Univerzita Komenského, Bratislava
Fakulta Matematiky, Fyziky a Informatiky

Učenie sekvencií pomocou hierarchickej neurónovej siete s echo stavmi

2011 Bc. Viliam Dillinger

Univerzita Komenského, Bratislava
Fakulta Matematiky, Fyziky a Informatiky
Katedra Informatiky

Učenie sekvencií pomocou hierarchickej neurónovej siete s echo stavmi

(diplomová práca)

Bc. Viliam Dillinger

Študijný program: Informatika
Študijný odbor: 9.2.1 Informatika
Školiace pracovisko: Katedra aplikovanej informatiky
Vedúci práce: doc. Ing. Igor Farkaš, PhD.
Evidenčné číslo: bb63e2b0-9413-4703-a7c6-740a66883927

Bratislava, 2011

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Viliam Dillinger
Študijný program: informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: 9.2.1. informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský

Názov: Učenie sekvencií pomocou hierarchickej neurónovej siete s echo stavmi

Cieľ: 1. Naštudujte a implementujte model hierarchickej siete s echo stavmi, vhodnej na predikciu a dekompozíciu sekvenčných dát. 2. Odlaďte model na dátach uvedených v Jaeger (2007). 3. Testujte model na vybratých úlohách predikcie signálu (testovacie dáta budú špecifikované začiatkom zimného semestra 2010-2011).

Vedúci: doc. Ing. Igor Farkaš, PhD.

Dátum zadania: 18.12.2009

Dátum schválenia: 18.02.2011

prof. RNDr. Branislav Rovan, PhD.
garant študijného programu

.....
študent

.....
vedúci

ČESTNÉ VYHLÁSENIE

Čestne vyhlasujem, že som diplomovú prácu vypracoval samostatne, a že som uviedol všetku použitú literatúru súvisiacu so zameraním diplomovej práce.

Bratislava . . . . . .

. . . . . .

Viliam Dillinger

POĎAKOVANIE

Touto cestou vyslovujem poďakovanie doc. Ing. Igorovi Farkašovi, PhD. za pomoc, odborné vedenie, cenné rady a pripomienky pri vypracovaní mojej diplomovej práce.

Bratislava . . . .

Viliam Dillinger

Abstrakt

Mnoho časových radov praktického významu má príznaky v rôznych časových a priestorových merítkach. Jedným zo spôsobov, ako spracovávať takéto rady je použitie hierarchických neurónových sietí s echo stavmi. Cieľom tejto diplomovej práce bolo implementovať a otestovať tento model na dvoch úlohách – aproximácia prechodovej funkcie Reberovej gramatiky a predikcia v časovom rade Mackey-glass. Sieť bola implementovaná a jej zdrojový kód sa nachádza v prílohe. Na riešenie prvej úlohy sme použili trojvrstvú sieť a úspešne sme ju natrénovali. Pozorovali a popísali sme Aha! efekty, ktoré pri jej trénovaní nastali. Na riešenie druhej úlohy sme taktiež použili trojvrstvovú sieť. Podarilo sa nám dosiahnuť chybu porovnateľnu s niektorými inými prístupmi.

Kľúčové slová: hierarchická neurónová sieť s echo stavmi, učenie, sekvencie, časový rad Mackey-glass, Reberova gramatika

Abstract

Many time series of practical importance have features at different time and spatial scales. One way how to deal with it is to use a recently proposed hierarchical echo-state network. The goal of this diploma thesis was to implement and gain insight into functioning of this model. This was approached by testing this model on two tasks - approximation of transition functions of Reber grammar given by symbolic sequences and prediction of continuous Mackey-Glass time series. To solve the first task we have used a three-layer network and we have successfully trained it. We have observed and described Aha! effects, which occurred during training and represented significant shifts in the organization of model's feature detectors. To solve the second task we also used a three-layer network and managed to achieve the prediction error comparable to other approaches.

Key words: hierarchical echo-state network, learning, feature detectors, symbolic sequence (Reber grammar), continuous time series (Mackey-Glass)

Obsah

1 Úvod

Mnohé časové rady, s ktorými sa môžeme stretnúť v praxi, majú príznaky v rôznych časových a priestorových merítkach.

Ako príklad si zoberme analýzu hovorenej reči. Môžeme ju analyzovať v rozsahu milisekúnd (hlásky), sekúnd (slová), či dlhších úsekov (frázy).

Podobné to je aj pri systémoch rozoznávajúcich rukopis, ovládajúcich pohyb robota, či aplikácie analyzujúce videosekvencie. Prirodzená stratégia pri spracovaní takýchto dát je vytvoriť hierarchický systém, ktorého moduly sa budú špecializovať na rôzne časové škály. (Jaeger, 2007).

Mnoho systémov v umelej inteligencii, robotike, rozpoznávaní vzorov a strojovom učení používa túto stratégiu. Aj keď tieto systémy dokážu spracovať mnohorozmerné vstupy, stále sú veľmi daľeko od výkonnosti ľudského či zvieracieho mozgu. Príčiny, prečo tomu tak je, môžu byť:

- Väčšina umelých modelov bola navrhnutá pre statický vstup vzoriek, avšak biologické signály sú časové rady;
- Väčšina modelov pracuje dobre len na málorozmerných vstupných dátach, ale biologické dátové toky sú vysokorozmerné;
- Mnohým hierarchickým modelom chýba učiaci mechanizmus, čo je nutné pre každý systém, ktorý sa pokúša priblížiť k biologickým systémom;
- Mnoho modelov sa spolieha na dáta predpripravené človekom, definície príznakov, alebo iné priame zásahy od človeka, bez ktorých by model nefungoval;
- Algoritmy pre hierarchické učenie sú často veľmi pomalé a výpočtovo zložité;
- Najčastejšie používané algoritmy pre hierarchické systémy sú biologicky nedôveryhodné.

Hierarchické neurónové siete vymyslel Herbert Jaeger z Jacobs University v Bremene. Prvý článok o nich publikoval v júli 2007 [1].

Autor si určil tri hlavné ciele pri tvorbe tohto modelu:

1. Sieť má byť schopná pracovať s vysokorozmernými časovými dátami,
2. má dynamicky objavovať príznaky s minimálnou pomocou človeka a
3. aby jej učiaci algoritmus bol výpočtovo jednoduchý, štatisticky efektívny a biologicky dôveryhodný.

Podľa nás všetky tri ciele táto sieť splnila. Sieť dokáže pracovať aj s vysokorozmernými vstupmi, avšak treba dať pozor na veľkosť súradníc vstupného vektora a poprípade ich vhodne preškálovať. Sieť pri našich testoch sama dokázala objavovať príznaky. Výpočet jedného kroku touto sieťou je rýchejší ako pri sieti s echo stavmi s rovnakým počtom prepojení, ktoré sa učia.

Práca je rozdelená do piatich kapitol. Prvou je úvod, ktorý práve čítate.

V druhej kapitole sú opísané neurónové siete s echo stavmi. V rámci tejto kapitoly opisujeme základný perceptrón, jeho spôsob radenia do vrstiev a základný učiaci algoritmus.

V tretej kapitole sú opísane hierarchické neurónové siete s echo stavmi. V rámci tejto kapitoly opisujeme princíp fungovania tejto siete, formálny popis prechodu signálu sieťou a jej učiaci mechanizmus.

V štvrtej kapitole je opísaná naša implementácia. Táto kapitola obsahuje ukážku, ako používať túto implementáciu a podrobný popis premenných a metód jednotlivých tried.

V piatej kapitole sú popísane naše pokusy s hierarchickou neurónovou sieťou s echo stavmi na dvoch úlohách – Reberovej gramatike (zástupca symbolovej dynamiky) a Mackey-glass time series (zástupca spojitej dynamiky). V tejto časti uvádzame naše výsledky na daných úlohách a naše skúmanie učenia siete na oboch úlohách.

Piatou kapitiolou je záver.

2 Neurónové siete s echo stavmi

Medzi prvé siete, ktoré riešili problematiku v čase, patrili dopredné siete s oknom. Klasická dopredná neurónová sieť dokáže pracovať iba so statickými dátami. Sieť s oknom dostávala na vstup okrem aktuálneho vstupu aj vstup, ktorý dostala v čase $t-1$ až $t-n$. Vznikala však otázka, aké veľké musí byť dané n? Pre každý problém musel človek hodnotu n zadať, čo nebolo veľmi praktické a ani biologicky dôveryhodné.

Neskôr sa začali využívať rôzne rekurentné siete, ktoré pri učení používali spätné šírenie chyby. Problém, ktorý bol pri dopredných sieťach s oknom, vyriešili nasledovne. Na výpočet aktivácie niektorej vrstvy v čase t použijeme aktiváciu niektorej vrstvy v čase $t-1$. Stav siete takto závisí od jej všetkých predchádzajúcich stavov. Takýmto spôsobom vzniklo nekonečné okno.

Vznikol však problém so zložitosťou učiaceho mechanizmu, nakoľko spätné šírenie chyby je výpočtovo zložité a rozšíriteľnosť siete pri jeho použití je prakticky nemožná.

Neurónové siete s echo stavmi sú špeciálnym prípadom rekurentnej siete a v súčasnosti sú veľmi populárnym riešením na spracovanie dát v čase. Majú špecifickú architektúru, ktorú neskôr popíšem. Vďaka nej sa veľmi zjednodušilo učenie, pretože sa obmedzilo len na jednu vrstvu, vďaka čomu sa odbúrala nutnosť spätného šírenia chyby. Ich hlavnými výhodami sú:

- **Presnosť modelu** – Neurónové siete s echo stavmi prekonali predchádzajúce metódy nelineárnej identifikácie, predikcie a klasifikácie (víťaz medzinárodnej súťaže NN3 vo finančných predpovediach, atď.)
- **Rozšíriteľnosť a delenie** – Vďaka odstráneniu spätného šírenia chyby je rozšírenie výstupov veľmi jednoduché. Stačí daný vstup pridať a jeho trénovanie je nezávislé od ostatných. Podobne sa dá postupovať aj pri delení, kde stačí zduplikovať rezervoár siete a vybrané výstupy naňho napojiť. Katastrofický scenár je v oboch prípadoch nemožný.

