6. Rekurentné neurónové siete

6.1 Prečo rekurentné siete?

V kapitole 5 venovanej viacvrstvovým neurónových sieťam a metódam ich trénovania sme videli, že tieto siete sú za určitých podmienok schopné naučiť sa asociovať vstupné vektory s požadovanými výstupnými vektormi. Zvyčajne požadujeme, aby sieť nefungovala len ako akási "look-up table" (kde sú dvojice vstup - požadovaný výstup "natvrdo" memorované), ale aby "inteligentne" reagovala aj na vstupy, ktoré jej pri trénovaní neboli ukázané. Inými slovami, boli by sme radi, ak by sieť správne "zovšeobecnila" trénovacie príklady. Rigoróznym úvahám o náročnosti trénovania, zovšeobecňovacím vlastnostiam sietí, kvalite trénovacej množiny (koľko trénovacích príkladov, aká je distribúcia trénovacích vzoriek, atď.) sa venuje disciplína nazvaná *teória učenia* (angl. *learning theory* [1]). Rozsah tejto práce neumožňuje podrobnejšie poznámky o tejto disciplíne. Obmedzíme sa len na konštatovanie, že viacvrstvová neurónová sieť zovšeobecní trénovacie vzorky tak, že nimi preloží nadplochu (pri lineárnych sieťach nadrovinu), ktorá je čo najmenej "zvlnená" (pozri obr. 5.21).

Táto požiadavka intuitívne reprezentuje staré pravidlo modelovania dát (tzv. *Occam's Razor* [2-3]), podľa ktorého by model nemal demonštrovať "štruktúry", ktoré nie sú obsiahnuté v dátach. Ak by napríklad trénovacia množina pozostávala z dvojíc (vstup, požadovaný výstup) ležiacich na nejakej nadrovine Π, intuitívne očakávame, že tam budú ležať aj dosiaľ nevidené asociačné dvojice (vstup, výstup). Lineárna sieť realizujúca zobrazenia vstupov na výstupy tak, že dvojice (vstup, výstup) ležia na Π, zrejme nevnáša do modelovania tie štruktúry, ktoré nemožno vystopovať v trénovacích dátach. Aj nelineárna viacvrstvová sieť zobrazujúca vstupy na výstupy, pričom dvojice (vstup, výstup) ležia na nadploche Ψ, ktorá je "mierne zvlnenou" verziou nadroviny Π, bude zrejme vyhovujúca. Naproti tomu, ak by nelineárna viacvrstvová sieť *presne* preložila trénovacími vzormi nadplochu Φ, ktorá by bola vysoko nelineárna (veľmi "zvlnená"), ťažko by sme mohli uveriť, že odpovede siete na vstupy neobsiahnuté v trénovacej množine budú mať niečo spoločné s tendenciou dát ležať na nadrovine Π. Voľne možno povedať, že takýto model vidí v dátach viac štruktúr, než v nich v skutočnosti je - fenomén známy pod menom *premodelovanie dát* (angl. *overfitting*, *overlearning*). V literatúre sú rozpracované metódy umožňujúce, aspoň do určitej miery, vyhnúť sa nástrahám premodelovania dát. Prípadných záujemcov odkazujeme na knihu [4].

Existujú však typy úloh, kde nelineárne viacvrstvové siete zlyhávajú, no nie v dôsledku nedostatku optimálnych trénovacích procedúr (pri nelineárnych sieťach dosiahneme len lokálne minimum na chybovom povrchu nad priestorom synaptických váh), či premodelovania dát. Inými slovami, nelineárne viacvrstvové siete by na tomto type úloh zlyhávali, aj keby sme dokázali spomenuté problémy spoľahlivo vyriešiť. Príčina tkvie v

samotnej podstate úlohy, ako je to napríklad pri úlohách, kde sa popri priestorových štruktúrach objavujú ešte aj časové štruktúry. Uvedieme si jednoduchý ilustračný príklad.

6.1.1 Príklad časovej štruktúry v dátach

Viacvrstvová sieť je schopná "naučiť sa" trénovaciu množinu pozostávajúcu napríklad z párov (vstup, požadovaný výstup)

$$A \rightarrow \alpha,\ B \rightarrow \beta,\ C \rightarrow \gamma,\ D \rightarrow \alpha$$

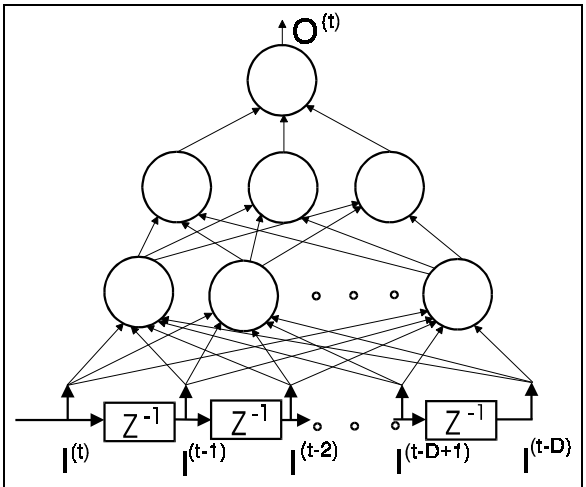
kde A,B,C,D sú vektory zo vstupného priestoru R^N a α,β,γ sú vektory z výstupného priestoru R^M. Sieť bude realizovať zobrazenie G: $R^N \rightarrow R^M$ tak, že $G(A) \cong \alpha$, $G(B) \cong \beta$, $G(C) \cong \gamma$ a $G(D) \cong \alpha$.

Trénovacie páry definujú určitú "priestorovú" štruktúru na priestore $R^N \times R^M$ párov (vstup, výstup) a táto štruktúra môže byť vystihnutá natrénovanou sieťou ako sme si spomenuli v úvode kapitoly. Predstavme si však, že trénovacia množina by mala nasledujúci tvar

$$A \rightarrow \alpha,\ B \rightarrow \beta,\ B \rightarrow \alpha,\ B \rightarrow \gamma,\ C \rightarrow \alpha,\ C \rightarrow \gamma, D \rightarrow \alpha$$

Vidíme, že k jednému vstupu môžeme mať viacero výstupov, v závislosti od *časového kontextu* tej-ktorej asociácie. Inými slovami, o výstupe siete by nemal rozhodovať len vstup siete, ale aj informácia o doterajšej histórii predkladaných vzoriek. Viacvrstvová sieť by mala byť rozšírená o možnosť reprezentovať časový kontext, aby tak mohla na základe predloženého vstupu lepšie rozhodnúť o výstupe. Architektonicky najjednoduchšie riešenie ponúka tzv. *neurónová sieť s časovým posunom* (angl. *Time Delay Neural Network,* TDNN) (obr. 6.1).

TDNN v podstate poskytuje viacvrstvovej sieti "okno do minulosti" - okrem momentálneho vstupu (v čase t) "vidí" sieť ešte aj vstupy z minulých D krokov (v časoch t-1, t-2, ..., t-D). Takúto sieť je možné trénovať klasickou procedúrou spätného šírenia (angl. *Back Propagation*, BP, pozri kapitolu 5), pričom je dôležité *zachovať poradie trénovacích vzoriek* v trénovacej množine.



Obrázok 6.1. Architektúra TDNN. Okrem aktuálneho prvku časovej postupnosti vstupov (v čase t), sú viacvrstvovej sieti prezentované aj vstupy z predošlých D krokov diskrétneho času. Bloky Z^{-1} sú oneskorovacie prvky.

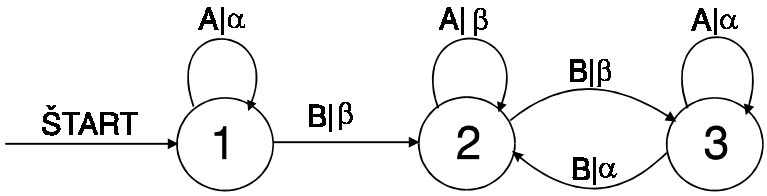
Ak máme šťastie, aj takéto jednoduché rozšírenie viacvrstvovej architektúry môže priniesť úspech a sieť typu TDNN je schopná popri priestorovej štruktúre postihnúť aj časovú štruktúru skrytú v trénovacích dátach. Výhodou architektúry TDNN je pomerná jednoduchosť a možnosť trénovania klasickou procedúrou BP vhodnou pre zvyčajné viacvrstvové siete. Nevýhodou tejto architektúry je, že spôsob reprezentácie časového kontextu nemusí byť dostatočne silný na zvládnutie časovej štruktúry trénovacích dát. Treba podotknúť, že aj v prípade, keď TDNN je schopná reprezentovať časovo-priestorovú štruktúru dát, nie je jednoduché len na základe trénovacej množiny správne odhadnúť dĺžku D "okna do minulosti" (viac podrobností nájde prípadný záujemca v [5-6]). Napriek tomu architektúra TDNN našla uplatnenie v mnohých oblastiach pracujúcich s časovo-priestorovými štruktúrami, napríklad v robotike, rozpoznávaní reči, atď. [7-10].

