Rekurentné neurónové siete

Peter Tiňo[1]

1. Prečo rekurentné siete?

V kapitole venovanej vrstvovým neurónových sieťach a metódam ich trénovania sme videli, že vrstvové siete sú za určitých okolností schopné naučiť sa asociovať k vstupným vektorom požadované výstupné vektory. 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* (*Learning Theory* [1])). Rozsah tejto práce neumožňuje podrobnejšie poznámky o tejto disciplíne. Obmedzíme sa len na konštatovanie, že vrstvová neurónová sieť zovšeobecní trénovacie vzorky tak, že nimi preloží hyperplochu (pri lineárnych sieťach hyperrovinu), ktorá je čo najhladšia.

Požiadavka hladkosti intuitívne reprezentuje staré pravidlo modelovania dát (tzv. Occam's Razor [2] [3]), podľa ktorého by model nemal demonštrovať viac "štruktúry", ako je reprezentované v dátach. Ak by napríklad trénovacia množina pozostávala z dvojíc (vstup, požadovaný výstup) ležiacich na nejakej hyperrovine 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 viac štruktúry než možno vystopovať v trénovacích dátach. Aj nelineárna vrstvová sieť zobrazujúca vstupy na výstupy, pričom dvojice (vstup, výstup) ležia na hyperploche Ψ, len "mierne zvlnenej" verzii hyperroviny Π, bude zrejme vyhovujúca. Naproti tomu, aj keby nelineárna vrstvová sieť *presne* preložila trénovacími vzormi hyperplochu Φ, ktorá by však bola vysoko nelineárna, ťažko by sme mohli uveriť, že odpovede siete na vstupy neobsiahnuté v trénovacej množine budú mať niečo spoločné s tendenciou dát ležať na hyperrovine Π. Voľne možno povedať, že takýto model viví v dátach viac štruktúry, než v nich v skutočnosti je – fenomén známy pod menom *overfitting* (premodelovanie dát). Existujú metódy umožňujúce, aspoň do určitej miery, vyhnúť sa nástrahám premodelovania dát. Prípadných záujemcov odkazujeme na knihu [4].

Existujú však typy úloh, kde nelineárne vrstvové 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 vrstvové 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.

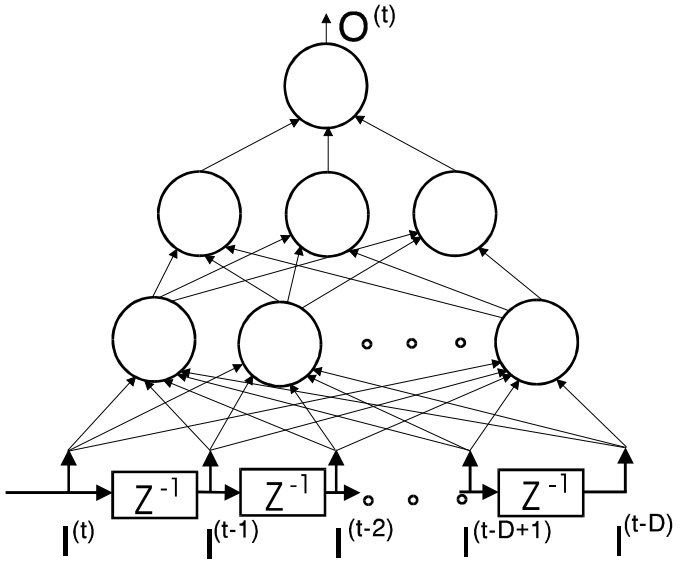
1.1 Príklad časovej štruktúry v dátach

Vrstvová sieť je schopná "naučiť sa" trénovaciu množinu pozostávajúcu napríklad z párov (vstup, požadovaný výstup): $A\to\alpha$, $B\to\beta$, $C\to\delta$, $D\to\alpha$, 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 $F: R^N \to R^M$ tak, že $F(A) \cong \alpha$, $F(B) \cong \beta$, $F(C) \cong \gamma$ a $F(D) \cong \alpha$. 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: $A\to\alpha$, $B\to\beta$, $B\to\alpha$, $B\to\gamma$, $C\to\alpha$, $C\to\gamma$, $D\to\alpha$…Vidíme, že k jednému vstupu môžeme mať viacero výstupov, v závislosti na **časovom kontexte** tej-ktorej asociácie. Inými slovami, o výstupe siete by nemal rozhodovať len vstup siete, ale aj informácia o doterajšej histórii predkladaných vzoriek. Vrstvová sieť by mala byť rozšírená o možnosť reprezentovať časový kontext, aby tak mohla

[1] Kapitola z knihy: Kvasnička, V., Beňušková Ľ., Pospíchal J., Farkaš I., Tiňo P., Kráľ A. Úvod do teórie neurónových sietí. Iris, Bratislava, 1997, str. 118-141.

na základe predloženého vstupu lepšie rozhodnúť o výstupe. Architektonicky najjednoduchšie riešenie ponúka tzv. **Time Delay Neural Network** (**TDNN**) (obrázok 1).



