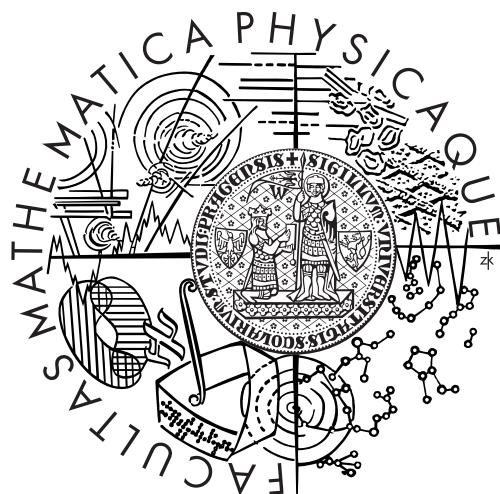


Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Bc. Radek Křížka

Dynamické Kohonenovy mapy a jejich struktura

Katedra teoretické informatiky a matematické logiky

Vedoucí diplomové práce: doc. RNDr. Iveta Mrázová, CSc.
Studijní program: informatika, teoretická informatika

2010

Poděkování

Na tomto místě bych chtěl poděkovat hlavně vedoucí práce paní doc. RNDr. Ivetě Mrázové, CSc. za podnětné připomínky, průběžnou kontrolu mých výsledků a neustálou podporu při psaní práce. Dále bych chtěl poděkovat Ing. Margitě Navrátilové za poskytnutí cenných dat. A samozřejmě také mým rodičům a přítelkyni, kteří mě po celou dobu podporovali.

Prohlašuji, že jsem svou diplomovou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze 22. března 2010

Radek Křížka

Obsah

Abstrakt	6
1 Úvod	7
2 Neuronové sítě	9
2.1 Neuron	9
2.2 Neuronová síť	12
2.2.1 Klasické modely neuronových sítí	14
3 Kohonenovy mapy	16
3.1 Základní pojmy a struktura Kohonenových map	16
3.2 Základní algoritmus učení bez učitele	20
3.2.1 Analýza konvergence algoritmu	21
4 Alternativní modely Kohonenových map	23
4.1 Algoritmy typu LVQ - učení s učitelem	23
4.1.1 Algoritmus LVQ 1	24
4.1.2 Algoritmus LVQ 2 (verze 2.1)	25
4.1.3 Algoritmus LVQ 3	25
4.2 Učení rostoucích Kohonenových map	27
4.2.1 Varianta učení bez učitele	27
4.2.2 Varianta učení s učitelem	31
4.3 Metoda rostoucích neuronových plynů	34
5 Struktura složitějších modelů Kohonenových map	36
5.1 Vícevrstvé Kohonenovy mapy	36
5.1.1 Struktura modelu	36
5.1.2 Metoda učení	37
5.2 Evoluční stromy	39
5.3 Metoda WEBSOM	42
6 Metody pro vizualizaci a shlukovou analýzu neuronů Kohonenových map	44
6.1 Metody vizualizace Kohonenových map	44
6.1.1 Sammonovo mapování	44

6.1.2 Zobrazení U-maticí	45
6.2 Metody shlukové analýzy neuronů Kohonenových map	47
7 Návrh implementace	50
7.1 Hlavní součásti systému	50
7.2 Architektura systému	53
8 Analýza a testování modelů	58
8.1 Požadavky kladené na modely	58
8.2 Rozbor jednotlivých modelů	60
8.2.1 Rozbor základního modelu Kohonenovy mapy s učením bez učitele	60
8.2.2 Rozbor rostoucích Kohonenových map	62
8.3 Experimenty a testování na "umělých" datech	63
8.3.1 Iris data	63
8.3.2 Test robustnosti modelů	64
8.3.3 Pima indians data	67
8.3.4 Zátěžový test na poškozených datech	69
8.4 Analýza a zhodnocení výsledků modelů	71
9 Nehodovost v silniční dopravě	74
9.1 Úvod do problematiky nehodovosti v silniční dopravě	74
9.2 Vstupní data a jejich význam	75
9.3 Předzpracování vstupních dat	78
10 Využití testovaných modelů při zpracování reálných dat o nehodovosti v dopravě	79
10.1 Realizace jednotlivých modelů	79
10.2 Aplikace na datech nehodovosti za rok (2008)	80
10.3 Aplikace na větším souboru dat	85
11 Zhodnocení a závěr	89
11.1 Závěr	89
11.2 Další směry a možnosti vývoje	90
Literatura	92
Seznam obrázků	97
Seznam tabulek	98
A Obsah CD	99

B Uživatelská příručka	100
B.1 Úvod a instalace	100
B.2 Struktura a spuštění aplikace	101
B.3 Ovládání aplikace	102

Název práce: Dynamické Kohonenovy mapy a jejich struktura

Autor: Radek Křížka

Katedra (ústav): Katedra teoretické informatiky a matematické logiky

Vedoucí diplomové práce: doc. RNDr. Iveta Mrázová, CSc.

e-mail vedoucího práce: Iveta.Mrazova@mff.cuni.cz

Abstrakt:

Tato diplomová práce se zabývá jedním z nejpoužívanějších modelů umělých neuronových sítí, a to Kohonenovými samoorganizujícími se mapami. Podrobně popíšeme jednotlivé modely Kohonenových map, provedeme jejich analýzu a vzájemné porovnání. Funkčnost, robustnost, míru zobecňování a další důležité vlastnosti modelů nejdříve ověřujeme na umělých vstupních datech. Možnosti jejich skutečného použití v praxi a vlastnosti vybraných typů Kohonenových map jsou testovány na reálných datech z oblasti nehodovosti v silniční dopravě. Zaměřujeme se přitom na detekci důležitých vstupních atributů dat. Zkoumáme možnosti řešení zajímavých otázek a aspektů silniční dopravy pomocí modelů Kohonenových map. V závěru práce je shrnuta celková analýza dosažených výsledků a návrhy možných variant modifikací, které by mohly zlepšit vlastnosti uvažovaných modelů.

Klíčová slova: neuron, Kohonenova mapa, struktura modelů, druhy učení, nehodovost.

Title: Dynamic Kohonen maps and their structure

Author: Radek Křížka

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor: doc. RNDr. Iveta Mrázová, CSc.

Supervisor's e-mail address: Iveta.Mrazova@mff.cuni.cz

Abstract:

My diploma thesis deals with one of the most widely used model of artificial neural network named self-organizing Kohonen neural network. We can find there a detailed description of several thoroughly analyzed mutually compared models of Kohonen map. We will verify their functionality, robustness and generalisation rates on artificial input data. Their real applicability and properties are tested on real data of traffic accident frequency. We will focus on the detection of significant input data attributes. The possibilities of solving the interesting questions and aspects of road transport are examined by means of Kohonen maps. At the end of the work there is presented a summarized review of the results and there are mentioned possible options of modifications that could improve the properties of these models.

Keywords: neuron, Kohonen map, model's structure, type of learning, traffic accident.

Kapitola 1

Úvod

Tato diplomová práce se bude zabývat jedním z nejpoužívanějších modelů umělých neuronových sítí, a to Kohonenovými samoorganizujícími se mapami. Neuronová síť představuje zjednodušený matematický model nervového systému živočichů. Základní myšlenkou je vycházet poznatků, jak funguje mozek a nervová soustava složitých organismů, které jsou schopny se učit. Tato relativně mladá oblast informatiky zažívá v současné době další období vzestupu. Teorie neuronových sítí se stále vyvíjí a její využití v praxi je neoddiskutovatelné.

Po seznámení se základními principy neuronových sítí (kapitola 2) a především Kohonenovými mapami (kapitola 3.1) se zaměříme na studium několika významných variant tohoto druhu neuronových sítí. Podrobně tedy popíšeme základní model Kohonenových map (viz kapitola 3.1 a kapitola 3.2), od kterého je odvozena téměř většina následujících variant modelů Kohonenových map. Dalším modelem bude reprezentant rostoucích Kohonenových map, a to Fritzkeho rostoucí model (podrobně v kapitole 4.2). Budeme se věnovat také složitějším strukturám, především vícevrstvé Kohonenové mapě a také zástupci nového druhu samoorganizujících se map - evolučním stromům.

Protože je naším úkolem také vizualizace dosahovaných výsledků, budeme se věnovat možnostem zobrazení modelů Kohonenových map. Budeme se zabývat především dvěma metodami zobrazování (kapitola 6.1), tzv. Sammonovým mapováním a zobrazením pomocí U-matice. Pokusíme se analyzovat jejich přednosti a nedostatky. Po vizualizaci modelů bude dalším úkolem nalezení shluků navzájem podobných neuronů v topologických mřížkách Kohonenových map. Budeme studovat různé metody shlukové analýzy, kde opět vybereme nejvhodnější metodu pro naše potřeby (viz kapitola 6.2).

Provedeme podrobnou analýzu jednotlivých modelů, zaměříme se na algoritmy jejich učení. Budeme se snažit modely navzájem porovnávat, zkoumat jak složitost algoritmů, tak možnosti adekvátního nastavení parametrů učení. Funkčnost, především dobrou approximaci pravděpodobnostního rozložení vstupních dat, budeme ověřovat

nejdříve na umělých vstupních datech v kapitole 8. Dále se zaměříme na vlastnosti modelů, jako jsou robustnost, míra zobecňování a další důležité vlastnosti (viz kapitola 8.3).

K analýzám a testům modelů Kohonenových map použijeme projekt, který bude součástí práce. Bude se jednat o projekt napsaný v jazyce Java, jehož strukturu a hlavní součásti popíšeme v kapitole 7. Tento projekt bude možné spustit z přiloženého CD.

Součástí práce bude také zhodnocení poznatků a zkušeností, které získáme při studiu vlastností modelů. Vzájemné porovnání modelů a návrhy jejich možného vylepšení spolu se shrnutím výsledků našich testů a postřehů uvedeme v kapitole 8

Možnosti skutečného použití těchto modelů v praxi budeme dále testovat na reálných datech z oblasti nehodovosti v silniční dopravě. Rozsáhlá data o dopravních nehodách ve městě Ostravě a okolí nejdříve předzpracujeme, normalizujeme a uložíme do vhodného formátu, který bude dále sloužit jako vstup jednotlivých modelů (kapitola 9). Zaměříme se na detekci důležitých vstupních atributů dat, budeme zkoumat možnosti řešení zajímavých otázek a aspektů silniční dopravy pomocí modelů Kohonenových map (kapitola 10). Tato problematika patří k vysoce aktuálním především s ohledem na rostoucí počet osobních i nákladních automobilů na silničních komunikacích.

Zkoumáním nehodovosti pomocí umělých neuronových sítí se zabývá také Akgüngör a Dogan v práci [1]. Jejich cílem je predikovat vývoj nehodovosti ve městě Ankara v Turecku. Jako vstupní parametry používají počet vozidel, smrtelné nehody, zranění při nehodách, celkový počet nehod a celkovou populaci. Jejich výsledkem jsou předpokládané počty nehod a při nich vzniklá zranění. V naší analýze se zaměříme spíše na identifikaci příčin vzniku nehod a jejich případnému předcházení. K tomu využijeme mnohem obsáhlější soubor dat, kdy vstupní data budou obsahovat až 30 atributů.

Pro přehlednost poznamenejme ještě několik věcí ke struktuře práce. Abychom mohli důkladně analyzovat vlastnosti vybraných modelů Kohonenových map, musíme v první teoretičtější části práce nastudovat podrobně funkčnost jednotlivých modelů Kohonenových map. Po té můžeme ve druhé praktičtější části zkoumat a testovat možnosti zobrazování, rychlosť učení, úspěšnost klasifikace a další zkoumané vlastnosti modelů Kohonenových map.

Kapitola 2

Neuronové sítě

Pro pochopení základních principů funkce matematických modelů neuronových sítí se budeme nejdříve krátce věnovat neurofyziologické motivaci. Prvním a hlavním impulzem pro zkoumání v této oblasti bylo modelovat chování lidského mozku. Dle [36] neurofyziologické znalosti umožnily vytváření jednoduchých matematických modelů, které již byly schopné řešit některé úlohy z oblasti umělé inteligence. Proto se v následující kapitole budeme stručně věnovat základnímu funkčnímu prvku nervové soustavy, a to nervové buňce, tzv. **neuronu**.

Dále budeme pokračovat popisem tzv. **formálního neuronu**, který představuje zjednodušenou matematickou formu biologického neuronu. Formální neuron je základní stavební jednotkou všech matematických modelů neuronových sítí, jimiž se budeme zabývat v dalších kapitolách. Pro přehlednost uvedeme, že v celém textu této práce budeme pojmem neuron označovat jeho zjednodušenou matematickou verzi.

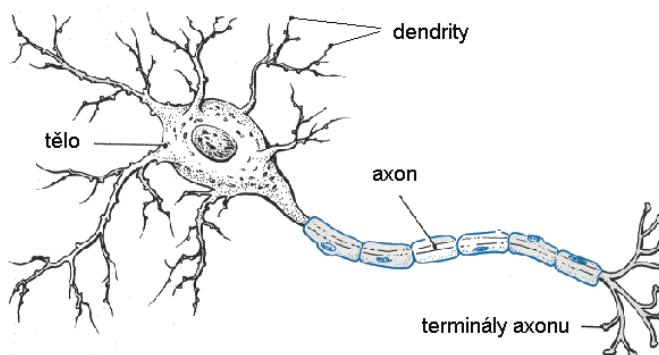
2.1 Neuron

Základním stavebním prvkem nervové soustavy je neuron. Dle [36] je mozková kůra člověka tvořena 13 až 15 miliardami neuronů. Neurony jsou vysoce specializované buňky schopné přenášet, zpracovávat a uchovávat informace z vnitřního i vnějšího prostředí a tím podmiňují schopnost celého organismu na tyto informace reagovat.

Ve stručnosti popíšeme model biologického neuronu (viz obrázek 2.1), na jehož základě je vytvořen model matematický. Největší částí biologického neuronu je tělo (soma). Neuron obsahuje dále dle [35], [29] a [36] vstupní a výstupní přenosové kanály, které nazýváme tzv. dendry a axon. Axon je dále větven do tzv. terminálů, které slouží pro spojení s dendry jiných neuronů. Tímto způsobem jsou neurony mezi sebou propojeny, jeden neuron může být spojen až s 5000 dalšími neurony.

Informace (elektrický signál) je mezi neurony přenášena přes speciální rozhraní mezi neurony, tzv. synapsi. Dle [35] a [36] představuje míra synaptické propustnosti mezi neurony hlavní základ přenosu informací v celém organismu. Synapse můžeme po-

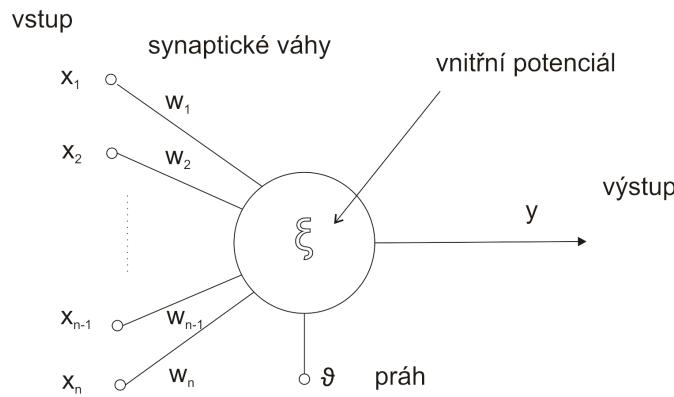
dle funkčnosti rozdělit na tzv. excitační, které zvýší hodnotu signálu (vzruchu) v nervové soustavě. Druhým typem jsou tzv. inhibiční synapse, které hodnotu signálu potlačují. Právě uložení hodnot synaptických vazeb znamená uložení informace v nervové soustavě.



Obrázek 2.1: Biologický neuron (zdroj server <http://www.mindcreators.com>).

Pro práci s modely neuronových sítí je třeba biologický neuron formalizovat a výše uvedené vztahy převést do matematického jazyka. Nyní se budeme zabývat strukturou formálního neuronu, kterou můžeme vidět na obrázku 2.2. Neuron může mít obecně n vstupů $\vec{x} = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n$, které představují informaci na vstupních dendritech. Hodnoty vstupních signálů jsou dále ovlivňovány synaptickými váhami $\vec{w} = (w_1, w_2, \dots, w_n) \in \mathbb{R}^n$. Každý neuron dále obsahuje hodnotu tzv. prahu neuronu ϑ . Vnitřní potenciál neuronu je poté vypočítán dle:

$$\xi = \sum_{i=1}^n x_i w_i + \vartheta. \quad (2.1)$$



Obrázek 2.2: Formální neuron.

Výstup neuronu y , který modeluje elektrický impulz axonu, je vypočten dle [29] a [36]:

$$y = g(\xi), \quad (2.2)$$

kde $g : \mathbb{R}^{n+1} \times \mathbb{R}^n \rightarrow \mathbb{R}$ je tzv. přenosová funkce.

Druhy možných přenosových funkcí můžeme vidět na obrázku 2.3. Nejdříve uvedeme nejjednodušší variantu přenosové funkce, a to je tzv. skoková přenosová funkce:

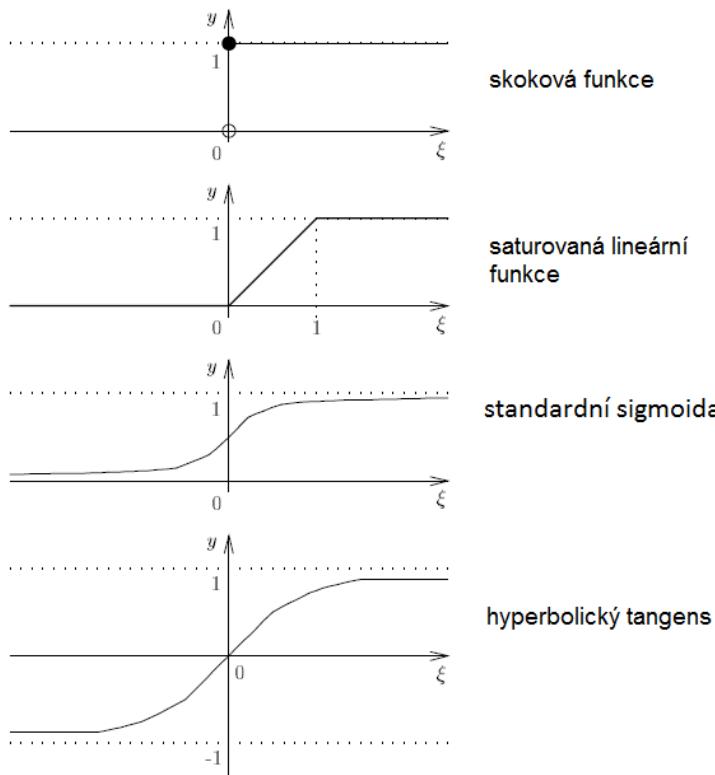
$$y = g[\vec{w}, \vartheta](\vec{x}) = g(\xi) = \begin{cases} 1 & \text{jestliže je vnitřní potenciál } \xi > 0, \\ 0 & \text{jinak.} \end{cases} \quad (2.3)$$

Ve většině případů ale volíme přenosovou funkci g jako nelineární tzv. sigmoidální přenosovou funkci (viz obrázek 2.3), kterou počítáme dle vzorce:

$$y = g[\vec{w}, \vartheta](\vec{x}) = g(\xi) = \frac{1}{1 + \exp(-\xi)}, \quad (2.4)$$

kde ξ představuje opět vnitřní potenciál neuronu. Podle hodnoty výstupu neuronu rozlišujeme dle [29] stavy neuronu na následující:

- Jestliže je výstupní hodnota rovna $y = 1$, říkáme, že neuron je *aktivní*.
- Jestliže je výstupní hodnota rovna $y = \frac{1}{2}$, říkáme, že neuron je *tichý*.
- Jestliže je výstupní hodnota rovna $y = 0$, říkáme, že neuron je *pasivní*.



Obrázek 2.3: Druhy přenosových funkcí (původní zdroj v [36]).

2.2 Neuronová síť

Neuronová síť (označována též jako umělá neuronová síť) je tvořena (formálními) neurony, které jsou vzájemně propojeny. Obecně je výstup neuronu dle [36] vstupem několika neuronů, stejně jako v biologickém případě jsou terminály spojeny přes synaptické spoje s dendrity ostatních neuronů. Vzájemné propojení neuronů a jejich počet v síti určuje tzv. topologii (architekturu) sítě. Podle základního využití můžeme neurony rozdělit na vstupní, skryté (síť tento typ neuronů nemusí obsahovat) a výstupní. Topologie umělé neuronové sítě je dle [36] a fyziologických vlastností analogická neurofysiologické sítě, kde vstupní neurony odpovídají receptorům a výstupní neurony efektorům.

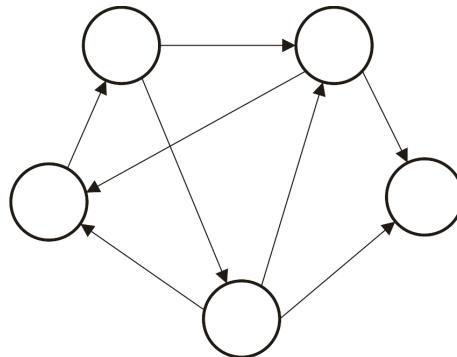
Nyní model neuronové sítě zformalizujeme a zapíšeme matematickým jazykem pro budoucí snazší manipulaci. Podle [29] a [2] definujeme neuronovou síť jako uspořádanou 6-tici (N, C, I, O, w, t) , kde

- N je konečná neprázdná množina neuronů.
- $C \subseteq N \times N$ je neprázdná množina orientovaných spojů mezi neurony.

- $I \subseteq N$ je neprázdná množina vstupních neuronů.
- $O \subseteq N$ je neprázdná množina výstupních neuronů.
- $w : C \rightarrow \mathbb{R}$ je váhová funkce.
- $t : N \rightarrow \mathbb{R}$ je prahová funkce.

Cílem učení neuronové sítě je najít optimální topologii sítě a vhodné zobrazení w a t tak, aby síť dávala námi požadované výsledky. Při procesu učení dochází ke změnám v neuronové síti, může se měnit topologie sítě, mění se stavy neuronů a adaptují se váhy. Tuto tzv. dynamiku sítě je dle [36] vhodné rozdělit do 3 kategorií, které uvedeme následovně. Volbou konkrétního typu příslušné dynamiky a jejich podrobným popisem definujeme tzv. **modely neuronových sítí**. Kategorie jsou následující:

- Organizační dynamika sítě se zabývá topologií sítě a jejími případnými změnami. Neuronové sítě můžeme dle topologie rozdělit na 2 základní typy, a to cyklické (neboli rekurentní) a acyklické sítě. Cyklická síť je definována tak, že obsahuje podmnožinu neuronů, které jsou zapojeny do kruhu (síť obsahuje tzv. cyklus). Příklad je uveden na obrázku 2.4. Další dělení neuronových sítí podle tohoto typu dynamiky je na sítě s pevnou topologií a na sítě s dynamickou topologií, u kterých v průběhu procesu učení dochází ke změnám počtu neuronů a spojů v síti. V této práci se budeme zabývat právě dynamickými modely neuronových sítí, kdy bude docházet jak k přidávání a příp. odebírání neuronů, tak spojů mezi nimi.



Obrázek 2.4: Příklad cyklické topologie.

- Aktivní dynamika sítě slouží k popisu změn stavů sítě. Tato dynamika specifikuje počáteční stav sítě a popisuje změny stavů v průběhu procesu učení za podmínky pevné topologie. Všechny možné stavy tvoří tzv. stavový prostor neuronové sítě. Ve většině případů uvažujeme spojitý stavový prostor. Čas učení ale předpokládáme diskrétní, na počátku se síť nachází v čase 0, stav sítě se aktualizuje pouze v čase 1, 2, ... atd.

Aktivní dynamika obsahuje také specifikaci pravidla pro výběr neuronu (nebo více neuronů), u kterého dojde k aktualizaci stavu na základě jeho vstupů (vstupy mohou být představované výstupy jiných neuronů). Podle toho, zda neurony mění svůj stav nezávisle na sobě nebo je proces aktualizace řízen centrálně, dělíme neuronové sítě dle [36] na tzv. asynchronní a synchronní. Součástí aktivní dynamiky je také popis přenosové funkce neuronů. Ta bývá u všech neuronů stejná - hovoříme pak o tzv. homogenní neuronové síti. Druhy přenosových funkcí jsme již uvedli v předchozí kapitole 2.1.

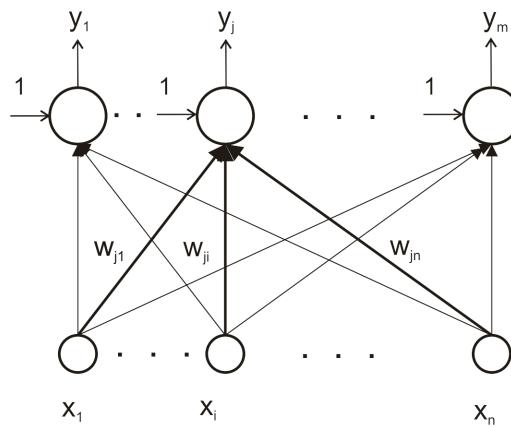
- Adaptivní dynamika sítě popisuje počáteční konfiguraci sítě a proces aktualizace vah sítě v průběhu času. Analogicky předchozímu případu tvoří všechny možné konfigurace vah tzv. váhový prostor neuronové sítě. Na začátku procesu učení se váhy nastaví na počáteční konfiguraci (ve většině případů náhodně), pak probíhá samotná adaptace. I zde je uvažován spojitý váhový prostor, čas uvažujeme jako diskrétní veličinu.

Adaptivní dynamika dále obsahuje popis metody učení pro konkrétní model neuronové sítě. Mezi nejznámější metody učení patří tzv. backpropagation algoritmus pro učení vícevrstvých neuronových sítí nebo základní metoda učení Kohonenových map (kterou se budeme dopodrobna zabývat níže).

2.2.1 Klasické modely neuronových sítí

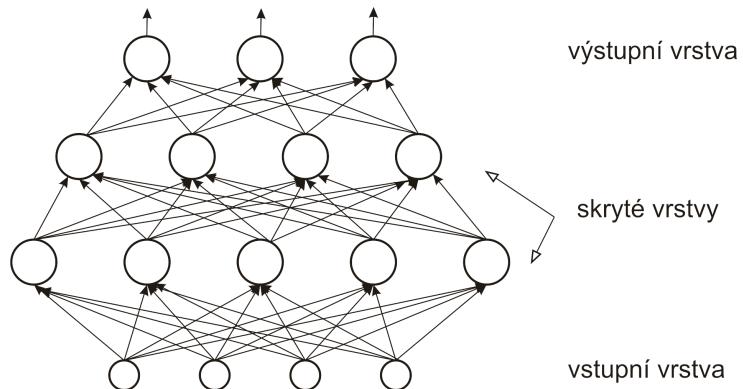
Seznámíme se s několika základními modely neuronových sítí, které jsou nebo byly aktivně využívány:

- **Kohonenvy samoorganizační mapy** [14] jsou modely využívající kompetiční strategii učení (detailněji bude vysvětleno v další kapitole). Společným rysem těchto modelů je, že neurony spolu navzájem soupeří o právo reprezentovat předložený vzor. Těmto modelům bude věnována celá práce, všechny podrobnosti a detaily budou popsány v následujících kapitolách.
- **Síť perceptronů** [30] je historicky prvním úspěšně používaným modelem umělých neuronových sítí. Organizační dynamiku můžeme specifikovat jako jednovrstvou síť $n - m$, jejíž příklad je na obrázku 2.5. Aktivní dynamika určuje způsob výpočtu sítě. Skutečné stavy neuronů ve vstupní vrstvě se nastaví podle vstupu sítě. Neurony ve výstupní vrstvě počítají svůj binární stav stejným způsobem jako formální neuron v kapitole 2.1.
Adaptivní dynamika specifikuje, jakým způsobem bude dosaženo, aby síť počítala požadovanou funkci. Učení je v tomto případě s učitelem, kdy ke vstupnímu vzoru máme k dispozici i požadovaný výstup. Váhy neuronů se pak upravují dle adaptačních rovnic podrobně popsaných v [36].



Obrázek 2.5: Topologie sítě perceptronů.

- K nejznámějším a také často používaným modelům neuronových sítí patří **vícevrstvé neuronové sítě** s učícím **algoritmem zpětného šíření chyby** popsaným v [31]. Model dokumentujeme obrázkem 2.6. Organizační dynamika je popisována jako pevná topologie vícevrstvé neuronové sítě. Topologie je acyklická, rozlišujeme vstupní, výstupní a skryté vrstvy neuronů. Tento model můžeme dle [36] nazvat zobecněním sítě perceptronů pro topologii se skrytými vrstvami. Podrobný popis aktivní a adaptivní dynamiky lze nalézt v [31], [29] a [36].



Obrázek 2.6: Topologie vícevrstvé neuronové sítě typu 4-5-4-3.

