Rekurentné neurónové siete

Michal ČERŇANSKÝ[1]

Abstrakt. Neurónové siete patria medzi úspešne používané nástroje strojového učenia. Rekurentné neurónové siete boli navrhnuté na modelovanie časových postupností a na rozdiel od klasických dopredných neurónových sietí obsahujú rekurentné prepojenia, ktoré umožňujú, aby informácia z predchádzajúceho obdobia mohla ovplyvniť aktivity neurónov v aktuálnom čase. Rekurentné siete boli úspešne použité v mnohých reálnych aplikáciách pri riešení úloh predikcie, adaptívneho riadenia, identifikácie systémov či spracovania signálu. Významným faktorom limitujúcim ich širšie použitie je komplexnosť algoritmov používaných na ich trénovanie.

1 Úvod

Umelé neurónové siete sú výpočtové modely inšpirované biologickými neurónovými sieťami, akou je napríklad aj ľudský mozog. Paralela s prírodou je v spôsobe realizácie výpočtu – výsledný zložitý výpočet je vytvorený interakciou mnohých elementov realizujúcich jednoduchšie výpočty. Bolo navrhnuté veľké množstvo typov neurónových sietí [10][17], avšak medzi najpopulárnejšie určite patria tzv. viacvrstvové perceptrónové neurónové siete, ktoré sú trénované algoritmom spätného šírenia chyby. V ďalšom texte pod pojmom neurónová sieť budeme označovať práve neurónové siete zložené z perceptrónov.

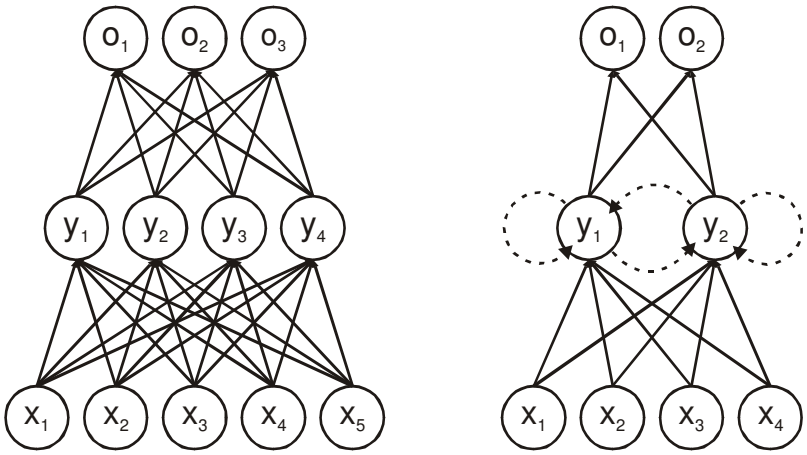
Z inžinierskeho hľadiska sú neurónové siete populárne a často používané výpočtové modely, a to najmä vzhľadom na jednoduchosť ich použitia a schopnosť rýchlo dosiahnuť uspokojivé výsledky. Neurónová sieť pozostáva z väčšieho počtu výpočtových elementov, nazývaných neuróny, ktoré sú vzájomne pospájané cez váhové prepojenia s rôznou intenzitou. Trénovanie neurónovej siete pozostáva z modifikovania intenzít váhových prepojení tak, aby neurónová sieť adekvátne reagovala na vstupy, t.j. aby produkovala požadované výstupy. Príklad jednoduchej neurónovej siete je znázornený na obr. 1a.

Klasické dopredné neurónové siete bývajú používané napr. pri riešení úloh klasifikácie (určenie, či objekt má danú vlastnosť) alebo regresie (nájdenie závislosti popisujúcej vzťah medzi premennými). Pre spracovanie úloh s časovým kontextom,

[1] Ústav aplikovanej informatiky, FIIT, STU Bratislava, E-mail: cernansky@fiit.stuba.sk

akou je napríklad predikcia (predpovedanie nasledujúcej hodnoty alebo hodnôt v postupnosti), boli navrhnuté rekurentné neurónové siete.

Rekurentné neurónové siete môžu byť považované za jednoduchú modifikáciu dopredných neurónových sietí, ktorá vzniká pridaním tzv. rekurentných prepojení. Rekurentné prepojenia spájajú neuróny takým spôsobom, že aktivita neurónov z predchádzajúceho časového kroku cez rekurentné prepojenie ovplyvňuje aktivitu neurónu v aktuálnom časovom kroku. Príklad rekurentnej neurónovej siete je znázornený na obr. 1b.



Obr. 1. Príklad (a) doprednej a (b) rekurentnej neurónovej siete. Dopredné prepojenia prenášajúce informáciu v rámci aktuálneho kroku výpočtu sú znázornené plnou čiarou a rekurentné prepojenia prenášajúce informáciu z minulého do súčasného kroku výpočtu sú znázornené prerušovanou čiarou.

Rekurentné neurónové siete boli úspešne použité vo viacerých praktických úlohách vyžadujúcich modelovanie časových postupností. Bolo navrhnutých viacero architektúr, trénovacích algoritmov a ich modifikácií, avšak rekurentné neurónové siete stále nie sú bežne používanými nástrojmi strojového učenia. Dôvodom je výrazne vyššia výpočtová náročnosť trénovacích algoritmov v porovnaní s klasickými doprednými neurónovými sieťami. Úspešná aplikácia rekurentných sietí si tiež zvyčajne vyžaduje hlbšiu znalosť problémovej úlohy, použitého typu rekurentnej siete, ako aj zvoleného trénovacieho algoritmu.

Cieľom tejto práce je zoznámiť čitateľa s problematikou rekurentných neurónových sietí a uľahčiť mu ich prípadnú aplikáciu. Štruktúra tejto práce je nasledovná: v ďalšej kapitole stručne predstavíme dopredné neurónové siete a zoznámime sa so spôsobom zápisu algoritmov pre dopredné, ako aj rekurentné neurónové siete, ktorý bude použitý v nasledujúcich kapitolách. Spôsob zápisu bude demonštrovaný na známom algoritme spätného šírenia chyby. V tretej kapitole si predstavíme niektoré z architektúr rekurentných neurónových sietí, s ktorými sa môže čitateľ stretnúť v odbornej literatúre. V štvrtej kapitole si vysvetlíme bežné algoritmy založené na gradientovej minimalizácií chyby a v piatej kapitole popíšeme aj

algoritmus využívajúci Kalmanovu filtráciu, ktorý v porovnaní s bežne používanými prístupmi vykazuje v mnohých parametroch výrazne lepšiu výkonnosť. V šiestej kapitole sa na úlohách modelovania postupností symbolov pokúsime preniknúť do spôsobu fungovania rekurentných neurónových sietí. Predstavíme si niektoré základné dynamické správania pozorovateľné v stavovej časti rekurentnej siete. V siedmej kapitole opíšeme dve praktické aplikácie rekurentných neurónových sietí. V poslednej kapitole využijeme navrhnutý spôsob zápisu algoritmov a navrhneme jednoduchý a elegantný spôsob zakódovania rekurentnej siete do údajových štruktúr. Potom trénovacie algoritmy formálne opísané v tretej kapitole zapíšeme v jednoduchom pseudojazyku v podobe, ktorá umožňuje ich takmer okamžité použitie v bežných procedurálnych programovacích jazykoch, akými sú napríklad jazyk C či Java.

2 Dopredné neurónové siete

Viacvrstvová neurónová sieť zložená z perceptrónov (angl. Multilayer Perceptron) je najčastejšie používaný typ neurónových sietí [17]. Prívlastok „dopredná“ (angl. Feedforward) používame v kontexte rekurentných neurónových sietí na zdôraznenie smeru toku informácie. V tejto podkapitole stručne zhrnieme základné informácie o dopredných neurónových sieťach a v krátkosti popíšeme algoritmus spätného šírenia chyby. Tiež sa oboznámime so spôsobom zápisu algoritmov nezávislom na konkrétnej architektúre.

2.1 Neurón

V oblasti umelých neurónových sietí (inšpirujúc sa neurobiológiou) pojmom neurón označujeme jednoduché elementy, z ktorých sú umelé neuronové siete zložené. Perceptrón je neurón, ktorý realizuje jednoduchý matematický výpočet. Obsahuje N vstupov x_1 až x_N, ktoré sú na neho pripojené prostredníctvom váhových prepojení s intenzitami w_1 až w_N. Intenzita, čiže váha prepojenia určuje, ako výrazne daný vstup ovplyvňuje výslednú aktivitu neurónu. Výpočet, ktorý neurón realizuje, je teda možné zapísať ako:

$$\tilde{o} = \sum_{i=1}^{N} w_i x_i + \Theta = \sum_{i=1}^{N} w_i x_i + w_0 = \sum_{i=0}^{N} w_i x_i \,, \tag{1}$$

$$o = f(\tilde{o}), \tag{2}$$

kde $\tilde{o}$ označuje tzv. vnútornú aktivitu neurónu, o označuje výstupnú aktivitu neurónu a Θ je prah neurónu. Je vhodné prah neurónu považovať za špeciálnu váhu w_0 ($\Theta = w_0$) smerujúcu zo špeciálneho vstupu x_0 vždy nastaveného na konštantu 1. V literatúre často uvádzaný vzťah, v ktorom je od sumy $\sum w_i x_i$ prah Θ odčítavaný, je ekvivalentný s rovnicou (1) vzhľadom na to, že prah môže byť aj záporný. Funkcia f je tzv. aktivačná alebo prechodová funkcia. Veľmi často používaná aktivačná funkcia

je sigmoidálna funkcia. Uvádzame vzťah na výpočet funkčnej hodnoty a tiež výpočet derivácie:

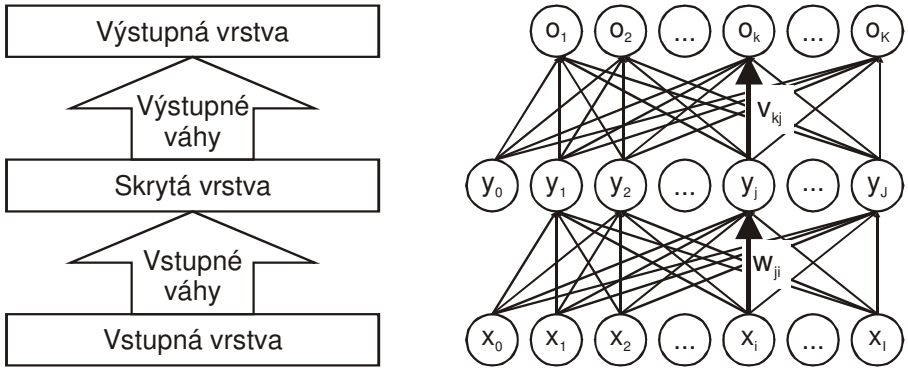
$$f(x) = \frac{1}{1+e^{-x}}, \tag{3}$$

$$f'(x) = f(x)(1 - f(x)). \tag{4}$$

V nasledujúcom texte pojmom neurón budeme označovať perceptrón realizujúci uvedený jednoduchý výpočet.

2.2 Viacvrstvová neurónová sieť

Napriek tomu, že vo všeobecnosti je možné skonštruovať doprednú neurónovú sieť so skoro ľubovoľne poprepájanými neurónmi, v praxi sa zvyčajne používajú neurónové siete s neurónmi organizovanými do vrstiev. Najčastejšie používaná architektúra je trojvrstvová neurónová sieť a je znázornená na obr. 2. Vstupné neuróny nerealizujú nijaký výpočet, iba udržujú vstupnú informáciu (a preto sa niekedy architektúra označuje ako dvojvrstvová sieť, obsahuje iba dve vrstvy „skutočných“ neurónov). Zo vstupnej vrstvy sa informácia šíri na skrytú vrstvu obsahujúcu skryté neuróny. Zvyčajne každý skrytý neurón je prostredníctvom vstupných váh prepojený s každým vstupným neurónom. Z neurónov skrytej vrstvy sa informácia šíri na výstupné neuróny na výstupnej vrstve. Opäť, každý výstupný neurón je prostredníctvom výstupných váh prepojený s každým skrytým neurónom.



Obr. 2. (a) Schématické znázornenie trojvrstvovej neurónovej siete. (b) Trojvrstvová neurónová sieť zložená z I vstupných, J skrytých a K výstupných neurónov.

Neurónová sieť z obr. 2b má I vstupných neurónov, na ktoré sa pred výpočtom umiestni vstupná informácia – I-tica reálnych čísel $(x_1, x_2, ..., x_i, ..., x_I) \in R^I$. Aktivity skrytých neurónov – J-tica reálnych čísel $(y_1, y_2, ..., y_j, ..., y_J) \in R^J$, sú vypočítané ako:

$$\tilde{y}_j = \sum_{i=0}^{I} w_{ji} x_i \,, \tag{5}$$

$$y_j = f\left(\tilde{y}_j\right), \tag{6}$$

kde f je aktivačná funkcia (najčastejšie sigmoida) a $\tilde{y}_j$ je vnútorný potenciál skrytého neurónu j. Aktivity výstupných neurónov – K-tica reálnych čísel $\left(o_1, o_2, ..., o_k, ..., o_K\right) \in R^K$, sú vypočítané ako:

$$\tilde{o}_k = \sum_{j=0}^{J} v_{kj} y_j \,, \tag{7}$$

$$o_k = f\left(\tilde{o}_k\right), \tag{8}$$

kde f je aktivačná funkcia (najčastejšie sigmoida alebo lineárna) a $\tilde{o}_k$ je vnútorný potenciál výstupného neurónu k. Tento výpočet predpokladá, že špeciálne „prahové“ neuróny x_0 a y_0 sú nastavené na hodnotu 1, a teda je možné s prahom j-teho skrytého neurónu w_{j0} a prahom k-teho výstupného neurónu v_{k0} pracovať ako s bežnou váhou. V ďalšom texte nebudeme špeciálne odlišovať prahy a váhy neurónovej siete.

2.3 Trénovanie neurónových sietí - spätné šírenie chybového signálu

Významnou vlastnosťou neurónových sietí je možnosť ich natrénovať na základe predpripravených vstupno-výstupných vzoroch. Trénovacia množina je tvorená typickými vstupmi a k nim prislúchajúcimi očakávanými výstupmi. Po natrénovaní sa od neurónovej siete očakáva, že dokáže zovšeobecňovať, čiže správne spracovávať aj vstupy, ktoré trénovacia množina neobsahovala.

Najznámejším a najčastejšie používaným algoritmom na trénovanie neurónových sietí je algoritmus spätného šírenia chybového signálu (ang. Error Backpropagation alebo iba Backpropagation, BP) [29]. Umožňuje vhodným spôsobom organizovať výpočet derivácií potrebných pre zmenu váh pomocou gradientovej metódy najprudšieho spádu. Tieto derivácie sa vypočítavajú postupne od neurónov na vyšších vrstvách smerom k neurónom na nižších vrstvách.

Typický scenár trénovanie neurónovej siete je nasledovný:

Príprava trénovacej (a aj testovacej) množiny
Riešená úloha je transformovaná do podoby umožňujúcej jej riešenie pomocou neurónovej siete. Je potrebné rozhodnúť, čím budú tvorené vstupy do neurónovej siete a čo budeme očakávať na výstupe neurónovej siete. Výstupom z tejto fázy je trénovacia (a väčšinou aj testovacia množina) pozostavajúca z viacerých vstupno-výstupných vzorov $\left(\mathbf{x}, \mathbf{d}\right)$, kde $\mathbf{x} = \left(x_1, ..., x_I\right)$ je vstupný vektor a $\mathbf{d} = \left(d_1, ..., d_K\right)$ je k nemu prislúchajúci vektor očakávaných hodnôt. Takýchto vzorov obsahuje trénovacia aj testovacia množina väčší počet.

Inicializácia neurónovej siete

Inicializácia spočíva vo vytvorení neurónovej siete. Hodnoty váhových prepojení sú iniciované náhodne na malé hodnoty, napr. z intervalu $(-0.5,0.5)$. Počet vstupných neurónov siete je samozrejme daný rozmerom vstupného vektora a počet výstupných neurónov je rovný rozmeru vektora požadovaných hodnôt. Počet skrytých neurónov je možné zvoliť experimentálne, typicky sú to jednotky až desiatky neurónov. Zvyčaje sa konštruujú siete s jedinou vrstvou skrytých neurónov.

Trénovanie metódou spätného šírenia chybového signálu

Náhodne sa vyberie jeden z prvkov trénovacej množiny $(\mathbf{x},\mathbf{d})$ a vstupný vektor $\mathbf{x}$ je umiestnený na vstup siete. Dopredným šírením sa vypočítajú skryté a potom aj výstupné aktivity $\mathbf{o}=(o_1,o_2,...,o_k,...,o_K)$. Po doprednom šírení očakávame na výstupe siete požadované hodnoty $\mathbf{d}=(d_1,d_2,...,d_k,...,d_K)$, ktoré sa zvyčajne od sieťou vypočítaných hodnôt $\mathbf{o}$ líšia. Túto chybu, vzniknutú na výstupe, sa snažíme minimalizovať gradientovou metódou najprudšieho spádu. Chybu v danom kroku trénovacieho algoritmu definovanú ako

$$E=\frac{1}{2}\sum_{k=1}^{K}(d_k-o_k)^2\,, \tag{9}$$

chceme minimalizovať gradientovou metódou najprudšieho spádu, a teda váhy chceme zmeniť pomocou vzťahov:

$$\Delta w_{ji}(t)=-\alpha\frac{\partial E}{\partial w_{ji}}+\beta\Delta w_{ji}(t-1)\,, \tag{10}$$

$$\Delta v_{ji}(t)=-\alpha\frac{\partial E}{\partial v_{kj}}+\beta\Delta v_{ji}(t-1)\,, \tag{11}$$

kde rýchlosť učenia α určuje mieru zmeny váhy. Cez momentum β je aktuálna zmena váhy ovplyvnená predchádzajúcou zmenou váhy. Momentum pôsobí ako istá zotrvačnosť pri menení váh, urýchľuje trénovanie a pomáha zabrániť uviaznutiu v lokálnych minimách chybovej funkcie.

Derivácie pre metódu najprudšieho spádu $\partial E/\partial w_{ji}$ a $\partial E/\partial v_{kj}$ sú vypočítané prostredníctvom spätne šíreného chybového signálu $\partial E/\partial\tilde{o}_k=\delta_k^{\text{out}}$ a $\partial E/\partial\tilde{y}_j=\delta_{\text{j}}^{\text{hid}}$ nasledovným spôsobom. Najskôr je vypočítaný chybový signál pre výstupné neuróny:

$$\delta_{\text{k}}^{\text{out}}=(d_k-o_k)f'(\tilde{o}_k)\,, \tag{12}$$

a následne je možné vypočítať chybový signál pre skryté neuróny:

$$\delta_{\text{j}}^{\text{hid}}=f'(\tilde{y}_j)\sum_{k=1}^{K}v_{kj}\delta_{\text{k}}^{\text{out}}\,. \tag{13}$$

Po vypočítaní chybových signálov $\delta_{\mathrm{k}}^{\mathrm{out}}$ a $\delta_{\mathrm{j}}^{\mathrm{hid}}$ je možné vypočítať zmeny pre výstupné $\Delta v_{kj}(t)$ a vstupné $\Delta w_{ji}(t)$ váhové prepojenia:

$$\Delta v_{kj}(t) = \alpha \delta_{\mathrm{k}}^{\mathrm{out}} y_j + \beta \Delta v_{kj}(t-1), \tag{14}$$

$$\Delta w_{ji}(t) = \alpha \delta_{\mathrm{j}}^{\mathrm{hid}} x + \beta \Delta w_{ji}(t-1), \tag{15}$$

a váhové prepojenia následne zmeniť:

$$v_{kj}(t+1) = v_{kj}(t) + \Delta v_{kj}(t), \tag{16}$$

$$w_{ji}(t+1) = w_{ji}(t) + \Delta w_{ji}(t). \tag{17}$$

Aby sme odlíšili pôvodné a nové hodnoty pre váhové prepojenia, sú váhy aj zmeny váh v rovnici (14) až (17) uvádzané s časovými indexmi.

Ukončenie trénovania a otestovanie siete
Algoritmus ukončíme, ak uplynul stanovený počet krokov algoritmu, popr. ak po stanovenom počte krokov algoritmu je výsledná chyba uspokojivo malá. Výslednú chybu vypočítame ako súčet čiastkových chýb vypočítaných podľa rovnice (9) po prechode cez celú trénovaciu alebo testovaciu množinu. Ak chyba ešte neklesla pod želanú úroveň, v trénovaní pokračujeme.

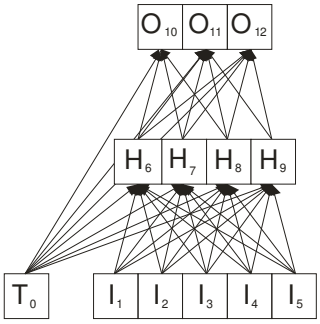
2.4 Alternatívny spôsob zápisu algoritmu spätného šírenia chyby

V literatúre je možné nájsť veľa spôsobov zápisu algoritmu spätného šírenia chyby. Vyššie uvedený spôsob zápisu algoritmu patrí medzi tie prehľadnejšie a ľahko pochopiteľné. Napriek tomu v tejto podkapitole uvedieme alternatívny spôsob zápisu algoritmu, ktorý s úspechom využijeme pri zápise algoritmov pre rekurentné neurónové siete. Tento alternatívny spôsob zápisu nebude závislý od konkrétnej architektúry neurónovej siete, nebude ani vyžadovať organizáciu neurónov do vrstiev, a ako uvidíme v poslednej kapitole, umožní nám jednoduchým a priamočiarym spôsobom zapísať algoritmy v pseudojazyku veľmi blízkom klasickým programovacím jazykom, akým je jazyk C či Java.

