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 prehlasujem, že som túto diplomovú prácu vypracoval samostatne len s použitím uvedenej literatúry a za pomoci môjho školiteľa diplomovej práce.

Poďakovanie: **Chcel by som poďakovať svojmu školiteľovi, doc. RNDr. Martin Takáčovi, PhD., za jeho pomoc, dlhé hodiny konzultácii, poskytnuté materiály a hlavne za usmernenie správnym smerom.**

Abstrakt

Použitie rekurentných neurónových sietí na spracovanie sekvenčných dát je čoraz viac dôležité v oblasti neurónových sietí. 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 hodnoty pamäťových hĺbok. Získané výsledky následne porovnávame a analyzujeme súvislosti medzi hodnotami parametrov a pamäťovou hĺbkou siete. Následne analyzujeme a porovnávame výsledky pre testované typy sietí.

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

Abstract

Usage of recurrent neural nets for processing sequential data is getting more important nowadays in the area of neural networks. In our work we implement several types of neural nets which we compare in terms of their memory span. We find optimal parameters for which different types of tested neural nets have the highest memory span values. Then we analyze the results and compare them 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“ povieme, 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 očakávame písmeno „t“ a pri druhom výskyte písmena „a“ tvrdíme, že očakávame „v“. Tento prípad sa model, ktorý nevyužíva žiadny historický kontext, nevie naučiť, pretože sa učí len asociácie vstup-výstup (aktuálne písmeno - nasledujúce písmeno) a nevyužíva žiadnu ďalšiu 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 budeme merať a porovnávať ich pamäťovú hĺbku. Neformálne: pamäťová hĺbka neurónovej siete vo všeobecnosti vyjadruje s akým dlhým kontextom do minulosti dokáže neurónová sieť pracovať. (formálnu definíciu pamäťovej hĺbky uvedieme v kapitole Návrh riešenia)

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

- RecSOM

- MSOM
- Rekurentná neurónova sieť s Elmanovou architektúrou

1.2 Úvod do umelých neurónových sietí

1.2.1 Perceptrón

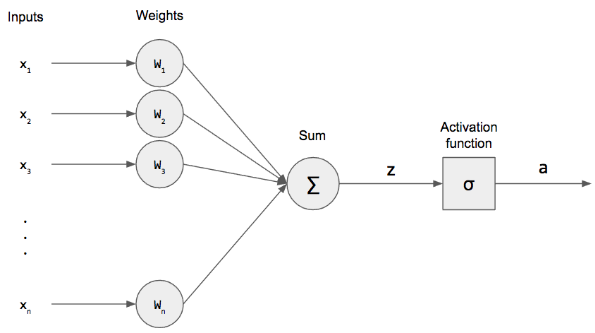
Je základným modelom neurónu, ktorý je používaný v neurónových sieťach.

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: Model perceptrónu

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:

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

Pričom p ide cez všetky vzory v trénovacej množine.

Pravidlo pre adaptáciu váh v spojitom perceptróne:

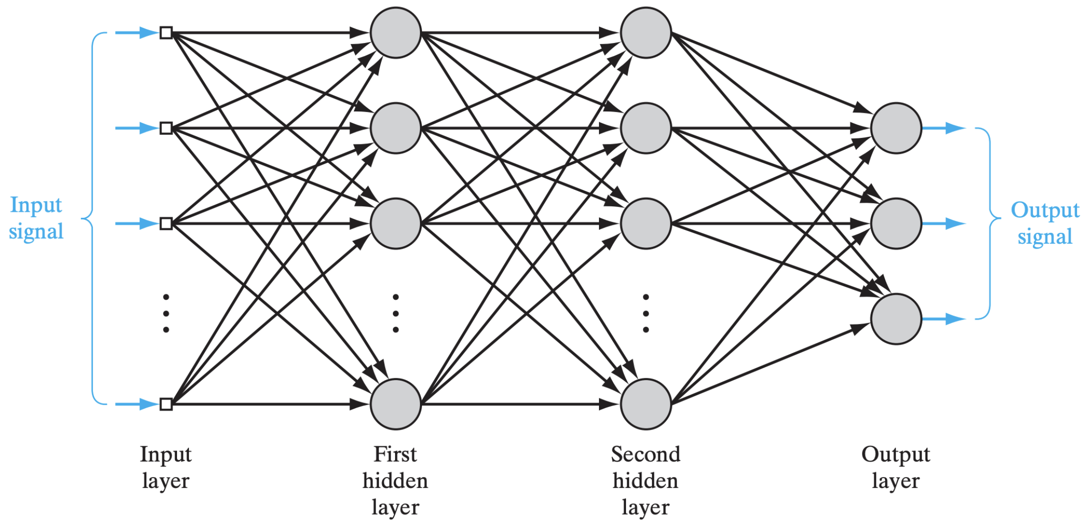
$$w_j(t+1) = w_j(t) + \alpha(d^{(p)} - y^{(p)})f'(net)x_j = w_j(t) + \alpha\delta^{(p)}x_j \tag{1.6}$$

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

Diskrétny 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. Model perceptrónu sa v strojovom učení niekedy označuje aj ako logistická regresia.

1.2.2 Viacvrstvová dopredná neurónová sieť

V našej práci sa budeme zaoberať rekurentnou neurónovou sieťou s Elmanovou architektúrou [1]. Najskôr však popíšeme základný viacvrstvový model doprednej 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: Architektúra viacvrstvovej doprednej neurónovej siete [3]

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:

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

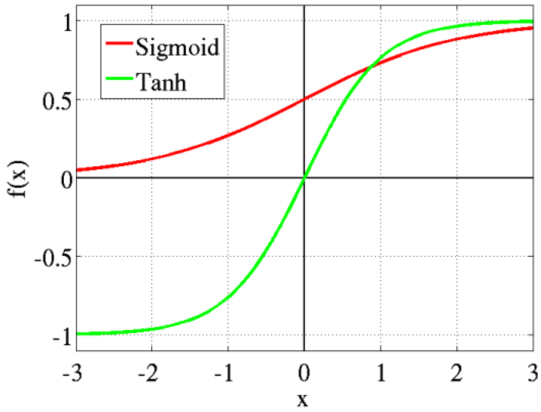
Prahové hodnoty:

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

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

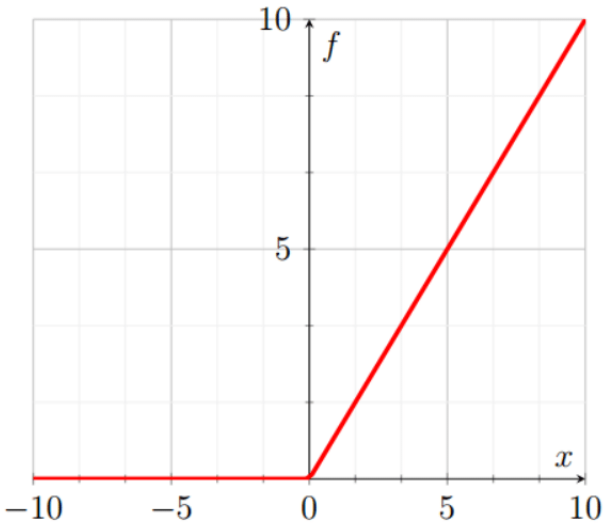
Najpoužívanejšie aktivačné funkcie sú:

- 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)

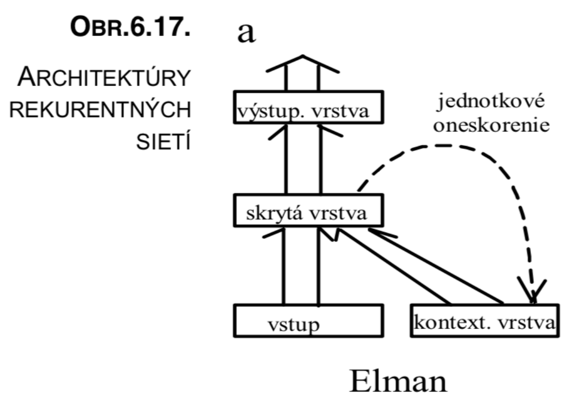
4. upravíme váhy Δw_{ik} a Δv_{kj}

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 uchovávaná informácia o aktiváciách neurónov alebo predchádzajúcich vstupoch z predošlých krokov. Neuróny, ktoré dostávajú informáciu z predchádzajúcich časových krokov 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. 1.5: Architektúra Elmanovej siete [5]

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.

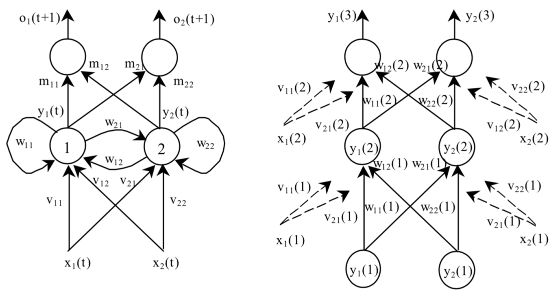
Využitie rekurentných neurónových sietí:

- 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.

Jeden z algoritmov na trénovanie rekurentných neurónových sietí 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 časových krokov do minulosti má. Počet časových krokov je daný parametrom T, ktorého veľkosť nastavujeme pred trénovaním siete. 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 [5]

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 popíšeme základný nerekurentný model SOM podľa [6].

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 sa 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 susedností (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.

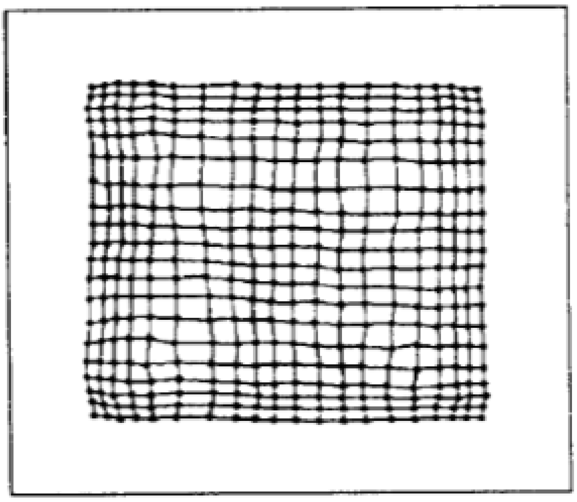
Pri učení rozlišujeme všeobecne dve fázy:

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

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.

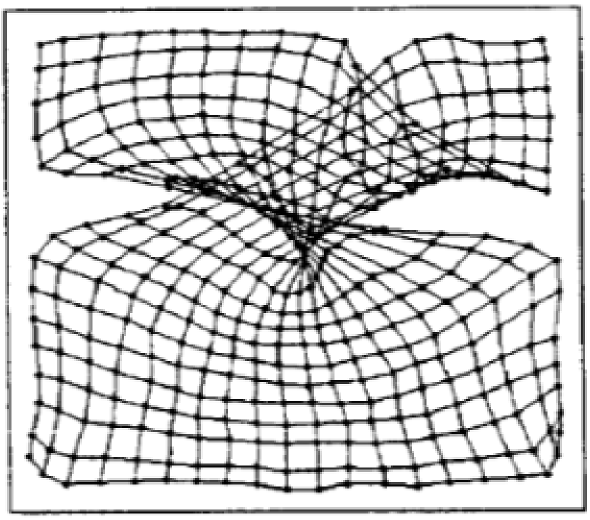
Počas trénovania SOM môžu nastať špeciálne situácie:

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



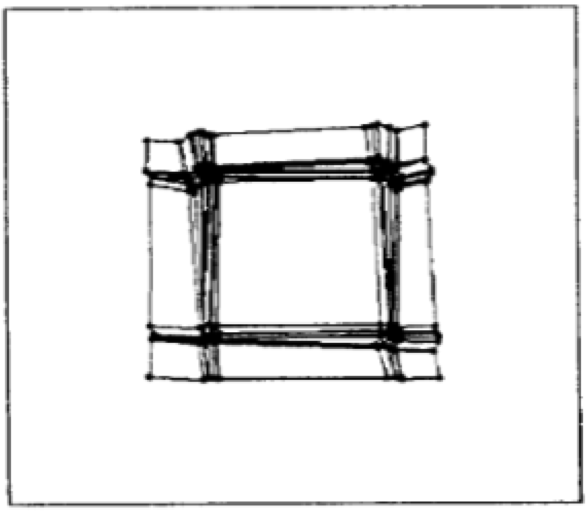
Obr. 1.7: Neúplne rozvinutá sieť [8]

- Motýlí efekt - príliš rýchle zmenšovanie okolia λ



Obr. 1.8: Motýlí efekt [8]

- Pinch efekt - príliš pomalé zmenšovanie okolia λ



Obr. 1.9: Pinch effect [8]

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 na disjunktné oblasti, ktoré tvoria tzv. Voronoiho mozaiku.

