Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Porovnanie niekoľkých typov rekurentných sietí z hľadiska hĺbky pamäte

Diplomová práca

2019

Bc. Jaroslav Ištok

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Porovnanie niekoľkých typov rekurentných sietí z hľadiska hĺbky pamäte

Diplomová práca

Študijný program:	Aplikovaná informatika
Študijný odbor:	2511 Aplikovaná informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Školiteľ:	doc. RNDr. Martin Takáč, PhD.

Bratislava, 2019
Bc. Jaroslav Ištok

22271982

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:	Bc. Jaroslav Ištok
Študijný program:	aplikovaná informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor:	aplikovaná informatika
Typ záverečnej práce:	diplomová
Jazyk záverečnej práce:	slovenský
Sekundárny jazyk:	anglický

Názov: Porovnanie niekoľkých typov rekurentných sietí z hľadiska hĺbky pamäte
Memory span in recurrent neural network types: a comparison

Anotácia: Cieľom práce je preskúmať a porovnať vlastnosti niektorých typov rekurentných samoorganizujúcich sa máp (MSOM, RecSOM a ich modifikácií) s Elmanovou jednoduchou rekurentnou sieťou (SRN), najmä z hľadiska hĺbky a kapacity pamäte. Práca zahŕňa implementáciu, výpočtové simulácie a analýzu vrátane preskúmania priestoru parametrov.

Literatúra: Elman, J. (1990). Finding structure in time. Cognitive Science, 14, 179-211.
Strickert, M. & Hammer, B. (2005). Merge SOM for temporal data. Neurocomputing, 64, 39-71.

Vedúci:	doc. RNDr. Martin Takáč, PhD.
Katedra:	FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry:	prof. Ing. Igor Farkaš, Dr.
Dátum zadania:	05.10.2017

Dátum schválenia: 12.10.2017 prof. RNDr. Roman Ďurikovič, PhD.
garant študijného programu

....

študent vedúci práce

Čestné vyhlásenie: Čestne vyhlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením školiteľa diplomovej práce, s použitím poznatkov z uvedenej literatúry.

Poďakovanie: Chcel by som poďakovať svojmu školiteľovi, doc. RNDr. Martin Takáčovi, PhD., za jeho pomoc, čas, poskytné materiály a hlavne za skvelé nápady počas experimentu.

Abstrakt

V našej práci implementujeme niekoľko typov neurónových sietí a porovnávame ich z hľadiska hĺbky pamäte. Hľadáme optimálne parametre, pri ktorých majú rôzne typy neurónových sietí najvyššie pamäťové hĺbky. Následne analyzujeme a porovnávame výsledky pre testované typy sietí.

Kľúčové slová: rekurentné neurónové siete, hĺbka pamäte, samorganizujúca sa mapa

Abstract

In our work we are implementing a several types of neural nets which we are comparing in terms of their memory span. We are finding optimal parameters for which have different types of tested neural nets highest memory span values. Then, we are analyzing and comparing results for tested neural nets.

Keywords: neural net, machine learning, memory span, self organizing map

Obsah

Zoznam obrázkov

Zoznam tabuliek

Kapitola 1

Úvod

1.1 Motivácia

V strojovom učení nastávajú situácie, kedy potrebujeme predikovať výsledok nie len na základe aktuálneho vstupu, ale aj na základe historického kontextu. Kontext je väčšinou reprezentovaný stavmi modelu strojového učenia z minulých krokov alebo kombináciou predchádzajúcich vstupov. Príkladom takejto úlohy môže byť spracovanie tzv. časových radov či generovanie postupností znakov. Predstavme si úlohu, v ktorej chceme model strojového učenia naučiť predikovať ďalšie písmeno v určitom slove. Ako príklad si môžeme zobrať slovo „Bratislava“. Trénovanie modelu strojového učenia bez využitia historického kontextu by mohlo prebiehať napríklad takto: Trénovaciou množinu by tvorili písmená daného slova, kde očakávanou hodnotou pre nejaké písmeno by bolo ďalšie písmeno, ktoré za ním v slove nasleduje. Čiže pre písmeno „B“ poviem, že očakávame „r“ a takto pokračujem ďalej cez všetky písmená v slove.

Problém nastane pri druhom písmene „a“ v slove Bratislava. Pri prvom výskyte písmena „a“ sme modelu tvrdili, že očakávam písmeno „t“ a pri druhom výskyte písmena „a“ tvrdíme, že očakávame „v“. Tento prípad sa model, ktorý nevyužíva žiadny historický kontext, nevie naučiť, pretože nemá žiadnu "pamäť"(okno do minulosti). Historický kontext umožňuje modelom strojového učenia pracovať s určitou formou pamäte.

V našej práci sa budeme zaoberať jedným z modelov strojového učenia, ktorý pracuje s kontextom: rekurentnými neurónovými sieťami viacerých typov pričom budem merať a porovnávať ich pamäťovú hĺbku. Pamäťová hĺbka neurónovej siete vo všeobecnosti vyjadruje s akým dlhým kontextom do minulosti dokáže pracovať.

Konkrétne pôjde o nasledujúce tri typy rekurentných neurónových sietí a ich modifikácie:

- RecSOM
- MergeSOM

- Elmanova sieť

1.2 Úvod do umelých neurónových sietí

1.2.1 Perceptrón

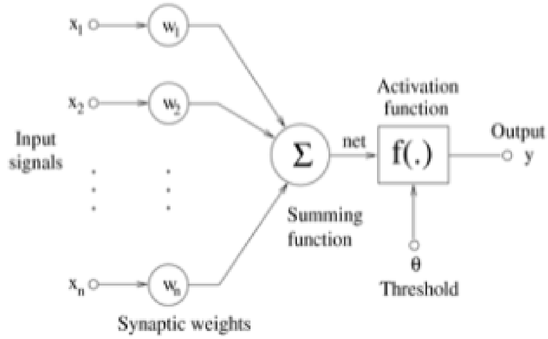
Je základným modelom neurónu, ktorý je používaný v neurónových sieťach.

Na vstupe každého perceptrónu je vektor vstupných hodnôt (vzor) $\overline{x}$ pričom hodnoty sú reálne čísla alebo binárne hodnoty, v ktorých môže byť zakódované napríklad slovo alebo obrázok. Každý perceptrón má vektor synaptických váh $\overline{w}$.

Výstup perceptrón vyjadruje rovnica:

$$y = f(\sum_{j=1}^{n+1} w_i x_j) \quad x_{n+1} = -1 \tag{1.1}$$

Funkcia f sa nazýva aj aktivačná funkcia perceptrónu, ktorá môže byť spojitá alebo diskrétna. Podľa toho, či je aktivačná funkcia spojitá alebo diskrétna, rozlišujeme spojitý alebo diskrétny perceptrón.