Od neurónovej siete budeme vyžadovať, aby jej neuróny boli usporiadané a očíslované tak, aby neurón s vyšším indexom bol cez svoje váhové prepojenia spojený iba s neurónmi s nižšími indexmi. Každá dopredná neurónová sieť spĺňa túto podmienku, ktorá vlastne vyjadruje fakt, že výpočet aktivít v doprednej sieti je možné realizovať dopredným spôsobom, a to postupným vypočítaním aktivít pre neuróny s indexom od 0 po $N-1$, kde N je počet neurónov v sieti.

Neurónová sieť s očíslovanými neurónmi je znázornená na obr. 3. Neurón s indexom 0 je špeciálny vstupný neurón vždy nastavený na konštantu 1, zvyčajne každý neurón, ktorý nie je vstupný, je na neho pripojený prostredníctvom váhového prepojenia plniaceho funkciu prahu.

Aj váhové prepojenia sú označené indexmi 0 až $M-1$, pričom M je počet všetkých váhových prepojení. Váhové prepojenie je vlastne trojica hodnôt (i, j, v), kde i je index cieľového neurónu a j je index zdrojového neurónu. Váha, čiže intenzita váhového prepojenia, je označená symbolom v. Aby sme sprístupnili jednotlivé položky usporiadanej trojici reprezentujúcej váhové prepojenie, zadefinujeme si operátory d, s a w. Operátor d (d ako angl. „destination“) nám sprístupní cieľový neurón váhového prepojenia s daným indexom, operátor s (s ako angl. „source“) nám sprístupní zdrojový neurón a w intenzitu váhového prepojenia. Nech (i, j, v) je váhové prepojenie s indexom l, tak potom $d_l = i$, $s_l = j$ a $w_l = v$. Množinu indexov všetkých váhových prepojení označíme W.



Obr. 3. Dopredná neurónová sieť zložená z 5 vstupných (I_1 až I_5), 4 skrytých (H_1 až H_4) a 3 výstupných (O_1 až O_3) neurónov. T_0 je špeciálny neurón s konštantnou jednotkovou aktivitou.

Dopredné šírenie je možné zapísať nasledovným spôsobom: neurón s indexom 0 je nastavený na konštantu 1, neuróny tvoriace vstupy do neurónovej siete sú nastavené na hodnoty podľa vstupného vzoru aktuálne prezentovaného sieti. Aktivity na ostatných neurónoch sú dopredným spôsobom (rastúci index i) nastavené na základe vzťahu:

$$\tilde{x}_i = \sum_{\forall j \in W \mid d_j = i} w_j x_{s_j}\,, \tag{18}$$

$$x_i = f\left(\tilde{x}_i\right), \tag{19}$$

kde zápis $j \in W \mid \mathrm{d}_\mathrm{j} = i$ je množina indexov takých váhových prepojení, ktorých cieľový neurón je neurón s indexom i. Sú to vlastne všetky váhové prepojenia patriace neurónu i. Cez váhové prepojenie w_j sa šíri aktivita x_{s_j} zdrojového neurónu s indexom s_j.

Spätné šírenie je možné zapísať podobne jednoduchým spôsobom. Chybový signál δ_i neurónu s indexom i je vypočítaný ako:

$$\delta_i = \left[(d_i - x_i) + \sum_{\forall j \in W \mid \mathrm{s_j} = i} w_j \delta_{\mathrm{d_j}} \right] f'(\tilde{x}_i). \tag{20}$$

Chyba $d_i - x_i$ vzniká iba na výstupných neurónoch, a iba keď je definovaný želaný výstup d_i. V opačnom prípade uvažujeme $d_i = x_i$, a teda člen $d_i - x_i = 0$. Pri výpočte chybového signálu δ_i sa cez všetky váhové prepojenia vychádzajúce z neurónu i prešíria už vypočítané chybové signály. Indexy váhových prepojení vychádzajúcich z neurónu i sú určené zápisom $\forall j \in W \mid \mathrm{s_j} = i$: sú to také prepojenia indexované pomocou j, ktorých zdrojový neurón je i. Výpočet je nevyhnutné realizovať od neurónov s vyšším indexom k neurónom s nižším indexom. Po vypočítaní chybových signálov je možné zmeniť váhy podľa:

$$\Delta w_i(t) = \alpha \delta_{\mathrm{d_i}} x_{\mathrm{s_i}} + \beta \Delta w_i(t-1), \tag{21}$$

$$w_i(t+1) = w_i(t) + \Delta w_i(t), \tag{22}$$

kde α je rýchlosť učenia a β je hodnota určujúca momentum.

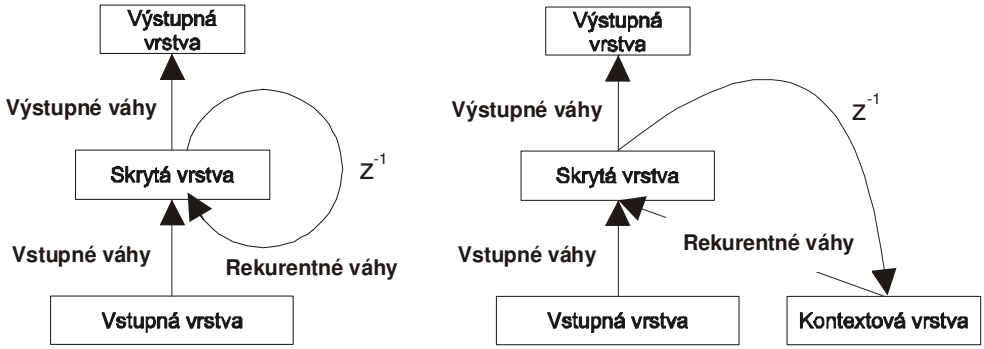
V uvedenom zápise dopredné šírenia signálu a aj spätného šírenia chybového signálu pri algoritme backpropagation nerozlišujeme vstupné, skryté a výstupné neuróny. Taktiež nepožadujeme, aby boli neuróny organizované do vrstiev, v ktorých je každý neurón z vrstvy spojený s každým neurónom predchádzajúcej vrstvy. S úspechom teda môžeme využiť uvedený zápis na trénovanie doprednej neurónovej siete s ľubovoľne poprepájanými neurónmi.

3 Rekurentné neurónové siete

Rekurentné neurónové siete (angl. Recurrent Neural Networks, RNNs) vznikli jednoduchým rozšírením dopredných sietí o váhové prepojenia prenášajúce do aktuálneho kroku výpočtu aktivity z predchádzajúceho časového kroku. Tieto prepojenia tvoriace v grafe reprezentujúceho neurónovú sieť cykly nazývame rekurentné váhové prepojenia [10][17].

3.1 Jednoduchá rekurentná neurónová sieť

Jednou z prvých navrhnutých architektúr je Elmanova jednoduchá rekurentná neurónová sieť (angl. Elman's Simple Recurrent Network, SRN) [7]. Elmanova architektúra je znázornená na obr. 4.



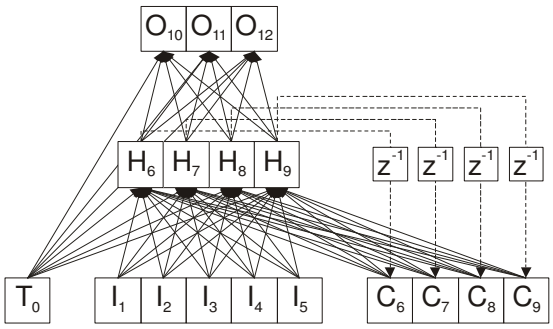
Obr. 4. Elmanova jednoduchá rekurentná neurónová sieť zobrazená (a) bez použitia kontextovej vrstvy a (b) s použitím kontextovej vrstvy. Časové oneskorenie je znázornené pomocou z^{-1}.

Výpočet aktivít pre skryté aj výstupné neuróny je realizovaný obdobným spôsobom, ako aj v prípade doprednej neurónovej siete. Po vypočítaní aktivít na skrytej vrstve sú tieto prekopírované na tzv. kontextovú vrstvu, z ktorej budú prostredníctvom rekurentých váh v nasledujúcom časovom kroku ovplyvňovať skryté neuróny. Takýmto spôsobom má sieť informáciu z predchádzajúceho obdobia a aktivity na kontextovej vrstve teda plnia funkciu pamäte. Je dôležité si uvedomiť, že kontextová vrstva je iba pseudovrstvou, v skutočnosti reprezentuje skrytú vrstvu z predchádzajúceho časového kroku. Rekurentné váhy na obr. 4b znázornené od kontextových ku skrytým neurónom sú v skutočnosti prepojenia zo skrytej vrstvy na seba samú (preto sa volajú rekurentné) tak, ako je uvedené na obr. 4a. Od doprednych váh sa líšia tým, že majú jednotkové časové oneskorenie, čiže sa ich prostredníctvom šíri aktivita z predchádzajúceho časového kroku.

Elmanova sieť je charakteristická tým, že pamäť je tvorená aktivitami neurónov skrytej vrstvy z predchádzajúceho časového kroku. Vo všeobecnosti ako pamäť môžu byť použité aktivity ľubovoľných neurónov a rovnako nemusia byť z bezprostredne predchádzajúceho časového kroku (môžu byť aj staršie).

3.2 Architektúry rekurentných neurónových sietí

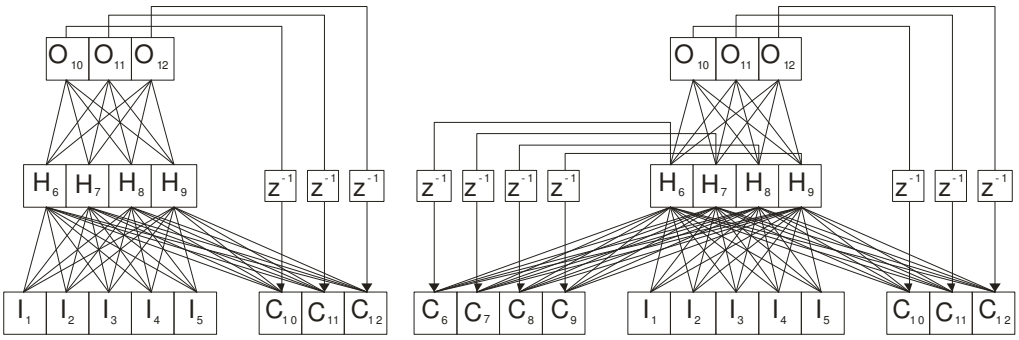
Viaceré architektúry rekurentných neurónových sietí sú popísané v literatúre. Najznámejšia a najčastejšie používaná Elmanova jednoduchá rekurentná neurónová sieť už bola opísaná v predchádzajúcej časti. Pre porovnanie s ostatnými architektúrami je na obr. 5 znázornená Elmanova sieť zložená z 5 vstupných, 4 skrytých (rekurentných) a 3 výstupných neurónov.



Obr. 5. Elmanova jednoduchá rekurentná neurónová sieť zložená z 5 vstupných, 4 skrytých (rekurentných) a 3 výstupných neurónov. Neprerušované čiary označujú vstupné, výstupné a rekurentné váhové prepojenia a prerušované čiary s oneskorovacím členom z^{-1} označujú kopírovanie hodnôt z predchádzajúceho časového kroku.

T_0 je špeciálny neurón s konštantnou jednotkovou aktivitou, ktorý je pomocou váh plniacich funkciu prahov prepojený s každým skrytým a výstupným neurónom.

Na obr. 6a. je znázornená tzv. Jordanova architektúra [13]. Na rozdiel od Elmanovej architektúry sú cez rekurentné prepojenia skrytej vrstvy šírené aktivity výstupnej vrstvy, a teda aktivity na výstupných neurónoch O_{10} až O_{12} prostredníctvom kontextovej vrstvy C_{10} až C_{12} tvoria pamäť neurónovej siete.

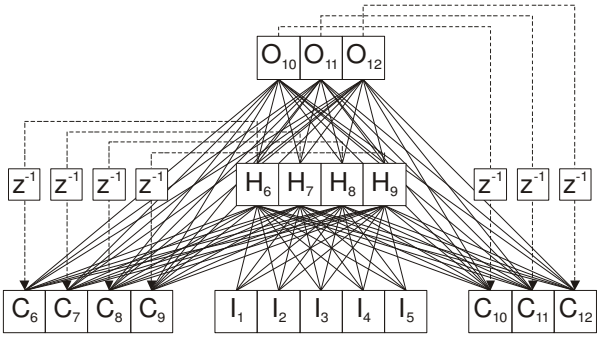


Obr. 6. (a) Jordanova architektúra a (b) architektúra podľa Bengia.

Táto tzv. výstupná rekurencia umožňuje pri trénovaní siete realizovať metódu „vnúteného učiaceho signálu“ (ang. Teacher Forcing). Táto metóda urýchľuje trénovanie tým, že v jeho priebehu sa na kontextovú vrstvu nekopírujú aktivity z výstupnej vrstvy, ale sa tam priamo umiestňujú požadované ideálne hodnoty. Vo všeobecnosti sa ale usudzuje, že rekurentné váhy na skrytej vrstve sú oveľa významnejšie než rekurentné váhy z výstupnej vrstvy.

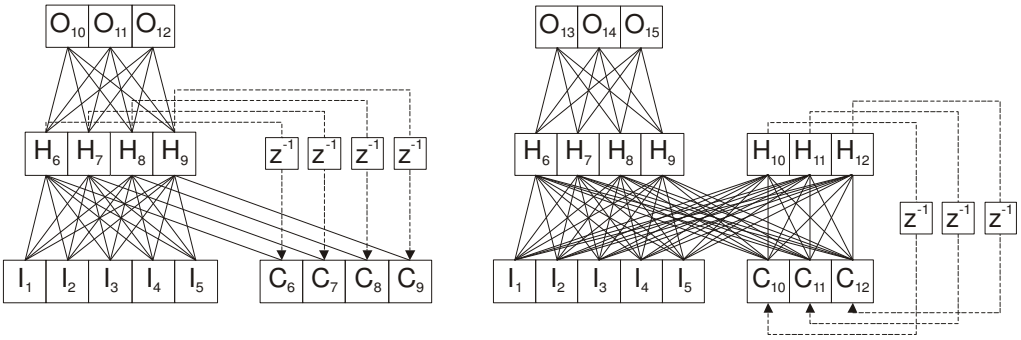
Na obr. 6b. je znázornená architektúra podľa Bengia. Obsahuje ako skrytú, tak aj výstupnú rekurenciu, čiže kontextová vrstva je tvorená aj aktivitami zo skrytých neurónov (C6 až C9) a aj z výstupných neurónov (C10 až C12).

Plne prepojená rekurentná neurónová sieť Williamsa a Zipsera [37] je znázornená na obr. **7**. Na rozdiel od Bengiovej architektúry prepojenia z kontextovej vrstvy smerujú aj do výstupnej vrstvy. Inak povedané, každý neurón (okrem vstupných neurónov) je prepojený cez rekurentnú váhu na každý neurón.



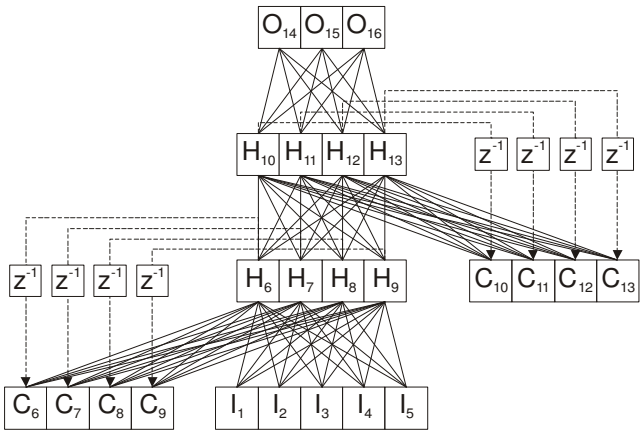
Obr. 7. Plne prepojená rekurentná neurónová sieť Williamsa a Zipsera.

Architektúra študovaná Frasconim, Gorim a Sodom [9] (obr. 8a.) sa od Elmanovej architektúry odlišuje tým, že obsahuje iba lokálne rekurentné prepojenia. Skrytý neurón je prostredníctvom rekurentnej váhy spojený iba so sebou samým, čiže má informáciu iba o svojej predchádzajúcej aktivite. Takáto lokálna rekurencia sa považuje za nedostatočnú pre mnohé úlohy.



Obr. 8. (a) Architektúra podľa Frasconiho, Goriho a Sodu. a (b) architektúra podľa Tiňa.

Zaujímavá je Tiňova architektúra [30] zobrazená na obr. 8b. Vyznačuje sa úplne oddelenou stavovou a asociačnou časťou siete. Asociačná časť siete je tvorená prvou skrytou vrstvou (neuróny H6 až H9) a výstupnou vrstvou. Stavová časť siete zodpovedajúca za vytvorenie a udržiavanie kontextovej informácie je tvorená druhou skrytou vrstvou (neuróny H10 až H12), ktorá je prostredníctvom rekurentných váh prepojená sama na seba (na obr. znázornené cez kontextovú vrstvu).



Obr. 9. Atrchitektúra 5-4R-4R-3L.

Posledná prezentovaná architektúra označená podľa počtu neurónov na jednotlivých vrstvách je charakteristická dvoma skrytými vrstvami prepojenými každá sama na seba prostredníctvom rekurentných prepojení. Označenie 5-4R-4R-3L znamená, že vstupná vrstva obsahuje 5 vstupných neurónov, za ňou nasledujú dve rekurentné vrstvy každá so 4 neurónmi a nakoniec výstupná vrstva je tvorená tromi neurónmi s lineárnou aktivačnou funkciou. Tento typ architektúry bol úspešne použitý pri riešení rôznych úloh z praxe [8][25].

4 Gradientové trénovacie algoritmy

4.1 Dopredné šírenie signálu

Dopredné šírenie signálu je v rekurentných neurónových sieťach realizované rovnakým spôsobom ako v prípade dopredných neurónových sietí. V rámci aktuálneho časového kroku na vstupné neuróny umiestnime aktivity zodpovedajúce aktuálnemu vstupu a postupne podľa vzťahu pre výpočet aktivity perceptrónu vypočítame aktivity neurónov. Aktivita neurónu je ovplyvňovaná cez jeho dopredné váhové prepojenie už vypočítanými aktivitami so súčasného časového kroku a cez rekurentné váhové prepojenia už vypočítanými aktivitami z predchádzdajúcich časových krokov.

Formalizmus použitý na zápis algoritmov v tejto kapitole už bol predstavený pri algoritmoch dopredného šírenia a spätného šírenia chyby v doprednej sieti. Aj v rekurentnej neurónovej sieti s N neurónmi (vstupné, skryté a výstupné) je nevyhnutné usporiadať neuróny tak, aby neurón s indexom i bol cez dopredné váhové prepojenia spojený iba s neurónom j s nižším indexom $j < i$. Cez rekurentné prepojenia môže byť spojený aj s neurónmi s vyšším indexom. Opäť indexom 0 označíme špeciálny vstupný neurón vždy nastavený na konštantu 1 a nasledovať ho budú vstupné, skryté a nakoniec výstupné neuróny. Týmto spôsobom zabezpečíme, aby sme mohli určiť aktivity postupne od neurónu s indexom 0 až po $N-1$.

Taktiež váhové prepojenia je potrebné označiť indexmi 0 až $M-1$, pričom M je počet všetkých váhových prepojení. V prípade rekurentných neurónových sietí je váhové prepojenie usporiadaná štvorica (i, j, t, v), kde t je časové oneskorenie váhového prepojenia a určuje, či je prepojenie dopredné ($t = 0$) alebo rekurentné ($t > 0$). Všetky architektúry uvedené v tretej kapitole obsahujú iba rekurentné prepojenia s $t = 1$, čiže také, cez ktoré sa šíri aktivita iba z bezprostredne predchádzajúceho časového kroku. Architektúry s väčšími časovými oneskoreniami nie sú bežne používané. Ostatné zložky určujúce váhovéhe prepojenie sú rovnaké ako v dopredných sieťach: i je index cieľového neurónu, j je index zdrojového neurónu a intenzita váhového prepojenia je v štvorici označená symbolom v. Na sprístupnenie jednotlivých položiek usporiadanej štvorici reprezentujúcej váhové prepojenie opäť použijeme operátory. Nech (i, j, t, v) je váhové prepojenie s indexom l, tak potom $d_l = i$, $s_l = j$, $\tau_l = t$ a $w_l = v$. Množinu indexov všetkých váhových prepojení označíme W.