- **Zložitosť algoritmu učenia** – siete s echo stavmi majú oproti viacerým modelom (najmä tým, ktoré používajú spätné šírenie chyby) obrovskú výhodu, nakoľko sa pri jej trénovaní mení len jedna matica prepojení.

V nasledujúcich troch podkapitolách sú popísané základy dopredných neurónových sietí – popis základného perceptrónu, popis základného učenia sa s učiteľom a základný popis vrstvy. Problematiky znalému čitateľovi odporúčame prejsť rovno na podkapitolu o neurónových sieťach s echo stavmi.

2.1 Jednoduchý perceptrón

Základnou stavebnou jednotkou neurónových sietí je neurón, ktorý je zjednodušenou verziou neurónov v mozgu. Na rozdiel od neurónov v mozgu abstrahujeme od chemických a fyzikálnych procesov v neuróne. Neuróny v mozgu vysielajú signály iným neurónom ako vlnenie, od čoho taktiež abstrahujeme.

V tejto práci sú použité neuróny typu perceptrón.

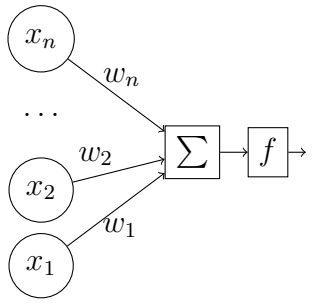
Formálne – nech má perceptrón n vstupov, x_i $(0 < i \leq n)$ je aktivácia i-teho vstupu s váhou w_i, potom aktiváciu y nášho perceptrónu vyrátame ako:

$$y = f(\sum_{i=1}^{n} x_i w_i) \tag{1}$$

kde $f(x)$ je spojitá monotónna funkcia. Najčastejšie sa používaju:

- $f(x) = x$
- hyperbolický tangens $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
- sigmoid $f(x) = \frac{1}{1+e^{-x}}$

Dôvod, prečo sa používajú je to, že každá z nich má "peknú" deriváciu, čo zjednodušuje vzorce pri učení. Zároveň hyperbolický tangens a sigmoid sú ohraničené (sigmoid $(0, 1)$, hyperbolický tangens $(-1, 1)$) čo nám značne zmenší obor hodnôt, v ktorých sa môže nachádzať aktivácia nášho neurónu (krajné stavy sa budú od seba len málo líšiť).



Obr. 1: Neurón - perceptrón

Výstup (aktivácia) perceptrónu je lineárnou kombináciou jeho vstupov. Ak chceme, aby dával výstupy podľa našich predstáv, musíme najprv správne nastaviť váhy prepojení (w_i). Toto docielime buď vyrátaním daných hodnôt, alebo učením. Poznáme viacero druhov učenia ako napríklad učenie bez učiteľa (unsupervised learning), učenie posilňovaním (reinforcement learning), a ďalšie. V tejto práci je použité učenie s učiteľom (supervised learning).

2.2 Učenie s učiteľom

Učenie popísané v tejto časti sa používa na trénovanie jednoduchého perceptrónu, neurónových sietí s echo stavmi a iných modelov. Rovnaký princíp sa využíva aj pri učení hierarchických neurónových sietí s echo stavmi.

Formálne – majme množinu trénovacích dát a požadovaných výstupov Z, $p \in Z$, $p = (\mathbf{x}, d)$, $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. Našim cielom pri učení bude minimalizovať kvadratickú chybu E:

$$e(p) = \frac{1}{2}(d_p - y_p)^2$$

$$E(Z) = \sum_{p \in \mathbf{Z}} e(p)$$

Učenie sa realizuje upravovaním váh prepojení. Váhu w_i v čase $t + 1$ upravíme nasledovne:

$$w_{i,t+1} = w_{i,t} + \alpha(d_p - y_p)f'(x_i)x_i$$

kde $\alpha \in (0, 1)$ je rýchlosť učenia. Trénovanie siete prebieha podľa nasledovného postupu:

1. Inicializujte všetky váhové vektory na náhodnú hodnotu (používajú sa čísla blízke 0, napr z intervalu $(-1, 1)$),
2. nastavte E=0,
3. vyberte p, zo vstupu $\mathbf{x}$ vypočítajte y,
4. vyrátajte $e(p)$, E=E+e(p),
5. upravte váhy podľa učiaceho pravidla,
6. ak neboli použité všetky p zo Z, tak prejdite na 3.
7. ak $E < \varepsilon$, tak trénovanie ukončite, inak náhodne pomiešajte poradie p a prejdite na 2.

Pri tomto učení nepožadujeme, aby sa $E = 0$. Mnohokrát to nie je možné (nekonzistentné trénovacie dáta) a dosahujú sa pri tom lepšie výsledky.

Množinu dát, ktoré používame na trénovanie, permutujeme z toho dôvodu, aby sa váhy nemenili v stále rovnakých cykloch (a teda by bolo učenie neúspešné).

2.3 Viacvrstvová dopredná sieť

Vrstva je usporiadaná n-tica neurónov. Jej aktivácia je vektor $\mathbf{y} = (y_1, y_2, \ldots, y_n)$. Pri prepojení dvoch vrstiev hovoríme o prepojení každého neurónu z jednej vrstvy s každým neurónom z druhej vrstvy (úplné párovanie). Váhovou maticou takéhoto prepojenia rozumieme maticu $\mathbf{W}$:

$$\mathbf{W} = \begin{pmatrix} w_{11} & w_{12} & \ldots & w_{1j} \\ w_{21} & w_{22} & \ldots & w_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ w_{i1} & w_{i2} & \ldots & w_{ij} \end{pmatrix}$$

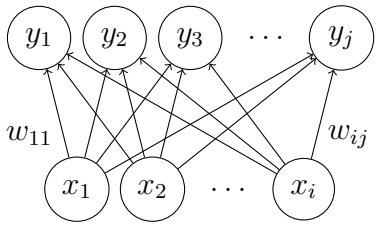
kde $w_{i,j}$ je váha prepojenia i-teho neurónu prvej vrstvy a j-teho neurónu druhej vrstvy.

Nech sú vrstvy $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}$ prepojené s vrstou $\mathbf{y}$ prepojeniami $\mathbf{W_1}, \mathbf{W_2}, \ldots, \mathbf{W_n}$, potom aktiváciu $\mathbf{y}$ vyrátame ako

$$\mathbf{y} = f(\sum_{i=1}^{n} \mathbf{x_i}\mathbf{W_i})$$

Funkcia f (spomenutá aj v časti perceptrón) je skalárna a aplikuje sa na každý prvok vektora zvlášť.

Rovnako ako výsledok aktivačnej funkcie je vektor, takisto aj chybová funkcia je vektor (snažíme sa minimalizovať $|\mathbf{E}|$), učenie sa však nezmení, pričom ako chybu pri učení j-teho neurónu použijeme j-tu súradnicu chybového vektora.



Obr. 2: Prepojenie dvoch vrstiev

Pomocou skladania rôznych vrstiev nad seba a ich prepájaním dokážeme

vstup aj nelineárne transformovať na výstup. Na učenie prepojení nižších vrstiev sa najčastejšie používa spätné šírenie chyby, ktoré ale nebudeme popisovať, nakoľko nie je v tejto práci využité.

2.4 Neurónová sieť s echo stavmi

Neurónová sieť s echo stavmi patrí medzi rekurentné siete. Má 1 skrytú vrstvu – rezervoár ($\mathbf{v_{res}}$), ktorý je vysokorozmerný. Vstupnú vrstvu budeme označovať $\mathbf{v_{in}}$ a výstupnú vrstvu $\mathbf{v_{out}}$. Vstupná vrstva je s rezervoárom prepojená pomocou $\mathbf{W_{in}}$, rezervoár s výstupnou vrstvou pomocou $\mathbf{W_{out}}$ a rezervoár je prepojený sám so sebou pomocou $\mathbf{W_{res}}$ (rekurentné prepojenie). Všetky prepojenia sa inicializujú náhodne na hodnoty medzi -1 a +1. Okrem toho $\mathbf{W_{res}}$ musí mať spektrálny rádius menší ako 1 a je riedke (väčšina súradníc je rovná 0).

To docielime tým, že po náhodnej inicializácii vynulujeme väčšinu (napr. 90%) prvkov, vypočítame veľkosť najväčšieho vlastného čísla a každú súradnicu ním predelíme a vynásobíme číslom menším ako 1. Takéto prepojenie nám bude signál v sieti tlmiť.

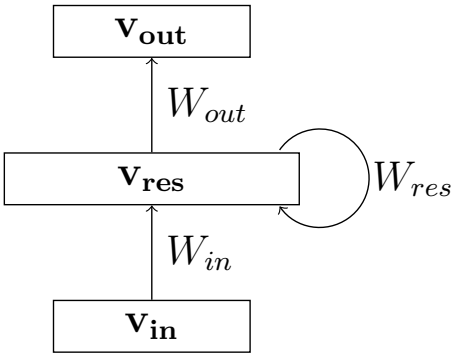
Aktivácia $\mathbf{v_{res_t}}$ v čase t sa vyráta ako

$$\mathbf{v_{res_t}} = \sigma(\mathbf{v_{in}}\mathbf{W_{in}} + \mathbf{v_{res_{t-1}}}\mathbf{W_{res}})$$

kde $\sigma(x)$ je najčastejšie hyperbolický tangens, sigmoid, alebo iná funkcia popísaná v časti perceptrón . Aktivácia výstupnej vrstvy sa vypočíta ako

$$\mathbf{v_{out}} = \mathbf{v_{res_t}}\mathbf{W_{out}}$$

Pri trénovaní siete sa menia váhy iba výstupnej matice $\mathbf{W_{out}}$.



