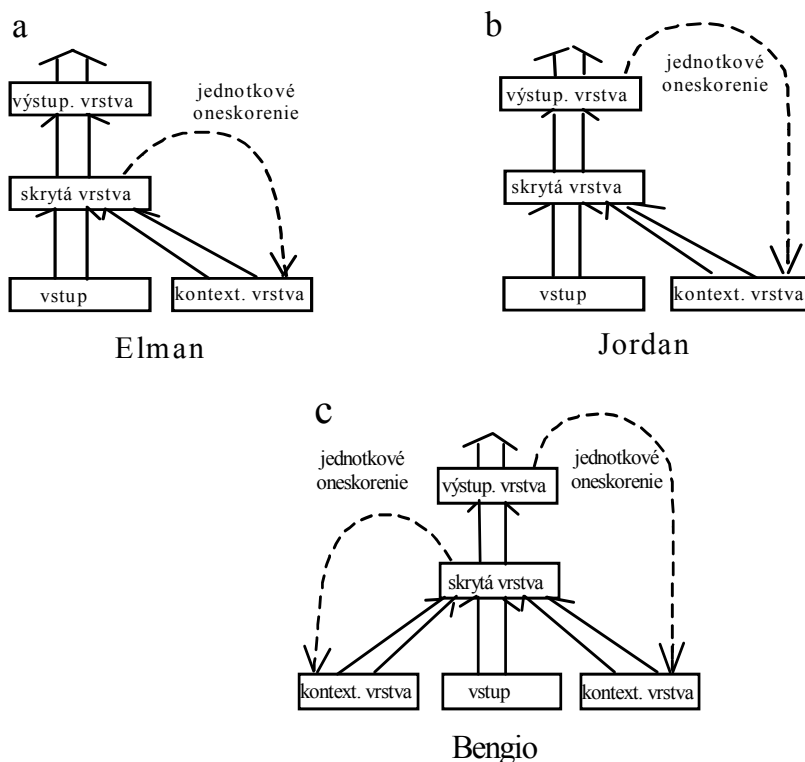
Takáto **stavová reprezentácia časového kontextu** predkladaných vzoriek môže byť omnoho úspornejšia ako reprezentácia časového kontextu pomocou okna do minulosti a niekedy aj nevyhnutná. Môže sa totiž stať, že by sme potrebovali neobmedzene dlhé okno do minulosti. Ak by sme boli v stave 1 automatu na obr. 6.17, môže prísť ľubovoľný počet vstupov  $a$  a asociovaný výstup je  $\alpha$ . To isté platí aj o stave 3. Podstatný rozdiel je však vo výstupe asociovanom so vstupom  $b$ . Ten je  $\beta$ , v prípade stavu 1 a  $\alpha$  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 tréovacej množiny reprezentujúcej časovo-priestorovú štruktúru popísanú automatom na obr. 6.17 bude architektúra TDNN nevyhovujúca, takisto ako aj RNS tréovaná pomocou BPTT.

REKURENTNÉ  
NEURÓNOVÉ SIEŤE

Ukázalo sa, že isté typy RNS (obr. 6.18) sú schopné vytvoriť si stavovú reprezentáciu časového kontextu v dátach (Tiňo a Šajda, 1995). Vo všeobecnosti možno za rekurentnú sieť považovať akúkoľvek neurónovú sieť, 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. 6.18). Toto kopírovanie sa deje s tzv. jednotkovým oneskorením. neurónové siete sa takto rozširujú 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 tréovacieho 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. 6.18a navrhol Elman (1990). Kontextová vrstva obsahuje kópie aktivácií skrytých neurónov z predošlého kroku. Autorom siete 6.18b je Jordan (1989). Obr. 6.18c predstavuje kombináciu predchádzajúcich modelov (Bengio a kol., 1990).

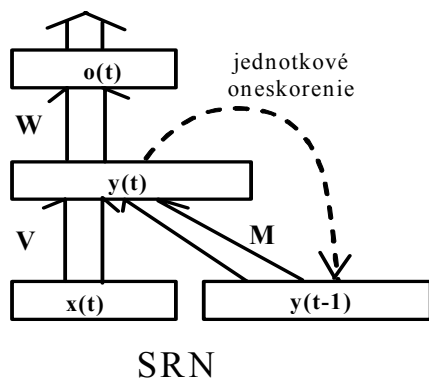
**OBR.6.18.**  
ARCHITEKTÚRY  
REKURENTNÝCH  
SIETÍ



REKURENTNÉ  
UČENIE V  
REÁLNOM ČASE

RNS sa trénujú na predikciu nasledujúceho požadovaného vektora v postupnosti pomocou metódy známej ako **rekurentné učenie v reálnom čase** (angl. Real Time Recurrent Learning, RTRL) (Williams a Zipser, 1989). Uvedieme hlavné rovnice tohto postupu, a to pre Elmanovu rekurentnú sieť ilustrovanú na obr. 6.18a a podrobnejšie rozkreslenú na obr. 6.19. Táto architektúra sa volá jednoduchá rekurentná sieť (angl. Simple Recurrent Network, SRN).

**OBR.6.19.**  
ELMANOVA  
JEDNODUCHÁ  
REKURENTNÁ SIET'



Aktivity výstupných neurónov sú vyjadrené ako

$$o_k^{(t)} = f(\text{net}_k^{(t)}) = f\left(\sum_{j=1}^J w_{kj} y_j^{(t)}\right) \quad (6.46)$$

aktivity skrytých neurónov ako

$$y_j^{(t)} = f(\text{net}_j^{(t)}), \quad (6.47)$$

kde

$$\text{net}_j^{(t)} = \sum_{i=1}^I v_{ji} x_i^{(t)} + \sum_{i=1}^J m_{ji} y_i^{(t-1)} \quad (6.48)$$