Použime teraz navrhnutý formalizmus na zápis dopredného šírenia signálu v ľubovoľnej rekurentnej neurónovej sieti. Štruktúra siete (architektúra) je algoritmu sprístupnená prostredníctvom operátorov d, s a τ. Pri určovaní aktivít postupujeme od neurónov s nižším indexom k neurónom s vyšším indexom. Najskôr špeciálny neurón s indexom 0 je nastavený na konštantu 1. Na nasledujúce vstupné neuróny je umiestnený aktuálny vstup siete. Ostatné aktivity $x_i(t)$ v časovom kroku t sú vypočítané podľa vzťahu:

$$\tilde{x}_i(t) = \sum_{\forall j \in W \mid d_j = i} w_j x_{s_j}(t - \tau_j), \tag{23}$$

$$x_i(t) = f(\tilde{x}_i(t)). \tag{24}$$

Zápis $j \in W \mid \mathrm{d}_\mathrm{j} = i$ je množina indexov takých váhových prepojení, ktorých cieľový neurón je neurón s indexom i. Sú to vlastne všetky váhové prepojenia patriace neurónu i. Cez váhové prepojenie w_j sa šíri aktivita x_{s_j} zdrojového neurónu s indexom s_j. Časové oneskorenia τ_j určuje časový krok do minulosti, ktorého aktivita je cez váhu šírená. Napr. ak je váhové prepojenie j dopredné, tak jeho $\tau_j = 0$ a použije sa vypočítaná aktivita $x_{s_j}(t)$ - aktivita zo súčasného kroku t. Ak je váhové prepojenie j rekurentné s časovým oneskorením 1, tak $\tau_j = 1$ a použije sa vypočítaná aktivita $x_{s_j}(t-1)$ - aktivita z predchádzajúceho časového kroku $t-1$. Aktivačná funkcia f tiež môže závisieť od neurónu, avšak uvedený zápis uvažuje rovnakú aktivačnú funkciu pre všetky neuróny v sieti.

4.2 Algoritmus spätného šírenia chyby

Elmanova jednoduchá rekurentná neurónová sieť bola v [7] trénovaná jednoduchým algoritmom spätného šírenia chyby, ktorý poznáme z kapitoly venujúcej sa dopredným neurónovým sieťam. Tento algoritmus nie je vo všeobecnosti vhodný na trénovanie rekurentných sietí, pretože nezohľadňuje spätné šírenie chybového signálu cez rekurentné prepojenia do predchádzajúcich časových krokov. Môže byť však úspešne použitý na jednoduchšie úlohy alebo ako referenčná metóda pre iné prístupy.

Pomocou formalizmu zavedeného pri popise dopredného šírenia uveďme vzťahy pre algoritmus spätného šírenia chyby upraveného pre rekurentné neurónové siete. Chybový signál – derivácie $\delta_i(t) = \partial E(t) / \tilde{x}_i(t)$ sú vypočítavané spätne, čiže od neurónov s vyšším indexom smerom k neurónom s nižším indexom:

$$\delta_i(t) = \left[[d_i(t) - x_i(t)] + \sum_{\substack{\forall j \in W \mid \\ \mid \mathrm{s_j} = i \wedge \\ \wedge \tau_j = 0}} w_j \delta_{\mathrm{d_j}}(t) \right] f'(\tilde{x}_i(t)). \tag{25}$$

Chyba $d_i(t) - x_i(t)$ vzniká iba na výstupných neurónoch, a iba keď je definovaný želaný výstup $d_i(t)$. V opačnom prípade $d_i(t) = x_i(t)$ a teda člen $d_i(t) - x_i(t) = 0$. Pri výpočte chybového signálu $\delta_i(t)$ sa cez všetky dopredné váhové prepojenia vychádzajúce z neurónu i prešíria už vypočítane chybové signály. Indexy dopredných váhových prepojení vychádzajúcich z neurónu i sú určené zápisom $\forall j \in W \mid (\mathrm{s_j} = i) \wedge (\tau_j = 0)$: sú to také prepojenia indexované pomocou j, ktorých zdrojový neurón je i ($\mathrm{s_j} = i$) a sú dopredné ($\tau_j = 0$). Po vypočítaní chybových signálov je možné zmeniť váhy podľa:

$$\Delta w_{\mathrm{i}}(t) = \alpha \delta_{\mathrm{d_i}}(t) x_{\mathrm{s_i}}(t - \tau_i) + \beta \Delta w_i(t-1), \tag{26}$$

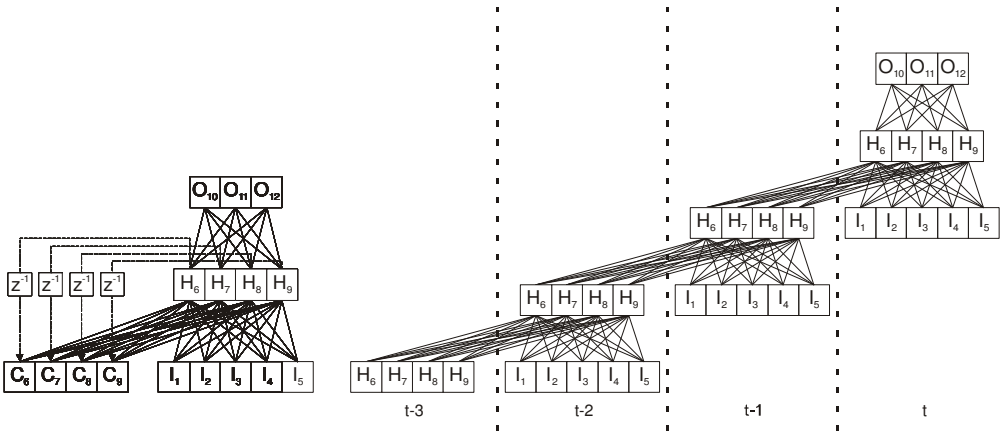
$$w_i(t+1) = w_i(t) + \Delta w_i(t), \tag{27}$$

kde α je rýchlosť učenia a β je hodnota určujúca momentum.

4.3 Algoritmus spätného šírenia chyby v čase

Rekurentné prepojenia významným spôsobom menia spôsob činnosti neurónových sietí. Aktivita neurónov sa cez rekurentné prepojenia šíri aj do nasledujúcich časových krokov a zmena váhových prepojení musí tento fakt zohľadniť. Klasické spätné šírenie mení váhy iba na základe hodnôt z aktuálneho časového kroku, aj keď chyba, ktorá vznikla v aktuálnom časovom kroku je výsledkom aktivít z predchádzajúcich časových krokov, ktoré ju prostredníctvom rekurentných váhových prepojení ovplyvnili. Riešením je šíriť chybu spätne nielen cez dopredné váhové prepojenia, ale aj cez rekurentné váhové prepojenia.

Algoritmus spätného šírenia chyby v čase (angl. Backpropagation Through Time, BPTT) umožňuje presný výpočet derivácií pre rekurentné neurónové siete. Princíp spočíva v rozvinutí rekurentnej siete v čase do potenciálne mnohovrstvovej doprednej siete a v následnom použití klasického BP algoritmu. Pravdepodobne najprínosnejší príspevok týkajúci sa BPTT algoritmu je [34]. V praxi nie je rekurentná sieť rozvíjaná v čase až do času 0, ale iba istý počet krokov do minulosti označovaných ako veľkosť okna. Tomuto variantu BPTT algoritmu sa budeme venovať neskôr. Elmanova jednoduchá rekurentná neurónová sieť (obr. 10a) rozvinutá do doprednej neurónovej siete je znázornená na obr. 10b.



Obr. 10. (a) Elmanova jednoduchá rekurentná neurónová sieť. (b) Tá istá sieť rozvinutá v troch časových krokoch do doprednej neurónovej siete.

Uveďme teraz vzťahy pre algoritmus vychádzajúci z uvedenej myšlienky. Táto verzia algoritmu uvažuje rozvinutie rekurentnej neurónovej siete do doprednej neurónovej siete cez všetky časové kroky (pre všetky vstupy z časovej postupnosti od času $t=0$ až do času $t=T$) a následné spätné prešírenie chybového signálu a určenie zmien váh. Chybový signál $\delta_i(t)$ je výpočítavaný spätne v čase (od časového kroku $t=T$ po $t=0$) a pre každý časový krok od neurónov s vyšším indexom smerom k neurónom s nižším indexom:

$$\delta_i(t)=\left[\left[d_i(t)-x_i(t)\right]+\sum_{\substack{\forall j\in W\,|\\ \mathrm{s_j}=i}} w_j\delta_{\mathrm{d_j}}\left(t+\tau_j\right)\right]f'\left(\tilde{x}_i(t)\right). \tag{28}$$

Ak $d_i(t)$ nie je definované, uvažujeme $d_i(t)=x_i(t)$, čiže na neuróne i v čase t nevzniká chyba. Chybový signál $\delta_i(t)=0$ pre $t>T$. Po vypočítaní všetkých $\delta_i(t)$ je možné zmeniť váhy využijúc vzťah:

$$\Delta w_{\mathrm{i}} = \alpha \sum_{t=1}^{T} \delta_{\mathrm{d_i}}(t) x_{\mathrm{s_i}}(t-\tau_i), \tag{29}$$

$$w_i = w_i + \Delta w_i . \tag{30}$$

Tento variant algoritmu BPTT je obdobou tzv. dávkového trénovania (angl. Batch Learning) algoritmom BP, pri ktorom sa váhy menia až po ukončení trénovacej epochy, čiže po prezentácií všetkých vzorov z trénovacej množine. Zmeny váh sú dané súčtom čiastkových zmien zodpovedajúcich jednotlivým vzorom. Podobne, ako dávkové trénovanie algoritmom BP, ani tento variant algoritmu BPTT nie je vo všeobecnosti úspešný, a teda ani používaný.

Výnimku tvoria prípady, keď je trénovacia množina tvorená kratšími postupnosťami, pre ktoré sú definované želané výstupné hodnoty. Napríklad úloha gramatickej inferencie spočíva v hľadaní modelu jazyka na základe pozitívnych a negatívnych príkladov slov patriacich do jazyka. Trénovacia množina je tvorená reťazcami reprezentujúcimi slová, ktoré do jazyka patria (želaný výstup po prezentovaní celého slova je napr. 1) a z reťazcov, ktoré do jazyka nepatria (želaný výstup po prezentovaní slova je napr. 0). Väčšinou sú tieto slová krátke (jednotky až desiatky symbolov) a rekurentná sieť je rozvinutá do doprednej siete pre každé vstupné slovo, pričom T je dané dĺžkou tohto slova. Želaný výstup $d_i(t)$ je teda definovaný iba v čase T. Váhové prepojenia sú následne na základe rovníc (28) až (30) zmenené.

V prípade dlhých časových postupností sa používa skrátené BPTT (angl. Truncated BPTT). Tento variant algoritmu spočíva v rozvinutí rekurentnej siete do doprednej spätne od aktuálneho času $k = t$ do časov $k = t-1$, $k = t-2$ až $k = t-h$, kde h je veľkosť časového okna. Toto rozvinutie siete sa vykonáva v každom kroku t pri prechode postupnosťou od $t = 0$ až po T. Index t určuje aktuálny časový krok pri prechode postupnosťou a index k je použitý na určenie kroku v rámci siete rozvinutej v kroku t. Vo vytvorenej doprednej sieti je spätne prešírený chybový signál, ktorý ale vzniká iba v čase $k = t$, čiže želané výstupy $d_i(k)$ sú definované iba v čase $k = t$. Chybový signál $\delta_i(k) = 0$ pre $k > t$. Vzťahy na výpočet chybového signálu a zmenu váh sú nasledovné:

$$\delta_i(k) = \left[[d_i(k) - x_i(k)] + \sum_{\substack{\forall j \in W | \\ \mathrm{s_j} = i}} w_j \delta_{\mathrm{d_j}}(k + \tau_j) \right] f'(\tilde{x}_i(k)), \tag{31}$$

$$\Delta w_{\mathrm{i}}(t) = \alpha \sum_{k=t-h}^{t} \delta_{\mathrm{d_i}}(k) x_{\mathrm{s_i}}(k-\tau_i), \tag{32}$$

$$w_i(t+1) = w_i(t) + \Delta w_i(t). \tag{33}$$

Veľkosť časového okna h sa volí 10 až 40. Z dôvodu existencie problému strácajúceho sa gradientu (angl. Vanishing Gradient Problem) [2] nemá zmysel spätne šíriť chybový signál cez viac krokov do minulosti.

4.4 Rekurentné učenie v reálnom čase

Rekurentné učenie v reálnom čase (angl. Real Time Recurrent Learning, RTRL) [37] bolo navrhnuté na trénovanie rekurentných neurónových sietí pri úlohách vyžadujúcich priebežnú zmeny váhových prepojení v reálnom čase. Neskôr navrhnutý a v predchádzajúcej kapitole opísaný variant algoritmu BPTT – skrátené BPTT tiež umožňuje trénovanie váhových prepojení v reálnom čase, a to dokonca s menšími nárokmi na výpočtový čas.

RTRL je podobne ako BP a BPTT založené na gradienovej minimalizácií metódou najprudšieho spádu. Podobne ako BPTT zohľadňuje existenciu rekurentných prepojení. Rozdiel oproti algoritmu BPTT je v spôsobe výpočtu derivácií určujúcich zmeny váh. Derivácie $\partial E/\partial w_j$ sú v RTRL vypočítané na základe derivácií $\partial x_i/\partial w_j$, ktoré sú vypočítavané „dopredným spôsobom“. Algoritmy BP a BPTT sú založené na spätnom šírení derivácií $\partial E/\partial \tilde{x}_i$, ktoré následne slúžia pre výpočet $\partial E/\partial w_j$. Formálne zapíšeme algoritmus RTRL pomocou nasledujúcich vzťahov:

$$\frac{\partial x_i}{\partial w_j}(t) = \left[\delta_{id_j} x_{s_j}(t) + \sum_{\forall k \in W \mid d_k = i} w_k \frac{\partial x_{s_k}}{\partial w_j}(t - \tau_k) \right] f'(\tilde{x}_i(t)), \tag{34}$$

$$\Delta w_j(t) = \alpha \sum_{\forall i \in O} [d_i(t) - x_i(t)] \frac{\partial x_i}{\partial w_j}(t) + \beta \Delta w_j(t-1), \tag{35}$$

$$w_i(t+1) = w_i(t) + \Delta w_i(t). \tag{36}$$

Derivácie $\partial x_i/\partial w_j$ sú vypočítavané postupne a dopredným spôsobom na základe už vypočítaných derivácií. Kroneckerova delta δ_{ij} je 1, ak $i = j$, inak $\delta_{ij} = 0$. Čiže člen $\delta_{id_j} x_{s_j}(t)$ je vyhodnotený ako $x_{s_j}(t)$, čiže ako aktivita na zdrojovom neuróne váhy j iba vtedy, ak váha w_j v derivácií $\partial x_i/\partial w_j(t)$ prislúcha neurónu x_i (vtedy $d_j = i$ a $\delta_{id_j} = 1$). Množinu indexov všetkých výstupných neurónov (neurónov, na ktorých môže vzniknúť chyba) sme označili symbolom O. Ak želané výstupné aktivity nie sú definované, uvažujeme $d_i(t) = x_i(t)$, a teda člen $d_i(t) - x_i(t) = 0$.

5 Trénovacie algoritmy využívajúce Kalmanovu filtráciu

Medzi najúspešnejšie prístupy používané na trénovanie neurónových sietí patria metódy založené na Kalmanovej filtrácií [23][32]. Štandardný Kalmanov filter (KF)[15] je možné použiť v lineárnom systéme na optimálny odhad skrytého (nepozorovaného) stavu na základe pozorovaní. V prípade nelineárneho systému je možné použiť tzv. rozšírený Kalmanov filter (Extended Kalman filter, EKF), ktorý pozostáva z linearizácie nelineárnych funkcií v každom kroku a následnom použití štandardných vzťahov Kalmanovej filtrácie.

V prípade neurónovej siete je odhadovaný stavový vektor stotožnený s trénovanými váhami siete. Na výpočet derivácií pre EKF je možné použiť algoritmus podobný BPTT (EKF-BPTT algoritmus). Hlavnou nevýhodou EKF je jeho vysoká výpočtová náročnosť. Boli navrhnuté viaceré aproximačné prístupy redukujúce náročnosť výpočtu. V tomto texte predstavíme pravdepodobne najpoužívanejšiu modifikáciu znižujúcu výpočtovú náročnosť, a to tzv. oddelený Kalmanov filter. Taktiež sa budeme venovať viacprúdovému Kalmanovmu filtru, ktorý umožňuje dosiahnuť robustné a stabilné riešenie v mnohých praktických aplikáciách.

5.1 Kalmanov filter

Kalmanov filter je súbor vzťahov popisujúcich rekurzívne riešenie diskrétneho lineárneho problému filtrovania. Je to efektívne riešenie problému odhadu neznámeho stavu ovplyvneného šumom metódou najmenších štvorcov [21].

Predpokladajme nasledujúci systém riadený podľa nasledujúcich vzťahov. Prvá lineárna stochastická diferenčná rovnica sa nazýva stavová rovnica, a popisuje zmenu stavu v priebehu diskrétnych časových krokoch:

$$\mathbf{x}(t) = \mathbf{F}(t)\mathbf{x}(t-1) + \mathbf{w}(t). \tag{37}$$

Stav systému $\mathbf{x}(t)$ je lineárnou transformáciou predchádzajúceho stavu, je určený aplikovaním transformácie reprezentovanou známou prechodovou maticou $\mathbf{F}(t)$ na predchádzajúci stav systému $\mathbf{x}(t-1)$ a výsledok je zaťažený bielym gausovským šumom $\mathbf{w}(t)$. Druhá lineárna rovnica sa nazýva rovnica pozorovania a popisuje, ako sú na základe stavu systému vypočítané pozorovania:

$$z(t) = \mathbf{H}(t)\mathbf{x}(t) + \mathbf{v}(t). \tag{38}$$

Získané pozorovanie alebo výstup systému $z(t)$ je určený aplikovaním známej výstupnej matice $\mathbf{H}(t)$ na stav systému. Výsledok je opäť zaťažený bielym gausovským šumom $\mathbf{v}(t)$. Matice $\mathbf{F}(t)$ a $\mathbf{H}(t)$ sa v čase môžu meniť. Taktiež poznáme parametre prechodového $\mathbf{w}(t)$ a výstupného šumu $\mathbf{v}(t)$, ich kovariančné matice $\mathbf{Q}(t)$ a $\mathbf{R}(t)$ majú všetky hodnoty okrem diagonálnych nulové:

$$\mathbf{Q}(t) = E\left[\mathbf{w}(t)\mathbf{w}(t)^T\right], \tag{39}$$

$$\mathbf{R}(t) = E\left[\mathbf{v}(t)\mathbf{v}(t)^T\right]. \tag{40}$$

Šumy $\mathbf{w}$ a $\mathbf{v}$ sú nekorelované s nulovou strednou hodnotou. Cieľom Kalmanovej filtrácie je vyfiltrovať čo najpresnejší odhad $\hat{\mathbf{x}}(t)$ stavu systému $\mathbf{x}(t)$. Zadefinujme si chybu odhadu stavu $\mathbf{e}(t)$ a kovarianciu odhadu chyby $\mathbf{P}(t)$:

$$\mathbf{e}(t) = \mathbf{x}(t) - \hat{\mathbf{x}}(t), \tag{41}$$

$$\mathbf{P}(t) = E\left[\mathbf{e}(t)\mathbf{e}(t)^T\right]. \tag{42}$$

Kalmanov filter funguje v dvojkrokovom cykle. Prvý krok je úprava v novom časovom kroku a je to jednoduchá predikcia $\hat{\mathbf{x}}^{-}(t)$ z už odhadnutého stavu $\hat{\mathbf{x}}(t-1)$ a kovariancie chyby $\mathbf{P}^{-}(t)$ z existujúceho odhadu $\mathbf{P}(t-1)$. Získaný odhad, v ktorom ešte nie je zahrnuté výstupné pozorovanie, nazývame apriórny odhad a entity označujeme pomocou $^{-}$.

$$\hat{\mathbf{x}}^{-}(t) = \mathbf{F}(t)\hat{\mathbf{x}}^{-}(t-1), \tag{43}$$

$$\mathbf{P}^{-}(t) = \mathbf{F}(t)\mathbf{P}(t-1)\mathbf{F}(t)^{T} + \mathbf{Q}(t). \tag{44}$$

Druhý krok je úprava na základe pozorovania $\mathbf{z}(t)$ a vykonáva opravu odhadnutého apriórneho stavu $\hat{\mathbf{x}}^{-}(t)$ a kovariancie chyby $\mathbf{P}^{-}(t)$. Najskôr je vypočítaný tzv. Kalmanov zisk $\mathbf{K}(t)$:

$$\mathbf{K}(t) = \mathbf{P}(t)\mathbf{H}(t)^{T}\left[\mathbf{H}(t)\mathbf{P}(t)\mathbf{H}(t)^{T} + \mathbf{R}(t)\right]^{-1}, \tag{45}$$

a následne je možné vypočítať tzv. aposteriórny odhad stavu $\mathbf{x}(t)$ a $\mathbf{P}(t)$:

$$\mathbf{P}(t) = \mathbf{P}^{-}(t) - \mathbf{K}(t)\mathbf{H}(t)\mathbf{P}^{-}(t), \tag{46}$$

$$\hat{\mathbf{x}}(t) = \hat{\mathbf{x}}^{-}(t) - \mathbf{K}(t)\left[\mathbf{z}(t) - \mathbf{H}(t)\hat{\mathbf{x}}^{-}(t)\right]. \tag{47}$$

Rovnice (43) až (47) popisujú jeden cyklus Kalmanovho filtra, v ktorom sa spresňuje odhad stavu na základe jediného pozorovania. Po príchode ďalšieho pozorovania v ďalšom časovom kroku $\mathbf{z}(t+1)$ sa vykoná ďalší cyklus Kalmanovej filtrácie.