Obr. 1.1: Percetrón

Aktivačná funkcia diskrétneho perceptrónu je jednoduchá bipolárna funkcia, ktorá vráti +1 alebo 1.

$$f(net) = sign(net) = \begin{cases} 1 & \text{ak } net \geq 0 \\ \text{-1} & \text{ak } net < 0 \end{cases}$$

Aktivačná funkcia spojitého perceptrónu môže byť napríklad sigmoida.

$$f(net) = \frac{1}{1 + \exp^{-net}} \tag{1.2}$$

Vstupom do aktivačnej funkcie (net) je skalárny súčin vstupného vektora $\overline{x}$ a váhového vektora perceptrónu $\overline{w}$.

$$net = \overline{x} \cdot \overline{w} \tag{1.3}$$

Trénovanie perceptrónu prebieha za pomoci učiteľa (angl. supervised learning). Pravidlo pre adaptáciu váh diskrétneho percetrónu:

$$w_j(t+1) = w_j(t) + \alpha(d - y) \cdot x_j \tag{1.4}$$

Spojitý perceptrón je trénovaný metódou najprudšieho spádu, pomocou ktorej minimalizujeme kvadratickú chybovú funkciu:

$$E(w) = \frac{1}{2} \sum_p \left(d^{(p)} - y^{(p)}\right)^2 \tag{1.5}$$

Pravidlo pre adaptáciu váh v spojitom perceptróne:

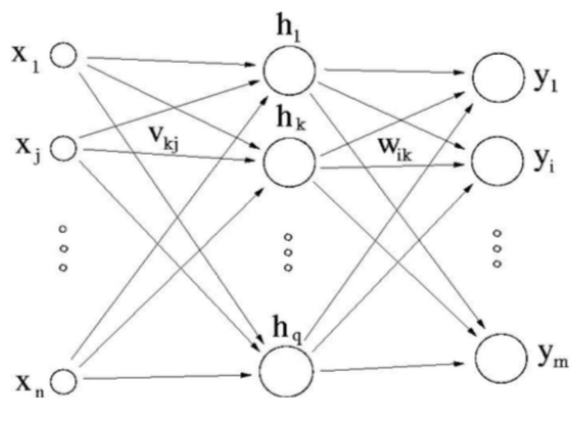
$$w_j(t+1) = w_j(t) + \alpha(d^{(p)} - y^{(p)}) f'(net) x_j = w_j(t) + \alpha\delta^{(p)} x_j \tag{1.6}$$

Kde p je index vzoru v trénovacej množine, y je skutočný výstup a d je požadovaný výstup.

Perceptrón je veľmi jednoduchým modelom. Nevieme ním vyriešiť celú škálu úloh. Napríklad dokáže klasifikovať iba lineárne separovateľné dáta. Je však dokázané, že ak dáta sú lineárne separovateľné, trénovací algoritmus konverguje a teda vie dáta separovať. Perceptrón sa v strojovom učení niekedy označuje aj ako logistická regresia.

1.2.2 Viacvrstvová dopredná neurónová sieť

V našej práci sa budeme zaoberať Elmanovou rekurentnou neurónovou sieťou. [1] Najskôr však popíšeme základný viacvrstvový model neurónovej siete. Viacvrstvová dopredná neurónová sieť má jednu vstupnú vrstvu, jednu výstupnú vrstvu a minimálne jednu skrytú vrstvu. Jednotlivé vrstvy sú tvorené perceptrónmi a sú pospájané väzbami, ktoré majú váhy, resp. váhové vektory. Medzi neurónmi v tej istej vrstve nie sú žiadne spojenia.



Obr. 1.2: Viacvrstvová dopredná neurónová sieť

Trénovací algoritmus pomocou ktorého sa trénujú dopredné neurónové sa nazýva „spätné šírenie chyby“ (angl. backpropagation).

Aktivácie na neurónoch výstupnej vrstvy (výstup) môžeme popísať vzťahom:

$$y_i = f(\sum_{k=1}^{q+1} w_{ik} h_k) \tag{1.7}$$

Aktivácie na neurónoch skrytej vrstve môžeme popísať vzťahom:

$$h_k = f(\sum_{j=1}^{n+1} v_{kj} x_j) \tag{1.8}$$

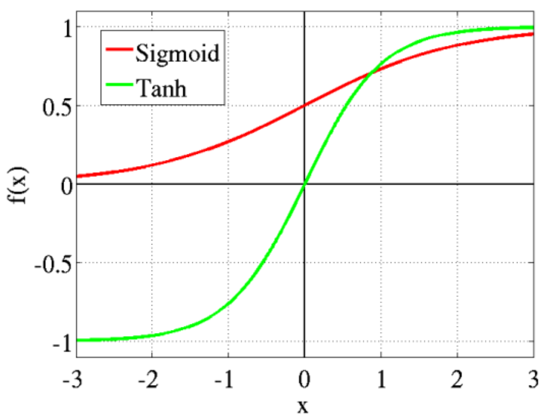
Prahové hodnoty:

$$x_{n+1} = h_{q+1} = -1 \tag{1.9}$$

Aby dopredná neurónová sieť vedela pracovať aj s nelineárnymi problémami, musí byť aktivačná funkcia neurónov nejaká nelineárna diferencovateľná funkcia.

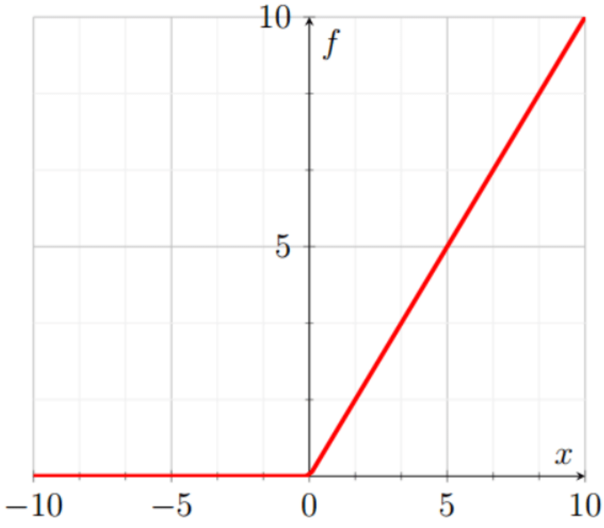
Najpoužívanejšie aktivačné funkcie sú:

- Logistická sigmoida
- Hyperbolický tangens



Obr. 1.3: Logistická sigmoida a Hyperbolický tangens

- ReLU



Obr. 1.4: Rectified Linear Unit

Vzťahy pre aktualizáciu váh sú nasledovné:

Váhy medzi vstupnou a skrytou vrstvou

$$w_{ik}(t+1) = w_{ik}(t) + \alpha\delta_i h_k \quad \delta_i = (d_i - y_i)f'(net_i) \tag{1.10}$$

Váhy medzi skrytou a výstupnou vrstvou:

$$v_{kj}(t+1) = v_{kj}(t) + \alpha\delta_k x_j \quad \delta_k = (\sum_i w_{ik}\delta_i)f'(net_k) \tag{1.11}$$

1.2.3 Trénovací algoritmus dopredných neurónových sietí

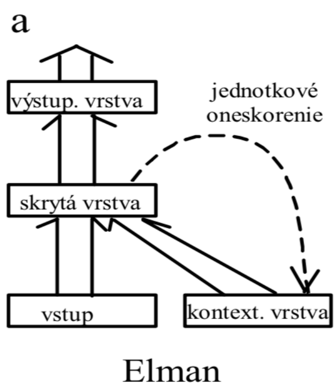
1. zoberie sa vstup $x^{(p)}$ a vypočíta sa výstup $y^{(p)}$ (dopredný krok)
2. vypočíta sa chyba pomocou zvolenej chybovej funkcie
3. vypočítame hodnoty δ_i a δ_k (spätný krok)
4. upravíme váhy Δw_{ik} a Δv_{kj}
5. ak už boli použité všetky trénovacie príklady z trénovacej množiny pokračuj bodom 6. inak pokračuj bodom 1.
6. ak bolo dosiahnuté nejaké zastavovacie kritérium, potom skonči, inak prepermutuj vstupy a pokračuj bodom 1.

1.2.4 Rekurentné neurónové siete

Rekurentná neurónová sieť je akákoľvek neurónová sieť, ktorá obsahuje množinu neurónov v ktorých je ochovávaná informácia o aktiváciách neurónov alebo predchádzajúcich vstupoch z predošlých krokov. Takéto neuróny nazývame aj rekurentné neuróny a tvoria vrstvu, ktorá sa nazýva kontextová vrstva. Týmto spôsobom je sieť rozšírená o vnútornú pamäť.

V našej práci budeme skúmať vlastnosti a skúšať zmerať pamäťovú hĺbku rekurentnej siete s Elmanovou architektúrou, ktorá je znázornená na nasledujúcom diagrame.