V čase  $t$  je na vstupe vektor  $\bar{x}^{(t)} = (x_1^{(t)}, x_2^{(t)}, \dots, x_I^{(t)})$ , skutočný výstup siete je vektor



aktivácií výstupných neurónov  $\bar{o}^{(t)} = (o_1^{(t)}, o_2^{(t)}, \dots, o_K^{(t)})$  a požadovaný výstup je  $\bar{d}^{(t)} = (d_1^{(t)}, d_2^{(t)}, \dots, d_K^{(t)})$ . Požadovaným výstupom môže byť nasledujúci vektor v postupnosti. Definujeme chybovú funkciu ako

$$E^{(t)} = \frac{1}{2} \sum_{k=1}^K (d_k^{(t)} - o_k^{(t)})^2 \quad (6.49)$$



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

$$\Delta w_{kj}^{(t)} = -\alpha \frac{\partial E^{(t)}}{\partial w_{kj}^{(t)}} \quad \Delta v_{ji}^{(t)} = -\alpha \frac{\partial E^{(t)}}{\partial v_{ji}^{(t)}} \quad \Delta m_{ji}^{(t)} = -\alpha \frac{\partial E^{(t)}}{\partial m_{ji}^{(t)}} \quad (6.50)$$

Pri výpočte týchto derivácií treba brať do úvahy architektúru siete (Tiňo, 1997). Ako vidíme, podstata učenia je opäť spätné šírenie chýb.

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

$$\Delta w_{kj}^{(t)} = \alpha \delta_k^{(t)} y_j^{(t)} = \alpha (d_k^{(t)} - o_k^{(t)}) f'_k(\text{net}_k^{(t)}) y_j^{(t)} \quad (6.51)$$

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

$$\Delta v_{ji}^{(t)} = \alpha \sum_{k=1}^K \left[ \delta_k^{(t)} \sum_{h=1}^J w_{kh} \frac{\partial y_h^{(t)}}{\partial v_{ji}^{(t)}} \right] \quad \text{resp.} \quad \Delta m_{ji}^{(t)} = \alpha \sum_{k=1}^K \left[ \delta_k^{(t)} \sum_{h=1}^J w_{kh} \frac{\partial y_h^{(t)}}{\partial m_{ji}^{(t)}} \right] \quad (6.52)$$

kde

$$\frac{\partial y_h^{(t)}}{\partial v_{ji}^{(t)}} = f'(\text{net}_h^{(t)}) \left[ x_i^{(t)} \delta_{jh}^{Kron.} + \sum_{l=1}^J m_{hl} \frac{\partial y_l^{(t-1)}}{\partial v_{ji}^{(t)}} \right] \quad (6.53)$$

$$\frac{\partial y_h^{(t)}}{\partial m_{ji}^{(t)}} = f'(\text{net}_h^{(t)}) \left[ x_i^{(t)} \delta_{jh}^{Kron.} + \sum_{l=1}^J m_{hl} \frac{\partial y_l^{(t-1)}}{\partial m_{ji}^{(t)}} \right] \quad (6.54)$$

kde  $\delta_{jh}^{Kron.}$  je Kroneckerova delta, pre ktorú platí  $\delta_{jh}^{Kron.} = 1$ , ak  $j=h$ , a inak,  $\delta_{jh}^{Kron.} = 0$ .

Pomocou rekurentných vzťahov (6.53–6.54) je možné v každom kroku tréningu nanovo prepočítať potrebné parciálne derivácie. Na začiatku tréningu pre  $t = 0$  je vhodné zvoliť tieto parciálne derivácie nulové.

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 tréningu 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 tréningu na predpoveď (predikciu) nasledujúceho člena postupnosti je tréningová 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 tréning predikcie na netriviálnych postupnostiach potrebujeme postupnosti, ktoré majú tisíce až desaťtisíce 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 novej 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. Sústreďme sa na analýzu aktivít rekurentných neurónov, napríklad v architektúre RNS Elmanovho typu (obr. 6.19). 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).

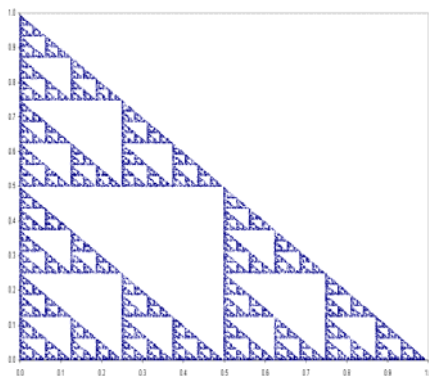
IFS

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

IFS ATRAKTOR

Nech  $X = \langle 0,1 \rangle$ . Majme jednu jedinečnú kontraktívnu transformáciu  $\omega$ , napr.  $\omega(x) = 0,5x + 0,5$ . Limitné správanie jednej kontraktívnej transformácie, teda jej aplikácia  $n \rightarrow \infty$  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** (obr. 6.20). Pre každý bod IFS atraktora definujeme tzv. IFS adresu. Táto adresa zodpovedá nekonečnej postupnosti transformácií, ktorá nás k nemu dovedla.

**OBR.6.20.**  
SIERPINSKÉHO  
TROJUHOLNÍK



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

$$\begin{aligned}\omega_1(x, y) &= (0.5x + 0.5, 0.5y) \\ \omega_2(x, y) &= (0.5x, 0.5y + 0.5) \\ \omega_3(x, y) &= (0.5x, 0.5y)\end{aligned}\quad (6.55)$$

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

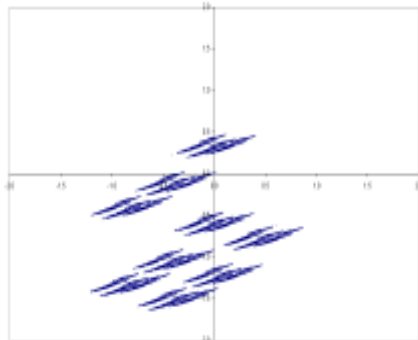
#### RNS AKO IFS

Správanie RNS pri spracovaní postupnosti symbolov sa dá vysvetliť pomocou IFS. Vnútrná dynamika (= časový vývoj aktivít rekurentných neurónov) sa riadia podľa rovnice (pozri obr. 6.19 a rovnice 6.47 a 6.48)

$$\bar{y}^{(t)} = f(\mathbf{M} \cdot \bar{y}^{(t-1)} + \mathbf{V} \cdot \bar{x}^{(t)}) \quad (6.56)$$

Vstupný vektor je  $\bar{x}$ , stavový vektor je  $\bar{y}$ .  $\mathbf{M}$  a  $\mathbf{V}$  sú matice váh a  $f$  je aktivačná funkcia. Výstupná vrstva nás zatiaľ nezaujímá. 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 6.56 tak môžeme 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.

