UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2017 Martin Síč

Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Analýza úspěšnosti léčby pomocí Kohenenových map

Martin Síč

Bakalářská práce

2017

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2016/2017**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Martin Síč**
Osobní číslo:	**I13220**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Analýza úspěšnosti léčby pomocí Kohonenových map**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Cílem práce bude navrhnout a naprogramovat aplikaci umožňující statistickou analýzu údajů o pacientech.
Tato aplikace se pomocí umělých neuronových sítí, shlukování jedinců, kontingenčních tabulek a logistické regrese pokusí vybrat vhodnou léčbu a poskytne odhad pravděpodobnosti uzdravení.
Při zpracování teoretické části práce je třeba prostudovat Kohonenovu metodu samoorganizujících se map, kontingenční tabulky a logistickou regresi.
V praktické části bude ve vhodném softwaru vytvořena spustitelná aplikace pro práci s datovým souborem (včetně možnosti filtrování a dalších úprav dat) a pro modelování Kohonenových map poskytující shluky pacientů. Aplikace bude také určovat statistické charakteristiky. Menu aplikace bude umožňovat nastavit parametry inicializace a učení sítě. Současně bude program umožňovat export výsledků a podstatných mezivýpočtů.
V práci budou popsány veškeré aplikované metody a bude vysvětleno uživatelské prostředí aplikace.

Rozsah grafických prací: **10**

Rozsah pracovní zprávy: **35**

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

ZVÁRA, K. a J. ŠTĚPÁN. Pravděpodobnost a matematická statistika. Vyd. 4. Praha: Matfyzpress, 2006, 230 s. ISBN 80-867-3271-1.

ENGELBRECHT Andries P. Computational Intelligence: An Introduction. Chichester: John Wiley & Sons, 2007. ISBN 978-0470035610

MAŘÍK V., O. ŠTĚPÁNKOVÁ a J. LAŽANSKÝ. Umělá inteligence 1. Praha: Academia, 1993. ISBN 80-200-0496-3.

MAŘÍK V., O. ŠTĚPÁNKOVÁ a J. LAŽANSKÝ. Umělá inteligence 2. Praha: Academia, 1997. ISBN 80-200-0504-8

MAŘÍK V., O. ŠTĚPÁNKOVÁ a J. LAŽANSKÝ. Umělá inteligence 3. Praha: Academia, 2001. ISBN 80-200-0472-6.

MAŘÍK V., O. ŠTĚPÁNKOVÁ a J. LAŽANSKÝ. Umělá inteligence 4. Praha: Academia, 2003. ISBN 80-200-1044-0.

Vedoucí bakalářské práce: **Mgr. Jaroslav Marek, Ph.D.**
Katedra matematiky a fyziky

Datum zadání bakalářské práce: **31. října 2016**

Termín odevzdání bakalářské práce: **12. května 2017**

Univerzita Pardubice Česká republika

L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

Mgr. Josef Horálek, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2017

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 13. 5. 2017

Martin Síč

PODĚKOVÁNÍ

Chtěl bych poděkovat vedoucímu práce Mgr. Jaroslavu Markovi, PhD. za odborné vedení mé bakalářské práce, za ochotu, trpělivost a za cenné rady.

ANOTACE

Cíl práce je navrhnout a naprogramovat aplikaci umožňující statistickou analýzu údajů o pacientech. Za pomoci umělých neuronových sítí, shlukování jedinců a kontingenčních tabulek se aplikace pokouší vybrat vhodnou léčbu a poskytuje odhad pravděpodobnosti uzdravení. Aplikace pracuje s datovým souborem, ze kterého se vytvoří Kohonenovy mapy. Aplikace také určuje statistické charakteristiky pacientů. Pomocí menu aplikace se nastavují parametry inicializace a učení sítě. Současně program exportuje výsledky a podstatné mezivýpočty do souboru.

KLÍČOVÁ SLOVA

Neuron, Neuronová síť, Kohonenova mapa, Kontingenční tabulka

TITLE

The analysis of success of treatment using Kohenen maps.

ANNOTATION

The aim of the thesis is to design and program an application enabling statistical analysis of patient data. Using artificial neural networks, clustering of individuals, contingency tables and logistic regression, the application attemps to select appropriate treatment and provides an estimate probability of recovery. The application works with data file, which helps it to model Kohonen maps. It also determines the statistical characteristics of the patients. Using the application menu, the parameters of inicializing and learning of the network are set. At the same time, the program exports the results and significant inter-calculations.

KEYWORDS

Neuron, Neural Network, Kohonen Map, Pivot Table

OBSAH

SEZNAM ILUSTRACÍ A TABULEK

ÚVOD

Tématem bakalářské práce je analýza úspěšnosti léčby leukémie pomocí Kohenenových map a kontingenčních tabulek. Práce se zabývá problematikou těchto map a vysvětlením klíčových prvků pro realizaci sítě. Nejsou nijak zvlášť složité, ale je nutné pochopit všechny jejich principy.

Na začátku práce se čtenář seznámí s mírami podobnosti, které slouží pro hledání podobného objektu v neuronové síti. S mírami podobnosti je spjatá také standardizace dat, pomocí které se data připravují pro tvorbu sítě.

Zmíněné neuronové sítě vycházejí z biologické inspirace, kde pomocí mnoha experimentů je bylo možné matematicky popsat a začít využívat pro řešení různých úloh. Základním prvkem sítě je neuron. Spojením většího množství neuronů vzniká neuronová síť.

Jedním z představitelů neuronových sítí jsou Kohonenovy mapy. Tento typ map využívá učení bez učitele, což znamená, že k učícímu procesu není potřeba cizí pomoci. Tento proces se s pomocí učících vzorů snaží připravit síť pro vložení objektů. Učící vzory musejí být samozřejmě co nejpodobnější objektům, které se mají do sítě vložit. Pokud je síť naučena, objekty se do sítě postupně vkládají.

Praktickou částí je aplikace na analýzu úspěšnosti jednotlivých léčebných metod leukémie. Tato analýza se provádí pomocí kontingenčních tabulek a funkce přežití. Aplikace umí pracovat s datovým souborem, který obsahuje data o pacientech. Data lze modifikovat a ukládat zpět do načteného souboru. Pomocí informací o pacientech aplikace vytváří Kohonenovy mapy a při přidání nového pacienta se doporučí vhodná léčba. V textu práce je uveden manuál, podle kterého lze aplikaci snadno ovládat.

1 MÍRY PODOBNOSTI A TRANSFORMACE DAT

V dalších kapitolách budou využívány Kohonenovy mapy pro hledání podobných objektů, kdy se použije podobných principů jako u shlukové analýzy. Obvykle jsou data předem zpracovávána pomocí standardizace a jejich podobnost se měří pomocí měr podobnosti. V dalším textu se bude předpokládat, že je k dispozici *m* objektů, u kterých bylo změřeno *n* znaků (faktorů) [1].

1.1 Míry podobnosti

Shluky se tvoří na základě podobnosti objektů. V první řadě je nutné stanovit znaky, které budou určovat podobnost, aby bylo možné dané objekty porovnávat mezi sebou. Objekty mohou navíc obsahovat některá data, která pro shlukování nejsou potřeba. Musí se tedy stanovit pouze ty znaky, podle kterých se budou objekty porovnávat. Většinou je nutné měřené hodnoty znaků standardizovat a teprve následně provádět shlukování. Měřit podobnost lze různými metrikami, z nichž některé budou popsány dále[1].

Nejčastěji se pro hledání podobnosti používají míry vzdálenosti. Znaky jednotlivých objektů představují souřadnice v prostoru. Po vložení objektů do prostoru se vypočítá vzdálenost mezi nimi. Pro každou míru a libovolné objekty a, b, c musí platit:

- Symetrie: $d(a,b) = d(b,a)$.
- Nezápornost: $d(a,b) \geq 0$.
- Identita: $d(a,a) = 0$.
- Trojúhelníková nerovnost: $d(a,b) \leq d(a,c) + d(c.b)$.

Pro výpočet vzdálenosti v *n*-rozměrném prostoru existuje velké množství metod. Nejvíce používané metody a jejich principy jsou popsány níže. [1].

Euklidovská vzdálenost je nejpoužívanější mírou vzdáleností. Vzdálenost dvou objektů měří podobně jako Pythagorova věta, tedy počítá velikost přepony pravoúhlého trojúhelníku. Rovnice pro výpočet euklidovské vzdálenosti:

$$d_E(x_a, x_b) = \sqrt{\sum_{i=1}^{n} (x_{ai} - x_{bi})^2}. \quad (1)$$

Při používání této metody se musí brát v potaz, že nemá horní hranici hodnot a ve většině případů je nutné provést standardizaci dat [1].

Hammingova vzdálenost se používá pro binární čísla. Počítá součet rozdílů hodnot mezi dvěma stejně velkými řetězci [1]. Je definovaná vztahem:

$$d_H(x_a, x_b) = \sum_{i=1}^{n} |x_{ai} - x_{bi}|. \tag{2}$$

Minkovského metrika slouží jako obecný výpočet vzdálenosti. Rovnice této metody:

$$d_M(x_a, x_b) = \sqrt[k]{\sum_{i=1}^{n} |x_{ai} - x_{bi}|^k}, \tag{3}$$

kde parametr *k*, mění typ používané metody. Při nastavení parametru na k = 1 se jedná o Hammingovu metriku a při $k = 2$ jde o eukleidovskou vzdálenost. Čím vyšší se určí parametr *k*, tím větší se klade důraz na rozdíly hodnot [1].

Mahalanobisova metrika je vlastně standardizace, která bere v potaz korelaci mezi znaky a není závislá na použitém rozsahu hodnot. Je definována rovnicí:

$$d_{MA}(x_a, x_b) = \sqrt{(x_a - x_b)^T C^{-1} (x_a - x_b)}. \tag{4}$$

kde C je kovariační matice [1].

Je nutné brát v potaz, že vybraná metrika vzdálenosti ovlivní výsledek shluků. Musí se tedy pečlivě vybrat správná metrika pro danou analýzu. Problém může nastat, při používání nestandardizovaných dat, jelikož často jsou hodnoty v jiných jednotkách. Při porovnávání hodnot v rozsahu od -1 do 1 a například od -100 do 100, je nutné hodnoty standardizovat, jak již už bylo zmíněno [1].

Pro zjištění míry lineární závislosti dvou objektů se používají korelační míry. Využívá se zde korelační koeficient, který nabývá hodnot od -1 do 1 a čím vyšší hodnota koeficientu, tím vyšší je jejich lineární závislost [1].

Pro objety obsahující binární proměnné se používají míry asociace. U této metriky se sleduje, zda daný znak se v objektu vyskytuje, či nikoli. Jako jednoduchý příklad lze uvést dotazník, ve kterém dva lidé odpovídají *ano* a *ne*. Mírou asociace pak bude procento otázek, u kterých odpověděli oba správně [1].