Obr. 3: Neurónová sieť s echo stavmi

3 Hierarchická neurónová sieť s echo stavmi

Ústrednou myšlienkou dizajnu tejto siete je, ako spojiť príznaky, ktoré su extrahované na každej vrstve tak, aby spoločne spracovali komplexný viacškálový vstup (Jaeger, 2007).

Hierarchická neurónová sieť s echo stavmi pozostáva z n vrstiev (levels), z ktorých každá použije výstup nižšej vrstvy v čase $t - 1$ ako svoj vstup (v prípade prvej sa berú vstupné dáta), ten transformuje na príznaky, ktoré za pomoci výstupu vyššej vrstvy (hlasov) spojí do výstupu tejto vrstvy (v prípade najvyššej vrstvy je len jeden príznak a ten sa dáva na výstup). Výstupom siete bude výstup prvej vrstvy.

Príznaky (výstupy sietí) na prvej vrstve sa teda učia správny výstup. Príznaky na druhej vrstve sa učia, ako najlepšie poskladať výsledok z príznakov na prvej vrstve. Príznaky na tretej vrstve sa učia, ako najlepšie poskladať príznaky na druhej vrstve, aby čo najlepšie poskladali príznaky na prvej vrstve, a tak ďalej.

Autor takto využil siete s echo stavmi na jednotlivých úrovniach ako špecialistov na rôzne veľké časové škály a vhodne ich spolu hierarchicky poprepájal.

Nakoľko siete na každej vrstve dostávajú rovnaký vstup a prepojenia vstupu s rezervoárom a rekurentné spojenie rezervoára netrénujeme, autor vo svojej architektúre využil jeden spoločný rezervoár pre všetky príznaky v rámci jednej vrstvy. Aby docielil rôznorodosť sietí na jednej vrstve, každá sieť "vidí" len časť spoločného rezervoára.

Autor hierarchických neurónových sietí vo svojej práci [1] ponúka porovnanie výsledkov hierarchickej neurónovej siete s echo stavmi a sieťou s echo stavmi s rovnakou výpočtovou zložitosťou na rovnakých problémoch.

Pri nehierarchickom signáli zistil, že obe siete mali rovnakú úspešnosť. Avšak na hierarchickom probléme mala hierarchická neurónová sieť s echo stavmi menšiu chybovosť ako obyčajná sieť s echo stavmi. Podobné porovnanie sme robili aj my v našich pokusoch na časovom rade Mackey-Glass s rovnakým výsledkom.

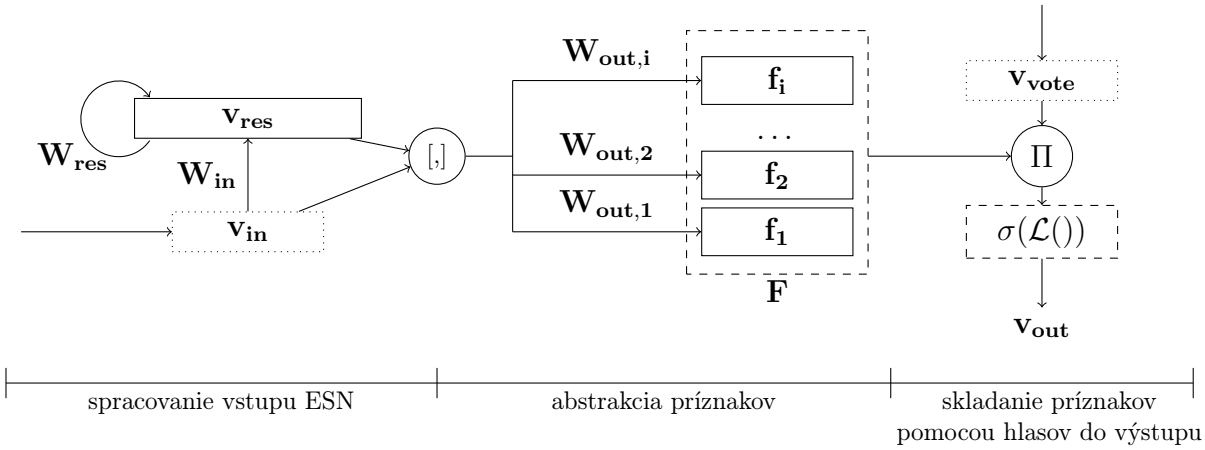
3.1 Architektúra a aktivačná dynamika

Vstupný vektor budeme označovať $\mathbf{v_{in}}$. V prvej vrstve to bude vstup siete, v ďalších to bude výstup vrstvy o jednu nižšej v čase $t-1$.

Aktiváciu rezervoára v čase t budeme označovať $\mathbf{v_{res,t}}$, prepojenie vstupu a rezervoára $\mathbf{W_{in}}$ a rekurentné prepojenie rezervoára $\mathbf{W_{res}}$. Aktiváciu rezervoára vypočítame ako:

$$\mathbf{v_{res,t}} = (1-a)\mathbf{v_{res,t-1}} + \sigma(\mathbf{v_{res,t-1}}\mathbf{W_{res}} + \mathbf{v_{in}}\mathbf{W_{in}})$$

kde σ je štandardný sigmoid aplikovaný po prvkoch a a je leaky rate, ktorý určuje v akom pomere sa použije predchádzajúci a aktuálny stav.



Obr. 4: Schéma vrstvy HESN

Následne sa z rezervoára abstrahujú príznaky. Nech má naša vrstva j príznakov $\mathbf{f_1}, \mathbf{f_2}, \ldots, \mathbf{f_j}$, potom pre každý máme maticu prepojení $\mathbf{W_{out,i}}$, $(0 < i \leq j)$. Ich aktiváciu vypočítame ako:

$$\mathbf{f_i} = [\mathbf{v_{res}}; \mathbf{v_{in}}]\mathbf{W_{out,i}}$$

kde $[\mathbf{u}; \mathbf{v}]$ je vektorové zreťazenie ($[\mathbf{u}; \mathbf{v}] = (u_1, \ldots . u_m, v_1, \ldots, v_n)$).

Jednotlivé príznaky zoradíme pod seba do matice $\mathbf{F}$, v ktorej x-ty riadok

je x-ty príznak. Výsledný výstup vrstvy ($\mathbf{v_{out}}$) vypočítame ako :

$$\mathbf{v_{out}} = \sigma(\mathcal{L}(\mathbf{v_{vote}}\mathbf{F}))$$

kde $\mathcal{L}(x_t)$ je leaky integration definované ako :

$$\mathcal{L}(x_t) = (1 - \lambda)\mathcal{L}(x_{t-1}) + \lambda x_t$$

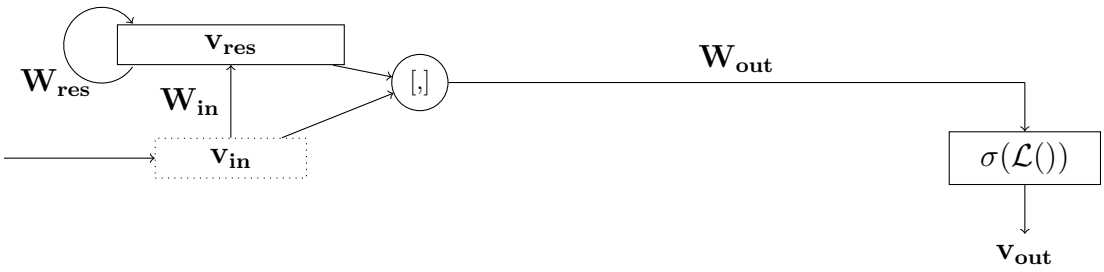
pričom λ je leaking rate.

Pri výpočte výstupu prvej vrstvy (a teda celej siete) neaplikujeme ani sigmoid, ani leaky integration, teda použijeme len