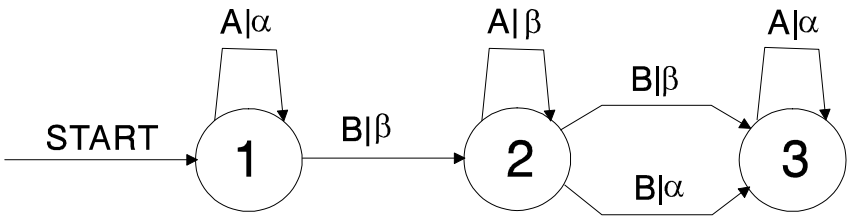
Obrázok 1: Architektúra TDNN. Okrem aktuálneho prvku časovej postupnosti vstupov (v čase t), sú vrstvovej sieti prezentované aj vstupy z predošlých D krokov diskrétneho času.

V podstate umožňuje vrstvovej 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 chodu (Back Propagation – **BP**), pričom je dôležité zachovať poradie trénovacích vzoriek v trénovacej množine. Ak máme šťastie, aj takéto jednoduché rozšírenie vrstvovej 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 vrstvové 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][8][9][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 považovať za symboly. 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→α, A→α, B→β, B→β, B→α, A→β, A→β, A→β, je popísaný Mealyho automatom [11] [12] na obrázku 2. Práca automatu začína v stave 1 označenom šipkou START. Po príchode vstupného symbolu $V \in \{A, B\}$ sa presunieme do nového stavu z množiny stavov $\{1,2,3\}$ pozdĺž šipky prislúchajúcej vstupnému symbolu V, pričom so symbolom V asociujeme výstup $W \in \{\alpha, \beta\}$ podľa pravidla V|W.



Obrázok 2: Mealyho automat popisujúci asociačnú úlohu s časovým kontextom. V závislosti od momentálneho stavu sú k vstupom A,B jednoznačne určené ich asociačné výstupy.

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 prenesiedo 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 obrázku 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 obrázku 2 bude architektúra TDNN nevyhovujúca.

1.2 Predbežný príklad rekurentej neurónovej siete

Inšpirovaní predchádzajúcimi úvahami uveďme architektúru neurónovej siete (obrázok 3) zloženej z dvoch vrstvových sietí

- **asociačnej siete** – realizujúcej asociáciu výstupu k danému vstupu na základe "vnútornej pamäte" siete
- **stavovej siete** – realizujúcej kódovanie doterajšej histórie vstupov predložených sieti

Architektúra bola predstavená v [13]. Obe vrstvové siete zdielajú spoločnú vstupnú vrstvu, ktorá sa skladá zo vstupných neurónov zabezpečujúcich prekopírovanie vstupného vektora $I^{(t)}$ v čase t do siete a zo "stavových" neurónov, ktorých aktivácie v čase t tvoria **stav siete** $S^{(t)}$ kódujúci históriu predkladaných vstupov $I^{(\tau)}, \tau < t$. Asociačná sieť má tri vrstvy, okrem vstupnej, ešte skrytú vrstvu neurónov druhého rádu, ktorých aktivácie v čase t sú rátané takto

$$H_j^{(t)} = g\left(\sum_{l,n} Q_{j\ln} S_l^{(t)} I_n^{(t)}\right), \tag{1}$$

kde g je obvyklá sigmoidálna aktivačná funkcia

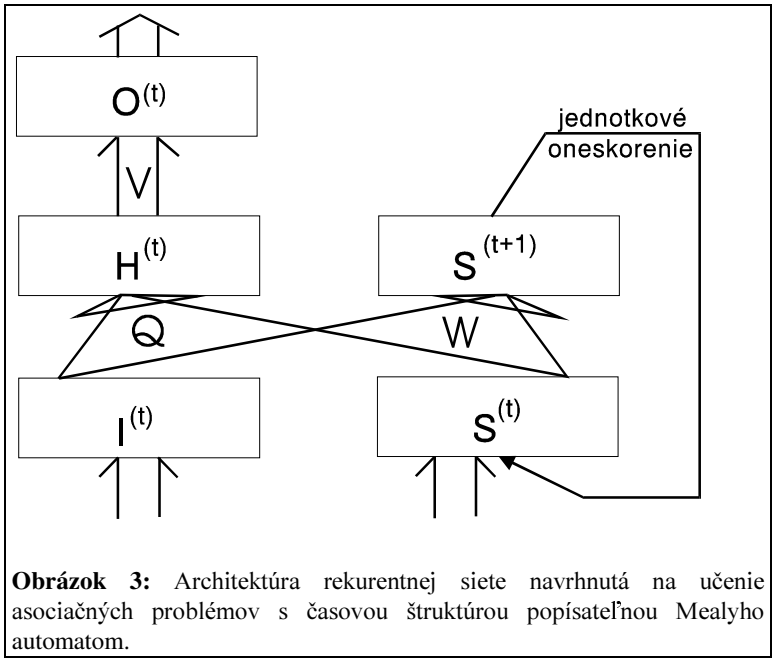
$$g(u) = \frac{1}{1+e^{-u}} \tag{2}$$

Výstupná vrstva neurónov prvého rádu reprezentuje výstup siete, ktorým sú v čase t aktivácie

$$O_m^{(t)} = g\left(\sum_k V_{mk} H_k^{(t)}\right) \tag{3}$$

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 $I^{(t)}$

$$S_i^{(t+1)} = g\left(\sum_{l,n} W_{i\ln} S_l^{(t)} I_n^{(t)}\right) \tag{4}$$



Obrázok 3: Architektúra rekurentnej siete navrhnutá na učenie asociačných problémov s časovou štruktúrou popísateľnou Mealyho automatom.