1.2 Standardizace dat

Standardizace dat je nedílnou součástí shlukové analýzy. Objekty často obsahují znaky, které jsou ve velmi odlišných jednotkách. Vytváření shluků z těchto rozdílných hodnot by vedlo

k tomu, že některé znaky by ovlivňovaly shlukování mnohem více než ostatní. Proto je důležité si uvědomit, zda je nutné standardizaci použít nebo zda jsou hodnoty ke shlukování připravené bez nutnosti jejich úprav[1].

Pro standardizaci dat se nejčastěji používá *výběrový průměr* a *výběrová směrodatná odchylka*. Výpočet se provede tak, že od každé hodnoty se odečte aritmetický průměr a vydělí směrodatnou odchylkou. Nově vypočítaná hodnota, která se nazývá *z-skóre*, bude přibližně z rozsahu od -3 do 3, kde její střední hodnota je rovna 0 a rozptyl roven 1 [1]. Rovnice:

$$z = \frac{x_{ij} - \bar{x_j}}{s_{jj}}. \tag{5}$$

Standardizace variačním rozpětím se používá převážně tam, kde jsou hodnoty ve stejném měřítku, ale rozsahy jsou velmi odlišné. Vypočítá se tak, že od každé hodnoty se odečte minimální hodnota počítaného znaku a vydělí se vše variačním rozpětím [1]. Rovnice:

$$z = \frac{x_{ij} - min_j(x_j)}{max_j(x_j) - min_j(x_j)}. \tag{6}$$

2 PRINCIP NEURONOVÝCH SÍTÍ

Neuronové sítě vznikly na základě neurofyziologické inspirace, která bude popsána v následující podkapitole. První práce, která zapříčinila vznik oboru neuronových sítí je práce Warrena McCullocha a Waltera Pittse v roce 1943, kteří vytvořili matematický model neuronu s cílem ukázat, že i ty nejjednodušší neuronové sítě mohou spočítat jakoukoli logickou nebo aritmetickou funkci. Přestože nečekali praktické využití jejich práce, i tak měla za následek další vývoj neuronových sítí [2].

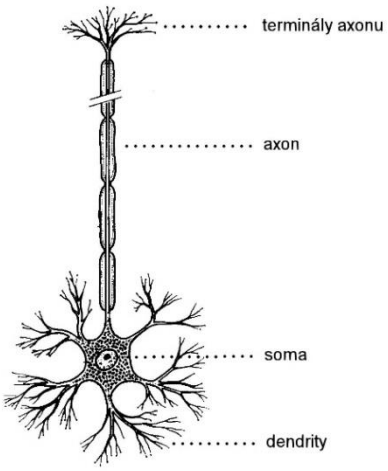
2.1 Neurofyziologická inspirace

Hlavním důvodem zkoumání neuronových sítí bylo pochopení lidského myšlení a jak vůbec lidský mozek funguje. Pro řešení úloh z umělé inteligence je možné použití neurovýpočtů, které vycházejí právě z neurofyziologických vědomostí a z nich vytvořených matematických modelů. Tyto zmíněné matematické modely sice vycházejí z neurofyziologie, ale jejich další rozvoj už nemusí nutně souviset s modelováním lidského mozku. Přesto se k modelům modelujícím lidský mozek vrací, jelikož lze podle nich popsat dané vlastnosti matematického modelu. Proto je důležité mít vědomosti z neurofyziologie na takové úrovni, aby bylo možné pochopit prvotní modely neuronových sítí [2].

Nervová soustava umožnuje živým organismům zajistit spojení mezi organismem, jeho jednotlivými částmi a prostředím okolo něj. Dovoluje tedy reagovat na vnější podnět, ale i na vnitřní stavy daného organismu. Reakce probíhá na základě posílání vzruchů ze snímačů, tzv. *receptorů*, které dovolují přijmout různé podněty (například tepelné podněty) a směrovat je k dalším nervovým buňkám. Buňky tyto signály zpracují a následně je pošlou daným výkonným orgánům, tzv. *efektorům*. K první úpravě informace dochází v *projekčních drahách*. Po těchto drahách se vzruchy dostanou až do mozkové kůry, která je pro nervový systém řídícím centrem [2].

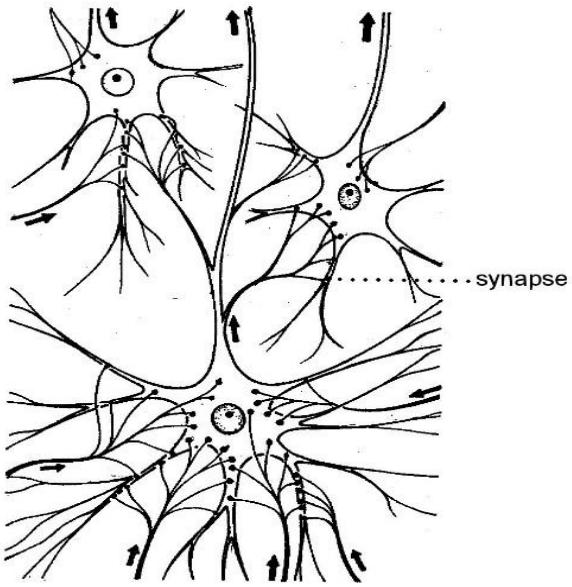
Povrch mozku se dá rozdělit na šest *projekčních oblastí*, které jsou mezi sebou propojeny. Tyto oblasti zhruba odpovídají smyslům, které paralelně zpracovávají informace. Až v *asociačních oblastech* se souhrnně zpracovávají informace, což je nutné pro řízení činnosti efektorů. Toto zpracovávání informací probíhá již sekvenčně [2].

V předešlém textu byl stručně vysvětlen princip přenosu informací v živém organismu a zmíněn základní prvek nervové soustavy, nervová buňka, tzv. *neuron*. Neurony jsou speciální buňky, které přenášejí, uchovávají a zpracovávají informace. Jsou nutné pro životní funkce organismu, bez nich by životní funkce nešly realizovat [2].



Obrázek 1: Biologický neuron, viz [2]

Jelikož neuron přenáší informace, musí k tomu byt přizpůsobený. Jeho tělo, tzv. *soma*, obsahuje přenosové kanály pro vstup a výstup. Vstupním kanály se nazývají *dendrity* a výstupní *axony*. Dendrity obsahují výběžky, tzv. *trny*, axony mají na konci větve nazývané *terminály*, které jsou zakončeny blánou. Neurony se pak stýkají pomocí blan terminálu a trnů. Pro představu jak vypadá biologický neuron, je tu obrázek 1 [2].



Obrázek 2: Biologická neuronová síť, viz [2]

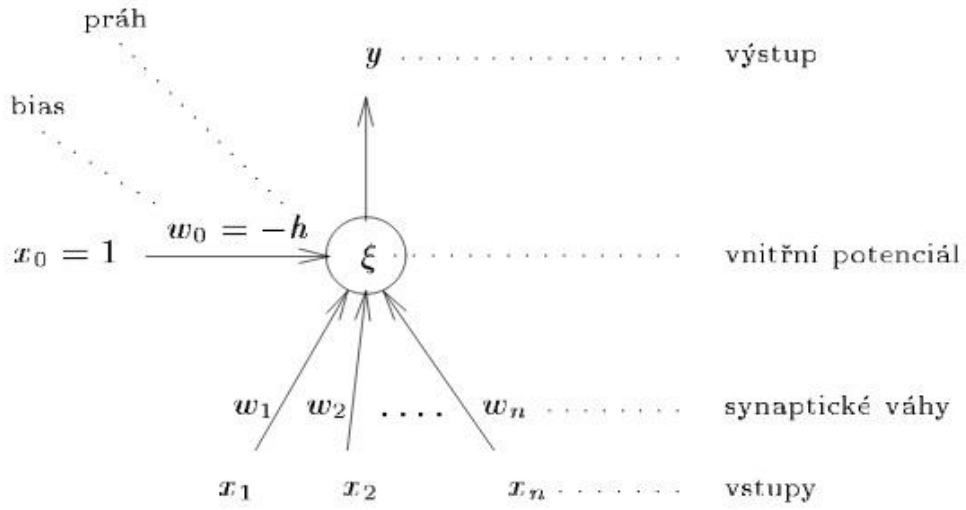
Spojení více neuronů pomoci terminálů a axonů se nazývá neuronová síť, která je znázorněna na obrázku č. 2. Na obrázku je také vidět přenos informací, který je prováděn pomocí rozhraní, tzv. *synapse*. Veškeré důležité informace obsahuje *míra synaptické propustnosti*.

Synapse se pak dělí na *excitační* a *inhibiční*. Excitační umožňuje v nervové soustavě zvětšit vzruch, naopak inhibiční umožňuje vzruch zmenšit [2].

Jelikož soma i axon mají na sobě membrány, je jimi umožněno posílat informace. Membrána totiž dokáže v jistých situacích vytvářet elektrické impulsy. Impulsy se pak přenášejí pomocí synaptických bran, z axonu jednoho neuronu, na dendrit jiného neuronu pomocí synaptických bran. Tyto brány pomocí propustnosti udávají míru podráždění následujících neuronů. Neurony mají určitou hranici podráždění, tzv. *prah*. Pokud neuron dosáhne prahu, vytvoří impuls a umožní tak přenesení dané informace. Schopnost neuronů si pamatovat informace je z tu důvodu, že synaptická propustnost se mění s každým průchodem signálu [2].

2.2 Matematický model neuronu

V předchozí části byla vysvětlena funkce biologického neuronu, která usnadní pochopit, jakým způsobem funguje matematický model neuronu. Aby bylo možné vytvářet umělé neuronové sítě, je nutné převést právě zmíněný biologický neuron na umělý, tzv. *formální neuron*, pomocí převedení funkcí biologického neuronu na matematické vyjádření [2].



Obrázek 3: Struktura formálního neuronu, viz [2]

Jak vypadá formální neuron, je možné vidět na obrázku 3. Vstupy formálního neuronu jsou modely dendritů, které obsahují n vstupů $x_1, \ldots, x_n$, a jsou hodnoceny synaptickými váhami $w_1, \ldots, w_n$. Tyto váhy udávají, jak velkou propustnost vstupy mají. Pokud se provede vážená suma vstupních veličin, jedná se o tzv. *vnitřní potenciál*. Mezi vstupní hodnoty se zařazují jak vstupy, tak i synaptické váhy. Pokud vnitřní potenciál dosáhne prahové hodnoty, tak se vytvoří výstup neuronu, který simuluje elektrický impuls axonu. Dále výstupní hodnota roste

nelineárně a tento růst je popsán tzv. *aktivační funkcí*, kde nejzákladnější je *ostrá nelinearita*. *Biasem* $w_0 = -h$ vstupu $x_0 = 1$, je váha prahu formálního neuronu, která má záporné znaménko, čehož lze docílit úpravou a aplikací aktivační funkce, poté bude mít nulový práh [2].