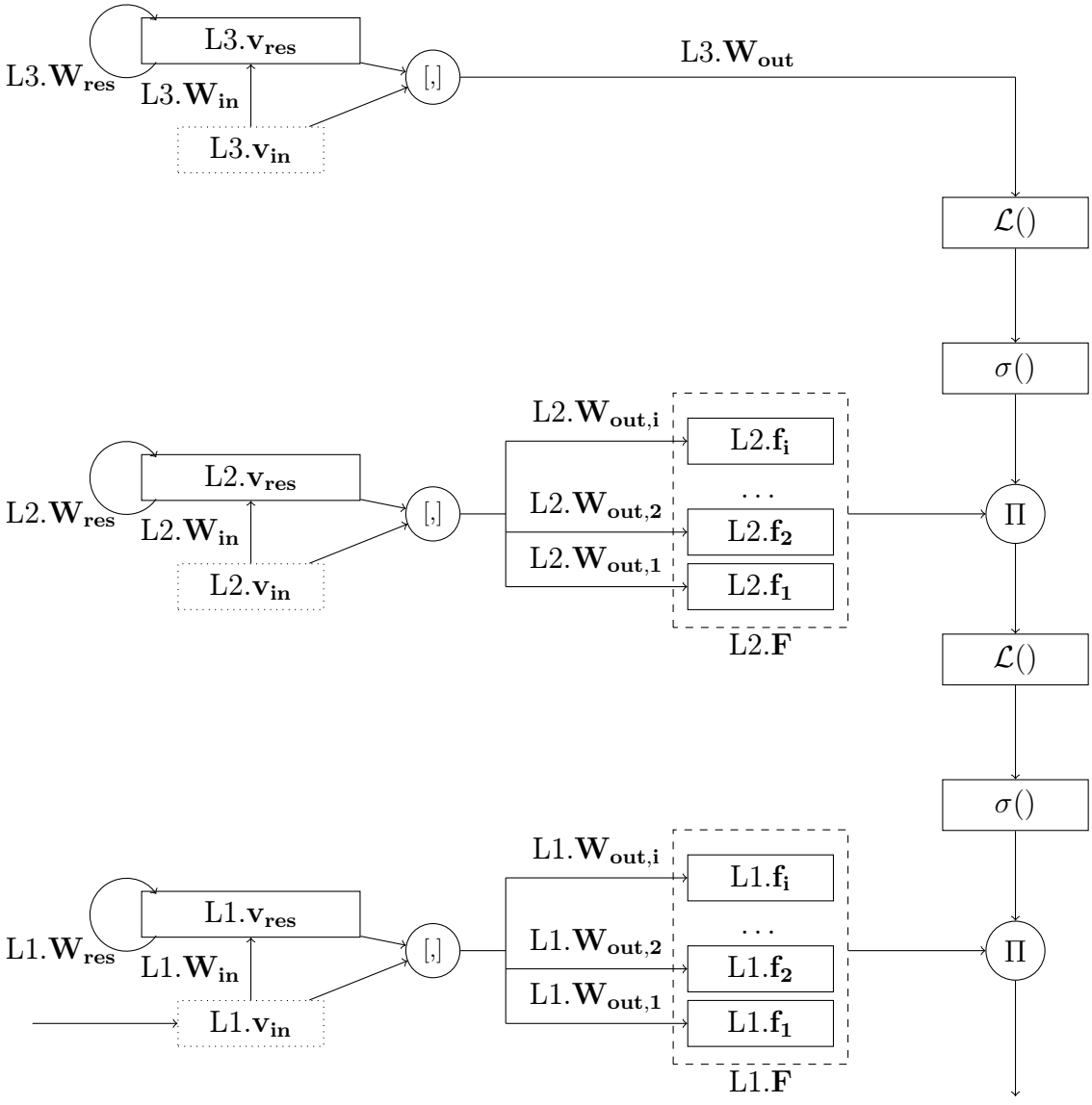
$$\mathbf{v_{out}} = \mathbf{v_{vote}}\mathbf{F}$$

Ďalšou výnimkou je najvyššia vrstva. Má práve jeden príznak, ktorý priamo používame ako výstup vrstvy, teda:

$$\mathbf{v_{out}} = \sigma(\mathcal{L}([\mathbf{v_{res}}; \mathbf{v_{in}}]\mathbf{W_{out}}))$$



Obr. 5: Schéma najvyššej vrstvy HESN



Obr. 6: Ukážka trojvrstvej HESN

3.2 Mechanizmus učenia

Pri učení hierarchických neurónových sietí s echo stavmi sa (podobne, ako pri učení sietí s echo stavmi) upravujú iba prepojenia vedúce von z rezervoára, teda na každej úrovni to sú matice $\mathbf{W_{out,1}}, \mathbf{W_{out,2}}, \ldots, \mathbf{W_{out,i}}$. Tieto matice sa upravujú po každom kroku s použitím gradient descent a kvadratickú predikčnú chybu.

Najskôr definujeme takzvaný "volebný potenciál" každej (okrem najnižšej) úrovne. Je to výstup z danej vrstvy, na ktorý bolo aplikované leaky integration, avšak nebol aplikovaný sigmoid.

$$\mathbf{p} = \mathcal{L}(\mathbf{v_{vote}F})$$

Pre každú úroveň taktiež definujeme chybový vektor ($\mathbf{E_k}$ je chybovým vektorom k-tej úrovne, $\mathbf{F_k}^T$ je matica príznakov na k-tej úrovni a $\mathbf{p_k}$ je volebný potenciál k-tej úrovne, $\bar{\mathbf{v}}_{\mathbf{out}}$ je správny výstup):

$$\mathbf{E_1} = \mathbf{v_{out}} - \bar{\mathbf{v}}_{\mathbf{out}}$$

$$\mathbf{E_{k+1}} = \mathbf{E_k}\mathbf{F_k}^T * \mathcal{L}(\sigma'(\mathbf{p_k}))$$

kde $*$ je násobenie po súradniciach dvoch vektorov a $\sigma'(x)$ je derivácia sigmoid-u:

$$\sigma'(x) = \sigma(x)(1 - \sigma(x))$$

Pomocou hore uvedených vzťahov dokážeme postupne (rekurzívne) vypočítať chybu na každej úrovni. Pomocou nej upravíme výstupy z rezervoára nasledovne:

$$\mathbf{W_{out,i}}(t+1) = \mathbf{W_{out,i}}(t) + \gamma v_{vote,i}[\mathbf{v_{res}}; \mathbf{v_{in}}]^T\mathbf{E_k}$$

kde $\gamma \in (0, 1)$ je rýchlosť učenia.

4 Implementácia

Všetky pokusy sme robili pomocou nami implementovanej hierarchickej neurónovej siete s echo stavmi. Implementovaná bola v jazyku C++. Pri jej implementácii sme použili dve externé matematické knižnice:

- Newmat – Knižnica implementuje matice, vektory a matematické operácie na nich. V licenčných podmienkach je dovolené používať ju na čokoľvek, avšak pri jej modifikácii je nutné jasne uviesť, ktoré časti boli modifikované a ktoré sú pôvodné.
- GSL (GNU science library) – Pomocou tejto knižnice boli počítané najväčšie vlastné čísla, ktoré sa využívali pri generovaní rekurzívnej matice v rezervoároch. Bolo nutné použiť túto knižnicu, nakoľko knižnica Newmat dokázala vyrátať vlastné čísla iba symetrickej matice. Táto knižnica je šírená pod GNU General Public License.

Sieť sa vytvára veľmi jednoducho – vytvorí sa objekt typu hesn, čo zároveň vytvorí prvú vrstvu siete. Ďalšie vrstvy je možné pridávať volaním `addLayer()`. Treba však dodržať to, aby na najvyššej vrstve bola práve jedna sieť.

Príklad vytvorenia trojvrstvovej siete s ôsmimi príznakmi na prvej vrstve, troma príznakmi na druhej vrstve a jedným na tretej:

```
hesn siet(6,6,8,50,0.005,0.9,0.0,0.3,15);
siet.addLayer(80,3,0.005,0.8,0.05,0.4,15);
siet.addLayer(20,1,0.005,1.0,0.1,0.4,20);
```

Na prechod signálu sieťou používame metódu `Next(RowVector in)`, ktorá dá na svoj výstup výstup celej siete. Na trénovanie sa používa metóda `Train(RowVector error)`.

Napríklad ak by sme mali v poli numbers časový rad a sieť by sme chceli naučiť jednokrokovú predikciu, použili by sme tento kód:

```
for (int i=0; i<velkost-2; i++) {
   vstup = numbers[i];
   RowVector out = siet.Next(vstup);

   target = numbers[i+1];
   RowVector error = target - out;
   siet.Train(error);
}
```

Podrobný popis obidvoch tried uvádzame v nasledujúcich častiach.

4.1 Trieda layer

Táto trieda implementuje jednu vrstvu siete. Pri jej vytvorení sa zároveň inicializujú všetky matice prechodov presne podľa kapitoly 3. Jej premenné sú:

- `Matrix *inputTOres` – matica prepojení vstupnej vrstvy a rezervoára
- `Matrix *resTOres` – matica rekurentného prepojenia rezervoáru
- `vector<Matrix> *resTOfeature` – sada matíc prepojení vektorovo spojeného rezervoára s vstupom s daným príznakom
- `RowVector *input` – posledný vstup vrstvy (zapamätané kvôli následnej analýze siete)
- `RowVector *res` – aktuálny stav rezervoáru
- `Matrix *features` – naposledy abstrahované príznaky, jeden riadok matice za každý príznak (zapamätané kvôli následnej analýze siete)
- `RowVector *fromup` – posledný výstup vyššej vrstvy (zapamätané kvôli následnej analýze siete)
- `RowVector *votepower` – posledný volebný potenciál
- `RowVector *votes` – posledné hlasy (zapamätané kvôli následnej analýze siete)
- `RowVector *inandres` – posledný vstup vektorovo spojený s aktuálnym rezervoárom (kvôli ladeniu a analýze siete)
- `RowVector *out` – posledný výstup vrstvy
- `bool type` – indikátor najvyššej vrstvy (false pre všetky ostatné)
- `double learningRate` – rýchlosť učenia
- `double leakyRate` – leaky rate používaný pri aplikácii leaky integration na hlasy

- `double leakyNeuron` – leaky rate používaný pri aplikácii leaky integration na rezervoár
- `int numberOfFeatures` – počet príznakov
- `int numberOfOutput` – veľkosť výstupu

Trieda má nasledovné metódy:

- `layer(int num_feature, int num_input, int num_output, int num_res, bool t, double alpha, double leaky,double leakyn, double spectral, int n_resTOout)`
  - konštruktor, ktorý inicializuje všetky premenné a nastaví matice prepojení podľa kapitoly 3. Jeho vstupy sú:
    - num_feature – počet príznakov
    - num_input – veľkosť vstupného vektora
    - num_output – veľkosť výstupného vektora
    - num_res – veľkosť rezervoára
    - t – indikátor najvyššej vrstvy
    - alpha – rýchlosť učenia
    - leaky – leaky rate používaný pri aplikácii leaky integration na hlasy
    - leakyn – leaky rate používaný pri aplikácii leaky integration na rezervoár
    - spectral – spektrálny rádius matice rekurentného spojenia rezervoára
    - n_resTOout – počet neurónov z rezervoára, ktoré "vidia" príznaky
- `layer(const layer &povodna)` – kopírovací konštruktor
- ∼`layer()` – deštruktor