Kapitola 3

Kohonenovy mapy

Kohonenovy mapy jsou speciálním druhem umělých neuronových sítí. Základní model Kohonenových map přitom vychází z principu učení bez učitele, kdy nemáme k dispozici pro daný vstupní vzor požadovaný výstup. Princip těchto sítí je dle [16], [15] a [29] založen na tzv. **kompetiční strategii**, kdy výstupní neurony sítě spolu soupeří o to, který z nich bude aktivní a získá právo reprezentovat předložený vzor. Vítězný neuron potlačuje - inhibuje ostatní výstupní neurony. Model byl navržen Willshawem (1976) a Kohonenem (1982). Nejčastější použití je v rozpoznávání (např. segmentaci obrázků, písma), své uplatnění má také v ekonomii a při rozpoznávání mluvené řeči. Dle Šímy a Nerudy [36] existuje řada fyziologických prací, podle nichž se opravdu v lidském mozku vyskytují oblasti založené na kompetičním principu.

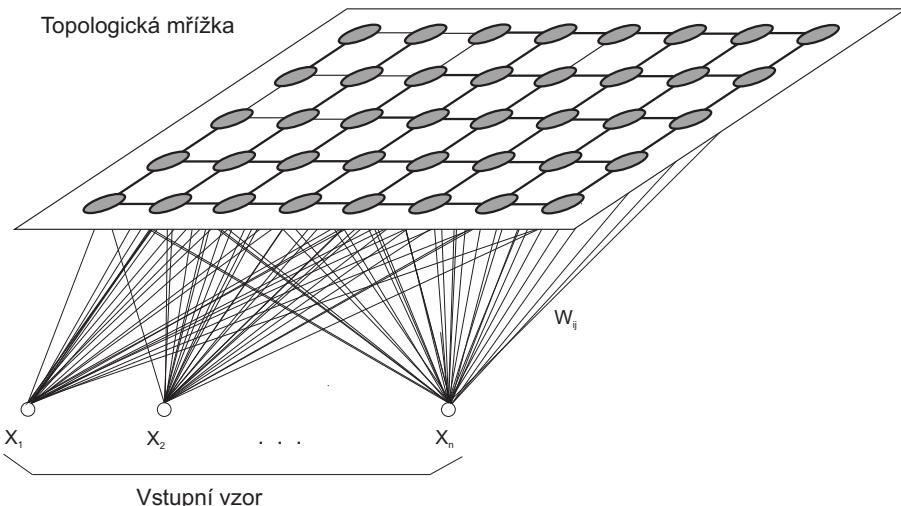
3.1 Základní pojmy a struktura Kohonenových map

Nyní se seznámíme se základními pojmy, které budeme dále používat při popisu modelů Kohonenových map. V základní verzi se jedná o dvouvrstvou síť, kde jsou všechny neurony mezi oběma vrstvami navzájem propojeny - propojení je tzv. úplné. První vstupní vrstva se skládá z n neuronů, které slouží k reprezentaci vstupního vektoru $\vec{x} \in \mathbb{R}^n$. Výstupní vrstva je uspořádána do určité topologické struktury, nejčastěji to bývá dvojrozměrná mřížka (též nazývána Kohonenova topologická mřížka) nebo jednorozměrný řetězec neuronů. Topologická struktura určuje, které neurony jsou mezi sebou propojeny v rámci dané vrstvy do tzv. **okolí** - neboli které neurony spolu navzájem sousedí. Topologická mřížka je vytvořena tak, aby byla možná snadná identifikace sousedů právě aktivního neuronu viz obrázek 3.1.

Okolí neuronu c o poloměru l , které značíme $N_l(c)$, určuje množinu neuronů j , které mají od neuronu c vzdálenost $m(c, j)$ menší nebo rovnu l :

$$N_l(c) = \{j \in \mathbb{N}; m(c, j) \leq l\}. \quad (3.1)$$

Použitá metrika $m(c, j)$ je závislá na zvolené topologii. Pokud například v jednorozměrné mřížce označíme konkrétní neuron l a vzdálenost nastavíme na hodnotu 1, do množiny sousedů $N_1(l)$ patří neurony s indexy $l - 1$ a $l + 1$. Na okrajích mřížky ale mohou mít neurony asymetrické okolí. Mřížka se může vyskytovat v mnoha provedeních a velikostech - čtvercová, hexagonální, atd.



Obrázek 3.1: Struktura Kohonenovy mapy.

Nyní ve stručnosti popíšeme základní princip funkčnosti Kohonenových map, a to proces **adaptace**. Po předložení vstupního vzoru nejprve určíme vítězný neuron s množinou neuronů z jeho okolí. Dalším krokem je proces aktualizace vah neuronů, kdy dochází k úpravě vah vítězného neuronu a neuronů z jeho okolí směrem k předloženému vstupnímu vzoru. Popis postupu při určování vítězného neuronu a procesu adaptace je vysvětlen v následující kapitole 3.2. Další možné druhy učení budeme podrobně analyzovat v kapitole 4.

Kohonenovy mapy mohou pracovat v různých režimech. Rozlišujeme 2 základní procesy:

- **Proces učení** - kdy plně využíváme topologickou strukturu sítě a vzájemné propojení sousedních neuronů v rámci definovaných okolí. Zde sice vítězí jeden neuron, ale následné úpravy vah se účastní také neurony z jeho okolí. Proto se adaptují také váhové vektory všech neuronů z okolí vítězného neuronu, které by neměly být daleko od sebe ani ve vstupním prostoru.

- **Proces rozpoznávání** - k výpočtu výstupu sítě se okolí neuronu prakticky vůbec nevyužívá. V případě standardního modelu Kohonenových map se v tomto režimu plně realizuje mechanismus laterální inhibice. Říkáme pak, že ”vítěz bere vše”¹. Po předložení vstupního vzoru je tedy aktivní pouze jeden neuron z celé mřížky, který má výstupní hodnotu rovnou jedné a potlačí aktivitu všech ostatních neuronů, které mají tudíž výstupní hodnotu rovnou nule.

Po určení vítězného neuronu spolu s množinou neuronů z jeho okolí následuje během učení aktualizace vah těchto neuronů. Velikost změny váhy není u všech neuronů stejná, ale je určena tzv. **funkcí laterální interakce**. Funkce laterální interakce $\phi(i, k)$ odpovídá ”síle laterální vazby” mezi neurony i a k v procesu učení. Obvykle ji volíme dle následujících příkladů:

- Diskrétní průběh funkce laterální interakce:

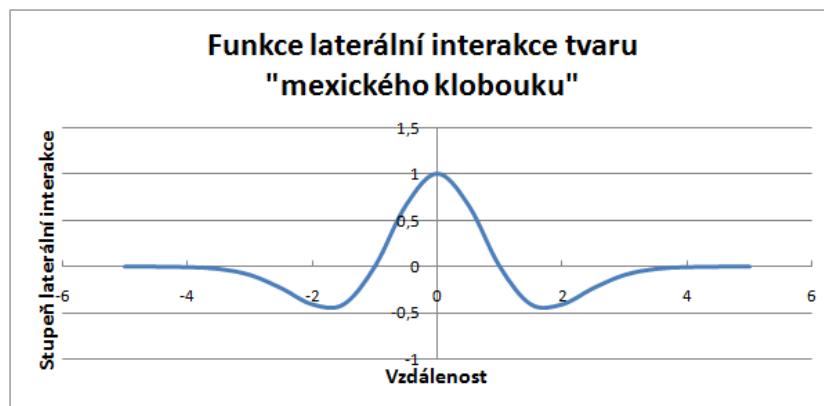
$$\begin{aligned}\phi(i, k) &= 1 \quad \forall i \in N_r(k), \\ \phi(i, k) &= 0 \quad \forall i \notin N_r(k),\end{aligned}\tag{3.2}$$

kde $\forall i \in N_r(k)$ značí všechny neurony i z okolí $N_r(k)$ neuronu k s poloměrem r .

- Dalším a často používaným typem funkce laterální interakce je tzv. funkce ”mexického klobouku”. Oproti předchozímu případu se jedná o funkci spojitu, což více odpovídá biologickým interakcím. Vypočteme ji dle:

$$\phi(i, k) = (1 - m^2(i, k)) \cdot e^{-\frac{m^2(i, k)}{p}},\tag{3.3}$$

kde $m(i, k)$ je námi zvolená metrika použitá při výpočtu okolí neuronu, p je parametr udávající ”šířku” této funkce. Průběh je možno vidět na obrázku 3.2.



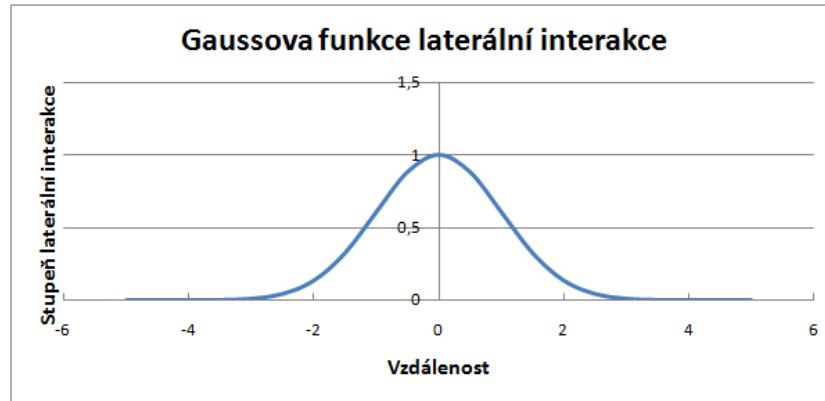
Obrázek 3.2: Funkce laterální interakce tvaru ”mexického klobouku”.

¹V anglické literatuře se popisuje pojmem ”winner takes all”.

- Třetím a velmi často používaným typem je Gaussova funkce laterální interakce, kterou vypočteme následovně:

$$\phi(i, k) = e^{-\frac{m^2(i, k)}{p}}, \quad (3.4)$$

kde metrika $m(i, k)$ a parametr p jsou stejné jako v předchozím případě. Průběh je znázorněn na obrázku 3.3.



Obrázek 3.3: Gaussova funkce laterální interakce.

3.2 Základní algoritmus učení bez učitele

Nyní popíšeme základní algoritmus učení Kohonenovy samoorganizující se mapy. Jedná se o algoritmus učení bez učitele², tzn. že pro jednotlivé trénovací vzory neznáme jejich požadované výstupy. Algoritmus, jímž se nyní budeme podrobněji zabývat, bude měnit váhy nejen vítězného neuronu, ale také váhy ostatních neuronů v jeho okolí. Velikost změny váhy není u všech neuronů stejná, ale je určena funkcí laterální interakce viz podkapitola 3.1. Označme počet vstupních neuronů jako n (n -rozměrný vstupní vektor), počet neuronů v topologické mřížce označíme m . Dále označíme počet trénovacích vzorů N a popíšeme samotný algoritmus uveden v [29].

Algoritmus 1: Základní učení Kohonenových map.

1. Inicializujeme váhy $w_{ij}(0)$ mezi n vstupními a m výstupními neurony jako malé náhodné hodnoty. Nastavíme poloměr okolí a inicializujeme funkci laterální interakce.
2. Předložíme nový trénovací vzor $\vec{x}(t) = (x_1(t), x_2(t), \dots, x_n(t))$ (trénovací vzory neobsahují informaci o požadovaném výstupu).
3. Pro každý výstupní neuron j spočítáme vzdálenost d_j mezi vstupním a výhovým vektorem dle vzorce:

$$d_j(t) = \sqrt{\sum_{i=0}^{N-1} (x_i(t) - w_{ij}(t))^2}, \quad (3.5)$$

kde $x_i(t)$ značí hodnotu vstupu i -tého neuronu v čase t , $w_{ij}(t)$ označuje hodnotu váhy synapse z i -tého vstupního do j -tého výstupního neuronu v čase t . (Euklidovskou vzdálenost je možno dále ladit pomocí různých koeficientů.)

4. Určíme výstupní neuron, který má nejmenší vzdálenost d_j od předloženého vstupního vzoru. Tento neuron označíme jako "vítěze" (nechť má vítězný neuron index c):

$$c = \operatorname{argmin}_{i=1,\dots,m} (d_i(t)). \quad (3.6)$$

5. Nyní budeme aktualizovat váhy vítězného neuronu c a všech neuronů z jeho okolí, které jsou definovány pomocí množiny N_c . Nové váhy jsou aktualizovány pomocí vzorce:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha(t) \phi(c, j) (x_i(t) - w_{ij}(t)), \quad (3.7)$$

kde $j \in N_c$; $0 \leq i \leq N - 1$; $\alpha(t)$ je tzv. vigilanční koeficient³, který může nabývat hodnot z intervalu $(0 \leq \alpha(t) \leq 1)$ a jehož hodnota klesá v čase.

²V anglické literatuře se označuje pojmem "unsupervised learning".

³Vigilance je synonymum pro bdělost.

Učení je tedy založeno na principu, že vítězný neuron upraví svůj váhový vektor směrem k aktuálnímu vstupnímu vektoru. Podobně jsou upravovány váhy z okolí ”vítěze”. Důležitou roli ve vzorci pro úpravu váhových vektorů hraje hodnota funkce laterální interakce $\phi(c, j)$, která klesá s rostoucí vzdáleností neuronů od středu okolí N_c .

6. Přejdeme ke kroku číslo 2.

Předložíme další trénovací vzor ($t \leftarrow t + 1$).

3.2.1 Analýza konvergence algoritmu

Nyní budeme zkoumat stabilitu řešení tohoto algoritmu za podmínky, že síť už dospěla do určitého uspořádaného stavu. Protože nejčastější topologie těchto sítí bývají jednorozměrné a dvojrozměrné, budeme analyzovat právě tyto případy. Jak je popsáno v [29], konvergence probíhá následovně:

- Jednorozměrný případ dle [29]:
 1. Nejdříve prozkoumáme konkrétní případ, kdy máme interval $[a, b]$ a 1 neuron s váhou x , prozatím nebudeme uvažovat jeho okolí.
Nyní tedy můžeme ukázat konvergenci váhy x do středu intervalu $[a, b]$:
 - Použijeme adaptační pravidlo ve tvaru: $x_n = x_{n-1} + \mu(\xi - x_{n-1})$
kde x_n, x_{n-1} jsou váhy v čase n a $n - 1$,
 ξ je náhodně zvolené číslo z intervalu $[a, b]$.
 - Pak tedy pro $0 < \mu \leq 1$ nám nemůže posloupnost x_1, x_2, \dots, x_n opustit interval $[a, b]$.
 - Omezená je i dokonce očekávaná hodnota $\langle x \rangle$ váhy x .
 - Očekávaná hodnota derivace x v čase je nulová: $\langle \frac{dx}{dt} \rangle = 0$
(jinak by mohlo být $\langle x \rangle < a$ anebo $\langle x \rangle > b$).
 - Protože tedy platí: $\langle \frac{dx}{dt} \rangle = \mu(\langle \xi \rangle - \langle x \rangle) = \mu(\frac{a+b}{2} - \langle x \rangle)$,
dostáváme $\langle x \rangle = \frac{a+b}{2}$.
 2. V druhém případě máme opět interval $[a, b]$, ale n neuronů s váhami x^1, x^2, \dots, x^n , neuvažováno okolí:
 - Předpokládáme, že váhy jsou monotónně uspořádány: $a < x^1 < x^2 < \dots < x^n < b$.
 - Stejným principem lze ukázat konvergenci jednotlivých vah: $\langle x^i \rangle = a + (2i - 1)\frac{b-a}{2n}$.
- Dvourozměrný případ:

Konvergenci zjištujeme na intervalu $[a, b] \times [c, d]$, $n \times n$ neuronů, není uvažováno okolí, máme monotónní uspořádání vah $w_1^{ij} < w_1^{ik}$ pro $j < k$ a $w_2^{ij} < w_2^{kj}$ pro

$i < k$.

Provedeme redukci problému na dva jednorozměrné:

- Nechť $w_1^j = \frac{1}{n} \sum_{i=1}^n w_1^{ij}$ značí průměr vah neuronů v j – tém sloupcem mřížky.
- Díky vlastnosti $w_1^{ij} < w_1^{ik}$ pro $j < k$ dostáváme monotonné uspořádání $w_1^j = a < w_1^1 < w_1^2 < \dots < w_1^n < b$.
- Z předcházejícího bodu plyne, že v prvním sloupci bude průměr vah oscilovat kolem očekávané hodnoty $\langle w_1^1 \rangle$.
- Podobným postupem získáme výsledky pro neurony v každém řádku.
- Tím získáváme konvergenci ke stabilnímu stavu (při volbě dostatečně malého parametru učení).

Kapitola 4

Alternativní modely Kohonenových map

V této kapitole se budeme zabývat dalšími možnostmi, jak lze učit Kohonenovy mapy. Budeme se odkazovat na základní algoritmus učení, který je podrobně rozebrán v kapitole 3.2. Tento algoritmus se využívá jako základ pro složitější typy učení, jež budou popsány v dalších podkapitolách. Budou to modely, které dokážou navíc využít při učení i informaci o požadovaném výstupu, který je k dispozici spolu se vstupními daty. Dále to budou metody schopné dynamicky přizpůsobovat svou topologii řešené úloze, a tak lépe řešit danou úlohu (rychlejší řešení, lepsí approximace vstupních dat).

4.1 Algoritmy typu LVQ - učení s učitelem

Nyní se již dostáváme k učení Kohonenových map s učitelem. V této podkapitole popíšeme algoritmy typu **LVQ** (Learning Vector Quantization). Základem tohoto principu učení je trénovací množina ve tvaru $\{(\vec{x}^{(p)}, y^{(p)}); p = 1, \dots, N\}$, kde $\vec{x}^{(p)} \in \mathbb{R}^n$ a $y^{(p)} \in \{C_1, \dots, C_M\}$. Pro každý vstupní vektor $\vec{x}^{(p)}$ máme jednoznačně určen požadovaný výstup $y^{(p)}$, jenž odpovídá jedné z konečného počtu kategorií C_i pro $1 \leq i \leq M$. Tato metoda učení je vhodná pro úlohy klasifikace dat do několika požadovaných kategorií.

Algoritmus LVQ se vyskytuje v několika variantách. Základní kostra tohoto algoritmu je dle [14] a [36] ale společná:

Algoritmus 2: Základní princip algoritmu LVQ.

1. Síť nejprve učíme základním algoritmem učení bez učitele viz podkapitola 3.2. Tímto algoritmem rozmístíme neurony po vstupním prostoru, a tím approximujieme hustotu pravděpodobnostního rozložení vstupních vzorů. V tomto kroku jsme ještě nevyužili požadovaný výstup $y^{(p)}$ z trénovací množiny.

2. Každý výstupní neuron označíme kategorií.

V této fázi učení nejprve pro každý trénovací vzor určíme neuron, který je ke vzoru nejbližše. U tohoto neuronu si uložíme informaci o požadované třídě, kterou získáme z nejbližšího trénovacího vzoru. Průchodem celé trénovací množiny takto získáme u neuronů seznam požadovaných kategorií. Každý neuron ale musí představovat pouze jednu kategorii, proto bude neuron i reprezentovat **nejčetnější** kategorii ze seznamu, kterou budeme značit v_i .

3. Doučení sítě konkrétní variantou algoritmu LVQ.

Poslední fáze slouží k doučení a doladění Kohonenovy mapy. Postupně bylo navrženo několik variant, jak síť učit. My popíšeme jeho 3 postupně se zlepšující varianty.

4.1.1 Algoritmus LVQ 1

Nejdříve se budeme zabývat nejstarším algoritmem LVQ 1. Je založen na myšlence, že pokud neuron klasifikuje správně, posuneme jeho váhy blíže ke vzoru $\vec{x}^{(p)}$. Naopak pokud klasifikuje špatně, změníme jeho váhy tak, aby se vzdálil od vstupního vzoru. Velikost tohoto posunu můžeme ladit velikostí parametru učení $\alpha(t)$, jehož hodnota je typicky závislá na čase. Dle [36] a praktických zkušeností se doporučuje nastavit počáteční hodnotu parametru α na 0.01 - 0.02 a s přibývajícím časem (řádově v průběhu několika tisíc iterací) ho snižovat až k nule. Algoritmus postupně prochází celou trénovací množinu a upravuje váhy dle následujícího postupu. V jednom kroku algoritmu přitom upravujeme vždy váhu pouze jednoho - vítězného neuronu.

Nejdříve pro vzor $(\vec{x}^{(p)}, y^{(p)})$ určíme nejbližší neuron, jehož index označíme c :

$$c = \operatorname{argmin}_k \{ \| \vec{x}^{(p)} - \vec{w}_k \| \}. \quad (4.1)$$

Následná úprava vah se tedy týká pouze vítězného neuronu c . Jeho váhy aktualizujeme dle:

- Pokud vstupní vzor $\vec{x}^{(p)}$ a neuron c s váhovým vektorem \vec{w}_c patří ke stejné třídě (neboli vzor $\vec{x}^{(p)}$ je správně klasifikován, tzn. $v_c = y^{(p)}$):

$$\vec{w}_c(t+1) = \vec{w}_c(t) + \alpha(t)[\vec{x}^{(p)} - \vec{w}_c(t)]. \quad (4.2)$$

- Pokud vstupní vzor $\vec{x}^{(p)}$ a neuron c s váhovým vektorem \vec{w}_c nepatří ke stejné třídě (neboli vzor $\vec{x}^{(p)}$ není správně klasifikován, tzn. $v_c \neq y^{(p)}$):

$$\vec{w}_c(t+1) = \vec{w}_c(t) - \alpha(t)[\vec{x}^{(p)} - \vec{w}_c(t)]. \quad (4.3)$$

- Váhy u ostatních neuronů zůstávají stejné (pro neurony s indexem $i \neq c$):

$$\vec{w}_i(t+1) = \vec{w}_i(t). \quad (4.4)$$

4.1.2 Algoritmus LVQ 2 (verze 2.1)

Při popisu algoritmu LVQ 2 vycházíme z předchozí verze algoritmu LVQ 1. Jedná se o mírnou modifikaci původního algoritmu. Algoritmus LVQ 1 vytvoří mezi třídami takovou hranici, která je dobrou approximací **bayesovské rozhodovací hranice**. Problémem u LVQ 1 je, že algoritmus posouvá neurony jako reprezentanty třídy spíše směrem od optimální rozhodovací hranice. Hranice, kterou vytvoří LVQ 1 je přímo uprostřed spojnice mezi 2 neurony patřícími do různých tříd.

Tento problém je právě řešen v algoritmu LVQ 2, kdy se snažíme vytvořit rozhodovací hranici, která by approximovala bayesovskou hranici. Dalsím rozdílem oproti první verzi algoritmu je, že algoritmus LVQ 2 posunuje v jednom kroku vždy 2 neurony a ne 1 neuron, jako v předchozí verzi. Po předložení vstupního vzoru $\vec{x}^{(p)}$ najdeme k tomuto vzoru 2 nejbližší neurony s indexy i a j , jejichž váhy označíme \vec{w}_i a \vec{w}_j .

Dalším rozdílem oproti předchozí verzi je splnění podmínky, kterou klademe při výběru 2 nejbližších neuronů. Požadujeme, aby se vstupní vzor $\vec{x}^{(p)}$ nacházel v okolí dělicí nadplochy mezi \vec{w}_i a \vec{w}_j . Toto okolí budeme označovat jako tzv. "okénko". Dále definujeme vzdálenost mezi $\vec{x}^{(p)}$ a \vec{w}_i jako $d_i = d(\vec{x}^{(p)}, \vec{w}_i)$. Nejpoužívanější metrikou pro výpočet vzdáleností je Euklidovská metrika. Pak splnění podmínky, že $\vec{x}^{(p)}$ patří do okénka je dáno rovnicí:

$$\min \left\{ \frac{d_i}{d_j}, \frac{d_j}{d_i} \right\} > s, \quad (4.5)$$

kde $s = \frac{1-q}{1+q}$. Dle [14] a [36] je optimální volbou parametru q hodnota v rozmezí $0.2 - 0.3$. Parametr q ovlivňuje šířku okénka, které by mělo být co nejužší, což umožní přesnější umístění hranice, ale také dostatečně široké pro zachycení statisticky významného množství dat.

Nyní přejdeme k nejdůležitější části algoritmu, a to aktualizaci vah¹:

$$\vec{w}_i(t+1) = \vec{w}_i(t) - \alpha(t)[\vec{x}^{(p)}(t) - \vec{w}_i(t)], \quad (4.6)$$

$$\vec{w}_j(t+1) = \vec{w}_j(t) + \alpha(t)[\vec{x}^{(p)}(t) - \vec{w}_j(t)], \quad (4.7)$$

kde \vec{w}_i a \vec{w}_j leží nejblíže k $\vec{x}^{(p)}$ a současně platí, že $\vec{x}^{(p)}$ a \vec{w}_j patří ke stejné třídě a $\vec{x}^{(p)}$ a \vec{w}_i patří k různým třídám a navíc se $\vec{x}^{(p)}$ nachází v okénku.

4.1.3 Algoritmus LVQ 3

Dle [36] adaptační pravidla LVQ 2.1 opravdu vedla ke zlepšení rozhodovací hranice, a to approximaci bayesovské hranice. Pokud ale algoritmus pracoval po delší počet

¹Předtím ještě poznamenejme, že tento algoritmus je mírnou modifikací původního algoritmu LVQ 2, v jiných literaturách ho můžeme najít pod označením LVQ 2.1. Původní algoritmus aktualizoval váhy pouze v tom případě, že neuron i byl označen špatnou kategorií a současně byl nejblíže ke vstupnímu vzoru $\vec{x}^{(p)}$. Následná adaptační pravidla se tedy řídí postupem algoritmu označovaného jako LVQ 2.1.

adaptačních kroků, docházelo k oddalování neuronů s váhovým vektorem \vec{w}_i od této hranice (\vec{w}_i je váhový vektor neuronu z předchozí podkapitoly o algoritmu LVQ 2.1 (4.6)). Doporučený počet iterací, po které je dobré algoritmus LVQ 2.1 používat je většinou kolem 10000. Proto byla vyvinuta třetí varianta toho algoritmu, která má za úkol aproximovat rozložení tříd a stabilizovat řešení.

Algoritmus LVQ 3 obsahuje navíc další pravidlo, kterým zajišťuje, že správně klasifikující neurony budou posunovány směrem ke vstupnímu vzoru. Adaptační pravidla algoritmu LVQ 3 jsou pak následující:

$$\begin{aligned}\vec{w}_i(t+1) &= \vec{w}_i(t) - \alpha(t)[\vec{x}^{(p)}(t) - \vec{w}_i(t)], \\ \vec{w}_j(t+1) &= \vec{w}_j(t) + \alpha(t)[\vec{x}^{(p)}(t) - \vec{w}_j(t)],\end{aligned}\quad (4.8)$$

kde neurony s indexem i a j jsou nejblíže ke vstupnímu vzoru $\vec{x}^{(p)}$, dále $\vec{x}^{(p)}$ patří ke stejné třídě jako neuron s váhovým vektorem \vec{w}_j ($v_j = d^{(p)}$), $\vec{x}^{(p)}$ a neuron s váhovým vektorem \vec{w}_i nepatří ke stejným třídám ($v_i \neq d^{(p)}$) a vstupní vzor $\vec{x}^{(p)}$ opět náleží do okénka tvořeného neurony i a j . Další pravidlo:

$$\vec{w}_k(t+1) = \vec{w}_k(t) + \epsilon\alpha(t)[\vec{x}^{(p)}(t) - \vec{w}_k(t)] \quad (4.9)$$

je aplikováno pro $k \in \{i, j\}$ a kde navíc platí, že vstupní vektor $\vec{x}^{(p)}$ a neurony s váhovými vektory \vec{w}_i , \vec{w}_j patří do stejné třídy ($v_i = v_j = d^{(p)}$). Hodnotu parametru ϵ volíme dle [14] a [36] v rozmezí $0.1 - 0.5$ a tato hodnota² je po celou dobu učení konstantní.

²Hodnota parametru ϵ byla zjištěna experimentálními pokusy.

4.2 Učení rostoucích Kohonenových map

Novým modelem Kohonenových map, který eliminuje některé z nedostatků základního modelu jsou tzv. **rostoucí Kohonenovy mapy** (Growing cell structures), jež jsou popsány v [7]. Je to především rychlost učení, která je u základního modelu malá díky velké topologické mřížce již na počátku učení. Rozlišujeme 2 základní varianty dle typu učení. První varianta funguje na principu učení bez učitele a používá se především v oblastech vizualizace a klastrování dat. Hlavní výhodou tohoto modelu je schopnost automaticky najít vhodnou strukturu a velikost sítě. To umožňuje řízený proces přidávání a prořezávání neuronů.

Druhou variantou je model učení s učitelem, který kombinuje první variantu spolu s RBF (radial basis function) modelem. Hlavním rysem a výhodou tohoto modelu je schopnost paralelní aktualizace umístění RBF jednotek ve vstupním prostoru společně se současným učením vah RBF jednotek. Díky této vlastnosti může být aktuální klasifikační chyba sítě použita pro určení místa, kam bude přidána nová RBF jednotka. Celkově model vede k vytváření malých sítí, které ale velmi dobře generalizují³.