OBR.6.17.
ARCHITEKTÚRY REKURENTNÝCH SIETÍ



Obr. 1.5: Architektúra elmanovej siete

Dvojité šípky reprezentujú spojenia neurónov medzi vrstvami. Neuróny v tej istej vrstve nie sú, podobne ako v nerekurentnej, teda doprednej, sieti, medzi sebou prepojené. Spojenia znázornené dvojitou šípkou majú váhy, ktoré sú upravované počas trénovania. Jednoduché šípky reprezentujú rekurentné spojenia. Tieto majú nemennú váhu s hodnotou 1. Tieto spojenia slúžia na odpamätanie aktivácii rekurentných neurónov.

Využitie rekurentných neurónových sietí:

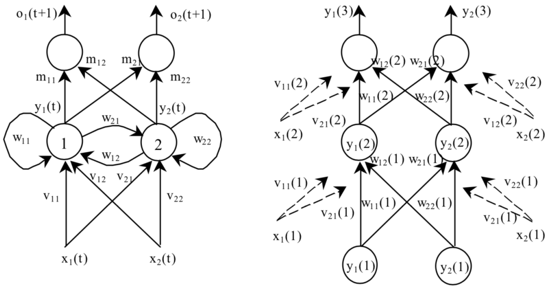
- Klasifikácia s časovým kontextom
  Príkladom úlohy môže byť napríklad určiť, či určitá postupnosť vstupov patrí do nejakej triedy. Praktickým problémom môže byť zistiť či postupnosť signálov z určitého zariadenia signalizuje poruchu zariadenia alebo nie.

- Predikcie
  Príkladom môže byť napríklad predikcia nasledujúceho člena postupnosti.

- Generovanie postupnosti
  Podobne ako predikcie, ale nepredikujeme iba ďalšiu hodnotu postupnosti, ale

celé pokračovanie. Je to komplexnejšia úloha. Napríklad 23123123123123123123 .. pokračovaním by bolo 123123123... Reálne úlohy sú komplexnejšie.

Trénovací algoritmus, ktorým sa trénujú rekurentné neurónové siete je spätné šírenie chyby v čase (angl. backpropagation through time). Pri použití tohto algoritmu sa sieť rozvíja v čase, t.j. rozvinutá sieť má toľko skrytých vrstiev, koľko je vstupov T vo vstupnej postupnosti. Takáto sieť sa potom trénuje podobne ako klasická nerekurentná dopredná sieť s T skrytými vrstvami pomocou jednoduchého spätného šírenia chyby. Aktivity neurónov kontextovej vrstvy sú na začiatku kažého cyklu trénovania nastavené na hodnotu 0.5.



Obr. 1.6: Rozvinutá Elmanova sieť

Výstup takejto siete počítame pomocou vzťahu:

$$o_k^{(T+1)} = f(\sum_{j-1}^{J} m_{kj}^{T+1} y_j^{T+1}) \tag{1.12}$$

$$y_j^{t+1} = f(\sum_{i=1}^{J} w_{ji}^t y_i^t + \sum_{i=1}^{I} v_{ji}^t x_i^t) \tag{1.13}$$

Chybová funkcia v čase $T+1$

$$E(T+1) = \frac{1}{2} \sum_{k+1}^{K} (d_k^{(T+1)} - o_k^{(T+1)})^2 \tag{1.14}$$

Zmena váh medzi vstupnou a výstupnou vrstvou:

$$\Delta m_{kj}^{T+1} = -\alpha \frac{\partial E(T+1)}{\partial m_{kj}} = \alpha \delta_k^{T+1} y_j T + 1 \tag{1.15}$$

$$\delta_k^{T+1} = (d_k^{T+1} - o_k^{T+1}) f'(net_k^{T+1}) \tag{1.16}$$

Váhy medzi rozvinutými vrstvami a medzi vstupom a rozvinutými vrstvami sa upravujú pomocou vzťahov:

$$\Delta w_{hj}^{T+1} = \frac{\sum_{t=1}^{T} \Delta w_{hj}^t}{T} \tag{1.17}$$

$$\Delta w_{ji}^{T+1} = \frac{\sum_{t=1}^{T} \Delta w_{ji}^{t}}{T} \tag{1.18}$$

1.2.5 Samoorganizujúce sa mapy

Ďalším typom rekurentných neurónových sietí, ktorých pamäťovú hĺbku budeme skúmať v našej práci, sú rekurentné samoorganizujúce sa mapy, ktoré sú rozšírením základnej nerekurentnej verzie samoorganizujúcich sa máp (ďalej iba SOM). Preto najskôr popíšem základný model nerekurentný model SOM.

SOM je typ neurónovej siete, v ktorej sú jednotlivé neuróny usporiadané do (väčšinou) dvojrozmernej štvorcovej mriežky. Samoorganizujúce sa mapy (ďalej iba SOM) sú trénované bez učiteľa, čiže váhy jednotlivých neurónov sú upravované iba na základe dát z trénovacej množiny. Čo je zaujímavé na spôsobe trénovania SOM je, že je veľmi podobný učeniu neurónov v mozgovej kôre živočíchov. Je to biologicky inšpirovaný model. Špeciálnou vlastnosťou SOM je, že po natrénovaní zobrazí trénovaciu množinu so zachovanou topológiou. To znamená, že blízke (podobné) vstupy aktivujú blízke neuróny v sieti. Vzdialenosť dvoch neurónov je ich euklidovská vzdialenosť v mriežke. Takéto topologické zobrazenie dát sa vyskytuje aj v biologických neurónových sieťach.

1.2.6 Trénovanie

Proces trénovania SOM je zložený z dvoch častí:

- hľadanie víťaza
- adaptácia váh neurónov

Na začiatku su váhy medzi vstupom a neurónmi v mriežke inicializujú na náhodné hodnoty z určitého intervalu. V každom kroku trénovania nájdeme najskôr víťazný neurón pre daný vstup. Postupne počítame euklidovské vzdialenosti vstupu od váhového vektora jednotlivých neurónov. Víťazom je neurón, ktorý je najbližšie k vstupu (má najkratšiu vzdialenosť).

$$i^* = argmin_i ||x - w_i|| \tag{1.19}$$

Druhým krokom je adaptácia váh víťazného neurónu a jeho okolia. Pravidlo pre zmenu váh neurónov:

$$w_i(t+1) = w_i(t) + \alpha(t) h(i^*, i)([x(t) - w_i(t)]) \tag{1.20}$$

Váhové vektory víťazného neurónu a jeho topologických susedov sa posúvajú smerom k aktuálnemu vstupu. $\alpha(t)$ predstavuje rýchlosť učenia, ktorá môže klesať v čase alebo môže zostať konštantná. Na funkcii, ktorá je použitá pre α, v praxi veľmi nezáleží, mala by to byť nejaká monotónne klesajúca funkcia (napríklad exponenciálna funkcia). $h(i^*, i)$ je funkcia okolia (tzv. excitačná funkcia), ktorá definuje koľko neurónov v okolí víťaza bude trénovaných a do akej miery. Inými slovami, excitačná funkcia definuje rozsah kooperácie medzi neurónmi. Používajú sa najčastejšie 2 typy okolia:

- pravouhlé(štvorcové) okolie

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

  $d_M(i^*, i)$ je Manhattanovská vzdialenosť (L1 norma) medzi neurónmi v mriežke mapy. Kohonen zistil, že najlepšie výsledky sú dosiahnuté, keď sa veľkosť okolia s časom postupne zmenšuje.

- gaussovské okolie

$$N(i^*, i) = \exp^{-\frac{d_E^2(i^*, i)}{\lambda^2(t)}} \tag{1.21}$$

  $d_E(i^*, i)$ je euklidovská vzdialenosť (L2 norma) neurónov v mriežke. Funkcia $\lambda^2(t)$ sa s časom postupne zmenšuje až k nule. Táto funkcia slúži na zmenšovanie okolia víťazného neurónu počas trénovania, čím sa zabezpečí ukončenie učenia.