- `void ResetReservoair()` – nastavenie rezervoára a výstupného vektora na náhodné malé hodnoty (-0.01 až 0.01)
- `RowVector Calculate(const RowVector& input, const RowVector& vote)` – vypočíta výstup vrstvy s vstupom input a hlasmi z vyššej vrstvy vote
- `RowVector Train(const RowVector& error)` – aplikovanie učiaceho mechanizmu s chybou error, na výstup dá chybový vektor pre vyššiu vrstvu.
- `void Test()` – vypíše všetky premenné na štandardný výstup

4.2 Trieda hesn

Táto trieda skladá jednotlivé vrstvy dohromady a rieši interakcie medzi nimi ako pri prechode signálu, tak pri učení. Jej premenné sú:

- `vector<layer> *vrstvy` – vrstvy siete zoradené zdola nahor
- `int top_num_feature` – počet príznakov v najvyššej vrstve (využíva sa pri pridávaní ďalšej vrstvy)
- `int top_num_output` – veľkosť výstupného vektora v najvyššej vrstve (využíva sa pri pridávaní ďalšej vrstvy)

Trieda má nasledovné metódy:

- `hesn(int n_input, int n_output, int n_feature, int n_res,`
- `double alpha, double leaky, double leakyn, double spectral,`
- `int n_resTOout)`
  - Konštruktor, ktorý zároveň vytvorí prvú vrstvu siete. Jeho vstupy sú:
    - int n_input – veľkosť vstupného vektora siete
    - int n_output – veľkosť výstupného vektora siete
    - int n_feature – počet príznakov prvej vrstvy
    - int n_res – veľkosť rezervoára prvej vrstvy
    - double alpha – rýchlosť učenia prvej vrstvy
    - double leaky – leaky rate používaný pri aplikácii leaky integration na hlasoch prvej vrstvy
    - double leakyn – leaky rate používaný pri aplikácii leaky integration na rezervoári prvej vrstvy
    - double spectral – spektrálny rádius rezervoára prvej vrstvy
    - int n_resTOout – počet neurónov, ktoré "vidia" jednotlivé príznaky na prvej vrstve

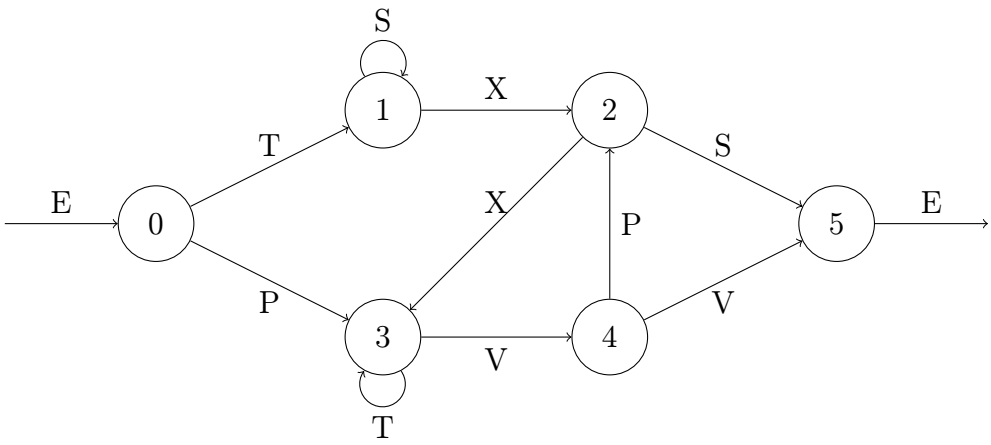
- `void addLayer(int n_res, int n_feature, double alpha, double leaky,`
- `double leakyn, double spectral, int n_resTOout)`
  - metóda pridá ďalšiu vrstvu do siete. Popis jej vstupov je rovnaký ako pri hore uvedenom konštruktore.
- `~hesn()` – štandardný deštruktor
- `RowVector Next(const RowVector& input)` – vypočíta a na výstup dá výstup celej siete za použitia vstupu input
- `void Train(const RowVector& error)` – trénovanie celej siete s chybovým vektorom
- `void ResetReservoairs()` – zavolá `ResetReservoair()` na všetkých vrstvách
- `void Test()` – zavolá `Test()` na všetkých vrstvách

5 Experimenty

5.1 Reberova gramatika

5.1.1 Popis

Ako zástupcu symbolovej dynamiky sme použili mierne upravenú Reberovu gramatiku. Táto gramatika je generovaná nasledovným automatom:



Obr. 7: Automat generujúci slová z Reberovej gramatiky

V každom stave má rovnakú pravdepodobnosť pre všetky možné prechody. Danú gramatiku sme si mierne upravili (pôvodná dostáva na začiatku B). Táto zmena úlohu nezjednodušila.

Symbol E v tejto gramatike symbolizuje akceptáciu (ukončenie) slova.

5.1.2 Trénovanie

Sieť dostávala na vstup symboly pomocou takzvaného one-hot kódovania a rovnaké kódovanie bolo použité aj pri výstupe.

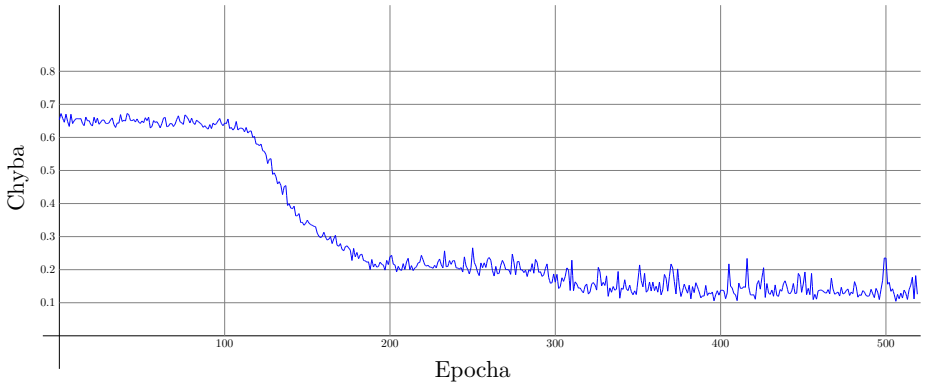
Zoberme si množinu písmen (entít) veľkosti n. One-hot je kódovanie, v ktorom ku každému písmenu (entite) je priradená jedna unikátna súradnica n-rozmerného vektora. Vektor reprezentujúci toto písmeno (entitu) bude mať všetky súradnice nulové okrem súradnice priradenej k danému písmenu, ktorá bude rovná jednej.

Pri trénovaní sieť dostávala na vstup aktuálny znak a mala predikovať (predpovedať) nasledujúci symbol. Následne sme zobrali symbol, ktorý reálne prišiel, a ten sme použili ako target pri trénovaní.

Sieť po natrénovaní týmto spôsobom aproximovala prechodovú funkciu – vo výstupnom vektore siete každá súradnica reprezentovala pravdepodobnosť, že bude nasledovať písmeno k nej priradené.

Sieť sme trénovali v niekoľkých epochách. Každá epocha mala 2 časti – trénovaciu a testovaciu. Počas testovacej sa neaplikoval učiaci mechanizmus, avšak zaznamenávala sa chyba aproximácie.

Ako chybu aproximácie sme použili uhol medzi predikovanou a reálnou prechodovou funkciou v danom stave automatu. Chyba epochy bola NRMSE (Normalized root mean square error) chýb počas testovacej epochy.



Obr. 8: Vývoj chyby počas trénovania siete – Reberova gramatika

Na obrázku 8 ukazujeme vývoj tejto chyby počas trénovania trojvrstvej siete s týmito parametrami:

- Na prvej vrstve mala sieť 5 príznakov, veľkosť rezervoára 50, každá sieť videla 15 súradníc rezervoára, spektrálny rádius 0.3, leaky rate na hlasoch 0.9 a leaky rate na rezervoári 0.
- Na druhej vrstve mala sieť 3 príznakov, veľkosť rezervoára 80, každá sieť videla 15 súradníc rezervoára, spektrálny rádius 0.4, leaky rate na hlasoch 0.8 a leaky rate na rezervoári 0.05.

- Na tretej vrstve mala sieť veľkosť rezervoára 20, spektrálny rádius 0.4 a leaky rate na rezervoári 0.1.

Na všetkých vrstvách bola rýchlosť učenia 0.005 a jedna trénovacia epizóda pozostávala z 1000 predikcií.

Po trénovaní tejto siete mala chybovosť približne 0.12. Sieť aproximovala funkciu veľmi presne (rozdiel oproti reálnej prechodovej funkcii na každom písmenku bola maximálne 0.05), avšak ak sa príliš dlho "točila" v stave 1 (slovo začalo na 'T' a za ním nasledovalo niekoľko krát 'S'), niekedy nevedelo aproximovať stav pri prechode do stavu 2 (všetky výstupy okrem 'E' ohodnotilo na približne 0.25).

5.1.3 Aha! efekt

Vo svojej práci Jaeger (2007) opísal pri učení tohto modelu takzvaný Aha! efekt, pri ktorom mala sieť dlhodobo približne rovnakú chybu, ktorá sa náhle rapídne zmenšila a zasa sa ustálila.

Pri vývoji chyby na obrázku 8 je možné si všimnúť 2 miesta, v ktorých nastal Aha! efekt – okolo 120tej epochy a okolo epochy číslo 300.

Rozhodoli sme sa analyzovať, čo sa udialo medzi jednotlivými Aha! efektmi – konkrétne v epoche 50, 200 a 500 sme sledovali správanie sa jednotlivých príznakov v modeli.

Po testovacej časti hore menovaných epoch sme sieť nechali spraviť 1000 ďalších predikcií, pri ktorých sme zaznamenávali aktuálny stav automatu generujúceho jazyk a všetky príznaky na jednotlivých úrovniach. Ku každej dvojici stav – príznak sme vyrátali priemerný vektor daného príznaku a jeho rozptyl a zaniesli sme ho do obrázkov 9, 10 a 11.