Pokúsme sa teraz odpovedať na otázku, kedy je architektúra TDNN apriori nevhodná na reprezentáciu časovo-priestorovej štruktúry trénovacích dát. Pre jednoduchosť si predstavme, že máme len konečnú množinu možných vstupných vektorov (napríklad $A,B \in R^N$) a konečnú množinu výstupných vektorov (napríklad $\alpha,\beta \in R^M$). Potom môžeme vstupy aj výstupy reprezentovať symbolmi z nejakej konečnej abecedy. Predpoklad architektúry TDNN, že na úvahu o možnom výstupe v čase t nám postačí informácia o terajšom vstupe a D predošlých vstupoch je analogický predpokladu stacionárneho markovovského procesu rádu D+1, kde pravdepodobnosť symbolu v reťazci závisí len od D+1 jeho bezprostredných predchodcov.

Predstavme si však, že proces reprezentovaný trénovacou množinou

$$A\rightarrow\alpha,\ A\rightarrow\alpha,\ B\rightarrow\beta,\ B\rightarrow\beta,\ B\rightarrow\alpha,\ A\rightarrow\beta,\ A\rightarrow\beta,\ A\rightarrow\beta$$

je popísaný Mealyho automatom [11,12} na obr. 6.2.



Obrázok 6.2. Mealyho automat popisujúci asociačnú úlohu s časovým kontextom. V závislosti od momentálneho stavu sú k vstupom *A,B* jednoznačne určené ich asociačné výstupy.

Spracovanie vstupného slova automatom sa začína v stave 1 označenom šípkou ŠTART. Po príchode vstupného symbolu $V \in \{A,B\}$ sa presunieme do nového stavu z množiny stavov {1,2,3} pozdĺž šípky prislúchajúcej vstupnému symbolu *V*, pričom so symbolom *V* asociujeme výstup $W \in \{\alpha,\beta\}$ podľa pravidla $V|W$. Teda po príchode symbolu *A* sa z počiatočného stavu 1 dostávame slučkou späť do stavu 1 a asociovaným výstupom je α. To sa zopakuje aj po opätovnom príchode vstupu *A*. Avšak vstup *B* nás prenesie do stavu 2 a príslušný asociovaný výstup je β, atď...

Stavy automatu kódujú históriu vstupných vektorov, aby sme mohli vždy bez váhania odpovedať na otázku, čo bude asociovaný výstup k danému vstupu pri danej histórii predkladaných vstupov. Vidíme, že 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 potenciálne neobmedzene dlhé okno do minulosti. Ak by sme boli v stave 1 automatu na obr. 6.2, môže prísť ľubovoľný počet vstupov *A* a asociovaný výstup je α. To isté patrí aj o stave 3. Podstatný rozdiel je však vo výstupe asociovanom so vstupom *B*. Ten je β, v prípade stavu 1 a α v prípade stavu 3. Nie je možné zvoliť žiadne konečné *D,* aby za každých okolností bolo možné na základe minulých vstupov rozhodnúť o výstupe asociovanom k vstupu *B*. Zrejme pre dobré zovšeobecnenie trénovacej množiny reprezentujúcej časovo-priestorovú štruktúru popísanú automatom na obr. 6.2 bude architektúra TDNN nevyhovujúca.

6.1.2 Predbežný príklad rekurentnej neurónovej siete

Inšpirovaní predchádzajúcimi úvahami uveďme architektúru neurónovej siete (obr. 6.3) zloženej z dvoch viacvrstvových sietí, a to

- *asociačnej siete* - realizujúcej asociáciu výstupu s daným vstupom na základe "vnútornej pamäti" siete a

- *stavovej siete* - realizujúcej kódovanie doterajšej histórie vstupov predložených sieti.

Architektúra bola navrhnutá v [13]. Obe viacvrstvové siete zdieľajú spoločnú vstupnú vrstvu, ktorá sa skladá zo vstupných neurónov zabezpečujúcich prekopírovanie vstupného vektora $\boldsymbol{I}^{(t)} = \left(I_1^{(t)}, I_2^{(t)}, \ldots, I_n^{(t)}, \ldots, I_N^{(t)}\right)$ v čase t do siete a zo "stavových" neurónov, ktorých aktivácie v čase t tvoria *stav siete* $\boldsymbol{S}^{(t)} = \left(S_1^{(t)}, S_2^{(t)}, \ldots, S_l^{(t)}, \ldots, S_L^{(t)}\right)$ kódujúci históriu predkladaných vstupov $I^{(\tau)}, \tau < t$. Asociačná sieť má tri vrstvy, okrem vstupnej vrstvy, ešte skrytú vrstvu neurónov druhého rádu, ktorých aktivácie v čase t sú počítané nasledovne (j je index neurónov v skrytej vrstve)

$$H_j^{(t)} = g\left(\sum_{l,n} Q_{j\,ln} S_l^{(t)} I_n^{(t)}\right) \tag{6.1}$$

kde g je obvyklá sigmoidálna aktivačná funkcia

$$g(u) = \frac{1}{1+e^{-u}} \tag{6.2}$$

Výstupná vrstva neurónov prvého poriadku reprezentuje výstup siete, ktorým sú v čase t aktivácie (m je index výstupného neurónu)

$$O_m^{(t)} = g\left(\sum_k V_{mk} H_k^{(t)}\right) \tag{6.3}$$

Stavová sieť sa skladá len z dvoch vrstiev. Úlohou druhej vrstvy je vypočítať reprezentácie nového časového kontextu (ktorý sa bude považovať za stav siete v čase t+1), ktorý vznikol príchodom vstupu $\boldsymbol{I}^{(t)}$

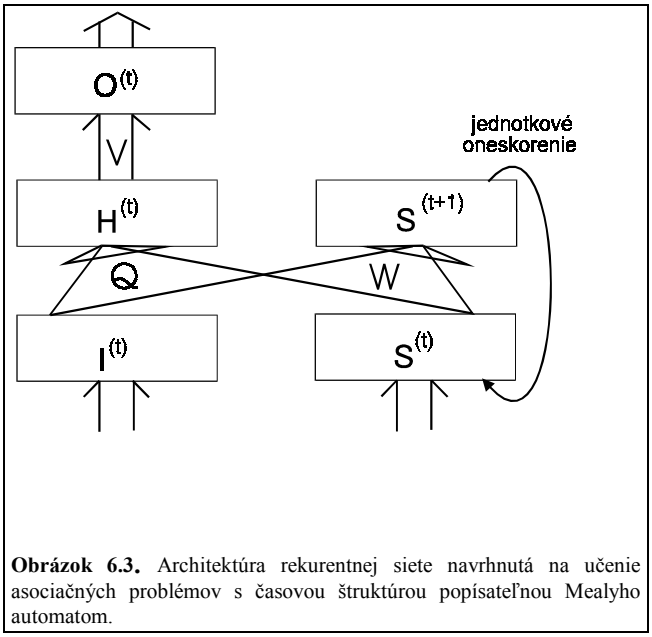
$$S_i^{(t+1)} = g\left(\sum_{l,n} W_{i\,ln} S_l^{(t)} I_n^{(t)}\right) \tag{6.4}$$

Ak sú vstupné vektory $\boldsymbol{I}^{(t)}$ z R^N, výstupné vektory $\boldsymbol{o}^{(t)}$ z R^M, stav siete je kódovaný L rozmerným vektorom z R^L (máme L stavových neurónov) a sieť má K neurónov v skrytej vrstve, potom v architektúre siete je L^2N váh $W_{i\,ln}$, KLN váh $Q_{j\,ln}$ a MK váh V_{mk}. V čase t sa na základe vstupu $\boldsymbol{I}^{(t)}$ a stavu $\boldsymbol{S}^{(t)}$ vypočíta stav siete v čase t+1, ktorý sa prekopíruje do časti vstupnej vrstvy v nasledujúcom kroku diskrétneho času.

Zrejme takáto architektúra siete je schopná reprezentovať časovo-priestorové štruktúry podobné štruktúre zobrazenej ako automat na obr. 6.2. Asociačnú *viacvrstvovú sieť sme* totiž *rozšírili o vnútornú pamäť*. Navyše, vo vrstve stavových neurónov si sieť môže utvoriť vlastnú stavovú reprezentáciu časového kontextu predkladaných vstupov.