$[x(t) - w_i(t)]$ je euklidovská vzdialenosť medzi vstupným vektorom a váhovým vektorom.

Na vyhodnocovanie trénovania SOM používame kvantizačnú chybu. Pri SOM je kvantizačná chyba euklidovská vzdialenosť vstupu od váh víťazného neurónu. Po každej epoche učenia vieme určiť celkovú kvantizačnú chybu siete pre danú trénovaciu množinu. Pre každý príklad z trénovacej množiny vypočítame kvantizačnú chybu a nakoniec spravíme priemer kvantizačných chýb. Táto by mala po každej epoche učenia (po natrénovaní a adaptácii váh na celej trénovacej množine) postupne klesať k určitému minimu.

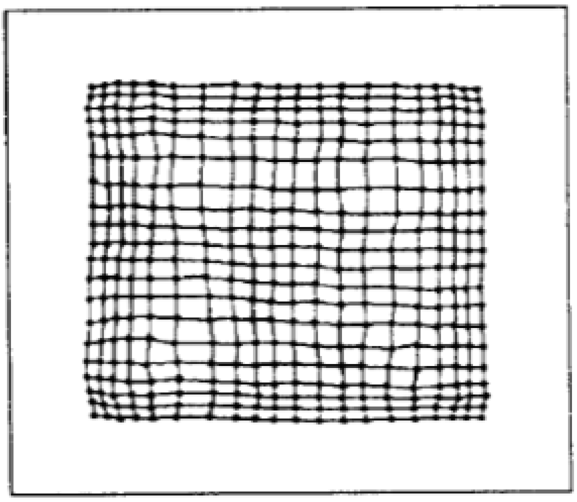
Pri učení rozlišujeme všeobecne dve fázy:

- fáza usporiadavania - s časom klesá veľkosť okolia víťazných neurónov
- fáza dolaďovania - veľkosť okolia sa zafixuje na nejakej malej hodnote až pokým učenie neukončí

Kohonen odhadol na základe pokusov, že počet iterácií trénovania, by mal byť rádovo 500-násobok počtu neurónov v sieti. Rovnako sa pozorovaním zistilo, že na fázu doladenia je lepšie ponechať viac času ako na fázu usporiadavania.

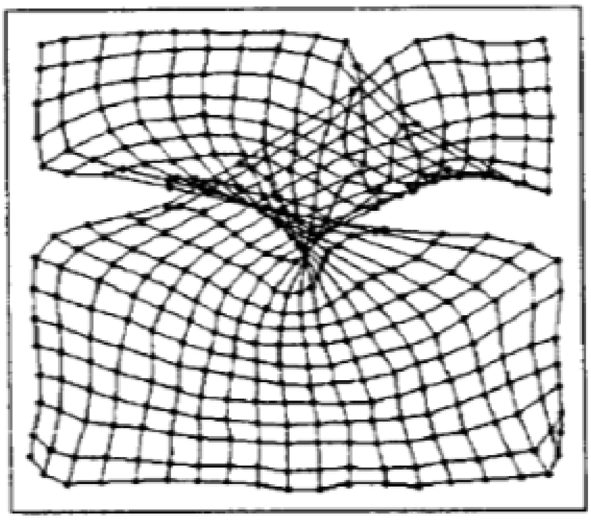
Počas trénovania SOM môžu nastať špeciálne situácie:

- Sieť je neúplne rozvinutá - príliš rýchle zmenšovanie rýchlosti učenia α



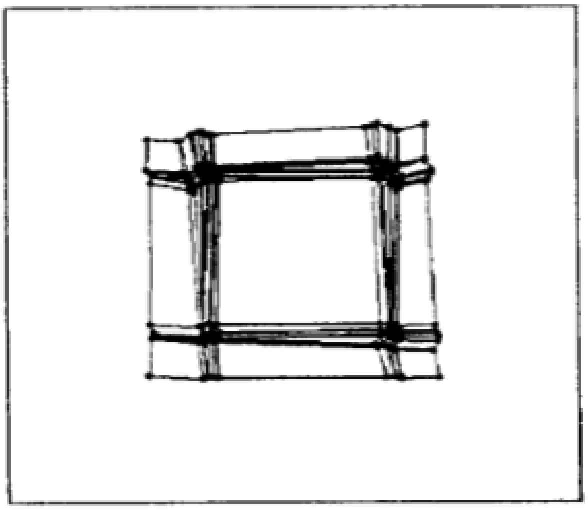
Obr. 1.7: Neúplne rozvinutá sieť

- Motýlí efekt - príliš rýchle zmenšovanie okolia λ



Obr. 1.8: Motýlí efekt

- Pinch efekt - príliš pomalé zmenšovanie okolia λ



Obr. 1.9: Pinch effect

1.2.7 Využitie SOM

- SOM môžeme využiť na mapovanie viacrozmerných dát do 2D - môžeme ju použiť na redukciu dimenzie dát.
- SOM je aj vektorovým kvantizátorom. Pri vektorovej kvantizácii nahrádzame množinu vektorov vstupných dát menšou množinou vektorov (nazývaných aj prototypy). V SOM sú prototypmi váhové vektory. Toto je možné využiť napríklad na kompresiu dát. Vďaka vektorovej kvantizácii stačí uchovať iba množinu prototypov a informáciu o tom, ktorý vstupný vektor patrí ku ktorému prototypu (centru). Ku každému centru sa potom priradí množina vstupných vektorov, ktoré ku nemu majú bližšie ako ku akémukoľvek inému centru (používa sa euklidovská vzdialenosť). Vektorou kvantizáciou teda rozdelíme vstupný priestor an disjunktné oblasti, ktoré tvoria tzv. Voronoiho mozaiku.

1.2.8 Rekurentné modely

Rekurentné samoorganizujúce sa mapy sú modifikáciou nerekurentnej SOM. Rozdiel oproti nerekurentnej verzii je v tom, že vstupy sú porovnávané nielen s váhovým vektorom jednotlivých neurónov, ale aj s kontextom. Rôzne verzie rekurentných SOM sa líšia v type kontextu, ktorý je v nich použítý. V kontexte rekurentnej SOM je spravidla uložené vlastnosti siete z minulého časového kroku alebo kombinácia minulých vstupov.

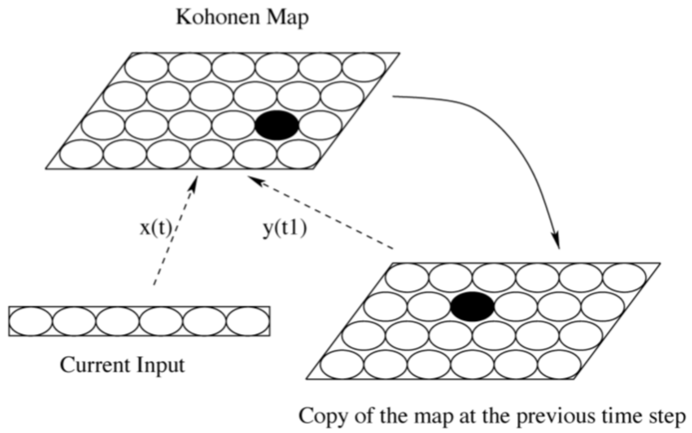
V mojej práci budem porovnávať 2 základné typy rekuretných SOM.

- Recursive SOM (RecSOM): Pri RecSOM je kontextom celá kópia aktívacií neurónov z minulého kroku.

- Merge SOM (mSOM): Pri mSOM sú kontextom vlastnosti víťazného neurónu z predchádzajúceho kroku učenia.