2.3 Umělá neuronová síť

Samotný formální neuron nedokáže řešit složitější úlohy, z tohoto důvodu se neurony spojují do neuronových sítí. U biologické neuronové sítě jsou neurony spojeny pomocí synaptické vazby, která spojuje terminály axonu s dendrity, jak již bylo zmíněno v předešlém textu. U umělé neuronové sítě jsou spojeny tak, že výstup slouží pro vstup dalších neuronů. V praxi se rozlišují tři typy neuronů podle toho, k čemu jsou využívány. První typ neuronu je vstupní, který odpovídá receptorům. Dále zde jsou výstupní neurony, které jsou shodné s efektory. V poslední řadě tu jsou pracovní neurony, které při spojení slouží jako *cesty* pro posílání vzruchů. Jelikož neurony ležící na cestě dokáží měnit své stavy, umožňují tedy posílat a zpracovávat informace. Hlavní aspekt, který řeší, jak neuronová síť bude vypadat, je *topologie neuronové sítě*, která je dána tím, jak jsou neurony spojeny a jejich počtem. *Stav neuronové sítě* je dán podle veškerých stavů neuronů, které neuronová síť obsahuje. *Konfigurace neuronové sítě* je dána všemi synaptickými váhami spojů [2].

Jelikož se neuronová síť v závislosti na čase postupně mění, má to za následek změnu stavu neuronů a jejich propojení. Při těchto změnách se také přizpůsobují váhy neuronů. Z důvodů těchto změn je nutné dynamiku sítě rozdělit na tři dynamiky a brát pak v potaz, že pro každou z těchto dynamik je jiný systém (režim) práce. Dynamika se tedy dělí na adaptivní dynamiku, aktivní dynamiku a organizační dynamiku. Základní princip jednotlivých dynamik bude popsán v následujícím textu. Takto rozdělená dynamika neuronové sítě nesouhlasí s tím, jak to je v neurofyziologii, protože ve skutečnosti veškeré změny probíhají v jednom okamžiku, ale u uměle vytvořené neuronové sítě to není prakticky možné. Změny charakteristik neuronové sítě v čase jsou určeny pravidlem. Toto pravidlo udávají dynamiky, které jsou převážně zadány rovnicí a prvotním stavem. Charakteristikami je myšlena konfigurace, topologie a stav neuronové sítě. Změny jsou prováděny v daných režimech práce, a obsluhují se právě pomocí těchto pravidel [2].

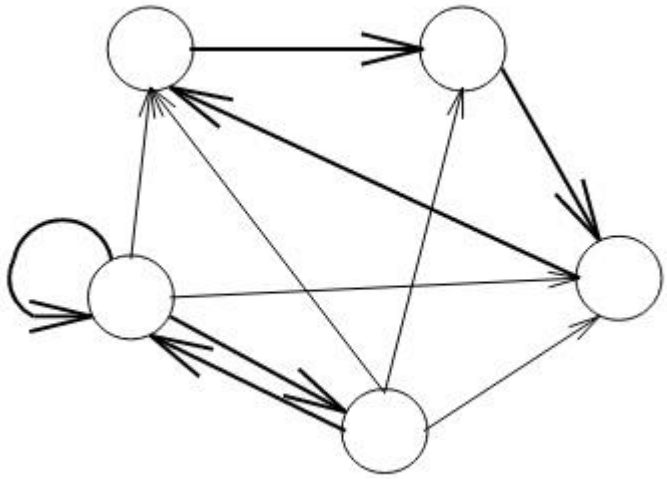
Aby bylo možné mít jiné modely neuronové sítě, které budou sloužit pro výpočet specifických příkladů, je nutné i dané dynamiky specifikovat. Specifikace se provádí tak, že

se určí aktívní, adaptivní a organizační dynamika. Dynamiky jsou tedy velmi důležité, co se týče vytváření neuronových sítí a je nutné pochopit jejich základní princip [2].

2.3.1 Organizační dynamika

Jedná se o dynamiku, která určuje, jakou architekturu síť bude mít a její možnou změnu. Pokud je nutné přidat další neurony, musí se síť zvětšit a přidat další spoje. To se provádí pomocí adaptivního systému a tím se mění topologie sítě. Ovšem tato dynamika většinou počítá, že architektura je stálá a v pozdějším čase se už neupravuje [2].

Architektura sítě se dělí na dva druhy, cyklickou a acyklickou síť. Cyklická topologie je to tehdy, pokud část sítě obsahuje neurony propojené v kruhu. Jedná se v podstatě o propojení neuronů tak, že výstup prvního neuronu je propojen se vstupem druhého, výstup druhého slouží k propojení se vstupem dalšího atd., až poslední neuron z tohoto kruhu je propojen se vstupem prvního. Nejzákladnější ukázkou je cyklus, kde neuron má výstup propojen se svým vstupem. Příkladem cyklické sítě může být například *úplná topologie*, která obsahuje maximum možných cyklů, kde na výstup jakéhokoli neuronu je připojen vstup všech neuronů. Jak může vypadat například cyklická architektura, to je ukázáno na obrázku 4 [2].

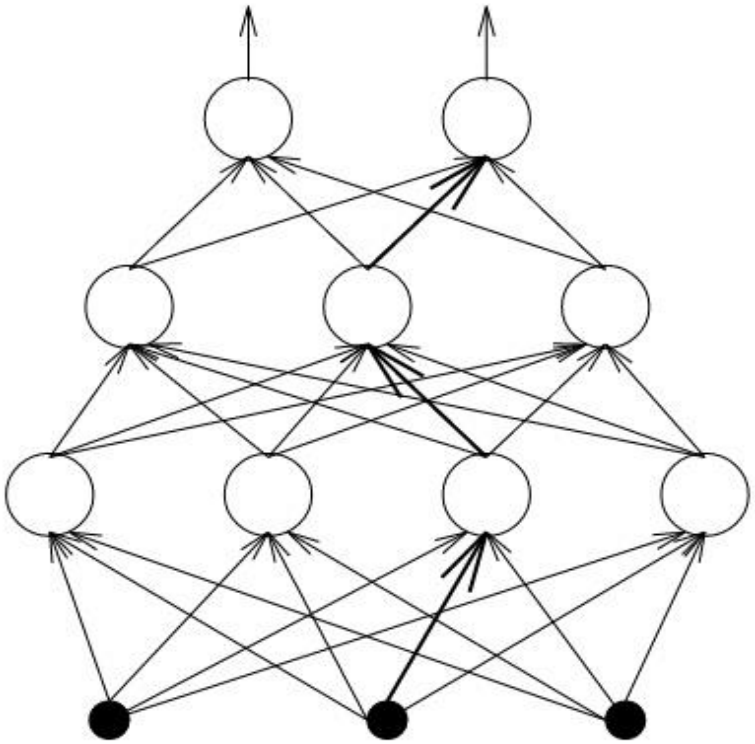


Obrázek 4: Ukázka cyklické architektury, viz [2]

U acyklické architektury žádný cyklus není a je možné neurony rozčlenit do jednotlivých vrstev tak, aby spoje vedly pouze od nejnižší vrstvy až do té úplně nejvyšší. Není dáno, že spoje neuronu musejí vést vždy k té další, mohou totiž vrstvy přeskakovat. Zvláštním druhem této architektury je síť zvaná *vícevrstvá neuronová síť*, která se dá rozložit na tři části. Každá z těchto částí má svůj specifický účel. První část je složena ze vstupních neuronů, které dohromady tvoří vstupní vrstvu. Další část obsahuje skryté neurony, tedy se nazývá skrytá

vrstva. Poslední částí je výstupní vrstva obsahující výstupní neurony. Jednotlivé vrstvy jsou číslovány od nuly výše, kde nula značí vstupní vrstvu, kterou do výsledného množství vrstev nezahrnujeme. Pokud bychom tedy měli například třívrstvou neuronovou síť, znamenalo by to, že síť je složena z jedné vstupní, jedné výstupní a dvou skrytých vrstev [2].

Architektura takovéto sítě se zadává pomocí množství neuronů, kde se zadá jako první množství neuronů vstupní vrstvy, poté jsou zadány počty neuronů jednotlivých skrytých vrstev a jako poslední neurony výstupní vrstvy. Ve většině případů se počty neuronů v jednotlivých vrstvách oddělují pomlčkou. Pro dvouvrstvou neuronovou síť 4-7-2 platí, že obsahuje 4 vstupní neurony, 7 neuronů v jedné skryté vrstvě a 2 neurony výstupní. Ve vícevrstvé neuronové síti co se týče topologie, jsou všechny neurony z vrstev ležících těsně vedle sebe propojeny. Pokud spoj obsahuje nulové váhy, lze takovýto spoj vyloučit. Na následujícím obrázku 5, je ukázka třívrstvé neuronové sítě [2].



Obrázek 5: Ukázka architektury vícevrstvé neuronové sítě 3-4-3-2, viz [2]

2.3.2 Aktivní dynamika

Tato dynamika udává prvotní stav neuronové sítě, a jak se síť bude měnit v průběhu, pokud je stejná konfigurace sítě i její topologie. Při počátečním nastavování stavu sítě je nutné na vstup nastavit stavy veškerých vstupních neuronů. Ostatní neurony potom jsou v daném prvotním

stavu. Vstupní prostor sítě je pak tvořen veškerými vstupy. Následně po prvotním nastavení se provádí výpočet. Většinou je stav sítě určen diferenciální rovnicí, která je funkcí času. Tento stav má spojitý průběh a je tu tedy řeč o spojitém modelu [2].

Převážně se počítá s diskrétním časem, kde v počátku je síť v nulovém čase a mění se pokaždé v určitých časových krocích (1,2,3,4,5,…). V těchto krocích se určí neuron, nebo skupina neuronů, které podle určitého kritéria změní svůj stav podle neuronů, jejichž výstup je vstupem neuronů, které budou měnit svůj stav [2].

Neurony upravují své stavy bez ohledu na sobě samém, nebo je tato změna řízena jednotně. Podle toho, jak neurony mění své stavy, se rozlišují synchronní a asynchronní modely. Konečný výsledek počítání je pak stav výstupních neuronů, který se měnil průběhem času. Vzhledem k tomu, že většinou se používá aktivní dynamika, kde se výstup postupem času nemění, tak se používá funkce na vstupu v aktivním režimu, kde se k jednotlivým vstupům sítě počítá vždy jeden výstup. Zmíněná funkce je určena aktivní dynamikou, která je závislá na konfiguraci a topologii. Z tohoto vyplývá, že síť v aktivním režimu se používá pro své vlastní výpočty [2].

2.3.3 Adaptivní dynamika

Poslední dynamikou neuronové sítě je adaptivní dynamika. Jak už z názvu této dynamiky vyplývá, říká nám, jak se adaptují váhy v průběhu času, a udává prvotní konfiguraci sítě. Veškeré konfigurace pak vytvoří váhový prostor. V režimu této dynamiky se na počátku určí váhy spojů na prvotní konfiguraci. Většinou tyto váhy bývají nastaveny náhodně. V okamžiku, kdy je nastavena počáteční konfigurace, se začne provádět adaptace. Zde se také používá model, který je spojitý a má spojitý průběh konfigurace. Váhy jsou funkcí času a tato funkce je zde většinou dána diferenciální rovnicí, obdobně jako v aktivní dynamice [2].