Namieste je však otázka, ako učiť takýto typ siete, teda ako na základe *trénovacej množiny časovo usporiadaných asociačných dvojíc* (vstup, výstup) vyprodukovať váhy *W*, *Q*, *V*, ktoré zabezpečia "správnu" funkciu siete (zodpovedajúcu trénovacej množine). Treba si uvedomiť, že k dispozícii máme len dvojice (vstup, výstup) a preto aj stavové neuróny možno považovať za skryté. Z tohoto pohľadu máme v architektúre dva typy skrytých neurónov:

- *rekurentné* - vo vrstvách $S^{(t)}$, $S^{(t+1)}$,
- *nerekurentné* - vo vrstve $H^{(t)}$.



Obrázok 6.3. Architektúra rekurentnej siete navrhnutá na učenie asociačných problémov s časovou štruktúrou popísateľnou Mealyho automatom.

V čase t je na vstupe vektor $\boldsymbol{I}^{(t)} = \left(I_1^{(t)}, I_2^{(t)}, \ldots, I_N^{(t)}\right)$ a výstup siete je vektor $\boldsymbol{O}^{(t)}$ aktivácií výstupných neurónov $\boldsymbol{O}^{(t)} = \left(O_1^{(t)}, O_2^{(t)}, \ldots, O_M^{(t)}\right)$. Pokúsme sa zareagovať na vzniknutú disproporciu medzi skutočným výstupom siete $\boldsymbol{O}^{(t)}$ a želaným výstupom $\boldsymbol{D}^{(t)}$ zmenou váh, ktorá by ju zmiernila. Inšpirovaní myšlienkou procedúry BP z viacvrstvových sietí definujeme chybový funkcionál (účelovú funkciu, pozri formulu (5.2a))

$$E = \frac{1}{2}\sum_m \left(D_m^{(t)} - O_m^{(t)}\right)^2$$

a upravme váhy V, Q, W proporcionálne k inverzným gradientom

$$\Delta V_{mk} = -\alpha \frac{\partial E}{\partial V_{mk}} \tag{6.5}$$

$$\Delta Q_{j\,In} = -\alpha \frac{\partial E}{\partial Q_{j\,In}} \tag{6.6}$$

$$\Delta W_{i\,In} = -\alpha \frac{\partial E}{\partial W_{i\,In}} \tag{6.7}$$

kde α je "malá" kladná konštanta nazývaná *rýchlosť učenia* (angl. *learning rate*).

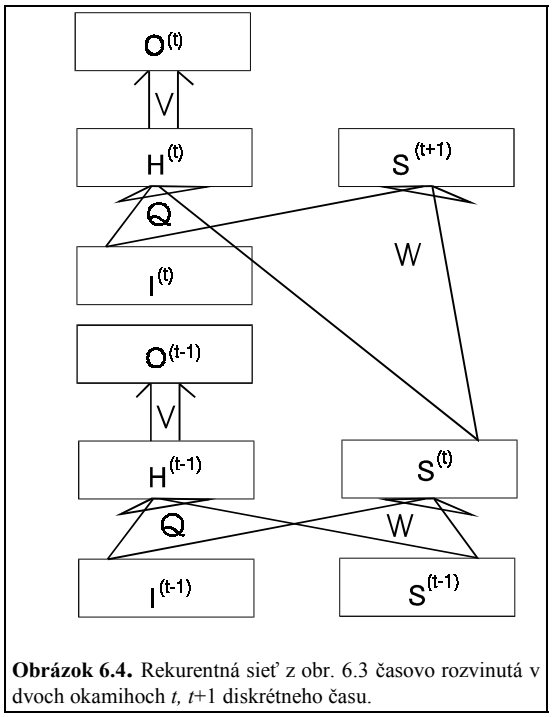
Pre podrobnejší výpočet parciálnych derivácií uvedených v (6.5), (6.6) a (6.7) je vhodné prekresliť obr. 6.3 na *sieť rozvinutú v čase* (konkrétne v časoch t, t+1) (obr. 6.4). Postupom analogickým postupu pri odvodení klasického algoritmu BP (pozri kapitolu 5) dostávame parciálne derivácie chybového funkcionálu E podľa váh V a Q

$$\frac{\partial E}{\partial V_{mk}} = \left(O_m^{(t)} - D_m^{(t)}\right) g'\left(\phi\left(O_m^{(t)}\right)\right) H_k^{(t)} \tag{6.8}$$

$$\frac{\partial E}{\partial \phi\left(O_m^{(t)}\right)} = \left(O_m^{(t)} - D_m^{(t)}\right) g'\left(\phi\left(O_m^{(t)}\right)\right) \tag{6.9}$$

$$\frac{\partial E}{\partial Q_{j\,In}} = S_l^{(t)} I_n^{(t)} g'\left(\phi\left(H_j^{(t)}\right)\right) \sum_m V_{mj} \frac{\partial E}{\partial \phi\left(O_m^{(t)}\right)} \tag{6.10}$$

kde g' je derivácia funkcie g, $g'(u) = g(u)(1 - g(u))$ a ϕ je inverzná funkcia k funkcii g. Konkrétne $\phi\left(O_m^{(t)}\right) = \sum_k V_{mk} H_k^{(t)}$ a $\phi\left(H_j^{(t)}\right) = \sum_{l,n} Q_{j\,In} S_l^{(t)} I_n^{(t)}$. Inverzná funkcia ϕ odpovedá tzv. postsynaptickému potenciálu neurónu (pozri kapitolu 1).



Obrázok 6.4. Rekurentná sieť z obr. 6.3 časovo rozvinutá v dvoch okamihoch *t, t*+1 diskrétneho času.

Výpočet parciálnych derivácií $\partial E/\partial W_{i\,ln}$ je troška zložitejší. V čase *t*-1 váha $W_{i\,ln}$ priamo ovplyvňuje iba aktiváciu $S_i^{(t)}$ *i*-teho stavového neurónu v budúcom kroku (v čase t) a teda

$$\frac{\partial E}{\partial W_{i\,ln}} = \frac{\partial E}{\partial S_i^{(t)}} \frac{\partial S_i^{(t)}}{\partial W_{i\,ln}} \tag{6.11}$$

Chyba *E* závisí od stavu $S_i^{(t)}$ prostredníctvom váh *V*,*Q* asociačnej siete, a preto

$$\frac{\partial E}{\partial S_i^{(t)}} = \sum_m \frac{\partial E}{\partial \phi\left(O_m^{(t)}\right)} \sum_k V_{mk}\, g'\left(\phi\left(H_k^{(t)}\right)\right) \sum_n Q_{kin} I_n^{(t)} \tag{6.12}$$

Stav $S_r^{(t)}$ *r*-tého stavového neurónu v čase *t* závisí priamo od váhy $W_{i\,ln}$ len ak *i*=*r*, no je dôležité si uvedomiť, že nepriamo závisí aj od všetkých ostatných váh $W_{i\,ln}$, keďže

$$S_r^{(t)} = g\left(\sum_{a,b} W_{rab} S_a^{(t-1)} I_b^{(t-1)}\right)$$

a stavy $S_b^{(t-1)}$ závisia len od váh *W* (a, pravda, vstupu $I^{(t-2)}$). Dostávame teda

$$\frac{\partial S_r^{(t)}}{\partial W_{i\,ln}} = g'\left(\phi\left(S_r^{(t)}\right)\right)\left[\delta_{ri} S_l^{(t-1)} I_n^{(t-1)} + \sum_{a,b} W_{rab} I_b^{(t-1)} \frac{\partial S_a^{(t-1)}}{\partial W_{i\,ln}}\right] \tag{6.13}$$

kde δ_{ri} je Kroneckerovo delta: $\delta_{ri} = 1$, pre $r = 1$, $\delta_{ri} = 0$, pre $r \neq i$. Pomocou rekurentného vzťahu (6.13) je možné v každom kroku trénovania nanovo prepočítať potrebné parciálne derivácie $\partial S_r^{(t)} / \partial W_{i\,ln}$. Tieto sa použijú v ďalšom kroku pre nový výpočet parciálnych derivácií podľa vzťahu (6.13). Na začiatku trénovania je vhodné zvoliť $\partial S_r^{(0)} / \partial W_{i\,ln}$ nulové.

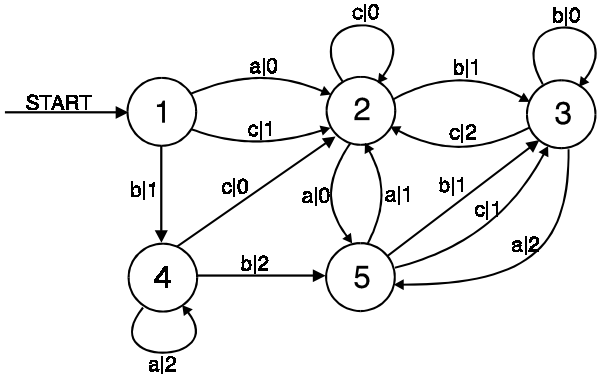
6.1.3 Príklad trénovania rekurentnej neurónovej siete

V tejto stati si krátko popíšeme proces trénovania rekurentnej siete na vzorkách generovaných Mealyho automatom. Pre ilustráciu procesu učenia rekurentnej siete uvažujme automat na obr. 6.5 [13].