Po 50-tej epoche bol stav siete nasledovný:

- Velkosť všetkých príznakov bola veľmi malá (absolutná hodnota súradníc všetkých príznakov menšia ako 0.15).

- Na prvej vrstve sieť reagovala na stavy automatu len veľmi mierne. Všetky reagovali úplne rovnako – pozorovali sme spoločné myslenie (hive mind)

- Na druhej vrstve veľmi jemne reagovala na akceptačný stav.
- Na tretej vrstve nereagovala vôbec a na výstup dávala stále takmer ten istý vektor (drobné rozdiely až na štvrtej cifre za desatinnou čiarkou).
- Rozptyl bol takmer nulový (¡0.01), teda príznaky reagovali v podstate rovnako na jednotlivé stavy.

Po 200-tej epoche bol stav siete nasledovný:

- Veľkosť všetkých príznakov sa zväčšila, najviac (až 100–násobne) na najvyššej vrstve.
- Na prvej vrstve príznaky výrazne reagovali na jednotlivé stavy. Zmena oproti stavu po 50-tej epoche bola veľmi výrazná.
- Na druhej vrstve taktiež začali príznaky výraznejšie reagovať na stav, avšak každý z nich reagoval takmer rovnako – pozorovali sme spoločné myslenie (hive mind).
- Na tretej vrstve začal príznak veľmi mierne reagovať na stav (menej než prvá vrstva po 50-tej epoche). Práve toto bolo dôsledkom spoločného myslenia na druhej úrovni.
- Rozptyl sa zväčšil, avšak stále ostával malý. Príznaky teda stále reagovali približne rovnako na jednotlivé stavy.

Po 500-tej epoche (na konci trénovania) bol stav siete nasledovný:

- Veľkosť príznakov na prvej a tretej úrovni sa nezmenila, avšak príznaky na druhej úrovni sa veľmi výrazne zväčšili (niektoré až 50–násobne)
- Na prvej vrstve sa reakcie príznakov na stavy zmenili len veľmi mierne (jemné doladenie).
- Na druhej vrstve príznaky začali na stavy reagovať rôzne, teda spoločné myslenie zmizlo, aj keď sa správali veľmi podobne (predtým sa správali totožne).

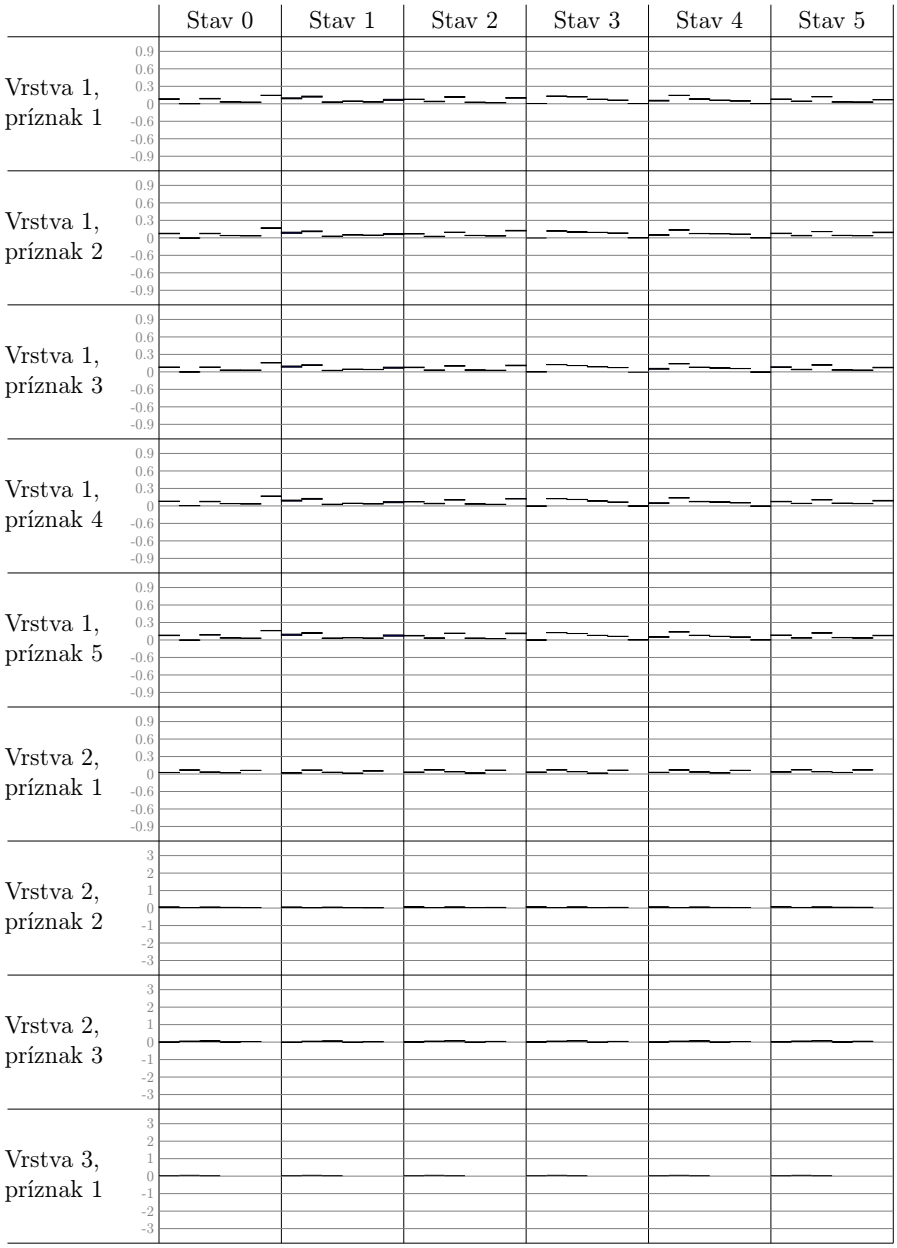
- Rozptyl na prvej vrstve ostal približne rovnaký a na druhej vrstve sa zväčšil, avšak pomer veľkosť priemeru/ veľkosť rozptylu ostal približne rovnaký.

- Na tretej vrstve nastali veľké zmeny. Rozptyl rapídne stúpol, z čoho je vidno, že príznak prestal byť približne rovnaký, začal na niečo reagovať, avšak nie na stav automatu. Jediný stav pri ktorom mal príznak rozptyl menší bol akceptačný stav. Z toho vyvodzujeme, že tento príznak reagoval na koniec slova (akceptačný stav).

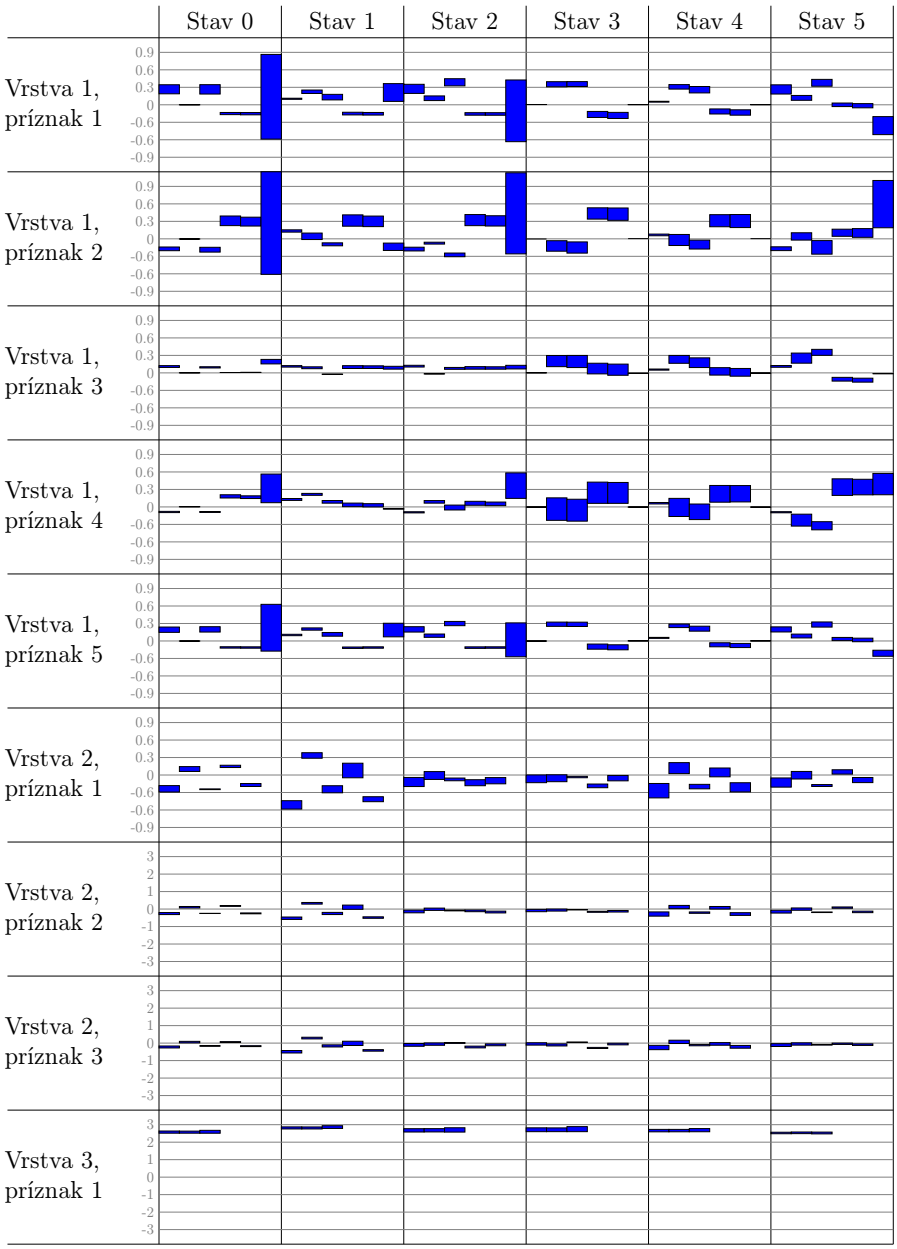
Z týchto poznatkov teda usudzujeme, že pred prvým Aha! efektom sa prvá vrstva začala učiť, zatiaľ čo druhá a tretia ostávali nezmenené. Počas prvého Aha! efektu sa začala učiť druhá vrstva, ktorá nadobudla spoločné myslenie (hive mind), pri ktorom boli príznaky rovnaké. Počas druhého Aha! efektu sa porušilo spoločné myslenie druhej vrstvy a kvôli tomu sa začala učiť tretia vrstva.