1.2.9 RecSOM

Pri RecSom je SOM algoritmus použítý rekurzívne na vstupný vektor $x(t)$ a tiež reprezentáciu mapy z minulého kroku $y(t-1)$. [6]



Obr. 1.10: Architektúra RecSOM

V RecSOM je každý vstup asociovaný k stavom z predchádzajúcich krokov a preto každý neurón reaguje na sekvenciu vstupov. Prerušované čiary predstavujú trénovateľné spojenia.

Každý neurón má 2 váhové vektory. Váhový vektor w_i, ktorý je porovnávaný so vstupným vektorom $x(t)$ a vektor kontextových váh c_i, ktorý je porovnávaný s kontextom z predchádzajúceho kroku $y(t-1)$. Kedže chceme aby boli dopredné a spätné spojenia v RecSOM homogénne, celkovú chybu určíme ako váhovaný súčet druhých mocnín kvantizačných chýb oboch prípadov. Chyba je vlastne váhovaný súčet euklidovských vzdialeností vstupu od váhového vektoru a kontextu z predchádzajúceho kroku od kontextových váh.

N je počet neurónov v sieti (pretože kontextom je kópia celej mapy)

$$d_i = (1-\alpha) \cdot ||x(t) - w_i||^2 + \alpha \cdot ||y(t-1) - c_i||^2 \quad c \in R^N \tag{1.22}$$

α a β sú parametre, ktoré určujú akú váhu pri trénovaní bude mať kontext a akú váhu bude mať aktuálny krok.

Kontextom je kópia aktivácii všetkých neurónov z predchádzajúceho kroku.

$$y(t) = [y_1(t-1), ..., y_N(t-1)] \quad c, r \in R^N \tag{1.23}$$

Hodnota aktivácie pre určitý neurón je vyjadrená vzťahom:

$$y_i = \exp(-d_i) \tag{1.24}$$

Pravidlo pre zmenu váh neurónov (podobné ako pri nerekurentnej SOM):

$$w_i(t+1) = w_i(t) + \alpha(t)h(i^*, i)[x(t) - w_i(t)] \tag{1.25}$$

Pre pre zmenu kontextových váh, platí to isté pravidlo ako pre normálne váhy, iba je aplikované na kontextové vektory:

$$c_i(t+1) = c_i(t) + \alpha(t)h(i^*, i)[y(t-1) - c_i(t)] \tag{1.26}$$

Pri RecSom majú kontext aj kontextové váhy veľkú dimenziu (rovnú počtu neurónov v sieti, keďže kontextom je vektor aktivácii všetkých neurónov z predchádzajúceho kroku), čo spomaľuje výrazne proces trénovania takejto siete. Výhodou môže byť, že kontext RecSOM obsahuje veľa informácií a teda môže mať v niektorých prípadoch lepšie vlastnosti. Jedným z cieľov našej práce je zistiť či je skutočne tento rozšírený kontext potrebný a či má nejaký vplyv na pamäťovú hĺbku.

1.2.10 mSOM

mSOM je podobná recSOM, líši sa v reprezentácii kontextu. [5]

Chybu (vzdialenosť vstupu od neurónu) vyjadríme vzťahom (podobne ako pri recSOM), d je dimenzia vstupov:

$$d = \alpha \cdot ||s(t) - w_i||^2 + \beta \cdot ||r(t) - c_i||^2 \quad x, c \in R^d \tag{1.27}$$

Kontextom pri mSOM nie je kópia aktivácii neurónov z predchádzajúceho kroku, ako je tomu pri RecSom. Kontextom pri mSOM je zlúčenie (lineárna kombinácia) vlastností víťaza z predchádzajúceho kroku. (Odtiaľ je odvodený názov - „zlučovacia samoorganizujúca sa mapa“ - Merge SOM). Kontext mSOM vyjadríme vzťahom:

$$y(t) = \beta \cdot w_{i^*}(t-1) + (1-\beta) \cdot y_{i^*}(t-1) \tag{1.28}$$

β je zlučovací parameter, ktorý určuje váhu kombinovaných vlastností víťazného neurónu z predchádzajúceho kroku. Typická hodnota tohto parametra počas trénovania je $\beta = 0.5$, čiže taká, aby obe zložky mali približne rovnakú váhu. Kedže vlastnosti víťazného neurónu sú reprezentované jeho váhovým vektorom, kontextom pri mSOM je lineárna kombinácia váhového vektora víťazného neurónu a kontextového vektora víťazného neurónu z predchádzajúceho kroku. Pravidlá pre adaptáciu váh váhového vektora a kontextových váh zostávajú rovnaké ako pri RecSom, resp. SOM. Líšia sa iba v kontexte.

$$w_i(t+1) = w_i(t) + \alpha(t)h(i^*, i)[s(t) - w_i(t)] \tag{1.29}$$

Pravidlo pre zmenu kontextových váh:

$$c_i(t+1) = c_i(t) + \alpha(t)h(i^*, i)[y(t-1) - c_i(t)] \tag{1.30}$$

Kontext pri štandartnej mSOM obsahuje odlišné informácie ako pri RecSom. Na rozdiel od RecSom, kde kontext tvorí vektor aktivít všetkých neurónov z predchádzajúceho kroku, pri mSOM je kontext tvorený iba lineárnou kombináciou vlastností víťazného neurónu z minulého kroku. Z toho vyplýva, že kontext v mSOM uchováva menšie množstvo informácii ako kontext v RecSOM. Má však značne menšiu dimenziu (dimenzia kontextu pri mSOM je rovnaká ako dimenzia vstupov), vďaka čomu je trénovanie mSOM rýchlejšie (výpočtovo menej náročné) a teda je možné experimentovať s viacrozmernými mapami, vyskúšať viac epoch trénovania, alebo použiť väčšie trénovacie množiny.

Cieľom našej práce je aj zistiť, či redukovaný kontext pri mSOM je postačujúci a či sme s ním schopní dosiahnuť podobné výsledky ako pri recSOM, kde kontext tvorí aktivácia všetkých neurónov v sieti.

Rozdielne typy kontextov a ich vplyv na pamäťovú hĺbku rekurentných neurónových sietí je hlavným cieľom mojej diplomovej práce.

1.2.11 Leaky integration mSOM

Pre potreby testovania pamäťovej hĺbky sme si vytvorili modifikovanú verziu mSOM, ktorá používa odlišný kontext. Je to modfikovaná verzia mSOM, ktorej kontext je tvorený minulými vstupmi. V pôvodnej mSOM bol kontext tvorený lineárnou kombináciou vlastností víťazného neurónu z predchádzajúceho kroku. V našej modifikovanej verzii mSOM bude kontext tvoriť kombinácia všetkých vstupných vektorov z predchádzajúcich krokov. Zvyšné vlastnosti siete zostávajú rovnaké ako v pôvodnej mSOM.

Samotný kontext počítame pomocou nasledujúceho rekurzívneho vzťahu:

$$c = \beta^0 \cdot x_t + \beta^1 \cdot x_{t-1} + \beta^2 \cdot x_{t-2} .... \tag{1.31}$$

β parameter je číslo z intervalu $\beta < 1 \wedge \beta > 0$ a x_t, x_{t-1}, x_{t-2}... sú vstupné vektory.

Z rekurzívneho vzťahu vyplýva, že kontext je tvorený kombináciou predchádzajúcich vstupov pričom čím dávnejší je vstup, tým menšiu váhu má v kontexte, čo je zabezpečené umocňovaním β parametra. Toto sa nazýva v matematike leaky integration. V našom prípade to znamená, že dávne vstupy postupne strácajú na dôležitosti pričom sa stále sa podieľajú na vytváraní výsledného kontextu.