4.2.1 Varianta učení bez učitele

Předtím než přejdeme k samotnému modelu se budeme nejprve věnovat značení. Budeme se snažit zachovat značení z předchozích modelů. Naším cílem je opět najít vhodné mapování ze vstupního prostoru \mathbb{R}^n do diskrétního k -dimenzionálního prostoru. Mapování by mělo mít několik následujících vlastností:

- Podobné vstupy (stejného nebo příbuzného charakteru) by měly být mapovány na stejný anebo blízký neuron v topologické mřížce.
- Na konkrétní neuron v topologické mřížce by měly být mapovány podobné vstupy.
- Oblast vstupního prostoru, kde je velká hustota pravděpodobnosti výskytu vstupů, by měla být reprezentována odpovídajícím (tzn. větším) počtem neuronů.

Jako výchozí strukturu topologické mřížky zvolíme k -dimenzionální simplexní útvary, který budeme dále značit A . Tzn. že pro $k = 1$ by byla výchozí topologická mřížka ve tvaru úsečky, pro $k = 2$ trojúhelník, pro $k = 3$ nebo vyšší je struktura topologické mřížky popsána čtyřsténem a dále vícedimenzionálním čtyřsténem. Každá k -dimenzionální mřížka tedy obsahuje $(k+1)$ neuronů. Neurony jsou mezi sebou spojeny $\frac{k(k+1)}{2}$ hranami, které určují vzájemné vztahy mezi neurony. Propojení je úplné, tudíž každý neuron je spojen se všemi ostatními neurony.

V průběhu učení budeme přidávat nové neurony nebo naopak nadbytečné odstraňovat.

³Generalizace je proces zobecnění znalostí extrahovaných z předkládaných dat během učení.

Většinou se jako výchozí struktura topologické mřížky volí vícedimenziorní čtyřstěn. Jeho výhody jsou minimální složitost a také vlastnost, že může být jednoduše rozšířen do větších struktur. Významná je také skutečnost, že počet neuronů k -dimenzionálního čtyřstěnu roste pouze lineárně s k , ale například u k -dimenzionální hyperkrychle je růst exponenciální (2^k). Proto volba vícedimenziorního čtyřstěnu je optimální pro mnoha dimenzionální sítě.

Váhový vektor \vec{w}_c neuronu c v topologické mřížce můžeme chápout jako pozici neuronu c ve vstupním prostoru. Dále budeme značit W množinu všech váhových vektorů \vec{w}_i , kde index i prochází přes všechny neurony v topologické mřížce. Pak definujeme mapování ϕ_w ze vstupního prostoru \mathbb{R}^n do topologické mřížky jako:

$$\phi_w : \mathbb{R}^n \rightarrow A, \quad (4.10)$$

kde pro vstupní vzor $\vec{x} \in \mathbb{R}^n$ budeme symbolem $\phi_w(\vec{x})$ značit vítězný neuron, který je definován následovně:

$$\| \vec{w}_{\phi_w(\vec{x})} - \vec{x} \| = \min_i \| \vec{w}_i - \vec{x} \| . \quad (4.11)$$

Pro výpočet vzdáleností $\| \cdot \|$ používáme opět Euklidovskou metriku.

Nyní přecházíme k samotnému procesu učení. Základem je opět **adaptační krok**, který je stejný jako v základním modelu učení Kohonenových map:

1. Výpočet vítězného neuronu pro předložený vstupní vzor.
2. Adaptace váhy vítězného neuronu a jeho topologických sousedů směrem ke vstupnímu vzoru.

V základním modelu učení bez učitele je míra adaptace postupně snižována vigilančním koeficientem (viz kapitola 3.2). V modelu rostoucích Kohonenových map jsou ale 2 důležité rozdíly:

- Míra adaptace je konstantní v čase. Speciálně používáme adaptační parametry ϵ_b a ϵ_n jako konstanty v procesu učení pro vítězný neuron, respektive pro neurony z okolí vítězného neuronu.
- Procesu adaptace se účastní pouze vítězný neuron a jeho **přímí** sousedé (tzn. neurony spojené hranou s vítězným neuronem).

Díky rozdílům není potřeba v tomto modelu definovat vigilanční koeficient. Dále budeme značit jako N_c množinu všech přímých sousedů neuronu c a zavedeme pro každý neuron c lokální čítač τ_c , který značí počet vstupních vzorů, pro které se stal neuron c vítězem v kompetici během učení. Pro zajištění větší míry adaptace, budeme navíc u ostatních neuronů čítač snižovat o určitou poměrnou část. Pro správnou funkci musíme tudíž zajistit, aby byl čítač reprezentován *reálnou* hodnotou.

Pak celý adaptační krok modelu rostoucích Kohonenových map můžeme popsat následovně:

1. Předložíme nový vstupní vzor $\vec{x} \in \mathbb{R}^n$.
2. Dle předchozího pravidla 4.11 určíme vítězný neuron $c = \phi_w(\vec{x})$.
3. provedeme aktualizaci vah vítězného neuronu a jeho přímých sousedů dle pravidel:

$$\vec{w}_c = \vec{w}_c + \epsilon_b(\vec{x} - \vec{w}_c), \quad (4.12)$$

pro vítězný neuron c ,

$$\vec{w}_i = \vec{w}_i + \epsilon_n(\vec{x} - \vec{w}_i), \quad (4.13)$$

pro neurony z množiny $i \in N_c$.

4. Zvýšíme čítač vítězného neuronu:

$$\tau_c = \tau_c + 1. \quad (4.14)$$

5. Snížíme čítač všech ostatních neuronů $j \in A$ o část α :

$$\tau_j = \tau_j - \alpha \tau_j. \quad (4.15)$$

Velikost parametrů ϵ_b a ϵ_n volíme jako malé hodnoty, neurony se pak pohybují z jejich počátečních pozic do oblastí s dynamickou rovnováhou ve všech směrech.

Dále zavedeme pojem relativní frekvence vstupu h_i pro každý neuron i jako:

$$h_i = \frac{\tau_i}{\sum_{j \in A} \tau_j}. \quad (4.16)$$

Ke konci učení by měly mít všechny neurony podobné hodnoty relativní frekvence vstupu. Právě vyšší hodnota h_i signalizuje místo, kam by měl být vložen nový neuron, který by redukoval vyšší hodnotu h_i . V následujících odstavcích popíšeme vkládání nových neuronů, jež je založeno právě na hodnotě relativní frekvence vstupu.

Nejprve určíme konstantu λ , která určuje počet adaptačních kroků, po nichž budeme přidávat nový neuron. Po λ krocích tedy vypočteme index neuronu q s následující vlastností:

$$h_q \geq h_i \quad (\forall i \in A). \quad (4.17)$$

V dalším kroku procházíme množinu neuronů N_q , tzn. množinu všech přímých sousedů neuronu q , a hledáme neuron $f \in N_q$ s největší vzdáleností od neuronu q ve vstupním prostoru:

$$\| \vec{w}_f - \vec{w}_q \| \geq \| \vec{w}_i - \vec{w}_q \| \quad (\forall i \in N_q). \quad (4.18)$$

Tím určíme 2 neurony q a f , mezi které bude vložen nový neuron r . Nový neuron je spojen hranou s neurony q , f a poté se všemi (přímými) sousedy obou neuronů q a f . Původní hrana mezi neurony q a f je odebrána. Váhový vektor neuronu r je inicializován hodnotou

$$\vec{w}_r = 0.5(\vec{w}_q - \vec{w}_f). \quad (4.19)$$

Nakonec provedeme aktualizaci čítačů vítězného neuronu r a neuronů z jeho okolí ($i \in N_r$) dle

$$\tau_r = - \sum_{i \in N_r} \Delta\tau_i, \quad (4.20)$$

kde změny čítačů $\Delta\tau_i$ vypočteme jako v předchozím případě:

$$\Delta\tau_i = -\alpha\tau_i. \quad (4.21)$$

Jedna z možností, jak určit parametr α , je výpočet pomocí objemu Voronoiových oblastí. Topologická struktura A je rozdělena neurony do několika oblastí $F_i (i \in A)$, kde každá z těchto oblastí má příslušný váhový vektor \vec{w}_i . Tento proces je též nazýván Voronoiovou teselací a jednotlivé oblasti F_i nazýváme Voronoiovy oblasti. Parametr α pak vypočteme dle:

$$\alpha = \frac{|F_i^{(puv.)}| - |F_i^{(nov.)}|}{|F_i^{(puv.)}|}, \quad (4.22)$$

a tím získáváme vztah pro $\Delta\tau_i$ pro $i \in N_r$:

$$\Delta\tau_i = \frac{|F_i^{(nov.)}| - |F_i^{(puv.)}|}{|F_i^{(puv.)}|} \tau_i, \quad (4.23)$$

kde $|F_i|$ je n -dimenzionální objem oblasti F_i . Princip vkládání neuronů do struktury je naznačen algoritmem číslo 3.

Algoritmus 3: Princip vkládání nových neuronů do struktury rostoucích Kohonenových map.

1. Algoritmus začínáme s k -dimenzionálním útvarem v prostoru \mathbb{R}^n .
2. Opakujeme následující kroky, dokud nenajdeme optimální strukturu a velikost síťě:
 - Zvolíme konstantní počet adaptačních kroků λ .
 - Vložíme nový neuron do struktury a aktualizuj hodnoty čítačů dle rovnic 4.20, 4.21 a 4.23.

Nyní se dostaváme k oddílu **odebírání neuronů**. Tato operace je přínosná hlavně v případech, kdy se pravděpodobnostní rozdělení vstupních vzorů sestává

z několika oddělených oblastí. Tehdy je přínosné modelovat odebírání ”nadbytečných” neuronů. Neuron prohlásíme za nadbytečný, jestliže se jeho váhový vektor nachází v oblasti s nízkou hustotou pravděpodobnosti vstupních vzorů. Všeobecně pravděpodobnostní rozložení vstupních vzorů neznáme, ale můžeme ho lokálně odhadovat pomocí podílu **relativní frekvence vstupu** vztaženou na Voronoiho oblast, kterou ”reprezentuje”:

$$p_c = \frac{h_c}{|F_c|}. \quad (4.24)$$

Tím získáme lokální odhad hustoty pravděpodobnosti výskytu vstupních vzorů v oblasti určenou vektorem \vec{w}_c . Periodické odebírání neuronů s hodnotou p_c menší než práh η vede k velmi přesnému modelování rozložení pravděpodobnosti.

4.2.2 Varianta učení s učitelem

Primárně jsou Kohonenovy mapy určeny pro učení bez učitele. Často mají zobrazovat vstupní vzor z n -dimenzionálního prostoru do méně dimenzionální topologické struktury. V některých typech úloh ale dostáváme trénovací data s příslušnými kategoriemi ve tvaru $\{(\vec{x}^{(p)}, \vec{y}^{(p)}); p = 1, \dots, N\}$, kde $\vec{x}^{(p)} \in \mathbb{R}^n$ a $\vec{y}^{(p)} \in \mathbb{R}^m$. Pro každý vstupní vektor $\vec{x}^{(p)}$ máme jednoznačně určen požadovaný výstup $\vec{y}^{(p)}$. Tato varianta rostoucího modelu využívá při učení právě i informaci o požadované třídě. Síť by měla být schopna po ukončení procesu učení určit odpovídající třídu i pro neznámá data bez informace o zadané třídě, ke které vzor patří.

Základem učení rostoucích Kohonenových map s učitelem jsou dle [7] tzv. RBF sítě, u kterých ale byly odstraněny jejich nedostatky. Pro bližší pochopení nyní stručně popíšeme strukturu RBF sítí (Moody & Darken) viz [25]. Skládají se z vrstvy L RBF jednotek s Gaussovskou aktivační funkcí a výstupní vrstvy m neuronů, realizující funkci lineárního součtu.

Každá RBF jednotka c má přiřazen vektor $\vec{w}_c \in \mathbb{R}^n$, který určuje pozici jednotky ve vstupním prostoru a směrodatnou odchylku σ_c . RBF jednotky jsou dále propojeny s výstupními neurony. Pro daný vstupní vzor $(\vec{x} \in \mathbb{R}^n, \vec{y} \in \mathbb{R}^m)$ je aktivace jednotky c dána podle [7] jako:

$$D_c(\vec{x}) = \frac{f_c(\vec{x})}{\sum_{i \in L} f_i(\vec{x})}, \quad (4.25)$$

kde

$$f_c(\vec{x}) = \exp\left(-\frac{\|\vec{x} - \vec{w}_c\|^2}{\sigma_c^2}\right). \quad (4.26)$$

Rovnice 4.25 provádí normalizaci, z čehož plyne že

$$\sum_{i \in L} D_i(\vec{x}) = 1. \quad (4.27)$$

Po předložení vstupního vzoru bude tedy součet aktivit RBF jednotek vždy stejný a roven 1. Každý výstup RBF jednotky je propojen se vstupem všech neuronů ve

výstupní vrstvě. Hodnoty vstupních signálů neuronů ve výstupní vrstvě dále závisí na hodnotě váhových vektorů w_c^{out} . Hlavním cílem při učení je nastavit parametry modelu tak, aby neurony ve výstupní vrstvě dávaly požadovaný výstup pro vstupní data.

Učení RBF sítí probíhá dle [7] ve 2 po sobě následujících fázích, první je učení bez učitele a druhá učení s učitelem:

1. RBF jednotky musí být rozmištěny po vstupním n -dimenzionálním prostoru. (Moody & Darken)[25] používají k tomuto účelu klastrovací algoritmus k-means.
2. Druhým krokem je nastavení váhových vektorů w_c^{out} tak, aby výstupní neurony dávaly požadovaný výstup. Tento krok je realizován pomocí metody minimizace druhé mocniny chyb.

Rozšíření struktury rostoucích Kohonenových map na strukturu podobnou RBF sítí provedeme dle [7] následovně:

- Váhový vektor \vec{w}_c neuronu c definuje střed Gaussovske aktivační funkce.
- Směrodatná odchylka σ_c RBF jednotky c je definována jako průměr délky všech hran vycházejících z neuronu c .
- U RBF sítí máme předem definovaný počet m výstupních neuronů, které jsou propojeny se všemi RBF jednotkami. To můžeme realizovat tak, že každému neuronu přiřadíme tzv. výstupní váhový vektor $\vec{w}_c^{out} = (w_{1c}, w_{2c}, \dots, w_{mc})$. Vektor w_{ic} značí váhu mezi RBF jednotkou c a výstupním neuronem i .

Jak jsme se zmínili v úvodu, rostoucí model s učením s učitelem se odlišuje od RBF sítí v několika bodech:

1. Místo 2 fázového procesu učení RBF sítí probíhá učení rostoucího modelu s učením s učitelem paralelně, tzn. samoorganizace RBF jednotek a druhá fáze učení s učitelem probíhají současně.
2. Klasifikační chyba vyskytující se při učení modelu na trénovacích datech je použita pro určení místa, kam bude vložen nový neuron.

Důležitý je především rozdíl, že nyní budeme provádět změnu výstupního váhového vektoru (krok s učitelem) hned po dokončení každého kroku adaptace neuronu (krok bez učitele). Další změnou je výpočet aktivace neuronu c , kterou vypočteme dle [7]:

$$D_c(\vec{x}) = \exp\left(-\frac{\|\vec{x} - \vec{w}_c\|^2}{\sigma_c^2}\right), \quad (4.28)$$

tudíž neprovádíme normalizaci jako u RBF sítí. Aktivaci m "výstupních neuronů" vypočítáme jako:

$$o_i = \sum_{c \in A} w_{ic} D_c \quad \forall i \in \{1, \dots, m\}, \quad (4.29)$$

kde A značí strukturu topologické mřížky. Úprava výstupního váhového vektoru je definována následovně:

$$\Delta w_{ic} = \eta(y_i - o_i) D_c \quad \forall i \in \{1, \dots, m\}, \forall c \in A, \quad (4.30)$$

kde η je vigilanční koeficient. Stejně jako u varianty s učením bez učitele zavedeme u neuronů lokální čítače τ . Lokální čítač vítězného neuronu s aktualizujeme podle [7] přičtením kvadratické chyby mezi aktuálním výstupem $\vec{o} = (o_1, \dots, o_m)$ a požadovaným výstupem $\vec{y} = (y_1, \dots, y_m)$:

$$\Delta \tau_s = \| \vec{y} - \vec{o} \|^2. \quad (4.31)$$

Pokud je model použit pro klasifikační problém, můžeme změnu čítače alternativně vyjádřit jako:

$$\Delta \tau_s = \begin{cases} 0 & \text{jestliže je } \vec{x} \text{ klasifikováno správně,} \\ 1 & \text{jinak.} \end{cases} \quad (4.32)$$

Nový neuron r je tedy vkládán do místa s největší klasifikační chybou. Procedura vkládání nového neuronu probíhá stejným způsobem jako u varianty s učením bez učitele (viz algoritmus 3 v kapitole 4.2.1). Po zvoleném počtu λ adaptačních a aktualizačních kroků provedeme vložení nového neuronu.

Hodnoty váhových vektorů a čítačů nastavujeme stejným způsobem jako u předchozí varianty, tzn. že se upravují podle vztahů 4.19 a 4.20. Rozdílem je pouze nastavení výstupního váhového vektoru, které u předchozího modelu není. Aktualizace výstupních váhových vektorů neuronů z okolí vítězného neuronu a inicializace výstupního váhového vektoru nového neuronu je velmi podobné jako u lokálních čítačů. Aktualizace u neuronů z okolí nově vloženého neuronu r se provádí dle [7] podle vztahu:

$$\Delta w_c^{out} = \frac{|F_c^{(nov.)}| - |F_i^{(puv.)}|}{|F_i^{(puv.)}|} w_r^{out} \quad \forall c \in N_r, \quad (4.33)$$

kde $|F_c|$ je n -dimenzionální objem oblasti F_c . Nakonec inicializaci výstupního váhového vektoru nově vloženého neuronu r provedeme dle [7]:

$$w_r^{out} = - \sum_{c \in N_r} \Delta w_c^{out}. \quad (4.34)$$

4.3 Metoda rostoucích neuronových plynů

Model učení, kterým se nyní budeme zabývat, je velmi podobný modelu rostoucích Kohonenových map popsanému v předchozí kapitole 4.2. Hlavní idea modelu, zveřejněna v [6], je přidávat nové neurony v průběhu procesu učení do malé počáteční sítě. Nové neurony vkládáme do sítě na základě informací, které jsme shromáždili v předešlém procesu učení. Tato myšlenka je velmi podobná způsobu vkládání nových neuronů ve Fritzkeho modelu [7], který má ovšem pevnou dimenzionalitu topologické mřížky (nastavena na hodnotu 2, 3 nebo více).

U rostoucích neuronových plynů je dle [6] topologie mřížky postupně zvětšována na základě tzv. Hebbovského učení (bude popsáno později spolu s algoritmem učení). Výsledná topologie modelu tedy závisí na vstupních datech a může se lokálně lišit.

Pro lepší manipulaci s topologickou mřížkou a jejími úpravami definuje model následovně:

- Množina neuronů A . Každý neuron $c \in A$ má přiřazen váhový vektor \vec{w}_c , který určuje pozici neuronu ve vstupním prostoru. Dále má každý neuron přiřazenu proměnnou, tzv. lokální čítač, který je využíván v procesu učení. Dle něho je určováno místo v topologické mřížce, kam bude vložen nový neuron.
- Množina hran N (vazeb) spojujících vždy páry neuronů. Tyto vazby nejsou vážené, jejich účel je čistě určovat spojení mezi 2 neurony. Hrany navíc obsahují informaci o jejich "stáří". Tato informace rozhoduje v procesu učení, zda budou dále zachovány či odebrány.

Nyní můžeme přejít k popisu algoritmu učení, který byl zveřejněn v [6].

Algoritmus 4: Učení rostoucích neuronových plynů.

1. Model začíná se 2 neurony a a b s náhodnými váhovými vektory \vec{w}_a a \vec{w}_b z prostoru vstupních vzorů \mathbb{R}^n .
2. Předložíme vstupní vzor $\vec{x} \in \mathbb{R}^n$.
3. V topologické mřížce najdeme nejbližší neuron ke vstupnímu vzoru (označme ho c_1) a dále druhý nejbližší neuron (c_2). Vzdálenost je opět počítána Euklidovskou metrikou jako ve všech předchozích modelech.
4. Zvýšíme o 1 "stáří" všech hran vycházejících z neuronu c_1 .
5. K lokálnímu čítači $error(c_1)$ neuronu c_1 připočítáme hodnotu vzdálenosti mezi neuronem c_1 a předloženým vstupním vzorem \vec{x} dle předpisu:

$$\Delta error(c_1) = \| \vec{w}_{c_1} - \vec{x} \| . \quad (4.35)$$

6. Ve vstupním prostoru aktualizujeme polohu neuronů c_1 a všech jeho přímých sousedů tak, že je posuneme směrem k předloženému vzoru o poměrnou část ϵ_b či ϵ_n :

$$\Delta \vec{w}_{c_1} = \epsilon_b (\vec{x} - \vec{w}_{c_1}) \quad \text{pro neuron } c_1, \quad (4.36)$$

$$\Delta \vec{w}_n = \epsilon_n (\vec{x} - \vec{w}_n) \quad \forall \text{ přímé sousedy } n \text{ neuronu } c_1. \quad (4.37)$$

7. Jestliže jsou neurony c_1 a c_2 spojeny hranou, pak nastavíme "stáří" této hrany na 0. Jestliže tato hrana ještě neexistuje, pak ji vytvoříme.

8. Odstraníme všechny hrany, jejichž "stáří" je větší než zvolená konstanta a_{max} .
9. Pokud počet vstupních vzorů předložených od předchozí aktualizace topologické mřížky přesáhl pevně zvolenou konstantu λ , vložíme do topologické mřížky nový neuron:

- Určíme neuron q s největší hodnotou lokálního čítače.
- Nový neuron r vložíme mezi neuron q a neuron f , který je určen jako topologický soused neuronu q s největší hodnotou lokálního čítače. Dále provedeme inicializaci váhového vektoru nového neuronu dle:

$$\vec{w}_r = 0.5 (\vec{w}_q + \vec{w}_f). \quad (4.38)$$

- Do množiny hran N vložíme hrany spojující neuron r s neurony q a f a odstraníme původní hranu mezi neurony q a f .
- Snížíme hodnotu lokálních čítačů neuronů q a f o poměrnou část α . Následně inicializujeme hodnotu lokálního čítače nového neuronu r na stejnou hodnotu jako je aktuální hodnota lokálního čítače u neuronu q .

10. Snížíme hodnotu lokálních čítačů všech neuronů o poměrnou část d .
11. Jestliže je splněna podmínka pro zastavení učení (např. předem daný počet iteračních kroků), pak ukončíme proces učení. Jinak přecházíme opět na krok číslo 1.

Šestý krok algoritmu představuje opět adaptační krok, kdy vítězný neuron a jeho sousedé v topologické mřížce adaptují své váhy směrem k předloženému vstupnímu vzoru. V průběhu učení jsou také odstraňovány hrany, jejichž koncové body (neurony) se dlouho neúčastní procesu adaptace. To je realizováno proměnnou hodnotou u každé hrany, tzv. "stářím"hrany. Přidáváním neuronů i hran a případným odebíráním hran v průběhu učení tak model konstruuje tzv. indukovanou Delaunayho triangulaci.

Tento model je velmi podobný Fritzkeho modelu [7]. U obou modelů si u každého neuronu udržujeme informaci tzv. lokálního čítače, který signalizuje, do kterých míst bude vložen nový neuron. Hlavním rozdílem ale je již zmíněný fakt, že Fritzkeho model udržuje pevnou topologii, kdežto u modelu rostoucích neuronových plynů se může topologie lokálně lišit.

Kapitola 5

Struktura složitějších modelů Kohonenových map

5.1 Vícevrstvé Kohonenovy mapy

5.1.1 Struktura modelu

Dále se budeme zabývat neuronovými sítěmi, které jsou složeny z více vrstev, jedná se o tzv. **vícevrstvé Kohonenovy mapy** viz [13]. Každá vrstva představuje Kohonenovu mapu popsanou v kapitole 3.1, jednotlivé vrstvy jsou propojeny tak, že výstup jedné vrstvy tvoří vstup vrstvy bezprostředně následující. Počet neuronů v topologických mřížkách jednotlivých Kohonenových map klesá směrem od vstupní vrstvy k výstupní, z čehož vyplývá pyramidální charakter tvar celkového modelu. Pro přehlednost budeme dále označovat jednotlivé Kohonenovy mapy tohoto modelu jako vrstvy, poslední vrstvou budeme chápát vrstvu s největším pořadovým číslem. Každá vrstva síť definuje tzv. reprezentativní vektory (bude podrobně vysvětleno později v procesu adaptace), celkový počet vrstev síť je úměrný počtu neuronů v topologických mřížkách jednotlivých vrstev. Poslední vrstva obsahuje jen několik málo neuronů, jejichž váhové vektory mohou představovat reprezentanty výsledných shluků.

První tj. vstupní vrstva slouží k reprezentaci vstupního vzoru a jeho předání neuronům v topologické mřížce první kompetiční vrstvy. Po dokončení procesu adaptace jsou váhové vektory neuronů této vrstvy dle [13] předkládány na vstup neuronů z vrstvy následující. Tento postup je opakován, dokud nedosáhneme poslední vrstvy. Jedná se vždy o jednosměrný dopředný přenos informace.

Označme množinu X_1 jako množinu vstupních vzorů $X_1 = \{\vec{x}_i; i = 1, \dots, k\}$, dále $W_1 = \{\vec{w}_i; i = 1, \dots, l\}$ množinu váhových vektorů neuronů uspořádaných na první topologické mřížce. Označme $O_1 \subseteq W_1$ jako podmnožinu váhových vektorů takových, které odpovídají vítězným neuronům první topologické mřížky po ukončení procesu adaptace. Tyto vektory se dle [13] nazývají tzv. reprezentativní vektory první vrstvy. Množina O_1 je dále použita jako vstup následující druhé vrstvy (tj. $X_2 = O_1$). Pouze

váhové vektory vítězných neuronů jsou však použity jako vstup následující vrstvy, protože dle [13] pouze ony jsou "správnými reprezentativními vektory" vstupních vzorů. Poznamenejme, že velikost této množiny není předem daná a je závislá na průběhu adaptace.

Jeden adaptační krok celého n -vrstvého modelu tak zle chápat jako zobrazení vstupní množiny X_1 na O_n pomocí $\prod_{i=1}^n TM_i$, kde TM_i je zobrazení i -té vrstvy modelu: $TM_i : X_i \rightarrow O_i$.

5.1.2 Metoda učení

Při učení vícevrstvého modelu Kohonenových map postupujeme po jednotlivých vrstvách, výstup jedné vrstvy je bezprostředně použit jako vstup vrstvy následující. Učení jednotlivých vrstev tohoto modelu probíhá dle [13] stejným způsobem jako základní algoritmus Kohonenova učení pro jednovrstvý model, který je podrobně popsán v kapitole 3.2. Celý algoritmus založený na základním algoritmu pro učení jednotlivých vrstev můžeme shrnout:

Algoritmus 5: Základní algoritmus pro učení vícevrstvých Kohonenových map.

. Postupně pro každou vrstvu opakujeme kroky (postupujeme směrem od vstupní vrstvy k výstupní):

. V každé vrstvě provedeme následující operace:

1. Určení vítězného neuronu spolu s množinou neuronů z jeho okolí.
Postupujeme stejným způsobem jako u základního modelu, kdy je vítězný neuron určen jako neuron s nejmenší vzdáleností od vstupního vzoru:

$$c = \operatorname{argmin}_{i=1,\dots,m} (d_i(t)), \quad (5.1)$$

kde index i prochází celou množinu neuronů topologické mřížky a d_i je vzdálenost váhového vektoru $\vec{w}_i(t)$ i -tého neuronu od předloženého vstupního vzoru $\vec{x}(t)$. Množina neuronů z jeho okolí je opět určena stejným způsobem jako u základního modelu viz Algoritmus 3.1.

2. Dalším krokem je adaptace vah vítězného neuronu a neuronů z jeho okolí dle vzorce:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha(t) \phi(c, j) (x_i(t) - w_{ij}(t)), \quad (5.2)$$

kde $j \in N_c$; $0 \leq i \leq N-1$; $\alpha(t)$ je vigilanční koeficient, $\phi(c, j)$ je funkce laterální interakce, která klesá s rostoucí vzdáleností neuronů od středu okolí N_c .