Sieti budeme predkladať dvojice (vstupné slovo, odozva na vstupné slovo), ktoré reprezentujú náš automat. Začneme s kratšími slovami a postupne pokročíme k dlhším slovám, napríklad

acccb→00001, *caaab*→10101, *bbbbbb*→121000, atď.

Dôležité je reprezentovať v trénovacej množine všetky aspekty automatu, teda aj to, že spracovanie každého vstupného slova sa začína v iniciálnom stave 1. Jedna z možností je zavedenie zvláštneho vstupného aj výstupného symbolu, ktoré by signalizovali "reset". Teda z každého stavu automatu by príchod symbolu "!" znamenal prechod do stavu 1 s príslušným asociovaným výstupom "*x*". Trénovacia množina by potom vyzerala nasledovne



Obrázok 6.5. Mealyho automat použitý na generovanie trénovacej množiny pre rekurentnú sieť na obr. 6.3.

$$acccb!\rightarrow 00001x,\ caaab!\rightarrow 10101x,\ bbbbbb!\rightarrow 121000x,\ \text{atď.}$$

alebo

$$acccb!caaab!bbbbbb!... \rightarrow 00001x10101x121000x...$$

čo je v prepise do časovo usporiadanej trénovacej množiny (vstup, výstup)

$$a\rightarrow 0,\ c\rightarrow 0,\ c\rightarrow 0,\ c\rightarrow 0,\ b\rightarrow 1,\ !\rightarrow x,\ c\rightarrow 1,\ a\rightarrow 0,\ a\rightarrow 1,\ a\rightarrow 0,\ b\rightarrow 1,\ 1\rightarrow x,\ \text{atď.}$$

Máme teda štyri vstupné symboly ***a,b,c,"!"*** a štyri výstupné symboly **0,1,2,"*x*".** Reprezentujme ich binárne v tzv. "one-hot" kódovaní - 4-dimenzionálne kódy s práve jednou 1 a troma 0, pričom pozícia 1 kóduje príslušný symbol. Možné kódovanie je uvedené v nasledujúcej tabuľke 6.1:

Tabuľka 6.1

vstupný symbol	výstupný symbol	kód
a	0	1000
b	1	0100
c	2	0010
!	x	0001

Zrejme potrebujeme 4 vstupné (N=4) a 4 výstupné (M=4) neuróny. Pre náš experiment použijeme 4 rekurentné stavové neuróny (L=4) a 4 skryté nerekurentné neuróny (K=4). Trénovaciu množinu sme vygenerovali tak, že k 600 náhodne vybratým vstupným slovám nad abecedou {a,b,c} sme pomocou automatu na obr. 6.5 určili prislúchajúce výstupné slová nad abecedou {0,1,2}. Na koniec každého trénovacieho vstupného slova sme vložili "resetovací" symbol "!" a na koniec odpovedajúceho výstupného slova bol vložený symbol "x". Dĺžka slov sa pohybovala od 3 do 12 a rástla od najkratších k najdlhším. Na začiatku trénovania boli náhodne vygenerované váhy V,Q,W z intervalu [-0.5, 0.5] podľa rovnomerného rozdelenia pravdepodobnosti. Počas učenia sa z trénovacej množiny postupne berie vstup za vstupom a ich asociované výstupy, pričom po prezentácii každého vstupu sa príslušne upravia váhy. Trénovací proces sa ukončil po 18 epochách (jedna epocha spočíva v *postupnom* prejdení všetkých asociačných dvojíc v trénovacej množine) s trénovacou chybou 0,075. Naučená sieť bola testovaná na náhodne vygenerovaných vstupných slovách omnoho väčšej dĺžky ako 12 (čo bola maximálna dĺžka trénovacích slov). Odpovede na všetky testovacie (vstupné) slová, t.j. k nim prislúchajúce výstupné slová generované sieťou, zodpovedali automatu na obr. 6.5. Pred predložením každého testovacieho vstupného slova bola sieť "resetovaná" pomocou vstupu "!".

Ako zaujímavosť spomenieme, že v tomto prípade bolo možné "porozumieť" vnútornej reprezentácii problému (implicitne určeného trénovacou množinou) v naučenej rekurentnej sieti. Experimentálne sa totiž ukázalo, že stavy siete (4-rozmerné vektory aktivácií stavových neurónov) nepokrývajú stavový priestor $(0,1)^4$ rovnomerne, ale sú koncentrované v dobre detekovateľných zhlukoch. Navyše, tieto zhluky zodpovedajú stavom automatu, na základe ktorého bola vygenerovaná trénovacia množina.

Takýmto spôsobom možno z naučenej siete "vytiahnuť" automat, ktorý vyhovuje trénovacej množine a navyše ju aj zovšeobecňuje.

6.2 Rekurentné siete a ich trénovanie

6.2.1 Modely rekurentných sietí

V predošlej podkapitole sme si na príklade intuitívne vymedzili pojem rekurentnej siete a ukázali sme si prístup k jej učeniu. Vo všeobecnosti možno za rekurentnú sieť považovať akúkoľvek neurónovú sieť, v ktorej istá podmnožina neurónov (*rekurentné neuróny*) je schopná uchovať informáciu o svojich aktiváciách v predošlých časoch pre výpočet aktivácií neurónov v čase *t*+1. "Odpamätané" hodnoty sa objavia v čase *t*+1 ako aktivácie tzv. *kontextových neurónov*. Napríklad v architektúre rekurentnej siete z predošlej podkapitoly sú rekurentné neuróny vo výstupnej vrstve stavovej siete a kontextové neuróny tvoria časť vstupnej vrstvy asociačnej aj stavovej siete, kde sa v čase *t*+1 objavia aktivácie rekurentných neurónov z kroku *t*. Povedali sme si, že takýmto spôsobom rozširujeme neurónovú sieť o *vnútornú pamäť*.

Historicky vzniklo niekoľko modelov rekurentných sietí, ktoré možno považovať za viacvrstvové siete obohatené o rekurentné neuróny. Na obr. 6.6 a-c uvádzame tri modely tohoto typu, ktoré možno nájsť v literatúre. Vrstvy neurónov sú zobrazované obdĺžnikmi. Podobne ako v tradičných viacvrstvových sieťach, v rámci jednej vrstvy nie sú neuróny navzájom prepojené a prepojenia existujú len medzi neurónmi susedných vrstiev. Hrubé šípky reprezentujú prepojenia z každého neurónu spodnej vrstvy do každého neurónu hornej vrstvy. Tieto prepojenia majú váhy, ktoré sú modifikovateľné (počas trénovacieho procesu). Jednoduché šípky predstavujú *rekurentné prepojenia* medzi zodpovedajúcimi neurónmi východzej a cieľovej vrstvy. Prepojenia majú váhu 1, ktorá je nemenná a existujú len medzi *i*-tym neurónom východzej a *i*-tym neurónom cieľovej vrstvy. Na týchto prepojeniach sú oneskorovacie členy, ktorých doba oneskorenia zodpovedá jednotke diskrétneho času. Funkcia rekurentných prepojení spočíva v odpamätaní aktivácií rekurentných neurónov a ich zavedení do kontextových neurónov.

Architektúru na obr. 6.6a navrhol Elman [14]. Kontextová vrstva obsahuje kópie aktivácií skrytých neurónov z predošlého kroku. Autorom siete uvedenej na obr. 6.6b je Jordan [15]. Obr. 6.6c predstavuje kombináciu modelov na obr. 6.6a-b. Ako navrhuje Bengio [16], je možné mať zvláštnu kontextovú vrstvu pre rôzne vrstvy pôvodnej viacvrstvovej siete, ako je tomu pri architektúre na obr. 6.6c.

Ďalším variantom odpamätávania aktivácií rekurentných neurónov v kontextovej vrstve je postupné "nabaľovanie" hodnôt aktivácií v minulých krokoch diskrétneho času, s prvkom "zabúdania" dávnejších aktivácií. Nech $S_i^{(t)}$ je aktivácia *i*-teho rekurentného neurónu v čase *t* a $K_i^{(t)}$ aktivácia *i*-teho kontextového neurónu v čase t. Potom

$$K_i^{(t+1)} = \alpha K_i^{(t)} + S_i^{(t)} \tag{6.14}$$

$0 < \alpha < 1$ je konštanta reprezentujúca "*rýchlosť zabúdania*". Iterovaním (6.14) totiž dostávame

$$K_i^{(t+1)} = S_i^{(t)} + \alpha S_i^{(t-1)} + \alpha^2 S_i^{(t-2)} + \ldots = \sum_{\tau=0}^{t} \alpha^{t-\tau} S_i^{\tau} \qquad (6.15)$$

Pre koeficient α blízky 1 kódujú kontextové aktivácie časový kontext aktivácií odpovedajúcich rekurentných neurónov v širokom časovom rozpätí, avšak zároveň strácame detailnú informáciu o posledných aktiváciach rekurentných neurónov. Naproti tomu, pri hodnotách koeficienta α blízkych 0 kódujeme vývoj aktivácií rekurentných neurónov len v bezprostrednej minulosti, avšak s väčším dôrazom na detailnú informáciu o ich hodnotách.