1.2.8 Rekurentné modely

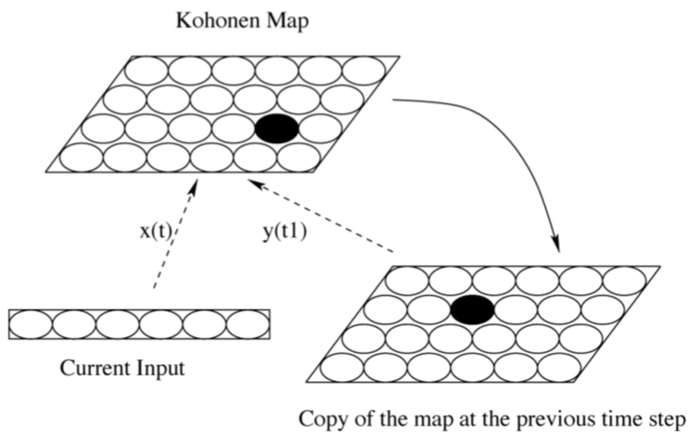
Rekurentné samoorganizujúce sa mapy sú modifikáciou nerekurentnej SOM. Rozdiel oproti nerekurentnej verzii je v tom, že vstupy sú porovnávané nielen s váhovým vektorom jednotlivých neurónov, ale aj s kontextom. Rôzne verzie rekurentných SOM sa líšia v type kontextu, ktorý je v nich použítý. V kontexte rekurentnej SOM sú 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 aktivácií 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 použitý rekurzívne na vstupný vektor $x(t)$ a tiež reprezentáciu mapy z minulého kroku $y(t-1)$. [13]



Obr. 1.10: Architektúra RecSOM [13]

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 kontext je tvorený aktiváciami neurónov celej mapy)

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

d_i je kvantizačná chyba i-teho neurónu. α je parameter, ktorý určuje 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. [9]

Chybu (vzdialenosť vstupu od neurónu) vyjadríme vzťahom (podobne ako pri RecSOM), n je dimenzia trénovacích príkladov:

$$d = (1-\alpha) \cdot ||s(t) - w_i||^2 + \alpha \cdot ||r(t) - c_i||^2 \quad x, c \in R^n \tag{1.27}$$

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:

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

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 kontextových váh 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:

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

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. Dimenzia kontextu pri MSOM je rovnaká ako dimenzia vstupov. Ak majú vstupné vektory malú dimenziu, tak aj kontext má malú dimenziu. V tomto prípade je výhodou, ak je dimenzia kontextu rovná dimenzii vstupu, avšak ak by mali vstupné vektory veľkú dimenziu (napríklad bitmapy), tak by aj kontext mal veľkú dimenziu a v tomto prípade by bola výpočtová náročnosť vysoká a bolo by vhodnejšie použiť na tento typ úlohy SOM s iným druhom kontextu. Vo väčšine prípadov je však vďaka tejto vlastnosti trénovanie MSOM rýchlejšie (výpočtovo menej náročné) a teda je možné experimentovať s väčším počtom neurónov, 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 sú hlavným cieľom mojej diplomovej práce.

Kapitola 2

Práce zaoberajúce sa hĺbkou pamäti neurónových sietí

Do podobných prác sme vybrali vedecké články, ktoré sa zaoberajú rekurentnými neurónovými sieťami aj rekurentnými samoorganizujúcimi sa mapami. Nasledujúce dve práce sa zaoberajú vlastnosťami SRN sietí:

- **Graded State Machines: The Representation of Temporal Contingencies in Simple Recurrent Networks [7]**

  Autori tejto práce analyzovali reprezentacie, ktore sa vyvinuli v skrytom stavovom priestore SRN a zistili, že výstup siete v priebehu trénovania závisí od čoraz dlhších historických kontextov. Zistili tiež, že tento typ neurónovej siete si dokáže vo svojom stavovovom priestore vytvoriť stavový automat alebo aj počítadlo.

- **Predicting the Future of Discrete Sequences from Fractal Representations of the Past [11]**

  Autori tejto práce zistili, že SRN funguje podobne ako markovoský model s variabilnou dĺžkou (VLMM). To znamená, že sieť sa naučí kontexty dĺžky potrebnej pre správny výstup. Navrhli tiež tzv. fractal prediction machine, ktorou sa inšpirovali v práci Markovian architectural bias of recurrent neural networks [10] ktorí vytvorili neural prediction machine z nantrénovanej SRN. Neural prediction machines využívajú fraktálovú vlastnosť SRN, to znamená, že postupnosti so spoločným postfixom zobrazujú blízko seba v stavovom priestore siete. Čo je zaujímavé je, že takúto vlastnosť má dokonca aj nenatrénovaná sieť, čo sa nazýva architectural bias.

Doteraz sme sa zaoberali prácami, ktoré sa zaoberali vlastnosťami SRN. Ako poslednú uvádzame prácu, ktorá sa zaoberá RecSOM a jej vlastnosťami:

- **Recursive Self-organizing Map as a Contractive Iterative Function System [12]** Autori tejto práce analyzovali RecSOM a zistili, že za určitých špecifických podmienok má, podobne ako SRN, takisto fraktálovú organizáciu. Odvodili vzorec pre maximálnu hodnotu alpha parametra (rovnica 1.22), pri ktorej je táto vlastnosť garantovaná.

Kapitola 3

Návrh riešenia

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

3.1 Pamäťová hĺbka neurónovej siete

Hĺbka pamäti vyjadruje dĺžku historického kontextu, ktorý má ešte vplyv na aktuálny výstup siete. Ak je hĺbka pamäti N, vstupy v čase menšom ako $(T - N)$ nemajú vplyv na výstup siete v čase T.

3.2 Určovanie hĺbky pamäte rekurentnej SOM

Hĺbku pamäte pre konkrétny neurón v sieti určíme ako dĺžku najdlhšej spoločnej postfixovej podpostupností znakov v množine historii posuvných okien neurónu, 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 celkový 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), čiže ako keby neexistovali a nemajú žiadny vplyv na pamäťovú hĺbku siete. Neuróny, ktoré boli víťazom iba pre jeden vstup, ich pamäťová hĺbka je rovná dĺžke uloženej sekvencie, čiže dĺžke pamäťového okna neurónu. Vo výslednej pamäťovej hĺbke majú však takéto neuróny, ktoré boli víťazom iba pre jeden vstup, nízku váhu vzhľadom na iné neuróny. Preto priemer pamäťových hĺbok jednotlivých neurónov musí byť vážený, aby neuróny, ktoré boli víťazmi pre väčší počet vstupov mali vyššiu váhu vo výslednej pamäťovej hĺbke siete, ako neuróny, ktoré boli víťazmi pre menší počet vstupov. Po každej trénovacej epoche (prechode trénovacou množinou) budeme vedieť určiť veľkosť pamäťovej hĺbky mapy.

Vďaka tomu, že neuróny rekurentných sietí majú okrem normálnych váh aj kontextové váhy a samotný kontext, ktoré uchovávajú informácie z predchádzajúcich krokov, môže sa stať, že rovnaké písmeno zo vstupnej sekvencie bude mať rôzne víťazné neuróny počas trénovania. Táto vlastnosť rekurentných SOM umožňuje to, že majú pamäťovú hĺbku.

Samotná pamäťová hĺbka je relatívna a závisí aj od veľkosti posuvného okna (sliding window), kvôli neurónom, ktoré boli víťazné iba pre jeden vstup (pre také je pamäťová hĺbka rovná veľkosti posuvného okna, t.j dĺžky uloženej sekvencie). Veľkosť posuvného okna musí byť dostatočne veľká, väčšia ako maximálna nájdená dĺžka spoločnej postfixovej podpostupnosti pre nejaký neurón, ale zároveň nie príliš veľká, aby sme zbytočne neskresľovali výsledky neurónmi, ktoré boli víťazmi iba pre jeden vstup. Hľadanie optimálnej veľkosti posuvného okna je súčasťou nášho experimentu.

3.3 Metóda uchovávania informácii v SOM

Na to aby sme boli schopní odmerať pamäťovú hĺbku SOM, 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. Všetky takéto množiny spolu tvoria dokopy tzv. receptívne pole SOM. Po natrénovaní siete viem z týchto okien vytvoriť hitmapu, ktorá nám vizualizuje, pre ktoré vstupy (resp. posuvné okná) boli neuróny víťazmi.

Obr. 3.1: Ukážka hitmapy

3.4 Pamäťová hĺbka SRN s Elmanovou architektúrou

SRN s Elmanovou architektúrou sme chceli porovnať s rekurentnými SOM najmä z toho dôvodu, že má niektoré vlastnosti spoločné so SOM. Napríklad skrytú v SRN si môžeme predstaviť ako viacrozmernú SOM, na ktorú sa zobrazujú vstupy do vysokorozmerného priestoru. Prekážkou je, že odmerať a hlavne porovnať pamäťovú hĺbku takejto siete s inými sieťami nie je jednoduché, keďže má úplne odlišnú architektúru. Pokúšali sme sa nájsť spôsob ako odmerať pamäťovú hĺbku takejto siete. Navrhli sme riešenie, pomocou ktorého vieme vizualizovať vzdialenosti medzi stavmi na kontextovej vrstve siete, ale na základe týchto informácii sme nedokázali rozumne kvantifikovať pamäťovú hĺbku takejto siete a teda ani ju porovnať s rekurentnými SOM.

3.5 Výber vhodných trénovacích dát

Trénovacie sekvencie sú tvorené písmenami anglickej abecedy (26 písmen). Spoločnou vlastnosťou týchto trénovacích sekvencií 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 a opakovania, ktoré si pamätajú vo svojom kontexte. Samotné vstupy (trénovacie príklady) pre sieť sú kódované jednotlivé písmená z trénovacej sekvencie. Keďže neurónové siete vedia najlepšie pracovať s vektormi číselných hodnôt, jednotlivé vstupné znaky 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.

3.6 Návrh experimentu

Experiment sme rozdelili na 2 hlavné časti. Prvá časť experimentu sa venuje meraniu pamäťovej hĺbky rekurentných SOM a hľadanie optimálnych parametrov, pri ktorých dosahujú najvyššie hodnoty pamäťových hĺbok.
V tejto časti sa venujeme tiež analýze výsledkov a hľadaniu súvislostí medzi veľkosťou pamäťovej hĺbky a hodnotami parametrov. Na záver sme spravili porovnanie pamäťových hĺbok všetkých testovaných typov rekurentných SOM s použitím optimálnych

váh a fixnou inicializáciou váh. V druhej časti experimentu sa venujeme pokusu s SRN a vizualizácii vnútorných stavov siete vo forme dendogramu a vyhodnoteniu výsledkov.

Experiment prebieha nasledujúcim spôsobom:

- Výber vhodnej trénovacej sekvencie, počtu epôch trénovania a vhodnú veľkosť pamäťového okna
- Trénovanie siete na všetkých kombináciach týchto dvoch parametrov
- Ukladanie hodnot pamäťovej hĺbky do súboru
- Vykreslenie heatmapy, ktorá znázorňuje aká bola pamäťová hĺbka pre rôzne kombinácie parametrov.
- Vyhodnotenie a analýza výsledkov

Kapitola 4

Implementácia

4.1 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.

4.2 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 optimálnych parametrov a umožnilo nám otestovať veľké množstvo kombinácii rôznych parametrov.

4.2.1 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)

4.3 Použité knižnice a softvér

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

4.3.1 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.

4.3.2 Matplotlib

Je knižnica na vykresľovanie grafov a vizualizáciu dát.

4.3.3 Seaborn

Je nadstavbou Matplotlib knižnice a zjednodušuje vykresľovanie rôznych grafov.