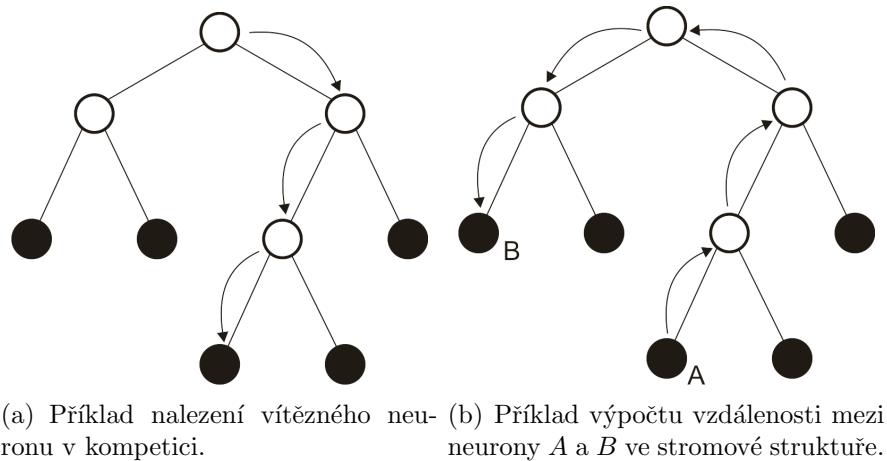
3. Váhový vektor vítězného neuronu se v tomto kroku stává prvkem množiny reprezentativních vektorů, které budou představovat vstup

následující vrstvy. Množina váhových vektorů vítězných neuronů v každé vrstvě se mění během procesu učení a nemusí odpovídat množině vítězných neuronů po naučení celého systému.

5.2 Evoluční stromy

Nyní se budeme zabývat novým modelem Kohonenových map, tzv. **evolučním stromem** [27]. Tento model se snaží kombinovat výhody předešlých známých modelů Kohonenových map, jako je flexibilnější topologie mapy, která by byla vhodná pro komplexní analýzu dat. Další přednost tohoto modelu je také snížení výpočetních nároků při učení modelů Kohonenových map, speciálně snížení doby potřebné pro nalezení vítězného neuronu pro daný vstupní vzor. Model využívá vlastnosti jak Fritzkeho [7] rostoucích Kohonenových map (kapitola 4.2), tak Kohonenovy mapy se stromovou strukturou představenou Koikkalainem a Ojou [19].

Na rozdíl od předešlých modelů jsou neurony v tomto modelu uspořádány do stromové struktury. Každý neuron má opět přiřazen váhový vektor, který budeme značit \vec{w}_i u neuronu s indexem i . Jako ve většině modelů s možností růstu sítě bude každý neuron i obsahovat lokální čítač b_i . Ten udává počet vstupních vzorů, pro něž se stal právě neuron i vítězem v kompetici. Jedná se o stejnou proměnnou, která byla použita u Fritzkeho rostoucího modelu (viz 4.2).



Obrázek 5.1: Základní operace modelu evolučního stromu.

Proces učení u modelu evolučního stromu začíná podle [27] s jedním neuronem. Inicializace tohoto neuronu je provedena tak, že jeho váhový vektor je nastaven do těžiště vstupních vzorů. Základní operací modelu je tzv. rozdělení neuronu. To znamená, že vytvoříme 2 nové neurony, které označíme jako tzv. potomky rozděleného neuronu a které se dočasně stávají tzv. listy stromové struktury. Váhové vektory potomků inicializujeme stejnou hodnotou jako má rodičovský (rozdělený) neuron. Pak při procesu učení je vítěz kompetice vybrán náhodně mezi potomky neuronu. Proces učení bude podrobně popsán později.

Po určení vítězného neuronu přichází na řadu adaptace váhového vektoru tohoto neuronu, což vede k tomu, že se váhové vektory jednotlivých neuronů navzájem

odlišují. Tímto krokem jsme tedy získali stromovou strukturu, která se skládá z jednoho rodičovského neuronu a jeho 2 potomků. Všechny operace, které budeme dále popisovat, se budou týkat pouze listů vytvářené stromové struktury.

Přejdeme nyní k části modelu, která je odlišná od ostatních modelů. Jde o metodu nalezení vítězného neuronu a vlastní algoritmus učení modelu stromové struktury. Pro ilustraci funkčnosti metody použijeme obrázek 5.1 zveřejněn v [27]. Proces nalezení vítězného neuronu je tzv. proces shora - dolů. Začínáme u kořenového neuronu, procházíme jeho potomky a najdeme ten neuron, jehož váhový vektor je nejbližší vstupnímu vzoru. Jestliže je tento neuron listem, pak se jedná o vítězný neuron. Pokud není, jsou prohledávány jeho potomci a je z nich opět vybrán neuron s nejmenší vzdáleností od vstupního vzoru. Proces opakujeme, dokud nalezený neuron není listem.

Po nalezení vítězného neuronu c je potřeba aktualizovat jak jeho váhový vektor, tak váhové vektory neuronů z jeho okolí. Aktualizaci provádíme klasickým způsobem dle vzorce:

$$\vec{w}_i(t+1) = \vec{w}_i(t) + \phi(c, i)(\vec{x}(t) - \vec{w}_i(t)), \quad (5.3)$$

kde $\phi(c, i)$ je funkce laterální interakce, c je index vítězného neuronu a index i prochází celou množinu neuronů. Tuto funkci volíme jako Gaussovskou funkci laterální interakce:

$$\phi(c, i) = \alpha(t) \exp\left(\frac{-\|\vec{r}_c - \vec{r}_i\|^2}{2\sigma^2(t)}\right), \quad (5.4)$$

kde vektory \vec{r}_c a \vec{r}_i udávají pozici neuronů c a i v topologické mřížce, $\alpha(t)$ je vigilanční koeficient a $\sigma(t)$ udává šířku Gaussovské funkce.

Parametry $\alpha(t)$ a $\sigma(t)$ jsou stejné jako v předešlých modelech. Nyní se budeme věnovat výpočtu normy $\|\vec{r}_c - \vec{r}_i\|$. U klasických modelů se symetrickou topologickou mřížkou je výpočet jednoduchý, u modelu evolučního stromu již musíme určit metodu vhodnou pro tento výpočet. Dle [27] byla zvolena jednoduchá metoda, jejíž princip je předveden na obrázku 5.1(b), která počítá normu jako nejkratší vzdálenost mezi 2 neurony.

Nejkratší vzdálenost mezi vítězným neuronem c a ostatními neuronami můžeme určit podle následujícího algoritmu. Základní myšlenka je počítat počet "přeskoků", které je nutno provést z vítězného neuronu do zadáного neuronu po nejkratší cestě. Přesná vzdálenost je definována jako počet "přeskoků" minus jedna. Jednička je odečtena z toho důvodu, že nás zajímá vzdálenost mezi neurony, které jsou listy. Nejbližší listy mají stejnýho rodiče, což znamená 2 přeskoky. Dle [27] bychom chtěli mít tuto vzdálenost rovnu 1, proto od počtu "přeskoků" odečítáme jedničku. Na obrázku 5.1(b) musíme provést 5 přeskoků, abychom se dostali z neuronu A do neuronu B . V této stromové struktuře je tedy vzdálenost neuronu A a neuronu B rovna 4.

K ukončení jednoho kroku učení je potřeba ještě aktualizovat hodnotu lokálního čítače b_c vítězného neuronu. To provedeme dle:

$$b_c(t+1) = b_c(t) + 1. \quad (5.5)$$

Jestliže po inkrementaci čítače dosáhne jeho hodnota předem zadané meze

$$b_c(t+1) = \theta, \quad (5.6)$$

rozdělíme vítězný neuron postupem, který jsme popsali na začátku této kapitoly. Rychlosť růstu sítě můžeme regulovat nastavováním nižší nebo vyšší hodnoty růstové meze θ . Pokud růst modelu nevyžadujeme, stačí zafixovat přírůstek lokálních čítačů na 0.

5.3 Metoda WEBSOM

Další metodou, kterou můžeme zařadit mezi pokročilejší - vícevrstvé modely Kohonenových map, je tzv. metoda **WEBSOM**¹ popsána v [11], [21] a [22]. Na úvod popíšeme příčiny vzniku a možnosti využití tohoto modelu. Dle [11] je dnes většina dokumentů dostupná v elektronické podobě, z čehož vyplývají požadavky na nástroje pracující s těmito kolekcemi dokumentů jako je vyhledávání, organizování, třídění a filtrování zajímavých dokumentů stejného oboru. Poznamenejme, že kolekce dokumentů jsou často velmi obsáhlé, a proto je potřeba zajistit efektivní nástroje, které s nimi budou pracovat.

WEBSOM je komplexní metoda pro práci s dokumenty a zahrnuje také metody pro předzpracování a úpravu dokumentů do požadovaných vstupních formátů. Metodami pro zpracování dokumentů se zabývat nebudeme, budeme se věnovat pouze části, která využívá Kohonenovy mapy.

Dle [11] je základem tohoto modelu 2-vrstvá Kohonenova mapa. Každý dokument je kódován jako histogram kategorií slov, z kterých je poté vytvořena jednoduchá Kohonenova mapa. Takto upravené dokumenty, z kterých jsou vytvořeny Kohonenovy mapy, slouží jako vstup druhé větší Kohonenově mapě, tzv. mapě dokumentů. Shluhy neuronů v první Kohonenově mapě signalizují jistou podobnost slov, naopak shluhy v druhé Kohonenově mapě značí podobnost celých dokumentů.

Funkčnost celého systému je stejná jako u vícevrstvého modelu popsáного v kapitole 5.1. V modelu nejsou použity žádné odlišné metody pro učení od klasických modelů. Volba jednotlivých parametrů se řídí stejnými pravidly jako u základního modelu (viz kapitoly 3.1 a 3.2). Model se podrobněji zabývá zpracováním slov a jejich reprezentací (podrobnosti viz [17] a [14]).

Proces zpracování dokumentů a vytváření WEBSOM modelu můžeme shrnout dle [11] následovně:

1. Konstrukce vektorů dokumentů.
 - Předzpracování textů.
 - Výpočet vektorů dokumentů jako vážených histogramů slov.
2. Konstrukce velké Kohonenovy mapy.
 - Inicializace malých Kohonenových map.
 - Učení malých map.

¹WEBSOM - Self-Organizing Maps for Internet Exploration.

- Opakování procesu odhadu velké Kohonenovy mapy na základě malých Kohonenových map.
3. Konstrukce uživatelského rozhraní.
 - Automatický proces vybírání charakteristických oblastí mapy.
 - Vytvoření potřebných součástí pro operace s uživatelským rozhraním.
 4. Možnost volby operací uživatelského rozhraní.
 - Procházení jednotlivých bodů.
 - Vyhledávání typu obsah - adresy.
 - Hledání klíčových slov.

Kapitola 6

Metody pro vizualizaci a shlukovou analýzu neuronů Kohonenových map

Předtím, než začneme jednotlivé modely testovat a analyzovat, seznámíme se s metodami vizualizace a klastrování Kohonenových map. Tyto metody budeme dále využívat v následujících kapitolách při testování a analýzách modelů.

6.1 Metody vizualizace Kohonenových map

Nyní se budeme zabývat zobrazováním výsledků Kohonenových map. Po skončení procesu učení Kohonenovy mapy chceme zobrazit 2-rozměrnou topologickou mřížku pro další analýzu výsledků. Obvykle jsou vstupními vzory Kohonenových map vysoce dimenzionální data a neurony topologické mřížky jsou rozmištěny v tomto vysoce dimenzionálním prostoru. Existuje několik metod řešících vizualizaci topologické mřížky. V této diplomové práci budeme používat k vizualizaci topologické mřížky především tzv. Sammonovo mapování, také se zmíníme o metodě používající U-matici. Na obrázku 6.1 je možno vidět porovnání obou zobrazení.

6.1.1 Sammonovo mapování

Naším úkolem je zobrazit pozice neuronů z vysoce dimenzionálního prostoru ve 2-dimenzionálním prostoru tak, abychom zachovali poměrné vzdálenosti mezi neurony ze vstupního prostoru. Tzn. že neuron, který má malou vzdálenost od jiného neuronu ve vstupním prostoru, by měl mít také malou vzdálenost od tohoto neuronu v 2-dimenzionálním prostoru. K řešení této úlohy využijeme tzv. **Sammonovo mapování**, které je podrobně popsáno v [33] a jeho varianty v [10] a [39]. Jedná se o nelineární zobrazení, které je založeno na myšlence zobrazení N L -dimenzionálních vektorů z L -dimenzionálního prostoru do méně dimenzionálního prostoru, které zachovává přibližnou strukturu dat. Tato struktura dat je zachovávána právě díky

udržování poměrných vzdáleností mezi neurony ve vstupním prostoru a v méně dimenzionálním prostoru.

Označme zobrazované vektory z L -dimenzionálního prostoru jako \vec{X}_i pro $i = 1, \dots, N$ a jím odpovídající vektory z d -dimenzionálního prostoru jako \vec{Y}_i pro $i = 1, \dots, N$. Pro nás případ položíme $d = 2$. Dále označme vzdálenost mezi 2 vektory \vec{X}_i a \vec{X}_j v L -dimenzionálním prostoru $d_{ij}^* \equiv d(\vec{X}_i, \vec{X}_j)$ a vzdálenost mezi odpovídajícími vektory \vec{Y}_i a \vec{Y}_j v d -dimenzionálním prostoru $d_{ij} \equiv d(\vec{Y}_i, \vec{Y}_j)$.

Inicializace Sammonova mapování začíná náhodným výběrem N vektorů z d -dimenzionálního prostoru:

$$\vec{Y}_1 = \begin{bmatrix} y_{11} \\ \vdots \\ y_{1d} \end{bmatrix}, \vec{Y}_2 = \begin{bmatrix} y_{21} \\ \vdots \\ y_{2d} \end{bmatrix}, \dots, \vec{Y}_N = \begin{bmatrix} y_{N1} \\ \vdots \\ y_{Nd} \end{bmatrix}.$$

Dále vypočteme vzdálenosti d_{ij} mezi všemi vektory $1 \leq i \leq j \leq N$ v d -dimenzionálním prostoru, které budeme dále používat při výpočtu chyby mapování E . Hodnota chyby mapování E značí, jak dobře odpovídá N vektorů z L -dimenzionálního prostoru odpovídajícím N vektorům z d -dimenzionálního prostoru. Vypočteme ji následovně:

$$E = \frac{1}{\sum_{i < j} d_{ij}^*} \sum_{i < j}^N \frac{(d_{ij}^* - d_{ij})^2}{d_{ij}^*}. \quad (6.1)$$

Jedná se tedy o funkci $d \times N$ proměnných $y_{p,q}$ pro $p = 1, \dots, N$ a $q = 1, \dots, d$, kterou se snažíme minimalizovat. Výpočet Sammonova mapování probíhá iterativně, kdy se v každém kroku snažíme opravit souřadnice vektorů \vec{Y}_i tak, abychom minimalizovali hodnotu chybové funkce. Na optimalizaci poloh bodů můžeme použít gradientní metodu, která vychází z náhodně vygenerovaných počátečních vektorů.

6.1.2 Zobrazení U-maticí

Dalším způsobem vizualizace topologické mřížky Kohonenovy mapy je metoda tzv. **U-matice**¹. Hned na začátku uvedeme jednu z velkých výhod Sammonova mapování oproti zobrazení U-maticí, a to možnost zobrazení libovolné topologické mřížky Kohonenovy mapy. U-matice totiž konstruujeme dle [38] nad topologickou mřízkou, což je u rostoucí Kohonenovy mapy dimenze > 2 (viz kapitola 4.2) nerealizovatelné ve 2D prostoru. Nyní přejdeme k samotné konstrukci U-matice.

Nechť c je neuron z topologické mřížky, potom označme množinu jeho sousedů (tj. neurony spojené s neuronem c hranou) N_c . Váhový vektor neuronu n označme obvyklým způsobem \vec{w}_n . U každého neuronu topologické mřížky zavedeme promennou:

¹V anglické literatuře se popisuje pojmem "U-matrix"- Unified distance matrix.

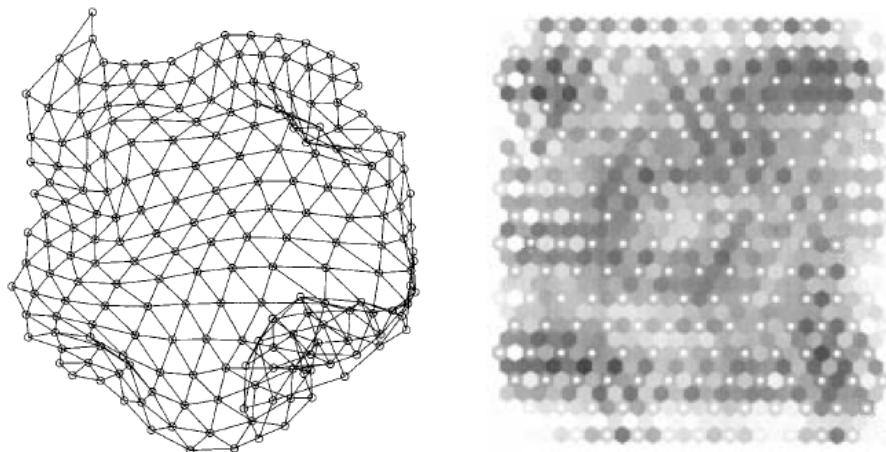
$$level(n) = \sum_{m \in N_n} d(\vec{w}_n, \vec{w}_m), \quad (6.2)$$

kde $d(x, y)$ je vzdálenost prvků x a y použitá při učení Kohonenovy mapy. Samotnou konstrukci U-matice provedeme zobrazením hodnot $level(n)$ na příslušné pozice neuronů v topologické mřížce pro všechny neurony.

U-matice se obvykle prezentuje jako obrázek ve stupních šedi. Na U-matici lze pohlížet jako na "terénní mapu" ve stupních šedi reprezentující vzdálenostní vztahy vstupních dat, z čehož vyplývají následující vlastnosti:

- Váhový vektor neuronu s velkou hodnotou $level(n)$ je velmi vzdálen od ostatních neuronů ve vstupním prostoru.
- Naopak váhový vektor neuronu s malou hodnotou $level(n)$ je obklopen neurony ze vstupního prostoru.
- Na "hranice hor" lze pohlížet jako na hranice různých klastrů.
- Naopak "údolí" značí centra klastrů.

Nakonec ještě poznamenejme, že U-matice ve stupních šedi se může vyskytovat i v inverzní podobě, tzn. centra shluků budou značena černou barvou, naopak hranice oblastí bílou barvou. Tímto způsobem je vyobrazena U-matice na obrázku 6.1.



(a) Zobrazení pomocí Sammonova mapování.

(b) Zobrazení pomocí U matice.

Obrázek 6.1: Porovnání 2 možných zobrazení topologické mřížky z [38].

6.2 Metody shlukové analýzy neuronů Kohonenových map

Po zobrazení topologické mřížky je v mnoha případech přínosné určit shluky neuronů, které reprezentují jednotlivé typy (kategorie) dat. Pokud topologická mřížka obsahuje stejný počet neuronů jako je počet kategorií dat, je ve většině případů každý neuron reprezentantem jedné kategorie. V praxi ale počet neuronů topologické mřížky převyšuje počet kategorií dat, a to někdy mnohonásobně. Více neuronů totiž dokáže mnohem přesněji approximovat rozložení vzorů ve vstupním prostoru. Extrémně velký počet neuronů také odpovídá fyziologickým vlastnostem lidského mozku, kde je počet neuronů neporovnatelně větší než ve zkoumaných modelech, a může se zde plně uplatnit princip kompetice. Neurony pak mohou soupeřit o právo reprezentovat předložený vzor.

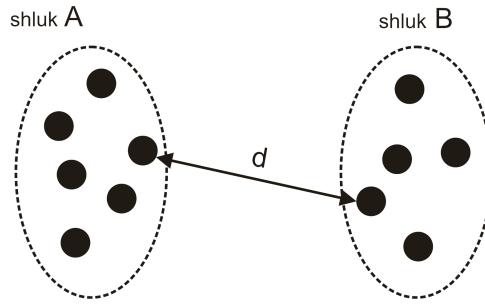
Analýza shluků je metoda, která se zabývá vyšetřováním vzájemné podobnosti objektů. Na začátku definujeme shluky podle [3] jako "Shluk je množina objektů, jejichž vzájemná podobnost je větší než jejich podobnost s objekty do shluku nepatřícími". Naším cílem je nyní najít shluky neuronů ve 2-rozměrné topologické mřížce, která už byla předtím vytvořena Sammonovým mapováním. Dle [24] existuje celá řada metod, kterými lze řešit problém nalezení shluků. V této práci použijeme jednu z metod využívající hierarchický shlukovací postup, a to tzv. **Wardovu metodu**. Stručně nastíníme princip několika metod, které jsou založeny na hierarchickém shlukovacím postupu, mezi nimiž je i Wardova metoda.

Obecně jsou hierarchické shlukovací metody založeny na hierarchickém uspořádání objektů a jejich shluků. Jak je popsáno v [24], budeme při hledání shluků používat *aglomerační shlukování*. To je založeno na myšlence, že 2 nejbližší objekty jsou sloučeny do prvního shluku a vypočte se nová matice vzdáleností, v níž jsou vynechány všechny objekty z prvního shluku, a naopak je tento shluk přidán do matice jako objekt. Postup se stále opakuje, dokud všechny objekty netvoří jeden shluk nebo je postup zastaven při dosažení předem daného počtu cílových shluků. Metody hierarchického shlukování se liší použitou metrikou. Používají se:

1. *Metoda nejbližšího souseda*: Zde je vzdálenost mezi 2 shluky definována jako minimální vzdálenost libovolného objektu z prvního shluku vzhledem k libovolnému prvku z druhého shluku. Praktické výsledky ukazují, že použitím této metody jsou objekty (shluky) přitahovány k sobě a dochází k vytváření dlouhých řetězců. Příklad je na obrázku 6.2. Formálně zapíšeme výpočet vzdálenosti mezi shluky A a B dle této metody následovně:

$$D_{min}(A, B) = \min_{a \in A} \min_{b \in B} d(a, b), \quad (6.3)$$

kde $d(a, b)$ je vzdálenost mezi objekty a a b , A a B jsou samotné shluky.

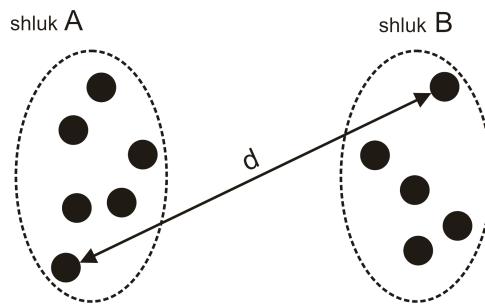


Obrázek 6.2: Příklad určení vzdálenosti u metody nejbližšího souseda.

2. *Metoda nejvzdálenějšího souseda:* Používá stejný princip měření vzdálenosti jako u předchozího případu, pouze místo minimální vzdálenosti je brána vzdálenost maximální. Dle praktických výsledků se metoda hodí na úlohy, kde jsou shluky přirozeně odděleny. Naopak se nehodí pro hledání shluků, které mají tendenci se řetězit (viz obrázek 6.3). Vzdálenost mezi shluky je počítána dle:

$$D_{max}(A, B) = \max_{a \in A} \max_{b \in B} d(a, b), \quad (6.4)$$

kde $d(a, b)$ je vzdálenost mezi objekty a a b , A a B jsou samotné shluky.

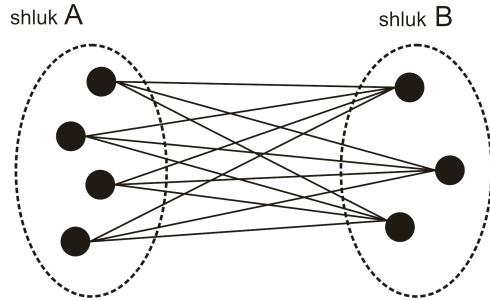


Obrázek 6.3: Příklad určení vzdálenosti u metody nejvzdálenějšího souseda.

3. *Metoda průměrné vzdálenosti:* U této metody je vzdálenost počítána jako průměrná vzdálenost mezi všemi objekty jednoho shluku vůči všem objektům druhého shluku. Metoda se může vyskytovat i ve vážené variantě, kdy jednotlivé vzdálenosti nemají stejný vliv na výpočet celkové vzdálenosti 2 shluků. Příklad je na obrázku 6.4. U metody průměrné vzdálenosti počítáme vzdálenost mezi shluky následovně:

$$D_{avg}(A, B) = \frac{1}{N_A \cdot N_B} \sum_{i=1}^{N_A} \sum_{j=1}^{N_B} d(a_i, b_j), \quad (6.5)$$

kde $d(a, b)$ je vzdálenost mezi objekty $a \in A$ a $b \in B$, A a B jsou samotné shluky, N_A a N_B jsou počty objektů ve shluku A respektive B .



Obrázek 6.4: Příklad určení vzdálenosti u metody průměrné vzdálenosti.

4. **Wardova metoda:** Metoda, kterou budeme používat při analyzování modelů v této práci, vychází z analýzy rozptylu (objektů ve shlucích). Metoda se snaží minimalizovat heterogenitu shluků, tím že minimalizuje přírůstek součtu čtverců odchylek objektů od těžiště shluku (značíme jej $E.S.S$ - z anglického výrazu "the error sum of squares"). V jednom kroku metody se spočítá přírůstek pro každé 2 shluky, který by vznikl jejich sloučením. Následuje spojení těch shluků, které mají minimální přírůstek. Jinými slovy můžeme říci, že metoda minimalizuje rozptyl uvnitř shluku. Příklad výpočtu $E.S.S$:

$$E.S.S.(A) = \sum_{i=1}^n (a_i - \bar{a})^2 = \sum_{i=1}^n \left(a_i - \frac{1}{n} \sum_{j=1}^n a_j \right)^2. \quad (6.6)$$

Vzdálenost mezi shluky je pak počítána dle:

$$D_{War}(A, B) = E.S.S.(AB) - [E.S.S.(A) + E.S.S.(B)], \quad (6.7)$$

kde AB je shluk vzniklý sloučením shluků A a B .

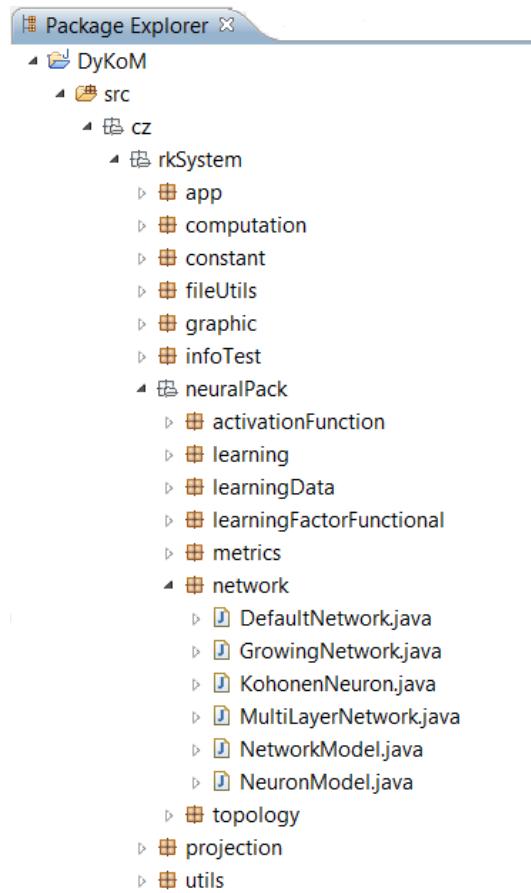
Kapitola 7

Návrh implementace

Nyní se stručně seznámíme s architekturou implementace celého projektu. Projekt **DyKoM** (dynamické Kohonenovy mapy) jsme implementovali v jazyce Java i s grafickým rozhraním.

7.1 Hlavní součásti systému

Celý projekt je rozdělen do tzv. balíčků dle jednotlivých funkčností systému. Projekt je navržen tak, aby mohl být jednoduše rozšířen o další modely či funkčnosti. Při implementaci jsme několik balíčků tříd základního modelu Kohonenových map použili z již existujícího projektu ”Java Kohonen Neural Network Library” od autorů Janusz Rybarski a Seweryn Habdank-Wojewódzki viz [32], který může být dále šířen pod licencí BSD. Tyto třídy jsou v projektu zřetelně označeny a tvoří asi 10% z celého systému, navíc byly značně upraveny k potřebám tohoto projektu. Kostra algoritmu Sammonova mapování byla převzata od autorů Matthew Careyho a Shalini Sewrazho [34]. Na obrázku 7.1 můžeme vidět celou hierarchii projektu DyKoM.



Obrázek 7.1: Hierarchie balíčků v projektu DyKoM.