Ak sú vstupné vektory $I^{(t)}$ z R^N, výstupné vektory 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 $I^{(t)}$ a stavu $S^{(t)}$ 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 obrázku 2. Asociačnú *vrstvovú 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 $S^{(t)}, S^{(t+1)}$
- **ne–rekurentné** – vo vrstve $H^{(t)}$

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

vrstvových sietí definujeme chybový funkcionál ako $E = \frac{1}{2}\sum_m \left(D_m^{(t)} - O_m^{(t)}\right)^2$, a upravme váhy V, Q, W proporcionálne k inverzným gradientom

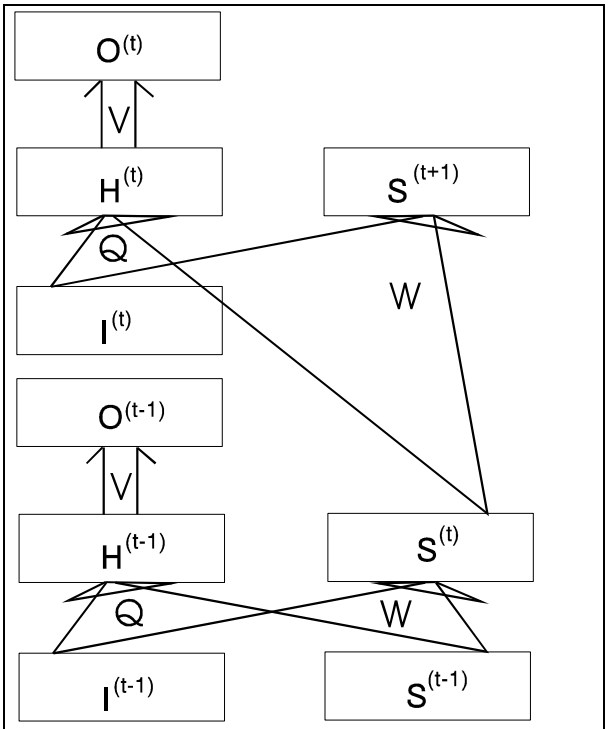
$$\Delta V_{mk} = -\alpha \frac{\partial E}{\partial V_{mk}} \tag{5}$$

$$\Delta Q_{j\ln} = -\alpha \frac{\partial E}{\partial Q_{j\ln}} \tag{6}$$

$$\Delta W_{i\ln} = -\alpha \frac{\partial E}{\partial W_{i\ln}} \tag{7}$$

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

Pre podrobnejší výpočet parciálnych derivácií uvedených v (5), (6) a (7) je vhodné prekresliť obrázok 3 na sieť rozvinutú v čase (konkrétne v časoch t, t+1) (obrázok 4).



Obrázok 4: Rekurentná sieť z obrázku 3 časovo rozvinutá v dvoch okamihoch t a t+1 diskrétneho času.

Postupom analogickým postupu pri odvodení klasického algoritmu BP (použitím tzv. chain rule pravidla pre deriváciu zloženej funkcie) dostávame parciálne derivácie chybového funkcionálu E podľa váh V a Q

$$\frac{\partial E}{\partial V_{mk}} = \left(O_m^{(t)} - D_m^{(t)}\right) g'\left(\phi\left(O_m^{(t)}\right)\right) H_k^{(t)} \tag{8}$$

$$\frac{\partial E}{\partial \phi\left(O_m^{(t)}\right)} = \left(O_m^{(t)} - D_m^{(t)}\right) g'\left(\phi\left(O_m^{(t)}\right)\right) \tag{9}$$

$$\frac{\partial E}{\partial Q_{j\ln}} = S_l^{(t)} I_n^{(t)} g'\left(\phi\left(H_j^{(t)}\right)\right) \sum_m V_{mj} \frac{\partial E}{\partial \phi\left(O_m^{(t)}\right)} \tag{10}$$

kde g' je derivácia funkcie g, $g'(u) = g(u)(1 - g(u))$ a ϕ je inverzná funkcia k funkcii g.

Výpočet parciálnych derivácií $\frac{\partial E}{\partial W_{i\ln}}$ 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

$$\frac{\partial E}{\partial W_{i\ln}} = \frac{\partial E}{\partial S_i^{(t)}} \frac{\partial S_i^{(t)}}{\partial W_{i\ln}} \tag{11}$$

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

$$\frac{\partial E}{\partial S_i^{(t)}} = \sum_m \frac{\partial E}{\partial \phi\left(O_m^{(t)}\right)} \sum_k V_{mk}\, g'\left(\phi\left(H_k^{(t)}\right)\right) \sum_n Q_{kin} I_n^{(t)} \tag{12}$$

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 $W_{i\ln}$, kedže

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

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

$$\frac{\partial S_r^{(t)}}{\partial W_{i\ln}} = g'\left(\phi\left(S_r^{(t)}\right)\right)\left[\delta_{ri} S_l^{(t-1)} I_n^{(t-1)} + \sum_{a,b} W_{rab} I_b^{(t-1)} \frac{\partial S_a^{(t-1)}}{\partial W_{i\ln}}\right] \tag{13}$$

kde δ_{ri} je kroneckerovo delta $\delta_{ri} = \begin{matrix} 1, r = i \\ 0, r \neq i \end{matrix}$.