4.3.4 MultiDendrograms

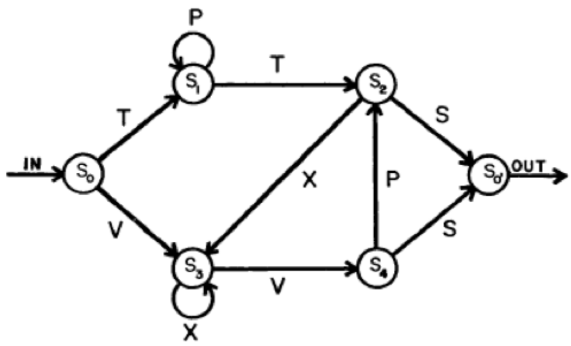
Jednoduchý java program, ktorý slúži na vykresľovanie dendrogramov z podobnostných matíc. Autorom je Sergio Gómez. V našej práci ho používame pri experimente s SRN sieťou na vizualizáciu súvislostí medzi vstupmi. `http://deim.urv.cat/~sergio.gomez/multidendrograms.php`

4.4 Algoritmus hľadania najdlhšej spoločnej postfixovej podpostupnosti viacerých reťazcov

Na hľadanie najdlhšej spoločnej podpostupnosti viacerých reťazcov som použil relatívne jednoduchý algoritmux. Na začiatku si inicializujeme veľkosť najdlhšej spoločnej podpostupnosti na nulu. V cykle prechádzame postupne všetky posuvné okná uložené v pamäťovom okne neurónu a kontrolujeme či sú znaky na i-tej pozícii od konca rovnaké. Ak áno inkrementujeme veľkosť najdlhšej spoločnej podpostupnosti. Ak nie, tak skončíme a vrátime veľkosť najdlhšej spoločnej podpostupnosti. V prípade, že pamäťové okno neurónu obsahuje iba jednu podpostupnosť, potom dĺžka najdlhšej podpostupnosti je rovná dĺžke uloženého posuvného okna. V prípade ak je pamäťové okno neurónu prázdne, dĺžka najdlhšej spoločnej podpostupnosti je nulová. Tento algoritmus je jednoduchý a dostatočne efektívny pre meranie pamäťovej hĺbky testovaných sietí. Na rýchlosť trénovania má výpočet pamäťovej hĺbky iba minimálny vplyv.

4.5 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 ďaľšieho stavu, pričom v každom stave je 50% šanca na prechod do jedného z možných stavov.



Obr. 4.1: Schéma reberovho automatu, Zdroj:`http://www.replicatedtypo.com/cultural-inheritance-in-studies-of-artifical-grammar-learning/3352.html`

Kapitola 5

Experiment

Experimentom v našej práci je meranie hĺbky pamäte a vyhodnotenie vplyvu rôznych hyperparametrov 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, z ktorého sú odstránené špeciálne znaky a medzery. 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.

Ďalšie parametre:

- **Rýchlosť učenia**
  Rovnako ako okolie aj rýchlosť učenia siete by mala počas procesu trénovania postupne klesať. Na začiatku chceme aby sa váhy menili čo najviac a ku koncu učenia chceme aby sa doľadovali iba detaily. Máme na výber 2 možnosti. Postupné znižovanie rýchlosti učenia po každom vstupe, alebo postupné znižovanie rýchlosti učenia po jednotlivých epochách, pričom počas každej epochy je rýchlosť učenia konštantná. V našich experimentoch sme dosiahli lepšie výsledky postupným znižovaním rýchlosti učenia po každom vstupe. Hodnoty rýchlosti učenia máme z intervalu $< 0, 1 >$.

- **Veľkosť posuvného okna**
  Vhodnú veľkosť posuvného okna sme určili postupním zvyšovaním jeho veľkosti pokiaľ pamäťová hĺbka stúpala. Zaujímavosť, ktorú sme zistili počas experimentovania s veľkosťou pamäťového okna, bolo že ak zvolíme príliš veľké posuvné okno, výsledná pamäťová hĺbka môže byť skreslená. Pri veľkom pamäťovom okne nám môžu neuróny, ktoré majú vo svojom pamäťovom okne uloženú iba jednu sekvenciu skreslovať výslednú pamäťovú hĺbku, pretože pamäťová hĺbka takýchto neurónov je rovná veľkosti posuvného okna. Z tohto dôvodu nie je dobré nastaviť veľkosť pamäťového okna na príliš veľkú hodnotu, ale treba nájsť optimálnu hodnotu.

- **Počet neurónov a počet epôch trénovania**
  Rozmery mapy a počet trénovacích epôch sme zvolili na základe vlastností zvolenej trénovacej množiny. Tiež sme museli brať do úvahy aj časovú náročnosť trénovania sietí (najmä pri RecSOM). Potrebovali sme aby sa sieť dokázala správne natrénovať na danej trénovacej množine a zároveň, aby nám experimenty dobehli v rozumnom čase. Keďže SOMky sa dokážu natrénovať relatívne rýchlo, zvolili sme väčšie rozmery mapy (30x30) a o niečo nižší počet trénovacích epôch (10). S touto kombináciou sme dosiahli nízke hodnoty kvantizačných chýb a dobré hodnoty pamäťových hĺbok.

  Pri určovaní vhodnej veľkosti siete je to vždy kompromis medzi rozlišovaciou schopnosťou jednotlivých vstupov, schopnosťou zachovať podobné vstupy topologicky čo najbližšie pri sebe a výpočtovou náročnosťou. SOMky sa trénujú relatívne rýchlo, preto sme nemuseli použiť veľký počet epôch. Pre naše experimenty sme použili trénovanie s 10-timi epochami.

- **Inicializácia váh**
  Váhy sietí sú inicializované hodnotami z intervalu $< 0, 1)$ s normálnym rozdelením.

5.2.1 Parametre pre RecSOM

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

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

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 kontextového vektora. Tento parameter určuje váhu aktuálneho vstupu a váhu kontextu vo výslednej vzdialenosti. Nepotrebujeme extra parameter β namiesto $(1-\alpha)$, pretože matematicky sú týmto spôsobom zahrnuté všetky kombinácie týchto dvoch parametrov.

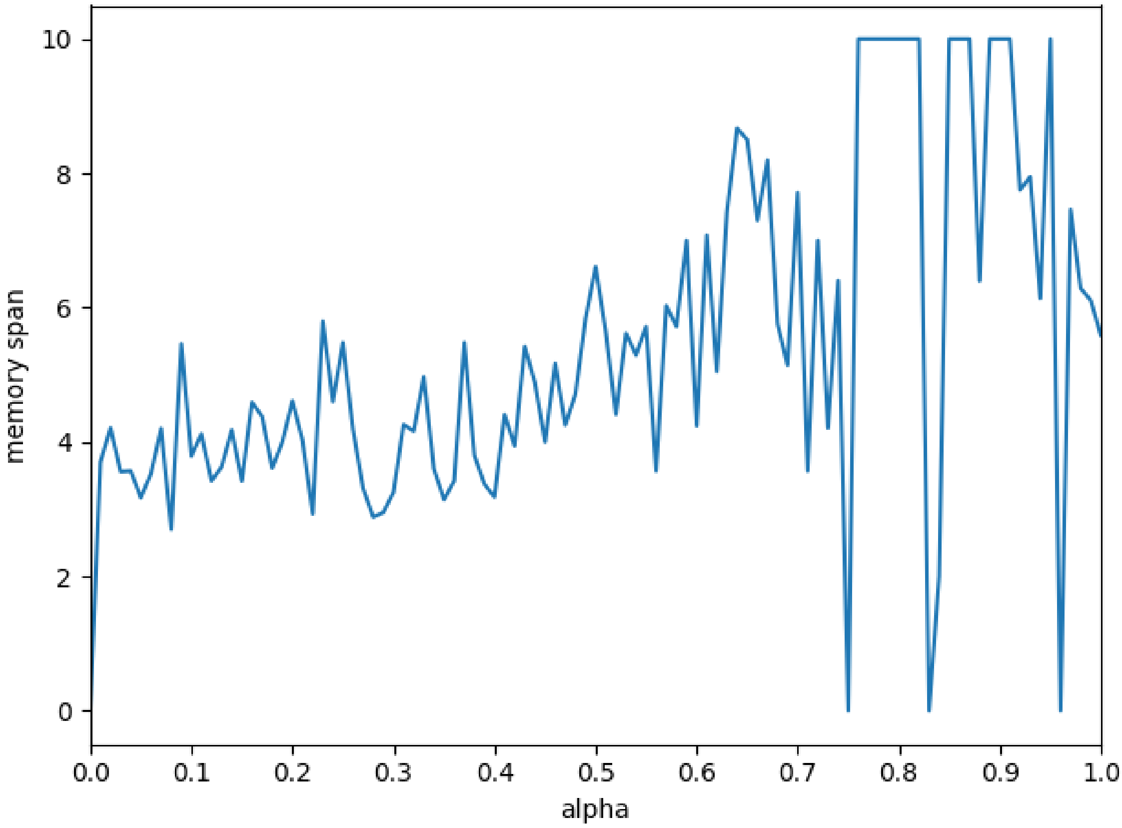
$$d_i = (1-\alpha) \cdot ||x(t) - w_i||^2 + \alpha \cdot ||y(t-1) - c_i||^2 \quad c_i \in R^N \tag{5.4}$$

V našich experimentoch sme testovali všetky hodnoty parametra α z uzavretého intervalu $< 0, 1 >$ s krokom 0.01 (dokopy 100 experimentov). Veľkosť pamäťového okna sme nastavili na hodnotu 10.

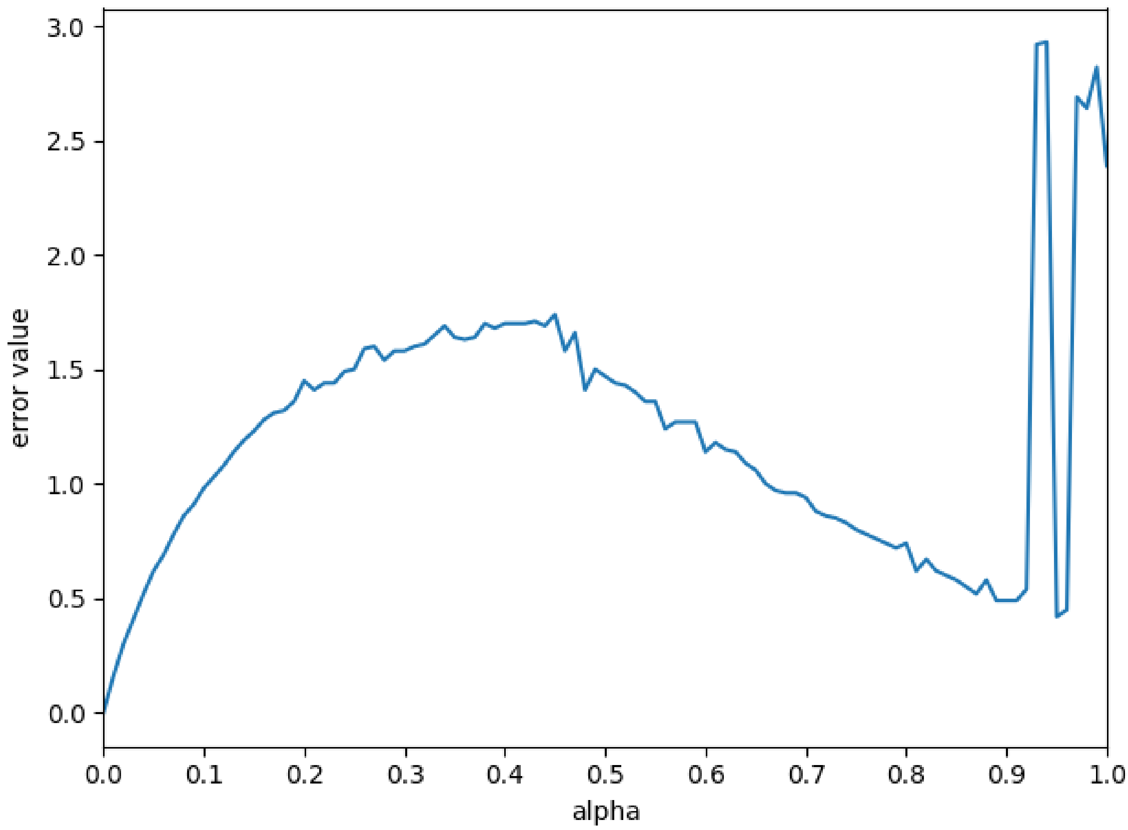
5.2.2 Výsledky pre RecSOM

Parameter	Hodnota
Hodnoty alpha	0 - 1 (krok: 0.01)
Rozmer	20x20
Počet epôch trénovania	10
Veľkosť posuvného okna	10

Tabuľka 5.1: Trénovacie parametre RecSOM siete



Obr. 5.1: RecSOM hodnoty pamäťových hĺbok



Obr. 5.2: RecSOM hodnoty kvantizačných chýb

5.2.3 Analýza výsledkov RecSOM

Keďže pri RecSOM máme iba jeden parameter, na vizualizáciu pamäťových hĺbok sme použili jednoduchý graf hodnôt. Hodnoty na x-ovej osy sú hodnoty parametra α a na y-ovej osy sú hodnoty pamäťových hĺbok. Čísla v jednotlivých políčkach sú pamäťové hĺbky na konci poslednej epochy pre určitú hodnotu parametra α.