Kapitola 2

Implementácia

2.1 Implementácia neurónových sietí

Pre potreby merania pamäťovej hĺbky sme potrebovali veľmi modifikované implementácie neurónových sietí, preto sme sa rozhodli pre ich vlastnú implementáciu. Vlastná implementácia nám umožnila experimentovať s rôznymi typmi kontextov a rôznymi spôsobmi trénovania sietí, čo s existujúcimi implementáciami bolo veľmi nepraktické a v istých prípadoch nemožné. Medzi špeciálne prípady patrí napríklad použitie modifikovaných kontextov, úprava excitačnej funkcie počas trénovania (zmenšovanie okolia), dynamické znižovanie rýchlosti učenia siete počas trénovania/po jednotlivých epochách trénovania, vytvorenie pamäťového okna v jednotlivých neurónoch a samotné meranie pamäťovej hĺbky.

2.2 Voľba programovacieho jazyka

Ako implementačný jazyk sme zvolili Python pretože preň existuje veľké množstvo kvalitných knižníc pre prácu s maticami a vektormi, či vykresľovanie grafov. Python je veľmi populárny v oblasti strojového učenia. Ďalšou výhodou je jednoduché spustenie skriptov na linuxovom serveri, čo urýchľuje samotné trénovanie a hľadanie najoptimálnejších parametrov a umožnilo nám otestovať veľké množstvo kombinácii rôznych parametrov.

2.2.1 Python

Python je interpretovaný vysokoúrovňový programovací jazyk. Python kladie dôraz na jednoduchosť a čitateľnosť programov, ktoré sú v ňom naprogramované. Je to jazyk, ktorý využíva dynamické typovanie a automatizovanú správu pamäte. Je to tiež multiplatformový jazyk a beží na všetkých bežne používaných platformách (Windows, Mac, Linux)

2.3 Použíté knižnice

Používame štandartný set knižníc pre implementáciu neurónových sietí: numpy, scipy. Na vykresľovanie a vizualizáciu dát používame knižnice matplotlib a seaborn.

2.3.1 Numpy

Je knižnica, ktorá uľahčuje prácu s maticami, používaná je takmer všetkými existujúcimi knižnicami, ktoré implementujú modely strojového učenia v Pythone. Má vysokú úroveň optimalizácie a požíva veľmi optimalizované funkcie na prácu s maticami, ktoré sú naprogramované v jazyku C.

2.3.2 Matplotlib

Je knižnica na vykresľovanie grafov a vizualizáciu dát.

2.3.3 Seaborn

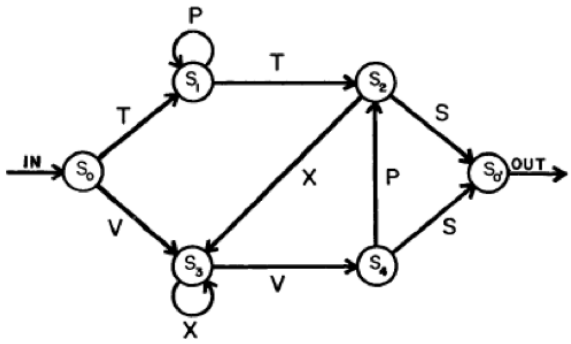
Je nadstavbou Matplotlib knižnice a zjednodušuje vykresľovanie rôznych grafov.

2.4 Algoritmus hľadania najdlhšej spoločnej podpostupnosti viacerých reťazcov

Na hľadanie najdlhšej spoločnej podpostupnosti viacerých reťazcov som použil relatívne jednoduchý naivný prístup. Z každej sekvencie sme si vytvorili všetky možné n-gramy (podpostupnosti), ktoré si ukladáme do množiny. Pre každú sekvenciu v množine vytvoríme takúto množinu n-gramov. Potom spravíme prienik týchto množín a dĺžka najdlhšieho reťazca z tohto prieniku je dĺžka najdlhšej spoločnej podpostupnosti. Ide o naivný algoritmus, ktorý by bol pri dlhých reťazcoch pamäťovo aj výpočtovo neoptimálny, avšak pre potreby merania pamäťovej hĺbky neurónových sietí je postačujúci a dostatočne efektívny, keďže sekvencie v pamäťových oknách jednotlivých neurónov nie sú dlhé (do 30 znakov maximálne) Použitie tohto algoritmu malo veľmi malý vplyv na rýchlosť trénovania.

2.5 Reberov automat

Na vytvorenie trénovacej množiny, ktorá pozotáva z reberových reťazcov sme si vytvorili vlastnú implementáciu pravdepodobnostného nedeterministického konečného reberového automatu, pomocou ktorého generujeme reberové reťazce. Každý stav okrem počiatočného a konečného stavu má práve dva prechody do ďalšieho stavu, pričom v každom stave je 50% šanca na prechod do jedného z možných stavov.



Obr. 2.1: Schéma reberovho automatu

Kapitola 3

Podobné práce

Kapitola 4

Návrh riešenia

Na to aby sme boli schopní merať a porovnávať pamäťovú hĺbku sietí, museli sme zvoliť vhodné trénovacie dáta, zadefinovať si spôsob merania pamäťovej hĺbky sietí a tiež spôsob vyhodnocovania výsledkov.

4.1 Metóda uchovávania informácii v SOM

Na to aby sme boli schopní odmerať pamäťovú hĺbku siete, potrebujeme si pamätať v jednotlivých neurónoch informáciu o tom, pre ktoré vstupy bol daný neurón víťazom. Každý neurón bude mať množinu vstupov, v ktorej si pamätá pre aký vstup bol počas trénovania víťazom. Nestačí však ukladať iba samotný vstup, pretože by sme prišli o historický kontext pre daný vstup, ktorý je nevyhnutný pri určovaní pamäťovej hĺbky. Preto si neukladáme iba aktuálny vstup (aktuálne písmeno), ale k posledných písmen z trénovacej sekvencie. Toto sa nazýva posuvné okno (ang. sliding window) na vstupnej množine. Po natrénovaní siete viem tieto okná vizualizovať ako hitmapu, ktorá nám vizualizuje, na ktoré vstupy (resp. okná) neuróny reagovali.

Obr. 4.1: Ukážka hitmapy

4.2 Určovanie hĺbky pamäte rekurentnej SOM

Hĺbku pamäte pre konkrétny neurón v siete určíme ako dĺžku najdlhšej spoločnej podpostupností písmen v množine posuvných okien vstupov, pre ktoré bol neurón víťazom. Dĺžku najdlhšej podpostupnosti budeme určovať od konca sekvencií v množine. Pamäťovú hĺbku celej siete následne určíme ako vážený priemer nameraných pamäťových hĺbok jednotlivých neurónov, kde váhami bude počet podpostupností v jednotlivých množinách (teda koľko krát bol nejaký neurón víťazom pre nejaký vstup). Neuróny, ktoré neboli víťazom pre žiadny vstup sa vo výpočte nezapočítavajú (majú nulovú váhu). Neuróny, ktoré boli víťazom iba pre jeden vstup, ich pamäťová hĺbka je dĺžka uloženej sekvencie, čiže dĺžka pamäťového okna. Vo výsledku majú však nízku váhu vzhľadom na iné neuróny. Priemer pamäťových hĺbok jednotlivých neurónov musí byť vážený, aby neuróny, ktoré boli víťazmi pre väčší počet vstupov mali vyššiu váhu ako neuróny, ktoré boli víťazmi pre menší počet vstupov. Po každej trénovacej epoche (prechode trénovacou množinou) budeme vedieť určiť pamäťovú hĺbku mapy. Vďaka tomu, že neuróny rekurentných sietí majú okrem normálnych váh aj kontextové váhy, ktoré uchovávajú informácie z predchádzajúcich krokov, môže sa stať, že rovnaké písmeno zo vstupu bude mať rôzne víťazné neuróny počas trénovania.

Samotná pamäťová hĺbka je relatívna a závisí aj od veľkosti posuvného okna (sliding window). Veľkosť posuvného okna musí byť dostatočne veľká. Hľadanie dostatočnej veľkosti posuvného okna je súčasťou nášho experimentu.