Uvedeme všechny důležité balíčky, které jsme použili v projektu a popíšeme jejich funkčnost:

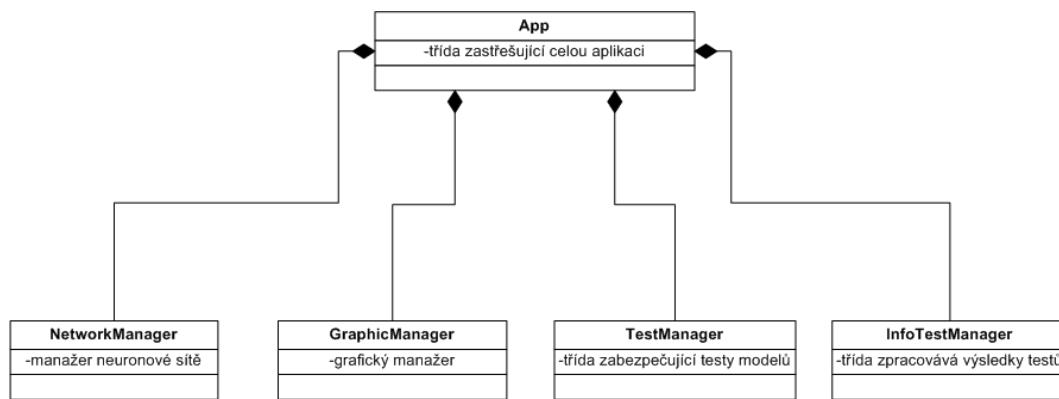
- Balíček **neuralPack** - představuje nejdůležitější součást projektu. Obsahuje jednotlivé modely, které jsou dále děleny na součásti, jež mohou být využity i u jiných modelů. Poznamenejme, že tyto součásti jsou vždy tvořeny abstraktní třídou se základními metodami, která se dále rozšiřuje pro potřeby konkrétního modelu. Tyto součásti jsou následující:
 - *topology* - obsahuje třídy potřebné pro práci s topologií modelu, tzn. metody pro určování množiny sousedů vítězného neuronu, topologii základního modelu (topologie je pravidelná mřížka) a topologie pro rostoucí modely umožňující přidávání neuronů.
 - *metrics* - skládá se z tříd pro výpočet vzdáleností vektorů za použití různých metrik. Jsou zde třídy pro Euklidovskou metriku, Cityblock metriku a další.

- *learningFactorFunctional* - balíček obsahuje různé varianty průběhu funkce vigilančního koeficientu, které mohou být navíc parametrizovány. Je zde možnost využít exponenciální, lineární, Gaussovy, konstantní a další průběhy funkcí.
 - *learningData* - tento balíček obsahuje třídy, které obsluhují vstupní data. Pomocí nich provádíme načtení vstupních dat, metody načítání můžeme parametrizovat (volba různých separátorů dat, permutace dat, atd.).
 - *network* - jedná se o jeden z nejdůležitějších balíčků tříd, který zastřešuje celé modely Kohonenových map (tzn. využívá metody předchozích balíčků). Obsahuje veškeré informace o modelu, jeho parametrech i topologii avšak mimo informaci o typu konkrétního učení. Tato informace se nachází v samostatném balíčku. V balíčku *Network* se nachází třídy zastupující základní, rostoucí a vícevrstvý model.
 - *learning* - jak je zmíněno v předchozím odstavci, tento balíček zabezpečuje učení modelů (určování vítězných neuronů, samotnou adaptaci, atd.). Jednotlivé třídy učení jsou upraveny speciálně pro různé modely, základní kostra učení je ovšem pro všechny modely shodná.
- Balíček **app** - představuje základ celé aplikace, obsahuje třídu *App* pro vytváření celé aplikace a třídu *NetworkManager* pro reprezentaci administrátora celé neuronové sítě (architektura bude popsána dále).
 - Balíček **graphic** - tento balíček se stará o veškeré grafické metody používané v rámci tohoto projektu jako je vykreslování topologické mřížky, uživatelských menu atd. Obsahuje rozhraní pro různá uživatelská menu a informační panely. Dále obsahuje třídu *GraphicManager* reprezentující grafického administrátora projektu.
 - Balíček **constant** - balíček obsahuje hodnoty konstant pro celou aplikaci. Jsou to jak konstanty pro grafické vykreslování, tak pro modely, vstupní data a akce prováděné na modelech.
 - Balíček **projection** - jak již plyne z názvu, balíček se bude zajišťovat projekce mezi různě dimenzionálními prostory. Hlavním úkolem třídy *SammonProjection* je provést zmiňované Sammonovo mapování z m -dimenzionálního do 2-dimenzionálního prostoru. Třída *DefaultProjection2D* provádí projekci do vykreslovaného okna (projekce do 2D prostoru).
 - Balíček **computation** - balíček obsahuje důležitou třídu *ComputationThread*, která vytváří vlákno pro výpočty nad modely Kohonenových map a zabezpečuje obsluhu veškerých procesů s modely. Dále obsahuje třídu *Permutation* pro výpočet permutací vstupních dat a další třídy používané pro třídění datových struktur.

- Balíček **infoTest** - poslední balíček vykonává všechny testy v projektu, třídy jako *TestManager* a *InfoTestManager* se starají jak o vykonávání testů, tak výpočet výsledných statistik testů a hledání shluků ve výsledných topologických mřížkách.

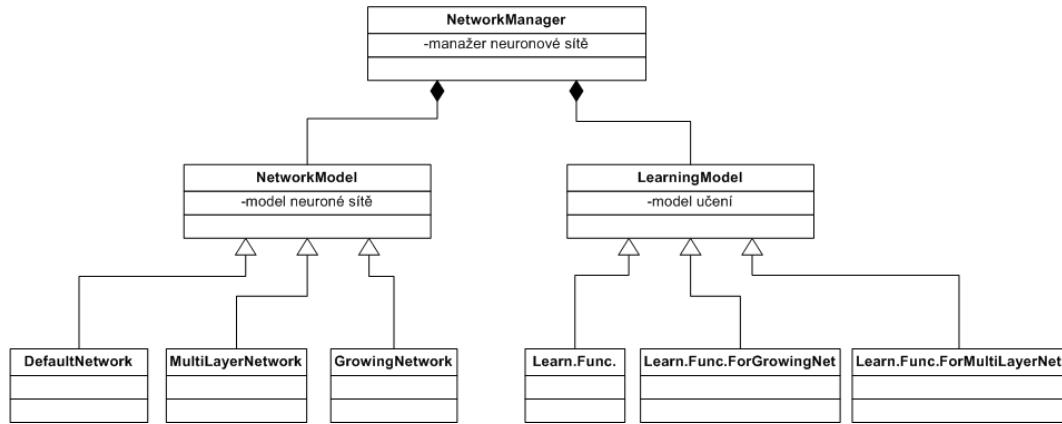
7.2 Architektura systému

Pro další možné rozšiřování projektu a lepší porozumění systému uvedeme nyní objektovou architekturu celého projektu. Na obrázku 7.2 je znázorněna základní architektura celého projektu. Základní třída aplikace *App* obsahuje 4 nezávislé moduly, a to manažera neuronové sítě *NetworkManager*, grafického manažera *GraphicManager*, manažera pro testování *TestManager* a manažera pro zobrazování výsledků *InfoTestManager*. Jednotlivým modulům se budeme věnovat podrobně dále. V konstruktoru aplikace probíhá inicializace všech těchto modulů.



Obrázek 7.2: Architektura celé aplikace.

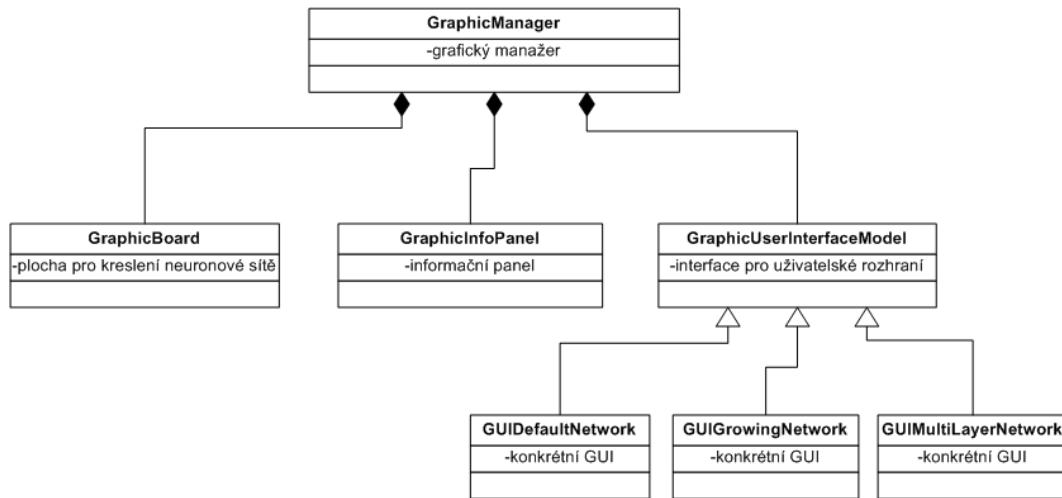
Nyní se budeme podrobně zabývat třídou síťového manažera *NetworkManager*. Tato třída se stará o vytváření a učení všech porovnávaných modelů Kohonenových map. Architekturu třídy je možno vidět na obrázku 7.3.



Obrázek 7.3: Architektura manažera sítě *NetworkManager*.

Třída obsahuje metody pro vytváření a učení jednotlivých modelů Kohonenových map jako jsou *CreateDefaultNetwork(...)*, *CreateGrowingNetwork(...)* atd. pro vytváření modelů a metody *LearningKohonenNet(...)*, *LearningKohonenNetWithGrowing(...)* atd. pro učení modelů.

Další významnou třídou je grafický manažer ***GraphicManager***, který se stará o veškeré vykreslování grafických komponent v projektu. Jeho objektový návrh je na obrázku 7.4.



Obrázek 7.4: Architektura grafického manažera *GraphicManager*.

Důležitým prvkem třídy je metoda *Init(...)*, kde proběhne vytvoření hlavního grafického prostředí *GraphicBoard* (kde budou vykreslovány topologické mřížky mo-

delů Kohonenových map) a všech uživatelských menu. Dále je to metoda *RePaint(...)*, která překreslí veškeré grafické komponenty aplikace. Metodou *SetGUI(...)* lze nastavit aktuální uživatelské menu.

Třetím důležitým modulem je třída ***TestManager***, která obsluhuje veškeré testy modelů. Jednoduchý test aktuálního modelu se provádí zavolením metody *TestModel(...)*. Každý test je parametrisován konkrétními vstupními daty a dalšími atributy viz programátorská příručka projektu. Součástí třídy je také metoda *CheckResult(...)*, která po ukončení učení zkонтroluje výsledné shluhy a výsledky uloží do datové struktury. Kontrolou rozumíme počet správně a špatně přiřazených vzorů k jednotlivým shlukům. Automatický test modelů Kohonenových map provedeme metodou *AutoTestingModel(...)*, který je parametrisován počtem opakujících se testů.

Posledním modulem je třída ***InfoTestManager***, která obstarává další potřebné akce související s testováním modelů jako hledání shluhy neuronů v topologických mřížkách před samotnou kontrolou atd. Metodou *ClusterData(...)* rozdělíme vstupní data podle nalezených shluhy v topologické mřížce do určitého počtu souborů (počet souborů odpovídá množství shluhy). Metoda *CheckClick(...)* slouží pro kontrolu, zda jsme kurzorem myši nevybrali určitý neuron. Pokud ano, metoda *SetInfo(...)* dokáže vypsat informace o vybraném neuronu do informačního panelu.

Ukázka zdrojového kódu z třídy *AdaptiveModLearningFunction.java*:

```
/*
 * provede změnu vah neuronu
 * @param cislo vitezneho neuronu
 * @param vstupni vektor
 * @param pocet iteraci
 */
public void changeWeight(int neuronNumber, double[] vector,
    int iteration) {
    // zmeni vahy vitezneho neuronu - vzdal. je proto 0
    changeNeuronWeight(neuronNumber, vector, iteration, 0);
    // vypocet polomeru
    int radius = (int)(neighbourhoodSizeFunction.
        getValue(iteration));
    topology.setRadius(radius);
    // urci mnozinu neuronu z "okoli" vitezneho neuronu
    TreeMap<Integer, Integer> neighbourhood = topology.
        getNeighbourhood(neuronNumber);
    Iterator<Integer> it = neighbourhood.keySet() .
```

```
        iterator();
int neuronNr;
// zmeni vahy neuronu v "okoli" vitezneho neuronu
while (it.hasNext()) {
    neuronNr = (Integer)it.next();
    changeNeuronWeight(neuronNr, vector,
                        iteration, (Integer)neighbourhood.get(
                            neuronNr));
}
}

protected void changeNeuronWeight(int neuronNumber, double []
    vector, int iteration, int distance) {

    // vahovy vektor neuronu
    double [] weightList = networkModel.getNeuron(
        neuronNumber).getWeight();
    // delka vahoveho vektoru
    int weightNumber = weightList.length;
    double weight;

    // komentar
    if (showComments) {
        String vectorText = "[";
        for (int i = 0; i < vector.length; i++) {
            vectorText += vector[i];
            if (i < vector.length - 1) {
                vectorText += ", ";
            }
        }
        vectorText += "]";
        System.out.println("Vector: " + vectorText);
        String weightText = "[";
        for (int i = 0; i < weightList.length; i++) {
            weightText += weightList[i];
            if (i < weightList.length - 1) {
```

```
                weightText += ", ";
            }
        }
        weightText += "]";
        System.out.println("Neuron " + (neuronNumber
            ) + " weight before change: " + weightText
            );
    }

// vypocet noveho vahoveho vektoru
for (int i = 0; i < weightNumber; i++) {
    weight = weightList[i];
    weightList[i] += learningFactor.getValue(
        iteration) * neighborhoodFunction.
        getValue(distance) * (vector[i] - weight);
}

// ulozim novou hodnotu vahoveho vektoru
networkModel.getNeuron(neuronNumber).setWeight(
    weightList);

// komentár
if (showComments) {
    String weightText = "[";
    for (int i = 0; i < weightList.length; i++)
    {
        weightText += weightList[i];
        if (i < weightList.length - 1) {
            weightText += ", ";
        }
    }
    weightText += "]";
    System.out.println("Neuron " + (neuronNumber
        ) + " weight after change: " + weightText)
        ;
}
```

Kapitola 8

Analýza a testování modelů

Naším úkolem je také zjistit a ověřit některé vlastnosti modelů Kohonenových map, jako jsou dobrá approximace pravděpodobnostního rozložení vstupních dat, rychlosť učení, robustnost modelů a další. Testování a experimenty budeme provádět nejprve na umělých datech, kdy máme k dispozici správné výsledky a také hodnoty výsledků naměřené při testování jiných modelů.

Cílem je zjistit, které z modelů jsou vhodné pro řešení ”reálné” úlohy a ty použít při zpracování reálných dat. Budeme analyzovat data o nehodovosti v silniční dopravě ve městě Ostravě. Bude se jednat o zpracování velkého objemu dat, kdy dopředu neznáme rozložení dat v příznakovém prostoru a navíc se toto rozložení může v průběhu času měnit (výstavba obchvatu, práce na komunikaci a tudíž uzavření části města pro silniční dopravu, změna rychlosti atd.). Navíc mohou být data zatížena chybou. Proto nejprve modely otestujeme na umělých datech a poté přejdeme k řešení ”reálné” úlohy.

8.1 Požadavky kladené na modely

Modely Kohonenových map budeme testovat na několika typech úloh dle [8]:

1. Zaprvé se bude jednat o klasifikační úlohy. V tomto případě je úkolem rozdělit data do předem známého počtu kategorií. Vstupní data budou obsahovat informaci o příslušné třídě, ale tuto informaci při učení modelu nevyužijeme. Ta bude použita až při kontrole klasifikačních schopností modelů. K testování použijeme několik souborů dat, které budou mít různou dimenzionalitu vstupních vzorů.
2. Druhým typem úloh jsou deskripcní úlohy. Cílem je nalézt dominantní struktury (shluhy) a vazby, které jsou skryté v daných datech. Jedná se většinou o zkoumání neznámých dat, kdy neznáme ani pravděpodobnostní rozložení dat.

Modely použitelné pro řešení výše uvedených typů úloh by proto měly vyhovovat požadavkům, které následně uvedeme. Po rozboru vlastností modelů Kohonenových

map budeme pokračovat jejich ověřením a otestováním na různých souborech dat. Tyto experimenty budou popsány v kapitole 8.3. Na konci celé kapitoly poté provedeme analýzu a zhodnocení výsledků. Požadavky kladené na modely jsou následující:

- Nejdůležitějším požadavkem, který klademe na modely Kohonenových map je, aby dobře approximovaly hustotu pravděpodobnostního rozdělení trénovacích vzorů. V takovém případě by měl mít každý neuron zhruba stejnou pravděpodobnost, že se vstupní vzor (rozložení vstupních vzorů odpovídá trénovací množině) zobrazí právě na něj. Formálně je cílem najít vhodné zobrazení ze vstupního prostoru \mathbb{R}^n do diskrétního k -dimenzionálního prostoru, který označíme A (topologická mřížka):

$$\phi_w : \mathbb{R}^n \rightarrow A. \quad (8.1)$$

- Sítě by měla vykazovat jisté prvky robustního chování. Pojem robustnost můžeme dle [20] chápat jako zobecnění 3 různých vlastností a požadavků:
 - Prvním typem robustnosti je vlastnost výstupu sítě, který by měl mít co nejmenší citlivost na změny vstupních dat a poškození. Tento typ robustnosti testujeme v naprosté většině případů pouze u vícevrstvých dopředných neuronových sítí. Zde při konstrukci sítě volíme váhy jako dostatečně malé hodnoty.
 - Druhým typem robustnosti dle [20] je odolnost vůči poškození sítě. Většinou se jedná o poškození váhového vektoru, např. zafixování některé složky váhového vektoru neuronu na konstantní hodnotu nebo nastavení náhodných hodnot ve váhovém vektoru neuronu po ukončení procesu učení. Tento problém můžeme řešit dostatečným počtem neuronů v topologické mřížce, které mohou nahradit funkci poškozeného neuronu.
 - Robustnost sítě můžeme chápat také z hlediska nastavení parametrů sítě. Změna hodnot parametrů sítě u robustních modelů nezpůsobí výrazné změny výstupu sítě a topologická mřížka se ustálí ve stejné poloze. U méně robustních modelů může ale tato změna vést ke zhroucení topologické mřížky a úplnému znehodnocení výsledků učení sítě.
- Požadujeme, aby proces učení a rozpoznávání sítě proběhl v co nejkratší době, tzn. že chceme minimalizovat počet kroků učení i rozpoznávání.
- Upřednostňujeme sítě s malým počtem neuronů. Z tohoto požadavku vyplývají menší paměťové nároky a také rychlejší doba učení, kdy ve fázi určování vítězného neuronu nemusíme procházet velkou množinu neuronů topologické mřížky.

8.2 Rozbor jednotlivých modelů

8.2.1 Rozbor základního modelu Kohonenovy mapy s učením bez učitele

Při analýze tohoto modelu musíme vycházet ze skutečnosti, že se jedná o základní model, od kterého jsou většinou odvozeny všechny ostatní složitější modely. Mezi hlavní výhody těchto modelů obecně patří snadná interpretace získaných výsledků, kde vzory blízké ve vstupním prostoru jsou většinou reprezentovány také blízkými neurony v topologické mřížce. Naopak odlišné vstupní vzory jsou typicky reprezentovány neurony velmi vzdálenými v topologické mřížce. Z jiného pohledu ale můžeme brát vytvářenou strukturu topologické mřížky jako nevýhodu. Interpretace výsledků může za jistých okolností záviset na subjektivním pohledu analytika a na jeho zkušenostech z předchozích případů. Tento problém se vyskytuje hlavně u deskripčních úloh, kdy vazby mezi daty teprve hledáme a neznáme dopředu např. odpovídající počet shluků.

Problém můžeme demonstrovat přímo na příkladě dat nehodnosti, která budeme analyzovat dále. Pokud analytik dostane výslednou topologickou mřížku Kohonenovy mapy bez znalosti dat, nemůže např. správně rozhodnout o počtu shuků (je např. zbytečné hledat více než 10 shluků - shluky by zastupovaly pouze velmi malé procento dat a neměly by patřičnou vypovídající hodnotu).

Základním modelem Kohonenových map, který je založen na principu učení bez učitele, lze dosáhnout podle [29] velmi dobrých výsledků při rozpoznávání různých typů (datových) vzorů (písma, hlasu atd.) a klastrování dat. Výborných výsledků lze ale často dosáhnout jen za předpokladu správné volby parametrů modelu. Tato volba není jednoznačná a není předem daná. Při volbě parametrů modelu vycházíme částečně z matematického popisu modelu, který ale pouze zhruba vymezuje interval volby parametrů. Ve většině případů jsou parametry závislé na řešené úloze a tudíž znalostech analytika.

Při učení modelu vyžadujeme konvergenci topologické mřížky ke stálé neměnné struktuře, tzn. konvergenci aktualizovaných váhových vektorů neuronů ke stálým hodnotám. Požadujeme, aby se neurony v topologické mřížce ustálily v (sub)optimálních polohách. Jedním z předpokladů tohoto jevu je správná volba vigilančního koeficientu $\alpha(t)$ viz sekce 3.1. Ten může nabývat dle [29] hodnot z intervalu $(0 \leq \alpha(t) \leq 1)$ a jeho hodnota by měla klesat s časem.

Pokud zvolíme lineární pokles koeficientu v závislosti na čase, dochází dle [29] ke dvěma problémům při učení sítě. Zaprvé je pokles vigilančního koeficientu na počátku učení velmi malý, tudíž se váhové vektory neuronů v topologické mřížce adaptují o velké hodnoty (velká adaptace) a nemusejí najít své (sub)optimální polohy v topologické mřížce. Naopak pokles koeficientu na konci učení je příliš velký, takže neurony nedokáží doladit hodnoty váhových vektorů (váhové vektory určují pozice neuronů v topologické mřížce). Jako vhodná volba se jeví exponenciální pokles vigilančního

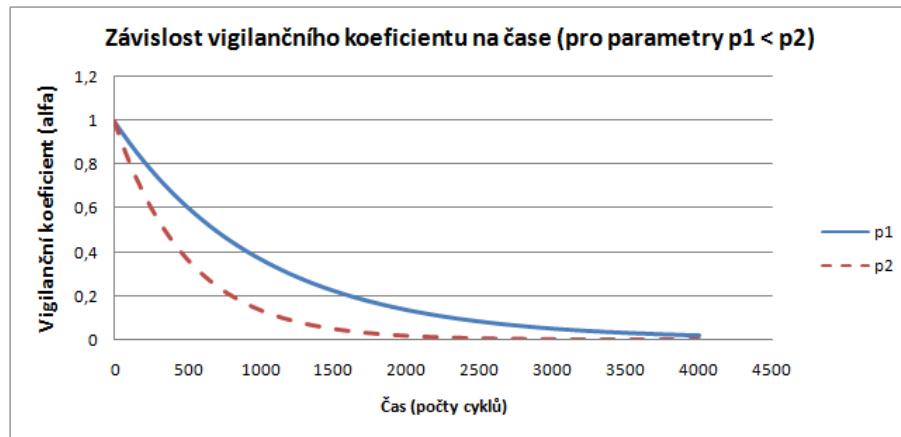
koeficientu. Vigilanční koeficient $\alpha(t)$ se pak tedy upravuje podle vztahu:

$$\alpha(t) = e^{-tp}, \quad (8.2)$$

kde t je čas učení a p je parametr, udávající rychlosť poklesu vigilančního koeficientu. Menší hodnota parametru p prodlouží dobu konvergencie, ale naopak by měla pomoci najít kvalitnější řešení. Volba lineárního poklesu koeficientu není vhodná. Ovšem v kombinaci, kdy nejdříve volíme exponenciální pokles vigilančního koeficientu a v druhé polovině adaptačních kroků změníme pokles na lineární, můžeme obdržet velmi dobré výsledky. Požadavky na volbu vigilančního koeficientu $\alpha(t)$ můžeme podle [29] shrnout do 2 bodů:

- $\sum_{t=t_0}^{\infty} \alpha(t) = \infty$,
- $\sum_{t=t_0}^{\infty} \alpha^2(t) < \infty$.

Tyto požadavky na volbu parametru zaručují, že neurony budou mít možnost (do stateku času), aby hodnoty jejich váhových vektorů konvergovaly ke stálým neměnným hodnotám (tzn. že neurony dosáhnou svých pevných pozic v topologických mřížkách). Druhá podmínka zaručuje, že konvergence nebude probíhat zbytečně dlouho. Lineární pokles koeficientu tyto podmínky nesplňuje, a proto je také vhodné u exponenciálního poklesu volit hodnotu parametru p v intervalu $\frac{1}{2} \leq p < 1$.

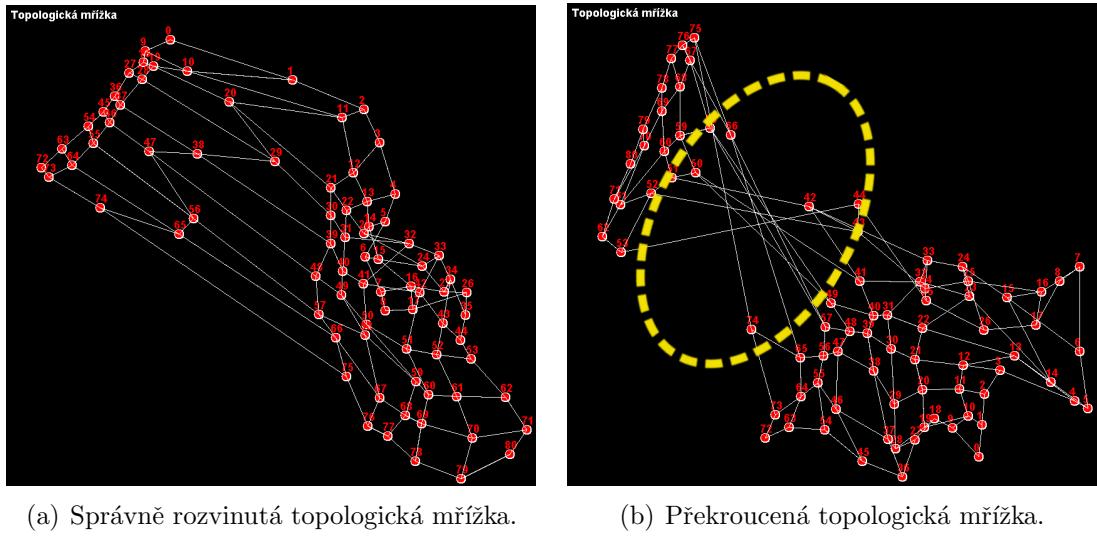


Obrázek 8.1: Závislost vigilančního koeficientu na čase.

Dalším důležitým parametrem je velikost okolí vítězného neuronu. Ta je částečně závislá na zvolené topologii mřížky. Obecně by mělo být okolí vítězného neuronu dle [37] a [2] na začátku učení velké, tzn. že se adaptace účastní i neurony vzdálenější od vítězného neuronu. Postupem času by se mělo okolí neuronu zmenšovat, na konci učení by mělo být okolí nulové, tzn. že adaptace se bude účastnit pouze vítězný neuron.

Průběh poklesu velikosti okolí by měl být opět vhodně volen vzhledem k řešené

úloze. Pokud bychom na počátku učení zvolili rychlé klesání velikosti okolí neurona, mohlo by dojít dle [37] a [2] k tzv. topologickým zvratům (překroucení nebo úplnému zhroucení struktury topologické mřížky), příklad je uveden na obrázku 8.2. Problém může také nastat v případě, jestliže dojde k rychlému poklesu (velkému skoku) velikost okolí na malou hodnotu. Tato operace může způsobit tzv. ”zamrznutí sítě” v tzv. metastabilních stavech, v horším případě i mimo lokální minima.



Obrázek 8.2: Porovnání 2 výsledných topologických mřížek.

8.2.2 Rozbor rostoucích Kohonenových map

Tento model byl vyvinut jako zobecnění několika variant Kohonenových map a měl by odstraňovat jejich předešlé nedostatky a naopak využít jejich dobrých vlastností. Právě pevná volba odpovídající topologie sítě na začátku procesu učení, která není u modelů triviální, je u tohoto modelu odstraněna. V průběhu učení jsou vkládány nové neurony do míst s větší hustotou pravděpodobnosti výskytu vstupních vzorů, a tak si síť sama určuje vhodnou topologii.

Jistého zjednodušení bylo dosaženo také ve volbě vigilančního koeficientu a velikosti okolí vítězného neurona. Vigilanční koeficient je konstantní v průběhu celého procesu učení. Experimentálními pokusy dle [7] byly zjištěny jako optimální hodnoty $\epsilon_b = 0, 1$ a $\epsilon_n = 0.006$. Okolí neuronu je definováno jako množina přímých sousedů tohoto neuronu, tudíž velikost okolí je rovna 1. Na druhou stranu musíme poznamenat, že určení přímých sousedů není tak jednoduché jako u základního modelu, kde je topologie v průběhu celého učení stejná. Zde je topologie vždy po určitém počtu kroků učení změněna vložením nového neurona.

8.3 Experimenty a testování na ”umělých” datech

Pokračujeme testováním modelů na referenčních datech. Jedná se o speciální veřejně dostupná data používaná běžně pro ověření vlastností studovaných modelů. Dalším typem dat mohou být vlastní vygenerovaná data pro ověření konkrétních vlastností modelů nebo data ”dobře známá”, která se velmi často používají pro testování podobných modelů, a tudíž můžeme jejich výsledky porovnávat s výsledky našich testovaných modelů.