RecSOM dosiahla najvyššie hodnoty pamäťovej hĺbky pri hodnotách parametra $\alpha = 0.76 - 0.82$, $\alpha = 0.85 - 0.87$ a $\alpha = 0.89, 0.9, 0.91, 0.95$. Najnižšiu hodnotu dosiahla keď je $\alpha = 0$, čiže keď nepracuje so žiadnym kontextom, iba s aktuálnymi vstupmi. Z grafu kvantizačných chýb vidíme, že sieť dosahuje minimálnu kvanizačnú chybu pri tých hodnotách parametra α, pri ktorých dosahuje aj najvyššiu pamäťovú hĺbku.

Vyššia kvantizačná chyba je pri stredných hodnotách parametra α, kde má aktuálny vstup aj kontext podobnú váhu a nedokáže sa na danom počte epôch správne natrénovať. So zväčšovaním parametra α, teda váhy kontextu, klesá aj kvantizačná chyba. Najvyššia kvantizačná chyba je pri hodnotách najvyšších hodnotách parametra α, je to z toho dôvodu, že sieť pracuje už iba s kontextom a aktuálny vstup má minimálny vplyv, preto sa nedokáže správne natrénovať, alebo potrebuje viacej trénovacích epôch. Príčinou môže byť tiež rozdiel v dimenziách váhových vektorov a kontextových vektorov. Kontextové vektory majú v našom prípade omnoho vyššiu dimenziu (400) ako váhové vektory (26) a teda kontextová zložka vzdialenosti má prirodzene vyššie hodnoty, čo negatívne ovplyvňuje trénovanie siete pri vysokých hodnotách parametra α. Trénovanie RecSOM bolo relatívne pomalé v porovnaní s MSOM, najmä kvôli veľkosti kontextu a sieť dosiahla uspokojivé výsledky.

5.2.4 Activity RecSOM

Keďže pri obyčajnej verzii RecSOM nevieme ovplyvniť žiadnym parametrom výpočet kontextu. Preto sme sa rozhodli vytvoriť si modifikovanú verziu RecSOM. Rozdiel oproti pôvodnej verzii je v spôsobe počítania aktivácie neurónov v kontexte. Upravili sme pôvodný vzorec

$$y_i = \exp\left(-d_i\right) \tag{5.5}$$

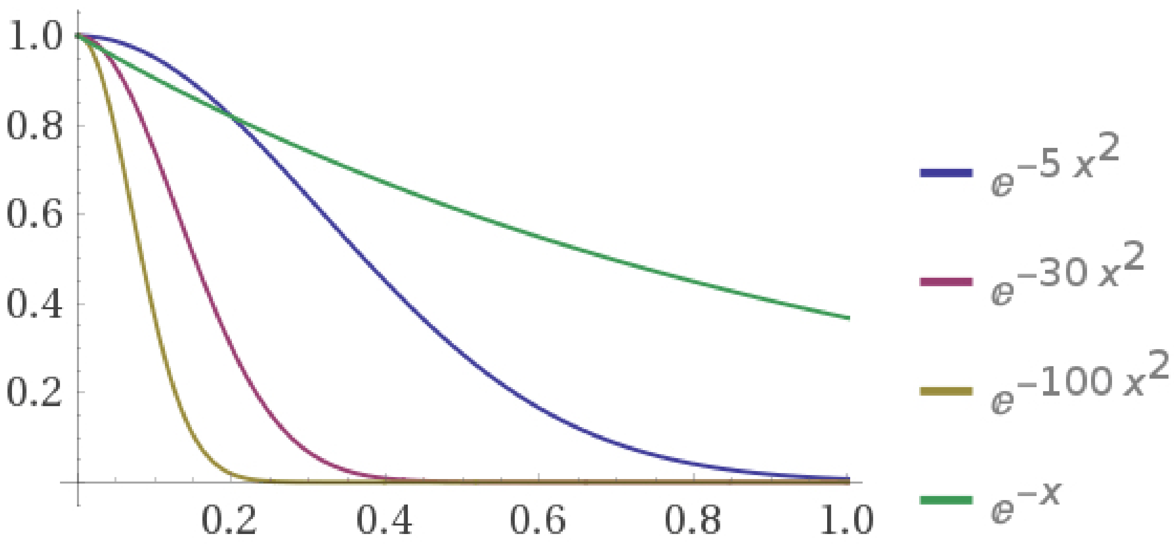
tak aby obsahoval parameter β.

$$y_i = \exp^{(-\beta \cdot d^2)} \tag{5.6}$$

Hodnota d^2 je umocnená euklidovská vzdialenosť neurónu v mriežke od víťazného neurónu. Na výpočet aktivity neurónu teda používame gaussovskú funkciu, ktorej "strmosťövplyvňujeme pomocou β parametra. To znamená, že ovplyvňujeme rozdiely medzi hodnotami aktivácie víťazného neurónu a susedných neurónov. Táto funkcia je podobná funkcii susednosti, ktorá sa používa počas trénovania.

Pre malé hodnoty parametra β hodnoty aktivácie neurónov so stúpajúcou vzdialenosťou od víťaza klesajú pomaly.

Čím je β parameter väčší tým je táto funkcia strmšia (väčší skok), čo znamená, že víťaz bude mať výrazne vyššiu hodnotu aktivácie ako neuróny, ktorých vzdialenosť d od vstupu a kontextu je vyššia.



Obr. 5.3: Grafy priebehu funkcií na výpočet aktivácií neurónov v Activity RecSOM

Na grafe je možno vidno rozdiely v priebehoch tejto funkcie s rôznymi hodnotami parametra β. Zelenou je vyznačený priebeh funkcie aktivácie, ktorý je používaný v pôvodnej RecSOM.

Samotnú hodnotu aktivácie sme chceli ešte normalizovať sumou všetkých aktivácii:

$$y_i = \frac{\exp^{(-\beta \cdot d_i^2)}}{\sum_j \exp^{(-\beta \cdot d_j^2)}} \tag{5.7}$$

Pri použití normalizácie sme dostávali signifikantne horšie výsledky ako bez použitia normalizácie. Dôvodom bolo pravdepodobne to, že vychádzali veľmi malé hodnoty aktivácií a rozdiely boli takmer veľmi malé. Z tohto dôvodu sme zostali pri pôvodnej nenormalizovanej verzii.

V prípade ak je hodnota aktivácie normovaná, potom môžeme túto hodnotu interpretovať aj nasledovne: Aktivita neurónu vyjadruje bayesovskú pravdepdobnosť, že vstup zodpovedá reprezentácii vo váhach daného neurónu.

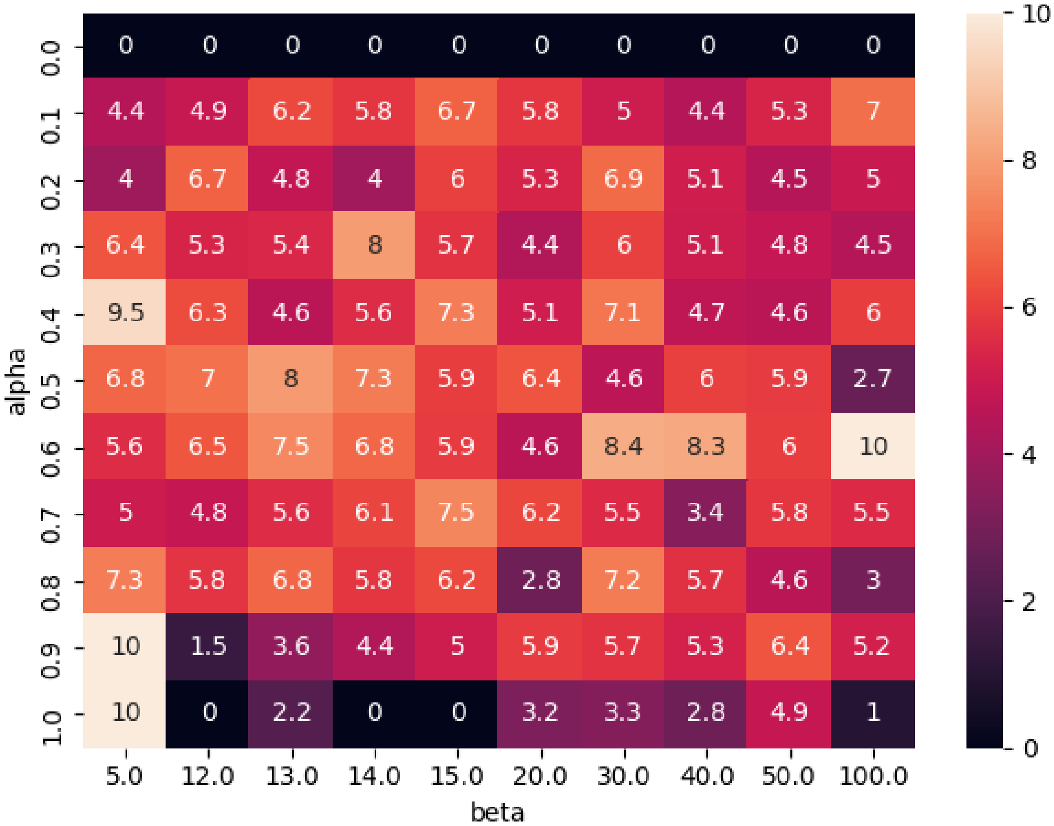
5.2.5 Activity RecSOM parametre

V našom experimente sme vyskúšali kombinácie parametrov α a β. Hodnoty parametra α sme zvolili z intervalu $< 0, 1 >$ s krokom 0.1 Hodnoty parametra β sme zvolili tak aby sme otestovali rôzne strmosti aktivačnej funkcie. Konkrétne sme použili tieto hodnoty: $[5.0, 12.0, 13.0, 14.0, 15.0, 20.0, 30.0, 40.0, 50.0, 100.0]$ Pustili sme trénovanie na všetkých kombináciach parametrov α a β.

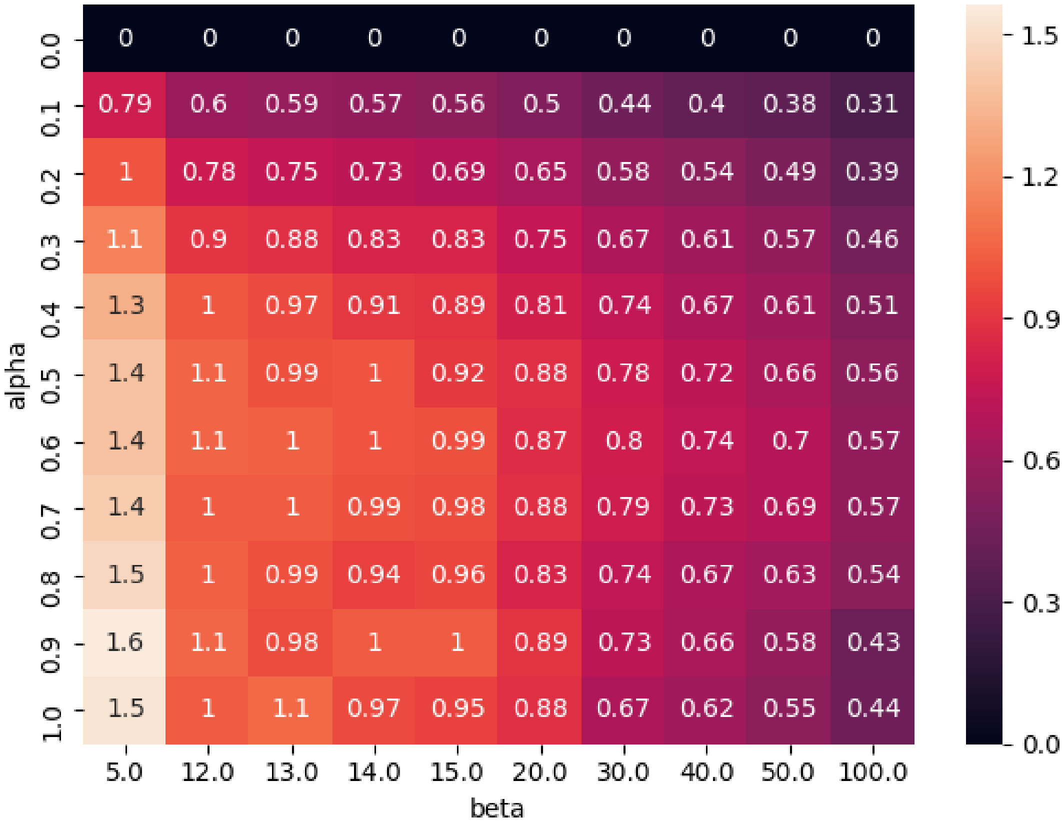
5.2.6 Výsledky pre Activity RecSOM

Parameter	Hodnota
Hodnoty alpha	0 - 1 (krok 0.1)
Hodnoty beta	[5.0, 12.0, 13.0, 14.0, 15.0, 20.0, 30.0, 40.0, 50.0, 100.0]
Veľkosť	20x20
počet epôch trénovania	10
veľkosť posuvného okna	10