**OBR.6.21.**  
FRAKTÁL



Z uvedených vzťahov vyplýva, že aj nenatréované RNS inicializované malými náhodnými váhami sa správajú ako IFS (Kolen, 1994). Tento jav sa nazýva **architekturný bias RNS** (Tiňo a kol., 2004). V stavovom priestore RNS sa vytvorí zaujímavá priestorová štruktúra, fraktál, odrážajúca časovú štruktúru postupnosti symbolov aj keď je sieť nenaučená (obr. 6.21, prevzatý z Čerňanský, 2001).

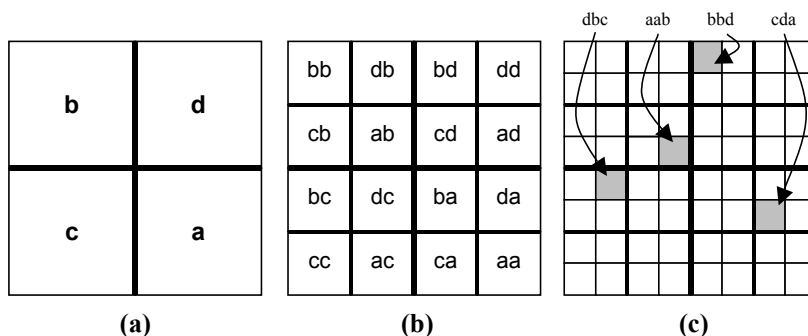
Vráťme sa teraz k pojmu IFS adresy. Uvažujme nasledujúcu sústavu IFS transformácií nad priestorom  $X = \langle 0,1 \rangle^2$

$$\begin{aligned}
 \omega_a(x, y) &= (0.5x + 0.5, 0.5y) \\
 \omega_b(x, y) &= (0.5x, 0.5y + 0.5) \\
 \omega_c(x, y) &= (0.5x, 0.5y) \\
 \omega_d(x, y) &= (0.5x + 0.5, 0.5y + 0.5)
 \end{aligned}
 \tag{6.57}$$

Atraktorom štvrtej transformácie je bod (1,1). Každá z týchto transformácií kontrahuje priestor  $X$  na štvrtinovú kópiu originálu vo svojej štvrtine štvorca (pozri nasledujúci obr. 6.22 prevzatý z Čerňanský, 2001).

OBR.6.22.

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 trérovacej (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 tréovanie RNS, keď sa vďaka architekturnému biasu časová štruktúra trérovacej postupnosti premietne do priestorovej štruktúry IFS atraktora aj v nenaučenej sieti. Po prvé, tréovaním sa výstupná vrstva učí asociovať históriu symbolov (klastre blízkych bodov v stavovom priestore) s nasledujúcim symbolom. 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 tréovania 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 a Šajda, 1995). RNS sú teda schopné vytvoriť si stavovú reprezentáciu len na základe príkladov slov generovaných konečno-stavovými automatmi.

POUŽITIE RNS NA  
POSTUPNOSTI  
REÁLNYCH DÁT

Reálne postupnosti dát ako ceny tovarov, ekonomické ukazovatele, a iné, sa obvykle kvantizujú na postupnosti symbolov, ktoré vyjadrujú kvantifikovaný relatívny pokles a nárast hodnoty sledovanej veličiny o určité percentá. Potom sa na tréningovanie RNS nahliada v zmysle prezentovanej teórie.

## 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 tréningovaných s učiteľom, ale ich tréningovanie je podstatne rýchlejšie (Bishop, 1995). RBF siete sú také, v ktorých je aktivačnou funkciou skrytých neurónov tzv. **radiálna bázo**vá funkcia (RBF) (Bishop, 1995; Oravec a kol., 1998). Najčastejšie sa používa gaussovská funkcia. Výstup  $n$ -tého výstupného neurónu je lineárny, t.j.

$$o_n = \sum_{j=1}^k w_{nj} \phi_j \quad (6.58)$$

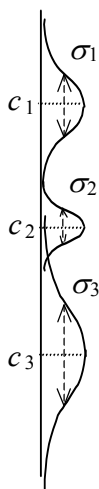
Čiže celkovo máme  $k$  skrytých neurónov. Výstup  $j$ -tého skrytého neurónu je gaussovská funkcia

$$\phi_j(\bar{x}) = \exp\left(-\frac{\|\bar{x} - \bar{c}_j\|^2}{\sigma_j^2}\right) \quad (6.59)$$

kde  $\sigma_j$  je šírka gaussiánu prislúchajúceho  $j$ -temu neurónu,  $\|\bar{x} - \bar{c}_j\|$  vyjadruje euklidovskú vzdialenosť medzi vstupným vektorom  $\bar{x}$  a centrom gaussiánu  $\bar{c}_j$ . Obr. 6.23 ilustruje tri gaussiány s rôznymi centrami a šírkami. Tréningovanie RBF sietí prebieha v troch krokoch:

OBR.6.23.

ILUSTRÁCIA RBF  
NA SKRYTEJ  
VRSTVE



### 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

$$A_{train} = \{(\bar{x}^1, \bar{d}^1)(\bar{x}^2, \bar{d}^2) \dots (\bar{x}^p, \bar{d}^p) \dots (\bar{x}^P, \bar{d}^P)\}.$$

Na určenie centier sa používajú 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  $k = 40 \div 60$ . 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) Nájďme 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:

$$\bar{c}_j(t+1) = \bar{c}_j(t) + \rho(t) \|\bar{x}^p(t) - \bar{c}_j(t)\|, \text{ kde } 0 \leq \rho(t) \leq 1. \quad (6.60)$$

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 širok RBF funkcií.