8.3.1 Iris data

Prvním testem, kterým ověříme vlastnosti modelů, je test na tzv. **Iris datech** (Ronald Aylmer Fisher). Jedná se o velmi známý soubor dat, na kterém se často ověřuje schopnost klasifikace různých modelů. Soubor dat obsahuje 50 vzorů od každého z 3 druhů kosatců, tj. druhů rostlin. Soubor dat je složen z druhů *Iris setosa*, *Iris virginica* a *Iris versicolor*. Každý vzor je popsán 4-rozměrným vektorem (s kategorií 5-rozměrným vektorem), jenž reprezentuje vlastnosti daného vzoru.

Popis atributů dat:

1. Délka oddělujícího okvětního lístku v centimetrech.
2. Šířka oddělujícího okvětního lístku v centimetrech.
3. Délka okvětního lístku v centimetrech.
4. Šířka okvětního lístku v centimetrech.
5. Druh kosatce (3 možnosti) - požadovaný výstup.

Úkolem je tedy vytvořit model, který bude schopen rozdělit daná data do 3 kategorií. Při učení modelů nevyužíváme znalost příslušné třídy vstupního vzoru, jedná se o učení bez učitele. Kategorie vstupního vzoru využijeme až při konečné kontrole, zda se ve 3 detekovaných shlucích dat vyskytují výhradně odpovídající vzory.

Při testování postupujeme následovně:

1. Učení modelu dle zadaných parametrů.
2. Pro zobrazení výsledné topologické mřížky využijeme Sammonovo mapování (viz kapitola 6.1.1).
3. Shluky neuronů reprezentující výsledné kategorie nalezneme pomocí Wardova algoritmu (viz kapitola 6.2) a neurony okalibrujeme patřičnou kategorií.
4. Dále provedeme rozdelení dat, tj. pro každý vstupní vzor určíme vítězný neuron (viz proces rozpoznávání v kapitole 3.1) a podle kategorie vítězného neuronu zařadíme vzor do příslušného shluku.

5. Provedeme kontrolu kategorií jednotlivých vzorů ve výsledných shlucích a vyčteme statistiky testu (jako např. průměrnou a maximální procentuální úspěšnost).

Proces testování automaticky opakujeme několikrát. V každém testu rozdělíme všechna data na trénovací a testovací, na kterých ověřujeme klasifikační schopnosti modelu. Zvolíme pevný poměr rozdělení dat, kdy $\frac{7}{10}$ použijeme pro trénování. Data jsou v tomto poměru v každém testu vybrána podle náhodné permutace. Výsledky jednotlivých modelů jsou uvedeny v tabulce 8.1.

Tabulka 8.1: Výsledky modelů pro Iris data I.

Model	Počet testů	Vstupních vzorů (Trén./Test.)	Procentuální úspěšnost na test. množině		Směr. odchylka	Čas výpočtu
			Prům.	Max.		
Základní m.	1000	150 (105/45)	86,5%	100%	6,8%	170 min.
Rostoucí m.	1000	150 (105/45)	87,5%	100%	6,9%	26 min.
Vícevrstvý m.	1000	150 (105/45)	87,9%	100%	5,3%	55 min.

Při testování byly parametry modelů a učení nastaveny následovně:

- Základní model - velikost topologické mřížky: 9 x 9 neuronů, vigilanční koeficient: $\alpha(t) = e^{-0.005 \cdot t}$, počet adaptačních kroků: 3000.
- Rostoucí model - počáteční dimenze sítě: 2 (3 počáteční neurony), parametr $\epsilon_b = 0.15$, parametr $\epsilon_n = 0.08$, parametr $\alpha = 0.002$, počet adaptačních kroků 3000, počet adaptačních kroků pro vložení nového neuronu $\lambda = 100$.
- Vícevrstvý model - počet vrstev: 3, velikosti topologických mřížek: 8 x 8, 4 x 4, 3 x 1, vigilanční koeficienty jednotlivých vrstev: $\alpha_1(t) = e^{-0.005 \cdot t}$, $\alpha_2(t) = 0.7 \cdot e^{-0.003 \cdot t}$, $\alpha_3(t) = 0.1 \cdot e^{-0.001 \cdot t}$, počet adaptačních kroků: 3000.

Nejlepších výsledků v testu dosáhl rostoucí a vícevrstvý model. Vícevrstvý model měl nejmenší rozptyl výsledků, ovšem rychlosť učení rostoucího modelu byla výrazně vyšší oproti ostatním, což spolu s vysokou klasifikační úspěšností činí tento model nejlepším v testu na Iris datech.

8.3.2 Test robustnosti modelů

Další testovanou vlastností modelů je robustnost. Pod tímto pojmem rozumíme vlastnost ve smyslu 2. bodu "robustnosti" v kapitole 8.1, tzn. odolnost vůči poškození

sítě. Budeme simulovat poškození váhového vektoru po ukončení procesu učení. To lze provést jak zafixováním některé složky váhového vektoru neuronu na konstantní hodnotu, tak nastavením náhodných hodnot ve váhovém vektoru neuronu po ukončení procesu učení. V této práci budeme simulovat poškození druhým způsobem - nastavením náhodných vektorů u vybraných neuronů.

Abychom mohli objektivně analyzovat a porovnávat robustnost modelů, budeme simulovat poškození váhových vektorů u 25% neuronů v topologických mřížkách. Pokud bychom zvolili konstantní hodnotu počtu neuronů, u kterých provedeme poškození váhových vektorů, znevýhodnili bychom některé modely, především rostoucí model. To by bylo zapříčiněno tím, že rostoucí model obecně obsahuje málo neuronů v topologické mřížce, a tudíž poškození sítě by bylo velké. Rostoucí model obsahuje na začátku učení malý počet neuronů a nové neurony vkládá do míst s velkou hustotou výskytu vstupních vzorů, proto nepotřebuje velký počet neuronů pro dobrou approximaci rozložení vstupních vzorů. Poškození sítě u hierarchického modelu jsme simulovali na každé vrstvě.

Tabulka 8.2: Výsledky modelů pro Iris data II.

Model	Počet testů	Procentuální poškození	Procentuální úspěšnost na testovací množině	
			bez poškození	s poškozením
Základní model	1000	25%	86,5%	79,3%
Rostoucí model	1000	25%	87,5%	79,8%
Vícevrstvý model	1000	25%	87,9%	82,0%

Dle výsledků můžeme obecně říci, že modely dosahovaly velmi dobré úrovně robustnosti. Při 25% poškození sítě byl pokles úspěšnosti modelů v průměrném případě 6,9%. Podle očekávání nejlepší robustnosti dosáhl hierarchický model, který se ovšem odlišuje od ostatních modelů větším počtem neuronů. U tohoto modelu je složitější testovat robustnost. Pokud by došlo k poškození pouze u poslední vrstvy (kde se nachází nejméně neuronů ze všech vrstev), bylo by rozdílení neuronů v této vrstvě téměř náhodné, což by způsobilo velké znehodnocení výsledků. Pokud ale dojde k většímu poškození u některé jiné (mezi)vrstvy, výsledky modelu příliš ovlivněny nebudou.

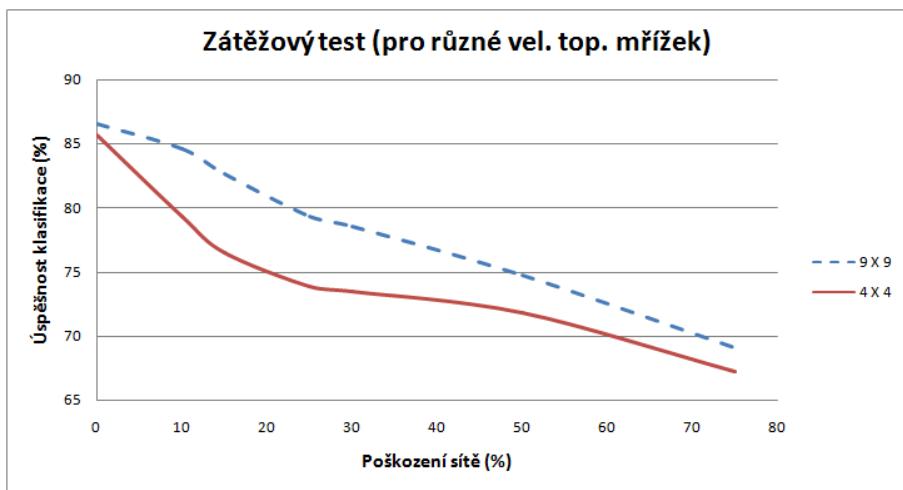
Pro lepší zhodnocení robustnosti modelů Kohonenových map jsme dále provedli záťezový test. Zde jsme zkoumali závislost mezi poškozením sítě a úspěšností klasifikace modelu. Provedli jsme několik měření při různém procentuálním poškození

sítě. Maximální hodnotu poškození jsme zvolili 75%, protože vyšší hodnota by znamenala téměř náhodné rozmístění neuronů po vstupním prostoru a toto měření by mělo velmi malou vypovídající hodnotu. Měření jsme prováděli na dvou základních modelech Kohonenových map, s různými velikostmi topologických mřížek. Výsledky jsou uvedeny v následující tabulce 8.3.

Tabulka 8.3: Výsledky zátěžového testu.

Procentuální poškození	Procentuální úspěšnost	
	topol. mřížka 9 x 9	topol. mřížka 4 x 4
0%	86,5%	85,7%
10%	84,6%	79,3%
15%	82,6%	76,5%
25%	79,3%	73,9%
30%	78,5%	73,5%
50%	74,7%	71,8%
75%	69,1%	67,2%

Z výsledků je možné pozorovat předpokládaný pokles klasifikační úspěšnosti se současným zvyšováním poškození sítě. U menší sítě dochází k prudšímu poklesu klasifikační úspěšnosti než u téměř 5x větší sítě (poměr počtu neuronů topologických mřížek) při stejném poškození. To může být zapříčiněno tím, že u větších sítí vyšší procentuální poškození ještě neznamená úplné ”roztržení shluků”. Tím rozumíme to, že Wardův algoritmus naleze stále stejný shluk, který reprezentuje správnou kategorii a neoznačí shluk, který by mohl vzniknout náhodně při poškození sítě. U sítí s větším počtem neuronů jsou shluky tvořeny větším počtem neuronů. Poškození sítě sice posune několik neuronů pryč z centra shluku (poměrově stejně jako u malé sítě), ovšem stále zde zůstává dostatečné množství neuronů pro to, aby Wardův algoritmus tento shluk správně označil. Výsledky zátěžového testu jsme znázornili graficky na obrázku 8.3.



Obrázek 8.3: Výsledky zátěžového testu pro 2 různé velikosti topologických mřížek.

8.3.3 Pima indians data

Dalším testem, na němž ověříme vlastnosti vybraných modelů, jsou tzv. **Pima indians data**. Jedná se o zdravotní údaje 768 žen z indiánského kmene Pima starších 21 let. Tato populace žije blízko Phoenixu v Arizoně v USA a byl u ní proveden test na diabetes, jehož výsledek je součástí dat. Úkolem modelů je dle zdravotních údajů rozdělit jedince do dvou kategorií, na jedince mající diabetes a zdravé jedince, bez znalosti informace o výsledcích testů na diabetes. Tyto výsledky jsou použity až při kontrole úspěšnosti modelů.

Jedná se poměrně o obtížný úkol, zabývalo a zabývá se jím mnoho výzkumníků ve zdravotnictví. Jde o úkol nalézt souvislost mezi naměřenými vstupními daty a diabetes. Dosud tato přesná souvislost není známa. Data jsou významná především ze dvou důvodů. Je to kvůli relativně četnému výskytu diabetes v tomto kmene a skutečnosti, že se jedná o uzavřenou komunitu, kde je drtivá většina manželství uzavírána mezi jedinci tohoto kmene. V získaných datech je 500 vzorků s negativním výsledkem testu pro výskyt diabetes, 268 pozitivních.

Popis atributů dat:

1. Počet těhotenství.
2. Koncentrace plazmové glukózy 2 hodiny po provedení testu tolerance na glukózu.
3. Diastolický krevní tlak (mm·Hg).
4. Tloušťka tukové řasy pod tricepsem (mm).
5. 2 hodinový sérový inzulín ($\mu\text{U}\cdot\text{ml}^{-1}$).
6. Body mass index BMI (hmotnost v kg / (výška v m) 2).

7. Diabetes rodová funkce.
8. Věk.
9. Výsledek testu na diabetes (0 nebo 1; hodnota 1 interpretována jako pozitivní test pro výskyt diabetes).

Při testování postupujeme stejně jako u testu na Iris datech (viz kapitola 8.3.3). Po naučení modelů na těchto datech provedeme shlukovou analýzu a kontrolu kategorií vzorů ve 2 určených shlucích. Testy opět opakujeme několikrát, výsledky jsou zapsány v tabulce 8.4.

Tabulka 8.4: Výsledky modelů pro Pima indians data.

Model	Počet testů	Vstupních vzorů (Trén./Test.)	Procentuální úspěšnost na test. množině		Směr. odchylka	Čas výpočtu
			Prům.	Max.		
Základní m.	1000	768 (538/230)	65,2%	74,5%	3,5%	418 min.
Rostoucí m.	1000	768 (538/230)	65,3%	75,3%	3,6%	114 min.
Vícevrstvý m.	1000	768 (538/230)	63,9%	74,9%	6,8%	237 min.

Při testování byly parametry modelů a učení nastaveny následovně:

- Základní model - velikost topologické mřížky: 6 x 6 neuronů, vigilanční koeficient: $\alpha(t) = e^{-0.005 \cdot t}$, počet adaptačních kroků: 4000.
- Rostoucí model - počáteční dimenze sítě: 3 (4 počáteční neurony), parametr $\epsilon_b = 0.15$, parametr $\epsilon_n = 0.02$, parametr $\alpha = 0.005$, počet adaptačních kroků 5000, počet adaptačních kroků pro vložení nového neuronu $\lambda = 160$.
- Vícevrstvý model - počet vrstev: 3, velikosti topologických mřížek: 6 x 6, 4 x 4, 3 x 3, vigilanční koeficienty jednotlivých vrstev: $\alpha_1(t) = 0.95 \cdot e^{-0.001 \cdot t}$, $\alpha_2(t) = 0.55 \cdot e^{-0.001 \cdot t}$, $\alpha_3(t) = 0.25 \cdot e^{-0.001 \cdot t}$, počet adaptačních kroků: 4000.

Velmi dobrých výsledků v testu na Pima indians datech dosáhl rostoucí i základní model Kohonenovy mapy, podobný byl také rozptyl jejich výsledků. Ovšem opět doba učení rostoucího modelu byla téměř 3,7 krát menší než u základního modelu. Dle těchto výsledků se nejlepším modelem u testu na Pima indians datech stal opět rostoucí model, vícevrstvý model měl v tomto testu menší klasifikační úspěšnost.

8.3.4 Zátěžový test na poškozených datech

Dalším testem ověříme klasifikační schopnosti modelů na poškozených datech. Tím rozumíme vlastnost ve smyslu 1. bodu ”robustnosti” v kapitole 8.1, tzn. že modely by měly mít co nejmenší citlivost na změny vstupních dat a poškození. Využijeme k tomu oba již použité soubory dat - Iris data (kapitola 8.3.1) i Pima indians data (kapitola 8.3.3). Testy provedeme na všech implementovaných modelech.

Poškození dat budeme simulovat tzv. zašuměním dat, kdy ke každé vstupní hodnotě přičteme či odečteme malou hodnotu. Výpočet této hodnoty se řídí normálním rozdělením se střední hodnotou rovnou 0 a rozptylem, který je roven jisté poměrné části této hodnoty. Čím vyšší poměrná část hodnoty bude vzata, tím větší zašumění vstupních dat získáme. Provedeme testy s různou mírou poškození vstupních dat, výsledky uvedeme v následujících tabulkách. Tabulka 8.5 simuluje poškození na Iris datech, tabulka 8.6 na Pima indians datech.

Tabulka 8.5: Zátěžový test na poškozených Iris datech.

Model	Počet testů	Procentuální úspěšnost na testovací množině			
		bez poškození dat	s rozptylem (5% hodnot) ²	s rozptylem (10% hodnot) ²	s rozptylem (20% hodnot) ²
Základní m.	1000	86,5%	84,3%	79,8%	68,2%
Rostoucí m.	1000	87,5%	84,7%	81,2%	70,0%
Vícevrstvý m.	1000	87,9%	84,1%	80,6%	71,8%

Tabulka 8.6: Zátěžový test na poškozených Pima indians datech.

Model	Počet testů	Procentuální úspěšnost na testovací množině			
		bez poškození dat	s rozptylem (5% <i>hodnot</i>) ²	s rozptylem (10% <i>hodnot</i>) ²	s rozptylem (20% <i>hodnot</i>) ²
Základní m.	1000	65,2%	65,1%	65,0%	63,8%
Rostoucí m.	1000	65,3%	65,2%	65,0%	64,2%
Vícevrstvý m.	1000	63,9%	63,7%	62,6%	62,4%

Poznamenejme, že poškozena byla jak data trénovací, tak testovací. V těchto zátěžových testech nejlépe dopadl rostoucí model Kohonenovy mapy. I při větším poškození dat s rozptylem náhodné veličiny normálního rozdělení (20% *hodnot*)² byla klasifikační úspěšnost tohoto modelu na Iris datech 81,2%.

Zajímavé výsledky jsme obdrželi u testů na Pima indians datech. U této obtížné klasifikační úlohy, kdy i bez poškození dat dosahují modely úspěšnosti kolem hodnot 65%, se poškození téměř neprojevilo. Může to být způsobeno tím, že modely při klasifikaci nenalezly žádné význačné atributy vstupních dat, podle kterých by jasně určily správnou kategorii (význam jednotlivých atributů pro klasifikaci je rozložen rovnoměrně). Tudíž poškození vstupních dat neovlivní tolik jejich klasifikační schopnosti jako u úloh, kde se očekává daleko větší úspěšnost a konkrétní atributy hrají významnou roli.

Především na Pima indians datech modely Kohonenových map ukázaly velmi dobrou odolnost vůči poškození vstupních dat. U všech modelů se pokles klasifikační úspěšnosti pohyboval pouze v desetinách procent.

8.4 Analýza a zhodnocení výsledků modelů

V předchozích kapitolách jsme u konkrétních testů uvedli rovnou tabulky s výsledky jednotlivých modelů Kohonenových map. Z těchto výsledků již samy o sobě plynou schopnosti modelů (např. procentuální vyčíslení klasifikačních schopností). Nyní se detailněji zaměříme na některé další vlastnosti modelů, porovnáme výsledky s předpoklady v rozboru a provedeme srovnání s ostatními modely.

Vrátíme se ještě k celkovým časům, které modely potřebovaly na provedení testů. Čas testu jsme měřili jako součet času učení, zobrazení topologické mřížky, nalezení shluků a výpočtu statistiky testu. Právě čas potřebný na zobrazení topologické mřížky hrál významnou roli především u testů s menším počtem předkládaných vstupních vzorů a adaptačních kroků. Při experimentu na Iris datedech (viz kapitola 8.3.1) nejdelsí čas potřeboval základní model. To bylo způsobeno díky dlouhé době potřebné k zobrazení topologické mřížky (tzn. výpočtu Sammonova mapování).

Přestože je učení vícevrstvého modelu složitější (jeden adaptační krok znamená adaptaci v každé vrstvě modelu), celkový čas testu tohoto modelu je menší. Tento jev je způsoben tím, že výsledné shluky hledáme v poslední vrstvě modelu, která obsahuje pouze několik málo neuronů. Proto je jejich zobrazení a hledání shluků mezi neurony časově méně náročné. Nejlepších časů dosahoval rostoucí model, který na počátku začíná s malým počtem neuronů. Postupně sice přidává do topologické mřížky další neurony, ale i přesto je rychlost učení jedna z největších výhod tohoto modelu.

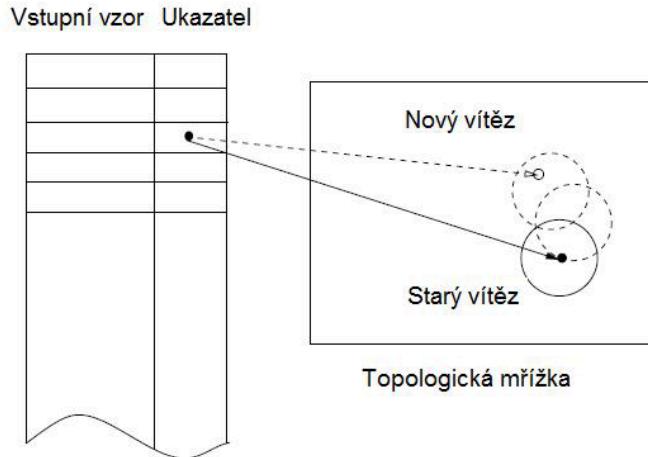
Čas potřebný pro učení Kohonenovy mapy je úměrný její velikosti. Vítězný neuron c v topologické mřížce určíme dle [29]:

$$c = \operatorname{argmin}_{i=1, \dots, m} \sqrt{\sum_{j=0}^{N-1} (x_j - w_{ji})^2}, \quad (8.3)$$

kde x_j značí hodnotu vstupu j -tého neuronu, w_{ji} označuje hodnotu váhy synapse z j -tého vstupního do i -tého výstupního neuronu. Pro jeho určení tedy musíme projít celou množinu $\{i = 1, \dots, m\}$ neuronů topologické mřížky.

Můžeme předpokládat, že v každé iteraci učení modelu Kohonenovy mapy předkládáme stejnou množinu vstupních vzorů (na pořadí předkládání nezáleží, např. v každé iteraci předkládáme vzory dle jiné permutace). Pak dle [18] existuje možnost, jak výrazně urychlit dobu potřebnou pro nalezení vítězného neuronu. Pokud předkládáme vstupní vzor po již několikáté, měl by se vítězný neuron c nacházet v blízkosti neuronu, který vyhrál v kompetici pro tento vzor v minulé iteraci. Pro nalezení vítěze tedy stačí dle [18] prohledat množinu sousedů vítězného neuronu z předchozí iterace. Pro realizaci této varianty algoritmu učení si musíme ovšem udržovat v paměti aktuální tabulku, kde ke každému vstupnímu vzoru uchováváme ukazatel na předešlého vítěze. Tuto metodu dokumentujeme obrázkem 8.4. Je jasné, že tato modifikace s sebou přináší zvýšení paměťových nároků při učení. Ovšem při současných

velkých kapacitách pamětí je tento nárůst minimální vůči zrychlení učení, kterého tímto dosáhneme.



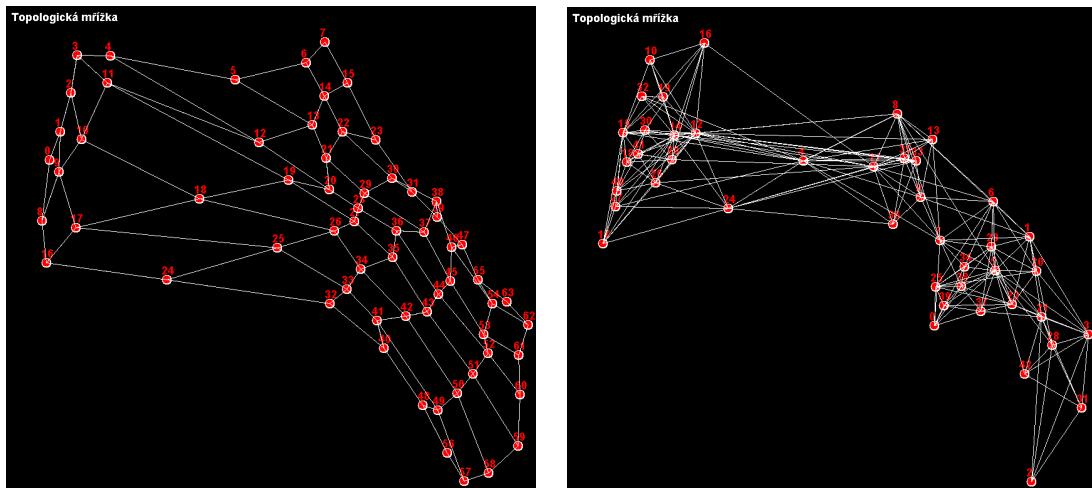
Obrázek 8.4: Schéma datové struktury pro uložení ukazatelů na vítěze z minulé iterace (původní obrázek uveden v [18] obrázek číslo 3.).

Obecně Kohonenovy samoorganizující se mapy dosahují velmi dobrých výsledků v oblastech analýzy dat. Problémem základních modelů je nutnost znát předem konkrétní počet neuronů topologické mřížky. Odhad jejich počtu je bez předchozích znalostí struktury dat obtížný, v praxi se musí odzkoušet několik variant topologie modelu, z nichž je zvolena nejlepší pro konkrétní úlohu, což ovšem znamená časové ztráty. Tento problém se snaží řešit hned několik modelů, které jsou založeny na myšlence přidávání nových neuronů do topologické mřížky v průběhu učení, a to rostoucí Kohonenovy mapy (viz 4.2) a evoluční stromy (viz 5.2). Na začátku procesu učení modely obsahují pouze malý počet neuronů a jsou zvoleny podmínky, po jejichž splnění jsou v průběhu učení přidávány nové neurony.

U deskripcních typů úloh bychom preferovali raději modely, kde lze lépe pozorovat vztahy mezi neurony. V drtivé většině případů analyzujeme neznámá data, potřebujeme zjistit jejich pravděpodobnostní rozložení a vztahy mezi daty. Těmto podmínkám vyhovuje základní model, kdy již od začátku můžeme zvolit dostatečný počet neuronů topologické mřížky a lze dobře pozorovat strukturu topologické mřížky během učení (za použití např. Sammonova mapování). Učení základního modelu je ale velmi pomalé a v testech tento model nedosahoval takových výsledků jako rostoucí model. Problémem nastane při zobrazování rostoucího modelu, jestliže zvolíme dimenzi topologické mřížky větší než 3. Model approximuje pravděpodobnostní rozložení velmi dobře, ovšem strukturu mezi neurony lze pozorovat již hůře. Topologické mřížky

těchto modelů můžeme srovnat na obrázku 8.5.

Možností, jak zlepšit zobrazování struktury rostoucího modelu a současně zachovat jeho velmi dobré vlastnosti (viz výsledky testů v kapitole 8.3), by bylo další zobrazení po naučení rostoucího modelu. Učení modelu by probíhalo standardním způsobem (viz kapitola 4.2), ale výsledná topologická mřížka by se stala vstupem další již menší základní Kohonenovy mapy. Po krátkém učení základní Kohonenovy mapy bychom tuto mapu zobrazili a využili tak její vlastnosti - dobře pozorovatelné struktury topologické mřížky. Tímto způsobem bychom zachovali jak výborné vlastnosti rostoucího modelu, tak využili dobře pozorovatelné struktury topologické mřížky základního modelu Kohonenovy mapy.



(a) Topologická mřížka základního modelu, (b) Topologická mřížka rostoucího modelu (diké lze dobře pozorovat strukturu topologické menze = 4), kde struktura mřížky není tak dobře pozorovatelná, ovšem shluky neuronů jsou daleko lépe rozeznatelné.

Obrázek 8.5: Porovnání výsledných struktur topologických mřížek Kohonenových map za použití Sammonova mapování.

Kapitola 9

Nehodovost v silniční dopravě

V této kapitole se seznámíme s reálnými daty, na jejichž rozbor a analýzu použijeme modely Kohonenových map. Bude se jednat o data o nehodovosti v silniční dopravě ve městě Ostrava. Pro naši práci využijeme všechna dostupná data, která byla evidována od roku 2003 až do roku 2008, tj. za posledních 6 let.

9.1 Úvod do problematiky nehodovosti v silniční dopravě

Silniční doprava je nejrozšířenějším a nejoblíbenějším druhem dopravy v České republice. Patří mezi nejflexibilnější typy dopravy, umožňuje spojení mezi libovolnými místy v rámci měst, v rámci České republiky, ale také spojení mezi evropskými metropolemi. Většina obyvatel České republiky vlastní řidičský průkaz a každodenně využívá prostředků silniční dopravy k přesunu do míst především pracovní činnosti a různých osobních potřeb. Z tohoto důvodu je městská silniční doprava nejhustším typem silniční dopravy, což s sebou přináší také více konfliktů a nehod v dopravě.

Právě proto se budeme zabývat nehodovostí silniční dopravy ve městě Ostrava [26]. Jedná se o 3. největší město České republiky. Ostrava představuje dle [23] významný silniční uzel a důležitou křižovatku Moravskoslezského kraje. Město Ostrava je napojeno na hustou síť silnic I. třídy, které spojují město s ostatními velkými městy tohoto kraje, ale také se sousedními regiony. Značná část z těchto silnic je proto dle [23] vícepruhových se středním dělicím pásem. Městská silniční komunikační síť v Ostravě přesahuje délku tisíce kilometrů.

Důležitým článkem silniční dopravy v Ostravě je městská hromadná doprava. Dopravní podnik Ostrava provozuje na sedm set vozidel v síti tramvajových, autobusových a trolejbusových linek. Cestující se mohou přepravovat po celém městě rozděleném do několika zón, ročně je městskou hromadnou dopravou přepraveno až 185 milionů osob.