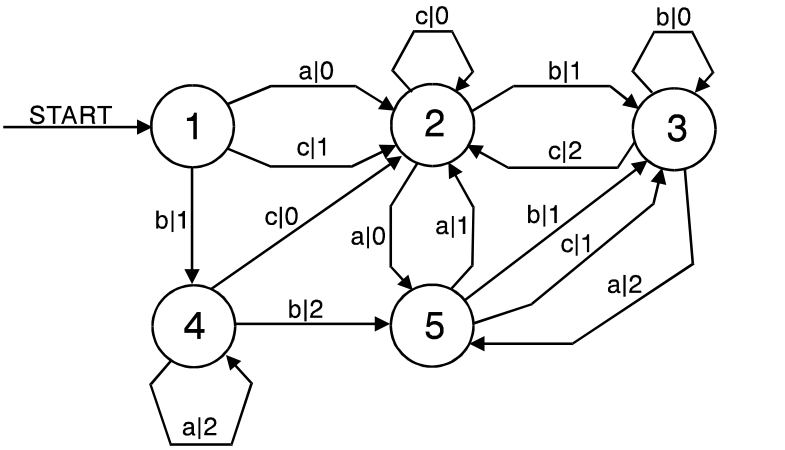
Pomocou rekurentného vzťahu (13) je možné v každom kroku trénovania nanovo prepočítať potrebné parciálne derivácie $\frac{\partial S_r^{(t)}}{\partial W_{i\ln}}$. Tieto sa použijú v ďalšom kroku pre nový výpočet parciálnych derivácií podľa vzťahu (13). Na začiatku trénovania je vhodné zvoliť $\frac{\partial S_r^{(0)}}{\partial W_{i\ln}}$ nulové.

1.2.1 Príklad trénovania rekurentej 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 obrázku 5.

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". Trénovcia množina by potom vyzerala nasledovne: acccb!→00001x, caaab!→10101x, bbbbbb!→121000x, atď, alebo acccb!caaab!bbbbbb!... →00001x10101x121000x... čo je v prepise do časovo usporiadanej trénovacej množiny (vstup, výstup): a→0, c→0, c→0, c→0, b→1, !→x, c→1, a→0, a→1, a→0, b→1, !→x, atď.



Obrázok 5: Mealyho automat použitý na generovanie trénovacej množiny pre rekurentnú sieť na obrázku 3.

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. Kódovanie môže mať napríklad takýto tvar:

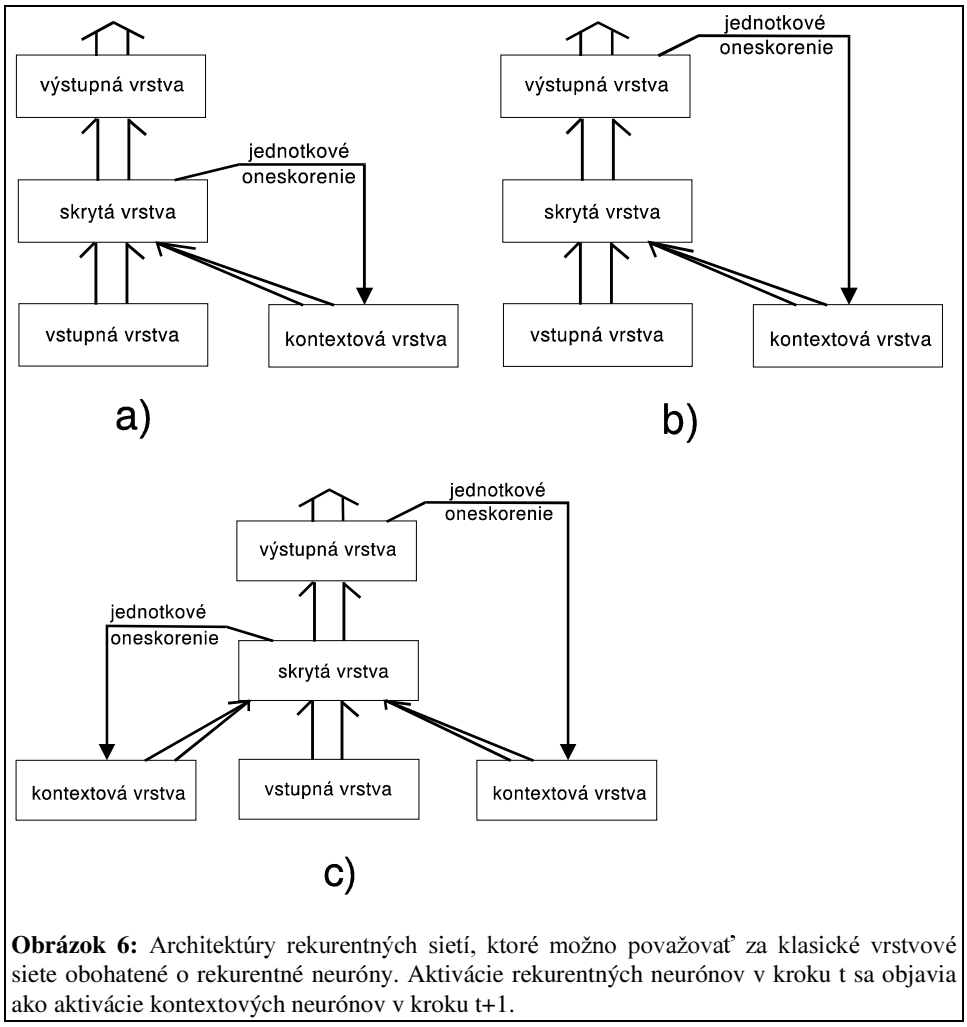
symbol	vstupný	výstupný	kód
	A	0	1000
	b	1	0100
	c	2	0010
	!	x	0001

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 ne-rekurentné neuróny (K = 4).