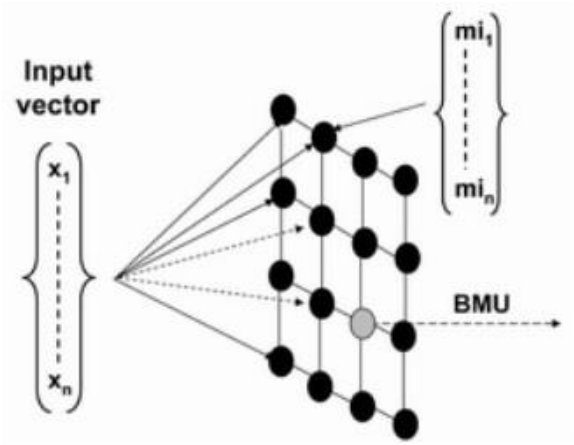
V aktivním režimu je funkce závislá na konfiguraci sítě. Adaptace se snaží najít konfiguraci z výše zmíněného váhového prostoru, aby v tomto režimu dokázala danou funkci provést. Pro vysvětlení, pokud se aktivní režim používá k vypočítání funkce pro vstup, pak tento režim se používá pro naučení dané funkce [2].

3 KOHONENOVY MAPY

Základním a nejpoužívanějším typem neuronových sítí jsou samoorganizující se mapy, které mají svůj název převzatý z anglického překladu Self-Organizing Maps, ale jsou převážně známé pod názvem Kohonenovy mapy. Tento typ patří mezi samoučící se neuronové sítě. Jak už z názvu vypovídá, tato síť se učí sama, tedy bez učitele, a ke své inicializaci nemusí mít perfektní vzory. K tomu, aby se mohla síť sama učit, postačí jen větší množství reálných signálů, kde alespoň některé mají danou společnou vlastnost, nebo se zásadním způsobem liší a nepotřebují se k nim přidat už žádné další učící signály nebo požadované hodnoty. Požadované hodnoty při učení s učitelem určují situaci, které chce síť svým učením dosáhnout. Nastává velmi často problém, jak tyto hodnoty získat. U Kohonenovy mapy postačí jen skupina signálů a v průběhu učení si sama najde stejné vlastnosti a odlišnosti, na jejichž základě se v aktivním režimu bude rozhodovat. To je obrovská výhoda těchto map, kvůli které jsou právě tyto mapy často používané a řadí se mezi nejpopulárnější [3].

3.1 Struktura

Základní strukturou je dvouvrstvá síť (Obrázek 6), ve které jsou veškeré neurony obou vrstev spolu spojeny, takovému spojení se říká úplné. První vrstva je vstupní a je složena z *n* neuronů, která reprezentuje vstupní vektor. Další vrstvou je vrstva výstupní, která je organizována do topologické struktury. Tato struktura převážně bývá dvourozměrná mřížka, nebo jednorozměrný řetězec neuronů. Udává také, jaké neurony jsou spojeny v určité vrstvě do okolí. Okolí určuje, které neurony spolu sousedí [1].

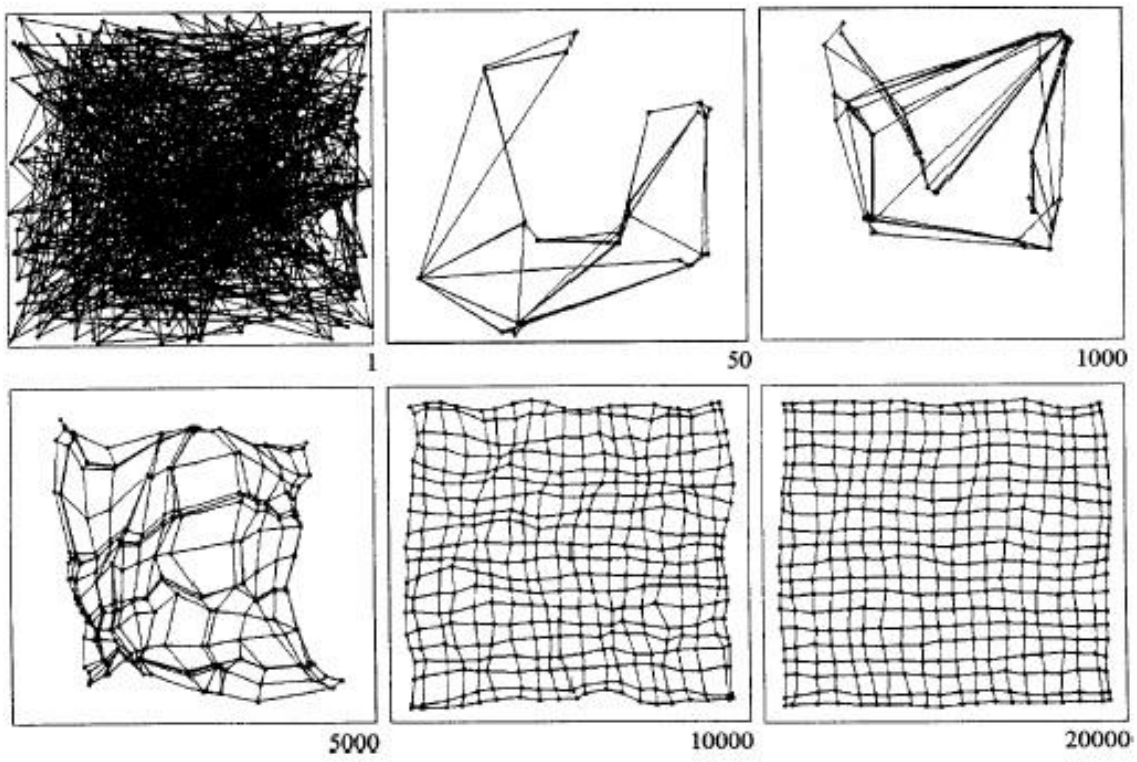


Obrázek 6: Struktura Kohonenovy mapy, viz [3]

3.2 Učení Kohonenovy mapy

Algoritmus učení má snahu seřadit neurony v mřížce do specifických oblastí, protože musí být schopné rozdělit příchozí hodnoty. Představit si členění neuronů se dá tak, že se vezme zmačkaný papír, který se postupně narovnává [4].

Průběh učení je *autonomní*, což značí právě učení bez učitele, tedy bez jakékoli pomoci zvenčí. Navíc je jeho průběh iterativní, takže v každém kroku se mění váhy. Při změně vah se porovnávají příchozí vzory s hodnotami uloženými ve vektoru, který obsahuje neuron. Po tom se najde vektor, který se nejvíce přibližuje předloženému vzoru a hodnoty ve vektoru změní. Při této změně se mění i vektory, které jsou v blízkosti neuronu, který se má upravovat. Tímto způsobem se postupně upravuje mřížka, aby se co nejvíce podobala předloženým cvičícím hodnotám. Před učením nejsou neurony nijak uspořádány, což je možné vidět na Obrázku 7. V průběhu učení se tedy neurony postupně rozpínají a v konečné fázi učení už má mřížka nějaký tvar, který odpovídá předloženým hodnotám. Podrobněji bude průběh učení vysvětlen dále [4].



Obrázek 7: Průběh učení Kohonenovy mapy, viz [5]

3.2.1 Algoritmus učení Kohonenovy mapy

Samotný průběh učení bude popsán v následujícím algoritmu a na pochopení to není nic složitého. Předpokládá se, že je síť s N vstupy a M neurony, kde všechny neurony mají nastavenou svoji váhu w_{ij}. Jednotlivé postupy jsou popsány v krocích [4].

1. Inicializace:
   - Úplně na samotném začátku se nastaví váhy w_{ij}, $0 \leq i \leq N-1, 0 \leq j \leq M-1$ pro všechny synaptické spoje z N příchozích do M odchozích neuronů na malé začáteční hodnoty.
   - Další, co je důležité nastavit, je parametr učení $\eta(0)$. Ten se nastavuju na hodnotu blízkou jedné. Hodnota se bere z intervalu $0 \leq \eta(t) \leq 1$, která se používá pro rychlost učení sítě.
   - Poslední, co je potřeba nastavit je, jak velké počáteční okolí bude kolem každého příchozího neuronu. Počáteční okolí se nastavuje pro všechny neurony stejné, aby okolí obsahovalo veškeré neurony. To znamená, že poloměr okolí se bude rovnat velikosti mřížky, tedy součtu neuronů na její straně. Aby byl proces úspěšný, je také potřeba nastavit nejmenší hodnotu okolí. Po dosažení nejmenší hodnoty okolí, se okolí přestane zmenšovat. Pokud to situace nevyžaduje jinak, tak se nejmenší okolí nastavuje na právě jediný neuron, který se mění.
2. Předložení učících hodnot:
   - V tomto kroku se předá učící vzor $X(t) = \{x_0(t), x_1(t), \dots, x_{N-1}(t)\}$, na začátek neuronové sítě.
3. Počítání vzdáleností:
   - Vzdálenosti se počítají pomocí euklidovské vzdálenosti (popsána v kapitole 1.1).
4. Výběr vítězného neuronu:
   - Výběr vítězného neuronu probíhá tak, že se vezme nejmenší spočítaná vzdálenost mezi vstupním vzorem a výstupním neuronem.
5. Změna vah
   - Změní se váhy pro vítězný neuron j^* a samozřejmě i okolí daného neuronu $N_j^*(t)$. To znamená, že veškeré neurony, které leží uvnitř tohoto okolí, změní své váhy podle vztahu:

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)\left[x_i(t) - w_{ij}(t)\right], \quad (7)$$

kde,

j – reprezentuje neurony, které jsou uvnitř okolí $N_j^*(t)$,

i – reprezentuje vstupy,

$w_{ij}(t)$ – je váha daného neuronu před změnou,

$\eta(t)$– je parametr učení,

$x_i(t)$ – jsou hodnoty vstupního vzoru.

6. Ukončení učícího procesu:
    - Pokud stále zbývají učící vzory, které chceme ještě síti předložit, nebo nebyl dosáhnut požadovaný počet iterací, je nutné se vrátit ke kroku 2 a v učení pokračovat. Pokud byla splněna podmínka ukončení, tak proces učení sítě končí [4].

3.2.2 Učení s funkcí sousednosti

Jak už bylo řečeno, upravují se váhy nejenom pro vítězný neuron, ale i pro neurony v jeho okolí. V biologických sítích je tato adaptace podobná. Na obrázku 8a je možné vidět Gaussovu funkci sousednosti, která popisuje pravděpodobnosti změny vah. Funkce byla získána pomocí experimentů biologických neuronových sítí a říká se jí zvonovitá funkce. Jak už z obrázku vyplývá, tak u nejbližších neuronů se upravuje váha téměř stejně jako vítěznému neuronu a čím dále jsou neurony od vítězného neuronu, tím méně se jejich váhy mění. Po protnutí osy x se váhy mění na opačnou stranu. Z toho vyplývá, že po dosažení osy *x*, se neurony tzv. odnaučují. Právě zmíněným odnaučováním, se zvyšuje efektivita učení. Funkce sousedství lze vyjádřit rovnicí:

$$\lambda(i^*, i) = h(t) \exp\left(\frac{d_E^2(i^*, i)}{r^2(t)}\right), \quad (8)$$

kde

$D_E(i^*, i)$ – je vzdálenost mezi porovnávaným neuronem $h(t)$ a vítězným neuronem i*,

$h(t)$ – je adaptační výška funkce sousedství,

$r(t)$ – je poloměr funkce sousednosti.