Tabuľka 5.2: Parametre Activity RecSOM siete



Obr. 5.4: Activity RecSOM hodnoty pamäťových hĺbok



Obr. 5.5: Activity RecSOM hodnoty kvantizačných chýb

5.2.7 Analýza výsledkov Activity RecSOM

Na x-ovej osy sú hodnoty parametra β a na y-ovej osy sú hodnoty parametra α. Na grafe pamäťových hĺbok pre Activity RecSOM môžeme vidieť vysoké hodnoty pri hodnotách $\alpha = 1.0/0.9 a \beta = 5.0$. Tieto však nie sú relevantné a sú spôsobené tým, že niektoré neuróny boli víťazmi iba jeden krát a dĺžka najdlhších spoločných podpostupností je pre ne rovná dĺžke posuvného okna. Je to možno vidieť na receptívnom poli siete pre parametre s hodnotami $\alpha = 1.0 a \beta = 5.0$ po poslednej trénovacej epoche:

```
[["7""8""2""63""17"""""""""""""""""""""""""""""""]
["844""3""""""""6"""""""""""""""""""""""""""""]
["aacabddabd1""""""""""""""""""""""""""""""""""""""""]
["2""""""""11""""""""""""""acdabcbdba1"""""""""""""""""]
["""""""""""""""""""""""2"""""""""""""""""]
["""""6""""""""""""""""""""""""""""""""""""]
["10""""""""""""""""""""""""""""""""""""""""]
["""""""""""""""""""""""""""""""""""""""""]
["""""""""""""""""""""""""""""""""""""""""]
```

```
[""""""""""""""""""""""""""""""""""""""""]
["""""5"""""""""""""""""""""""""""""""""""]
[""""""""""""""""""""""""""""""""""""""""]
[""""""""""""""""""""""""""""""""""""""""]
[""""""""""""""""""""""""""""""""""""""""]
[""""""""""""""""""""""""""""""""""""""""]
["cadacdabcb1"""""""""""""""""""""""""""""""""""""""]
[""""""""""""""""""""""""""""""""""""""""]
["""""""""""""""""""""""""""""""cdbcadacda1"""""""""]
[""""""""""""""""""""""""""""""""""""""""]
[""""""""""""""""""""""""""""""""""""""""]]
```

Políčka s číslami označujú neuróny, ktorých pamäťové okno obsahuje uložené sekvencie, ale nemajú žiadnu spoločnú podpostupnosť. Zvyšné políčka zobrazujú najdlhšiu spoločnú podpostupnosť spoločne s celkovým počtom podpostupností v danom pamäťovom okne neurónu. Na tomto receptívnom poli je vidno, prečo je hodnota pamäťovej hĺbky vysoká.

Najvyšie (neskreslené) hodnoty pamäťovej hĺbky sieť dosahuje pri hodnotách $\beta = 30, 40$ a $\alpha = 0.6$, kde je aj relatívne nízka kvantizačná chyba. Najnižšie hodnoty pamäťovej hĺbky sieť dosahuje ak je hodnota parametra $\alpha = 1$, čiže ak sieť pracuje iba s kontextom a tiež v prípade, keď je hodnota $\alpha = 0$, kedy sieť nepracuje s kontextom do minulosti a teda ani nemôže mať pamäťovú hĺbku. Výsledky Activity RecSOM z hľadiska maximálnej hĺbky pamäte nám ukazujú, že hodnota patametra β nemá veľký vplyv na pamäťovú hĺbku siete. Zaujímavý pri Activity RecSOM je graf kvantizačných chýb, kde môžeme vidieť ako nám parametre ovplyvňujú schopnosť siete správne sa natrénovať. Ak je hodnota parametra β nízka a podiel kontextu (hodnota parametra α) vo výpočte vzdialenosti vysoká, tak má sieť vyššiu kvantizačnú chybu. Zvyšovanie hodnoty β parametra znižuje kvantizačnú chybu siete. Pri vyšších hodnotách parametra β majú neuróny blízke víťazovi výrazne vyššiu hodnotu aktivácie vďaka strmšiemu priebehu funkcie na výpočet aktivácie. Inými slovami, informácie uložené v kontexte sú viac sústredené na aktivitu víťazného neurónu. Podobne ako pri RecSOM rozdiel medzi dimenziami váhových vektorov (26) a kontextových vektorov (400) je veľký a vzdialenosť medzi kontextom a kontextovým vektorom je vo výpočte vzdialenosti je kvôli tomu vyššia. Vyššie hodnoty parametra β spôsobujú, že aktivita väčšiny neurónov v sieti je blízka nule, čo znižuje aj hodnotu vzdialenosti medzi kontextovými váhami a kontextom a teda klesá aj kvantizačná chyba. Výsledkom experimentu s Activity RecSOM je, že β parameter síce minimálne ovplyvňuje pamäťovú hĺbku siete, ale znižuje kvantizačnú chybu (teda schopnosť siete sa správne natrénovať). Čas trénovania bol podobný ako pri RecSOM.

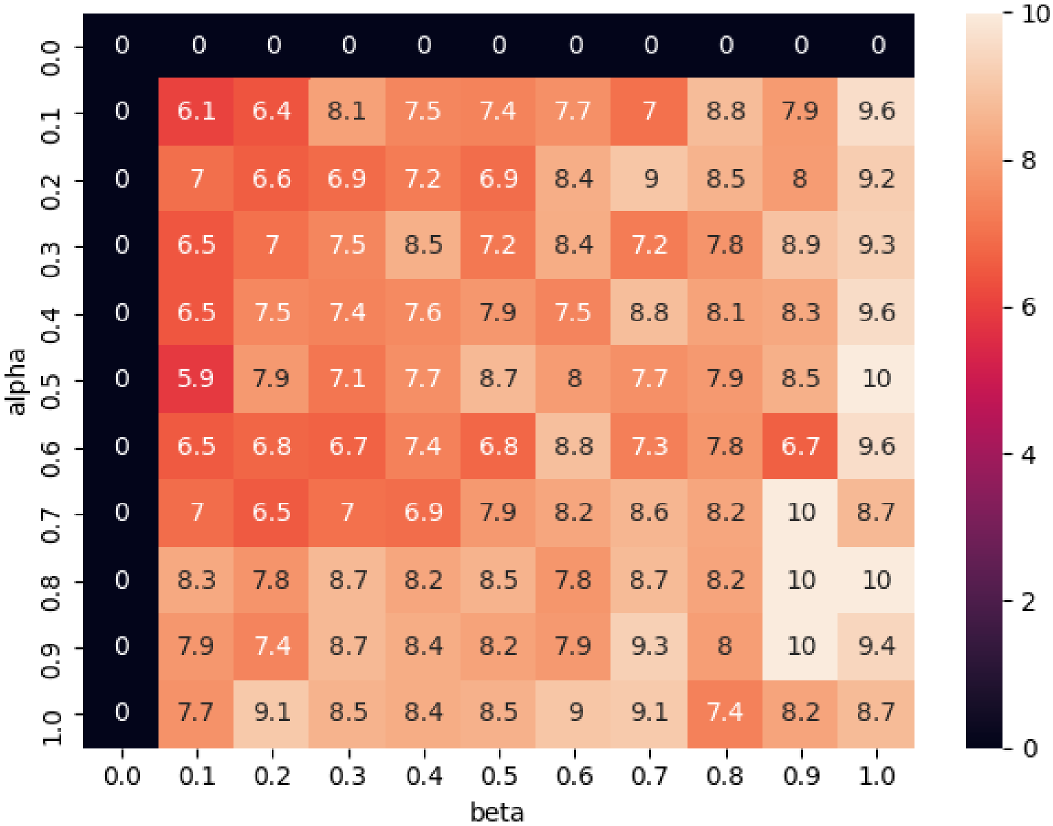
5.2.8 MSOM parametre

Pri MSOM máme okrem α parametra, používaného pri výpočte vzdialenosti, opäť aj β parameter, ktorý určuje váhu váhového vektora víťaza z predchádzajúceho kroku w_{i^*} a váhu kontextu z predchádzajúceho kroku y_{i^*} pri výpočte kontextu. Je nazývaný aj ako "zmiešavací"parameter a určuje váhu jednotlivých zložiek vlastností víťazného neurónu v kontexte. 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). Pri experimentovaní s MSOM sa snažíme zistiť aký vplyv má odlišný kontext, ktorý obsahuje iba informáciu o víťazovi z predchádzajúceho kroku, na pamäťovú hĺbku siete. MSOM má veľkú výhodu v signifikantne vyššej rýchlosti učenia, vďaka zredukovanej dimenzie kontextu.

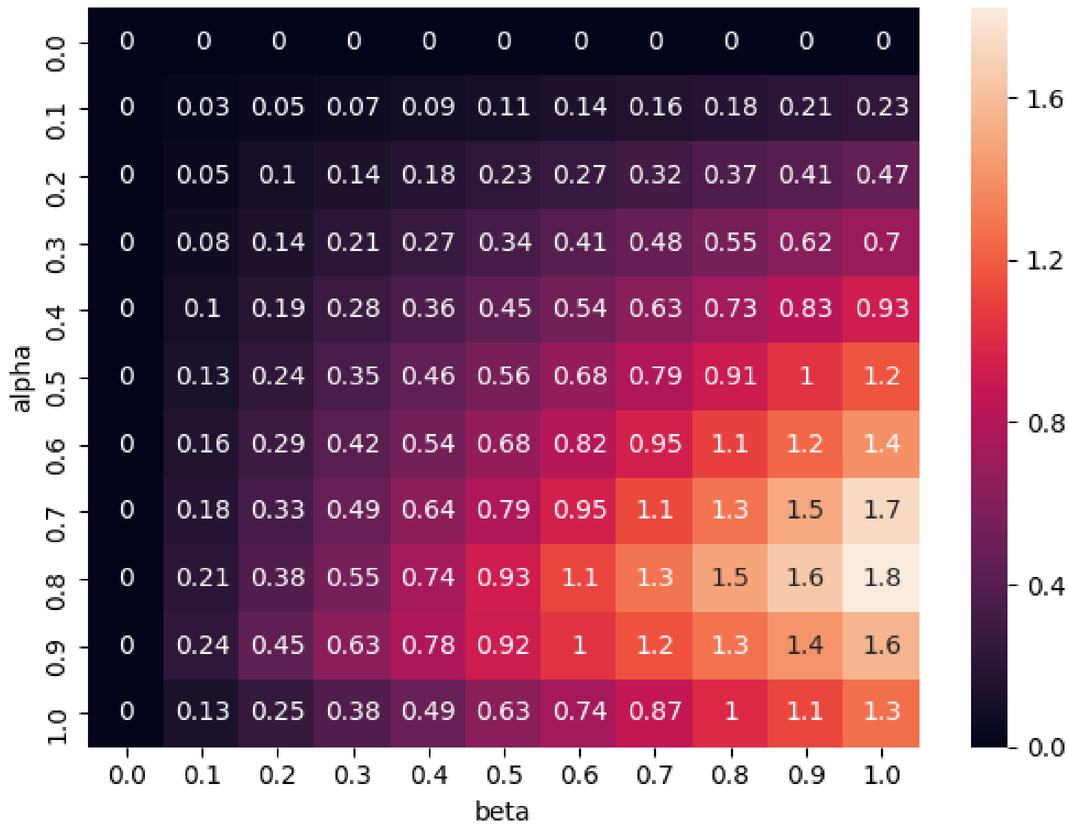
5.2.9 Výsledky pre MSOM

Parameter	Hodnota
Hodnoty alpha	0 - 1 (krok 0.1)
Hodnoty beta	0 - 1 (krok 0.1)
Veľkost	20x20
Počet epôch	10
Veľkosť posuvného okna	10

Tabuľka 5.3: Parametre MSOM siete



Obr. 5.6: MSOM hodnoty pamäťových hĺbok



Obr. 5.7: MSOM hodnoty kvantizačných chýb

Na x-ovej osy sú hodnoty parametra β a na y-ovej osy sú hodnoty parametra α. Pamäťová hĺbka pri MSOM dosahuje minimá ak je jeden z parametrov α alebo β nulový a teda sieť nemá žiadny alebo len minimálny kontext. Ak je $\alpha = 0$, tak sa zanedbáva kontextová zložka pri výpočte vzdialenosti a teda aj pri samotnom trénovaní siete a keď je $\beta = 0$ tak sa zanedbávajú váhy víťaza z predchádzajúceho kroku pri výpočte kontextu. Minimum pamäťovej hĺbky MSOM dosahuje ak sú α aj β nulové.