Trénovaciu množinu sme vygenerovali tak, že k 600 náhodne vybratým vstupným slovám nad abecedou $\{a,b,c\}$ sme 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 obrázku 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 nerónov) 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.



Obrázok 6: Architektúry rekurentných sietí, ktoré možno považovať za klasické vrstvové siete obohatené o rekurentné neuróny. Aktivácie rekurentných neurónov v kroku t sa objavia ako aktivácie kontextových neurónov v kroku t+1.

2. Rekurentné siete a ich trénovanie

2.1 Modely rekurentných sietí

V predošlej kapitole 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 aktiváciach v niektorých časoch $\leq t$ 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 kapitoly 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 vrstvové siete obohatené o rekurentné neuróny. Na obrázkoch 6a,b,c uvádzame tri modely tohoto typu, ktoré možno nájsť v literatúre. Vrstvy neurónov sú zobrazované obdĺžnikmi. Podobne ako v tradičných vrstvových sieťach, v rámci jednej vrstvy neuróny nie sú 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 repojenia 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 je odpamätanie aktivácií rekurentných neurónov a ich zavedenie do kontextových neurónov.

Architektúra na obrázku 6a bola navrhnutá Elmanom [14]. Kontextová vrstva obsahuje kópie aktivácií skrytých neurónov z predošlého kroku. Autorom siete predstavenej na obrázku 6b je Jordan [15]. Obrázok 6c predstavuje kombináciu modelov na obrázkoch 6a a 6b. Ako navrhuje Bengio [16], je možné mať zvláštnu kontextovú vrstvu pre rôzne vrstvy pôvodnej vrstvovej siete ako je tomu pri architektúre na obrázku 6c.

Ďalšia varianta spôsobu odpamätávania aktivácií rekurenných 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

$$K_i^{(t+1)} = \alpha K_i^{(t)} + S_i^{(t)} \tag{14}$$

$0 < \alpha < 1$ je konštanta reprezentujúca "rýchlosť zabúdania". Iterovaním (14) totiž dostávame

$$K_i^{(t+1)} = S_i^{(t)} + \alpha S_i^{(t-1)} + \alpha^2 S_i^{(t-2)} + \ldots = \sum_{\tau=0}^{t} \alpha^{t-\tau} S_i^{\tau} \tag{15}$$

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

Takýto typ odpamätávania aktivácií rekurentných neurónov budeme označovať čiarkovaným prepojením medzi rekurentnou a kontextovou vrstvou. Architektúra na obrázku 7a bola navrhnutá Jordanom [15]. Ide o rozšírenie siete z obrázku 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 obrázku 7b pochádza od Stornetta a kol. [17]. Ide vlastne o klasickú vrstvovú sieť, ktorej vstupné neuróny kódujú históriu predložených vstupov v minulosti. Mozer [18] navrhol architektúru na obrázku 7c. Sieť má k dispozícii informáciu o minulom vývoji aktivácií neurónov v skrytej vrstve.

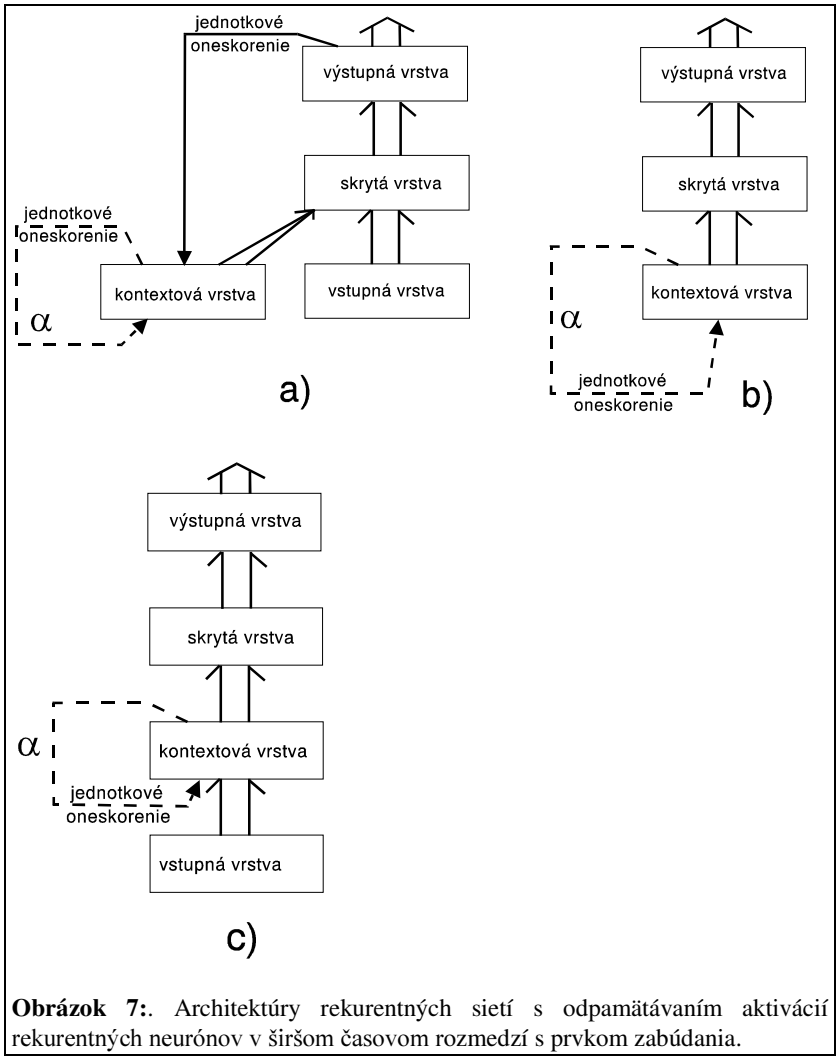
Je len pochopiteľné ak si čitateľ na tomto mieste položí dve otázky:
1. Prečo existuje toľko rôznych variant rekurentných sietí?
2. Ako sformulovať pravidlá pre učenie takýchto sietí ?