4.3 Určovanie hĺbky pamäte SRN

Pri SRN sme skúšali nájsť spôsob ako odmerať pamäťovú hĺbku. Navrhli sme riešenie pri ktorom si vizualizujeme vnútorné stavy siete, ale nevedeli sme rozumne kvantifikovať pamäťovú hĺbku takejto siete. Kontextovú vrstvu v SRN si môžeme predstaviť ako veľa viacrozmernú SOM, kde sa zobrazujú informácie zo vstupu vo viacrozmernom priestore.

Tradeoff medzi rozlišovaciou schopnosťou jednotlivých slov a schopnosťou zachovať podobné slová topologicky čo najbližšie pri sebe.

4.4 Výber vhodných trénovacích dát a ich reprezentácia

Trénovacie sekvencie sú tvorené písmenami anglickej abecedy (26 písmen). Spoločnou vlastnosťou týchto trénovacích množín je, že sú tvorené/generované určitým nenáhodným spôsobom (obsahujú napríklad opakujúce sa podsekvencie) a teda môžeme na nich trénovať rekurentné neurónové siete. Inými slovami dokážu v nich rekurentné neurónové siete zachytiť určité vzory, ktoré si pamätajú vo svojom kontexte. Samotné vstupy (trénovacie príklady) pre sieť sú zakódované jednotlivé písmená z trénovacej sekvencie. Keďže neurónové siete vedia najlepšie pracovať iba s číselnými hodnotami, jednotlivé vstupy zo vstupnej sekvencie kódujeme počas trénovania do 26 prvkového vektora metódou one-hot, a teda jeho prvky sú nuly a jednotka (pre každé písmeno je jednotka na unikátnej pozícii). Napríklad písmeno A bude reprezentované vektorom $[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$, písmeno B vektorom $[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$ atď. ktorý bude na vstupe neurónovej siete. Tento spôsob reprezentácie vstupov pre sieť je jednoduchý a siete s ním vedia dobre pracovať.

Kapitola 5

Experiment

Experimentom v našej práci je určovanie hĺbky pamäte a vyhodnotenie vplyvu rôznych hyper parametrov a typov kontextov na hĺbku pamäte rekurentných SOM. Cieľom nášho experimentu je aj nájdenie optimálnej kombinácie parametrov pre všetky typy porovnávaných sietí a ich vzájomné porovnanie.

5.1 Výber konkrétnych trénovacích množín pre experiment

Trénovacie množiny sme sa snažili zvoliť takým spôsobom aby sme na nich vedeli otestovať rôzne vlastnosti rekurentných sietí.

Pre náš experiment sme vybrali 3 trénovacie množiny. Hlavnou trénovaciou množinou, ktorú používame v našom experimente je náhodne generovaná sekvencia dlhá 1000 znakov, ktorá pozostáva z písmen *abcd*. Táto sekvencia obsahuje dostatočné množstvo regularít a malé množstvo unikátnych znakov a teda aj siete s relatívne malým počtom neurónov sa na nej vedia dobre natrénovať. Používame ju pri hľadaní optimálnych parametrov pre jednotlivé typy sietí.

Ako druhú trénovaciu množinu sme zvolili sekvenciu dlhú 1000 znakov, pričom znaky sú generované špeciálnym pravdepodobnostným stavovým automatom (Reberov automat). Automat generuje znaky z množiny znakov *ptvxse*. Táto sekvencia je pre SOMky ťažšia na naučenie a používame ju na overenie toho, či sú siete schopné natrénovať sa aj na zložitejších nenáhodných sekvenciách. Pri SRN je použitie tejto trénovacej množiny zaujímavejšie, vďaka vlastnostiam, ktoré SRN má.

Tretí dataset je úryvok z korpusu anglického textu. Keďže ide o reálny zmysluplný text, nie je to úplne náhodná postupnosť znakov, ale obsahuje určité vzory a opakovania, ktoré by siete mohli vedieť zachytiť vo svojej vnútorenej reprezentácii. Tento dataset používame čisto iba na overenie, či sú SOMky schopné zachytiť vzory aj v prirodzenom jazyku a teda či sú použiteľné aj pre reálne dáta.

5.2 Hľadanie optimálnych parametrov sietí

Na to aby sme mohli porovnať hĺbku pamäte rôznych typov sietí museli sme najskôr nájsť kombináciu parametrov pri ktorých daný typ siete dosahuje najnižšiu kvantizačnú chybu a najvyššie hodnoty pamäťových hĺbok.

Pri trénovaní samoorganizujúcich sa máp môžeme meniť a optimalizovať veľké množstvo parametrov.

Ako prvé sme museli správne nastaviť veľkosť okolia víťazného neurónu. Veľkosť okolia by nemala byť počas trénovania konštantná, ale mala by sa postupne zmenšovať. Vo fáze doľaďovania by mala byť čo najmenšia. Excitáciu neurónu v určitom kroku trénovania určuje excitačná funkcia. Zvolili sme spojitú excitačnú funkciu so spojitým gausovským okolím.

$$N(i^*, i) = \exp^{-\frac{d_E^2(i^*,i)}{\lambda^2(t)}} \tag{5.1}$$

Najvyššiu hodnotu má excitačná funkcia pre víťazný neurón, hodnota excitačnej funkcie pre ostatné neuróny v sieti závisí od ich euklidovskej vzdialenosti v mriežke neurónov od ich víťaza. Veľmi vzdialené neuróny majú takmer nulovú excitáciu a updatujú svoje váhy minimálne. Dôležitý je parameter λ, ktorým znižujem veľkosť okolia postupne v jednotlivých epochách. Najlepšie výsledky (najnižšie hodnoty kvantizačnej chyby) sme dosiahli pri použití nasledujúceho vzťahu pre výpočet hodnoty tohto parametra v jednotlivých epochách:

$$\lambda(t) = \lambda_i \cdot (\lambda_f / \lambda_i)^{t/\ t_{max}} \tag{5.2}$$

Kde λ_f je konštanta, ktorá určuje rýchlosť klesania. λ_i je polovica maximálnej vzdialenosti dvoch neurónov v mape, resp. vzdialenosť dvoch neurónov na koncoch diagonály mriežky neurónov. t je číslo aktuálnej epochy trénovania. Parametrer t_{max} je celkový počet epôch trénovania.

5.2.1 Parametre pre RecSOM

Pri RecSOM kontext tvorí vektor aktivít neurónov z predchádzajúceho kroku. Aktivita neurónu y je určená vzťahom:

$$y_i = \exp(-d_i) \tag{5.3}$$

Neobsahuje žiadny meniteľný parameter. Hodnota d_i je súčet vzdialenosti vstupného vektora od váhového vektora a kontextového vektora od kontextového vektora. So zmenšujúcou sa vzdialenosťou excitácia neurónu rastie exponenciálne, čo znamená, že víťaz a susedné neuróny budú mať najvyššiu excitáciu a vzdialené neuróny budú mať malú excitáciu. Výpočet kontextu pri RecSOM nevieme ovplyvnovať žiadnym parametrom.

Môžeme však meniť parameter α, ktorý sa používa pri samotnom výpočte vzdialenosti vstupu od váhového vektora a kontextu od vkontextového vektora. Tento parameter určuje váhu aktuálneho vstupu a váhu kontextu vo výslednej vzdialenosti.

$$d_i = (1-\alpha)\cdot ||x(t) - w_i||^2 + \alpha \cdot ||y(t-1) - c_i||^2 \quad c \in R^N \tag{5.4}$$

V našich experimentoch sme testovali všetky hodnoty parametra α z uzavretého intervalu $< 0, 1 >$ s krokom 0.01 (dokopy 100 experimentov).