Šírky  $\sigma_j$  funkcií  $\phi_j$  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:

$$\sigma_j = \sqrt{(1/L) \sum_{l=1}^L \|\bar{c}_j - \bar{c}_l\|^2}, \quad (6.61)$$

kde  $L$  je počet najbližších susedov.

**III. Učenie váh výstupných neurónov.** Tieto váhy sa učia pomocou spätného šírenia chyby, teda


$$\Delta w_{nj} = -\alpha (\partial E / \partial w_{nj}), \quad (6.62)$$

kde  $E$  je chybová funkcia (6.10). Váhy meníme až po nájdení  $\bar{c}_j$  a  $\sigma_j$ .

## VÝHODY RBF SÍETÍ

Dopredné vrstvomé 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.

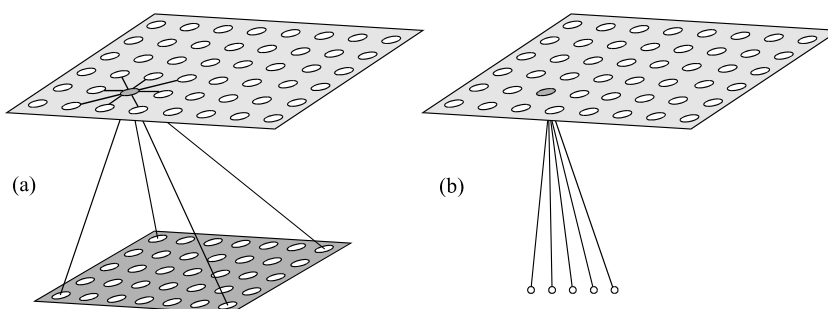
## 6.6 SOM – SAMOORGANIZUJÚCA SA MAPA

 **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 tréningu (Kohonen, 1995). Adaptácia váh prebieha len na základe tréningovej množiny, ktorá je sieti prezentovaná vo forme vstupných vzorov (vektorov), s binárnymi alebo reálnymi zložkami. Architektúra<sup>9</sup>, ale hlavne spôsob učenia SOM sú v súlade s neurobiologickými poznatkami týkajúcimi sa mozgovej kôry živočíchov.

Na obr. 6.24a 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 excitačné aj inhibičné. Vpravo, obr. 6.24b, je ešte viac zjednodušená architektúra, ktorá sa používa vo výpočtoch. Receptorová vrstva je nahradená vstupným vektorom a laterálne väzby neurónov tzv. funkciou okolia neuróna, súťaženie a kooperáciu neurónov, ako uvidíme ďalej. (Obrázok je prevzatý z Farkaš, 1997).

**OBR.6.24.**

ARCHITEKTÚRA  
SOM



ZACHOVANIE  
TOPOLÓGIE  
VSTUPU

Špecifickou črtou SOM je to, že po natrénovaní umožňuje realizovať **zobrazenie zachovávajúce topológiu** tréningovej 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 (napr. mapa povrchu tela).

<sup>9</sup> Aj predchádzajúce architektúry neurónových sietí sa nachádzajú v nervovom systéme živočíchov.

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 prispôbiť zmenému prostrediu. Proces modifikácie prebieha na základe podnetov prichádzajúcich z okolitého prostredia a je potrebný na vývoj normálnych topografických máp. Ako príklady možno uviesť vizuálne mapy (napr. mapa orientácií svetelných kontrastov) a sluchové mapy (napr. mapa frekvencií akustických stimulov).

HEBBOVO  
PRAVIDLO UČENIA

Samooorganizácia, 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 neurónových sietí (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 tréning 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 tréningovú množinu vzorov  $A_{train} = \{\bar{x}_p\}_{p=1}^P$ . Neuróny v sieti sú lineárne s nulovými prahmi, t.j.

$$o_i = \sum_{j=1}^m w_{ij} x_j = \bar{w}_i \bar{x} \quad (6.63)$$

NÁJDENIE VÍŤAZA

pričom  $m$  je dimenzia vstupu. Nech  $i = 1, \dots, n$  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.  $i^* = \operatorname{argmax}(\bar{w}_i \bar{x})$ , kde  $i^*$  je index víťazného neurónu (DP, „dot product“ verzia). V základnej, tzv. ED (angl. Euclidean Distance) verzii algoritmu SOM figuruje iná miera podobnosti: nájde sa neurón, ktorého váhový vektor je najbližšie k aktuálnemu vstupu v zmysle euklidovskej vzdialenosti:

$$i^* = \operatorname{argmin}_i \|\bar{x} - \bar{w}_i\| \quad (6.64)$$



## ADAPTÁCIA VÁH

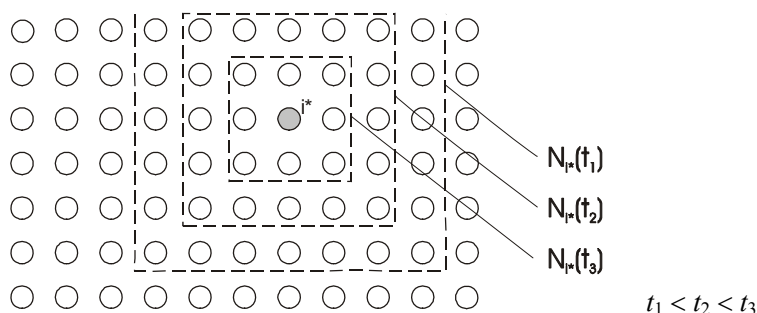
Po nájdení víťaza nasleduje adaptácia váh – **učenie**. To zabezpečí, že váhové vektory víťazného neurónu a jeho topologických susedov sa posunú smerom k aktuálnemu vstupu podľa vzťahu

$$\bar{w}_i(t+1) = \bar{w}_i(t) + \alpha(t) \cdot N(i^*, i) \cdot [\bar{x}(t) - \bar{w}_i(t)] \quad (6.65)$$

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  $N(i^*, i)$  (obr. 6.25) definuje rozsah *kooperácie* medzi neurónmi, t.j. koľko váhových vektorov prislúchajúcich neurónom v okolí víťaza bude adaptovaných, a do akej miery.

OBR.6.25.