Opoveď 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 praktičnosti trénovacieho procesu pri danom spôsobe kódovania časového kontextu. zásade sa neurónové siete uplatňujú pri troch typoch úloh.

I. Klasifikačné alebo **asociačné úlohy** aké poznáme už z kapitoly o vrstvových sieťach, avšak **s časovým kontextom.** Pri prvom type úloh ide, v prípade klasifikácie, o rozhodnutie či je práve ukončená postupnosť vstupov z nejakej triedy, alebo nie, poprípade do ktorej z možných tried patrí. 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 kapitole.

II. Predikčné úlohy. 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.

III. Generatívne úlohy. 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 tentenciu 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.



Obrázok 7:. Architektúry rekurentných sietí s odpamätávaním aktivácií rekurentných neurónov v širšom časovom rozmedzí s prvkom zabúdania.

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 obrázku 6a bol úspešne použitý pre klasifikáciu kratších slov nad konečnou abecedou symbolov, ako aj na generovanie krátkodobých pokračovaní symbolických postupností [14].

Model z obrázku 7a bol použitý Andersonom [20] na katerizovanie hovorených slabík anglického jazyka. Sieť trénovaná na jednej skupine hlasov bola schopná správne fungovať pri nových, doteraz nepočutých hlasoch. Architektúra na obráazku 7c sa lepšie hodí na problém klasifikácie postupností ako pre generatívne úlohy [4].

2.2 Trénovanie rekurentných sietí

V predošlej kapitole sme si na príklade ukázali ako zovšeobecniť trénovaciu procedúru BP, pôvodne navrhnutú pre vrstvové siete, aby bola použiteľná aj pre vrstvové siete obsahujúce rekurentné neuróny. V tejto kapitole 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 postupného zostupu do minima chybového funkcionálu E pohybom proti smeru gradientu ∇E .

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

$$S_i^{(t+1)} = g\left(\sum_{j=1}^{2} w_{ij} S_j^{(t)} + I_i^{(t)}\right), \; i = 1, 2 \tag{16}$$

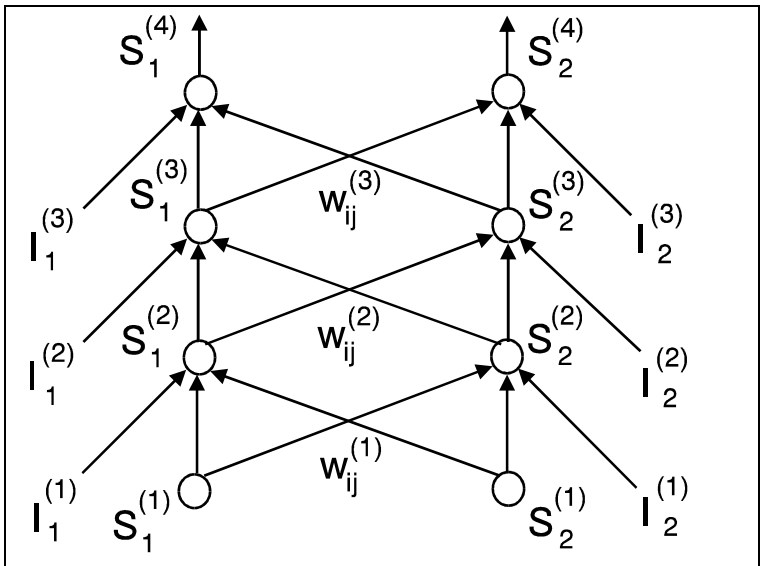
kde $S_1^{(t)}$ (resp. $S_2^{(t)}$) je aktivácia prvého (resp. druhého) rekurentného neurónu v čase t a $I_1^{(t)}$ (resp. $I_2^{(t)}$) je vonkajší vstup cez (kanál váhy 1) do prvého (resp. druhého) rekurentného neurónu v čase t, g môže byť napríklad známa sigmoidálna funkcia (2). Prípad, že aj vonkajšie vstupy vchádzajú do rekurentných neurónov cez kanály s modifikovateľnými váhami by bol riešený analogicky, ale prezentácia by bola zbytočne zaťažená.

2.2.1 Back Propagation Through Time (BPTT)

Predstavme si, že na vstup $\left(I_1^{(t)}, I_2^{(t)}\right)$ 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 uedomiť, že učiaci signál nie je vo všeobecnosti k dispozícii v každom kroku diskrétneho času (na rozdiel od príkladu uvedenom v predošllej kapitole). 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 vrstvovú 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.