Parametry $h(t)$ a $r(t)$, se postupem času zmenšují směrem k nule. Následně nastává úprava funkce na změnu vah a ta má potom následující rovnici:

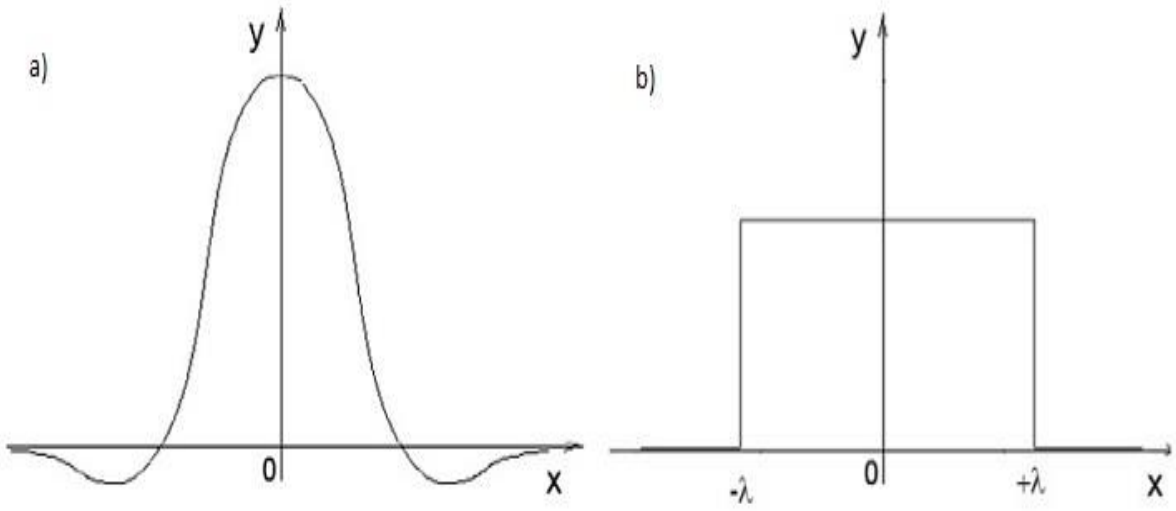
$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)\lambda(i^*, i)\left[x_i(t) - w_{ij}(t)\right]. \quad (9)$$

To má za následek úpravu vah v celé síti, ne jenom v okolí daného neuronu. Ovšem nejjednodušší funkcí je pravoúhlé okolí, které je znázorněno na obrázku 8b a je definované následujícím způsobem:

$$h(i^*, i) = \begin{cases} 1, & pokud\ d(i^*, i) \leq \lambda(t) \\ 0, & u\ všech\ dalších\ možností' \end{cases}$$

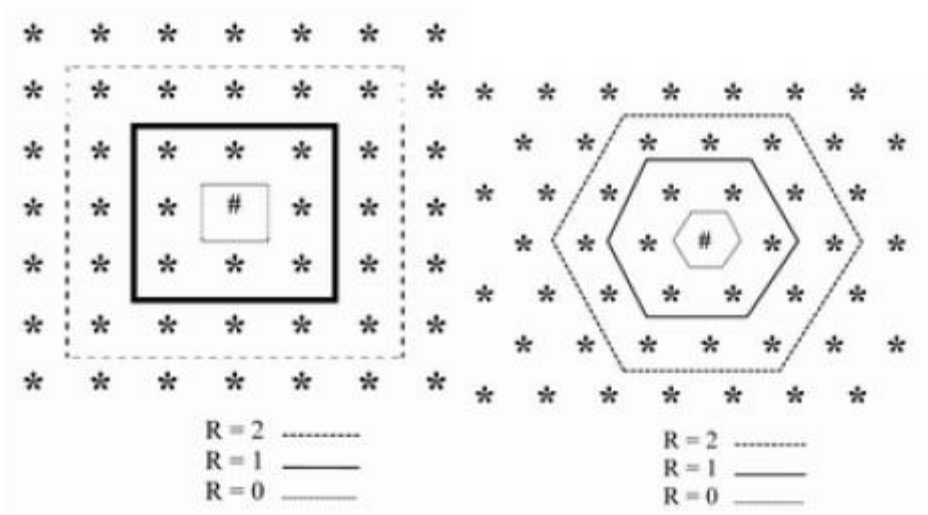
kde

$d(i^*, i)$ – je vzdálenost mezi i^* a i.



Obrázek 8: a) Gaussova funkce sousednosti, b) Pravoúhlé okolí (vlastní)

Co se týče okolí, není dáno, že musí být pouze čtvercové (Obrázek 9). Existuje celá řada dalších použitelných okolí, hexagonální, kruhové a další. To samé platí i u vstupních neuronů, které se také dají uspořádat různě. Mohou se zařadit za sebou do jednoho velkého pásu. Potom při změně vah v okolí neuronu se mění neurony nalevo a napravo od zvoleného neuronu. To bude mít za následek, že mřížka nebude mít tvar čtverce, ale bude to jeden dlouhý řetěz. Dá se taktéž vytvořit například jeden dlouhý obdélník. Na obrázku 9 je znázorněno postupné klesání vah [4].

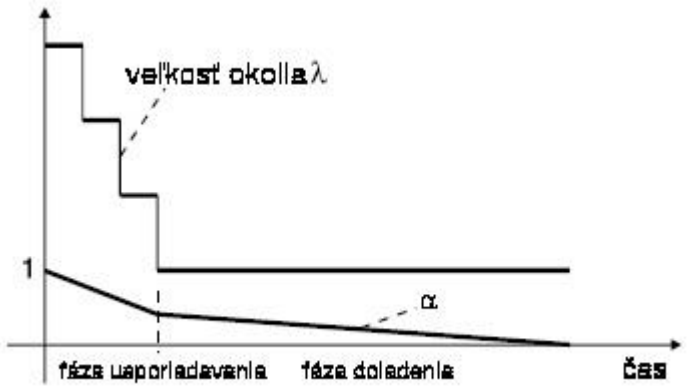


Obrázek 9: Druhy okolí, viz [3]

Pro správné chování Kohonenovy mapy je velmi důležitým procesem změna vah a okolí. Je tedy důležité si na tyto procesy dávat pozor.

3.2.3 Změny parametrů v průběhu učení

Při simulacích se Kohonen dostal k tomu, že když se okolí postupně s časem diskrétně zmenšuje, získá tím daleko lepší výsledky. Potom průměr okolí vyjadřuje hodnotu $2\lambda(f)$. Stejně jako zmenšování okolí, je důležité měnit i hodnotu parametru učení. Ten se používá pro ovlivňování rychlosti učení sítě. Jak už bylo zmíněno v předchozím textu, tento parametr se nachází v intervalu $0 \leq \eta(t) \leq 1$. Na samém počátku se dává tento parametr blízký hodnotě jedné, aby se síť velmi rychle roztáhla a měla pak tedy velmi velkou oblast definičního oboru hodnot vstupujících vektorů. Při následném zmenšování parametru učení síti umožní jemně doučit váhy vstupujících neuronů. Vhodný způsob změny parametru učení a okolí je znázorněn na obrázku 10 [5].



Obrázek 10: Příklad doporučené volby poklesu parametru učení a okolí, viz [5]

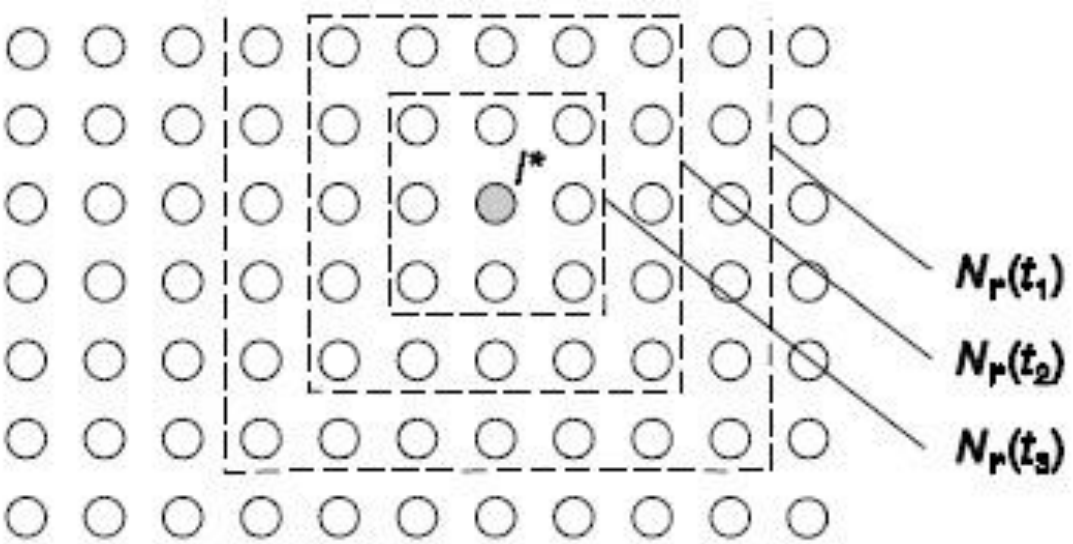
Na obrázku 10 se rozlišují dvě fáze učícího procesu. První fáze se nazývá *fáze uspořádávání*, kde se okolí zmenšuje diskrétně s časem. Další fází je *fáze doladění*, kde v této fázi se sousední neurony ponechají v okolí a probíhá už pouze jejich doladění [5].

Důležitými otázkami jsou:

- jak velké volit na začátku okolí,
- k jak velkému okolí se ke konci dostat,
- jak měnit v průběhu jednotlivé váhy,
- jak zmenšovat parametr učení,
- jaký tvar vybrat pro okolí a jakým způsobem ho zmenšovat.

Žádná daná pravidla k těmto otázkám nejsou, aby bylo možné optimálně definovat parametry. Kohonen tvrdí, že malé odchylky těchto parametrů nepředstavují nějaký razantní vliv na učení sítě. Existují však doporučené zásady, podle kterých by se učení mělo řídit. Některé z těchto zásad byly už řečeny v algoritmu učení, ale je bylo by dobré si je zopakovat a detailněji popsat [5].

Co se týče volby okolí, tak se nastavuje co největší, aby pokrývalo veškeré neurony. Může ale nastat jistá situace pro neurony ležící v blízkosti hrany mřížky. Na obrázku 11 je znázorněna situace, kde poloměr okolí má větší velikost a zasahuje tedy mimo mřížku. Tedy neurony v blízkosti okraje nemají symetrické okolí [5].