Prvá a druhá vrstva v natrénovanej sieti viditeľne reagovala na stav automatu, zatiaľ čo tretia vrstva reagovala na koniec slova.

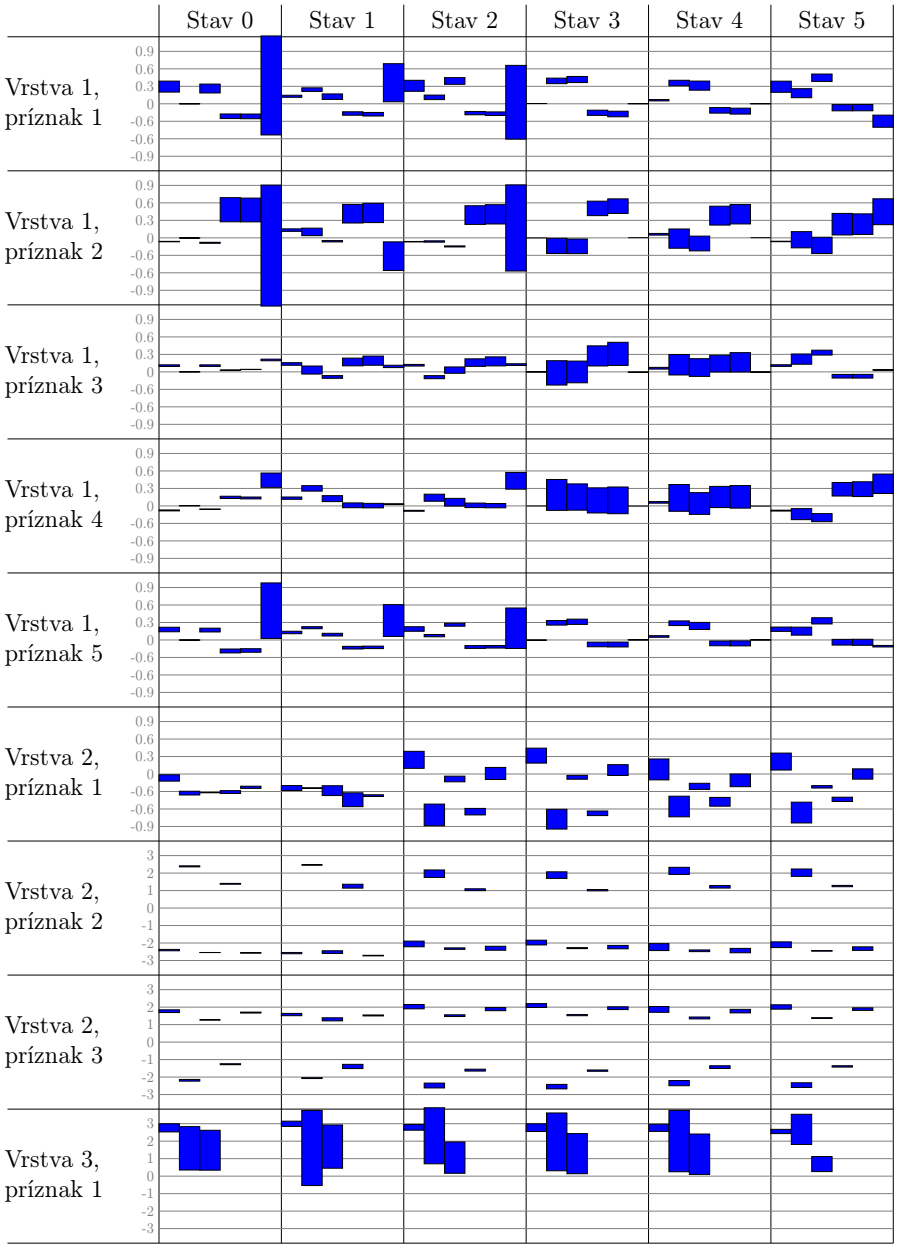
S týmito poznatkami sme spravili rovnaký pokus s dvojvrstvovou sieťou. Pri jej trénovaní nastal iba jeden Aha! efekt a jej chyba sa ustálila na približne rovnakej úrovni, akú mala hore popísaná trojvrstvová sieť medzi prvým a druhým Aha! efektom.



Obr. 9: Reakcie príznakov na stavy automatu po 50-tej epoche



Obr. 10: Reakcie príznakov na stavy automatu po 200-stej epoche



Obr. 11: Reakcie príznakov na stavy automatu po 500-stej epoche

5.2 Časový rad Mackey-Glass

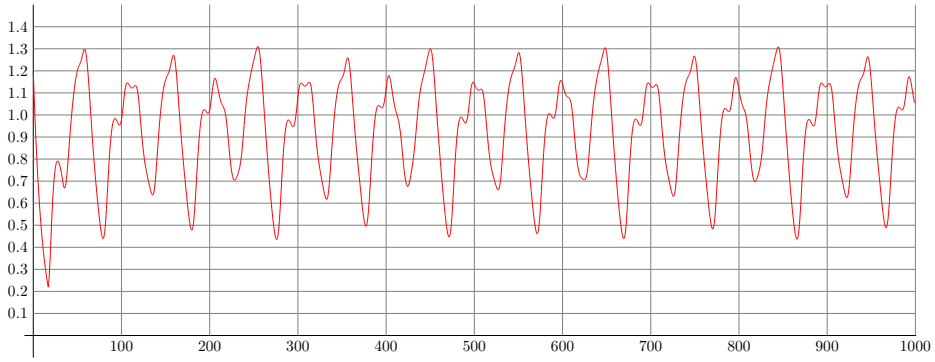
5.2.1 Popis

Ako zástupcu spojitej dynamiky sme použili časový rad Mackey-Glass. Táto funkcia modeluje dynamiku tvorby bielych krviniek u človeka. Je definovaná touto diferenciálnou rovnicou:

$$\frac{ds(t)}{dt} = \alpha \frac{s(t-\tau)}{1+s^{10}(t-\tau)} - \beta s(t) \tag{2}$$

Pri našich pokusoch sme používali tieto hodnoty pre dané premenné (tieto dáta sú súčasťou Matlabu v súbore mgdata.dat)

- $\alpha = 0.2$
- $\beta = 0.1$
- $s(0) = 1.2$ (pre $t < 0$ $s(t) = 0$)
- $\tau = 17$



Obr. 12: Časový rad Mackey-glass

5.2.2 Trénovanie

Sieť dostávala na vstup vektor $[s(t-18), s(t-12), s(t-6), s(t)]$ a mala sa naučiť predikovať $s(t+85)$.

Boli použité dva časové rady – trénovací a testovací. Počas žiadneho prechodu testovacieho radu sa sieť neučila. Testovací rad slúži na ohodnotenie, ako dokáže sieť generalizovať. V našom prípade sme používali prvých 10000 čísel z hore uvedenej postupnosti ako trénovací rad a nasledujúcich 500 ako rad testovací.

Na ohodnotenie správnosti fungovania siete sme použili Normalized root mean square error (NRMSE) testovacieho radu.

$$NRMSE = \sqrt{\frac{\sum_{k=1}^{n}(y_k - \hat{y}_k)^2}{\sum_{k=1}^{n}(y_k - \bar{y}_k)^2}} \tag{3}$$

$$\bar{y} = \frac{\sum_{k=1}^{n} y_k}{n} \tag{4}$$

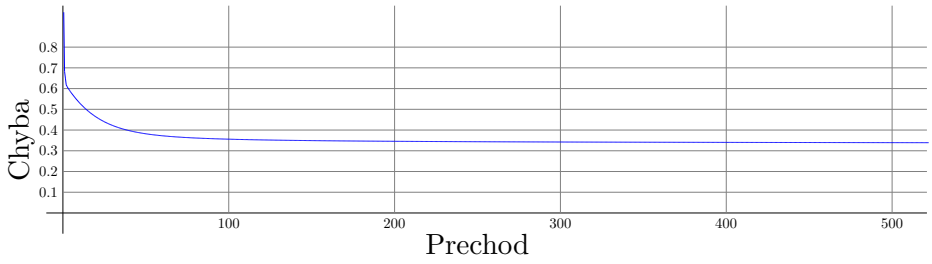
Najlepšie výsledky sme dosiahli s trojvrstvovou sieťou s týmito parametrami:

- Na prvej vrstve mala 5 príznakov, veľkosť rezervoára 80, každá sieť videla 30 súradníc rezervoára, spektrálny rádius 0.3.
- Na druhej vrstve mala 3 príznaky, veľkosť rezervoára 80, každá sieť videla 30 súradníc rezervoára, spektrálny rádius 0.3.
- Na tretej vrstve mala veľkosť rezervoára 20, spektrálny rádius 0.4.

Na všetkých vrstvách bola rýchlosť učenia 0.00003 a leaky rate na hlasoch ani leaky rate na rezervoári neboli použité (zhoršovali výsledky). Sieť bola veľmi citlivá na zmenu rýchlosti učenia a už aj jej malá zmena mala výrazný vplyv na výslednú chybu.