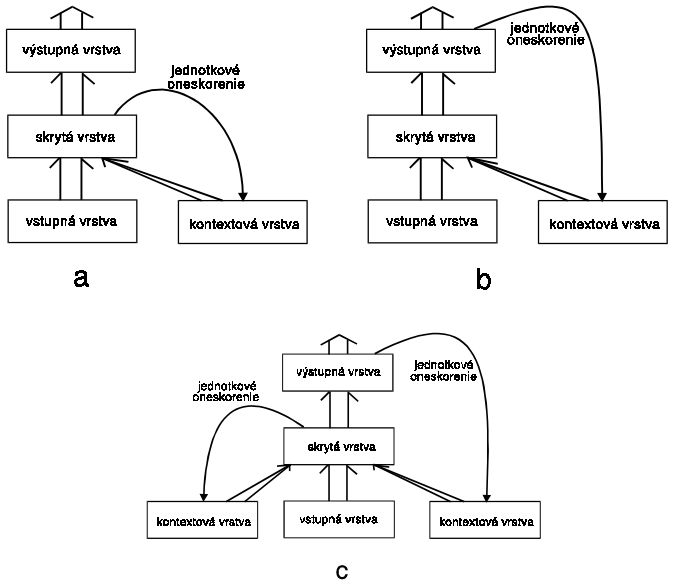
Takýto typ "odpamätávania" aktivácií rekurentných neurónov budeme označovať čiarkovaným prepojením medzi rekurentnou a kontextovou vrstvou. Architektúru na obr. 6.7a navrhol Jordan [15]. Ide o rozšírenie siete z obr. 6.6b. *i*-ty kontextový neurón uchováva informáciu o svojich aktiváciach v minulých krokoch a o aktivácii *i*-teho výstupného neurónu v predošlom kroku.

Model na obr. 6.7b pochádza od Stornetta a spol. [17]. Ide vlastne o klasickú viacvrstvovú sieť, ktorej vstupné neuróny kódujú históriu predložených vstupov v minulosti. Mozer [18] je autorom architektúry na obr. 6.7c. Sieť má k dispozícii informáciu o minulom vývoji aktivácií neurónov v skrytej vrstve.

Je len pochopiteľné, ak si čitateľ na tomto mieste položí dve otázky:

1. Prečo existuje toľko rôznych variantov rekurentných sietí?

2. Ako sformulovať pravidlá pre učenie takýchto sietí ?



Obrázok 6.6. Architektúry rekurentných sietí, ktoré možno považovať za klasické viacvrstvové siete obohatené o rekurentné neuróny. Aktivácie rekurentných neurónov v kroku t sa objavia ako aktivácie kontextových neurónov v kroku t+1.

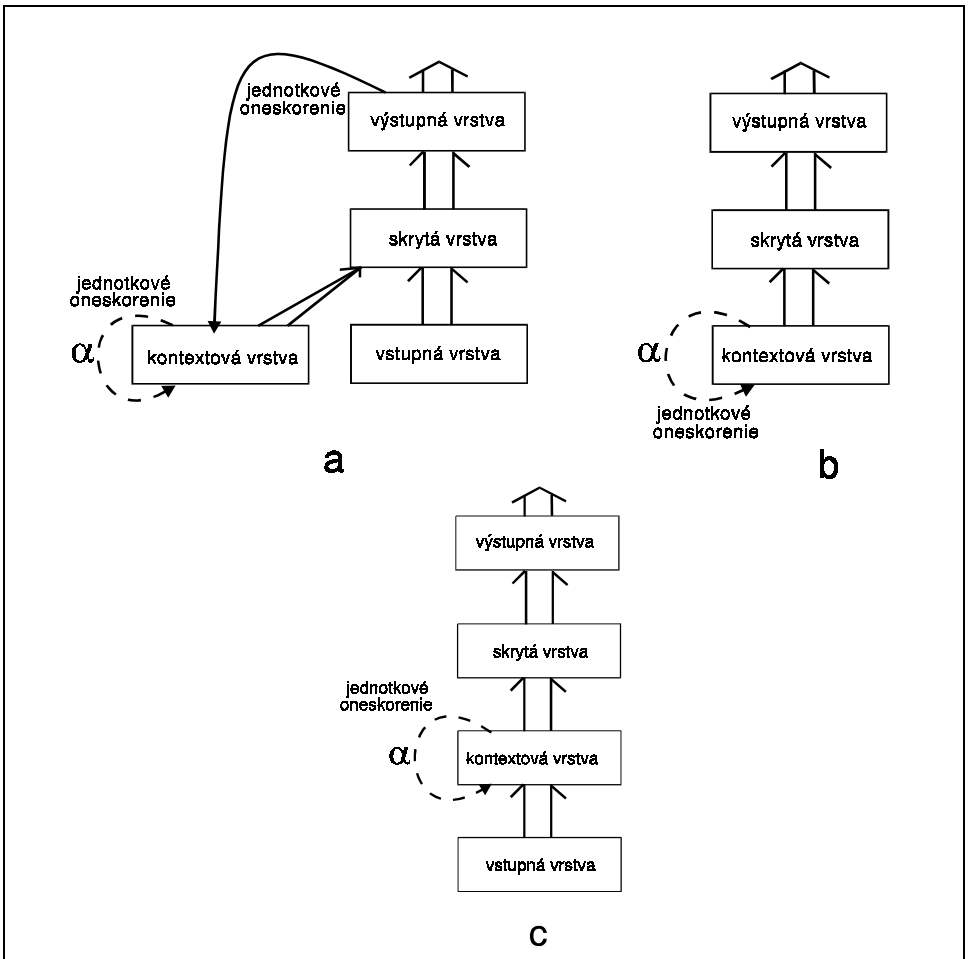
Odpoveď na prvú otázku je viac-menej priamočiara. Rôzne typy úloh sa vyznačujú rôznym typom časovo-priestorovej štruktúry a rôzne spôsoby kódovania časového kontextu vyhovujú rôznym časovým štruktúram v dátach. Nemalú úlohu hrá samozrejme aj otázka použiteľnosti trénovacieho procesu pri danom spôsobe kódovania časového kontextu. V zásade sa neurónové siete uplatňujú pri troch typoch úloh:

I. Klasifikačné alebo *asociačné úlohy*, aké poznáme už z kapitoly o viacvrstvových sieťach, avšak *s časovým kontextom.*

II. Predikčné úlohy.

III. Generatívne úlohy.

Pri prvom type úloh ide, v prípade klasifikácie, o rozhodnutie či práve ukončená postupnosť vstupov patrí, alebo nepatrí do nejakej triedy, prípadne do ktorej z možných tried ju možno zaradiť. Sem možno zaradiť napríklad klasifikáciu postupností symbolov z nejakej konečnej abecedy A (teda slov nad abecedou A) na základe príslušnosti k danému jazyku [19]. Príklad asociačnej úlohy s časovým kontextom bol uvedený v minulej podkapitole.



Obrázok 6.7. Architektúry rekurentných sietí s odpamätávaním aktivácií rekurentných neurónov v širšom časovom rozmedzí s prvkom zabúdania.

V druhom type úloh sa pokúšame nájsť časovú štruktúru v postupnosti dát, ktorá by umožnila na základe určitého úseku histórie dát v časoch menších ako *t* predpovedať dáta v čase väčšom ako *t*.

Tretím typom 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. V prípade zložitejších postupností je cieľom modelovania časových radov vystihnutie základných štatistických či dynamických charakteristík modelovanej časovej postupnosti, napríklad, invariantnej miery, metrickej entropie, fraktálnej dimenzie atraktora, atď. Samotné generovanie pokračovania úseku časovej rady sa môže realizovať napríklad nasledovným spôsobom: Po predložení dostupného úseku dát (do času t) sieť vygeneruje predikciu možnej hodnoty dát v nasledujúcom čase t+1. Táto predikcia sa priradí k pôvodnému úseku a na základe takto vytvoreného nového úseku dát vygenerujeme predikciu pre čas t+2, atď... Napríklad model siete zobrazený na obr. 6.6a bol úspešne použitý pre klasifikáciu kratších slov nad konečnou abecedou symbolov, ako aj na generovanie krátkodobých pokračovaní symbolických postupností [14].