MSOM dosahuje v priemere vysoké hodnoty pamäťovej hĺbky, pričom je to zároveň výpočtovo najefektívnejšia verzia rekurentnej SOM pri nízko dimenzionálnych vstupoch. Ukazuje sa, že kontext redukovaný na vlastnosti víťaza z predchádzajúceho kroku neovplyvňuje negatívne pamäťvú hĺbku siete. Maximálne (neskreslené) hodnoty pamäťovej hĺbky dosahuje pri nízkych hodnotách $\alpha = 0.1 - 0.4$, čo znamená že pri výpočte vzdialenosti má vyššiu váhu vzdialenosť vstupu od váhového vektora ako vzdialenosť kontextu od kontextového vektora (Pri MSOM sú hodnoty vzdialeností porovnateľné, keďže kontext a vstup majú rovnaké dimenzie). Pri maximách je hodnota parametra β sú 1.0, čo znamená, že pre pamäťovú hĺbku MSOM je dôležité zloženie kontextu z predchádzajúceho kroku. Najvyššie hodnoty kvantizačnej chyby sieť dosahuje keď je $\alpha = 0.8$ a $\beta = 1.0$.

5.2.10 Decaying MSOM

Pre potreby nášho experimentu sme si vytvorili ďalšiu modifikovanú verziu rekurentnej SOM. Pri RecSOM kontext tvorí vektor aktivácii všetkých neurónov z predchádzajúceho kroku, pri MSOM je to kombinácia vlastností víťazného neurónu z predchádzajúceho kroku. Preto sme sa rozhodli použiť odlišný typ kontextu, ktorý bude tvorený kombináciou predchádzajúcich vstupov siete a nie stavmi siete z minulých krokov. To znamená, že kontext nie je ovplyvnený samotným procesom trénovania ani tým, čo sa sieť naučila v predchádzajúcich krokoch, ale iba samotnými vstupnými dátami. Zvyšné vlastnosti siete zostávajú rovnaké ako v iných rekurentných SOM.

Kontext počítame pomocou nasledujúceho rekurzívneho vzťahu:

$$c_t = x_t + \beta * c_{t-1} \tag{5.8}$$

ktorý v rozvinutej forme môžeme zapísať ako:

$$c = \beta^0 \cdot x_t + \beta^1 \cdot x_{t-1} + \beta^2 \cdot x_{t-2} \ddots \beta^n \cdot x_{t-n} \tag{5.9}$$

β parameter je číslo z intervalu $\beta < 1 \wedge \beta > 0$ a x_t, x_{t-1}, x_{t-2}... sú vstupné vektory z predchádzajúcich krokov. t je číslo aktuálneho kroku a n je veľkosť trénovacej množiny.

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

Čím je hodnota parametra β vyššia, tým viac informácii z predchádzajúcich vstupov v sebe kontext obsahuje. Dôležitosť dávnejších vstupov exponenciálne klesá.

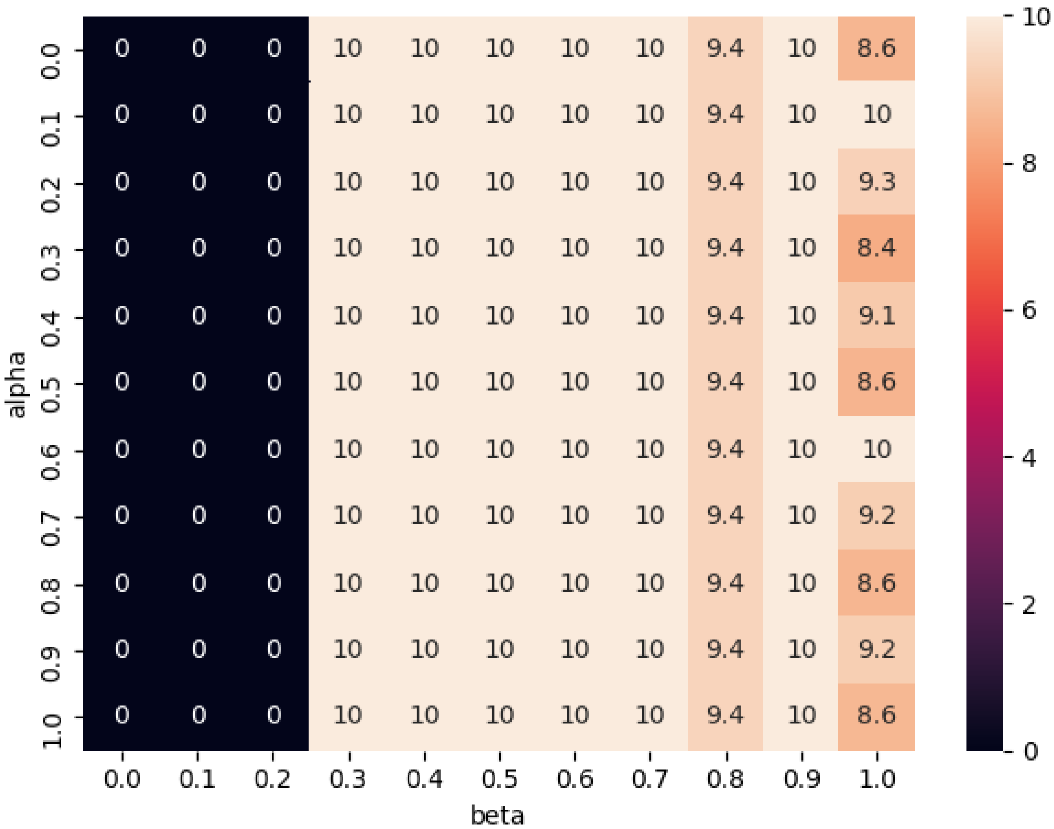
5.2.11 Decaying MSOM parametre

V experimente opäť skúšame všetky kombinácie α a β parametrov. Hodnoty pre oba parametre sú z uzavretého intervalu $< 0, 1 >$ s krokom 0.1.

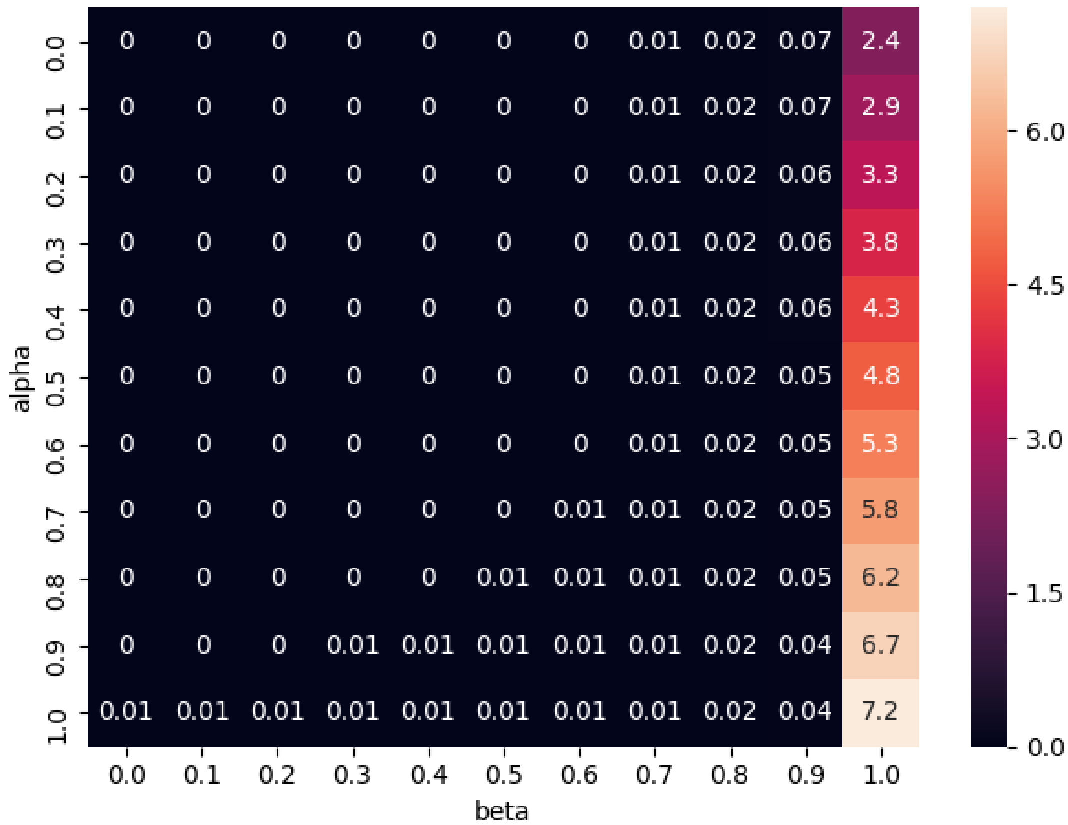
Parameter	Hodnota
Hodnoty alpha	0 - 1 (krok 0.1)
Hodnoty beta	0 - 1 (krok 0.1)
Veľkosť	20x20
Počet epôch trénovania	10
Veľkosť posuvného okna	15

Tabuľka 5.4: Parametre Decaying MSOM siete

5.2.12 Výsledky pre Decaying MSOM



Obr. 5.8: Decaying MSOM hodnoty pamäťových hĺbok



Obr. 5.9: Decaying MSOM hodnoty kvantizačných chýb

Na x-ovej osy sú hodnoty parametra β a na y-ovej osy sú hodnoty parametra α. Decaying MSOM dosahuje veľmi vysoké hodnoty pamäťových hĺbok. V receptívnom poli je však vidno, že je to pamäťovými oknami, ktoré obsahujú iba jednu sekvenciu. Čo trochu skresľuje výsledky. Na grafe kvantizačných chýb je možno vidieť, že sieť sa učí veľmi dobre v prípade, že je $\beta < 1$. Ak je $\beta = 1$ kvantizačná chyba stúpa, keďže všetky minulé vstupy siete majú rovnakú váhu, sieť sa nie je schopná správne natrénovať. Týmto experimentom sme vyskúšali aký vplyv na pamäťovú hĺbku siete má úplne odlišný druh kontextu a či sme s ním schopný sieť natrénovať. Ukázalo sa, že s takýmto kontextom dokážeme dosiahnuť podobné výsledky ako s RecSOM. Pri predchádzajúcich rekurentných sieťach kontext obsahuje reprezentáciu vplyvov jednotlivých minulých vstupov na sieť a tu je to kombinácia samotných minulých vstupov siete.

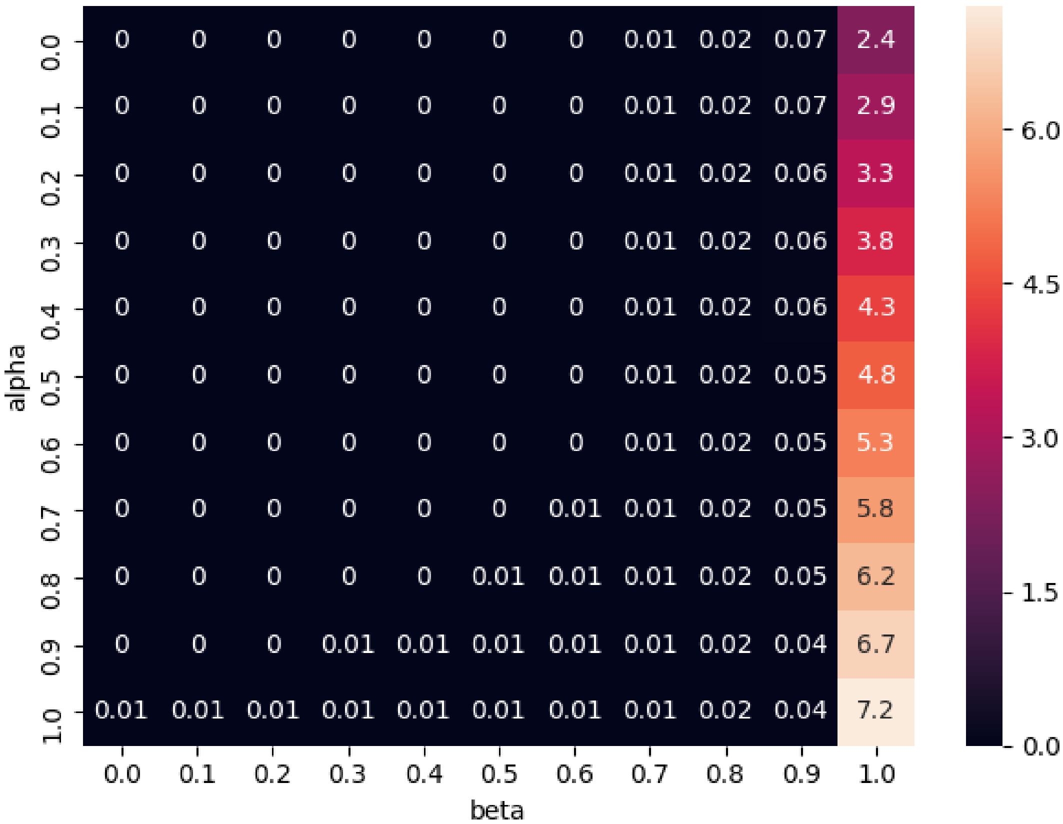
5.3 Porovnanie výsledkov SOM

Po nájdení ideálnych parametrov pre všetky 3 typy sietí, sme spustili 5 behov pre tieto kombinácie parametrov a všetky 3 trénovacie množiny, pričom sme použili rovnakú počiatočnú inicializáciu váh a spravili priemer týchto hodnôt. Zvyšné hodnoty parametrov sme ponechali rovnaké ako pri hľadaní optimálnych parametrov.