5.2 Rozšírený Kalmanov filter

Aplikovanie KF na nelineárne problémy môže byť vykonané niekoľkými spôsobmi. Pravdepodobne najpriamočiarejšie je aplikovanie KF na systém aproximovaný v každom kroku pomocou Taylorovho rozvoja. Taylorov rozvoj nám umožňuje aproximovať zodpovedajúco krát derivovateľnú funkciu f v okolí bodu a pomocou polynómu:

$$f(x) \approx f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \ldots \tag{48}$$

Tento spôsob aplikovania KF na nelineárne systémy sa nazýva rozšírený Kalmanov filter (angl. Extended Kalman filter, EKF).

Uvažujme nasledujúcu stavovú rovnicu a rovnicu pozorovania:

$$\mathbf{x}(t) = f(t, \mathbf{x}(t-1)) + \mathbf{w}(t), \tag{49}$$

$$\mathbf{z}(t) = h(t, \mathbf{x}(t)) + \mathbf{v}(t), \tag{50}$$

kde f a h sú nelineárne funkcie. Linearizáciou týchto rovníc je možné odvodiť nasledujúce rovnice úprav na základe času:

$$\hat{\mathbf{x}}^{-}(t) = f\left(t, \hat{\mathbf{x}}^{-}(t-1)\right), \tag{51}$$

$$\mathbf{P}^{-}(t) = \mathbf{F}(t)\mathbf{P}(t-1)\mathbf{F}(t)^{T} + \mathbf{Q}(t), \tag{52}$$

a rovnice úprav na základe pozorovania:

$$\mathbf{K}(t) = \mathbf{P}(t)\mathbf{H}(t)^{T}\left[\mathbf{H}(t)\mathbf{P}(t)\mathbf{H}(t)^{T} + \mathbf{R}(t)\right]^{-1}, \tag{53}$$

$$\mathbf{P}(t) = \mathbf{P}^{-}(t) - \mathbf{K}(t)\mathbf{H}(t)\mathbf{P}^{-}(t), \tag{54}$$

$$\hat{\mathbf{x}}(t) = \hat{\mathbf{x}}^{-}(t) - \mathbf{K}(t)\left[\mathbf{z}(t) - h\left(t, \hat{\mathbf{x}}^{-}(t)\right)\right], \tag{55}$$

kde $\mathbf{F}(t)$ a $\mathbf{H}(t)$ sú matice derivácií reprezentujúce prvý (lineárny) člen Taylorovho rozvoja:

$$\mathbf{F}(t) = \frac{\partial f\left(t, \hat{\mathbf{x}}(t)\right)}{\partial \mathbf{x}}, \tag{56}$$

$$\mathbf{H}(t) = \frac{\partial h\left(t, \hat{\mathbf{x}}(t)\right)}{\partial \mathbf{x}}. \tag{57}$$

Z uvedeného vyplýva, že tento prístup je možné použiť iba pre derivovateľné funkcie f a h, čo je ale v prípade neurónovej siete splnené.

5.3 Prispôsobenie rozšíreného Kalmanovho filtra na trénovanie dopredných aj rekurentných neurónových sietí

Trénovanie viacvrstvovej neurónovej siete zloženej z perceptrónov môže byť transformované na problém optimálneho filtrovania [36]. Neurónová sieť môže byť popísaná ako nelineárny systém:

$$\mathbf{w}(t) = \mathbf{w}(t-1), \tag{58}$$

$$\mathbf{z}(t) = h\left(\mathbf{w}(t), \mathbf{u}(t)\right) + \mathbf{v}(t), \tag{59}$$

pričom neznámy stav systému $\mathbf{w}(t)$ je vektor hodnôt váhových prepojení neurónovej siete. Váhy v natrénovanej siete sa nemenia, a teda prechodová matica má tvar $\mathbf{F}(t) = \mathbf{I}$, kde $\mathbf{I}$ je matica identity. Pozorovanie $\mathbf{z}(t)$ sú želané výstupné hodnoty siete a funkcia pozorovania $h()$ je nelineárna funkcia. Jej argumentmi sú váhy siete $\mathbf{w}(t)$ a vstupy $\mathbf{u}(t)$. Matica $\mathbf{H}(t)$ je vypočítaná v každom časovom kroku t ako:

$$\mathbf{H}(t) = \frac{\partial h\left(\hat{\mathbf{w}}(t), \mathbf{u}(t)\right)}{\partial \mathbf{x}}, \tag{60}$$

kde $\hat{\mathbf{w}}(t)$ je odhad vektora hodnôt váhových prepojení neurónovej siete, čiže aktuálne váhy siete v priebehu procesu trénovania. Vektor $\mathbf{w}(t)$ sú ideálne váhy siete, ktoré sa snažíme Kalmanovou filtráciou nájsť.

Sformulujme teda nakoniec rovnice rozšíreného Kalmanovho filtra aplikovaného na trénovanie neurónových sietí [4]. Vo vzťahoch už použijeme symboly $\mathbf{w}$, $\mathbf{d}$ a $\mathbf{o}$, ktoré sme používali na označenie intenzít váhových prepojení, želaných výstupov a skutočných výstupov neurónovej siete. Jeden krok trénovania neurónovej siete je opísaný nasledujúcimi vzťahmi:

$$\mathbf{K}(t) = \mathbf{P}(t-1)\mathbf{H}(t)^T \left[\mathbf{H}(t)\mathbf{P}(t-1)\mathbf{H}(t)^T + \mathbf{R}(t)\right]^{-1}, \tag{61}$$

$$\mathbf{w}(t) = \mathbf{w}(t) - \mathbf{K}(t)\left[\mathbf{d}(t) - \mathbf{o}(t)\right], \tag{62}$$

$$\mathbf{P}(t) = \mathbf{P}(t-1) - \mathbf{K}(t)\mathbf{H}(t)\mathbf{P}(t-1) + \mathbf{Q}(t). \tag{63}$$

Počet všetkých váh v sieti bude označený ako n_w a počet výstupných neurónov ako n_o. Vektor $\mathbf{w}$ s dĺžkou n_w je vektor všetkých váh neurónovej siete (vstupných, výstupných, rekurentných, prahových, ...). Jakobián, matica $\mathbf{H}$ s rozmermi $n_o \times n_w$, je vypočítaný v každom časovom kroku. V riadkoch obsahuje derivácie zodpovedajúcej výstupnej aktivity k všetkým váham. Tieto derivácie môžu byť vypočítané skoro rovnakými rutinami ako sú algoritmy RTRL alebo BPTT. Matica $\mathbf{P}$ s rozmermi $n_w \times n_w$ je matica kovariancie chyby obsahujúca kovariancie zodpovedajúce všetkým možným párom váh. Kalmanov zisk $\mathbf{K}$, matica s rozmermi $n_w \times n_o$ slúži na úpravu váh $\mathbf{w}$ na základe rozdielu medzi želaným výstupom $\mathbf{d}$ a aktuálnym výstupom siete $\mathbf{o}$. Matica $\mathbf{R}$ s rozmermi $n_o \times n_o$ reprezentuje šum pozorovania a podobne ako rýchlosť učenia v gradientových algoritmov riadi rýchlosť konvergenice EKF. Vyššie hodnoty šumu zodpovedajú vyššej miere neurčitosti kladenej na rozdiel $\mathbf{d}(t) - \mathbf{o}(t)$, čo vedie k pomalšiemu učeniu. $\mathbf{Q}$ s rozmermi $n_w \times n_w$ je kovariančná matica prechodového šumu. Malé hodnoty prechodového šumu sú stále uvažované, nenulové hodnoty zlepšujú konvergenciu filtra.

Hlavnou nevýhodou tohto prístupu je jeho vysoká výpočtová náročnosť. Boli navrhnuté viaceré aproximačné prístupy redukujúce náročnosť výpočtu inverznej matice uvažovaním iba diagonálnej, resp. blokovo diagonálnej matice $\mathbf{P}$. Tento prístup je označovaný ako oddelený EKF.

5.4 Oddelený rozšírený Kalmanov filter

Princíp trénovania pomocou tzv. „oddeleného“ EKF (angl. Decoupled EKF) [26] pozostáva z rozdelenia váh do navzájom disjunktných skupín, pričom informácia o kovariancii chyby je udržiavaná iba medzi váhami v skupine. Najčastejšie sú v skupine váhové prepojenia prislúchajúce rovnakému cieľovému neurónu. Takýto prístup výrazne redukuje výpočtové nároky algoritmu. Ak má neurónová sieť N neurónov (neuvažujeme vstupné neuróny), derivácie výstupu siete voči váham

smerujúcich do neurónu i v čase t sú umiestené v matici $\mathbf{H}_i(t)$, $i \in \{1,2,...,N-1\}$. Matica kovariancie chyby pre váhy neurónu i je $\mathbf{P}_i(t)$ a matica kovariancie šumu je $\mathbf{R}_i(t)$. Rovnice oddeleného EKF sú:

$$\mathbf{A}(t) = \left[\sum_{i=1}^{N} \mathbf{H}_i(t) \mathbf{P}_i(t-1) \mathbf{H}_i(t)^T + \mathbf{R}_i(t) \right]^{-1}, \tag{64}$$

$$\mathbf{K}(t) = \mathbf{P}(t-1) \mathbf{H}(t)^T \mathbf{A}(t), \tag{65}$$

$$\mathbf{w}(t) = \mathbf{w}(t) - \mathbf{K}(t) [\mathbf{d}(t) - \mathbf{o}(t)], \tag{66}$$

$$\mathbf{P}(t) = \mathbf{P}(t-1) - \mathbf{K}(t) \mathbf{H}(t) \mathbf{P}(t-1) + \mathbf{Q}(t). \tag{67}$$

Hoci oddelený EKF umožňuje realizovať výpočet s menšou výpočtovou zložitosťou, klasický EKF s „plnou“ maticou $\mathbf{P}$ môže viesť k lepším riešeniam a pri súčasnej dostupnosti zariadení s vyšším výpočtovým výkonom môže byť pre väčšinu úloh vhodnejší.

5.5 Viacprúdový rozšírený Kalmanov filter

Významnou a úspešnou modifikáciou štandardného EKF je trénovanie rekurentných sietí na viacerých „trénovacích prúdoch“ – viacprúdový EKF (Multistream EKF, MS-EKF) [8]. MS-EKF bol úspešne použitý na trénovanie rekurentných sietí na zložitých reálnych problémoch z priemyselného prostredia [24]. Princíp pozostáva z trénovania viacerých verzií tej istej siete na rôznych častiach trénovacej postupnosti, pričom každá verzia zdieľa tie isté hodnoty váhových prepojení, ale udržiava si vlastný stav siete. Trénovacia množina je teda rozdelená do viacerých „prúdoch“ a viaceré vstupno-výstupné vzory sú zohľadnené v jednom časovom kroku. Klasické trénovanie pomocou EKF s jediným „prúdom“ môže byť vhodné na časovej postupnosti s homogénnou dynamikou. Na modelovanie heterogénnych postupností s dynamikou meniacou sa v čase je vhodnejšie použiť viacprúdový EKF.

Viacprúdovým EKF je tiež možné natrénovať sieť na spracovanie viacerých navzájom nezávislých úloh a natrénovaný model môže byť jednoducho aplikovaný na riešenie ľubovoľnej z nich. Bežná sekvenčná prezentácia takýchto postupností na sieť trénovanú jednoprúdovým EKF môže viesť k nežiaducemu efektu, keď sa model príliš špecializuje na aktuálne prezentovaný vzor a „zabúda“ už natrénované vlastnosti umožňujúce kvalitné spracovanie dávnejšie prezentovaných vzorov (angl. Recency Effect).

Pre trénovanie pomocou MS-EKF je trénovacia množina rozdelená na niekoľko častí, čo už v mnohých konkrétnych situáciách môže byť zabezpečené. Napríklad trénovacia množina môže pozostávať z podmnožín získaných za rôznych pracovných podmienok alebo v rôznych pracovných módoch sledovaného procesu. Pre trénovanie pomocou MS-EKF je teda potrebné vytvoriť N_s trénovacích množín rovnakej dĺžky, kde N_s je počet prúdov viacprúdového EKF. Každá z trénovacích množín je spracovávaná jednou inštanciou neurónovej siete a v každom časovom kroku je

spracovávaných N_s trénovacích vzorov. Pre každý prúd je propagácia dopredného signálu realizovaná rovnakým spôsobom ako v N_s nezávislých sieťach. Taktiež derivácie výstupov sietí voči váham sú vypočítané nezávisle a je na to možné použiť BPTT či RTRL algoritmus. Proces trénovania by mohol ďalej pokračovať N_s nezávislými EKF krokmi a vypočítaním celkových zmien váhových prepojení spriemerovaním jednotlivých čiastkových zmien. Avšak v prípade MS-EKF sú už vypočítané derivácie združené do jedinej globálnej matice $\mathbf{H}(t)$. Označme $\mathbf{H}_i(t)$ maticu derivácií výstupu voči váham pre neurónovú sieť prislúchajúcu prúdu i v čase t. Potom matica $\mathbf{H}(t)$ môže byť vyjadrená ako $\mathbf{H}(t) = \left[\mathbf{H}_1(t), \mathbf{H}_2(t), ..., \mathbf{H}_{N_s}(t)\right]$. Rovnakým spôsobom aj vektor želaných hodnôt $\mathbf{d}(t)$ a vektor vypočítaných výstupných hodnôt $\mathbf{o}(t)$ sú zostavené z čiastkových vektorov zodpovedajúcich jednotlivým prúdom: $\mathbf{d}(t)^T = \left[\mathbf{d}_1(t)^T, \mathbf{d}_2(t)^T, ..., \mathbf{d}_{N_s}(t)^T\right]$, $\mathbf{o}(t)^T = \left[\mathbf{o}_1(t)^T, \mathbf{o}_2(t)^T, ..., \mathbf{o}_{N_s}(t)^T\right]$. Okrem týchto zmien prebieha Kalmanova filtrácia klasickým už opísaným spôsobom (rovnice (61) až (63)). Len pripomeňme, že váhové prepojenia sú zdieľané všetkými inštanciami siete.

5.6 Kalmanove filtere so sigma bodmi a duálna Kalmanova filterácia

Vyššie spomenuté algoritmy patria medzi najmodernejšie prístupy a reprezentujú súčasnú úroveň vedeckého poznania v oblasti rekurentných neurónových sietí. Existuje množstvo variantov a modifikácií Kalmanovej filtrácie a mnohé sú študované v kontexte trénovania rekurentných neurnových sietí. V krátkosti spomeňme dva zaujímavé prístupy.

Kalmanove filtre so sigma bodmi (angl. Sigma Point Kalman Filter) [14][22] spočívajú v presnejšom šírení koviariancie chýb cez jednotlivé kroky Kalmanovej filtrácie. Namiesto linearizácie sú vhodne zvolené body transformované cez nelineárny systém a matica kovariancie chýb je v ďalšom kroku odhadnutá na základe týchto transformovaných bodov.

Ďalším zaujímavým prístupom použiteľným na trénovanie rekurentných neurónových sietí je tzv. duálna Kalmanova filtrácia (angl. Dual EKF) [6][33]. Jej princíp spočíva v súbežnom odhadovaní stavu aj parametrov systému dvojicou Kalmanových filtrov. V kontexte trénovania neurónových sietí je jeden z filtrov používaný na adaptáciu váhových prepojení a druhý na odhad aktivít neurónov.

6 Dynamika rekurentných neurónových sietí

V tejto kapitole sa zameriame na dynamiku rekurentnej neurónovej siete, čiže jej správanie v čase. Špeciálne sa budeme venovať dynamike nenatrénovanej siete inicializovanej s malými váhami, pretože zaujímavé vlastnosti tejto dynamiky sú úspešne používané vo viacerých v súčasnosti populárnych prístupoch. Dynamické

správanie siete budeme pre jednoduchosť študovať na úlohách predikcie nasledujúceho symbolu.

6.1 Modelovanie postupností symbolov

Modelovanie postupností symbolov ma v oblasti rekurentných neurónových sietí dlhú históriu. Najznámejšia a najčastejšie používaná architektúra – Elmanova jednoduchá rekurentná neurónová sieť bola v [7] navrhnutá na študovanie jednoduchých lingvistických postupností. Práve v oblasti kognitívnej vedy sú rekurentné neurónové siete požívané na modelovanie jednoduchých aj zložitejších jazykových štruktúr [18].

V kontexte modelovania postupností symbolov sú rekurentné neurónové siete často používane na predikovanie nasledujúceho symbolu v postupnosti [3]. Úlohou siete je po prezentácií časti postupnosti vyhodnotiť, aké symboly môžu nasledovať. Pre názornosť si uveďme nasledujúci príklad.

Nech je daná gramatika $G = (T, N, P, A)$. Množina terminálnych symbolov $T = \{a, b, c, s\}$ je množina symbolov, z ktorých môžu byť vytvorené slová jazyka. Množina neterminálnych symbolov je $N = \{A, B, C\}$ a je to množina pomocných symbolov, ktoré môžu byť použité iba pri generovaní slov. Množina pravidiel P je tvorená nasledujúcimi jednoduchými pravidlami:

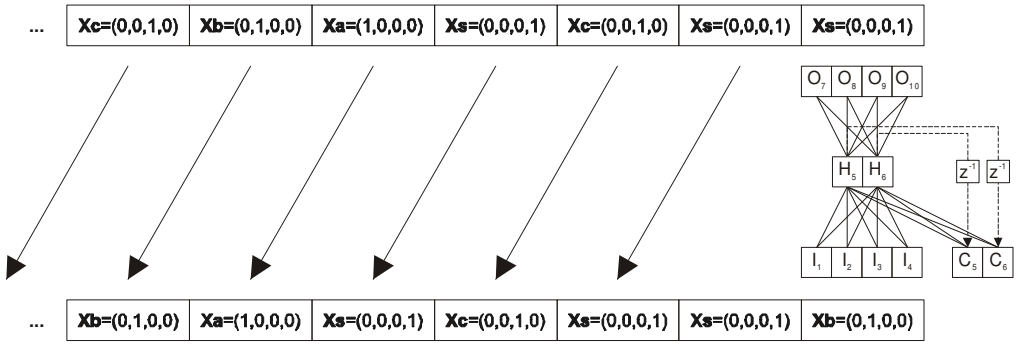
$$\begin{aligned} A &\to sA \mid bB \mid \varepsilon \\ B &\to sB \mid cC \\ C &\to sC \mid aA \end{aligned} \quad . \tag{68}$$

Štartovacím neterminálom je symbol A a ε je prázdne slovo. Táto gramatika generuje slová, v ktorých nasledujú za sebou symboly v poradí a, b a c, medzi ktorými môže byť na ľubovoľnom mieste a ľubovoľne mnohokrát symbol s. Príklady slov patriacich do jazyka generovaných touto gramatikou sú napríklad: bca, $sbcssas$, $sssbscasbcssa$ a pod. Postupnosť pre úlohu predikcie vytvoríme zreťazením veľkého počtu slov generovaných touto gramatikou takým spôsobom, že pravidlá, ktoré je možné pri generovaní v danom okamihu použiť, sú vyberané s rovnakou pravdepodobnosťou. Takýmto spôsobom bola vytvorená postupnosť začínajúca reťazcom *bsscsabcsssssabsssscsssa...*.

Takto vytvorená postupnosť je symbol po symbole prezentovaná rekurentnej neurónovej sieti, ktorej úlohou je predikovať nasledujúci symbol. Táto situácia je znázornená na obr. 11. Je zrejmé, že symboly nie je možné jednoznačne predikovať, pretože napr. po symbole b môže nasledovať c aj s, avšak nemôže nasledovať ani b a ani a.

Pre riešenie úlohy sme skonštruovali jednoduchú rekurentnú neurónovú sieť, zvolili sme 2 skryté neuróny. Tento počet nám na jednej strane umožní sledovať aktivity na skrytých neurónoch v klasickom dvojrozmernom grafe, a tiež vzhľadom na jednoduchosť úlohy je plne postačujúci na jej zvládnutie. Vstupy sú tvorené 4 symbolmi a, b, c, s a pre ich zakódovanie do vstupného vektora neurónovej siete bolo

zvolené kódovanie tzv. „jeden z viacerých“ (angl. One Hot Encoding). V tomto spôsobe kódovania každému vstupu prislúcha práve jeden vstupný neurón, ktorý je aktívny (nastavený na 1) práve vtedy, keď je sieti prezentovaný daný vstup. Inak je pasívny, teda nastavený na 0. Rovnaký spôsob kódovania sme zvolili aj na výstupe siete, a teda neurónová sieť mala 4 vstupné a 4 výstupné neuróny. Skryté aj výstupné neuróny boli samozrejme vybavené prahmi a použili sme sigmoidálnu aktivačnú funkciu pre skryté a aj výstupné neuróny.