ZMENŠUJÚCE SA  
OKOLIE VÍŤAZNÉHO  
NEURÓNA



Najjednoduchšou používanou funkciou je pravouhlé okolie

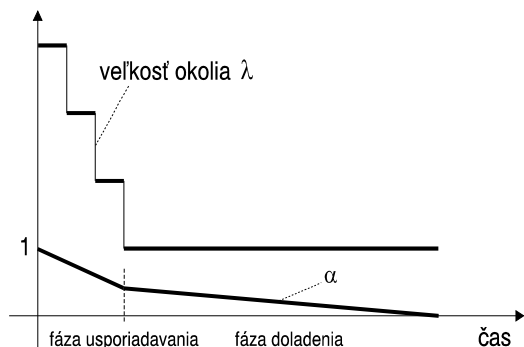
$$N(i^*, i) = \begin{cases} 1 & \text{ak } d_M(i^*, i) \leq \lambda(t) \\ 0 & \text{inak} \end{cases} \quad (6.66)$$

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 diskkrétne znižuje (priemer okolia odpovedá hodnote  $2\lambda(t)$ ) (obr. 6.25, 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

$$N(i^*, i) = \exp\left(-\frac{d_E^2(i^*, i)}{\lambda^2(t)}\right) \quad (6.67)$$

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 znižovanie okolia počas učenia. Na obr. 6.26 je spôsob ako upravovať veľkosť okolia a  $\alpha$ .

**OBR.6.26.**  
AKTUALIZÁCIA  
PARAMETROV  
UČENIA



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

Dôležité je, aby  $\alpha$  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 tréovania tak, že na fázu doladenia sa ponechá viac času ako na prvú fázu.



**Krok 1:** Zvolíme  $\alpha_0$ ,  $\lambda_0$  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  $\alpha$  a  $\lambda$  (obr. 6.26).

**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 tréovací cyklus (epocha), t.j. nový prechod cez  $A_{train}$ .

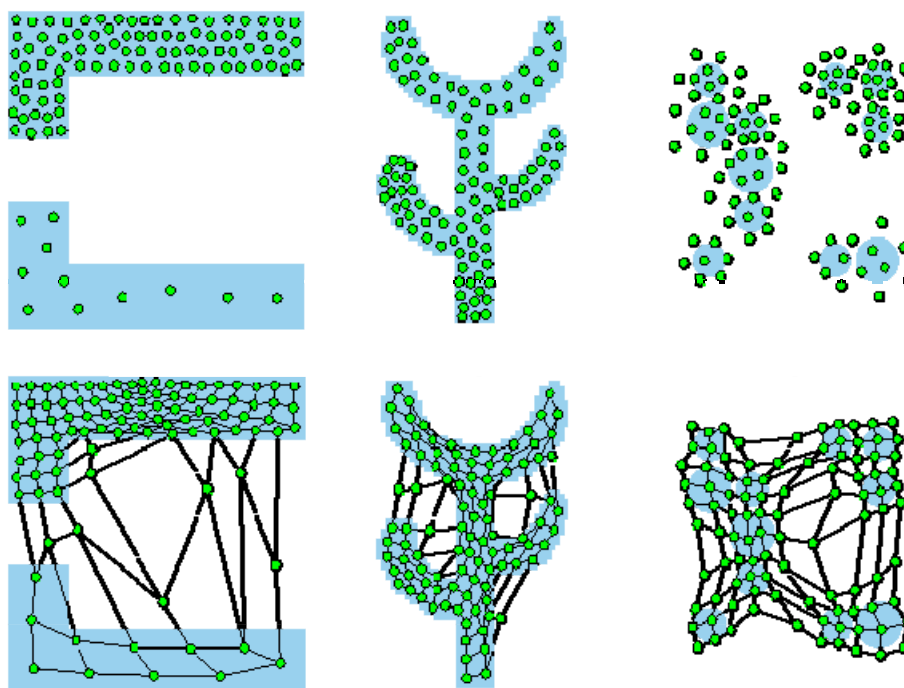
### Príklady topologických zobrazení pomocou SOM<sup>10</sup>

V nasledujúcich príkladoch chceme ilustrovať schopnosť SOM zachovávať topológiu vstupných dát, schopnosť klasterizácie vstupných dát a redukcie dimenzie dát. Horná trojica obrázkov na obr. 6.27 znázorňuje tri rôzne množiny vstupných dát. Každému znázornenému bodu zodpovedá 100 skutočných bodov. Tréovacie vektory tvoria súradnice bodov v rovine, t.j. reálne čísla  $(x, y)$ . Spodná trojica obrázkov znázorňuje váhy,  $\bar{w} = (w_x, w_y)$  všetkých  $10 \times 10$

<sup>10</sup> Všetky príklady sú prevzaté z (Farkaš, 1997).

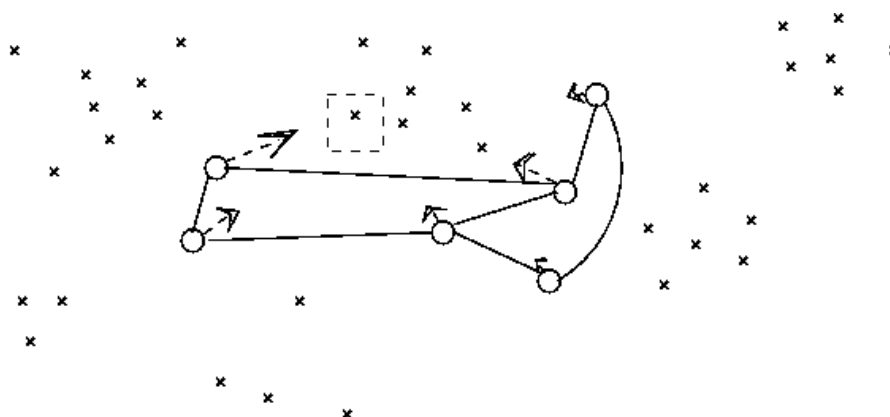
neurónov SOM po natrénovaní (10000 iterácií), ktoré kopírujú topológiu hornej trojice vstupných dát. Váhy susedných neurónov sú spojené čiarou. Tam, kde sú dáta hustejšie, tam sú váhové vektory susedných neurónov v SOM k sebe bližšie.