Obrázok 8 redstavuje rekurentnú sieť rozvinutú v čase $1 \le t \le 4$ pri prezentácii vstupnej postupnosti dĺžky T = 3.



Obrázok 8: Časovo rozvinutá sieť s dvoma rekurentnými neurónmi v krokoch t=1,2,3 diskrétneho času. Každý vstup ide na každý neurón.

Idea rozvinutia rekurentnej siete v čase $1 \le t \le T$ bola pôvodne navrhnutá už Minskim 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 šíriť cez časovo rozvinutú sieť (obrázok 8) metódou BP.

1. Pri rátaní parciálnych derivácií $\frac{\partial E}{\partial w_{ij}}$ považujeme váhy[2] $w_{ij}^{(t)}$ v rôznych časoch t za nezávislé a štandardným postupom spätného chodu získame parciálne derivácie $\frac{\partial E}{\partial w_{ij}^{(t)}}, i, j = 1,2; t = 1,2,3.$

2. Modifikácia váhy w_{ij} bude priamo úmerná súčtu navrhnutých modifikácií v rôznych krokoch diskrétneho času

$$\Delta w_{ij} = -\varepsilon \sum_{t=1}^{T} \frac{\partial E}{\partial w_{ij}^{(t)}}$$

$\varepsilon: > 0$ je konštanta nazývaná rýchlosť učenia (learning rate), podobne ako v klasickej procedúre BP. Takáto procedúra trénovania rekurentných sietí založná na modifikácii tradičného prístupu BP v časovo rozvinutej sieti sa nazýva ***spätné šírenie v čase*** (**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, Rumelart, Hinton a Williams [22] ju úspešne použili pre učenie rekurentných sietí trénovaných imitovať správanie sa posuvného registra.

2.2.2 Real Time Recurrent Learning (RTRL)

Hlavná myšlienka tohoto prístupu k trénovaniu rekurentných sietí spočíva v úprave váh v <u>*každom*</u> 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 metódy ***rekurentné učenie v reálnom čase*** (RTRL) sú Williams a Zipser [23].

Uvažujme opäť jednoduchú dvoj-neurónovú sieť s dynamikou určenou vzťahom (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

$$E_k^{(t)} = \begin{cases} O_k - S_k^{(T+1)}, t = T+1 \\ 0, 1 \le t \le T \end{cases} \tag{17}$$

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

$$E^{(t)} = \frac{1}{2} \sum_{k=1}^{2} \left(E_k^{(t)}\right)^2 \tag{18}$$

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

$$\Delta w_{ij}(t) = -\varepsilon \frac{\partial E(t)}{\partial w_{ij}} \tag{19}$$

kde $\varepsilon > 0$ je rýchlosť učenia. Z (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 (13) (tentoraz máme neuróny prvého rádu) dostávame

[2] Váhy w označujú všetky typy váh v sieti, t.j. váhy medzi vstupmi a skrytými neurónmi, váhy medzi rozvinutými vrstvami skrytých neurónov a váhy medzi najvyššou skrytou vrstvou a výstupnou vrstvou, ktorá nie je na obrázku pre prehľadnosť znázornená. Neuróny môžu mať aj trénovateľné prahy.

$$\frac{\partial S_r^{(t)}}{\partial w_{ij}} = g'\left(\phi\left(S_r^{(t)}\right)\right)\left[\delta_{ri} S_j^{(t-1)} + \sum_{a=1}^{2} w_{ra} \frac{\partial S_a^{(t-1)}}{\partial w_{ij}}\right] \tag{20}$$

Pripomíname, že podobne ako vo vzťahu (13), ϕ je inverzná funkcia k funkcii g a δ_{ri} je Kroneckerovo delta $\delta_{ri} = \begin{matrix} 1, r = i \\ 0, r \neq i \end{matrix}$. Je vhodné celý postup inicializovať postulovaním $\frac{\partial S_r^{(0)}}{\partial W_{i\ln}} = 0.$

Analogicky príkladu z úvodnej kapitoly, aj tu využívame vzťah (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] sme použili momentový člen pre zvýšenie robustnosti (anti)gradientového poklesu na chybovom povrchu voči slabším lokálnym minimám.

Metóda RTRL našla živnú pôdu medzi používateľmi 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 (pozri časť 1.2.) na základe príkladov jeho činnosti [13] [26] a bola použitá aj v pípade inferencie iných automatov rekurentnými sieťami [28] [29].

3. Na záver

Po úvodnej časti intuitívne navodiacej potrebu neurónových sietí s vnútornou pamäťou a naznačujúcej spôsob učenia takýchto sietí sme si v 2. časti organizovanejším spôsobom predstavili niektoré zákadné 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 pre, napríklad, predikciu možného budúceho vývoja postupnosti. V mnohých praktických aplikáciach je úspešnosť rekurentných neurónových sietí zrovnateľná 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 pomene malá. Pre dosiaľ nevidenú vzorku sieť síce ponúkne odpoveď, ovšem 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 vyššou pamäťovou a časovou náročnosťou.