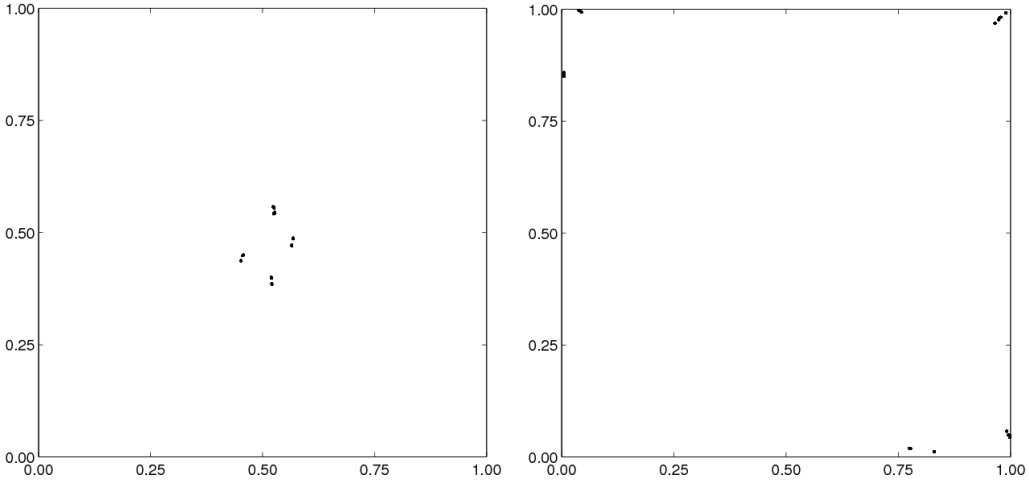
Obr. 11. Predikcia nasledujúceho symbolu riešená pomocou Elmanovej jednoduchej rekurentnej neurónovej siete.

Sieť bola inicializovaná malými váhami z intervalu $(-0.5,0.5)$ a následne bola trénovaná algoritmom BPTT s veľkosťou okna $h=10$, rýchlosťou učenia $\alpha=0.1$, momentum nebolo použité. Trénovacia postupnosť bola vytvorená vyššie uvedeným spôsobom a obsahovala 1000 symbolov. Postupnosť bola sieti prezentovaná 10 krát, a pred každým cyklom boli vynulované aktivity na skrytých neurónoch (kontextová vrstva bola inicializovaná na nulové hodnoty). Po natrénovaní bolo pozorované, že sieť je schopná s vysokou presnosťou predpovedať nasledujúce symboly. V každom kroku postupnosť môže pokračovať jedným z dvoch symbolov. Natrénovaná sieť na výstupných neurónoch zodpovedajúcich symbolom, ktorými postupnosť mohla pokračovať, správne umiestňovala aktivitu 0.5, čo môže byť považované za pravdepodobnosť pokračovania postupnosti príslušným symbolom.

Aby sme zistili, akým spôsobom rekurentná sieť danú úlohu rieši, je potrebné analyzovať štruktúru jej stavového priestoru. Pod stavom siete budeme rozumieť aktivity na skrytých neurónoch. Na správne riešenie úlohy si totiž sieť musí vytvárať také aktivity na skrytých neurónoch, aby bola na jednej strane schopná produkovať želaný výstup v aktuálnom časovom kroku, a na druhej strane si v nich tiež musí zapamätať informáciu o kontexte, aby správne vyprodukovala nasledujúce skryté aktivity (aj v závislosti od ďalšieho vstupu samozrejme).

Aktivity na skrytých neurónoch si v priebehu prechodu postupnosťou môžeme zaznamenať a tieto aktivity – stavy siete – môžeme umiestniť ako body do grafu. Osi grafu zodpovedajú aktivitám na dvoch skrytých neurónoch. Zobrazenie stavového priestoru sme spravili nielen pre natrénovanú sieť, ale aj pre nenatrénovanú náhodne inicializovanú sieť a výsledky sú zobrazené na obr. 12. V nasledujúcich častiach

analyzujeme najskôr dynamiku natrénovanej a potom aj nenatrénovanej rekurentnej neurónovej siete.



Obr. 12. Reprezentácia stavového priestoru nenatrénovanej (a) a natrénovanej (b) Elmanovej jednoduchej rekurentnej neurónovej siete s 2 skrytými neurónmi vytvorená zaznamenaním aktivít skrytých neurónov pri spracovaní postupnosti symbolov.

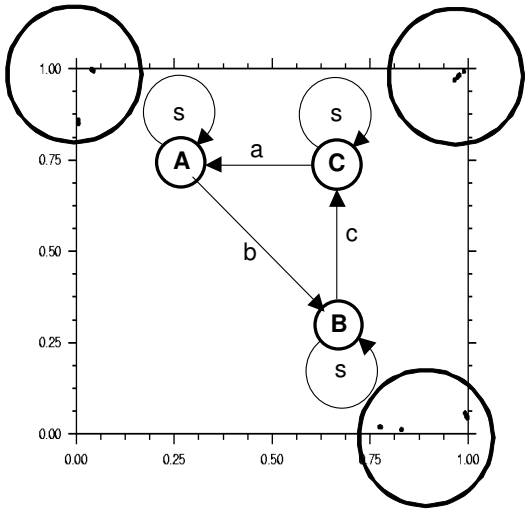
6.2 Dynamika natrénovanej rekurentnej neurónovej siete

Pozrime sa najskôr na dynamiku natrénovanej rekurentnej neurónovej siete. Z obr. 12b je zrejmé, že aktivity skrytej vrstvy sa koncentrujú v troch zhlukoch v rohoch priestoru $(0.0,1.0)\times(0.0,1.0)$. Po analýze prechodov medzi aktivitami skrytej vrstvy (reprezentovanými bodmi v grafe) v závislosti na vstupoch bolo možné stotožniť zhluky bodov so stavmi konečného automatu, ktorý si sieť trénovaním vytvorila.

Na obr. 13 je tento automat znázornený a je zrejmé, že zodpovedá gramatike, ktorou bola postupnosť symbolov vygenerovaná. Tri zhluky v rohoch grafu tvoria stavy automatu tak, ako je na obrázku naznačené. Ak je napr. stav siete v jednom z týchto zhlukov a sieti je prezentovaný symbol s, nasledujúci stav siete bude stále v tom istom zhluku. Ak je sieti prezentovaný jeden z troch symbolov a, b alebo c, stav siete, čiže aktivity skrytej vrstvy, budú v zhluku zodpovedajúcom príslušnému stavu automatu. Takýmto spôsobom si rekurentná sieť udržiava všetku potrebnú kontextovú informáciu. Vytvorila si model časovej postupnosti, ktorý jej umožňuje predpovedať nasledujúci symbol s vysokou presnosťou.

Tento príklad demonštruje schopnosť rekurentných neurónových sietí fungovať ako konečné stavové automaty, a tým modelovať regulárne jazyky. Stavová časť siete v takomto prípade slúži na zapamätanie si konečného počtu stavov. Rekurentné siete sú ale schopné modelovať aj zložitejšie jazykové štruktúry, ktoré si vyžadujú zložitejšiu pamäť. Takýmito sú napríklad bezkontextové jazyky, ktoré môžu byť modelované

zásobníkovými automatmi. Uveďme si príklad, ako je rekurentná neurónová sieť schopná si vytvoriť v svojej stavovej časti dynamické správanie modelujúce zásobník.



Obr. 13. Konečný stavový automat vytvorený v stavovej časti Elmanovej jednoduchej rekurentnej neurónovej siete

Uvažujme jednoduchú bezkontextovú gramatiku generujúcu známy jazyk $a^n b^n$. Formálne je gramatika definovaná takto: $G = (T, N, P, S)$, čiže množina terminálnych symbolov je $T = \{a, b\}$, množina neterminálnych symbolov je $N = \{S\}$ a množina pravidiel P je tvorená dvojicou pravidiel:

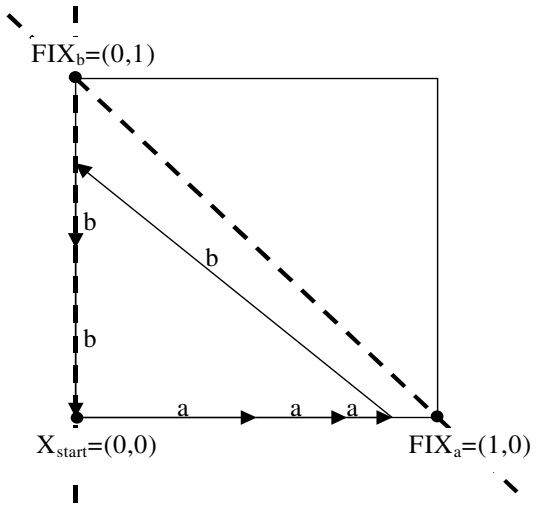
$$S \to aSb \mid \varepsilon . \tag{69}$$

Touto gramatikou je možne vygenerovať slová ab, $aabb$, $aaaaabbbbb$ a pod. Postupnosť pre úlohu predikcie vytvoríme rovnakým spôsobom ako v predchádzajúcom príklade a sieť opäť použijeme na riešenie úlohy predikcie. Je vhodné si uvedomiť, že je možné jednoznačne predikovať symboly až v druhej časti slova, po prvom symbole b, a síce je možné predikovať všetky ostatné symboly b, ktorých musí byť spolu práve toľko, koľko bolo symbolov a v prvej časti slova. Pre správne fungovanie je nevyhnutné vytvoriť si mechanizmus umožňujúci počítanie symbolov.

Na obrázku obr. 14 je znázornený princíp dynamického správania natrénovanej rekurentnej neurónovej siete [35]. Rekurentné a aj vstupné váhy boli v priebehu trénovania upravené do takých hodnôt, ktoré spôsobili vytvorenie bodového atraktora pre vstupný symbol a a sedlového bodu pre vstupný symbol b. Dynamika pre vstupné symboly a a b je daná vzťahmi:

$$\begin{aligned} \omega_a(x_1, x_2) &= (0.5x_1 + 0.5, 0) \\ \omega_b(x_1, x_2) &= (0{,}2x_1 + 2x_2 - 1)' \end{aligned} \tag{70}$$

kde x_1 a x_2 sú aktivity na skrytých neurónoch, ω_a je transformácia skrytých aktivít, ak je sieti prezentovaný symbol a a ω_b je transformácia stavu siete zodpovedajúca symbolu b. Transformácie sú v sieti vytvorené pomocou ideálne nastavených rekurentných a vstupných váhových prepojení a tiež pre jednoduchosť uvažujeme lineárnu aktivačnú funkciu.



Obr. 14. Zásobník vytvorený v stavovej časti rekurentnej neurónovej siete pomocou bodového atraktora a sedlového bodu.

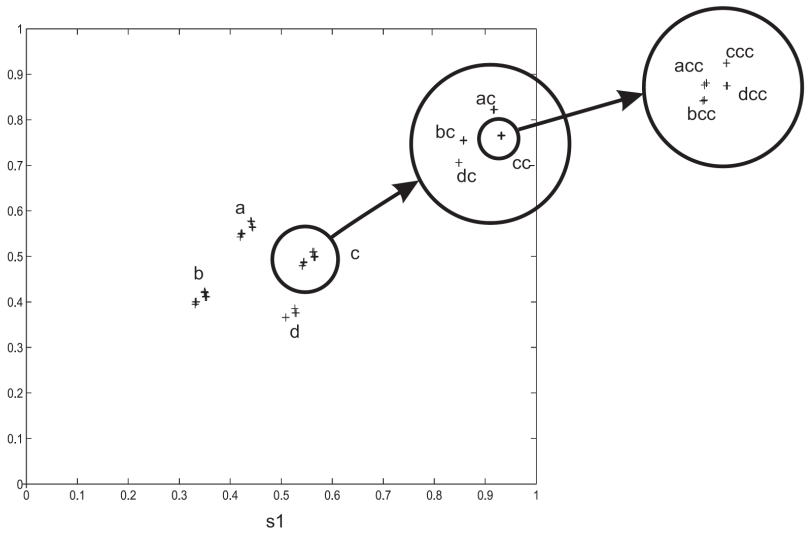
Transformácia ω_a umiestňuje stav siete smerom k bodu $(1.0,0.0)$, čo je bodový atraktor pre túto transformáciu. Nekonečne veľa zreťazení tejto transformácie, čiže nekonečne veľa symbolov a postupne prezentovaných sieti, by sieť uviedlo do stavu $(1.0,0.0)$. Čím viac symbolov a je prezentovaných sieti, tým bližšie je stav siete k bodovému atraktoru. Takto si sieť v aktivite x_1 pamätá, koľko symbolov a jej bolo prezentovaných. Transformácia ω_b vytvára sedlový bod $(0.0,1.0)$, ktorý „priťahuje“ stavy zo smeru $(1.0,0.0)$ a „odpudzuje“ stavy v smere $(0.0,0.0)$. Počet symbolov a prezentovaných v sieti sa pri prvej prezentácií symbolu b premietne do aktivity x_2 a následné prezentácie symbolu b ju znižujú rovnakou mierou, akou ju transformácia ω_a zvyšovala. Takto sieť presne vie, koľko symbolov b má nasledovať za symbolmi a. Sieť si vlastne takýmto spôsobom vytvorila počítadlo, transformácia ω_a počítadlo inkrementuje a ω_b dekrementuje, sieť simuluje jednoduchý počítadlový automat.

Vidíme, že rekurentné neurónové siete dokážu modelovať postupnosti s regulárnymi a bezkontextovými prvkami. V [1][27] bolo ukázané, akým spôsobom je rekurentná neurónová sieť schopná vo svojom stavovom priestore vytvárať aj pamäťové štruktúry nevyhnutné na spracovanie kontextových jazykov.

Uvedené príklady vo veľkej jednoduchosti približujú, akým spôsobom rekurentná neurónová sieť môže riešiť problémy. V reálnych rekurentných neurónových sieťach s väčším počtom skrytých neurónoch môže byť dynamika veľmi komplexná a jej analýza je zvyčajne náročná.

6.3 Dynamika nenatrénovanej rekurentnej neurónovej siete inicializovanej s malými váhami

V predchádzajúcej podkapitole sme naznačili, že v natrénovanej rekurentnej sieti je možné pozorovať dynamické správanie rôznej komplexnosti. Ale aj dynamika nenatrénovanej rekurentnej neurónovej siete inicializovanej s malými váhami má veľmi zaujímavé a potenciálne využiteľné vlastnosti [5][20][31].



Obr. 15. Organizácia stavového priestoru nenatrénovanej rekurentnej neurónovej siete inicializovanej malými váhami. Sieti je prezentovaná postupnosť vytvorená nad abecedou zloženou zo štyroch symbolov.

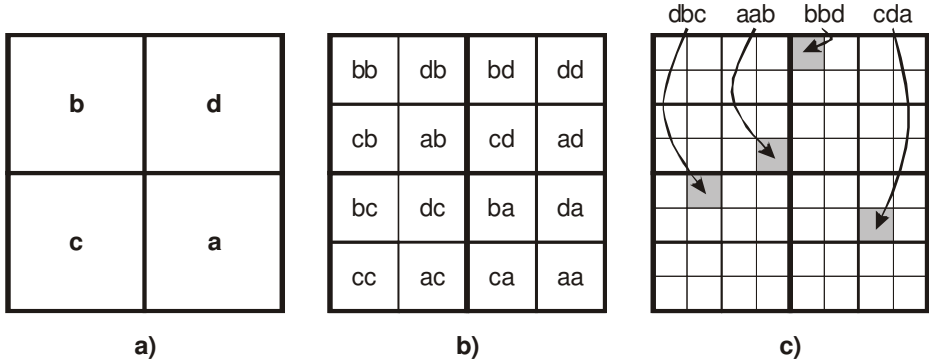
Pozrime sa bližšie na stavový priestor reprezentovaný grafom na obr. 12a. Hoci je sieť nenatrénovaná, jej stavový priestor vykazuje vysokú mieru štruktúrovanosti [16]. Stavy siete – aktivity skrytých neurónov sú organizované do zhlukov, pričom ich umiestnenie v stavovom priestore je dané postupnosťou prezentovanou sieti. Bližšie si vysvetlíme organizáciu stavového priestoru pomocou obr. 15. V stavovom priestore môžeme pozorovať 4 zhluky, pričom každý prislúcha práve jednému zo štyroch symbolov, z ktorých bola vytvorená vstupná postupnosť. Ak je rekurentnej sieti prezentovaný symbol, stav siete sa bude nachádzať v oblasti zodpovedajúcej tomuto symbolu, a teda zo stavu siete je možné jednoznačne určiť, ktorý symbol bol sieti prezentovaný. Ak sa ale bližšie pozrieme na zhluk zistíme, že má veľmi podobnú štruktúru ako celý stavový priestor, tiež sa skladá zo 4 podzhlukov. Ako je na obrázku

naznačené, pozícia v rámci tohto zhluku je určená predposledným symbolom prezentovaným sieti. Ak by sme sa bližšie pozreli na niektorý z podzhlukov, opäť by sme mohli pozorovať podobnú štruktúru.

Pre dôslednejšie pochopenie uvedenej vlastnosti použijeme už z predchádzajúcej časti známy formalizmus. Zmeny stavu siete budeme modelovať pomocou systému transformácií $\Omega = \{\omega_a, \omega_b, \omega_c, \omega_d\}$, kde jednotlivé transformácie reprezentujú zmenu stavu pri vstupnom symbole a až d:

$$\begin{aligned} &\Omega = \{\omega_x : X \to X, x \in \{a,b,c,d\}\} \\ &\omega_a(x_1, x_2) = (0.5x_1 + 0.5, 0.5x_2) \\ &\omega_b(x_1, x_2) = (0.5x_1, 0.5x_2 + 0.5) \\ &\omega_c(x_1, x_2) = (0.5x_1, 0.5x_2) \\ &\omega_d(x_1, x_2) = (0.5x_1 + 0.5, 0.5x_2 + 0.5) \end{aligned} \quad . \tag{71}$$

Kontraktívne transformácie ω_x pracujú nad priestorom $X = (0.0, 1.0)^2$, ktorý reprezentuje stavový priestor rekurentnej siete. Kontraktívnosť znamená, že transformácia zobrazí stavový priestor do jeho časti, formálne $\omega_x(X) \in X$. Ak takéto transformácie zreťazíme, výsledný stav bude odzrkadľovať poradie transformácií tak, ako je naznačené na obr. 16. Pozícia v stavovom priestore určuje, aké transformácie boli vykonané, resp. ktoré symboly boli sieti prezentované. Požiadavka na kontraktívnosť je pre rekurentné siete zabezpečená tým, že siete sú skoro vždy inicializované s malými váhami, ako aj tým skryté neuróny sú zvyčajne vybavené aktivačnou funkciou, ktorej obor hodnôt je ohraničený.



Obr. 16. Mapovanie stavového priestoru podľa vykonaných transformácií. a) Rozdelenie stavového priestoru podľa poslednej vykonanej transformácie, b) podľa posledných dvoch a c) posledných troch transformácií.

Teoreticky je takýmto spôsobom možné určiť všetky tranformácie/vstupy ľubovoľne ďaleko do minulosti, avšak v konkrétnej implementácií nás zvyčajne limituje presnosť počítačovej reprezentácie reálneho čísla. Avšak sieť si aj tak takýmto

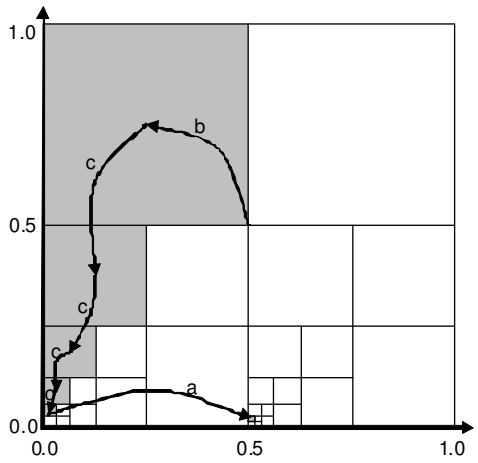
spôsobom pamätá aj desiatky symbolov do minulosti. Príklad vývoja stavu po prezentácií konkrétnej postupnosti je znázornený na obr. 17.

Schopnosť nenatrénovanej rekurentnej neurónovej siete mapovať vo svojom stavovom priestore históriu vstupov sme v [31] nazvali Markovovská architekturálna predispozícia (angl. Markovian Architectural Bias) rekurentných neurónových sietí, pretože je možné nájsť podobnosť medzi takouto organizáciou stavového priestoru a kontextami Markovovych modelov s premenlivou dĺžkou pamäti (angl. Variable Length Markov Models) [19][28].

Prezentované symboly: **b c c c c a**

Počiatočný stav: (0.5,0.5).

$\bar{x}_0 = (0.5,0.5)$
$\bar{x}_1 = \omega_b(\bar{x}_0) = (0.25,0.75)$
$\bar{x}_2 = \omega_c(\bar{x}_1) = (0.125,0.375)$
$\bar{x}_3 = \omega_c(\bar{x}_2) = (0.0625,0.1875)$
$\bar{x}_4 = \omega_c(\bar{x}_3) = (0.03125,0.09375)$
$\bar{x}_5 = \omega_c(\bar{x}_4) = (0.015625,0.046875)$
$\bar{x}_6 = \omega_a(\bar{x}_5) = (0.5078125,0.0234375)$



Obr. 17. Mapovanie symbolov v stavovom prietore siete. Postupne, ako sú symboly sieti prezentované, je stav siete transformovaný do zodpovedajúcich oblastí.

Takéto správanie nenatrénovaných rekurentných neurónových sietí inicializovaných malými váhami je úspešne využívané viacerými modelmi. Najznámejším sú v súčasnosti pomerne populárne siete s echo stavmi.