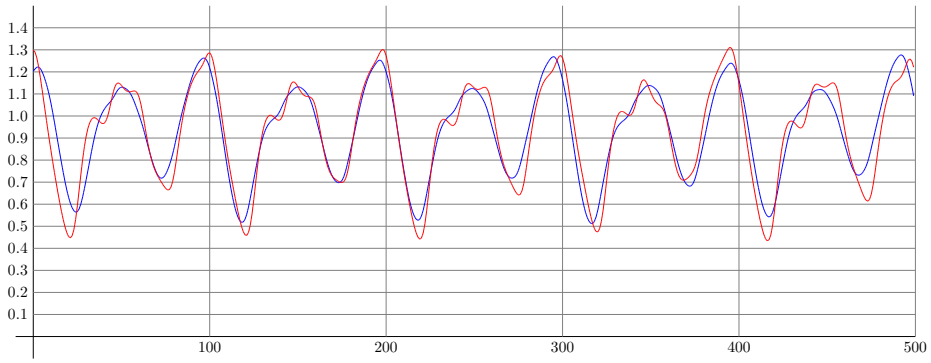
Na obrázku 13 je uvedený vývoj chyby počas jednotlivých prechodov trénovacou množinou (epoch).

Táto sieť sa dokázala natrénovať na chybu 0.336 po 1000 prechodoch trénovacou množinou. Na obrázku 14 je znázornený originálny signál



Obr. 13: Mackey-glass time series - vývoj chyby počas trénovania

(modrou) a aproximácia signálu (červenou). Uviedli sme len prvých 500 prechodov, nakoľko v ďalších chyba klesala len veľmi pomaly, čo by na grafe nebolo vidno.



Obr. 14: Mackey-glass time series - aproximácia siete. Červená čiara je originálny signál, modrá je aproximácia siete

5.2.3 Porovnanie

Rozhodli sme sa porovnať túto sieť s klasickou sieťou s echo stavmi (ktorú sme taktiež implementovali). Hore uvedená hierarchická echo state sieť s troma vrstvami mala spolu 796 učiacich sa prepojení (175 na prvej vrstve, 525 na druhej, 69 na tretej). Z toho dôvodu sme na porovnanie zvolili rezervoár siete

s echo stavmi veľkosti 796, aby mala rovnaký počet učiacich sa prepojení.

Táto sieť dosiahla NRMSE = 0.379, teda mierne horšiu ako jej hierarchická verzia. Obidve siete sa učili približne rovnaký počet prechodov trénovacou množinou, avšak jeden prechod siete s echo stavmi trval približne 18-krát dlhšie.

Vo svojej práci González et al. (2006) riešil rovnaký problém pomocou nimi upravených aproximátorov fuzzy function. Najlepší výsledok, ktorý so svojím modelom dosiahli bolo NRMSE = 0.0855 ± 0.0058.

Ďalší, ktorí riešili rovnakú úlohu boli Falco et. al. (1998), ktorí používali viacvrstvovú neurónovú sieť trénovanú pomocou Breeder genetic algorithmu. Dosiahli NRMSE = 0.266.

Vo svojej práci Platt (1991) využíval Resource Allocation Network a dosiahol NRMSE = 0.373. Úspešnosť tohto modelu na danej úlohe zlepšil Rosipal et. al. (1998). Pri využití Givens QR decomposition dosiahli NRMSE = 0.165 a pri využití Givens QR decomposition spolu s pruning criterion dosiahli NRMSE = 0.160 .

5.2.4 Aha! efekt

Ako je vidno z obrázku 13, počas trénovania nedošlo k Aha! efektu (pre bližší popis Aha! efektu nájdete v časti Reberova gramatika – Aha! efekt). Po skúmaní siete sme zistili, že príznaky na druhej a tretej vrstve dávali na svoj výstup neustále hodnoty blízke 0 a príznaky na prvej vrstve dávali na výstup takmer to isté, teda nadobudli spoločné myslenie.

Na to, aby druhá a tretia vrstva začali byť používané, by bolo potrebné prelomiť spoločné myslenie na prvej vrstve. To sa nám zatiaľ nepodarilo dosiahnuť (skúšali sme rôzne parametre siete), avšak toto je cieľom ďalšieho výskumu.

Toto bolo, podľa nás, dôvodom, prečo sieť nedosiahla lepšie výsledky a veríme, že ak by sa podarilo porušiť spoločné myslenie na prvej úrovni, NRMSE siete by sa znížilo a mohla by byť schopná konkurovať iným modelom.

6 Záver

Cieľom tejto práce bolo implementovať model hierarchickej neurónovej siete s echo stavmi a otestovať ho na niekoľkých úlohách.

Model sme úspešne implementovali, popísali a jeho zdrojový kód sa nachádza v prílohe k tejto práci.

Jeho úspešnosť sme testovali na dvoch úlohách – aproximácia prechodovej funkcie Reberovej gramatiky a predikcia v časovom rade Mackey-glass.

V úlohe aproximácie prechodovej funkcie Reberovej gramatiky sa nám podarilo úspešne natrénovať trojvrstvovú sieť. Pri jej trénovaní sme pozorovali dva Aha! efekty a na základe toho sme sieť analyzovali. Sledovali sme správanie jednotlivých príznakov na jednotlivých úrovniach pri jednotlivých stavoch automatu generujúceho slová z gramatiky v troch časoch počas trénovania. Prvé sledovanie bolo uskutočnené pred prvým Aha! efektom (epocha 50), druhé medzi prvým a druhým Aha efektom (epocha 200) a tretie po druhom Aha! efekte (epocha 500). Pri prvom pozorovaní boli príznaky na druhej a tretej vrstve konštantné a teda sa na výsledku siete podieľali v minimálnej miere. Pri druhom pozorovaní príznak na tretej vrstve ostával stále konštantný, avšak príznaky na druhej vrstve začali reagovať na stavy automatu. Na druhej vrstve sme pozorovali spoločné myslenie – všetky príznaky boli vždy takmer rovnaké. Pri treťom pozorovaní príznaky na prvej a druhej úrovani viditeľne reagovali na stavy automatu (na druhej vrstve došlo k prelomeniu spoločného myslenia) a príznak na tretej vrstve reagoval na koniec slova.

V úlohe dlhodobej predikcie časového radu Mackey-glass sa nám sieť podarilo natrénovať len čiastočne. Pri použití trojvrstvej siete sa nám podarilo dosiahnuť chybu NRMSE = 0.336. Pri trénovaní sme nepozorovali ani jeden Aha! efekt. Pri bližšom skúmaní sme pozorovali spoločné myslenie príznakov na prvej vrstve a konštantné výstupy na druhej aj tretej vrstve. Toto považujeme za dôvod, prečo nebol spozorovaný ani jeden Aha! efekt a prečo sa nám nepodarilo dosiahnuť nižšiu chybu. Veríme, že ak by sa nám podarilo prelomiť spoločné myslenie príznakov prvej vrstvy, chyba by sa znížila. Tento problém ponúka možnosti pre ďalší výskum.

Pri trénovaní siete sme pozorovali 3 fázy učenia príznakov na jednotlivých vrstvách. V prvej fáze sú všetky príznaky konštantné. V druhej fáze sa všetky príznaky menia, ale sú takmer rovnaké – maju spoločné myslenie. V tretej fáze nastáva prelomenie spoločného myslenia. Taktiež sme pozorovali, že každá vrstva sa začne učiť až keď nižšia vrstva prejde do tretej fázi.

Literatúra

[1] Jaeger H.: Discovering multiscale dynamical features with hierarchical Echo State Networks , Jacobs University, Bremen (2007)

[2] Lukoševičius M., Jaeger H.: Reservoir Computing Approaches to Recurrent Neural Network Training , Jacobs University, Bremen (2009)

[3] Russell, S.J., Norvig, P.: Artificial Intelligence: a Modern Approach. Prentice-Hall, Englewood Cliffs, New Jersey (1994)

[4] Jaeger, H.: Tutorial on training recurrent neural networks, covering BPTT, RTRL, EKF and the "Echo state network" approach. Technical Report GMD Report 159, German National Research Center for Information Technology (2002)

[5] István Szita, Viktor Gyenes, and András Lőrincz, Reinforcement Learning with Echo State Networks, In: International Conference on Artificial Neural Networks (ICANN), Proceedings, Part I. Lecture Notes in Computer Science Springer 2006, pp. 830-839.

[6] Sutton, R., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA (1998)

[7] Bakker, P.B.: The State of Mind - Reinforcement Learning with Recurrent Neural Networks. PhD thesis, Universiteit Leiden (2004)

[8] Jesús González, Ignacio Rojas, Héctor Pomares, Luis J. Herrera, Alberto Guillén, José M. Palomares, Fernando Rojas - Improving the accuracy while perserving the interpretability of fuzzy function aproximators by means of multi-objective evolutionary algorithms, Granada (2006)

[9] I. De Falco, A. Della Cioppa, A. Iazzetta, P. Natale, E. Tarantino, Optimizing neural networks for time series prediction, in: R. Roy, T. Furuhashi, P.K. Chawdhry (Eds.), Proceedings of the third On-line World Conference on Soft Computing (WSC3), Advances in Soft

Computing – Engineering Design and Manufacturing, Internet, Springer Verlag, 1998.

[10] J. Platt, A resource allocation network for function interpolation, Neural Computation 3 (1991) 213–225.

[11] R. Rosipal, M. Koska, I. Farkaš, Prediction of chaotic time-series with a resource-allocating RBF network, Neural Processing Letters 7 (1998) 185–197.

Príloha – CD médium

Na priloženom CD je uložený text tejto diplomovej práce v elektronickej podobe. Okrem toho sú na ňom uložené zdrojové kódy použité pri implementácii a vyhodnocovaní neurálneho modelu.