Model z obr. 6.7a bol použitý Andersonom [20] na kategorizovanie hovorených slabík anglického jazyka. Sieť trénovaná na jednej skupine hlasov bola schopná správne fungovať pri nových, vopred nepočutých hlasoch. Architektúra na obr. 6.7c sa lepšie hodí na problém klasifikácie postupností ako pre generatívne úlohy [4].

6.2.2 Trénovanie rekurentných sietí

V predošlej podkapitole sme si na príklade ukázali ako zovšeobecniť trénovaciu procedúru BP, pôvodne navrhnutú pre viacvrstvové siete, aby bola použiteľná aj pre viacvrstvové siete obsahujúce rekurentné neuróny. V tejto podkapitole sa budeme systematickejšie zaoberať problémom učenia rekurentných sietí. Spomenieme si dva najčastejšie používané prístupy, ktoré je možné vystopovať v literatúre. Oba sú založené na myšlienke minimalizačnej metódy najprudšieho spádu (angl. *steepest descent*) do minima chybového funkcionálu E pohybom proti smeru gradientu ∇E.

Ako pri algoritme BP, aj tu bude základnou úlohou nájdenie analytických vzťahov vyjadrujúcich parciálne derivácie chybového funkcionálu E podľa jednotlivých modifikovateľných váh siete. Kvôli jednoduchosti prezentácie budeme uvažovať rekurentnú sieť zloženú z dvoch rekurentných neurónov prvého rádu. Pre ich aktivácie platí

$$S_i^{(t+1)} = g\left(\sum_{j=1}^{2} w_{ij} S_j^{(t)} + I_i^{(t)}\right) \qquad (i = 1,2) \tag{6.16}$$

kde $S_1^{(t)}$ (resp. $S_2^{(t)}$) je aktivácia prvého (resp. druhého) rekurentného neurónu v čase t a $I_1^{(t)}$ (resp. $I_2^{(t)}$) je vonkajší vstup (cez kanál váhy 1) do prvého (resp. druhého) rekurentného neurónu v čase t, g môže byť napríklad známa sigmoidálna funkcia (6.2). Prípad, že aj vonkajšie vstupy vchádzajú do rekurentných neurónov cez kanály s modifikovateľnými váhami by bol riešený analogicky, ale prezentácia by bola zbytočne zaťažená.

6.2.3 Spätné šírenie v čase

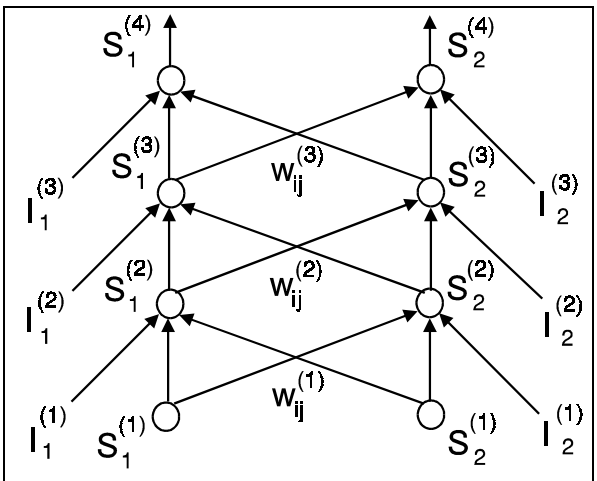
Predstavme si, že na vstup $(I_1^{(t)}, I_2^{(t)})$ privádzame konečné postupnosti (napríklad kódovaných symbolov) a až na konci postupností (teda napríklad na konci slov nad nejakou abecedou) máme k dispozícii učiaci signál, ktorý vypovedá o charaktere práve predvedenej postupnosti (napríklad, či patrí do nejakého jazyka, alebo nie). Treba si uvedomiť, že učiaci signál nie je vo všeobecnosti k dispozícii v každom kroku diskrétneho času (na rozdiel od príkladu uvedenom v predošlej podkapitole). Musíme teda počkať T krokov, zodpovedajúcich dĺžke vstupného reťazca, aby sme získali informáciu o smere v korekcii váh. Uvažujme, že prezentácia prvého vstupu zo vstupného reťazca sa udiala v čase t=1 a prezentácia posledného vstupu vstupného reťazca sa udeje v čase T.

Je možné si predstaviť rekurentnú sieť pracujúcu v T krokoch ako jednoduchú viacvrstvovú sieť, ktorá má 2(T+1) neurónov. V sieti sú kópie rekurentných neurónov s príslušnými prepojeniami pre každý krok diskrétneho času, pričom váhy prepojení sa v časoch $1 \le t \le T$ nemenia. Obr. 6.8 predstavuje rekurentnú sieť rozvinutú v čase $1 \le t \le 4$ pri prezentácii vstupnej postupnosti dĺžky T=3.

Idea rozvinutia rekurentnej siete v čase $1 \le t \le T$ bola pôvodne navrhnutá už Minskym a Papertom [21] a kombinovaná s procedúrou BP je uvedená v [22]. Opäť zdôrazňujeme, že váhy w_{ij} sú nezávislé od času $1 \le t \le T$ a v tomto intervale sa ich hodnoty nemenia. Chybový signál, ktorý sa objaví v čase t=4 (po skončení vstupnej postupnosti dĺžky T=3), necháme spätne sa šíriť cez časovo rozvinutú sieť (obr. 6.8) metódou BP. Algoritmus učenia môžeme vyjadriť pomocou týchto dvoch krokov:

1. Pri počítaní parciálnych derivácií $\partial E / \partial w_{ij}$ považujeme váhy $w_{ij}^{(t)}$ v rôznych časoch t za nezávislé a štandardným postupom spätného chodu získame parciálne derivácie $\partial E / \partial w_{ij}^{(t)}$, pre i,j=1,2 a t=1,2,3.
2. Modifikácia váhy w_{ij} bude priamo úmerná súčtu navrhnutých modifikácií v rôznych krokoch diskrétneho času $\Delta w_{ij} = -\varepsilon \sum_{t=1}^{T} \partial E / \partial w_{ij}^{(t)}$, kde $\varepsilon > 0$ je konštanta nazývaná rýchlosť učenia (angl. *learning rate*), podobne ako v klasickej procedúre BP.

Takáto procedúra trénovania rekurentných sietí založená na modifikácii tradičného prístupu BP v časovo rozvinutej sieti sa nazýva spätné šírenie v čase (angl. *Back Propagation Through Time*, BPTT) [25]. Nevýhodou procedúry BPTT sú veľké pamäťové nároky v prípade rozsiahlejších sietí a dlhších trénovacích postupností (veľké T). Aj keď BPTT sa neujala ako široko používaná trénovacia metóda, Rumelhart, Hinton a Williams [22] ju úspešne použili pre učenie rekurentných sietí trénovaných imitovať správanie sa posuvného registra.



Obrázok 6.8. Časovo rozvinutá sieť s dvoma rekurentnými neurónmi v krokoch t=1,2,3 diskrétneho času.

6.2.4 Rekurentné učenie v reálnom čase

Hlavná myšlienka tohoto prístupu k trénovaniu rekurentných sietí spočíva v úprave váh v *každom* kroku diskrétneho času bez potreby čakania na ukončenie vstupnej trénovacej postupnosti. Týmto sa redukuje problém premenlivej dĺžky postupnosti. Nie je totiž potrebné dopredu určiť maximálne prípustnú dĺžku trénovacej postupnosti a mizne aj potreba alokovania pamäti proporcionálne k dĺžke postupnosti. Autormi tejto metódy známej ako *rekurentné učenie v reálnom čase* (angl. *Real Time Recurrent Learning*, RTRL) sú Williams a Zipser [23].