7 Siete s echo stavmi

Siete s echo stavmi (ang. Echo State Networks, ESNs) [11][12] predstavujú nový zaujímavý prístup v oblasti rekurentných neurónových sietí a v súčasnosti patria medzi modely, ktorých vlastnosti sú aktívne študované. ESN siete sú založené na vlastnostiach náhodne inicializovaných RNN sietí, ktoré sú schopné špecifickým spôsobom reprezentovať postupnosť vstupov vo svojej skrytej vrstve. ESN sieť je v podstate klasická RNN sieť Elmanovského typu obsahujúca skrytú vrstvu zloženú z veľkého počtu náhodne poprepájaných neurónov. Táto vrstva sa v ESN sieti nazýva dynamický rezervoár. Výstupné neuróny sú použité na extrakciu zaujímavých vlastností z dynamického rezervoára. Váhové prepojenia skrytých neurónov nie sú modifikované, trénovaniu podliehajú iba výstupné neuróny. Významným prínosom ESN architektúry je možnosť využiť výpočtovo nenáročný algoritmus lineárnej regresie na adaptáciu váhových prepojení výstupných neurónov.

7.1 Architektúra

Architektúra ESN siete je znázornená na obr. 18. ESN sieť sa skladá z klasických vstupných, skrytých a výstupných neurónov. Charakteristickou črtou ESN sietí je dynamický rezervoár. Obsahuje stovky až tisíce neurónov, pričom tieto sú zvyčajne rekurentnými váhami iba riedko poprepájané. Neuróny v dynamickom rezervoári sú klasické perceptróny so sigmoidálnou aktivačnou funkciou $f(x) = 1/(1+e^{-x})$, popr. sa používa funkcia $f(x) = \tanh(x)$, ktorá má rovnaký tvar ako sigmoida a interval reálnych čísel mapuje do otvoreného intervalu $(-1.0,1.0)$.

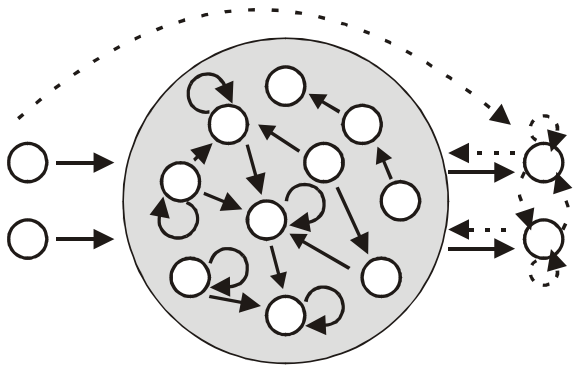
Opíšme formálne fungovanie ESN siete. Nech $\mathbf{u}(t)$ je vstupný vektor v časovom kroku t, aktivity na skrytých neurónov podliehajú nasledujúcemu vzťahu:

$$\mathbf{x}(t) = f\left(\mathbf{W}^{\text{IN}} \cdot \mathbf{u}(t) + \mathbf{W} \cdot \mathbf{x}(t-1) + \mathbf{W}^{\text{BACK}} \cdot \mathbf{y}(t-1)\right), \tag{72}$$

kde f je aktivačná funkcia skrytého neurónu, $\mathbf{W}$ je matica rekurentných váh, $\mathbf{W}^{\text{IN}}$ je matica vstupných váh a $\mathbf{W}^{\text{BACK}}$ je matica tzv. spätných váhových prepojení smerujúcich z výstupných neurónov na skryté neuróny. Aktivity na výstupných neurónoch je možné vypočítať podľa:

$$\mathbf{y}(t) = f\left(\mathbf{W}^{\text{OUT}}\left[\mathbf{u}(t)^T, \mathbf{x}(t)^T, \mathbf{y}(t-1)^T\right]^T\right), \tag{73}$$

kde $\mathbf{W}^{\text{OUT}}$ je matica váhových prepojení smerujúca zo vstupných, skrytých a vstupných neurónov na výstupné neuróny.



Obr. 18. Architektúra ESN siete. Prepojenia naznačené prerušovanými čiarami nemusia byť použité.

Úlohou dynamického rezervára je produkovať potenciálne zaujímavé správanie, ktoré je možné jednoduchým spôsobom pomocou výstupných váhových prepojení použiť na vytvorenie želaných výstupných aktivít. Aktivity na neurónoch rezervoára sú odozvou vstupov ovplyvňujúcich rezervoár prostredníctvom vstupných váhových prepojení. Prípadne aj výstupné aktivity môžu cez tzv. spätné váhové prepojenia ovplyvniť rezervoár podobným spôsobom ako vstupné aktivity.

Podstatnou podmienkou fungovania ESN siete je existencia tzv. „echo“ stavov v stavovom priestore siete. Od siete sa vyžaduje, aby jej stav bol „ozvenou“ vstupov prezentovaných sieti. Ak je táto podmienka splnená, na dosiahnutie vysokovýkonnej siete býva často postačujúca iba adaptácia výstupných váhových prepojení (samozrejme v závislosti na riešenej úlohe). Zvyčajne je potrebné skonštruovať sieť s mohutným a bohatým dynamickým rezervoárom, zloženým zo stoviek neurónov.

Vlastnosť existencie echo stavov znamená, že pre každý skrytý neurón x_i existuje tzv. echo funkcia e_i taká, že aktuálny stav $x_i(t)$ neurónu môže byť vyjadrený ako $x_i(t) = e_i(u(t), u(t-1), u(t-2), ...)$ [11]. ESN sieť využíva kontraktívnu dynamiku náhodne inicializovaného a netrénovaného dynamického rezervoára. Aktuálny vstup $u(t)$ prezentovaný sieti má väčší vplyv na stav siete ako vstup prezentovaný v skoršom časovom kroku. Vplyv vstupu postupne s časom a s prichádzajúcim ďalšími vstupmi slabne a rovnaká dostatočne dlhá postupnosť vstupov $u(t), u(t-1), u(t-2), ...$ uvedie sieť do rovnakého stavu bez ohľadu na iniciálny stav siete a predchádzajúce vstupy.

7.2 Trénovanie

Trénovanie ESN sietí spočíva v nastavení váhových prepojení smerujúcich do výstupných neurónov. Váhové prepojenia medzi vstupnými a skrytými neurónmi a rekurentné váhové prepojenia nepodliehajú trénovaniu. Pri trénovaní siete teda nie je potrebné šíriť chybovú informáciu cez rekurentné prepojenia, čo výrazne znižuje výpočtovú náročnosť trénovania. Napriek prípadnej nelinearite spôsobenej použitím výstupných neurónov s nelineárnou aktivačnou funkciou je zvyčajne možné na trénovanie ESN sietí použiť algoritmy lineárnej regresie.

Nevyhnutným predpokladom pre úspešné použitie takéhoto jednoduchého a rýchleho spôsobu adaptácie je vhodne nastavený dynamický rezervoár. Dynamika siete musí byť dostatočne bohatá na to, aby bolo možné z aktivít skrytých neurónov vytvoriť požadované výstupné aktivity. Táto požiadavka je zvyčajne riešená konštruovaním rezervoára s veľkým počtom neurónov. Neuróny v rezervoári sú zvyčajne riedko poprepájané, čím sa zvyšuje pravdepodobnosť, že ich výstupné aktivity budú rozdielne. Významnými parametrami rezervoára je teda jeho veľkosť (počet neurónov v skrytej vrstve, rádovo 10 – 1000 neurónov) a pravdepodobnosť existencie rekurentného váhového prepojenia (0.1 – 10 %). Ďalšou významnou a potrebnou vlastnosťou rezervoára je existencia už spomínaných „echo“ stavov. Aby mala ESN sieť potrebné vlastnosti, musí mať kontraaktívnu dynamiku, alebo, inak povedané, jej rekurentné váhy musia byť patrične „malé“. Zaujímavé vlastnosti rekurentných sietí inicializovaných s malými váhami sme opísali v predchádzajúcej časti. Kontraktívnosť je v ESN sieti zvyčajne dosiahnutá tým, že matica rekurentných váh je nastavená tak, aby mala požadovaný spektrálny polomer. Spektrom matice nazývame jej vlastné čísla a spektrálny polomer matice je maximum z absolútnych hodnôt jej vlastných čísel. Parametre sa zvyčajne určujú experimentálne na základe expertnej skúsenosti a výber vhodnej siete je realizovaný bežnými technikami, ako je napríklad krosvalidácia.

Cieľom trénovania je nájdenie výstupných váhových prepojení, čiže určenie matice $\mathbf{W}^{\mathrm{OUT}}$. Hodnoty ostatných váhových prepojení, čiže hodnoty v maticiach $\mathbf{W}^{\mathrm{IN}}$, $\mathbf{W}$ a , $\mathbf{W}^{\mathrm{BACK}}$ sú určené pri inicializácií siete a nie sú v priebehu trénovania modifikované. Trénovanie spočíva v prezentovaní vstupnej postupnosti ESN siete (vstupy $\mathbf{u}(t)$ pre $t=1$ až T) a zaznamenaní vypočítaných skrytých aktivít do matice $\mathbf{X}$, kde stĺpce matice tvoria vektory $\mathbf{v}(t)$ vytvorené zreťazením vektorov $\mathbf{u}(t), \mathbf{x}(t), \mathbf{y}(t-1)$, čiže $\mathbf{v}(t)^T = \left[\mathbf{u}(t)^T, \mathbf{x}(t)^T, \mathbf{y}(t-1)^T\right]$. Je dôležité, aby výstupné neuróny boli vybavené prahovým váhovým prepojením, čiže aby každý vstupný vektor $\mathbf{u}(t)$ obsahoval aj špeciálny vstup neustále nastavený na hodnotu 1. Želané výstupné aktivity $\hat{\mathbf{y}}(t)$ určujú maticu $\mathbf{Y}$. Ak výstupné neuróny majú lineárnu aktivačnú funkciu, tak stĺpce matice $\mathbf{Y}$ sú tvorené priamo požadovanými výstupmi $\hat{\mathbf{y}}(t)$, v opačnom prípade, pretože je aktivačná funkcia invertovateľná, sú stĺpce $\mathbf{Y}$ tvorené vektormi vypočítanými ako $f^{-1}(\hat{\mathbf{y}}(t))$. Nájdenie matice $\mathbf{W}^{\mathrm{OUT}}$ spočíva vo vyriešení rovnice:

$$\mathbf{Y} = \mathbf{W}^{\mathrm{OUT}} \cdot \mathbf{X}. \tag{74}$$

Rovnica zvyčajne nemá riešenie a na nájdenie aproximatívneho riešenia je možné použiť metódu najmenších štvorcov:

$$\mathbf{W}^{\mathrm{OUT}} = \mathbf{Y} \cdot \mathbf{X}^{\oplus}, \tag{75}$$

kde $\mathbf{X}^{\oplus}$ je pseudoinverzná matica, ktorá môže byť vypočítaná ako:

$$\mathbf{X}^{\oplus} = \mathbf{X}^T \cdot \left(\mathbf{X} \cdot \mathbf{X}^T\right)^{-1}. \tag{76}$$

Na trénovanie ESN siete je tiež možné použiť aj iteratívnu metódu, a to rekurzívnu metódu najmenších štvorcov (angl. Recursive Least Squares, RLS), ktorá má menšie nároky na pamäť. Úprava váh sa potom riadi podľa vzťahov:

$$\mathbf{k}(t) = \frac{\mathbf{P}(t-1)\mathbf{v}(t)}{\mathbf{v}(t)^T \mathbf{P}(t-1)\mathbf{v}(t) + \gamma}, \tag{77}$$

$$\mathbf{P}(t) = \gamma^{-1}\left[\mathbf{P}(t-1) - \mathbf{k}(t)\mathbf{v}(t)^T \mathbf{P}(t-1)\right], \tag{78}$$

$$\mathbf{W}^{\mathrm{OUT}}(t) = \mathbf{W}^{\mathrm{OUT}}(t-1) + \mathbf{k}(t)\left[\hat{\mathbf{y}}(t) - \mathbf{y}(t)\right]^T, \tag{79}$$

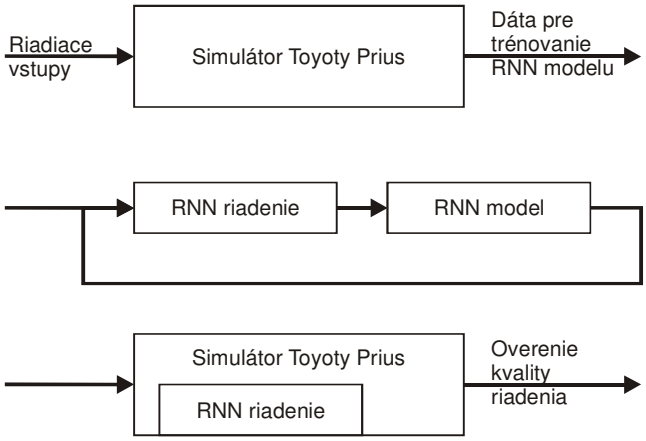
kde $\mathbf{k}(t)$ je inovačný vektor vypočítaný v každom časovom kroku t. Vektor $\mathbf{v}(t)$ je vytvorený zreťazením vektorov $\mathbf{u}(t)$, $\mathbf{x}(t)$ a $\mathbf{y}(t-1)$. Vektor želaných a vektor vypočítaných výstupných aktivít sú označené $\hat{\mathbf{y}}(t)$ a $\mathbf{y}(t)$. $\mathbf{P}(t)$ je kovariančná matica chýb odhadu váh inicializovaná vysokými hodnotami na diagonále a je upravovaná v každom kroku. Parameter γ je inicializovaný na hodnotu menšiu alebo rovnú 1 a určuje mieru zabúdania.

8 Aplikácie rekurentných neurónových sietí

Rekurentné neurónové siete boli navrhnuté na riešenie mnohých úloh z praxe. Súčasnú úroveň poznania budeme demonštrovať aplikáciou Elmanovej rekurentnej neurónovej siete adaptovanou viacprúdovým Kalmanovým filtrom na riešenie úlohy riadenia hybridného pohonu automobilu Toyota Prius [24]. Na druhej strane ESN siete patria v súčasnosti medzi modely, ktorých vlastnosti sa zatiaľ iba intenzívne študujú. Jednou z navrhnutých aplikácií ESN sietí je modelovanie nelineárneho bezdrôtového komunikačného kanála [12].

8.1 Riadenie hybridného pohonu Toyoty Prius

Hybridné pohony sú čoraz populárnejšie, pretože predstavujú možnosť, ako výrazným spôsobom zlepšiť úspornosť prevádzky automobilov a zredukovať škodlivé emisie. Stratégia riadenia automobilu vybaveného hybridným pohonom je podstatne zložitejšia, ako v prípade klasických automobilov vybavených iba spaľovacími motormi. Hlavnou úlohou riadenia pohonu je určovanie, aká časť energie bude dodávaná z elektrického motoru a aká časť zo spaľovacieho motoru. Pritom je nevyhnutné minimalizovať spotrebu paliva a emisií, avšak je nevyhnutné zabezpečiť potrebnú pojazdnosť a spoľahlivosť, a tiež je potrebné brať do úvahy úroveň nabitia batérie. Viaceré odborné články poukázali na možnosť výrazného zlepšenia efektivity prevádzky hybridného vozidla voľbou vhodného riadenia pohonu [24].



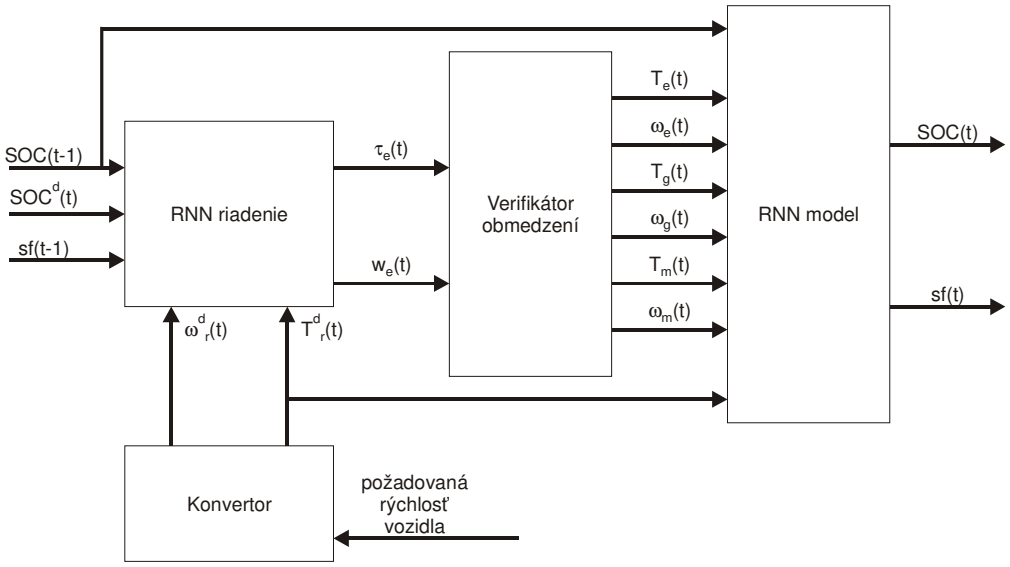
Obr. 19. Kroky trénovania riadenia pohonu. Simulátor automobilu je najskôr použitý na vytvorenie dát pre natrénovanie modelu pohonovej časti automobilu. Potom je takto vytvorený model použitý na trénovanie riadenia pohonu. Nakoniec je kvalita vytvoreného riadenia pohonu overená v simulátore automobilu. Obrázok je vytvorený podľa [24].

V tradičných schémach hybridného riadenia pohonu sú spaľovací a elektrický pohon zapojené sériovo alebo paralelne. Toyota Prius využíva inovatívny spôsob prepojenia pohonov, v ktorom je spaľovací motor prepojený s elektrickým pohonom a generátorom pomocou planetárnej prevodovky. Úlohou riadenia je udržať spaľovací

pohon v optimálnom režime s vysokým výkonom a nízkymi emisiami. Hoci je efektivita riadenia pohonu v automobile Toyota Prius vysoká, v [24] poukázali na možnosť jeho podstatného zlepšenia využitím rekurentnej neurónovej siete.

Pri navrhovaní riadiaceho člena realizovaného pomocou rekurentnej neurónovej siete bol využitý simulátor Toyoty Prius. Tento simulátor je distribuovaný komplexný systém, a tak jeho priame použitie je sťažené. Simulátor bol teda použitý na vytvorenie modelu simulujúceho podstatné vlastnosti automobilu z hľadiska riadenia hybridného pohonu. Tento model bol tiež vytvorený pomocou rekurentnej neurónovej siete. Kroky trénovania riadiaceho člena sú znázornené na obr. 19.

Model pohonu bol vytvorený rekurentnou neurónovou sieťou s 25 neurónmi. Trénovacia množine pozostávala z údajov získaných počas 20 jázd vykonaných na simulátore. Rekurentná sieť bola trénovaná viacprúdovým Kalmanovým filtrom a trénovanie trvalo 3000 epoch. Časť siete bola trénovaná na modelovanie spotreby a druhá časť na modelovanie stavu batérie elektrického pohonu.



Obr. 20. Blokový diagram systému na trénovanie radiacej časti pohonu využívajúcej rekurentnú neurónovú sieť. Symboly ω a w označujú rýchlosti a τ a T krútiace momenty, sf spotrebu paliva a SOC je stav nabitia batérie. Symbol d ako horný index označuje požadované hodnoty. Symboly e, g, m a r ako dolné indexy označujú premenné patriace elektrickému pohonu, generátoru, spaľovaciemu motoru a celkovému pohonu. Obrázok je vytvorený podľa [24].

Aj radiaca časť pohonu bola vytvorená pomocou Elmanovej rekurentnej neurónovej siete tvorenej 5 vstupmi, 10 skrytými a 2 výstupnými neurónmi. Vstupom do siete je požadovaná rýchlosť jazdy ω_r^d a požadovaný krútiaci moment ω_r^d, predchádzajúca spotreba spaľovacieho motora sf, predchádzajúci stav nabitia batérie SOC a želaný stav nabitia batérie $\mathrm{SOC}^{\mathrm{d}}$. Výstupom siete sú dva riadiace signály, a to

krútiaci moment spaľovacieho motora τ_e a jeho rýchlosť w_e, z ktorých sa po kontrole obmedzení stane T_e a ω_e. Bloková schéma trénovania je zobrazená na obr. 20.

Cieľom trénovania je vytvoriť také riadenie, ktoré bude minimalizovať spotrebu sf. Druhým, nemenej významným, cieľom je zachovať stav nabitia batérie v bezpečnej zóne definovanej ako $\mathrm{SOC} > 0.4$. Stav nabitia batérie SOC je hodnota z intervalu $\langle 0.0,1.0 \rangle$, kde 0.0 reprezentuje úplne vybitú batériu a 1.0 maximálne nabitú batériu. Tieto dva ciele boli skombinované do účelovej funkcie:

$$\mathrm{cost}(t) = \lambda_1 \mathrm{sf}(\mathrm{t})^2 + \lambda_2(t)\left(SOC^d(t) - SOC(t)\right)^2, \tag{80}$$

kde parameter $\lambda_1 = 1$ a parameter $\lambda_2 = 10$ ak $SOC(t) \geq SOC^d(t)$ a $\lambda_2 = 50$ ak $SOC(t) < SOC^d(t)$.