Výsledky pre hodnoty parametra $\alpha =$ a $\beta =$

5.3.1 RecSOM

- **abcd**
- **reber**
- **corpus**

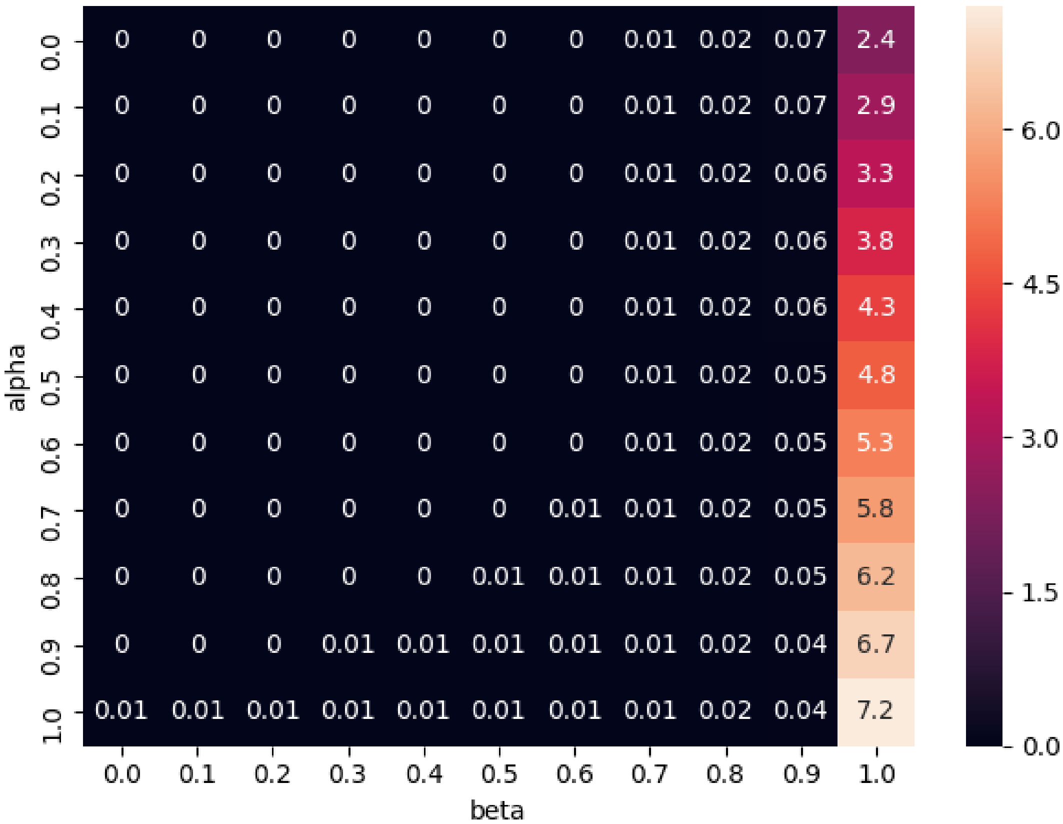


Obr. 5.10: Graf klesajúcej kvantizačnej chyby počas trénovania

5.3.2 Activity RecSOM

Výsledky pre hodnoty parametra $\alpha =$ a $\beta =$

- **abcd**
- **reber**
- **corpus**

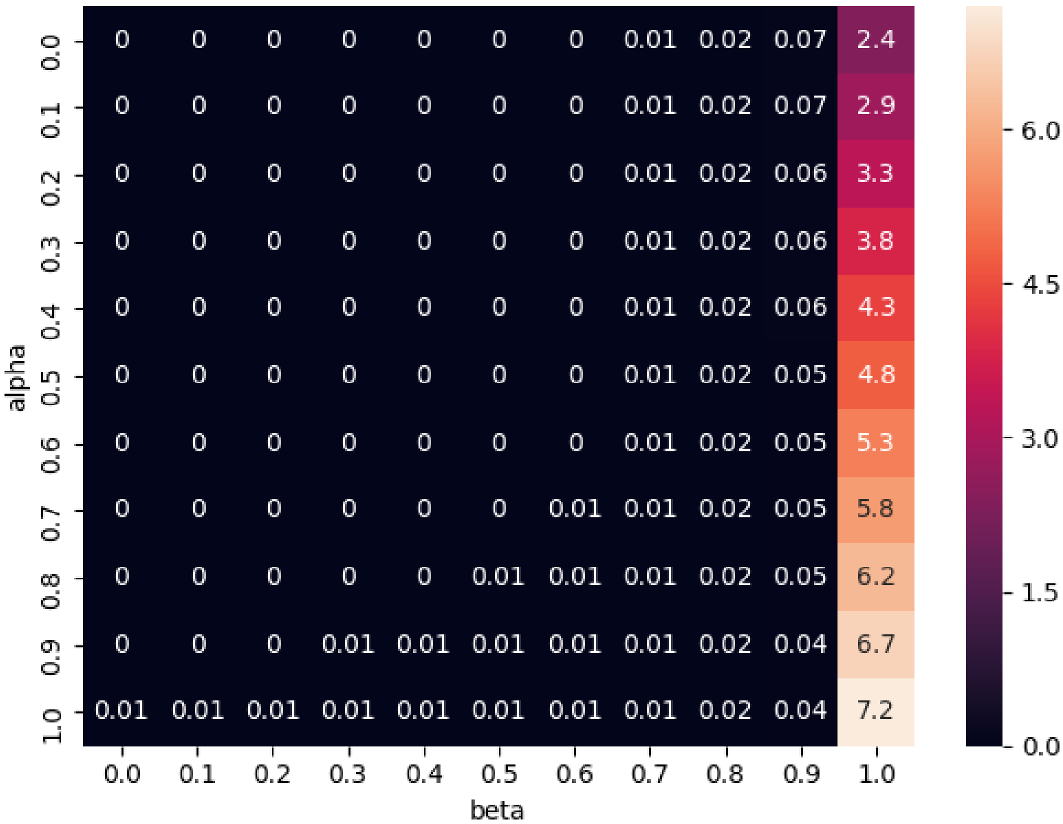


Obr. 5.11: Graf klesajúcej kvantizačnej chyby počas trénovania

5.3.3 MSOM

Výsledky pre hodnoty parametra $\alpha =$ a $\beta =$

- **abcd**
- **reber**
- **corpus**

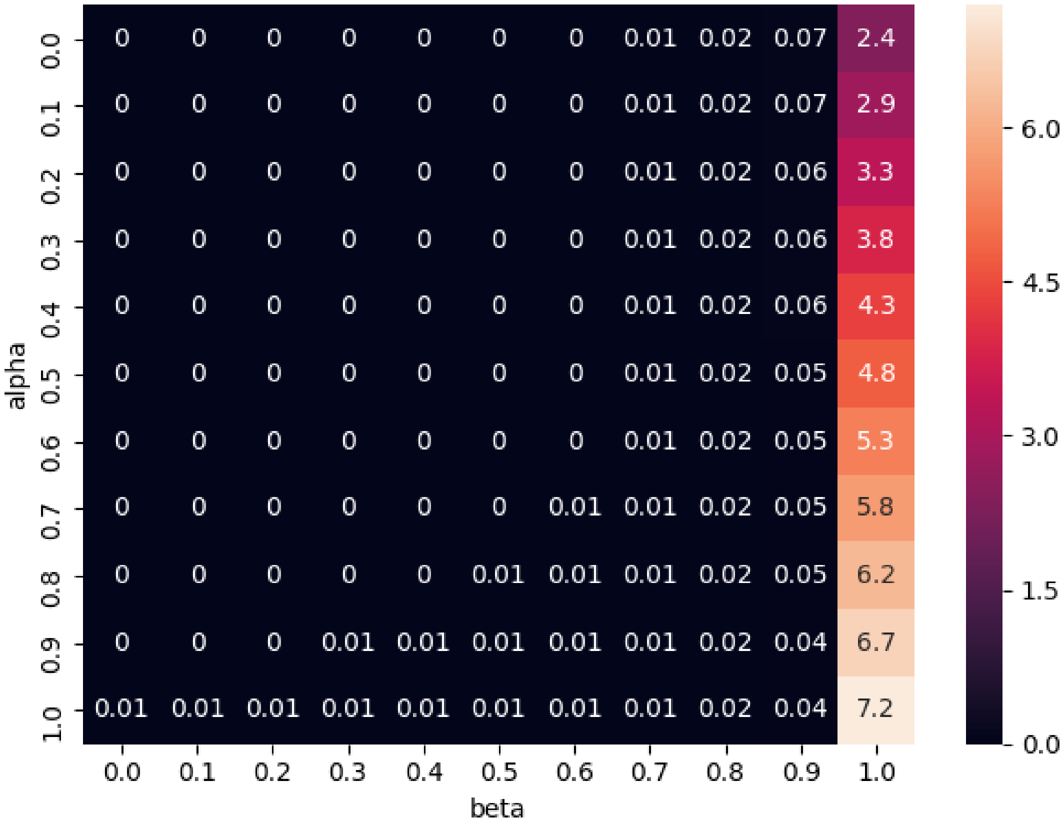


Obr. 5.12: Graf klesajúcej kvantizačnej chyby počas trénovania abcd

5.3.4 Decay MSOM

Výsledky pre hodnoty parametra $\alpha =$ a $\beta =$

- **abcd**
- **reber**
- **corpus**



Obr. 5.13: Graf klesajúcej kvantizačnej chyby počas trénovania abcd

5.4 Vyhodnotenie experimentu

Experimentami sme zistili, že na hĺbku pamäte je najviac ovplyvnená zložením samotného kontextu rekurentnej SOM. Parametre, ktoré sme testovali a ktoré vplývajú na hĺbku pamäte rekurentných SOM sú α a β α vystupuje vo vzťahu pre výpočet vzdialenosti vstupu od určitého neurónu v sieti. β zase ovplyvňuje výpočet samotného kontextu. Pre nízkodimenzionálne vstupy je najvhodnejšie použiť MSOM, ktorá je výpočtovo najefektívnejšia a dosahuje veľmi dobré výsledky. Pre vysokodimenzionálne vstupy je z hľadiska veľkosti kontextu a výpočtovej náročnosti vhodnejšie použiť RecSOM alebo Activity RecSOM.

5.5 Experiment so SRN a Reberovým automatom

SRN má niektoré vlastnosti podobné s RecSOM. V RecSOM máme vrstvu neurónov, ktorá je prepojená s kontextovou vrstvou, podobne aj v SRN s Elmanovou architektúrou máme skrytú vrstvu, ktorá je prepojená s kontextovou vrstvou. Túto analógiu je dobre vidieť ak by sme z Elmanovej siete odstránili výstupnú vrstvu a ponechali iba vstupnú, skrytú a kontextovú vrstvu, potom nám zostane architektúra RecSOM siete. Spôsob trénovania a rozmiestňovania vstupov v priestore je úplne odlišný v prípade Elmanovej siete.

Naším hlavným cieľom pri tomto experimente s SRN bolo preskúmať vlastnosti siete a pokúsiť sa nájsť spôsob merania a vyhodnotenia jej pamäťovej hĺbky. Tiež sme chceli preskúmať niekoľko zaujímavých vlastností SRN.

