Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

# Porovnanie niekoľkých typov rekurentných sietí z hľadiska hĺbky pamäte

Diplomová práca

2019
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: | školitel |

Bratislava, 2019
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**

**Poďakovanie**

# Abstrakt

Abstrakt - obsah

**Kľúčové slová:** rekurentné neurónové siete, hĺbka pamäte

# Abstract

Abstract

**Keywords:** neural net, machine learning, memory span

# Obsah

# Zoznam obrázkov

# 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. Kontextom sú väčšinou stavy modelu z minulých krokov. Príkladom takejto úlohy môže byť spracovanie tzv. časových radov či generovanie postupností. 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 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).

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

Konkrétne pôjde o nasledujúcim tri typy rekurentných neurónových sietí:

- Elmanova sieť
- RecSOM
- MergeSOM

## 1.2 Úvod do neurónových sietí

### 1.2.1 Perceptrón

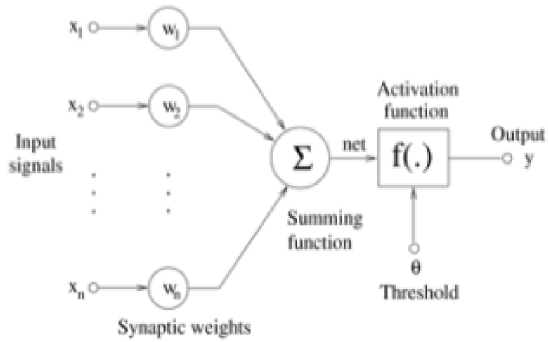
Je základným modelom neurónu, ktorý je používaný v neurónových sieťach.

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 je 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:

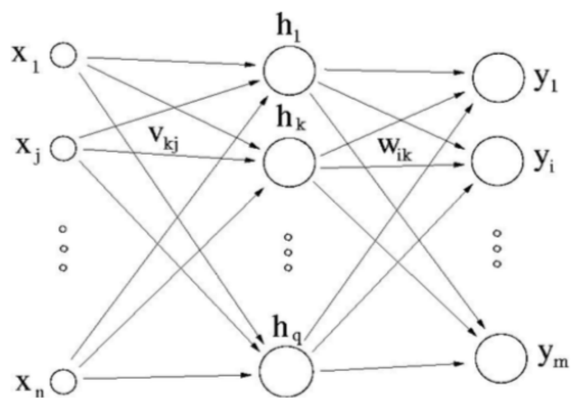
$$w_j(t+1) = w_j(t) + \alpha(d^{(p)} - y^{(p)})f'(net)x_j = w_j(t) + \alpha\delta^{(p)}x_j \tag{1.6}$$

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:

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

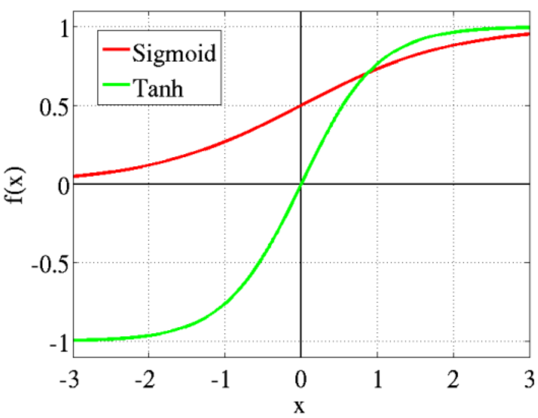
Prahové hodnoty:

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

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

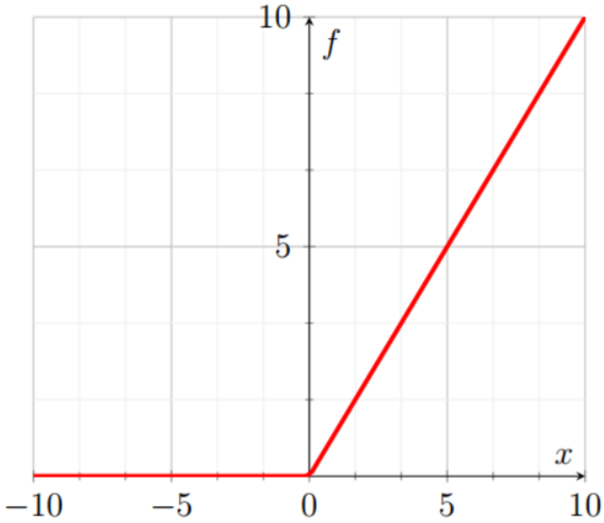
Najpoužívanejšie aktivačné funkcie sú:

- 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:

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

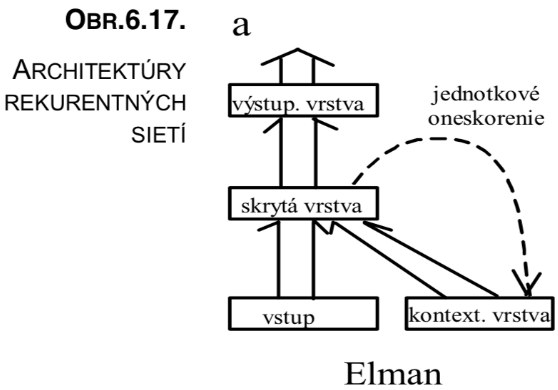
### 1.2.3 Trénovací algoritmus dopredných neurónových sietí

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 $\delta_i$ a $\delta_k$ (spätný krok)
4. upravíme váhy $\Delta w_{ik}$ a $\Delta v_{kj}$
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 sa uchováva informácia o aktiváciách neurónov 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 mojej práci budem skúmať vlastnosti a pamäťovú hĺbku rekurentnej siete s Elmanovou architektúrou, ktorá je znázornená na nasledujúcom diagrame.




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.

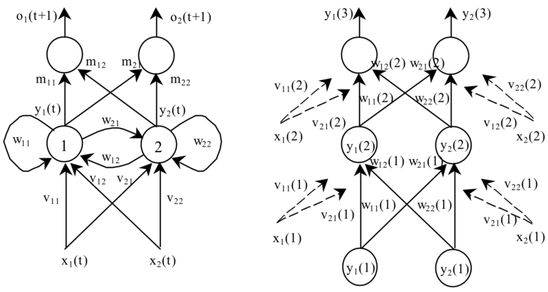
Využitie rekurentných neurónových sietí:

- 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 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 $\alpha$, 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. Je to 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 spriemerujeme. Táto by mala po každej epoche učenia (po natrénovaní a adaptácii váh na celej trénovacej množine) postupne klesať.

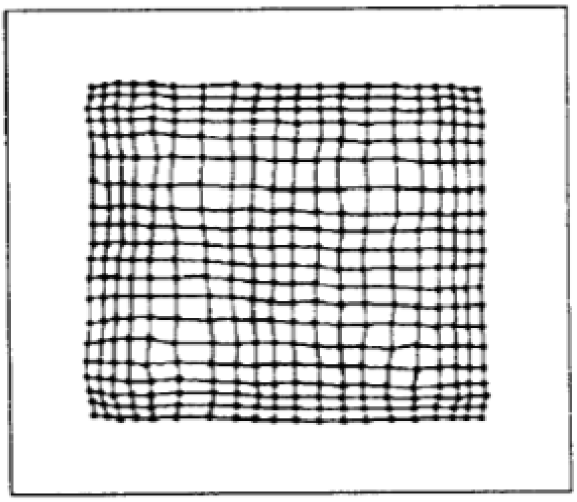
Pri učení rozlišujeme všeobecne dve fázy:

- usporiadavanie - s časom klesá veľkosť okolia víťazných neurónov
- dolaďovanie - veľkosť okolia sa zafixuje na nejakej malej hodnote až pokým učenie neskončí.

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.

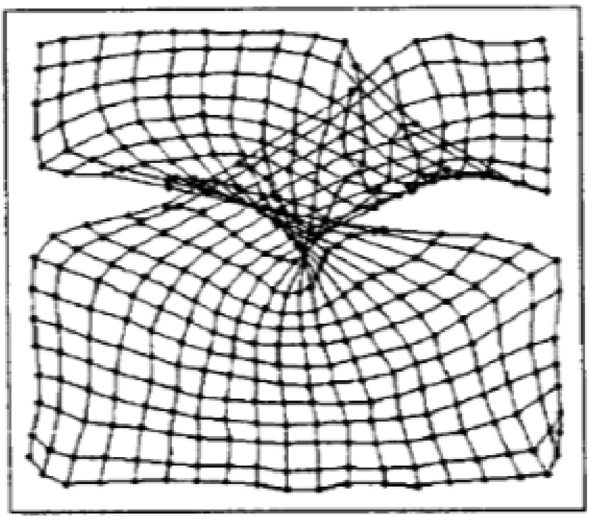
Počas trénovania SOM môžu nastať špeciálne situácie:

- Sieť je neúplne rozvinutá - príliš rýchle zmenšovanie rýchlosti učenia $\alpha$



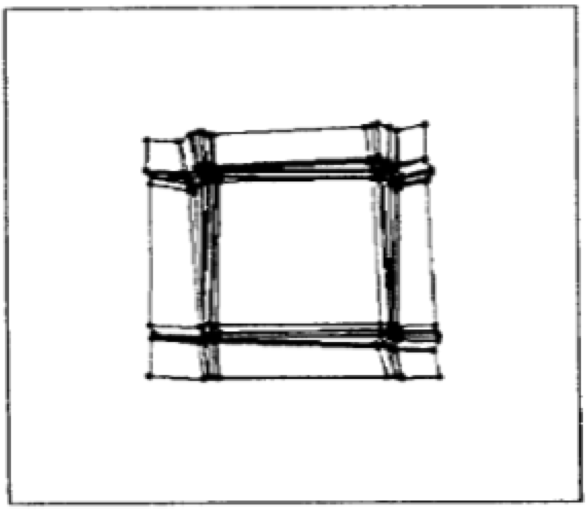
Obr. 1.7: Neúplne rozvinutá sieť

- Motýlí efekt - príliš rýchle zmenšovanie okolia $\lambda$



Obr. 1.8: Motýlí efekt

- Pinch efekt - príliš pomalé zmenšovanie okolia $\lambda$



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 iba v type kontextu, ktorý je v nich použítý. V kontexte je spravidla uložený stav siete alebo časti siete z minulého časového kroku.

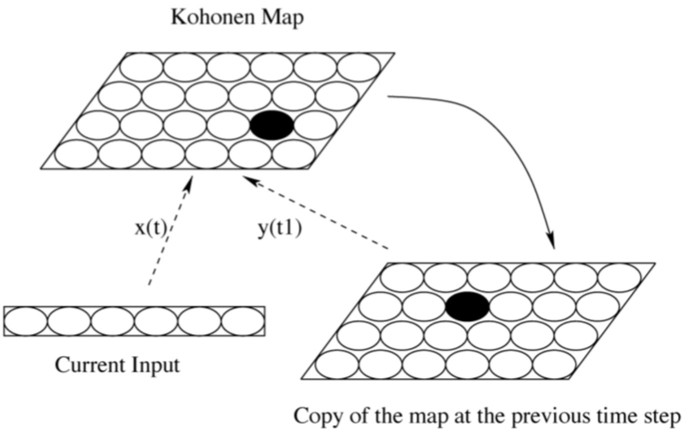
V mojej práci budem porovnávať 2 základné typy rekuretných SOM.

- 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)

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

$\alpha$ a $\beta$ 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):

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

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

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

Pri RecSom majú kontext aj kontextové váhy veľkú dimenziu (rovnú počtu neurónov v sieti), čo môže spomaľovať proces trénovania. Výhodou môže byť, že kontext RecSOM obsahuje veľa informácií a teda môže mať v niektorých prípadoch lepšie vlastnosti.

### 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:

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

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:

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

$\beta$ 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. Kontextom pri mSOM je teda lineárna kombinácia váhového vektora a kontextového vektora víťazného neurónu z predchádzajúceho kroku. Pravidlá pre zmenu váh a kontextových váh zostávajú rovnaké ako pri RecSom, resp. SOM. Líšia sa iba v kontexte.

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

Pravidlo pre zmenu kontextových váh:

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

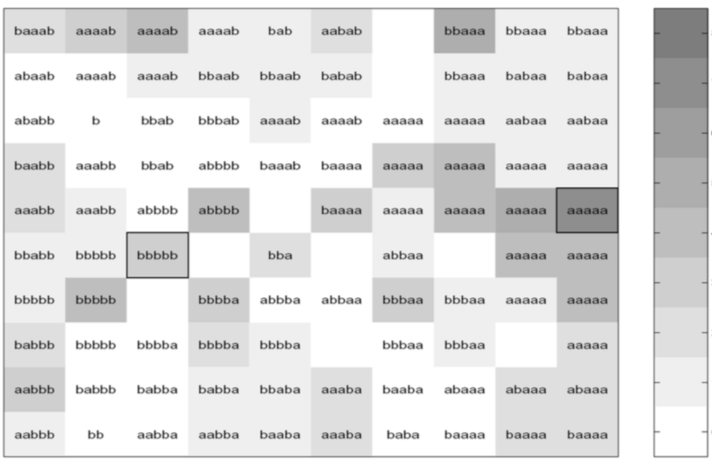
Kontext pri mSOM obsahuje odlišné informácie ako pri RecSom. Na rozdiel od recSom, kde kontext tvorí vektor aktivít neurónov z predchádzajúceho kroku, pri mSOM je kontext tvorený iba lineárnou kombináciou vlastností víťazného neurónu z minulého kroku. Má však menšiu dimenziu (dimenziu vstupu), vďaka čomu je trénovanie mSOM rýchlejšie a teda je možné použiť viacrozmerné mapy a viac epoch trénovania. Rozdielne typy kontextov a ich vplyv na pamäťovú hĺbku rekurentných sietí je jedným z cieľov mojej diplomovej práce.