Obrázek 11: Neuron ležící blízko hranice, viz [5]

Okolí, na kterém se zmenšování zastaví, obvykle zahrnuje pouze vítězný neuron, ale nemusí to být vždy, jelikož můžeme dojít k podmínce, která celý učící proces ukončí. Změna vah v okolí neuronu je dána právě pomocí adaptační funkce, kde dvě z nich jsou znázorněny na obrázku 8a,b. Způsob, jakým zmenšovat okolí je celá řada. Kohonen tvrdí, že postačí zmenšovat okolí lineárně. Co se týče tvaru okolí, tak to je možné si zvolit jakkoli. Ideální řešení by bylo, kdyby se použilo kruhové okolí, ale jelikož je to v praxi složitější realizovat, používá se čtvercové okolí. Jak rychle zmenšovat okolí neuronu, je přímo závislé na tom, jak dlouho trvá učení. Je tedy nutné, při předložení dalšího učícího vzoru zmenšit okolí, aby po předání posledního, se dosáhlo předem stanovené velikosti [5].

Jaká se použije funkce pro změnu parametru učení, není až tak moc podstatné. Ovšem měla by to být funkce, která postupně klesá od čísla přibližující se jedné a s klesajícími hodnotami v řádech setin. Je celá řada funkcí, které se dají použít, například lineární lomená funkce. Lze také určit koeficient učení pomocí času [5]. Příklady některých funkcí:

- Exponenciální: $\eta(t) = 1 - e^{-\beta/t}$
- Lineární: $\eta(t+1) = \eta(t) - \Delta\eta$
- Hyperbolická: $\eta(t) = 1/t$

Na počtu iterací moc nezáleží. Podle Kohonena by to mělo být zhruba 500 krát více, než je neuronů v síti. Většinou tento počet bývá mezi 10 000 a 100 000. Dokud má parametr učení velkou hodnotu, tak v průběhu fáze uspořádání se síť doučuje a zařazuje její váhové vektory, které se při vyšším čase doladí. Bývá také dobré ponechat fázi doladění víc času, než na fázi předchozí [5].

3.3 Vybavování sítě

Ve fázi vybavování se neurony vkládají do již naučené sítě. Během této fáze není potřeba inicializace, protože síť je už naučena To znamená, že i váhy už jsou připravené a naučené. Není zde už ani potřeba okolí a koeficient učení, protože sít je připravena pro vložení nových vzorů. Vybavování sítě probíhá v těchto krocích:

1. Předání nového vzoru:
    - V tomto kroku se se do sítě předkládají nové vzory, které už budou vloženy na odpovídající místo.
2. Výpočet vzdálenosti:
    - Tento krok je také podobný jako u učení sítě. Vypočítají se vzdálenosti mezi novým vzorem a neurony sítě, pomocí euklidovské vzdálenosti. Následně se

vezme ten neuron, který má tuto vzdálenost nejmenší. Neuron s nejmenší vzdáleností od nového vzoru je vítězný a vzor se do něho zařadí.

3. Přidání dalšího vzoru:
    - Poslední část je pouze rozhodování, zda se má vložit další nový vzor. Pokud nějaké vzory ještě jsou, jde se zpět na krok 1. Při předložení všech nových vzorů vybavování sítě končí [4].

3.4 Doučení Kohonenovy mapy pomocí algoritmů LVQ

Doposud se Kohonenovy mapy požívaly s učením bez učitele. V tomto textu bude vysvětleno, jak lze danou sít využít k vyřešení problému *klasifikace* dat do určitých kategorií. Bude uveden postup, pomocí kterého se označí vstupní neurony sítě kategoriemi. Pro tento postup se používají tři algoritmy *učící vektorové kvantizace*. Pomocí těchto algoritmů se síť doučí [2].

Data jsou ve tvaru $\{(x^{(t)}, d^{(t)}); t = 1, \dots, k\}$, kde pro $x^{(t)}$ náleží hodnoty reálných čísel a $d^{(t)}$ náleží hodnotám$\{C_1, \dots, C_q\}$. Veškeré vstupní vektory $x^{(t)}$, mají přiřazenou jednu kategorii C_k. Nyní učící proces má tři kroky:

1. Naučení sítě bez učitele (popsáno v kapitole 3.2).
2. Vstupní neurony se označí kategoriemi.
3. Použití jednoho s doučovacích algoritmů LVQ.

První krok je tedy naučit Kohonenovy mapy normálním učícím algoritmem, který rozprostře neurony do vstupního prostoru tak, aby aproximovaly hustotou pravděpodobnosti vzorů. Zatím zde nebyly použity výstupy $d^{(t)}$ z tréninkové množiny. Ten se použije až dalším kroku učení, kde se postupně prochází tréninková množina a u všech vzorů se najde, je mu nejbližší neuron. U tohoto neuronu se pak zapíše, ke které kategorii vzor přísluší. Tímto způsobem se při procházení tréninkovými daty u výstupních neuronů vytvoří tabulka, v které je uvedeno, kolik tento neuron má kategorií. Následně se vybere pro neuron pouze jedna kategorie, v které byl reprezentován nejčastěji. Rozdělí se tedy neurony do jednotlivých oddílů, které představují právě jednotlivé kategorie. Poté zbývá pouze využít jeden z algoritmů LVQ pro doladění [2].

3.4.1 LVQ1

Tento algoritmus vylepšuje klasifikaci tím, že posune neuron směrem ke vstupu a naopak špatnou klasifikaci posune dále od vstupu. Posouvá se pokaždé pouze jeden výstupní neuron,

a to ten, který vyhrál v soutěžení. Posunuje se pouze malá část vzdálenosti neuronu od příchozího vzoru [2].

Prvně se vypočítá vítězný neuron s nejmenší vzdáleností od vstupního vzoru a poté se u něho změní váhy. Pro správně klasifikovaný vstup váhy přepočtem podle vzorce:

$$w(t) = w(t-1) + \eta\big(x(t) - w(t-1)\big). \tag{10}$$

Pro špatně klasifikovaný vstup váhy určíme podle vztahu:

$$w(t) = w(t-1) + \eta\big(x(t) - w(t-1)\big). \tag{11}$$

Parametr učení, by se měl nastavit na hodnotu 0,01 až 0,02 a v průběhu mnoha iterací se postupně dostat na hodnotu nula. Vytvoří se tedy hranice mezi shluky. Tato hranice aproximuje s *bayerskou rozhodovací hranicí*. Rozdíl je ten, že rozhodovací hranice vznikne uprostřed spojnice, mezi neurony odlišných kategorií. [2].

3.4.2 LVQ2

Tento algoritmu v jeden okamžik přesouvá hned dva neurony. Podmínkou je, že jeden neuron musí být klasifikován správně a ten druhý špatně. Dále nesmějí být neurony moc blízko vstupu. Neurony volíme tak, aby vstup ležel mezi nimi, tedy v tzv. *okně*. Šířka toho okna je zhruba deset až třicet procent vzdálenosti neuronů. Po splnění výše zmíněných podmínek se provede změna vah (uvedené v 3.4.1) [5].

V praxi se zjistilo, že na začátku tento algoritmus vylepšuje pozici rozhodovací hranice. Bohužel při větším počtu iterací nastane, že se neurony, které jsou správně klasifikované, postupně pak vzdalují, místo toho aby se dále přibližovaly. Z toho důvodu se používá menší počet iterací (cca 10 000) [5].

3.4.3 LVQ3

Jelikož verze LVQ2 není příliš stabilní, tak se algoritmus LVQ3 vylepšil přidáním pravidla ε. Díky tomuto pravidlu se správně klasifikované neurony ještě navíc posouvají směrem k tréninkovému vzoru. Použije se tedy změna vah popsána v 3.4.1, a navíc se provede u správně klasifikovaného neuronu posunutí, podle rovnice:

$$w(t) = w(t-1) + \varepsilon\eta\big(x(t) - w(t-1)\big). \tag{12}$$

Velikost ε je dána podle velikosti okna. Čím menší velikost okna, tím menší ε. Volí se v rozmezí od 0,1 do 0,5 [5].

3.5 Využití Kohonenových map

Využití Kohonenových map je celá řada. V následujícím textu bude popsáno pár z nich.

1. Formování hierarchických reprezentací:

   Kohonenovy samoorganizační mapy umožňují mezi prvky topologicky zobrazit hierarchické vztahy. Prvky musejí být popsány svými souřadnicemi. Závislosti mezi nimi lze znázornit obyčejnou metodou, kde se získá nejmenší strom. Pomocí Kohenenovy mapy se z těchto hodnot získá mapa [5].

2. Rozpoznávání řeči:

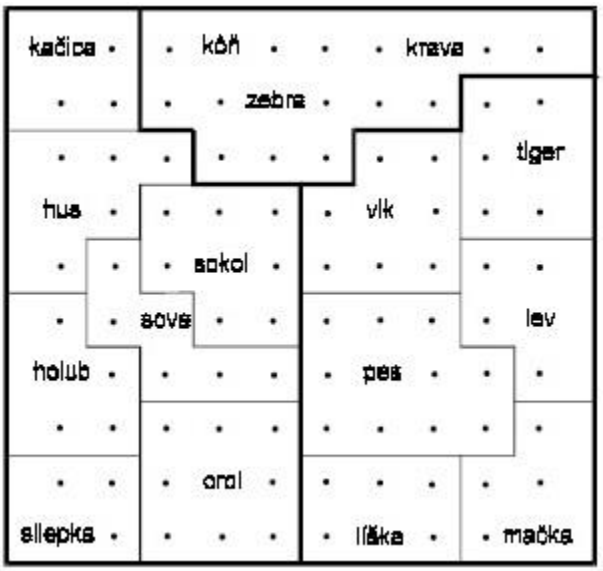
   Při rozpoznávání řeči se promění akustický řečový signál na sekvenci hlásek, této změně se říká *akustický přepis* (například slovo ve slovenštině dievča, se převede na ď-j-e-u-č-a). Pro převod slouží již naučené Kohonenovy mapy, kterým se říká *fonémová mapa*. Ta dokáže svojí odpovědí nalézt odezvu na současný vstup, který odpovídá právě hlásce. Je nutné v první řadě řečový signál nějakým způsobem předpřipravit, jinak by nebylo možné mapu natrénovat [5].

3. Topografické mapy abstraktních dat:

   U tohoto typu dat je často pokládaná otázka, jakým způsobem zjistit a následně zobrazit vztahy mezi nimi. U fyzických dat to není žádný složitý problém, jelikož zastoupení samo o sobě vypovídá vzájemným vztahům podobnosti. Lze si to představit na příkladu blízkosti, která se dá vyjádřit pomocí souřadnic vektorů. Jiné je to u symbolů, protože neexistuje spojitost mezi kódem a jeho významem. Příkladem můžou být slova v přirozeném jazyce, kde není vztah s psanou formou těchto slov. Protože tyto souvislosti nelze nalézt z jejich kódové prezentace, je nutné předložit je ve špatnosti s kontextem, ve kterém se nalézají. Pro lepší pochopení funkce, bude dobé si zjednodušený postup ukázat na příkladu [5].