Prvá časť experimentu s SRN prebiehala v nasledujúcich krokoch:

- Podobne ako pri experimentoch so samoorganizujúcimi sa mapami, ako prvé sme si potrebovali vytvoriť vhodnú trénovaciu množinu. Rozhodli sme sa, že použijeme podobné zloženie trénovacej množiny ako pri samoorganizujúcich sa mapách. Vstupom je vždy jedno písmeno z náhodne generovanej sekvencie písmen *abcd* a ako očakávaný výstup je vždy nasledujúce písmeno v sekvencii. Takýmto spôsobom sme vytvorili mnnožinu trénovacích príkladov.

- Zvolili sme si vhodné aktivačné funkcie:
  Ako aktivačnú funkciu na skrytej vrstve sme použili hyperbolický tangens.

$$tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{5.10}$$

  Aktivačnú funkciu na výstupnej vrstve sme použili softmax.

- Následne sme museli overiť funkčnosť našej implementácie siete. Ako chybovú funkciu sme použili log loss. Počas testovania nám chyba klesala a sieť po natrénovaní predikovala korektné výsledky na testovacích sekvenciách.

- Keď sme mali funkčnú implementáciu SRN Elmanovej siete, natrénovali sme ju na našej trénovacej množine. Parametre, ktoré sme použili počas trénovania:

Parameter	Hodnota
veľkosť skrytej vrstvy	30
počet epôch	100
veľkosť posuvného okna na trénovacej množine	3

Tabuľka 5.5: Parametre SRN s Elmanovou architektúrou

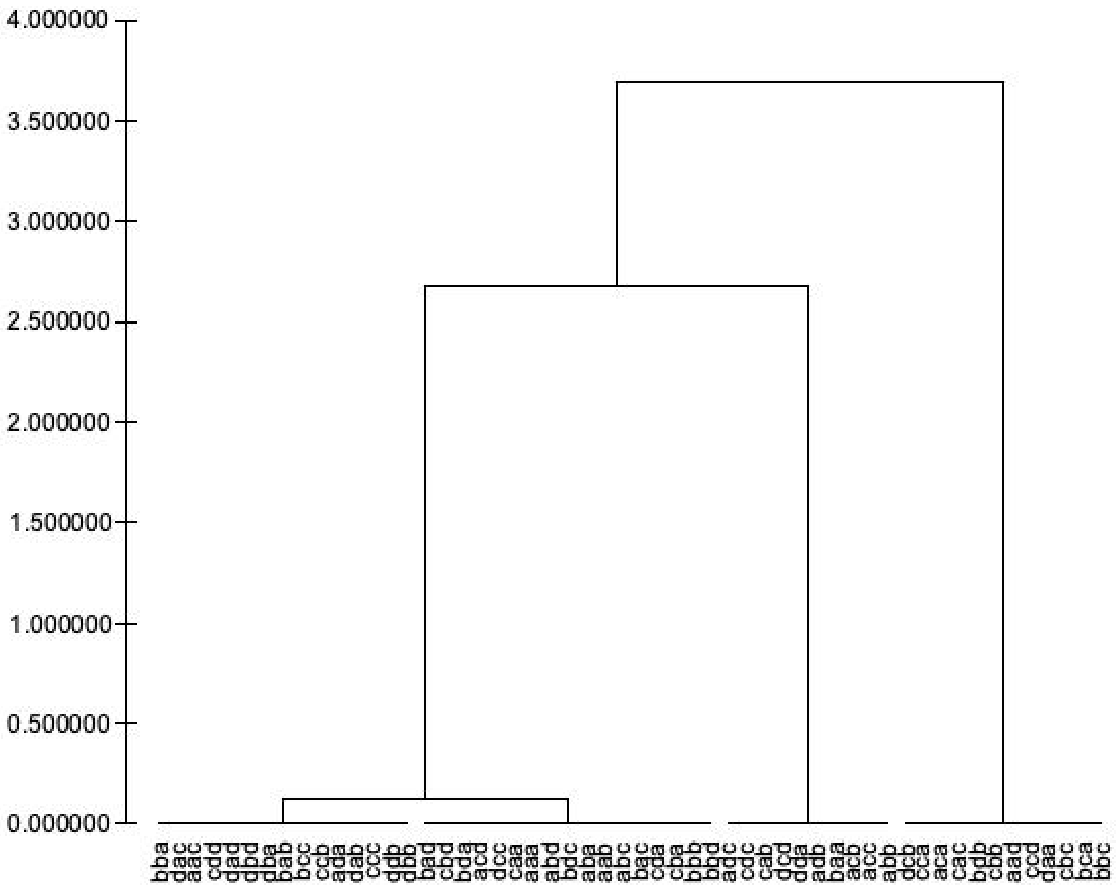
SRN si na skrytej vrstve vytvára určitú reprezentáciu vstupov. Preto sme sa rozhodli, že správne natrénovanej SRN budeme postupne predkladať písmená z trénovacej množiny, pričom si po každej predikcii, ktorú sieť spraví, uložíme aktivácie neurónov (vektor) na skrytej vrstve siete do nejakej množiny. Ku každému takémuto vektoru priradíme posuvné okno daného znaku z trénovacej množiny (podobne ako pri experimentoch so SOMkami). Po predložení všetkých znakov z trénovacej množiny sieti sme dostali dvojice posuvných okien s prislúchajúcimi aktiváciami neurónov na skrytej vrstve siete.

Tieto dáta sme potrebovali nejakým spôsobom vizualizovať, aby sme v nich vedeli identifikovať prípadné súvislosti medzi reprezentáciou vstupov na skrytej vrstve a podobnosťou samotných vstupov. Rozhodli sme sa, že použijeme vizualizáciu pomocou dendrogramu. Vizualizácie vo forme dendrogramu sa často používajú ak potrebujeme vizualizovať hierarchické klastrovanie dát.

Na vytvorenie denrogramu potrebujeme vytvoriť tzv. podobnostnú maticu (ang. similarity matrix), kde riadky aj stĺpce reprezentujú jednotlivé kontextové vektory a hodnoty v samotnej matici sú euklidovské vzdialenosti medzi týmito vektormi. Z toho vyplýva, že na diagonále sú samé nulové hodnoty (rovnaké vektory majú medzi sebou nulovú vzdialenosť).

Z takejto matice potom vieme vytvoriť stromový graf, dendrogram, ktorý vizualizuje súvislosti medzi euklidovskou vzdialenosťou jednotlivých vektorov a samotnými posuvnými oknami z trénovacej množiny. Z toho vieme potom povedať, ktoré postupnosti vstupov sú vo vnútornej reprezentácii SRN blízke.

Takáto reprezentácia nám hovorí o tom, ako sieť interne reprezentuje dáta z trénovacej množiny.



Obr. 5.14: Dendrogram pre Elmanovu sieť. Na x-ovej osy sú posuvné okná a na y-ovej osy sú vzdialenosti medzi jednotlivými vektormi

Na dendrograme vidíme, že pre sieť podobné vstupy majú rovnakú euklidovskú vzdialenosť vektorov. V našej sieti sa vytvorili 4 klustre podobnej veľkosti. Nehovorí nám to bohužiaľ nič o pamäťovej hĺbke Elmanovej siete. Nevieme ju z takejto vizualizcáie kvantifikovať.

5.5.1 Stavový automat na skrytej vrstve

Zaujímavou vlastnosťou SRN je aj to, že si na skrytej vrstve dokáže vytvoriť vlastnú reprezentáciu stavového automatu, ak je trénovaná na trénovacej množine, ktorá je tvorená reťazcom generovaným napríklad reberovým automatom [7] Vďaka tejto vlastnosti by sa mala SRN natrénovať na takejto trénovacej množine s nulovou chybou. Túto vlastnosť sme sa rozhodli overiť. Vytvorili sme si podobnú trénovaciu množinu ako pri prvom experimente a skutočne sa sieť bola schopná natrénovať takmer s nulovou chybou na sekvenciách generovaných reberovým automatom.

Záver

V našej práci sme porovnali pamäťovú hĺbku niekoľkých typov neurónových sietí. Hlavným cieľom bolo zistiť, či sa dajú rekurentné samoorganizujúce sa mapy použiť na vizualizáciu sekvenčných vstupov, teda s akým dlhým kontextom do minulosti dokážu pracovať. Chceli sme tiež navrhnúť spôsob ako merať a hlavne porovnať pamäťovú hĺbku Elmanovej siete so rekurentnými SOM-kami, prípadne nájsť súvislosti.

V našich experimentoch sme porovnávali pamäťovú hĺbku dvoch základných typov rekurentných SOM - RecSOM a MSOM. Okrem toho sme si vytvorili z oboch modelov modifikované verzie - Activity RecSOM a Decaying MSOM. Activity RecSOM umožňuje experimentovať s hodnotami aktivít neurónov, ktoré tvoria kontext siete. Upravili sme v nej vzorec na výpočet aktivity, tak aby obsahoval meniteľný parameter, ktorý môžeme nastavovať na rôzne hodnoty. To nám umožnilo preskúmať ďalšie vlastnosti RecSOM a vplyv zloženia kontextu na hĺbku pamäte RecSOM. Pri modifikácii MSOM sme sa rozhodli, že použijeme úplne odlišný kontext ako používa MSOM. Pri klasickej MSOM je kontext tvorený lineárnou kombináciou vlastností víťazného neurónu z predchádzajúceho kroku, v našej modifikovanej verzii je kontext tvorený iba kombináciou minulých vstupov a teda nie je závislý od minulých stavov samotnej siete.

Zistili sme, že pri nízkodimenzionálnych vstupoch je vhodnejšie použiť RecSOM.

V experimente s Elmanovou sieťou sa nám podarilo nájsť spôsob ako vizualizovať súvislosti medzi aktiváciami neurónov na skrytej vrstve a posuvnými oknami na trénovacej množine.

5.6 Limity a nedostatky riešenia

V našej práci sa nám nepodarilo nájsť spôsob ako zmerať a porovnať pamäťovú hĺbku Elmanovej siete. Otázne je, či takáto sieť má nejakú kvantifikovateľnú pamäťovú hĺbku a či ju vieme odmerať a kvantifikovať.

5.7 Možnosti ďalšej práce

- Medzi ďalšie možnosti patrí vyskúšanie ďalších modifikácií rekurentných SOM. Napríklad pri Acitivty RecSOM by sme mohli použiť euklidovskú vzdialenosť neurónu od víťazného neurónu a pozrieť sa na to aký vplyv by to malo na pamäťovú hĺbku siete s rôznymi hodnotami parametra β.
- Experimentovať s ďalšími kombináciami parametrov
- Vhodné by bolo lepšie kvantifikovať pamäťovú hĺbku v prípade, že neuróny rekurentných SOM majú vo svojom receptívnom poli uloženú iba jednu sekvenciu. Tiež by bolo vhodné analyzovať lepšie súvislosti medzi kvantizačnou chybou a hodnotami pamäťovej hĺbky.
- Spraviť podrobnú matematickú analýzu experimentov
- Hľadanie a skúšanie alternatívnych spôsobov ako zmerať pamäťovú hĺbku Elmanovej siete. Napríklad podrobnejšiou analýzou dendrogramov.

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] S. Haykin a S.S. Haykin. *Neural Networks and Learning Machines*. Neural networks and learning machines v. 10. Prentice Hall, 2009. ISBN: 9780131471399. URL: `https://books.google.sk/books?id=K7P36lKzI\_QC`.

[4] Teuvo Kohonen. “Essentials of the Self-organizing Map”. In: *Neural Netw.* 37 (jan. 2013), s. 52–65.

[5] Pavol a kol. Návrat. “Umelá inteligencia”. In: 45.2 (2007), s. 393.

[6] H. Ritter a T. Kohonen. “Self-organizing Semantic Maps”. In: *Biol. Cybern.* 61.4 (aug. 1989), s. 241–254.

[7] David Servan-Schreiber, Axel Cleeremans a James L. Mcclelland. “Graded state machines: The representation of temporal contingencies in simple recurrent networks”. In: *Machine Learning* (1991).

[8] M. Štefanko et al. *Úvod do teórie neurónovỳch sietí.* Iris, 1997. ISBN: 9788088778301.

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

[10] P. Tino, M. Cernansky a L. Benuskova. “Markovian architectural bias of recurrent neural networks”. In: *IEEE Transactions on Neural Networks* (2004), s. 6–15.

[11] Peter Tino a Georg Dorffner. “Predicting the Future of Discrete Sequences from Fractal Representations of the Past”. In: *Machine Learning* 45.2 (2001), s. 187–217.

[12] Peter Tiňo, Igor Farkaš a Jort van Mourik. “Recursive Self-organizing Map as a Contractive Iterative Function System”. In: (2005). Ed. Marcus Gallagher, James P. Hogan a Frederic Maire, s. 327–334.

[13] Thomas Voegtlin. “Recursive Self-organizing Maps”. In: *Neural Netw.* 15.8-9 (okt. 2002), s. 979–991.