Vývoj počtu dopravních nehod má v posledních letech vzrůstající trend. V 80. letech,

kdy docházelo k nárůstu dopravního výkonu, došlo také k odpovídajícímu nárůstu dopravních nehod. Ovšem od roku 1990 roste i nadále nehodovost při stále stejném dopravním výkonu. To má na svědomí mnoho různých faktorů, které se budeme snažit analyzovat. Nejhorším následkem dopravních nehod jsou jistě újmy na lidských životech a zranění různých rozsahů.

Druhým důležitým následkem dopravních nehod jsou hmotné škody. Nejsou to pouze škody vzniklé při průmě u nehody, ale i výdaje s tím spojené. Podle [12] se jedná především o škody vzniklé na zdraví účastníků nehody, dále i finanční vyčíslení všech procesů, vedoucích k odstranění následků nehody. Především náklady na zdravotní péči, administrativní náklady na policii, sociální výdaje, atd. Tyto náklady tvoří celý komplex tzv. socio-ekonomických nákladů nehodovosti, což znamená v důsledku i dodatečnou finanční zátěž pro státní rozpočet, a tím současně pro všechny daňové poplatníky.

9.2 Vstupní data a jejich význam

K analyzování dopravních nehod máme k dispozici data Policie České republiky od roku 2003 až do roku 2008. Jedná se o data o všech dopravních nehodách v okrese Ostrava-město. Každá nehoda je zaznamenána spolu s 250 možnými atributy a údaji o všech účastnících nehody. K analyzování využijeme 30 nejvýznamnějších atributů dat (viz další kapitola o předzpracování dat 9.3). V následujících tabulkách 9.1, 9.2 a 9.3 uvedeme důležité atributy, které budou hrát významnou roli při analýze dopravních nehod:

Tabulka 9.1: Použité atributy vstupních dat, část 1.

Atribut	Možné hodnoty
den nehody	den v měsíci
měsíc nehody	měsíc v roce
rok nehody	rok nehody
hodina nehody	hodina nehody
minuta nehody	minuta nehody
lokalita nehody	v obci mimo obec

Tabulka 9.2: Použité atributy vstupních dat, část 2.

Atribut	Možné hodnoty
druh nehody	jiný druh nehody
	srážka s jedoucím nekolejovým vozidlem
	srážka s vozidlem zaparkovaným, odstaveným
	srážka s pevnou překážkou
	srážka s chodcem
	srážka s lesní zvěří
	srážka s domácím zvířetem
	srážka s vlakem
	srážka s tramvají
	havárie
druh srážky jedoucích vozidel	čelní
	boční
	z boku
	zezadu
	nejde o srážku jedoucích vozidel
druh pevné překážky	strom
	sloup - telefonní, veřejné osvětlení, atd.
	odrazník, patník, sloupek
	svodilo
	překážka vzniklá provozem jiného vozidla
	zed', pevná část mostů, podjezdů, tunelů
	překážka vzniklá provozem jiného vozidla
	překážka vzniklá stavební činností
	jiná překážka - zábradlí, oplocení, násep
	nejde o srážku s pevnou překážkou
charakter nehody	nehoda s následky na životě nebo zdraví
	nehoda pouze s hmotnou škodou
zavinění nehody	řidičem motorového vozidla
	řidičem nemotorového vozidla
	chodcem
	lesní zvěří, domácím zvířectvem
	jiným účastníkem silničního provozu
	závadou komunikace
	technickou závadou vozidla
	jiné zařízení
alkohol u viníka nehody	přítomen
	nepřítomen
	nezjištěváno
hlavní příčiny	nepřiměřená rychlosť jízdy
	nesprávné předjízdění
	nedání přednosti v jízdě
	nesprávný způsob jízdy
	technická závada vozidla
následky nehody - stav do 24 hodin	usmrcenou osobou
	těžce zraněnou osobou
	lehce zraněnou osobou
celková hmotná škoda	ve stokorunách

Tabulka 9.3: Použité atributy vstupních dat, část 3.

Atribut	Možné hodnoty
druh povrchu vozovky	dlažba
	živice
	beton
	panely
	štěrk
	jiný
stav povrchu vozovky v době nehody	povrch suchý, neznečištěný
	povrch suchý, znečištěný (písek, listí)
	povrch mokrý
	na vozovce je bláto
	na vozovce je náledí, posypáno
	na vozovce je náledí, neposypáno
	na vozovce je rozlitý olej
	souvislá sněhová vrstva
stav komunikace	dobrý, bez závad
	podélňý sklon vyšší než 8%
	nesprávně umístěná dopravní značka
	zvlněný povrch v podélném směru
	souvislé výtluky
	nesouvislé výtluky
	trvalé zúžení vozovky
	propadlé kolejky
	přechodná uzavírka jednoho jízdního pruhu
povětrnostní podmínky	neztížené
	mlha
	na počátku deště, slabý déšť
	déšť
	sněžení
	tvoří se námraza, náledí
	nárazový vítr

9.3 Předzpracování vstupních dat

Dříve než začneme data analyzovat jednotlivými modely Kohonenových map, musíme data vhodně předzpracovat. Vstupní data obsahují také atributy, které jsou velmi řídce obsazeny. Jedná se o informace o dalších účastnících dopravních nehod, jako je např. druhý nebo třetí automobil či chodec, který se účastnil dopravní nehody. Tyto atributy k analýze dopravních nehod využívat nebudem, protože se vyskytují jen u velmi malého procenta dopravních nehod a nemají téměř žádný vliv na charakter nehody, jedná se často pouze o přímé svědky dopravní nehody.

Dalšími atributy, které odstraníme z dat při předzpracování, jsou atributy s konstantní hodnotou. Jsou to především informace o kraji a okresu místa nehody a příslušném útvaru Policie České republiky, které jsou stále stejné. Tyto atributy by zbytečně zvyšovaly dimenzionalitu vstupních vzorů, což by způsobilo narůst nároků na počet trénovacích vzorů a prodloužení doby učení.

Kapitola 10

Využití testovaných modelů při zpracování reálných dat o nehodovosti v dopravě

V této kapitole se dostáváme k reálnému použití modelů Kohonenových map. Bude se jednat o typ deskripcní úlohy, kdy se budeme snažit nalézt dominantní struktury a vazby, které jsou skryté v daných datech. Motivace tohoto úkolu vyplývá již z typu a formátu vstupních dat. Chceme analyzovat dopravní nehody takovým způsobem, abychom získali charakteristiku několika hlavních typů dopravních nehod. Po naučení modelů Kohonenových map budeme analyzovat shluky neuronů v topologických mřížkách a jím odpovídající vstupní vzory. Z výsledků budeme požadovat zisk mnohem většího množství informací, než bychom obdrželi pouhým statistickým zpracováním.

Statistickými metodami použitelnými na tato data bychom jistě získali velmi cenné informace, ovšem pouhé procentuální vyčíslení hodnot dané veličiny samo o sobě moc informací nepřináší. To můžeme uvést na příkladu, kolik nehod se stalo v denních a nočních hodinách. Procentuální hodnota této veličiny nám přináší určitou informaci o nehodovosti v denní době, ale k analýze a případným opatřením na snížení nehodovosti moc nepřispěje. Pokud ale tuto informaci připojíme k informaci o místu nehody, počasí, zavinění nehody, alkoholu a dalším atributům, získáme mnohem ucelenější představy o typech nehodovosti a možnostech různých protiopatření.

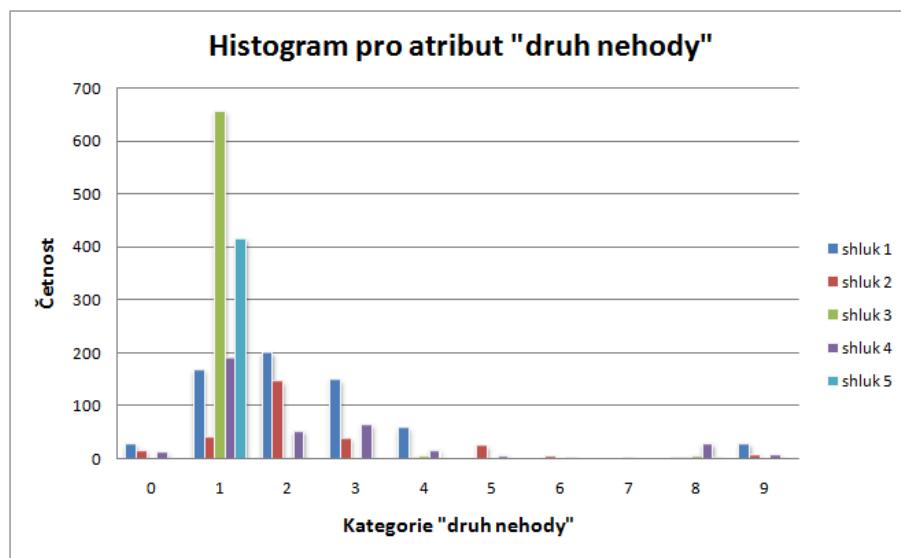
10.1 Realizace jednotlivých modelů

Vstupním datům a jejich předzpracování jsme se již věnovali v předchozí kapitole (viz kapitola 9.3). Modelům Kohonenových map budeme předkládat 2 různé soubory těchto dat, a to menší soubor obsahující data za jeden rok (2008) a větší soubor dat obsahující veškerá dostupná data o nehodovosti za posledních 6 let (2003 - 2008). Využijeme k tomu jak základní model Kohonenovy mapy, kde v pravidelné mřížce můžeme velmi dobře pozorovat vzájemné vztahy mezi neurony, tak rostoucí

Kohonenovu mapu. Protože se jedná o velmi rozsáhlá data, můžeme předpokládat, že doba učení základního modelu bude několika násobně vyšší než u rostoucího modelu. Právě u úloh s rozsáhlými vstupními daty je doba učení rostoucích Kohonenových map velkou předností.

Bude se jednat o deskripční úlohy, kdy budeme chtít zobrazit výslednou topologickou mřížku Kohonenovy mapy. K tomuto účelu použijeme Sammonovo mapování (viz kapitola 6.1.1). Po skončení procesu učení a zobrazení výsledného tvaru topologické mřížky budeme pomocí Wardova algoritmu (kapitola 6.2) hledat význačné shluky neuronů v topologické mřížce.

Po nalezení shluků provedeme rozdělení dat do odpovídajícího počtu shluků (počet souborů, tj. počet kategorií, závisí na konkrétní úloze a bude analyzován až přímo u dané úlohy). Každý výsledný shluk obsahuje vstupní vzory patřící do stejného shluku, které by měly mít i podobné vlastnosti. Pro každý atribut vstupních dat zhotovíme histogram (příklad histogramu na obrázku 10.1) konkrétního shluku a výsledky budeme porovnávat a analyzovat.

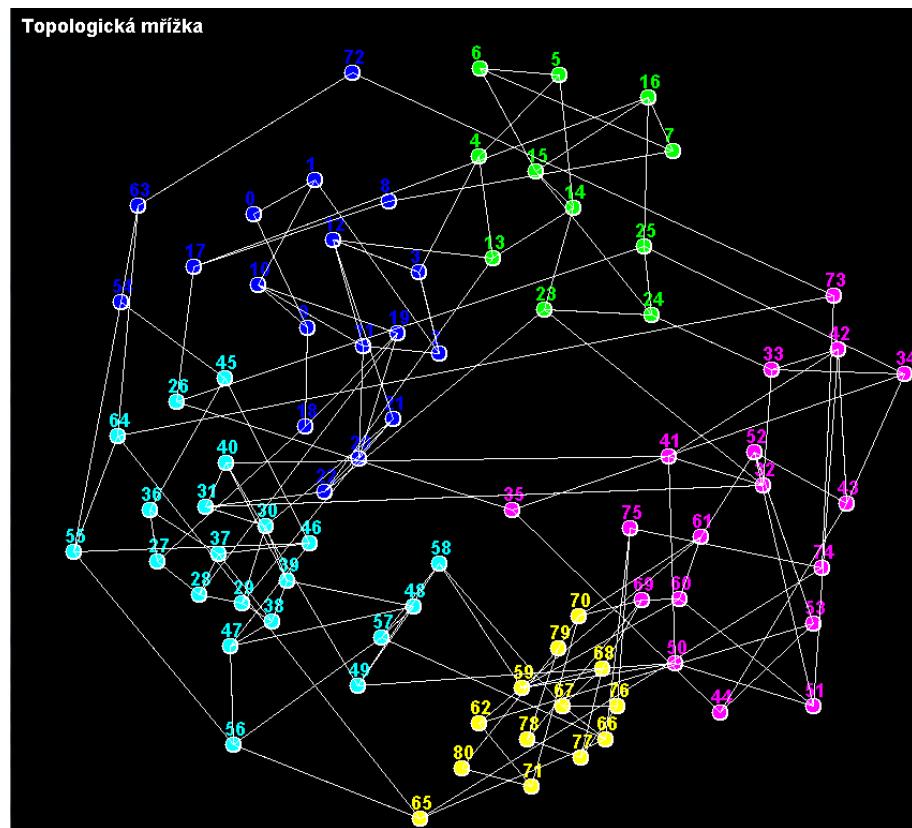


Obrázek 10.1: Histogram pro atribut "druh nehody".

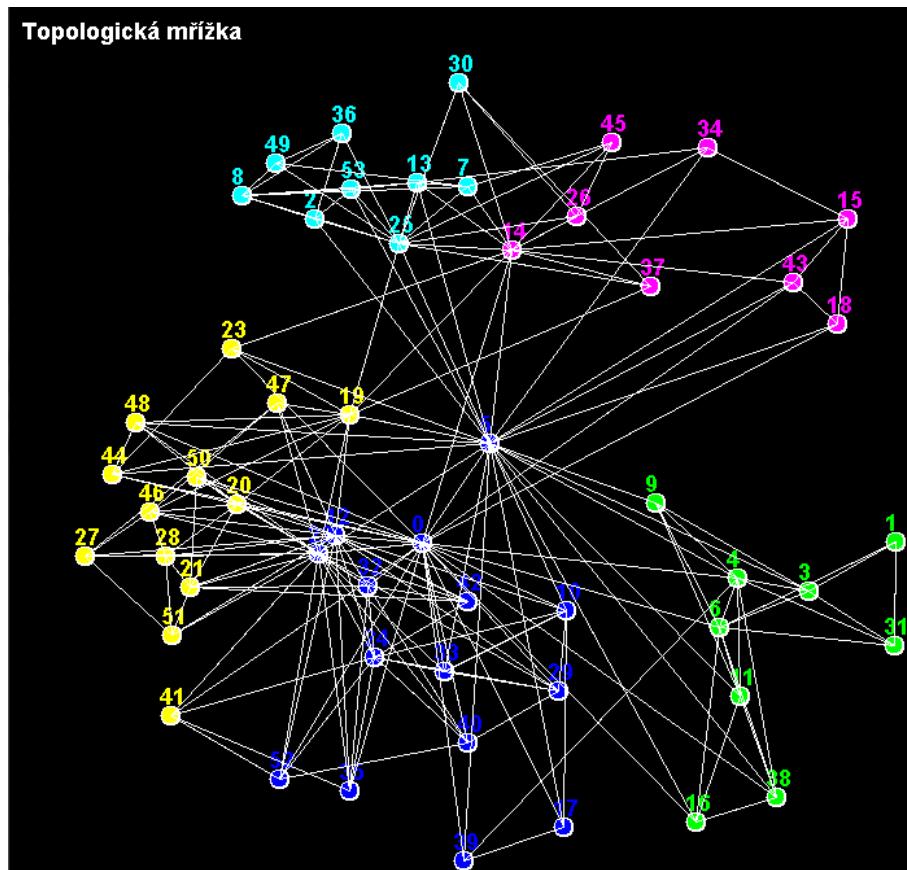
10.2 Aplikace na datech nehodovosti za rok (2008)

Jak jsme se zmínili v úvodu kapitoly, snažíme se nalézt dominantní struktury a vazby, které jsou skryté v daných datech. Dle základního rozboru charakteristik dopravních nehod (viz kapitola 9) budeme požadovat nalezení několika málo druhů

(shluků) dopravních nehod. Nemá význam hledat větší (více než 10) počet shluků, protože velký počet typů dopravních nehod by již neměl takovou vypovídající hodnotu o charakteru nehod (některé shluky by zastupovaly velmi malé procento dat, rozdělením velkého shluku na dva bychom ztratili ucelenou informaci o typu nehody). Dle několikanásobného pozorování výsledných tvarů topologických mřížek (viz obrázky 10.2 a 10.3) a předchozího rozboru jsme experimentálně zvolili počet shluků roven 5. Poznamenejme, že barevně odlišené shluky v obrázcích jsou výsledkem Wardova algoritmu pro nalezení shluků a konkrétní barva nemusí označovat stejný shluk u základního modelu a rostoucí Kohonenovy mapy.



Obrázek 10.2: Topologická mřížka základního modelu Kohonenovy mapy pro data za rok 2008.



Obrázek 10.3: Topologická mřížka rostoucí Kohonenovy mapy pro data za rok 2008.

Analyzovali jsme histogramy jednotlivých shluků neuronů v topologických mřížkách obou modelů. Typy nehod (charakterizovány hodnotami atributů ve shlucích) odpovídající shlukům byly identické u obou modelů. Atributy odpovídající typům nehod se u obou modelů lišily maximálně v jednotkách procent v jednotlivých attributech, tudíž je lze bezpečně rozpoznat. Počty nehod v odpovídajících typech nehod se lišily maximálně také v jednotkách procent mezi základním a rostoucím modelem Kohonenovy mapy. Proto můžeme výsledné typy dopravních nehod za rok 2008 charakterizovat jednou tabulkou následovně (procentuální hodnoty jsou použity ze shluků základního modelu):

- Stručný přehled hlavních charakteristik nehod 1. typu:

Atribut	Hodnota
procentuální zastoupení v nehodách	26,8%
zavinění nehody	srážka se zaparkovaným vozidlem, s pevnou překážkou, s chodcem
újmy na zdraví	těžká zranění
hmotné škody	více menších škod
příčina	nesprávné otáčení nebo couvání

Tento typ nehod zastupuje relativně velkou část dopravních nehod. Můžeme ho charakterizovat jako nehody vzniklé při nesprávném otáčení nebo couvání, kdy došlo k nárazu do zaparkovaného vozidla, pevné překážky či chodce. Druh pevné překážky nehrál žádnou roli v příčinách nehody. Překvapivě při tomto relativně "bezpečném pohybu na komunikaci" docházelo velmi často k újmám na životě nebo zdraví. Zavinění nehody je především na straně řidiče vozidla, ovšem nezanedbatelnou roli při zavinění dopravní nehody hrají chodci. Při tomto typu nehod se alkohol u řidičů téměř nevyskytoval, v několika případech byla dopravní nehoda ovlivněna i okolními podmínkami stavu vozovky (výtluky, bláto). Jistý podíl na závažnosti zranění osob sehrála kombinace druhu povrchu vozovky (dlažba) s povětrnostními podmínkami (náledí).

- Stručný přehled hlavních charakteristik nehod 2. typu:

Atribut	Hodnota
procentuální zastoupení v nehodách	28,1%
zavinění nehody	srážka s jedoucím vozidlem
újmy na zdraví	lehká zranění
hmotné škody	v celém rozsahu finančních škod
příčina	nedodržení bezpečné vzdálenosti

Dalším významným typem dopravních nehod jsou srážky s jedoucím vozidlem, v tomto případě téměř výhradně srážky ze zadu. Příčinou je v drtivé většině nedodržení bezpečné vzdálenosti mezi vozidly, zavinění dopravní nehody je na straně řidiče vozidla. Opět přítomnost alkoholu v krvi řidiče nebyla prokázána. Nehody tohoto typu se stávaly především ve 2 časových obdobích, a to v ranní dopravní špičce kolem 8. hodiny a odpolední špičce po 15. hodině.

Oproti očekávání při srážkách jedoucích vozidel docházelo jen výjimečně ke zraněním osob, případná zranění byla lehkého charakteru. Tento fakt přisuzujeme hlavně skutečnosti, že se jedná o dopravní nehodovost ve městě (obci), tudíž maximální povolená rychlosť je omezena na 50 km/h. Hmotné škody se rozprostíraly v celém spektru finančních škod. Povětrnostní podmínky nehrály důležitou roli v tomto typu nehod, povrch vozovky byl většinou suchý, v několika případech se nehody staly po dešti, příp. sněžení.

- Stručný přehled hlavních charakteristik nehod 3. typu:

Atribut	Hodnota
procentuální zastoupení v nehodách	15,9%
zavinění nehody	srážka s tramvají, chodcem, pevnou překážkou
újmy na zdraví	smrtelná zranění, těžká zranění
hmotné škody	menší škody
příčina	nebyla analyzována hlavní příčina

Následující typy dopravních nehod mají podobné procentuální zastoupení, ovšem tento typ musíme uvést přednostně, a to především díky počtu smrtelných a velmi těžkých zranení. Nehody tohoto typu se staly výhradně v nočních hodinách. Místo dopravních nehod bylo většinou na neznačené komunikaci, kde přednost vozidel vyplývá z dopravní vyhlášky. Zavinění dopravní nehody není jasné dánno, své podíly mají jak především chodci, tak řidiči vozidel a tramvají. Alkohol u těchto nehod nebyl zjištován, ale v několika případech byl u chodců pozitivní. Většinou se jedná o čelní srážky chodců s tramvají či vozidlem, kde tyto střety, bohužel, končí smrtelným zraněním.

- Stručný přehled hlavních charakteristik nehod 4. typu:

Atribut	Hodnota
procentuální zastoupení v nehodách	17,5%
zavinění nehody	srážka s jedoucím vozidlem
újmy na zdraví	bez zranění, lehká zranění
hmotné škody	v celém rozsahu finančních škod
příčina	porušení dopravního příkazu "Dej přednost v jízdě", "Stůj, dej přednost v jízdě"

Podle příčiny tohoto druhu dopravních nehod můžeme rovnou analyzovat typ dopravní nehody. Tyto nehody se stávají především v dobách dopravních špiček, opět kolem 8. hodiny dopoledne a po 15. hodině odpoledne. Zavinění je na straně řidiče vozidla, kdy dochází k porušení příkazu dopravní značky "Dej přednost v jízdě" a "Stůj, dej přednost v jízdě". Jedná o typ srážky jedoucích vozidel, kde dochází ke střetům ze všech stran vozidel. Z toho vyplývající hmotné škody jsou různého rozsahu, jak fyzického poškození, tak finančního dopadu.

- Stručný přehled hlavních charakteristik nehod 5. typu:

Atribut	Hodnota
procentuální zastoupení v nehodách	11,7%
zavinění nehody	srážka se zaparkovaným či odstaveným vozidlem, s lesní zvěří
újmy na zdraví	lehká zranění
hmotné škody	poměrně velké škody
příčina	vyhýbání se bez dostatečné vůle, nezaviněno řidičem

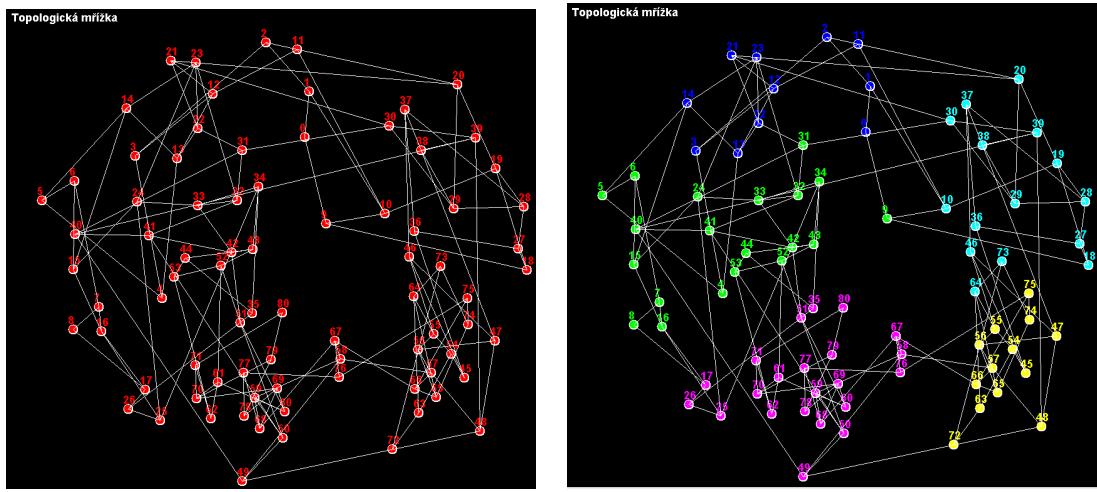
Posledním typem dopravní nehod jsou nehody, kde poměr zavinění řidičem vozidla a jiným vlivem je téměř vyrovnaný. Překvapivým typem nehod ve městě jsou střety s lesní zvěří. Město Ostrava má největší poměr plochy zeleně na obyvatele, a to 30 m^2 na obyvatele. Ve městě se nachází část chráněné krajinné oblasti, vyskytuje se zde louky s mokřady a velký počet parků. Proto dopravní nehody s lesní zvěří především v nočních hodinách nejsou neobvyklé. Do této kategorie patří také nehody zapříčiněné nedostatečným vyhýbáním se překážce. Tento fakt může být také spojen s výskytem lesní zvěře, kdy se řidič vozidla snaží zvěři vyhnout, ale následuje náraz do zaparkovaného či odstaveného vozidla. Při střetu jak přímo se zvěří nebo odstaveným vozidlem vznikají poměrně velké hmotné škody, zranění osob jsou spíše lehčího charakteru.

10.3 Aplikace na větším souboru dat

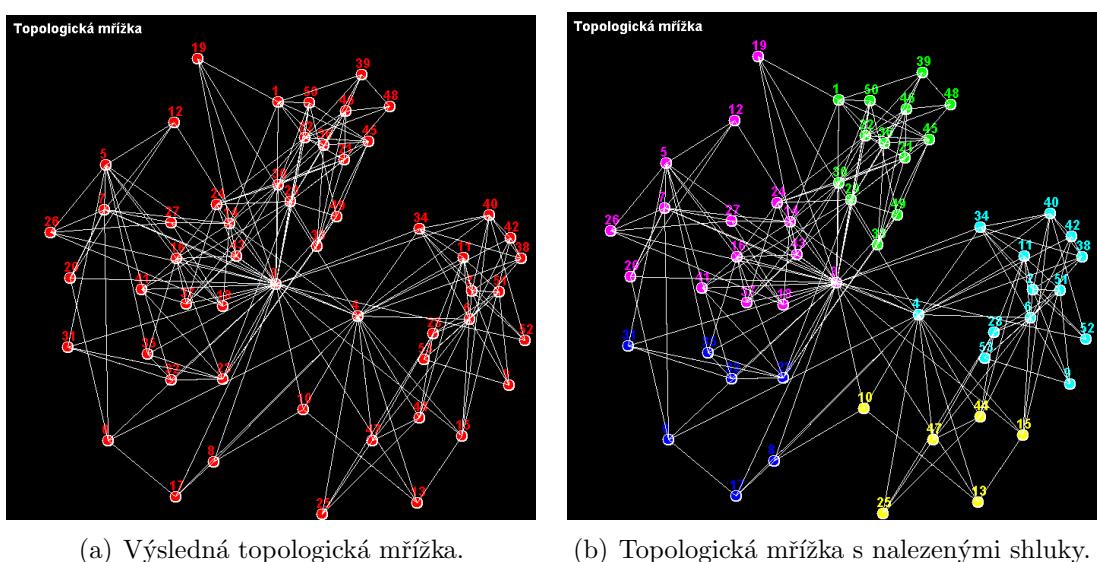
Nyní budeme analyzovat celý soubor dat, tj. data o nehodovosti za posledních 6 let (2003 - 2008). Soubor dat obsahuje údaje o 31700 nehod s atributy uvedenými v kapitole 9.2. Budeme postupovat stejným způsobem jako při analýze souboru dat za rok 2008 (viz předchozí kapitola).

Získané typy dopravních nehod byly opět u základního modelu (viz obrázek 10.4) i rostoucí Kohonenovy mapy (viz obrázek 10.5) velmi podobné (maximální rozdíly počtů nehod mezi jednotlivými typy nehod u obou modelů byly opět v jednotkách procent). Podstatný rozdíl byl ovšem v době učení, která byla u rostoucí Kohonenovy mapy téměř 6x kratší.

Po naučení základního modelu Kohonenovy mapy i rostoucí Kohonenovy mapy a nalezení shluků dostáváme následující topologické mřížky:



Obrázek 10.4: Výsledné topologické mřížky základního modelu Kohonenovy mapy aplikované na celém souboru dat.



Obrázek 10.5: Výsledné topologické mřížky rostoucí Kohonenovy mapy aplikované na celém souboru dat.