Výsledky:

5.2.2 Activity RecSOM parametre

Pri modifikovanej verzii RecSOM v ktorej počítame aktivitu neurónov odlišným spôsobom máme tiež β parameter. Na výpočet aktivity neurónu používame gaussovskú funkciu, ktorej priebeh ovplyvňujeme pomocou β parametra. To znamená, už to nie je exponenciálna funkcia, ale má odlišný priebeh, čo ovplyvňuje výsledné hodnoty aktivácii jednotlivých neurónov. Navyše výslednú hodnotu normalizujeme na hodnoty z intervalu $< 0, 1 >$.

$$\exp^{-\beta \cdot d^2} \tag{5.5}$$

V experimente sme skúšali všetky kombinácie parametrov α a β.

5.2.3 MSOM parametre

Pri mSOM máme okrem α parametra aj β parameter, ktorý určuje váhu váhového vektora víťaza z predchádzajúceho kroku w_{i^*} a váhu kontextu z predchádzajúceho kroku y_{i^*}. V našom experimente skúšame všetky kombinácie α a β parametrov. Hodnoty pre oba parametre sú z uzavretého intervalu $< 0, 1 >$ s krokom 0.1 (100 experimentov).

5.2.4 Decaying MSOM parametre

Pri modifikovanej verzii mSOM máme, podobne ako pri mSOM, β parameter, ktorý vo vzťahu pre výpočet kontextu určuje váhy jednotlivých minulých vstupov. Čím vyššia je hodnota β parametra tým vyššiu váhu majú jednotlivé minulé vstupy. Opäť skúšame všetky kombinácie α a β parametrov. Hodnoty pre oba parametre sú, podobne ako pri mSOM, z uzavretého intervalu $< 0, 1 >$ s krokom 0.1

Experimentami sme zistili, že na hĺbku pamäte siete majú vplyv iba niektoré z nich. Najdôležitejšie parameters, ktoré vplývajú na hĺbku pamäte neurónovej siete sú parameters α a β vo vzťahu pre výpočet vzdialenosti vstupného vektora od určitého neurónu v siete (čiže od jeho váhového a kontextového vektora). Tieto dva parameters určujú pomer dôležitosti aktuálneho vstupu a dôležitosť kontextu, pri výpočte vzdialenosti (kvantizačnej chyby).

Experiment prebiehal nasledujúcim spôsobom:

- Vybrali sme vhodnú trénovaciu sekvenciu, počet epôch trénovania a dostatočnú veľkosť pamäťového okna
- Spustili sme trénovanie na všetkých kombináciach týchto dvoch parametrov s krokom 0.1
- Hodnoty pamäťovej hĺbky sme ukladali do súboru
- Na záver sme vykreslili heatmapu, ktorá znázorňuje aká bola pamäťová hĺbka pre rôzne kombinácie parametrov.

Počet epôch sme určili na základe kvantizačnej chyby. Počet epôch sme postupne zvyšovali a keď kvantizačná chyba prestala signifikantne klesať, resp. dosiahla svoje minimum zastavali sme ho na tejto hodnote a ďalej nezvyšovali. SOM sa dokážu relatívne rýchlo učiť a teda počet epôch nemusí byť vysoký, čo je veľkou výhodou pri experimentovaní, kedže trénovanie netrvá príliš dlhú dobu a tým pádom sme mohli vyskúšať viac kombinácii a modifikácii.

Dostatočnú veľkosť pamäťového okna sme určili podobne ako počet epôch. Parameter sme postupne zvyšovali a zastavili na hodnote, keď pamäťová hĺbka siete prestala stúpať, čiže veľkosť pamäťového okna už neovplyvňovala hĺbku pamäte a ďalšie zvyšovanie parametra nemalo zmysel.

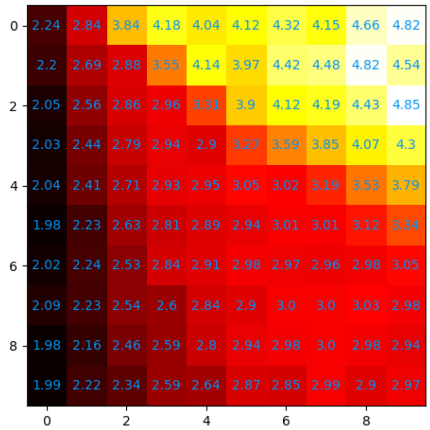
Na základe tohto sme zistili, že pre každý typ siete sú ideálne hodnoty týchto parametrov odlišné.

5.3 Vyhodnotenie rôznych kombinácii alpha a beta parametrov

The table 5.1 is an example of referenced LaTeXelements.

Parameter	Hodnota
1	6
2	7
3	545
4	545
5	88

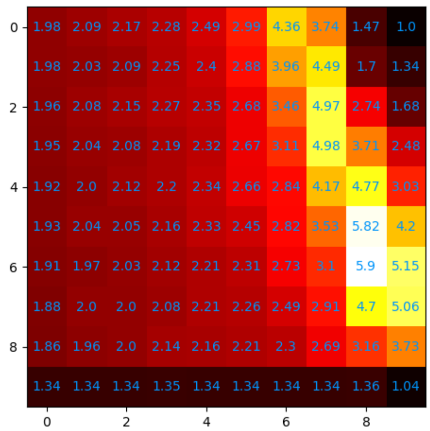
Tabuľka 5.1: Parametre RecSOM siete



Obr. 5.1: Rec SOM results

Parameter	Hodnota
1	6
2	7
3	545
4	545
5	88

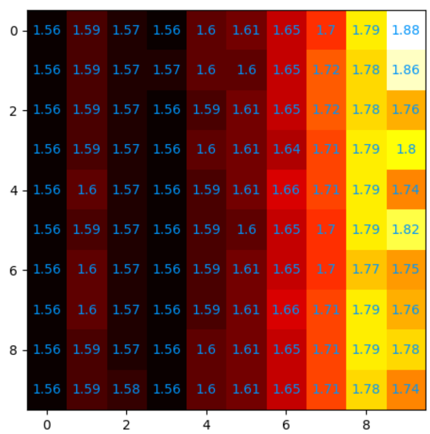
Tabuľka 5.2: Parametre mSOM siete



Obr. 5.2: MSOM results

Parameter	Hodnota
1	6
2	7
3	545
4	545
5	88

Tabuľka 5.3: Parametre leaky mSOM siete



Obr. 5.3: Leaky MSOM results

5.4 Experiment so SRN a Reberovým automatom

Kapitola 6

Záver

6.1 Limity a nedostatky riešenia

6.2 Možnosti ďalšej práce

Bibliografia

[1] Jeffrey L. Elman. “Finding structure in time”. In: *COGNITIVE SCIENCE* 14.2 (1990), s. 179–211.

[2] Jiang Guo. “Backpropagation through time”. In: *Unpubl. ms., Harbin Institute of Technology* (2013).

[3] Teuvo Kohonen. “Essentials of the Self-organizing Map”. In: *Neural Netw.* 37 (jan. 2013), s. 52–65. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2012.09.018. URL: http://dx.doi.org/10.1016/j.neunet.2012.09.018.

[4] H. Ritter a T. Kohonen. “Self-organizing Semantic Maps”. In: *Biol. Cybern.* 61.4 (aug. 1989), s. 241–254. ISSN: 0340-1200. DOI: 10.1007/BF00203171. URL: http://dx.doi.org/10.1007/BF00203171.

[5] Marc Strickert a Barbara Hammer. “Merge SOM for temporal data”. In: *Neurocomputing* 64 (2005), s. 39–71.

[6] Thomas Voegtlin. “Recursive Self-organizing Maps”. In: *Neural Netw.* 15.8-9 (okt. 2002), s. 979–991. ISSN: 0893-6080. DOI: 10.1016/S0893-6080(02)00072-2. URL: http://dx.doi.org/10.1016/S0893-6080(02)00072-2.