**OBR.6.27.**  
APROXIMÁCIA DÁT  
S ROZLIČNÝM  
ROZDELENÍM  
HUSTOTY



Obr. 6.28 ilustruje proces usporiadavania váhových vektorov SOM počas tréovania, ktoré postupuje takto: (1) Vyber náhodný vstupný bod, (2) nájdi najbližší váhový vektor a jeho susedov, (3) posuň ich smerom k tomuto bodu. Iteruj dovtedy pokiaľ sa viac váhové vektory neposúvajú. Veľmi jednoduché a funguje to.

**OBR.6.28.**  
USPORIADAVANIE  
VÁHOVÝCH  
VEKTOROV



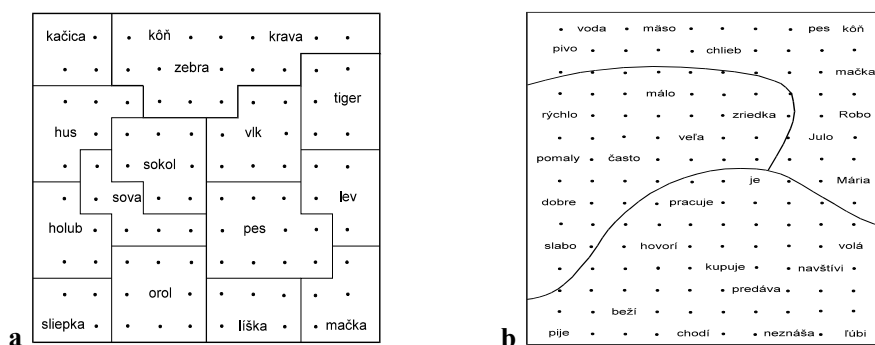
ZLÉ

Pri tréovaní SOM môžu vzniknúť niektoré špeciálne efekty, ktoré

NATRÉNOVANIE	<p>charakterizujú zlé natrénovanie siete:</p> <p>(a) Neúplné rozvinutie siete v dôsledku príliš rýchleho poklesu rýchlosti učenia <math>\alpha</math> v porovnaní so znižovaním okolia <math>\lambda</math>.</p> <p>(b) Tzv. motýlí efekt spôsobený príliš rýchlym zmenšením okolia <math>\lambda</math> v porovnaní s poklesom <math>\alpha</math>.</p> <p>(c) Tzv. „pinch“ efekt vznikajúci pri príliš pomalom znižovaní <math>\lambda</math>.</p> <p>Ak sa trénovanie vyvíja zle po každej inicializácii siete, treba zmeniť aktualizáciu parametrov učenia.</p>
VIACROZMERNÉ DÁTA	<p>Mapovanie viacrozmerných dát na 2D resp. 3D SOM sa dá využiť na redukciu dimenzie dát a na odhalenie vnútornej štruktúry ich distribúcie.</p>
VEKTOROVÁ KVANTIZÁCIA	<p>Na natrénovanú SOM sa môžeme pozeráť ako na <b>vektorový kvantifikátor</b>. Pri vektorovej kvantifikácii je úlohou nahradit' danú množinu (distribúciu) vstupných dát početne menšou množinou referenčných vektorov, nazývaných <b>prototypy</b>. V SOM hrajú úlohu prototypov váhové vektory. Takáto náhrada sa môže uplatniť napr. pri prenose údajov v telekomunikáciách, či v kompresii 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 <math>X</math> rozdelí na disjunktné oblasti, ktoré tvoria tzv. <b>Voronoiho mozaiku</b>.</p>
TOPOGRAFICKÉ MAPY ABSTRAKTNÝCH DÁT (SYMBOLOV)	<p>Vzťahy medzi symbolmi nie sú obvykle zistiteľné z ich kódových reprezentácií. SOM umožňuje využiť <b>kontext</b>, v ktorom sa symboly opakovane vyskytujú na vytvorenie topografických sémantických máp (obr. 6.29). Uvedieme dva príklady prevzaté z (Farkaš, 1997).</p> <p>V prvom prípade bol ku každému symbolu (meno zvierat'a) priradený vektor binárnych atribútov. Prítomnosť atribútu bola označená jednotkou, absencia nulou, ako veľkosť zvierat'a (malé, stredné, veľké), vonkajší popis tela (má 2 nohy, 4 nohy, srst', kopytá, hrivu, perie) a čo rado robí (loví, behá, lieta, pláva). Takýto 13-rozmerný vektor atribútov <math>\mathbf{x}_a</math> bol vygenerovaný pre každé zviera, pričom kódy zvierat <math>\mathbf{x}_s</math> boli zámerne vytvorené tak, aby nenesli žiadnu informáciu o vzájomnej podobnosti medzi zvieratami: každý vektor <math>\mathbf{x}_s</math> obsahoval samé nuly, až na jednu hodnotu <math>z</math>, ktorá figurovala na pozícii udávajúcej poradové číslo zvierat'a (1 až 16). Oba vektory boli zlúčené v jeden 29-rozmerný vektor <math>\mathbf{x} = [\mathbf{x}_s, \mathbf{x}_a]</math> charakterizujúci každé zviera, pričom hodnota <math>a</math> bola stanovená na <math>z = 0.2</math>, aby vplyv atribútovej časti vektora <math>\mathbf{x}</math> bol väčší ako vplyv symbolovej časti. Vektory <math>\mathbf{x}</math> boli napokon normované kvôli lepšej stabilizácii učenia. Počas trénovania bolo SOM prezentovaných 2000 náhodne vybraných vzorov <math>\mathbf{x}</math> z 16-prvkovej množiny. Proces určovania prototypov</p>
USPORIADANIE SYMBOLOV PODĽA ICH ATRIBÚTOV	

(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. 6.29a. Z toho vyplýva, že hoci reprezentácia vzájomných vzťahov podobnosti bola získaná vďaka prítomnosti atribútových častí počas tréningu, 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  $\mathbf{x} = [\mathbf{x}_s, \mathbf{x}_a]$ .