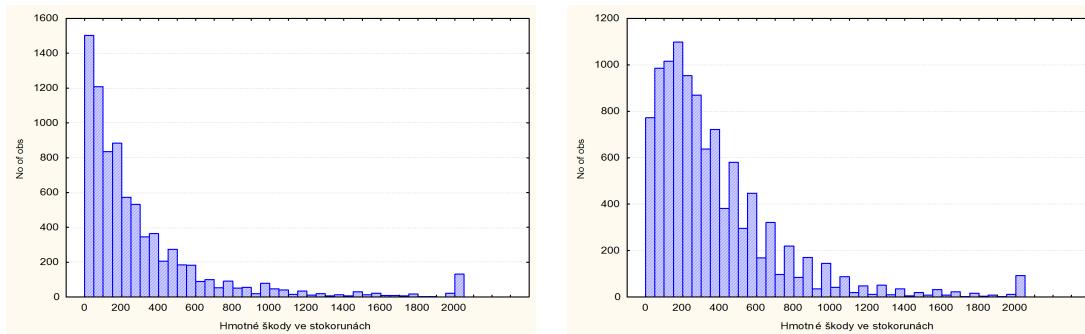
Výsledky z celého souboru dat můžeme vzít jako zobecnění jak nehodovosti v roce 2008, tak celé nehodovosti ve městě Ostravě. Získané výsledky jsou typově velmi podobné výsledkům z roku 2008. Stručně proto popíšeme hlavní charakteristiky.

1. Nehody způsobené srážkou jedoucích vozidel. Ke srážkám dochází výhradně ze zadu, příčinou je nedodržení bezpečné vzdálenosti. Viníkem nehody je řidič

vozidla, nehody nejsou ovlivněny řízením dopravního provozu, stávají se uprostřed některého z více jízdních pruhů. Ke zvýšení počtu nehod tohoto typu přispívá mlha, déšť a sněžení. Zranění jsou lehčího charakteru, zato hmotné škody jsou vyšší.

2. Srážky se zaparkovaným či odstaveným vozidlem a srážky s pevnou překážkou. Překážka nemá žádný vliv na charakter nehody, často to jsou zábradlí, oplocení atd. Viníkem je opět řidič vozidla. Hlavní příčinou je nesprávné otáčení nebo couvání. Část nehod se stává mimo komunikaci a na parkovištích. Větší vliv na četnost těchto nehod má kombinace náledí, sněžení a možné závady či opravy komunikace.
3. Nehody vzniklé srážkou jedoucích vozidel, typ srážky je ze všech stran. Nehody jsou způsobené řidičem jedoucího vozidla, který v drtivé většině poruší příkaz "Dej přednost v jízdě" nebo nedá přednost vozidlu přijíždějícímu zprava. Místo nehod bývá opatřeno dopravními značkami, které vyznačují přednost. Zranění jsou opět výjimečná, spíše lehká, ale hmotné škody při střetu dvou jedoucích vozidel jsou velké.
4. Samostatnou kategorii tvoří srážky s chodcem, pevnou překážkou nebo lesní zvěří. Pouze v polovině případů je viníkem řidič vozidla. Bohužel typickým znakem těchto nehod jsou smrtelná, popř. těžká zranění. Část nehod se stane mimo komunikaci. Alkohol u tohoto typu nehod nebývá vůbec zjištován.
5. Posledním typem jsou nehody způsobené srážkou jedoucích vozidel, příp. tramvají a střety s pevnou překážkou. Pokud se jedná o střet jedoucích vozidel, náraz bývá boční. Viníkem nehody je řidič jedoucího vozidla, zranění bývají těžká, v několika případech smrtelná. Přednost v jízdě bývá vyznačena dopravními značkami, ke střetům dochází po porušení příkazu "Dej přednost v jízdě".

Zajímavým aspektem dopravních nehod jsou hmotné škody. Tyto ukazatele jsou významné jak pro pojišťovny, tak pro samotnou analýzu dopravních nehod. Při zkoumání hmotných škod v jednotlivých shlucích jsme zjistili, že jejich rozložení jsou přibližně stejného charakteru. Nejčetnější byly nehody, při nichž vznikla hmotná škoda ve výši 20000 Kč. Počet nehod se stoupající hmotnou škodou klesá. Vyšší počet nehod s hmotnou škodou okolo 200000 Kč může být způsoben nehodami nových či luxusních automobilů, kde i každá relativně malá nehoda může způsobit na těchto automobilech velké finanční škody. Příklad rozložení hmotných škod je na obrázku 10.6.



(a) Histogram hmotných škod ve shluku číslo 1. (b) Histogram hmotných škod ve shluku číslo 3.

Obrázek 10.6: Histogramy hmotných škod.

Při charakterizování nehod nehrál žádnou roli druh povrchu vozovky, ve velké většině případů to byla živice¹. Suchý nebo mokrý stav vozovky neměl při analyzování význam, ovšem náledí a námraza už vliv měly. Přítomnost alkoholu v krvi řidiče zvyšuje výrazně nehodovost, ale přímý vliv na charakter nehod neměl. Jeho přítomnost byla rovnoměrně rozložena mezi všemi druhy nehod.

¹Dle [28] je živice neboli bitumen souhrnné označení pro organické kapaliny, které jsou vysoce viskózní, černé barvy a zcela rozpustné v sirouhlíku. Asfalt a dehet jsou nejčastější formy živic.

Kapitola 11

Zhodnocení a závěr

11.1 Závěr

Podrobně jsme analyzovali a popsali několik různých variant modelu Kohonenových map (kapitola 3). Jejich nejčastější použití je v rozpoznávání (např. segmentaci obrázků, písma), své uplatnění mají také v ekonomii a dále při rozpoznávání mluvené řeči. Studovali jsme jak různé možnosti učení v kapitole 4, tak i složitější struktury modelů Kohonenových map v kapitole 5.

Nejdříve jsme se zabývali předpokládaným chováním modelů a rozborem vlastností (kapitola 8.2). Dle provedené analýzy a získaných výsledků z testů na referenčních úlohách jsme modely navzájem porovnávali. Pro úlohy, kde je cílem analyzovat neznámá data a hledat v nich skryté vzájemné vazby, bychom měli zvolit základní model. Pravidelná struktura jeho topologické mřížky umožňuje nejlépe sledovat změny poloh neuronů v průběhu učení. Naopak pro klasifikační úlohy, kde je důležitá především rychlosť učení, bychom měli upřednostnit model rostoucí Kohonenovy mapy. Mezi jeho významné přednosti patří právě rychlosť učení, která je způsobena především malým počtem neuronů topologické mřížky na počátku učení. Tyto závěry jsou patrné z naměřených hodnot, které lze nalézt v kapitole 8.4.

Velmi dobrou robustnost, tj. odolnost vůči poškození sítě, vykazoval vícevrstvý model (viz výsledky testů v kapitole 8.3). Při 25% poškození sítě byl poměr úspěšnosti poškozené sítě a úspěšnosti nepoškozené 72/100 (při poškození byla úspěšnost 82,0%, bez poškození 87,9%). Dále jsme provedli zátěžový test základního modelu, kde jsme zkoumali procentuální úspěšnost klasifikace v závislosti na poškození sítě. Výsledky testů potvrdily naši hypotézu, že u menší topologické mřížky bude pokles úspěšnosti větší než u téměř 5x větší topologické mřížky. Výsledky jsou graficky znázorněny na obrázku 8.3 v kapitole 8.3.2. Další vlastnosti modelů jsou analyzovány v kapitole 8.4.

Pro přehledné zobrazení výsledků jsme v kapitole 6.1 zkoumali možnosti zobrazení topologických mřížek modelů. Z tohoto důvodu jsme se rozhodli v naší práci používat Sammonovo mapování. Především pro jeho vlastnost, kterou je nezávislost na dimenzi zobrazované topologické mřížky. Takto můžeme zobrazovat libovolně dimenzionální topologickou mřížku, což by nebylo za použití U-matice realizovatelné. Srov-

nání metod je uvedeno v podkapitole 6.1.

Protože cílem práce bylo jak hledání skrytých vazeb a dominantních struktur v neznámých datech, tak také klastrování dat do předem známého počtu kategorií, museli jsme aplikovat metodu shlukové analýzy po procesu zobrazení topologické mřížky. Tímto problémem jsme se zabývali v kapitole 6.2, kde jsme pro náš úkol vybrali jako nejvhodnější Wardovu metodu. Tato metoda má obecně tendenci tvořit malé shluky. Zkoumali jsme i další metody shlukové analýzy, v kapitole 6.2 jsme uvedli výhody jednotlivých metod a možnosti jejich uplatnění.

Reálné použití modelů jsme testovali na datech o nehodovosti ve městě Ostravě. Rozsáhlá data s velkým počtem atributů (viz kapitola 9) jsme nejprve vhodně předzpracovali (kapitola 9.3). Odstranili jsme atributy, které nepřinášely žádnou informační hodnotu pro analýzu dopravních nehod, jako např. informace o útvaru Police České republiky, který nehodu vysetřoval.

Po naučení modelů Kohonenových map na těchto datech jsme Wardovou metodou nalezli významné shluky neuronů v topologických mřížkách (viz kapitoly 10.2 a 10.3). Poté jsme zkonztruovali histogramy význačných atributů všech pěti nalezených shluků. Dle těchto výsledků jsme analyzovali a sepsali vlastnosti těchto typů nehod.

Překvapivě se nejzávažnější nehody stávaly při srážkách jedoucího automobilu se zaparkovaným vozidlem či pevnou překážkou. Příčinou bylo v drtivé většině případů nesprávné otáčení nebo couvání. Naopak při srážkách jedoucích vozidel nedocházelo k újmám na zdraví, ale pouze hmotným škodám. Ty se rozprostíraly v celém spektru finančních škod. Vysvětlením může být fakt, že jde o analýzu nehodovosti ve městě (obci), kde je rychlosť automobilů omezena na 50 km/h. Tudíž střet jedoucích vozidel nemá tak drtité dopady.

Další typ nehod, jehož procentuální zastoupení bylo téměř 16%, je význačný hlavně počtem velmi těžkých zranění, bohužel i smrtelných. Tyto nehody se stávaly výhradně v nočních hodinách. Zavinění dopravních nehod není přesně dánno, svůj podíl mají především chodci, tak řidiči vozidel i řidiči tramvají. V několika případech byl u chodců pozitivní test na alkohol.

Výsledky jsme analyzovali také na základě informací a podkladů o nehodovosti v silniční dopravě získané jak z Magistrátu města Ostravy [23], tak z výroční zprávy společnosti Ostravské komunikace, a. s. [26]. Naše výsledky jsou podrobně sepsány v kapitole 10.2 a 10.3, kde jsme v několika tabulkách uvedli jak procentuální vyčíslení atributů (příčin, vlivů, různých okolností dopravních nehod atd.) v jednotlivých kategoriích, tak i slovní rozbor jednotlivých typů dopravních nehod.

11.2 Další směry a možnosti vývoje

Provedli jsme analýzu a testování několika modelů Kohonenových map, ale existuje jistě celá řada dalších modelů, které jsou různě úspěšné. Nastudovali jsme jak

základní efektivní modely, tak nově používané varianty modelů Kohonenových map. Byl to jak základní model, tak zástupce rostoucích modelů, a to Fritzkeho rostoucí model. Ze složitějších modelů to byla vícevrstvá Kohonenova mapa a zástupce nového druhu samoorganizujících se map - evoluční stromy. Každý model má své výhody, kvůli kterým byl publikován a je dnes využíván. Možným směrem vývoje je kombinace jejich předností, kterými jsme se zabývali v analýze. Nový model by mohl kombinovat vlastnosti jako je rychlosť učení rostoucích Kohonenových map nebo robustnost vícevrstvého modelu.

Problém při učení modelů je všeobecně nastavení vhodných parametrů učení. Možným směrem vývoje je vytvoření modelu, jenž by své parametry upravoval v průběhu učení. Podle dílčích výsledků při procesu učení by mohl model dynamicky měnit své parametry. Jedná se o podobný princip, jako rostoucí model hledá vhodné místo pro vložení nového neuronu.

Modely Kohonenových map dosahují velmi dobrých výsledků při řešení různých úloh. Výsledky jsou mnohdy lepší než často používané metody. Problém ovšem je, že je potřeba určit vhodný model pro danou úlohu, což znamená jisté časové ztráty. Proto možným cílem další práce by mohlo být zevšeobecnění modelu, jenž by mohl být použit na řešení různých typů úloh. Obdobným způsobem se v současné době testuje použití Kohonenových map pro předpověď poruchy leteckým motorů přímo za letu viz [4]. Vhodnou modifikací a vylepšením modelů Kohonenových map lze tedy řešit mnoho složitých úloh.

Literatura

- [1] Akgüngör A. P., Dogan E.: *An artificial intelligent approach to traffic accident estimation: Model development and application*, Transport 24(2), str. 135-142, 2009.
- [2] Beale R. and Jackson T.: *Neural computing: An introduction*, IOP Publishing, Bristol and Philadelphia, 1990.
- [3] Bezdek J. C.: *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, New York, 1981.
- [4] Cottrell M., Gaubert P., Eloy C., François D., Hallaux G., Lacaille J., Verleysen M.: *Fault prediction in aircraft engines using Self- Organizing Maps*, St. Augustine, USA, str. 37-44, 2009.
- [5] DeSieno D.: *Adding a conscience to competitive learning*, Proceedings of IEEE the Second International Conference on Neural Networks, str. 117-124, 1988.
- [6] Fritzke B.: *A Growing Neural Gas Network Learns Topologies*, G. Tesauro et al. (eds): *Advances in Neural Information Processing Systems 7*, str. 625-632, MIT PRESS, 1995.
- [7] Fritzke B.: *Growing cell structures - A self-organizing network for unsupervised and supervised learning*, Neural Networks, Vol. 7, No. 9, str. 1441-1460, 1994.
- [8] Han J., Kamber M.: *Data Mining: Concepts and Techniques*, Second Edition, Elsevier Inc., 2006.
- [9] Hecht-Nielsen R.: *Counterpropagation networks*, Applied Optics, Vol. 26, str. 4979-4984, 1987.

- [10] Inswandy K. and Konig A.: *Improvement of non-linear mapping computation for dimensionality reduction in data visualization and classification*, Proceedings of the Fourth International Conference on Hybrid Intelligent, str. 260-265, 2004.
- [11] Kaski S., Honkela T., Lagus K., Kohonen T.: *WEBSOM - Self-organizing maps of document collections*, Neurocomputing 21, str. 101-107, Elsevier, 1998.
- [12] Knotek M.: *Nehodovost na silnicích*,
URL <http://referaty.portik.cz/2008/04/29/nehodovost/>, 2006.
- [13] Koh J., Suk M. and Bhandarkar S. M.: *A multilayer self-organizing feature map for range image segmentation*, Neural Networks, Vol. 8, No. 1, str. 67-86, 1995.
- [14] Kohonen T.: *Self-Organizing Maps*, 2nd extended ed., Springer, Berlin, 1997.
- [15] Kohonen T.: *Self-organization and associative memory*, Springer- Verlag, Berlin, 1984.
- [16] Kohonen T.: *Self-organized formation of topologically correct feature maps*, Biological Cybernetics, Vol. 43, str. 59–69, 1982.
- [17] Kohonen T.: *Speedy SOM*, Report A33, Laboratory of Computer and Information Science, Helsinki University of Technology, 1996.
- [18] Kohonen T., Kaski S., Lagus K., Salojarvi J., Honkela J., Paatero V., Saarela A.: *Self organization of a massive documet collection*, IEEE transactions on neural networks, vol. 11, no. 3, 2000.
- [19] Koikkalainen P., Oja E.: *Self-Organizing hierarchical feature maps*, Proceedings of 1990 International Joint Conference on Neural Networks, vol. II, str. 279-284, San Diego, 1990.
- [20] Kukal J.: *Robustnost*, Automatizace, roč. 51, č. 9, str. 570-573, 2008.
- [21] Lagus K.: *Text Mining with the WEBSOM*, Acta Polytechnica Scandinavica, Mathematics and Computing Series no. 110, Espoo, 2000.

- [22] Lagus K., Honkela T., Kaski S., Kohonen T.: *WEBSOM for textual data mining*, Artificial Intelligence Review, Vol. 13, str. 345-364, Kluwer Academic Publishers, 1999.
- [23] Magistrát města Ostravy: *Informace o silniční dopravě*, URL <http://www.ostrava.cz/jahia/Jahia/lang/cs/ostrava/turista/doprava/silnicni-doprava>, 2008.
- [24] Meloun M., Militký J.: *Přednosti analýzy shluků ve vícerozměrné statistické analýze*, Sborník přednášek z konference Zajištění kvality analytických výsledků, str. 29-46, Medlov, 2004.
- [25] Moody J. and Darken C.: *Learning with localized receptive fields*, San Mateo: Morgan Kaufmann, 1989.
- [26] Ostravské komunikace, a.s.: *Nehodovost – evidence a rozbor dopravní nehodovosti na pozemních komunikacích v Ostravě v roce 2008*, Ostravské komunikace, a.s., 2009.
- [27] Pakkanen J., Iivarinen J., Oja E.: *The Evolving Tree - A Novel Self-Organizing Network for Data Analysis*, str. 199-211, Neural Processing Letters 20, 2004.
- [28] Petránek J.: *Malá encyklopédie geologie*, JIH, České Budějovice, 1993.
- [29] Rojas R.: *Neural networks - a systematic introduction*, Springer-Verlag, Berlin, New-York, 1996.
- [30] Rosenblatt F.: *The perceptron: A probabilistic model for information storage and organization in brain*, Psychological Review, Vol. 65, str. 386-408, 1958.
- [31] Rumelhart D. E., Hinton G. E., Williams R. J.: *Learning internal representation by error propagation*, Parallel Distributed Processing: Exploration in the Microstructure of Cognition, volume I, str. 318-362, MIT Press, Cambridge MA, 1986.
- [32] Rybarski J., Habdank-Wojewódzki S.: *Java Kohonen Neural Network Library (JKNNL)*, URL <http://jknnl.sourceforge.net/tutorial.html>, 2006.

- [33] Sammon J. W., JR: *A nonlinear mapping for data structure analysis*, IEEE Transactions on computers, Vol. C-18, No. 5, 1969.
- [34] Shalini S., M. Carey: *Java implementation of restricted sammon map algorithm*, URL <http://www.uni-trier.de/index.php?id=2031>, Universität Trier.
- [35] Stevens C.: *Die Nervenzelle*, in Gehirn und Nervensystem, str. 2-13, 1988.
- [36] Šíma J., Neruda R.: *Teoretické otázky neuronových sítí*, MATFYZPRESS, 1996.
- [37] Tank D., Hopfield J.: *Simple "Neural"Optimization Networks*, IEEE TCS CAS-33, str. 533-541, 1986.
- [38] Ultsch A.: *U*-Matrix: a tool to visualize clusters in high dimensional data*, University of Marburg, Technical Report, Nr. 36, 2003.
- [39] Yang L.: *Distance-preserving projection of high-dimensional data for nonlinear dimensionality reduction*, IEEE Transactions on pattern analysis and machine intelligence, Vol. 26, No. 9, 2004.

Seznam obrázků

2.1	Biologický neuron (zdroj server http://www.mindcreators.com).	10
2.2	Formální neuron.	10
2.3	Druhy přenosových funkcí (původní zdroj v [36]).	12
2.4	Příklad cyklické topologie.	13
2.5	Topologie sítě perceptronů.	15
2.6	Topologie vícevrstvé neuronové sítě typu 4-5-4-3.	15
3.1	Struktura Kohonenovy mapy.	17
3.2	Funkce laterální interakce tvaru "mexického klobouku".	18
3.3	Gaussova funkce laterální interakce.	19
5.1	Základní operace modelu evolučního stromu.	39
6.1	Porovnání 2 možných zobrazení topologické mřížky z [38].	46
6.2	Příklad určení vzdálenosti u metody nejbližšího souseda.	48
6.3	Příklad určení vzdálenosti u metody nejvzdálenějšího souseda.	48
6.4	Příklad určení vzdálenosti u metody průměrné vzdálenosti.	49
7.1	Hierarchie balíčků v projektu DyKoM.	51
7.2	Architektura celé aplikace.	53
7.3	Architektura manažera sítě <i>NetworkManager</i>	54
7.4	Architektura grafického manažera <i>GraphicManager</i>	54
8.1	Závislost vigilančního koeficientu na čase.	61
8.2	Porovnání 2 výsledných topologických mřížek.	62
8.3	Výsledky zátěžového testu pro 2 různé velikosti topologických mřížek.	67
8.4	Schéma datové struktury pro uložení ukazatelů na vítěze z minulé iterace (původní obrázek uveden v [18] obrázek číslo 3.).	72
8.5	Porovnání výsledných struktur topologických mřížek Kohonenových map za použití Sammonova mapování.	73
10.1	Histogram pro atribut "druh nehody".	80
10.2	Topologická mřížka základního modelu Kohonenovy mapy pro data za rok 2008.	81
10.3	Topologická mřížka rostoucí Kohonenovy mapy pro data za rok 2008.	82

10.4 Výsledné topologické mřížky základního modelu Kohonenovy mapy aplikované na celém souboru dat.	86
10.5 Výsledné topologické mřížky rostoucí Kohonenovy mapy aplikované na celém souboru dat.	86
10.6 Histogramy hmotných škod.	88
B.1 Adresářová struktura projektu.	101
B.2 První menu aplikace.	102
B.3 Členění aplikace.	102

Seznam tabulek

8.1	Výsledky modelů pro Iris data I.	64
8.2	Výsledky modelů pro Iris data II.	65
8.3	Výsledky zátežového testu.	66
8.4	Výsledky modelů pro Pima indians data.	68
8.5	Zátežový test na poškozených Iris datech.	69
8.6	Zátežový test na poškozených Pima indians datech.	70
9.1	Použité atributy vstupních dat, část 1.	75
9.2	Použité atributy vstupních dat, část 2.	76
9.3	Použité atributy vstupních dat, část 3.	77

Příloha A

Obsah CD

Součástí diplomové práce je CD, na kterém se nachází vlastní aplikace a také ostatní důležité dokumenty v elektronické formě. Vše je popsáno v této kapitole.

Vlastní obsah CD:

- Adresář **Data** - nachází se zde všechna originální data o nehodovosti ve městě Ostrava.
- Adresář **Diploma thesis** - obsahuje elektronickou podobu této práce.
- Adresář **Documentation** - zde se nachází programátorská a uživatelská dokumentace k projektu.
- Adresář **Install** - obsahuje archiv DyKoM.zip pro instalaci aplikace, více v části B.1.
- Adresář **Project** - zde se nachází zdrojové kódy projektu DyKoM.

Příloha B

Uživatelská příručka

B.1 Úvod a instalace

Projekt **DyKoM** slouží k návrhu a učení modelů Kohonenových map, jejich grafickému zobrazení a testování. Byl navržen tak, aby mohl být dále jednoduše rozšířen o nové modely Kohonenových map nebo další funkčnosti. Projekt má sloužit především pro testování nově přidaných modelů Kohonenových map a jejich parametrů, ale má implementováno i grafické uživatelské rozhraní pro názornější zobrazení výsledků. Po instalaci je k dispozici několik souborů dat, na kterých mohou být modely testovány. Opět můžeme projekt rozšířit o nové soubory dat.

Program umožňuje také ukládání a načítání jak nově vygenerovaných modelů, tak uložení již naučených modelů Kohonenových map.

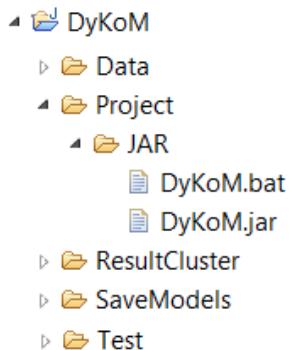
Požadavky na výpočetní prostředky:

- Procesor s taktovací frekvencí 2.0 GHz nebo vyšší.
- Operační paměť alespoň o velikosti 1 GB.
- 500 MB volného místa na pevném disku.
- Grafické rozlišení displeje alespoň 1024 x 768.
- Operační systém Windows XP/Vista/7.

Samotnou instalaci provedeme rozbalením archivu DyKoM.zip z přiloženého CD (CD:\Install\DYKoM.zip) na pevný disk. Program **DyKoM** jsme implementovali v jazyce Java, proto je nutné k jeho spuštění mít nainstalované Java Runtime Environment (JRE) verze 6 (možnost stažení na adrese <http://java.sun.com> nebo je instalační soubor k dispozici na CD v adresáři CD:\Install).

B.2 Struktura a spuštění aplikace

Adresářová struktura aplikace po instalaci:



Obrázek B.1: Adresářová struktura projektu.

Samotné spuštění aplikace se provede souborem **DyKoM.bat**, který se nachází v adresáři Project\JAR.

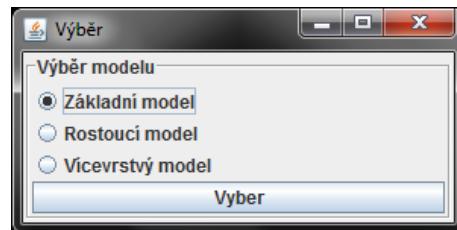
Tento spustitelný soubor obsahuje příkaz *java -jar -DyKoM.jar* (program lze také spustit z tohoto adresáře i přímo souborem DyKoM.jar).

Po spuštění aplikace se objeví konzole, kam jsou vypisovány údaje o průběhu vytváření, učení a testování modelů Kohonenových map. Po inicializaci dojde ke spuštění grafického prostředí, kde se nachází veškeré ovládání aplikace.

B.3 Ovládání aplikace

Po spuštění aplikace je uživateli nabídnuto první menu s výběrem 3 modelů (viz obrázek B.2):

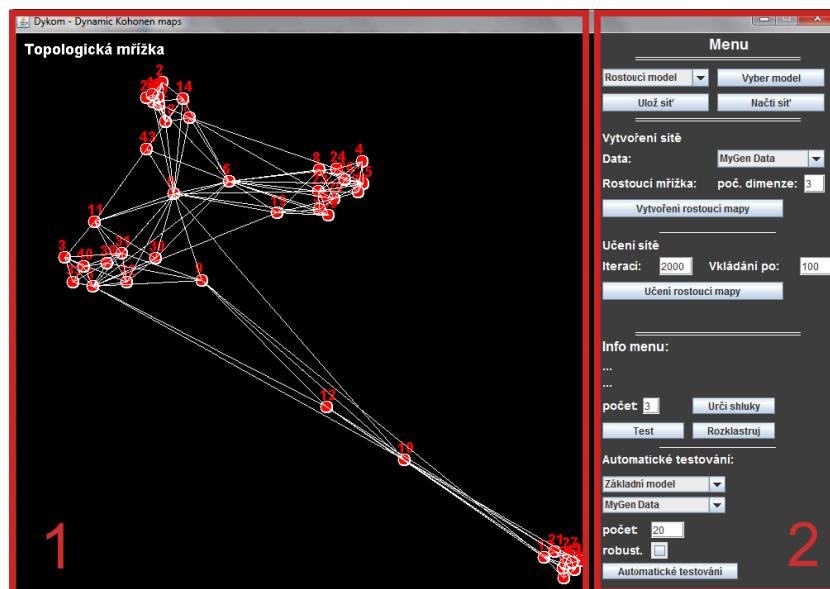
- Základní model.
- Rostoucí model.
- Vícevrstvý model.



Obrázek B.2: První menu aplikace.

Po výběru modelu je vytvořena grafická aplikace. Ta je rozdělena na 2 části (viz obrázek B.3):

- Grafickou plochu, kam je vykreslována topologická mřížka modelu (na obrázku značeno číslem 1).
- Uživatelské menu, které slouží k ovládání modelů (na obrázku značeno číslem 2).



Obrázek B.3: Členění aplikace.

Důležité ovládací prvky uživatelského menu jsou:

- Uzavření aplikace - provede se klasicky stiskem tlačítka "křížku" v pravém horním rohu aplikace.
- Panel **Menu**:
 1. Výběr modelu - po vybrání typu modelu je třeba volbu potvrdit tlačítkem "Vyber model".
 2. Uložení a načtení sítě - tlačítky "Načti síť" a "Ulož síť" je možné provést načtení a uložení sítě do XML souboru.
- Panel **Vytvoření sítě**:
 1. Vytvoření sítě - po vybrání dat, na kterých bude model učen, je potřeba provést potvrzení tlačítkem "Vytvoření mapy".
- Panel **Učení sítě**:
 1. Učení sítě - po zvolení počtu iterací je učení aktivováno tlačítkem "Učení mapy".
- Panel **Info**:
 1. Pro zobrazení informací o konkrétním neuronu je nutno myší kliknout na požadovaný neuron v grafické ploše. V info panelu se poté zobrazí data, která nejlépe odpovídají vybranému neuronu.
 2. Určení shluků - do pole počet shluků je potřeba zadat číselnou hodnotu a následovně stisknout tlačítko "Urči shluky".
 3. Roztřídění dat do souborů dle určených shluků - provede se stisknutím tlačítka "Roztříd".
- Panel **Automatické testování**:
 1. Automatické testování - po zadání počtu opakujících se testů a volbě, zda má být model testován i na robustnost, je tlačítkem "Automatické testování" zahájeno testování.

Učení modelů či testování lze kdykoli přerušit stisknutím tlačítka "Storno". Pro obnovení práce je potom nutné vytvořit model neuronové sítě znovu.