Aj na trénovanie tejto siete autori použili viacprúdový Kalmanov filter a derivácie určovali algoritmom BPTT s oknom do minulosti nastaveným na 20. Každý prúd v jednom trénovacom cykle spracovával 50 hodnôt získaných z referenčnej jazdy, ktorej iniciálne hodnoty boli určené náhodne. Aj $SOC^d(0)$ bolo inicializované náhodne z intervalu $\langle 0.5,0.8 \rangle$ a bolo udržiavané konštantné počas celého trénovacieho cyklu (jazdy). Celkovo trénovanie pozostávalo z 1200 epoch (približne 60000 zmien váhových prepojení).

Po natrénovaní bola riadiaca časť umiestnená do simulátora Toyoty Prius a jej výkonnosť bola porovnaná s pôvodnou riadiacou časťou. Navrhnutá metóda vykazovala v priemere o 17 % lepšiu spotrebu a zmeny stavu nabitia batérie boli v priemere o 35 % nižšie.

8.2 Modelovanie nelineárneho komunikačného kanála

Nelineárnym bezdrôtovým komunikačným kanálom je možné prenášať informáciu nasledovný spôsobom. Odosielateľ chce odkomunikovať postupnosť symbolov $s(t)$. Táto postupnosť je najskôr transformovaná do analógových hodnôt $d(t)$, potom je modulovaná do vysokofrekvenčného prenosového signálu a následne je odoslaná. Po prijatí je signál demodulovaný do analógového signálu $u(t)$, ktorý je zašumenou a poškodenou verziou originálneho signálu $d(t)$. Hlavným zdrojom poškodenia je šum (tepelný alebo spôsobený interferenciou signálov), potom propagácia viacerými cestami, ktorá sa prejaví ako superpozícia susedných symbolov (medzisymbolová interferencia) a nakoniec nelineárna deformácia spôsobená vysielačom v časti zosilňovača. Aby bola táto deformácia čo najnižšia, signál býva zosilnený hlboko pod možné maximum, čo vedie k neefektívnemu využitiu možností zariadení mobilnej a satelitnej komunikácie. Poškodený signál $u(t)$ vstupuje do ekvalizačného filtra s výstupom $y(t)$, ktorého úlohou je obnoviť originálny signál. Nakoniec je signál $y(t)$ spätne transformovaný na postupnosť symbolov. Kvalita celého procesu je určená množstvom nesprávne prenesených symbolov.

Model nelineárneho kanála bol v [12] tvorený lineárnym systémom s veľkosťou pamäti 10 a následnou nelineárnou transformáciou. Vstup do kanála je postupnosť $d(t)$ vytvorená z hodnôt $\{-3;-1;1;3\}$ zodpovedajúcich symbolom s_1 až s_4. Vstupno-výstupná rovnica lineárnej časti kanála je daná vzťahom:

$$q(n) = 0.08d(n+2) - 0.12d(n+1) + d(n) + 0.18d(n-1) - 0.1d(n-2) + \\ + 0.09d(n-3) - 0.05d(n-4) + 0.04d(n-5) + 0.03d(n-6) + 0.01d(n-7) \quad (81)$$

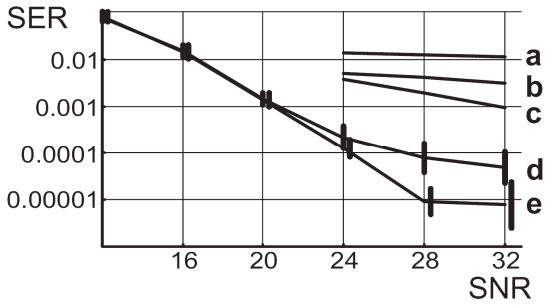
Nelineárna časť kanála nemá pamäť a je určená vzťahom:

$$u(n) = q(n) + 0.036q(n)^2 - 0.011q(n)^3 + v(n), \quad (82)$$

kde v je aditívny biely gausovský šum.

Úlohou ESN siete je zrekonštruovať pôvodný signál zo zašumených poškodených hodnôt, čiže realizovať funkciu ekvalizačného filtra. V [12] navrhnutá ESN sieť obsahovala rezervoár so 46 neurónmi. Pravdepodobnosť existencie rekurentných váhových prepojení bola 20 % a hodnoty váh boli inicializované náhodne z intervalu $(-1.0,1.0)$. Následne bola matica rekurentných váhových prepojení $\mathbf{W}$ upravená tak, aby jej spektrálny polomer bol 0.5. Úprava je realizovaná jej prenásobením konštantou $\lambda^{\text{new}}/\lambda^{\text{old}}$, čiže $\mathbf{W}^{\text{new}} = \mathbf{W}^{\text{old}}\lambda^{\text{new}}/\lambda^{\text{old}}$ kde λ^{old} je spektrálny polomer pôvodnej matice $\mathbf{W}^{\text{old}}$ a $\lambda^{\text{new}} = 0.5$ je zvolený spektrálny polomer. Vstupné váhy boli vygenerované z intervalu $(-0.025,0.025)$. Výstupný neurón mal lineárnu aktivačnú funkciu. Vstupom pre ESN sieť nebol priamo poškodený signál $u(n)$, ale hodnoty vytvorené pričítaním konštanty, a to $u(n)+30$. Želaný signál použitý na trénovanie bol $d(n-2)$, čiže úlohou siete bolo minimalizovať $(y(n)-d(n-2))^2$. Týmto spôsobom je odozva ekvalizačného filtra posunutá oproti vstupu až o 4 časové kroky, nakoľko $u(n)$ závisí od hodnôt d až do času $n+2$. Parameter zabúdania bol zvolený $\gamma = 0.998$.

Pre zvolené odstupy signál-šum (angl. Signal-to-Noise Ratio, SNR) od 12 do 32 db autori natrénovali 20 sietí na 5000 krokoch rekurzívnou metódou najmenších štvorcov. Počas prvých 100 krokov trénovanie neprebiehalo, tieto kroky slúžili na ustálenie aktivít v rezervoári. Sieť bola testovaná na 10^7 krokoch. Výstup siete bol konvertovaný na symboly priamočiarym spôsobom, napr. ak hodnota $y(t) > 2$, uvažuje sa symbol zodpovedajúci hodnote 3, čiže symbol s_4. Kvalita modelu bola určená ako podiel nesprávne prenesených symbolov ku všetkým prenesenýmsymbolom (angl. Symbol Error Rate, SER). Výsledky experimentov sú znázornené na obr. 21. V [12] bola ESN porovnaná s lineárnym ekvalizérom so spätnou väzbou (angl. Decision Feedback Equalizer, DFE), s Volterrovým DFE a s bilineárnym DFE.



Obr. 21. Porovnanie chybovosti SER v závislosti na SNR (odstup signál – šum). ESN siete sú porovnané s inými prístupmi: (a) Lineárna DFE, (b) Volterra DFE, (c) bilinear DFE. (d) označuje priemer vytvorený z výsledkov viacerých ENS sietí a (e) sú výsledky najlepších sietí použitých v (d). Obrázok je prevzatý z [12].

Relatívne malá ESN sieť s rezervoárom so 46 neurónmi bola zvolená preto, aby počet adaptovaných parametrov bol porovnateľný s ostatnými modelmi. Napriek tomu bola ESN sieť schopná úlohu riešiť s výrazne lepšou chybovosťou ako iné metódy [12]. Tento príklad poukazuje na možnosť použitia ESN sietí na riešenie praktických úloh. Významnou výhodou použitia ESN sietí je možnosť ich rýchleho natrénovania, čo umožňuje experimentovať s mnohými variantmi a parametrami.

9 Implementácia rekurentných neurónových sietí

V predchádzajúcom texte sme algoritmy na trénovanie rekurentných neurónových sietí formalizovali pomocou zápisu, ktorý je možný jednoduchým spôsobom prepísať do pseudojazyka. V tejto časti teda zapíšeme algoritmy v jednoduchom pseudojazyku takým spôsobom, ktorý umožňuje ich takmer okamžité použitie v bežných procedurálnych programovacích jazykoch, akými sú napríklad jazyk C či Java.

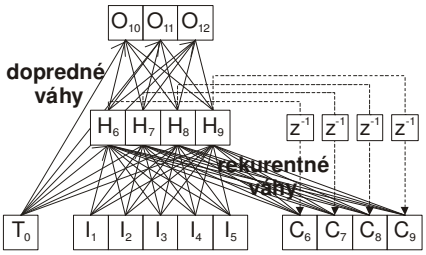
9.1 Reprezentácia neurónovej siete v dátových štruktúrach

Pri zápise algoritmov pre trénovanie rekurentných neurónových sietí sme využívali operátory d, s, τ a w pracujúce nad usporiadanou množinou váhových prepojení, ktorá určovala štruktúru siete. Každé váhové prepojenie v rekurentnej neurónovej sieti malo svoj jedinečný index l a bolo reprezentované usporiadanou štvoricou (i, j, t, v), kde i bol index cieľového a j index zdrojového neurónu, t časové oneskorenie a v intenzita váhy. Nech (i, j, t, v) je váhové prepojenie s indexom l, tak potom $d_l = i$, $s_l = j$, $\tau_l = t$ a $w_l = v$. Množinu indexov všetkých váhových prepojení sme označili W.

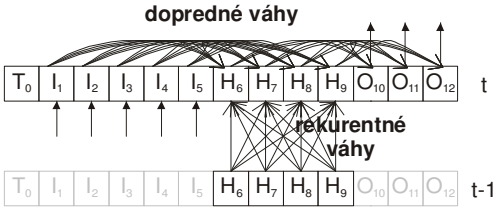
Tento spôsob zápisu štruktúry siete je možné jednoduchým spôsobom previesť na zápis v pseudojazyku. Množinu váhových prepojení si môžeme predstaviť ako tabuľku záznamov, ktoré reprezentujú váhové prepojenia. Pre jednoduchší zápis trénovacích algoritmov budeme pomocou tabuľky udržiavať aj informácie o neurónoch v sieti. Na

obr. 22 je znázornený spôsob reprezentácie Elmanovej rekurentnej neurónovej siete v údajových štruktúrach.

a) Elmanova architektúra - kontextová vrstva



b) Elmanova architektúra - rozvinutie v čase



c) Štruktúra váhových prepojení

Index váhy	Zdrojový neurón	Cieľový neurón	Časové ones.	Hodnota
	wSource	wDest	wDelay	wValue
0	0	6	0	
1	1	6	0	
2	2	6	0	
3	3	6	0	
4	4	6	0	
5	5	6	0	
6	6	6	1	
7	7	6	1	
8	8	6	1	
9	9	6	1	
10	0	7	0	
11	1	7	0	
12	2	7	0	
13	3	7	0	
14	4	7	0	
15	5	7	0	
16	6	7	1	
17	7	7	1	
18	8	7	1	
19	9	7	1	
20	0	8	0	
21	1	8	0	
22	2	8	0	
23	3	8	0	
24	4	8	0	
25	5	8	0	
26	6	7	1	
27	7	7	1	
28	8	7	1	
29	9	7	1	

Index váhy	Zdrojový neurón	Cieľový neurón	Časové ones.	Hodnota
	wSource	wDest	wDelay	wValue
30	0	9	0	
31	1	9	0	
32	2	9	0	
33	3	9	0	
34	4	9	0	
35	5	9	0	
36	6	7	1	
37	7	7	1	
38	8	7	1	
39	9	7	1	
40	0	10	0	
41	6	10	0	
42	7	10	0	
43	8	10	0	
44	9	10	0	
45	0	11	0	
46	6	11	0	
47	7	11	0	
48	8	11	0	
49	9	11	0	
50	0	12	0	
51	6	12	0	
52	7	12	0	
53	8	12	0	
54	9	12	0	

d) Štruktúra neurónov

Index neurónu	Prvá váha	Posledná váha	Typ	Aktivačná funkcia
UI	uFirstWeight	uLastWeight	uType	uActFunc
0			T	
1			I	
2			I	
3			I	
4			I	
5			I	
6	0	5	H	SGM
7	6	11	H	SGM
8	12	17	H	SGM
9	18	23	H	SGM
10	24	28	O	LIN
11	29	33	O	LIN
12	34	38	O	LIN

Obr. 22. Reprezentácia Elmanovej rekurentnej neurónovej siete s 5 vstupnými, 4 skrytými a 3 výstupnými neurónmi. a) sieť znázornená s využitím kontextovej vrstvy, b) sieť znázornená v časovom kroku (rozvinutie), c) tabuľka obsahujúca informácie o váhových prepojeniach a d) tabuľka s informáciami o neurónoch.

Zápis predpokladá, že všetky neuróny v sieti sú usporiadané a oindexované. Prvý bude prahový neurón s indexom 0, potom nasledujú vstupné, skryté a výstupné neuróny. Neuróny s vyšším indexom nesmú poskytovať svoju aktivitu cez dopredné váhové prepojenie (váhové prepojenie s časovým oneskorením 0) neurónom s nižším indexom. Táto požiadavka reprezentuje fakt, že keď počítame aktivitu neurónu musíme mať už vypočítané aktivity neurónov, s ktorými je daný neurón spojený váhovými prepojeniami. Pre všetky bežne používané architektúry neurónových sietí pracujúcich v diskrétnom čase je možné neuróny jednoduchým spôsobom usporiadať tak, aby táto požiadavka bola splnená. O každom neuróne je vhodné si uchovať informáciu o jeho

prvej a poslednej váhe, jeho type (prahový, vstupný, skrytý, výstupný) a napr. o jeho aktivačnej funkcii. Spôsob výpočtu aktivity bude zrejmý z ďalšieho textu. Váhy sú usporiadané najskôr podľa indexu cieľového neurónu, potom podľa indexu zdrojového neurónu. Tiež je potrebné pre každé váhové prepojenie uchovávať informáciu o jeho intenzite (vlastná váha) a pre rekurentné siete aj o jeho časovom oneskorení.

9.2 Dopredné šírenie signálu

Použitie uvedeného spôsobu kódovania neurónovej siete bude zrejmé z nasledovného algoritmu zapísaného v pseudojazyku (obr. 23), ktorého úlohou je výpočet aktivít na jednotlivých neurónoch zo vstupov prezentovaných sietí. Už aj táto relatívne jednoduchá súčasť každého simulátora sietí je vďaka použitiu zvoleného kódovania ešte jednoduchšia a prehľadnejšia.

```
NW                                  - počet váh
wSource[0..NW-1]                    - zdrojové neuróny
wDest[0..NW-1]                      - cieľové neuróny
wDelay[0..NW-1]                     - časové oneskorenia
wValue[0..NW-1]                     - vlastné hodnoty váh. prepojení

NU                                  - počet neurónov
uFirstWeight[0..NU-1]               - indexy prvých váh
uLastWeight[0..NU-1]                - indexy posledných váh
uType[0..NU-1]                      - typy neurónov (THRESHOLD,INPUT...OUTPUT)


NSTEPS                              - počet krokov
ACT[0..NU-1,0..NSTEPS-1]            - všetky aktivity neurónov
ACTD[0..NU-1,0..NSTEPS-1]           - derivácie aktivít neurónov

Sgm(iact)                           - aktivačná funkcia
SgmDer(iact)                        - derivácia aktivačnej funkcie
Input(ui,ts)                        - f. vráti vstup zadaný sieti
Output(ui,act,ts)                   - f. nastaví výstup siete
Target(ui,ts)                       - f. vráti želaný výstup siete
ts                                  - aktuálny časový krok
```

```
1.  for ui:=0 to NU-1 do
2.    begin
3.      if uType[ui] = THRESHOLD then ACT[ui,ts] := 1.0;
4.      else if uType[ui] = INPUT then
5.        ACT[ui,ts] := Input(ui,ts);
6.      else
7.      begin
8.        iact := 0.0;
9.        for wi:=uFirstWeight[ui] to uLastWeight[ui] do
10.         iact +:= wValue[wi]*ACT[wSource[wi],ts-wDelay[wi]];
11.       ACT[ui,ts] := Sgm(iact);
12.       ACTD[ui,ts] := SgmDer(iact);
13.     end;
14.     if uType = OUTPUT then Output(ui,ACT[ui,ts],ts);
15.   end;
```

Obr. 23. Zápis algoritmu pre dopredné šírenie signálu v rekurentnej neurónove sieti.

Obr. 23. Zápis algoritmu pre dopredné šírenie signálu v rekurentnej neurónovej sieti.

Tento algoritmus opisuje jeden krok (krok ts) dopredného šírenia. Prvý neurón s indexom 0 zodpovedá prahovej váhe a jeho aktivita je vždy rovná 1 (riadok 3).

Aktivity na vstupných neurónoch sú nastavené podľa aktuálneho vstupu predloženého sieti (riadky 4 a 5). Vnútorné aktivity na skrytých a výstupných neurónoch sú počítané násobením váh neurónov s aktivitami na zdrojových neurónoch zo zodpovedajúceho časového kroku (riadky 9 až 10). Operátor +:= reprezentuje pričítanie výrazu na jeho pravej strane k premennej na ľavej strane. Pre dopredné váhy majúce časové oneskorenie 0 sú použité už vypočítané aktivity v aktuálnom časovom kroku, pre rekurentné váhy sú použité aktivity z minulých krokov. Vlastná aktivita na neuróne je vypočítaná aktivačnou funkciou na základe internej aktivity neurónu (riadok 11). Výpočet derivácie aktivačnej funkcie je potrebný iba vtedy, keď sa bude následne realizovať adaptácia váhových prepojení gradientovými algoritmami (riadok 12). Aktivity na výstupných neurónoch sú poskytnuté ako výstupy zo siete (riadok 14).

V prípade rekurentnej siete je potrebné sa zamyslieť nad iniciálnymi časovými krokmi. Pre čas ts menší ako časové oneskorenie váhy hrozí, že pole ACT bude pristupované so záporným indexom. (riadok 10). Vhodným riešením môže byť nastavenie ts pre čas 0 na hodnotu najdlhšieho časového oneskorenia v sieti (najväčšia hodnota v poli wDelay) a aktivity pre menšie ts prehlásiť za iniciálny stav siete.

9.3 Spätné šírenie chybového signálu

Elmanova jednoduchá rekurentná neurónová sieť bola v [2] trénovaná jednoduchým algoritmom spätného šírenia chyby v čase (angl. Backpropagation, BP). BP algoritmus je najpoužívanejším prístupom na adaptáciu váh v doprednej sieti. Aj keď nie je vhodný pre trénovanie rekurentných sietí, môže byť použitý na jednoduchšie úlohy alebo ako referenčná metóda. Zápis BP algoritmu v pseudojazyku je na obr. 24.

Derivácie DE_DNA[ui] sú postupne spätne vypočítavané, a preto musia byť na začiatku nastavené na 0 (riadok 1). Ďalsia časť zdrojového kódu slúži na výpočet derivácií (riadky 3 až 13). Začínajúc posledným neurónom s indexom NU-1 sú postupne vypočítavané derivácie DE_DNA[ui]. Hneď, ako sú spracované všetky výstupné a skryté neuróny, cyklus končí (riadok 5), lebo už boli vypočítané všetky derivácie. Ak je aktuálne spracovávaný neurón ui výstupným neurónom, rozdiel medzi želanou a skutočnou aktivitou je pripočítaný do práve vypočítavanej derivácie (riadky 6 a 7). Nakoniec, keďže všetky relevantné chybové signály boli už z neurónov s vyšším indexom spätne prešírené do aktuálne počítanej derivácie DE_DNA[ui], je táto hodnota prenásobená deriváciou aktivačnej funkcie (riadok 8) a je ďalej spätne šírená na všetky neuróny, ktoré sú vstupmi pre aktuálny neurón (riadky 9 až 12). Nemá zmysel spätne šíriť chybový signál na vstupné neuróny (riadok 10) a taktiež chybový signál sa nešíri cez rekurentné váhy (riadok 11). Posledná časť algoritmu je zmena váhových prepojení (riadky 15 až 20). Zmena váhy (DLT_W[wi]) je vypočítaná prenásobením rýchlosti učenia alpha s chybovým signálom na danom neuróne a aktivitou na zdrojovom neuróne. K zmene váhy v súčasnom kroku je pridaná aj zmena váhy vypočítaná v predchádzajúcom časovom kroku prenásobená momentom beta (riadky 17 a 18). Nakoniec je zmenená samotná sila váhového prepojenia wValue[wi] (riadok 19).

```
DE_DNA[0..NU-1]                    - chyb. sig., derivácia chyby voči int. akt.
DLT_W[0..NW-1]                     - vypočítané zmeny váh
beta                               - momentum
alpha                              - rýchlosť učenia

1.  for ui=NU-1 downto 0 do DE_DNA[ui] := 0.0;
2.
3.  for ui=NU-1 downto 0 do
4.    begin
5.      if uType[ui] = INPUT then break;
6.      if uType[ui] = OUTPUT then
7.        DE_DNA[ui] +:= Target(ui,ts)-ACT[ui,ts];
8.      DE_DNA[ui] *:= ACTD[ui,ts];
9.      for wi := uLastWeight[ui] downto uFirstWeight[ui] do
10.       if uType[wSource[wi]] = INPUT then break;
11.       else if wDelay[wi] = 0 then
12.         DE_DNA[wSource[wi]] +:= wValue[wi]*DE_DNA[ui];
13.   end;
14.
15. for wi:=0 to NW-1 do
16. begin
17.   DLT_W[wi] := beta*DLT_W[wi] +
18.     alfa*DE_DNA[wDest[wi]]*ACT[wSource[wi],ts-wDelay[wi]];
19.   wValue[wi] +:= DLT_W[wi];
20. end;
```

Obr. 24. Algoritmus spätného šírenia chybového signálu.