Značné úsilie je venované aj skúmaniu rekurentných neurónových sietí aparátom teórie dynamických systémov [32]. ekurentné siete toti mo no pova ovať za dynamické systémy. Parametrami zobrazenia stavového priestoru sú váhy synaptických prepojení. Keďže premenlivé vonkajsie 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 dynamickýh systémov sú

1) popis invariantných atraktívnych množín v stavovom priestore siete – Invariantné atraktívne množiny sú dôležitým faktorom pri determinovaní asymptotického správania rekurentnej siete [33] [34] [35] [36]. Zaujímavým výsledkom bol aj experimentálny a analytický dôkaz existencie chaotického režimu v rekurentnýh 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 siete nezodpovedá želanému stavu. Objasnenie bifurkačného mechanizmu vzniku napríklad nových atraktívnych pevných bodov [13] [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 chotických trajektórií na malé zmeny v počiatočných podmienkach. Okrem toho, aj keď postupnosť nie je chaotická, ale len dostatočne zložitá, t.j. zahàňa korelácie medzi časovo značne vzdialenými prvkami postupnosti, učenie rekurených 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 v režime na použitej 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) vrstvovými 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 vrstvových siťach, 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 a 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 a P.A. Ligomenides. Trajectory Production With The Adaptive Time-Delay Neural Network. Neural Networks, 3: 447–461, 1995.

[6] U. Bodenhausen a A. Waibel. The Tempo2 Algorithm: Adjusting Time-Delays By Supervised Learning. V knihe J.E. Moody, R.P. Lippmann a D.S. Touretzky, editori, Advances in neural information processing systems, str. 155–161, Vol. 3, Morgan Kaufmann Publishers, Inc., San Mateo, California, 1991.

[7] J.L. McClelland a J.L.Elman. Interactive Processes in Speach Perception: The TRACE Model. V knihe J.L. McClelland, D.E. Rumelhart a PDP Research Group, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 2, kapitola 15, MIT Press, Cambridge, 1986.

[8] J.L.Elman a D. Zipser. Learning the Hidden Structure of Speach. Journal of the Acoustical Society of America, 83: 1615–1626, 1988.

[9] A. Weibel. Modular Construction of Time–Delay Neural Networks for Speach Recognition. Neural Computation, 1: 39–46, 1989.

[10] R.P. Lippmann. Review of Neural Networks for Speach Recognition. Neural Computation, 1: 1–38, 1989.

[11] J.E. Hopfcroft a 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 a 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. V knihe J.L. Elman a D.E. Rumelhart, editori, Advances in Connectionist Theory: Speach, Erlbaum, Hillsdale, 1989.
[16] Y. Bengio, R. Cardin a R. De Mori. Speaker Independent Speech Recognition with Neural Networks and Speech Knowledge. V knihe D.S. Touretzky, editor, Advances in neural information processing systems II, str. 218–225, Vol. 3, Morgan Kaufmann Publishers, Inc., San Mateo, California, 1990.
[17] W.S. Stornetta, T. Hogg a B.A. Huberman. A Dynamical Approach to Temporal Pattern Processing. V knihe D.Z. Anderson, editor, Neural Information Processing Systems, str. 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 a 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 a R. Port. Dynamic Speech Categorization With Recurrent Networks. V knihe D.S. Touretzky, G. Hinton a T. Sejnowski, editori, Proceedings of the 1988 Connectionist Models Summer School, Pittsburg, str. 398–406, Morgan Kaufmann Publishers, Inc., San Mateo, California, 1989.
[21] M.L. Minski a S.A. Papert. Perceptrons. MIT Press, Cambridge, 1969.
[22] D.E. Rumelhart, G.E. Hinton a R.J. Williams. Learning Internal Representations by Error Propagation. V knihe J.L. McClelland, D.E. Rumelhart a PDP Research Group, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations, kapitola 8, MIT Press, Cambridge, 1986.
[23] R.J. Williams a D. Zipser. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. Neural Computation, 1: 270–280, 1989.
[24] R.J. Williams a 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 a P.C. Collingwood. Finite State Machines and Recurrent Neural Networks – Automata and Dynamical Systems Approaches. V knihe J.E. Dayhoff a O. Omnivar, editori, 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 a J.L. McClelland. Finite State Automata and Simple Recurrent Neural Networks. Neural Computation, 3: 372–381, 1989.
[29] Z. Zeng, R.M. Goodman a 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. V knihe A. Weigend a N. Gershenfeld, editori, 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 a P. Holmes. Nonlinear Oscilations, Dynamical Systems, and Bifurcations 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 a 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 a X. Wang. Stability of Fixed Points and Periodic Orbits and Bifurcations in Analog Neural Networks. Neural Networks, 5: 577–587, 1992.

[37] X. Wang. Period-Doublings to Chaos in a Simple Neural Network: An Analytical Proof. Complex Systems, 5: 425–441, 1991.
[38] K. Hornik, M. Stinchcombe a H. White. Multilayer Feedforward Networks Are Universal Aproximators. Neural Networks, 2: 359–366, 1989.
[39] K. Doya. Bifurcations in the Learning of Recurrent Neural Networks. V knihe, Proceedings of 1992 IEEE International Symposium on Circuits and Systems, str. 2777–2780, 1992.
[40] Y. Bengio, P. Simard a P. Frasconi. Learning Long–Term Dependences with Gradient Is Difficult. IEEE Transactions on Neural Networks, 2: 157–166, 1994.
[41] J.M. Kuo a J.C. Principe. A Systematic Approach to Chaotic Time Series Modeling With Neural Networks. V knihe, IEEE Workshop on Neural Nets for Signal Processing, Ermioni, Greece, 1994.
[42] J.C. Principe, A. Rathie a 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.