**OBR.6.29.**  
SÉMANTICKÉ  
MAPY



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 zahŕňovala podstatné mená, slovesá a príslovky. Generované tréningové vzory pozostávali zo zmysluplných trojslovných viet (napr. Robo pomaly beží, lev je mäso, atď.), teda symbol+dvojslovný kontext. Každý z troch symbolov bol nejakým spôsobom kódovaný ako 7-rozmerný vektor bez toho, aby sa v kódach 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. 6.29b. 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

MCCULLOCH A  
PITTS

Jedna z tém v histórii neurónových sietí súvisí s realistickým modelovaním nervových buniek a mozgu (Bower a 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 a Bishop, 1999).

V r. 1943 McCulloch a Pitts predstavili prvú umelú neurónovú sieť (McCulloch a 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

PERCEPTRÓNY	zloženou z takýchto elementov. Ak je táto sieť nekonečne veľká, tak je výpočtovo ekvivalentná univerzálnemu Turingovmu stroju.
UČENIE METÓDOU SPÄTNÉHO ŠÍRENIA CHÝB	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 <b>lineárne separovateľné problémy</b> (Minsky a Papert, 1969). Tento <b>problém sa vyriešil</b> až skoro o 20 rokov neskôr, v r. 1986, keď Rumelhart, Hinton a Williams zaviedli pravidlo učenia <b>metódou spätného šírenia chýb</b> (angl. <i>error back-propagation learning</i> ) pre viacvrstvové dopredné neurónové siete (Rumelhart a kol., 1986).
HOPFIELDOV MODEL	Špeciálnu kapitolu tvoria tzv. <b>Hopfieldove</b> alebo <b>atraktorové neurónové siete</b> (Amit, 1989; Beňušková, 1997). Sú založené na analógii s fyzikálnymi systémami. Tieto siete sa neučia, ale ich <b>pamäťové stavy sú predprogramované</b> 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 <b>učenia bez učiteľa</b> (angl. <i>unsupervised learning</i> ) a na systémy, ktoré sú <b>schopné samoorganizácie</b> . 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 <b>štatistické pravidelnosti</b> 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**. Na doplnenie vedomostí o trénovaní umelých neurónových sietí z domácej literatúry odporúčame (Sinčák a Andrejková, 1996; Kvasnička a kol., 1997; Oravec a kol., 1998).

NEUROBIOLÓGIA	Umelé neurónové siete sú inšpirované poznatkami o mozgu • Objekty sú v mozgu reprezentované redundantne a distribuovane • Učenie sprevádzajú <b>zmeny váh synáps</b> v mozgových neurónových sieťach • Umelé neurónové siete majú emergentné správanie • „Vidia štatistiku“ svojich stimulov
PERCEPTRÓN	Perceptrón má binárny výstup • Svojim učením <b>koriguje chybu na výstupe</b> • Rieši iba lineárne separovateľné problémy
DOPREDNÉ UNS RBF	Prvky dopredných neurónových sietí majú sigmoidálnu vstupno-výstupnú funkciu • Môžu mať aj inú prechodovú charakteristiku, napr. v tvare radiálnej bázevej 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 <b>univerzálny aproximátor</b> mnohorozmerných funkcií • Vie riešiť aj <b>nelineárne problémy</b>
SPÄTNÉ ŠÍRENIE CHÝB	

	<b>klasifikácie</b> • 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 siete povedali zakaždým správne riešenie, iba jej <b>výkon oznámujeme</b> • Aj tak sa naučí zovšeobecňovať • Je dobré kombinovať umelé neurónové siete s inými prístupmi UI
PREDIKCIA BUDÚCEHO VÝVOJA DÁT	Dopredná neurónová sieť s <b>oknom do minulosti</b> a rekurentné neurónové siete s <b>vnútornou pamäťou</b> dokážu reprezentovať časový kontext údajov • Učia sa veľmi pomaly • Na ich dobré natrénovanie potrebujeme tisícky opakovaní
SOM SAMOORGANIZÁCIA	Samoorganizujúca sa mapa zachováva topológiu vstupov • Biologicky motivované Kohonenovo učenie vedie ku klasterizácii vstupov a k redukcii dimenzie dát
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 <b>výpočtovú univerzálnosť</b> umelých neurónových sietí.

## KLÚČOVÉ POJMY

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

## CVIČENIA

- 6.1. 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 tréningovom kroku.

- 6. 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  $\leq 3$  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

- 6. 3.** Navrhňte a natrénujte doprednú dvojvrstvovú neurónovú sieť tak, aby vykonávala binárne násobenie 3 bitov 3 bitmi. Jeden príklad tréningového páru je vstup=011011 a požadovaný výstup=001001 (t.j.  $3 \times 3 = 9$ ). Celkovo je 64 tréningových párov.
- 6. 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 tréningovanie použite asi 20 vzorov. Testujte sieť na zovšeobecňovanie pomocou ďalších 20 hodnôt, na ktoré nebola natréňovaná.
- 6. 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.
- 6. 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.
- 6. 7.** Navrhňte si nejaký konečno-stavový automat. (a) Vytvorte si tréningovú 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 natréňovaná. (b) Trénujte RNS pomocou RTRL na tzv. „next-symbol prediction“ na slovách z automatu. Nechajte ju generovať pokračovanie testovacích slov.
- 6. 8.** Naprogramujte SOM a trénujte ju na 2D a 3D dátach rozličnej geometrickej štruktúry.