Jednoduchý BP algoritmus nie je vhodný na trénovanie rekurentných neurónových sietí, lebo nešíri chybový signál spätne aj cez rekurentné prepojenia. Jednoduchá modifikácia uvedeného algoritmu, nazývaná spätné šírenie chyby v čase, rieši tento problém.

9.4 Spätné šírenie chybového signálu v čase

Algoritmus spätného šírenie chyby v čase (angl. Backpropagation Through Time, BPTT) umožňuje presný výpočet derivácie chyby pre rekurentné neurónové siete. Princíp spočíva v rozvinutí rekurentnej siete v čase do potenciálne mnohovrstvovej doprednej siete a v následnom použití klasického BP algoritmu [34]. V praxi nie je rekurentná sieť rozvíjaná v čase až do času 0, ale iba istý počet krokov do minulosti označovaných ako veľkosť okna. Táto modifikácia už bola popísaná a nazýva sa skrátené BPTT. Zápis BPTT algoritmu v pseudojazyku je na obr. 25.

Podobne, ako v prípade BP algoritmu, musia byť derivácie DE_DNA inicializované na 0. (cyklus na riadku 1). Výpočet v časovom kroku ts spočíva v spätnom šírení chybového signálu winSize-1 časových krokov späť (vonkajší cyklus na riadkoch 3 až 17). Pre každý neurón ui (cyklus na riadkoch 4 až 17) a pre všetky jeho váhy (cyklus na riadkoch 10 až 16) sú derivácie DE_DNA[ui,hi] spätne šírené cez zodpovedajúce prepojenia do derivácii v zodpovedajúcom čase (riadok 14), rekurentné prepojenia sú tiež uvažované. Pre výstupné neuróny a iba pre hi rovné 0 sa chybový signál obohatí o rozdiel medzi želanou a skutočnou hodnotou (riadok 7 a 8). Výpočet derivácie DE_DNA[ui,hi] je ukončený prenásobením akumulovanej hodnoty deriváciou aktivačnej funkcie v zodpovedajúcom čase (riadok 9). Ako aj v prípade BP

algoritmu signál nie je šírený na vstupné neuróny (riadok 12). Rovnako je zabezpečený iba povolený prístup do poľa DE_DNA (podmienka na riadku 13). Zmeny váh sú vypočítavané v rovnakom cykle ako spätné šírenie. Do zmeny váhy DLT_W[wi] je pripočítaná časť pripadajúca na práve spočítaný chybový signál v čase (riadok 15). Váhy sú zmenené na riadkoch 19 až 23. Časový krok ts musí byť dostatočne vysoký, aby nedošlo k nepovolenému prístupu do poľa ACT (riadok 15).

```
DE_DNA[0..NU-1,0..winSize-1]        - spätne šírený chybový signál
winSize                             - veľkosť okna

1.   for hi=0 to winSize-1 do for ui=NU-1 downto 0 do DE_DNA[ui,hi] := 0.0;
2.
3.   for hi=0 to winSize-1 do
4.     for ui=NU-1 downto 0 do
5.     begin
6.       if uType[ui] = INPUT then break;
7.       if (uType[ui] = OUTPUT) AND (hi = 0) then
8.         DE_DNA[ui,hi] +:= Target(ui,ts)-ACT[ui,ts];
9.       DE_DNA[ui,hi] *:= ACTD[ui,ts-hi];
10.      for wi := uLastWeight[ui] downto uFirstWeight[ui] do
11.      begin
12.        if uType[wSource[wi]] = INPUT then break;
13.        if wDelay[wi]+hi < winSize then
14.           DE_DNA[wSource[wi],wDelay[wi]+hi] +:= wValue[wi]*DE_DNA[ui,hi];
15.        DLT_W[wi] +:= alfa*DE_DNA[ui,hi] * ACT[wSource[wi],ts-hi-wDelay[wi]];
16.      end;
17.    end;
18.
19.  for wi:=0 to NW-1 do
20.  begin
21.    wValue[wi] +:= DLT_W[wi];
22.    DLT_W[wi] *:= beta;
23.  end;
```

Obr. 25. Algoritmus spätného šírenia chybového signálu v čase.

9.5 Rekurentné učenie v reálnom čase

Ďalšou možnosťou ako zrealizovať výpočet derivácií chyby je použiť rekurenté učenie v reálnom čase (real-time recurrent learning - RTRL). Na rozdiel od BPTT algoritmu RTRL nepotrebuje presne definovaný časový interval určujúci rozvinutie siete. Hlavnou nevýhodou je jeho väčšia výpočtová náročnosť. Zápis RTRL algoritmu v pseudojazyku je na obr. 26.

Dopredne šírené derivácie DA_DW zodpovedajúce iniciálnym časovým krokom sú ešte pred začiatkom trénovania vynulované. Jeden krok trénovania algoritmom RTRL je nasledovný. Pre každé váhové prepojenie wi (cyklus na riadkoch 1 až 16) a každý neurón ui (cyklus na riadkoch 2 až 16) okrem vstupných a prahových neurónov (riadok 4) aktuálne počítaná derivácia DA_DW[ui,wi,ts] je postupne vytvorená doprednou propagáciou z už vypočítaných derivácií DA_DW[wSource[uwi],wi,ts-wDelay[uwi]] cez váhy wValue[uwi] (cyklus na riadkoch 6 až 9). Vstupné a prahové neuróny sú vynechané (riadok 8), im zodpovedajúca derivácia je vždy nulová. Riadky 11 a 12 reprezentujú explicitný efekt

ovplyvňujúci derivácie. Nakoniec je výpočet derivácie DA_DW[ui,wi,ts] dokončený jej prenásobením deriváciou aktivačnej funkcie (riadok 14).

```
NO                                  - počet výstupných neurónov
oIndex[0..NO-1]                     - pole s indexmi výstupných neurónov
DA_DW[0..NU-1,0..NW-1,0..NSTEPS-1]  - dopredne šírené derivácie

1.  for wi:=0 to NW-1 do
2.    for ui:=0 to NU-1 do
3.    begin
4.      if uType[ui] <> INPUT AND uType[ui] <> THRESHOLD then
5.      begin
6.        for uwi:=uFirstWeight[ui] to uLastWeight[ui] do
7.          if uType[wSource[uwi]] <> INPUT AND
8.            uType[wSource[uwi]] <> THRESHOLD then
9.          DA_DW[ui,wi,ts] +:=wValue[uwi]*DA_DW[wSource[uwi],wi,ts-wDelay[uwi]];
10.
11.       if wDest[wi] = ui then
12.         DA_DW[ui,wi,ts] +:= ACT[wSource[wi],ts-wDelay[wi]];
13.
14.       DA_DW[ui,wi,ts] *:= ACTD(ui,ts);
15.     end;
16.   end;
17.
18. for wi:=0 to NW-1 do
19. begin
20.   DLT_W[wi] *:= beta;
21.   for oi:=0 to NO-1 do
22.    DLT_W[wi] +:=
23.      alfa*(Target(oIndex[oi],ts)-ACT[oIndex[oi],ts])*DA_DW[oIndex[oi],wi,ts];
24.   wValue[wi] +:= DLT_W[wi];
25. end;
```

Obr. 26. Algoritmus pre rekurenté učenie v reálnom čase.

Posledný cyklus (riadky 18 až 25) implementuje zmeny váh. Najskôr je momentum beta aplikované na staré zmeny váh (riadok 20), potom je zmena váhy DLT_W[wi] postupne vypočítaná násobením rýchlosti učenia alpha s chybami na výstupných neurónoch Target(oIndex[oi],ts)-ACT[oIndex[oi],ts] a derivácii DA_DW[oIndex[oi],wi,ts] (riadky 21 až 23). Nakoniec sú na riadku 24 zmenené váhy.

9.6 Rozšírený Kalmanov filter

Ako sme už uviedli, medzi najúspešnejšie prístupy používané na trénovanie neurónových sietí patria metódy založené na Kalmanovej filtrácií. Odhadovaný stavový vektor je v prípade neurónovej siete stotožnený s trénovanými váhami siete. Na výpočet derivácií pre maticu **H** je možné použiť algoritmus skoro identický s BPTT alebo RTRL. Matica šumu pozorovaní **R** určuje rýchlosť konvergencie podobne, ako rýchlosť učenia v klasických gradientových algoritmoch. R je diagonálna matica a môže ostať v priebehu učenia konštantná alebo sa môže meniť, podobne ako rýchlosť učenia. Matica šumu procesu **Q** je tiež diagonálna inicializovaná na malé hodnoty a pomáha udržať numerickú stabilitu výpočtu. Matica kovariancie chyby **P** je tiež zvyčajne inicializovaná ako diagonálna. Typické hodnoty iniciálnych parametrov

pre matice môžu byť napr. $\mathbf{R} = 100\mathbf{I}$, $\mathbf{Q} = 1.10^{-5}\mathbf{I}$ a $\mathbf{P} = 1.10^{3}\mathbf{I}$, kde $\mathbf{I}$ je jednotková matica. Zápis RTRL algoritmu v pseudojazyku je na obr. 27.

```
NO                                  - počet výstupných neurónov
oIndex[0..NO-1]                     - pole s indexmi výstupných neurónov
DO_DW[0..NO-1,0..NW-1]              - vypočítané derivácie
DO_DNA[0..NU-1,0..winSize-1]        - spätne šírený signál
winSize                             - veľkosť okna

H[0..NO-1,0..NW-1]                  - matica derivácií
P[0..NW-1,0..NW-1]                  - matica kovariancie chýb
Q[0..NW-1,0..NW-1]                  - matica šumu procesu
R[0..NO-1,0..NO-1]                  - matica šumu pozorovaní
W[0..NW-1,0..0]                     - odhadovaný váhový vektor
K[0..NW-1,0..NO-1]                  - Kalmanov zisk
D[0..NO-1,0..0]                     - vektor želaných výstupov
O[0..NO-1,0..0]                     - vektor vypočítaných výstupov

Tr(X)                               - f. vyp. transponovanú maticu
Inv(X)                              - f. vyp. inverznú maticu
```

```
1.  for oui:=0 to NO-1 do for wi:=0 to NW-1 do DO_DW[oui,wi] := 0.0;
2.
3.  for oui:=0 to NO-1 do
4.  begin
5.    for hi:=0 to winSize-1 do for ui:=0 to NU-1 do DO_DNA[ui,hi] := 0.0;
6.    DO_DNA[oIndex[oui],0] := 1.0;
7.
8.    for hi:=0 to winSize-1 do
9.      for ui:=NU-1 downto 0 do
10.     begin
11.       if uType[ui] = INPUT then break;
12.       DO_DNA[ui,hi] *:= ACTD[ui,ts-hi];
13.       for wi := uLastWeight[ui] downto uFirstWeight[ui] do
14.       begin
15.         if (uType[wSource[wi]] = INPUT) then break;
16.         if (wDelay[wi]+hi < winSize)) then
17.           DO_DNA[wSource[wi],wDelay[wi]+hi] +:= wValue[wi]*DO_DNA[ui,hi];
18.         DO_DW[oui,wi] +:= DO_DNA[ui,hi]*ACT[wSource[wi],ts-hi-wDelay[wi]];
19.       end;
20.     end;
21. end;
22.
23. for wi:=0 to NW-1 do W[wi,0] := wValue[wi];
24.
25. for oui:=0 to NO-1 do
26. begin
27.   D[oui,0] := Target(oIndex[oui],ts);
28.   O[oui,0] := ACT[oIndex[oui],ts];
29.   for wi:=0 to NW-1 do H[oui,wi] := DO_DW[oui,wi];
30. end;
31.
32. K := P * Tr(H) * Inv(H * P * Tr(H) + R);
33. P := P - K * H * P + Q;
34. W := W + K * (D - O);
35.
36. for wi:=0 to NW-1 do wValue[wi] := W[wi,0];
```

Obr. 27. Rozšírený Kalmanov filter aplikovaný na trénovanie rekurentných neurónových sietí, derivácie sú určované spätným šírením v čase.

Hodnoty DO_DW (derivácie výstupných aktivít vzhľadom na váhy) sú inicializované na 0 (riadok 1). Následne pre každý výstupný neurón je vykonané spätné šírenie (cyklus na riadkoch 3 až 21). Pre jedno spätné šírenie je chybový signál pre jediný výstupný neurón nastavený na 1 (riadok 6), ostatné prvky poľa DO_DNA (derivácie výstupných aktivít vzhľadom na všetky vnútorné aktivity neurónov) sú nastavené na 0 (riadok č. 5) a sú vypočítavané v nasledujúcom cykle (riadok 8 až 20). Spätné šírenie je vykonané rovnakým spôsobom ako pre BPTT, iba namiesto výpočtu zmien váh sú vypočítavané derivácie DO_DW. Jedno spätné šírenie sa postará o výpočet DO_DW pre jeden výstupný neurón. Nakoniec sú nastavené matice a vektory pre KF (riadky 23 až 30) a môže byť vykonaný vlastný krok Kalmanovej filtrácie (riadky 32 až 34). Nakoniec sú vypočítané váhy umiestnené do poľa používaného pri doprednom a spätnom šírení (riadok 36).

10 Záver

Rekurentné siete sú na rozdiel od dopredných sietí schopné spracovávať úlohy s časovým kontextom. Napriek ich nesporne vysokému aplikačnému potenciálu rekurentné siete stále nie sú bežne používané na riešenie konkrétnych problémov. Tento text sa pokúsil priblížiť čitateľovi problematiku rekurentných neurónových sietí. Ťažisko príspevku tvorí opis trénovacích algoritmov, nakoľko ich pochopenie, ale aj implementácia, sú náročnejšie ako v prípade dopredných neurónových sietí. Algoritmy sú formálne opísané netradičným, ale do značnej miery univerzálnym spôsobom. Príspevok sa venuje aj algoritmom využívajúcim Kalmanovu filtráciu, ktoré reprezentujú súčasnú úroveň vedeckého poznania.

Poďakovanie: Táto kapitola vznikla za podpory grantovej agentúry VEGA v rámci grantových úloh VG-1/0848/08 a VG-1/0822/08.

Literatúra

[1] Boden, M., Wiles, J.: On learning context free and context sensitive languages. *IEEE Transactions on Neural Networks* 2, 2002, 491-493.

[2] Bengio, Y., Simard, P., Frasconi, P.: Learning Long-Term Dependencies with Gradient Descent is Difficult. IEEE Transactions on Neural Networks 2(5), 1994, 157-166.

[3] Čerňanský, M., Beňušková, L.: Finite-state Reber automaton and the recurrent neural networks trained in supervised and unsupervised manner. *Artificial Neural Networks - ICANN 2001*, Wienna, Austria. 2001, 737-742.

[4] Čerňanský, M., Beňušková, L.: Simple recurrent network trained by RTRL and extended Kalman filter algorithms. *Neural Network World* 13(3), 2003.

[5] Čerňanský, M., Makula, M., Beňušková, Ľ.: Organization of the state space of a simple recurrent neural network before and after training on recursive linguistic structures. *Neural Networks* 20, 2007, 236-244.

[6] Wan, E. A., Nelson, A. T.: Dual EKF methods. *Kalman Filtering and Neural Networks*, Hillsdale, N.J., 2000, 123–173.

[7] Elman, J. L.: Finding structure in time. *Cognitive Science* 14(2), 1990, 179-211.

[8] Feldkamp, L. A., Prokhorov, D., Eagen, C.F., Yuan, F.: Enhanced multi-stream kalman filter training for recurrent networks. *Nonlinear modeling: advanced black-box techniques*, Boston, 1998, 29-53.

[9] Frasconi, P., Gori, M., Soda, G.: Local feedback multilayered networks. *Neural Computation* 4, 1992, 120-130.

[10] Haykin, S.: *Neural Networks*. Prentice-Hall, New Jersey, 1994.

[11] Jaeger, H.: The "echo state" approach to analysing and training recurrent neural networks. Technical Report GMD 148, German National Research Center for Information Technology, 2001.

[12] Jaeger, H., Haas,. H.: Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. *Science* 5667, 2004, 78-80.

[13] Jordan, M. I.: Attractor dynamics and parallelism in a connectionist sequential machine. *Proceedings of the Eighth Conference of the Cognitive Science Society*, 1986, 531-546.

[14] Julier, S., Uhlmann, J.: A new extension of the Kalman filter to nonlinear systems. *Proceedings of the Int. Symp. Aerospace/Defense Sensing, Simul. and Controls,* Orlando. 1997, 182–193.

[15] Kalman, R. E.: A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82, 1960, 35–45.

[16] Kolen, J. F.: The origin of clusters in recurrent neural network state space. *Proceedings from the Sixteenth Annual Conference of the Cognitive Science Society* 1994, 508-513.

[17] Kvasnička, V., a kol.: *Úvod do teórie neurónových sietí*, IRIS, Bratislava 1997.

[18] Lawrence, S., Giles, C. L., Fong, S.: Natural language grammatical inference with recurrent neural networks. *IEEE Transactions on Knowledge and Data Engineering* 12(1), 2000, 126-140.

[19] Machler, M., Bühlmann, P.: Variable length Markov chains: methodology, computing and software. *Journal of Computational and Graphical Statistics*, 13, 2004, 435–455.

[20] Makula, M., Čerňanský, M., Beňušková, Ľ.: Approaches based on Markovian architectural bias in recurrent neural networks. *SOFSEM 2004: Theory and Practice of Computer Science*, 2004, 257–264.

[21] Maybeck, P., S.: Stochastic models, estimation, and control, 1979.

[22] Nørgaard, M., Poulsen, N. K., Ravn, O.: Advances in derivative-free state estimation for nonlinear systems. Technical Report IMMREP-1998-15, Department of Mathematical Modelling, DTU, 1998.

[23] Pérez-Ortiz, J. A., Gers, F. A., Eck, D., Schmidhuber, J.: Kalman filters improve LSTM network performance in problems unsolvable by traditional recurrent nets. *Neural Networks*, 16(2), 2003.

[24] Prokhorov, D. V.: Toyota Prius HEV neurocontrol and diagnostics. *Neural Networks*, 21, 2008, 458–465.

[25] Puskorius, G.V., Feldkamp, L. A.: Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks. *IEEE Transactions on Neural Networks*, 5(2), 1994, 279-297.

[26] Puskorius, G.V., Feldkamp, L. A.: Extensions and enhancements of decoupled extended Kalman filter training. *Proceedings of international Conference on Neural Networks - ICNN*, 1997, 1879-1883.

[27] Rodriguez, P.: Simple Recurrent Networks Learn Contex-free and Contex-Sensitive Languages by Counting. *Neural Computation* 13, 2001 2093-2118 .

[28] Ron, D., Singer, Y., Tishby, N.: The power of amnesia. *Machine Learning*, 25 1996, 117-149.

[29] Rumelhart, D. E, Hinton, G. E., Williams, R. J.: Learning internal representations by error propagation. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. I: Foundations*, 1986, 318-362.

[30] Tiňo, P., Vojtek, V.: Extracting stochastic machines from recurrent neural networks trained on complex symbolic sequences, *Neural Network World,* 8, 1998, 517-530.

[31] Tiňo, P., Čerňanský, M., Beňušková, Ľ.: Markovian Architectural Bias of Recurrent Neural Networks. *IEEE Transactions on Neural Networks*, 15(1), 2004, 6-15.

[32] Trebatický, P.: Recurrent Neural Network training with the Kalman Filter-based techniques, *Neural network world* 15(5) 2005, 471-488.

[33] Wan, E. A., Nelson, A. T.: Dual EKF methods. In *Kalman Filtering and Neural Networks*, Wiley, Hillsdale, N.J., 2000, 123–173.

[34] Werbos, P.J.: Backpropagation through time; what it does and how to do it. *Proceedings of the IEEE*, 78, 1990, 1550-1560.

[35] Wiles, J., Elman, J.: Learning to count without a counter: A case study of dynamics and activation landscapes in recurrent networks. *Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society*, 1995, 482 – 487.

[36] Williams, R. J.: Training recurrent networks using the extended Kalman filter. *Proceedings of the International Joint Conference on Neural Networks*, volume 4, Baltimore, 1992, 241-246.

[37] Williams, R. J., Zipser, D.: A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1, 1989, 270-280.

Kľúčové slová pre index

Neurón
Neurónová sieť
Rekurentná neurónová sieť
Spätné šírenie chybového signálu
Spätné šírenie chybového signálu v čase
Rekurentné učenie v reálnom čase
Kalmanov filter
Rozšírený Kalmanov filter
Oddelený Kalmnaov filtre
Viacprúdový Kalmanov filter
Elmanova jednoduchá rekurentná neurónová sieť
Sieť s echo stavmi
Modelovanie časových postupností
Spracovanie postupností symbolov
Predikcia nasledujúceho symbolu
Dynamika nenatrénovanej rekurentnej neurónovej siete
Kontraktívne transformácie
Markovovská architekturálna predispozícia
Markovov model
Markovov model s premenlivou dĺžkou pamäti
Nelineárny bezdrôtový komunikačný kanál
Riadenie hybridného pohonu