Uvažujme opäť jednoduchú neurónovú sieť s dvoma neurónmi s dynamikou určenou vzťahom (6.16). Na vstupe siete prezentujeme postupnosť $\left\{\left(I_1^{(t)}, I_2^{(t)}\right)\right\}_{t=1}^{T}$, a nech očakávaný výstup siete po skončení postupnosti (v čase T+1) je (O_1, O_2). Definujme chybové funkcionály

$$E_k^{(t)} = \begin{cases} O_k - S_k^{(T+1)} & \left(\text{pre } t = T+1\right) \\ 0 & \left(\text{pre } \quad 1 \le t \le T\right) \end{cases} \tag{6.17}$$

pre k=1,2. Potom celkový chybový funkcionál je

$$E^{(t)} = \frac{1}{2}\sum_{k=1}^{2}\left(E_k^{(t)}\right)^2 \tag{6.18}$$

Zmena $\Delta w_{ij}(t)$ váhy w_{ij} v čase t bude

$$\Delta w_{ij}(t) = -\varepsilon \frac{\partial E(t)}{\partial w_{ij}} \tag{6.19}$$

kde $\varepsilon > 0$ je rýchlosť učenia. Z (6.19) máme

$$\Delta w_{ij}(t) = \varepsilon \sum_{k=1}^{2} E_k^{(t)} \frac{\partial S_k^{(t)}}{\partial w_{ij}}$$

a podobnou úvahou ako pri zavedení vzťahu (6.13) (tentoraz máme neuróny prvého rádu) dostávame

$$\frac{\partial S_r^{(t)}}{\partial w_{ij}} = g'\left(\phi\left(S_r^{(t)}\right)\right)\left[\delta_{ri} S_j^{(t-1)} + \sum_{a=1}^{2} w_{ra} \frac{\partial S_a^{(t-1)}}{\partial w_{ij}}\right] \tag{6.20}$$

Pripomíname, že podobne ako vo vzťahu (6.13), ϕ je inverzná funkcia k funkcii g a δ_{ri} je Kroneckerovo delta. Celý postup je vhodné inicializovať postulovaním

$$\frac{\partial S_r^{(0)}}{\partial W_{i\,ln}} = 0.$$

Analogicky príkladu z úvodnej podkapitoly aj tu využívame vzťah (6.20) pre postupné prepočítavanie parciálnych derivácií pre budúce kroky. Williams a Zipser [23,24] odporúčajú menšie hodnoty rýchlosti učenia ε. Pochopiteľne, triky známe zo štandardnej procedúry BP pre urýchlenie učenia, či zabránenie uviaznutiu vo veľmi nežiadúcom lokálnom minime možno použiť aj v procedúre RTRL. Napríklad v [13] bol použitý momentový člen pre zvýšenie robustnosti antigradientového poklesu na chybovom povrchu voči slabším lokálnym minimám.

Metóda RTRL našla živnú pôdu medzi užívateľmi rekurentných neurónových sietí. Úspešne bola použitá pri problémoch inferencie konečného akceptora regulárneho jazyka na základe pozitívnych príkladov slov patriacich do jazyka a negatívnych príkladov slov nepatriacich do daného regulárneho jazyka [19]. Použili sme ju aj pre inferenciu Mealyho automatu na základe príkladov jeho činnosti [13,26] a bola použitá aj v prípade inferencie iných automatov rekurentnými sieťami [28,29].

6.3 Na záver

Po úvodnej podkapitole, intuitívne navodivšej potrebu neurónových sietí s vnútornou pamäťou a naznačujúcej spôsob učenia takýchto sietí, sme si v podkapitole 6.2 metodickejším spôsobom predstavili niektoré základné modely rekurentných neurónových sietí a dva najbežnejšie prístupy k ich učeniu.

V poslednom čase je záujem o rekurentné neurónové siete obrovský a s tým súvisí aj explózia literatúry venovanej takýmto sieťam [30]. Určitý podiel na záujme o rekurentné siete má aj existencia výsostne praktických problémov reálneho sveta vykazujúcich časovo-

priestorové štruktúry. Či už je to v oblasti riadenia technologických procesov, robotiky, predikcie odberu elektrickej energie v rozvode generátora, predikcie vývoja na aukčnej burze, atď.

Iste, existujú mnohé iné (napríklad štatistické) metódy pre hľadanie štruktúry v časovej postupnosti a následnom využití vystopovanej štruktúry, napríklad pre predikciu možného budúceho vývoja postupnosti. V mnohých praktických aplikáciách je úspešnosť rekurentných neurónových sietí porovnateľná s úspešnosťou tradične používaných metód. Ako vo všetkých oblastiach modelovania dát, aj tu treba zvoliť rozumný kompromis. Neurónové siete, napríklad, pracujú v testovacom (t.j. pracovnom) móde pomerne rýchlo, pretože väčšina "modelovacej práce" bola presunutá do trénovacej fázy. Na druhej strane, rigorozita dosiahnutých výsledkov je pomerne malá. Pre dosiaľ nevidenú vzorku sieť síce ponúkne odpoveď, avšak bez informácie o miere dôveryhodnosti v ponúknutú odpoveď. V tomto ohľade sú výstupy tradičných štatistických metód rigoróznejšie. Čas potrebný pre získanie odpovede pre každú vzorku však môže byť podstatne väčší ako pri neurónových sieťach. V literatúre sa dajú nájsť pokusy spojiť výhody neurónového a tradične štatistického prístupu k modelovaniu dát [3,31], avšak spravidla musia byť zaplatené väčšou pamäťovou a časovou náročnosťou.

Značné úsilie je venované aj skúmaniu rekurentných neurónových sietí pomocou aparátu teórie dynamických systémov [32]. Rekurentné siete totiž možno považovať za dynamické systémy. Parametrami zobrazenia stavového priestoru sú váhy synaptických prepojení. Keďže premenlivé vonkajšie vstupy sa v tej, či onej miere podieľajú na determinovaní stavového zobrazenia, rekurentná sieť predstavuje neautonómny dynamický systém, vyšetrovanie ktorého je veľmi obtiažne. Väčšina prác je preto venovaná rekurentným sieťam v autonómnom režime (vonkajšie vstupy považujeme za konštantné).

Dvoma najhlavnejšími prúdmi v skúmaní rekurentných sietí ako dynamických systémov sú tieto oblasti:

1) *Popis invariantných atraktívnych množín v stavovom priestore siete.* Invariantné atraktívne množiny sú dôležitým faktorom pri determinovaní asymptotického správania sa rekurentnej siete [33-36]. Zaujímavým výsledkom bol aj experimentálny a analytický dôkaz existencie chaotického režimu v rekurentných sieťach [37].

2) *Interpretácia trénovacieho procesu ako postupnosti bifurkácií vedúcej k indukcii želanej dynamiky siete.* V priebehu trénovania meníme váhy synaptických prepojení (parametre dynamického systému), až pokiaľ dynamické správanie sa siete nezodpovedá želanému stavu. Objasnenie bifurkačného mechanizmu vzniku napríklad nových atraktívnych pevných bodov [26,36], či atraktívnych periodických orbít [39] napomáha pochopeniu trénovacieho procesu.

Viac-menej úspešné modelovanie chaotických časových postupností rekurentnými neurónovými sieťami [41,42] je obmedzené na krátkodobé predpovede budúceho možného vývoja pokračovania danej postupnosti. Hlavným problémom je obrovská citlivosť chaotických trajektórií na malé zmeny v počiatočných podmienkach. Okrem toho, aj keď postupnosť nie je chaotická, ale len dostatočne zložitá, t.j. zahŕňa korelácie medzi časovo značne vzdialenými prvkami postupnosti, učenie rekurentných sietí gradientovými metódami zlyháva. Časovo vzdialenejšie prvky postupnosti totiž majú omnoho menší vplyv

na výsledný gradient ako súčasné prvky postupnosti (ak je stav siete "blízko" nejakej atraktívnej množiny v stavovom priestore siete) [40]. Sieť nie je schopná premietnuť potenciálne dôležitú informáciu o vzťahu minulých vstupov k súčasnému vstupu do primeranej úpravy váh prepojení, umožňujúcej modelovanie takýchto vzťahov.

Isté východisko ponúka napríklad Schmidhuber [44]. Trénuje rekurentnú sieť na pôvodnej postupnosti vstupov. Po dosiahnutí lokálneho minima chybového funkcionálu testuje sieť na pôvodnej postupnosti vstupov. Zaznamenáva vstupy, pri ktorých sa sieť v odpovedi pomýlila a v ďalšom kroku trénuje novú rekurentnú sieť už len na vstupoch, na ktorých predošlá sieť zlyhala (spolu s týmito vstupmi predkladá aj kódovanú informáciu o čase a kontexte v ktorom sa objavili). Tento postup možno rekurentne opakovať a vybudovať hierarchickú štruktúru rekurentných sietí modelujúcu danú "zložitú" vstupnú postupnosť.

Na záver už len spomenieme, že analogicky k teorémam o ľubovoľne dobrej aproximácii spojitých funkcií nad kompaktnou oblasťou (napríklad v L_2 norme) viacvrstvovými neurónovými sieťami [38] možno vysloviť tvrdenia o ľubovoľne presnej aproximácii diskrétneho dynamického systému daného spojitým zobrazením kompaktného stavového priestoru rekurentnými neurónovými sieťami. Ako sme už videli aj v kapitole o viacvrstvových sieťach, teoretická schopnosť systému modelovať určité dáta nemusí mať veľa spoločného s našou schopnosťou nastaviť parametre systému tak, aby dané dáta skutočne dobre modeloval. Navyše, pokrytie stavového priestoru prvkami postupnosti pri veľmi zložitých, "chaotických" postupnostiach nemusí byť "rovnomerné" ako by sa žiadalo pre dobrú aproximáciu stavového zobrazenia, ale je dané invariantnou mierou príslušného generujúceho zobrazenia. Dôsledkom môže byť, že aproximácia stavového zobrazenia nad oblasťami malej miery rekurentnou sieťou bude veľmi nepresná.