## LITERATÚRA

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

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



- BARNESLEY, M.: *Fractals Everywhere*. Academic Press, San Diego 1988.
- BAXTER, J. - TRIDGELL, A. - WEAVER, L.: *KnightCap: a chess program that learns by combining TD( $\lambda$ ) with MINIMAX search*. Australian National Univ., Canberra 1997.
- BEŇUŠKOVÁ, L.: Hopfieldov model. In: V. Kvasnička et al.: *Úvod do teórie neuronových sietí*, Iris, Bratislava 1997, pp. 190-236.
- BEŇUŠKOVÁ, L.: Kognitívna neuroveda. In: J. Rybár, L. Beňušková, V. Kvasnička (Eds.): *Kognitívne vedy*, Kalligram, Bratislava 2002, pp. 47-104.
- BENGIO, Y. - CARDIN, R. - DeMORI, R.: Speaker independent speech recognition with neural networks and speech knowledge. In: D.S. Touretzky (Ed.): *Advances in Neural Information Processing Systems II*, Morgan Kaufmann, San Mateo, CA 1990, pp. 218-225.
- BISHOP, C. M.: *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford 1995.
- BOWER, J. M. - BEEMAN, D.: *The Book of GENESIS: Exploring Realistic Neural Models with the General NEural Simulation System*. Springer Verlag, New York 1995.
- ČERŇANSKÝ, M.: *Recurrent neural networks*. Písomná práca k rigoróznej skúške, FEI STU, Bratislava 2001.
- COTTRELL, G. W. - MUNRO, P. - ZIPSER, D.: Image compression by back propagation: an example of extensional programming. In: N. E. Sharkey (Ed.): *Models of Cognition: A Review of Cognition Science*, Ablex, Norwood, NJ 1989.
- EFRON, B. - TIBSHIRANI, R. J.: *An Introduction to the Bootstrap*. Chapman & Hall, London 1993.
- ELMAN, J. L.: Finding structure in time. *Cognitive Science*, Vol. 14, 179-211. 1990.
- FARKAŠ, I.: Samoorganizujúce sa mapy. In: V. Kvasnička et al.: *Úvod do teórie neuronových sietí*, Iris, Bratislava 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, 1135-1140. 1988.
- HEBB, D.: *The Organization of Behavior*. John Wiley and Sons, New York 1949.
- HORNIK, K. - STINCHCOMBE, M. - WHITE, H.: Multilayer feedforward networks are universal approximators. *Neural Networks*, Vol. 2, 359-366. 1989.
- JEDLIČKA, P. - BEŇUŠKOVÁ, L. - MAČÁKOVÁ, J. - OSTATNÍKOVÁ, D.: In: I. Hulín (Ed.) *Patofyziológia*. Šieste vydanie, Slovak Academic Press (SAP), s.r.o, Bratislava 2002, pp. 1183-1199.
- JORDAN, M. I.: Serial order: a parallel distributed processing approach. In: J.L. Elman and D. E. Rumelhart (Eds.): *Advances in Connectionist Theory*, Erlbaum, Hillsdale 1989.
- KOHONEN, T.: *Self-Organizing Maps*. Springer Verlag, Berlin 1995.
- KOLEN, J.F.: The origin of clusters in recurrent network state space. In: *Proc. 16<sup>th</sup> Annual Conf. Cognitive Science Society*, Atlanta, GA, Aug. 13-16, 1994.
- KVASNIČKA, V. - BEŇUŠKOVÁ, L. - POSPÍCHAL, J. - FARKAŠ, I. - TIŇO, P. - KRÁL, A.: *Úvod do teórie neuronových sietí*. Iris, Bratislava 1997.
- KVASNIČKA, V. - POSPÍCHAL, J. - TIŇO, P.: *Evolučné algoritmy*. STU, Bratislava 2000.
- LeCUN, Y. - BOSER, B. - DENKER, J. S. et al.: Backpropagation applied to handwritten Zip code recognition. *Neural Computation*, Vol. 1, 541-551. 1989.
- MARŠALA, J.: *Systematická a funkčná neuroanatómia*. Osveta, Martin 1985.
- MAAS, W. - BISHOP, C. M.: *Pulsed Neural Networks*. MIT Press, Cambridge, MA 1999.
- McCULLOCH, W. S. - PITTS, W.: A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, Vol. 5, 115-137. 1943.
- MINSKY, M. - PAPER, S.: *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA 1969.
- ORAVEC, M. - POLEC, J. - MARCHEVSKÝ, S. et al.: *Neuronové siete pre číslicové spracovanie signálov*. Faber, Bratislava 1998.

- POMERLEAU, D. A.: ALVINN: An autonomous land vehicle in a neural network. In: D.S. Touretzky (Ed.): *Advances in Neural Information Processing Systems I*, Morgan Kaufmann, San Mateo, CA 1989, pp. 305-313.
- ROSENBLATT, F.: The Perceptron, a probabilistic model for information storage and organization in the brain. *Psychological Review*, Vol. 62, 386-408. 1958.
- RUMELHART, D. E. - HINTON, G. E. - WILLIAMS, R. J.: Learning internal representations by error propagation. In: D. E. Rumelhart and J. L. McClelland (Eds.): *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1., Foundations*, MIT Press / Bradford Books, Cambridge, MA 1986, pp. 318-363.
- SEJNOWSKI, T. - ROSENBERG, C.: Parallel networks that learn to pronounce English text. *Complex Systems*, Vol. 1, 145-168. 1987.
- SINČÁK, P. - ANDREJKOVÁ, G.: *Neurónové siete I, II*. Elfa, Košice 1996.
- SUTTON, R. S.: Learning to predict by the methods of temporal differences. *Machine Learning*, Vol. 3, 9-44. 1988.
- TESAURO, G.: Neurogammon wins computer olympiad. *Neural Computation*, Vol. 1, 321-323. 1990.
- TIŇO, P.: Rekurentné neurónové siete. In: V. Kvasnička et al.: *Úvod do teórie neurónových sietí*, Iris, Bratislava 1997, pp. 118-141.
- TIŇO, P. - ŠAJDA, J.: Learning and extracting initial Mealy machines with a modular neural network model. *Neural Computation*, Vol. 4, 822-844. 1995.
- TIŇO, P. - ČERŇANSKÝ, M. - BEŇUŠKOVÁ, L.: Markovian architectural bias of recurrent neural networks. *IEEE Transactions on Neural Networks*, Vol. 15, No. 1, 6-15. 2004.
- WEIBEL, A.: Modular construction of time-delay neural networks for speech recognition. *Neural Computation*, Vol. 1, 39-46. 1989.
- WILLIAMS, R. J. - ZIPSER, D.: A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, Vol. 1, 270-280. 1989.
- ZURADA, J.M.: *Introduction to Artificial Neural Systems*. West Publ. Comp., St. Paul, MN 1992.