   Na obrázku 12 byl kontext veškerých symbolů reprezentován vektorem binárních atributů. Pokud se vyskytoval, označil se číslem jedna, pokud ne, tak nulou. Také se popsal vzhled daného zvířete - kolik má nohou, co za srst má apod. Nakonec se řešilo, co dané zvíře baví, jestli běhá, skáče atd. Poté se vytvořil vektor z daných atributů pro jednotlivá zvířata. Kódy zvířat se vytvořily tak, aby na sobě neměly žádné sdělení o vzájemné podobnosti mezi

znaky. Tento vektor se nastavil tak, že měl pouze nulové hodnoty, kromě jedné, která říkala, jaké je pořadí zvířete. Tyto dva zmíněné vektory se spojily v jeden několikarozměrný vektor popisující každé zvíře. Kolikarozměrný vektor určuje také, kolik vlastností se vybere (v tomto případě celkem 16). Pro menší vliv symbolů se zmenšila hodnota kódové části (vzalo se pouze 0,2 hodnoty). Během učení se předložilo 2 000 učících vzorů (z 16 prvkové množiny), které byly náhodně vybrány a předloženy. Postup pro realizování návěstí byl proveden ze spojeného vektoru s absencí vektoru obsahující kódy zvířat. Výsledek je tedy možné vidět na obrázku 12 [5].



Obrázek 12: Příklad topografické mapy abstraktních dat, viz [5]

4 STATISTICKÉ METODY

V této kapitole budou prezentovány statistické metody pro pozdější analýzu léčebných metod.

4.1 Poměr šancí

První statistickou metodou je poměr šancí. Tato metoda umožňuje porovnat výskyt sledovaného jevu ve dvou různých skupinách, kde porovnává jejich šance. Šance je ve své podstatě poměr dvou pravděpodobností [7].

Pro poměr šancí se použije čtyřpolní kontingenční tabulka:

Tabulka 1: Čtyřpolní kontigenční tabulka

	1	2	Σ
1	n_{11}	n_{12}	$n_{1.}$
2	n_{21}	n_{22}	$n_{2.}$
Σ	$n_{.1}$	$n_{.2}$	n

Poměr šancí má následující rovnici:

$$OR = \frac{n_{11}n_{22}}{n_{12}n_{21}} \tag{13}$$

Za předpokladu nulové hypotézy o nezávislosti ($p_{ij} = p_{i.}p_{.j}$) je veličina $\frac{p_{11}p_{22}}{p_{12}p_{21}} = 1$. Pro testování nulové hypotézy se používá testovací statistika:

$$u = \frac{ln\, b - ln\frac{p_{11}p_{22}}{p_{12}p_{21}}}{\sqrt{\frac{1}{n_{11}} + \frac{1}{n_{12}} + \frac{1}{n_{21}} + \frac{1}{n_{22}}}} \sim N(0,1) \tag{14}$$

Nulová hypotéza se zamítá, pokud:

$$|u| < u_1(\alpha),$$

kde při $\alpha = 0{,}05$ je $u_1(\alpha) = 1{,}96$ [7].

Pokud se nulová hypotéza nezamítne, znamená to, že data jsou statisticky odlišná.

4.2 Funkce přežití

Funkce přežití popisuje dobu do určité události. Graf této funkce je základní nástroj pro názorné zobrazení analýzy přežití. Je to ve své podstatě pravděpodobnost, že čas, kdy měla nastat událost, překročí čas t [7].

Funkce má tvar:

$$S(t) = P(T > t). \tag{15}$$

Tato funkce je doplňkem distribuční funkce, protože

$$F(t) = P(T \leq t), \tag{16}$$

Neboli

$$S(t) = 1 - F(t). \tag{17}$$

Z toho vyplývá, že funkce přežití je nerostoucí a v čase $t = 0$ má nastavenou hodnotu na 1. Postupem času tato hodnota se snižuje až na 0. V následujícím postupu se uvažuje, že T je diskrétní veličina [7].

Předpokládá se, že T nabývá hodnot t_j, kde $j = 1, 2, 3, \dots, n$ s pravděpodobnostní funkcí:

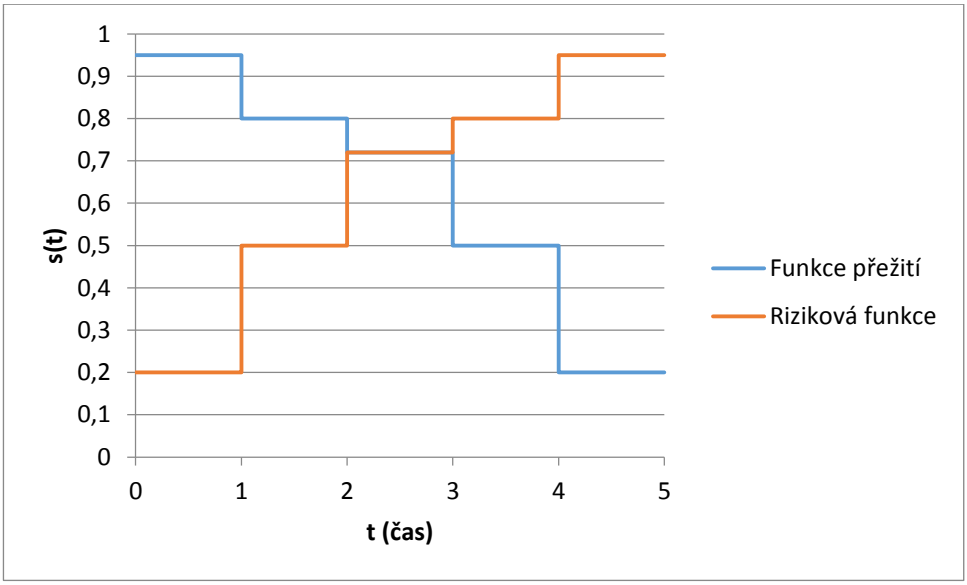
$$p(t_j) = P(T = t_j), \tag{18}$$

kde musí platit, že $x_1 < x_2 < \cdots < x_n$. Funkce přežití diskrétní veličiny je pak dána vztahem:

$$s(t) = P(T > t_j) = \sum_{t_j > t} p(t_j). \tag{19}$$

Jelikož je T diskrétní veličina, proto je funkce přežití klesající schodovitá funkce (Obrázek13) [7].

Opakem funkce přežití je riziková funkce. Tato funkce sleduje, jestli daná událost v určitém čase nastala (Obrázek 13) [7].



Obrázek 13: Graf funkce přežití diskrétní veličiny (vlastní)

5 APLIKACE

Celá aplikace je psána v programovacím jazyku JAVA. Před spuštěním aplikace se načtou data o pacientech a uloží se do aplikace. Data se načítají ze souboru *csv*. Aplikace umožňuje také načtená data upravovat v hlavním okně a ukládat je zpět do souboru. Na základě uložených dat v aplikaci se tvoří Kohonenovy mapy. Pomocí těchto map se následně počítá pravděpodobnost nově příchozího pacienta na přežití v jednotlivých léčbách.

5.1 Metody

Veškeré metody se vždy zavolají po provedení nějaké akce (zpravidla to bývá použití tlačítka). Dále v textu je popsáno pár metod. Z důvodu velikosti kódu používaných metod je tento kód k nahlédnutí pouze ve zdrojovém kódu aplikace na CD, kde jsou patřičně popsány.

5.1.1 Příklad použitých metod

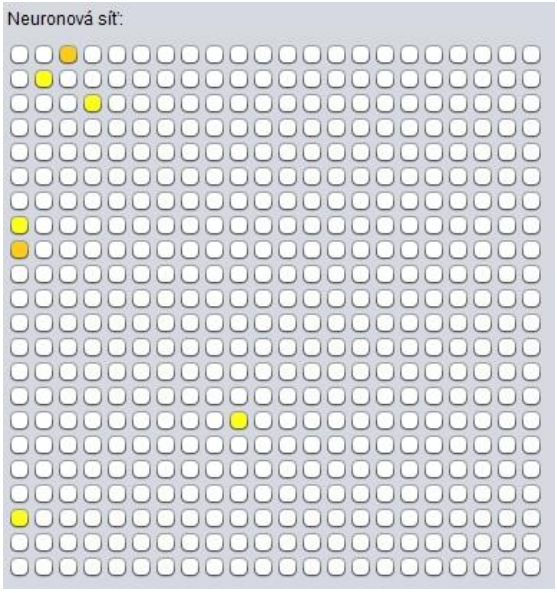
Metoda *standardizaceDat(double[][] pole, int pocet)* počítá standardizované hodnoty pacientů pro tvorbu Kohonenovy mapy. Do této metody vstupuje pole pacientů, u kterých se bude provádět úprava dat a počet těchto pacientů. Po vypočítání nových hodnot vrací metoda nové pole standardizovaných dat. Veškeré výpočty se pak ukládají do *txt* souboru. Ukládání používají všechny metody, co počítají nové hodnoty

Metoda *uceniSite(int pocetInteraci, int velikostOkoli, double parametrUceni, int pocet, double[][] pole, Neuron[][] poleNeuronu)* připravuje neuronovou síť pro vložení pacientů. Tato metoda přijímá parametry sítě, počet pacientů, pole standardizovaných hodnot a nenaučené pole neuronů. Učení sítě probíhá v cyklu, kde počet opakování určuje předaný parametr *pocetIteraci*. V první části tohoto cyklu se počítá nejmenší vzdálenost učících vzorů od jednotlivých neuronů. Poté co se vybere nejmenší vzdálenost se vítěznému neuronu a jeho okolí (velikost okolí podle parametru velikostOkoli) změní váhy. Po skončení každého cyklu se aktualizuje parametr učení a velikost okolí. Po skončení cyklu metoda vrací naučené pole neuronů.

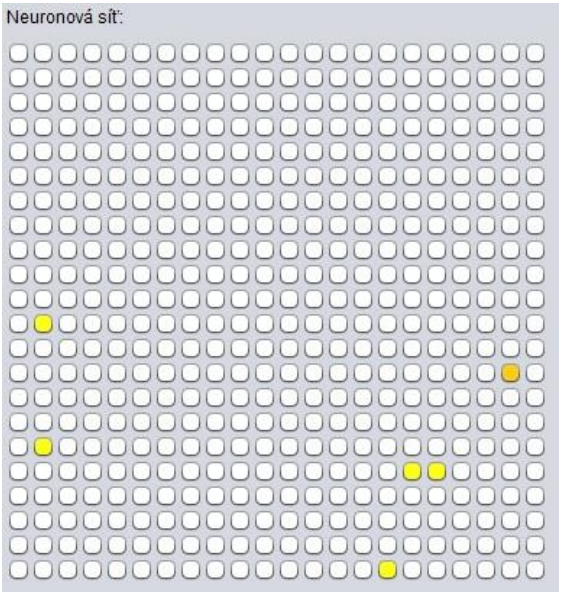
Metoda *kohogenPrirazeniPacientu(int pocet, double[][] pole, Neuron[][] poleNeuronu)* vkládá nové pacienty do neuronové sítě. V této metodě se projdou v cyklu pacienti a ty se podle nejmenší vzdálenosti postupně vkládají na příslušné neurony podle nejmenší vzdálenosti.

5.2 Vytváření Kohonenových map

V předešlém textu byly popsány základní metody na tvorbu Kohonenových map. Metody se volají vždy při použití tlačítka **Vytvořit Kohonenovy mapy** a při přidávání pacienta, kterému se nepřiřadila žádná léčba. Při stejně zadaných parametrech učení sítě se vytvoří pokaždé jinak uspořádaná mapa (Obrázek 14, 15). To je uskutečněno díky tomu, že se pro každé vytvoření mapy generují nové váhy neuronů.