Literatúra

[1] B.K. Natarajan. *Machine Learning - A Theoretical Approach*. Morgan Kaufmann Publishers, Inc., San Mateo, California, 1991.

[2] D.J.C. MacKay. Bayesian interpolation. *Neural Computation*, 4: 415-447, 1992.

[3] D.J.C. MacKay. The evidence framework applied to classification networks. *Neural Computation*, 4: 720-736, 1992.

[4] J. Hertz, A. Krogh, and R.G. Palmer. *Introduction To The Theory Of Neural Computation*. Addison-Wesley Publishing Company, Redwood City, California, 1991.

[5] D-T. Lin, J.E. Dayhoff, and P.A. Ligomenides. Trajectory production with the adaptive time-delay neural network. *Neural Networks*, 3: 447-461, 1995.

[6] U. Bodenhausen and A. Waibel. The Tempo2 algorithm: Adjusting time-delays by supervised learning. In: J.E. Moody, R.P. Lippmann, and D.S. Touretzky, editors, *Advances in Neural Information Processing Systems*, pp. 155-161, Vol. 3, Morgan Kaufmann Publishers, Inc., San Mateo, California, 1991.

[7] J.L. McClelland and J.L.Elman. Interactive processes in speech perception: The TRACE Model. In: J.L. McClelland, D.E. Rumelhart, and PDP Research Group,

Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 2, chapter 15, MIT Press, Cambridge, 1986.

[8] J.L.Elman and D. Zipser. Learning the hidden structure of speech. *Journal of the Acoustical Society of America*, 83: 1615-1626, 1988.

[9] A. Weibel. Modular construction of time-delay neural networks for speech recognition. *Neural Computation*, 1: 39-46, 1989.

[10] R.P. Lippmann. Review of neural networks for speech recognition. *Neural Computation*, 1: 1-38, 1989.

[11] J.E. Hopfcroft and J.D. Ullman. *Introduction To Automata Theory, Languages and Computation*. Addison-Wesley Publishing Company, Redwood City, California, 1991.

[12] M.W. Shields. *An Introduction To Automata Theory*. Blackwell Scientific Publications, London, UK, 1987.

[13] P. Tiňo and J. Šajda. Learning and extracting initial mealy machines with a modular neural network model. *Neural Computation*, 4: 822-844, 1995.

[14] J.L.Elman. Finding structure in time. *Cognitive Science*, 14: 179-211, 1990.

[15] M.I. Jordan. Serial Order: A parallel distributed processing approach. In: J.L. Elman and D.E. Rumelhart, editors, *Advances in Connectionist Theory: Speech*, Erlbaum, Hillsdale, 1989.

[16] Y. Bengio, R. Cardin and R. De Mori. Speaker independent speech recognition with neural networks and speech knowledge. In: D.S. Touretzky, editor, *Advances in Neural Information Processing Systems* II, pp. 218-225, Vol. 3, Morgan Kaufmann Publishers, Inc., San Mateo, California, 1990.

[17] W.S. Stornetta, T. Hogg, and B.A. Huberman. A dynamical approach to temporal pattern processing. In: D.Z. Anderson, editor, *Neural Information Processing Systems*, pp. 750-759, Vol. 3, American Institute of Physics, New York, 1988.

[18] M.C. Mozer. A focused back-propagation algorithm for temporal pattern recognition. *Complex Systems*, 3: 349-381, 1989.

[19] C.L. Giles, C.B. Miller, D. Chen, G.Z. Sun, H.H. Chen, and Y.C. Lee. Learning and extracting finite state automata with second order recurrent neural networks. *Neural Computation*, 4: 393-405, 1992.

[20] S.J. Anderson, J.W.L Merrill, and R. Port. Dynamic speech categorization with recurrent networks. In: D.S. Touretzky, G. Hinton, and T. Sejnowski, editors, *Proceedings of the 1988 Connectionist Models Summer School*, Pittsburgh, pp. 398-406, Morgan Kaufmann Publishers, Inc., San Mateo, California, 1989.

[21] M.L. Minsky and S.A. Papert. *Perceptrons*. MIT Press, Cambridge, 1969.

[22] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning Internal Representations by Error Propagation. In: J.L. McClelland, D.E. Rumelhart, and PDP Research Group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Volume 1: Foundations, chapter 8, MIT Press, Cambridge, 1986.

[23] R.J. Williams and D. Zipser. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. *Neural Computation*, 1: 270-280, 1989.

[24] R.J. Williams and D. Zipser. Experimental Analysis of the Real-Time Recurrent Learning Algorithm. *Connection Science*, 1: 87-111, 1989.

[25] P.J. Werbos. Backpropagation Through Time: What It Is and How to Do It. *Proceedings of the IEEE*, 10: 1550-1560, 1990.

[26] P. Tiňo, B.G. Horne, C.L. Giles, and P.C. Collingwood. Finite State Machines and Recurrent Neural Networks - Automata and Dynamical Systems Approaches. In: J.E. Dayhoff and O. Omnivar, editors, *Progress in Neural Networks*, special volume on "*Temporal Dynamics and Time-Varying Pattern Recognition*", Albex, 1996.

[27] M.P. Casey. *Computation in Discrete-Time Dynamical Systems*. Ph.D. Thesis, University of California, San Diego, Department of Mathematics, 1995.

[28] A. Cleeremans, D. Servan-Schreiber, and J.L. McClelland. Finite State Automata and Simple Recurrent Neural Networks. *Neural Computation*, 3: 372-381, 1989.

[29] Z. Zeng, R.M. Goodman, and P. Smyth. Learning Finite State Machines With Self-Clustering Recurrent Networks. *Neural Computation*, 6: 976-990, 1993.

[30] M.C. Mozer. Neural Net Architectures For Temporal Sequence Processing. In: A. Weigend and N. Gershenfeld, editors, *Predicting the Future and Understanding the Past*, Addison-Wesley Publishing Company, Redwood City, California, 1993.

[31] D.J.C. MacKay. A Practical Bayesian Framework for BackProp Networks. *Neural Computation*, 3: 448-472, 1992.

[32] J. Guckenheimer and P. Holmes. *Nonlinear Oscillations, Dynamical Systems, and Bifurcation of Vector Fields*. Springer-Verlag, 1982.

[33] M. Vidyasagar. Location and Stability of the High-Gain Equilibria of Nonlinear Neural Networks. *IEEE Transactions on Neural Networks*, 4: 660-672, 1993.

[34] L. Jin, P.N. Nikiforuk, and M.M. Gupta. Absolute Stability Conditions for Discrete-Time Recurrent Neural Networks. *IEEE Transactions on Neural Networks*, 6: 954-963, 1994.

[35] M.W. Hirsh. Saturation at High Gain in Discrete Time Recurrent Networks. *Neural Networks*, 3: 449-453, 1994.

[36] E.K. Blum and X. Wang. Stability of Fixed Points and Periodic Orbits and Bifurcation in Analog Neural Networks. *Neural Networks*, 5: 577-587, 1992.

[37] X. Wang. Period-Doubling to Chaos in a Simple Neural Network: An Analytical Proof. *Complex Systems*, 5: 425-441, 1991.

[38] K. Hornik, M. Stinchcombe and H. White. Multilayer Feedforward Networks Are Universal Approximators. *Neural Networks*, 2: 359-366, 1989.

[39] K. Doya. Bifurcation in the Learning of Recurrent Neural Networks. In: *Proceedings of 1992 IEEE International Symposium on Circuits and Systems*, pp. 2777-2780, 1992.

[40] Y. Bengio, P. Simard, and P. Frasconi. Learning Long-Term Dependencies with Gradient Is Difficult. *IEEE Transactions on Neural Networks*, 2: 157-166, 1994.

[41] J.M. Kuo and J.C. Principe. A Systematic Approach to Chaotic Time Series Modeling With Neural Networks. In: *IEEE Workshop on Neural Nets for Signal Processing*, Ermioni, Greece, 1994.

[42] J.C. Principe, A. Rathie, and J.M. Kuo. Prediction of Chaotic Time Series with Neural Networks and the Issue of Dynamic Modeling. *International Journal of Bifurcation and Chaos*, 4: 989-996, 1992.

[43] M. Casdagli. Nonlinear Prediction of Chaotic Time Series. *Physica D*, 35: 335-356, 1989.

[44] P. Schmidhuber. Learning Complex, Extended Sequences Using the Principle of History Compression. Neural *Computation*, 4: 234-242, 1992.