# Kapitola 2

# Návrh riešenia

## 2.1 Meranie hĺbky pamäte samorganizujúcich sa máp

Ako trénovaciu množinu budem používať sekvenciu písmen abecedy (26 písmen) zostavenú nenáhodným spôsobom (napr. tvoriacu anglické slová z nejakého korpusu). Vstupmi (trénovacie príklady) pre sieť budú zakódované jednotlivé písmená z trénovacej sekvencie. Písmená kódujem do 26 prvkového vektora, ktorého prvky budú nuly a jednotka (pre každé písmeno na inej pozícii). Každý neurón bude mať množinu, v ktorej si bude pamätať pre aký vstup bol víťazom. Nebude si však ukladať iba konkrétne písmeno zo vstupu, ale $k$ posledných písmen z trénovacej množiny (tzv. sliding window). Z toho si viem ďalej vytvoriť hitmapu, ktorá mi bude vizualizovať, na aké vstupy neuróny reagovali.




Obr. 2.1: Hitmapa

Mierou hĺbky pamäte mapy bude potom vážený priemer dĺžky najdlhších spoločných podpostupností písmen v množinách jednotlivých neurónov. Dĺžku najdlhšej podpostupnosti budem určovať od konca sekvencií v množine. Priemer pamäťových hĺbok jednotlivých neurónov musí byť vážený, aby neuróny, ktoré zvíťazili pre viac vstupov, mali vyššiu váhu ako víťazné neuróny pre menší počet vstupov. Po každej trénovacej epoche (prechode trénovacou množinou) budem vedieť určiť pamäťovú hĺbku mapy. Vďaka tomu, že neuróny rekurentných sietí majú okrem normálnych váh aj kontextové váhy, ktoré uchovávajú informáciu z predchádzajúceho kroku, môže sa stať, že rovnaké písmeno zo vstupu bude mať rôzne víťazné neuróny počas trénovania.

Samotná pamäťová hĺbka je relatívna a závisí od veľkosti sliding window.

Počet neurónov v mape volíme podľa množstva rôznych slov v použitej trénovacej množine.

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

Na trénovanie a vyhodnocovanie pamäťovej hĺbky som si vytvoril 3 sady trénovacích príkladov. Prvá sada sú najednoduchšie sekvenice (slová), ktoré obsahujú veľké množstvo regularít a malé množstvo rôznych písmen. Napríklad sekvencie ako *aaabbb*, *bbbaaaaa*. Druhou sadou budú slová generované Reberovým automatom. Treťou sadou trénovacích príkladov budú najzložitejšie sekvencie, resp. náhodný text s veľkým množstvom rôznych písmen. Túto sadu budem používať ako performance benchmark.

## 2.2 mSOM s leaky integration contextom

Pre potreby testovania pamäťovej hĺbky sme si vytvorili modifikovanú verziu mSOM. V pôvodnej mSOM bol kontext tvorený vlastnosťami víťazného neurónu z posledného kroku. V našej modifikovanej verzii mSOM budú kontext tvoriť samotné vstupné vektory z predchádzajúcich krokov. Všetko ostatné zostane rovnaké ako v pôvodnej mSOM.

Samotný kontext budem počítať pomocou nasledujúceho rekurzívneho vzťahu:

$$c = \beta^0 \cdot x_t + \beta^1 \cdot x_{t-1} + \beta^2 \cdot x_{t-2} \tag{2.1}$$

$\beta$ parameter je číslo $\beta < 1$

Tým zabezpečím, že kontext bude tvorený nejakou lineárnou kombináciou predchádzajúcich vstupov a čím ďalej idem do minulosti, tým majú tieto vstupy menšiu váhu.

# Kapitola 3

# Implementácia

# Kapitola 4

# Experiment

# Kapitola 5

# Vyhodnotenie a výsledky experimentu

# 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`.