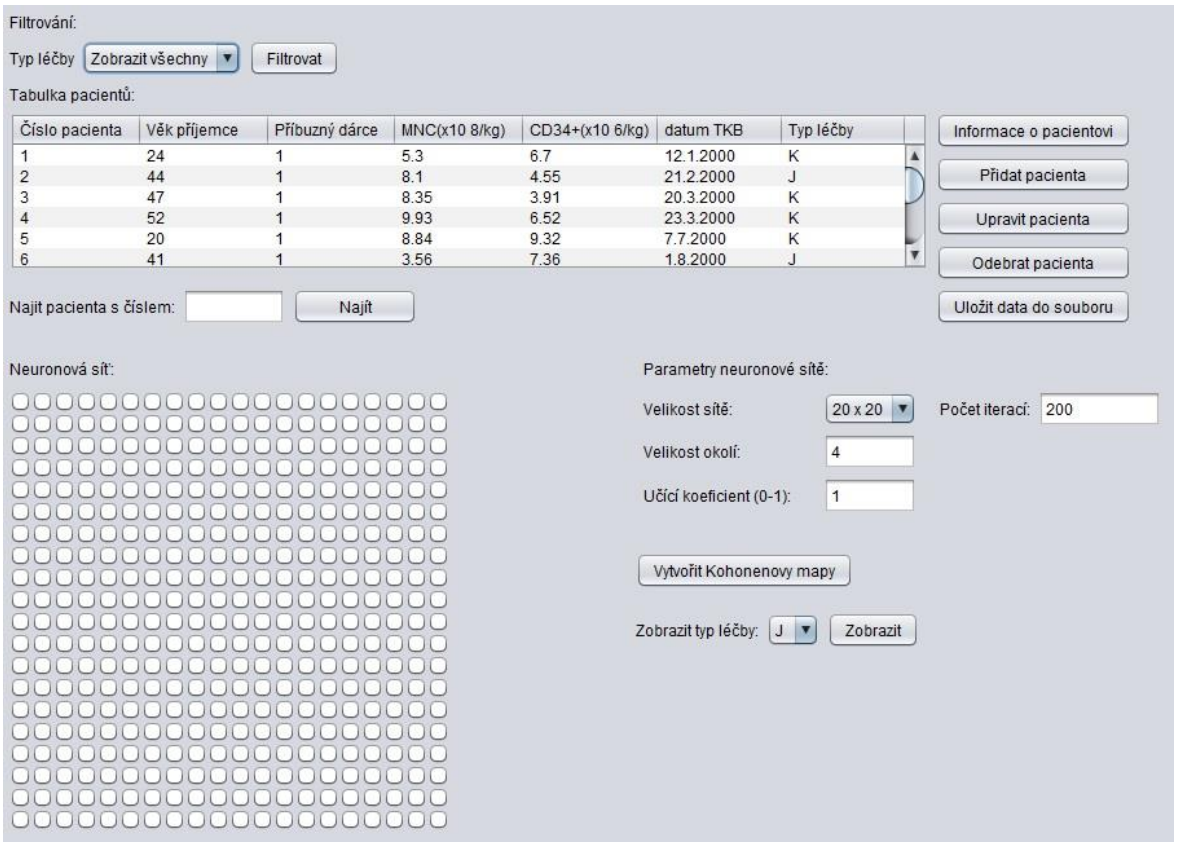
Obrázek 14: Zobrazená 1. Kohonenova mapa v aplikaci (vlastní)



Obrázek 15: Zobrazená 2. Kohonenova mapa v aplikaci (vlastní)

6 UŽIVATELSKÁ PŘÍRUČKA

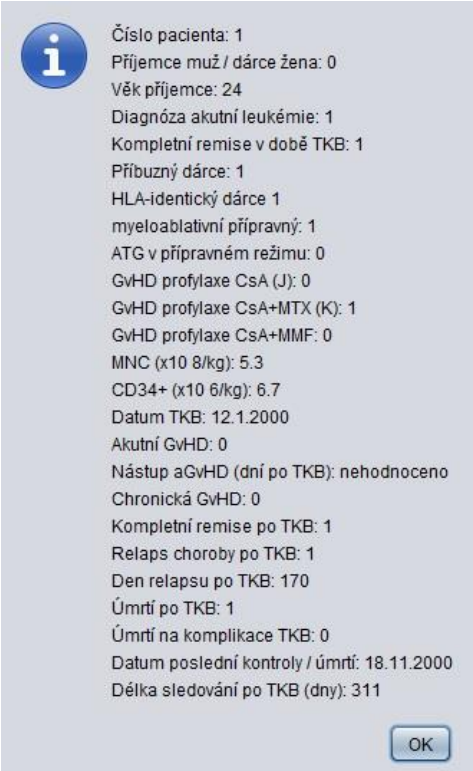
Vytvořená aplikace umožňuje vytvářet Kohonenovy mapy z načtených dat o pacientech. V těchto datech má každý pacient údaje o průběhu léčby a jejím typu. Při vytváření Kohonenovy mapy si aplikace sama vybírá potřebné údaje. Data se mohou upravovat, přidávat a mazat. Po spuštění aplikace se objeví hlavní okno (Obrázek 13), které obsahuje veškeré prvky pro ovládání aplikace. Seznam léčených pacientů je zobrazen v tabulce pacientů (Obrázek 16).



Obrázek 16: Hlavní menu aplikace (vlastní)

6.1 Detailní informace o pacientech

Informací o průběhu léčby pacienta je velké množství. Z tohoto důvodu se všechny nezobrazují v tabulce pacientů. Pro zobrazení veškerých údajů o pacientovi se nejdříve musí vybrat pacient z tabulky pacientů, u kterého chceme údaje zobrazit. Po vybrání pacienta se použije tlačítko **Informace o pacientovi** (Obrázek 16). Následně se zobrazí dialog se všemi informacemi (Obrázek 17).



Obrázek 17: Detailní informace o pacientovi (vlastní)

6.2 Filtrování pacientů

Obrázek 18: Filtrováni podle typu léčby (vlastní)

Aplikace umožňuje filtrovat pacienty na základě typu léčby (Obrázek 18). V první řadě se vybere typ, podle kterého se bude filtrovat. Na výběr jsou tyto možnosti:

- Zobrazit všechny.
- J.
- K.
- L.

Po výběru typu léčby se použije **Filtrovat**. Následně se v tabulce pacientů objeví pouze pacienti, kteří jsou léčeni daným typem léčby. Příklad vyfiltrovaných pacientů s typem léčby J, je zobrazen na obrázku 19. Pro následné zobrazení všech pacientů stačí zvolit možnost pro typ léčby zobrazit všechny a použít **Filtrovat**.

Tabulka pacientů:

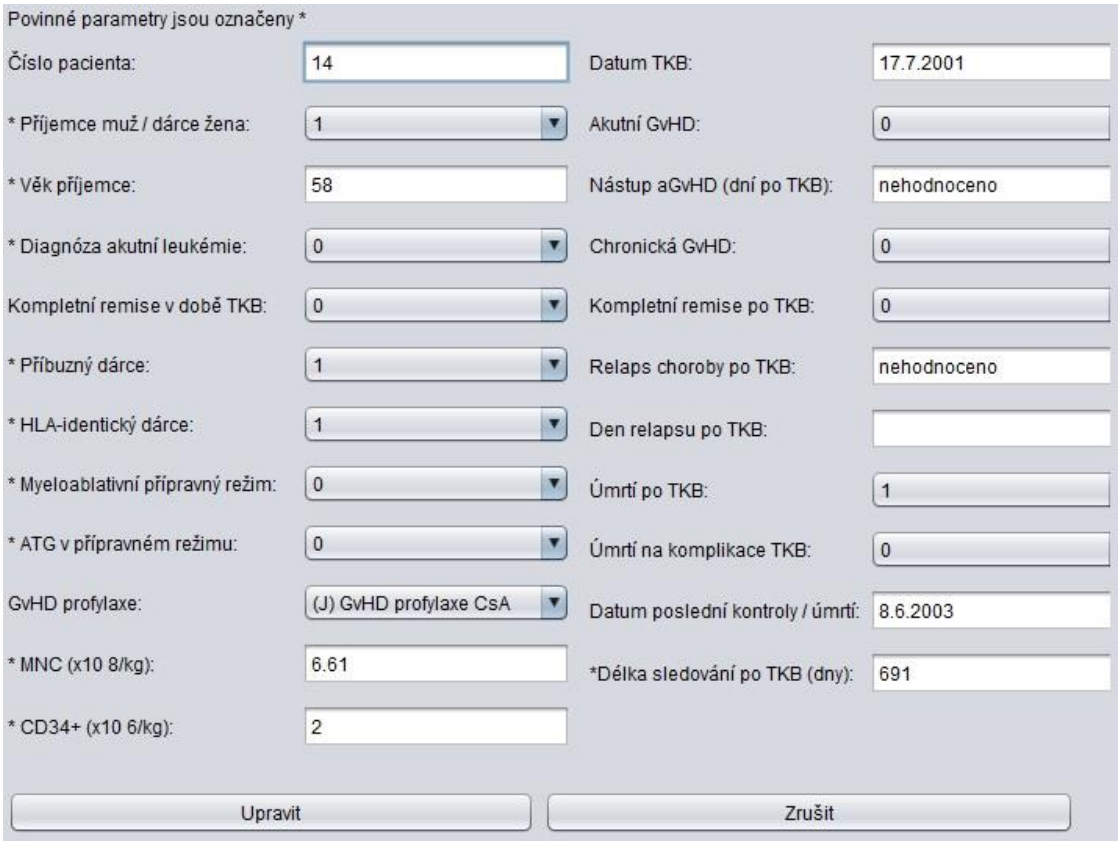
Číslo pacienta	Věk příjemce	Příbuzný dárce	MNC(x10 8/kg)	CD34+(x10 6/kg)	datum TKB	Typ léčby
2	44	1	8.1	4.55	21.2.2000	J
6	41	1	3.56	7.36	1.8.2000	J
14	58	1	6.61	2	17.7.2001	J
18	50	1	8.37	3.92	22.1.2002	J
25	48	1	7.67	5.8	6.6.2002	J
26	49	1	10.54	6.34	3.7.2002	J

Obrázek 19: Zobrazení pacientů léčených typem J (vlastní)

Další možností jak filtrovat pacienty je najít přímo jednoho konkrétního. Stačí zadat do kolonky **Najít pacienta s číslem** (obrázek 20) číslo pacienta a potvrdit pomocí **Najít**. Pokud existuje pacient s tímto číslem, zobrazí se jako jediný v tabulce pacientů.

Obrázek 20: Hledání pacienta pomocí čísla (vlastní)

6.3 Modifikace pacientů



Obrázek 21: Okno pro nastavení hodnot pacientovi (vlastní)

V okně pro nastavení hodnot pacientovi (Obrázek 21) je nutné zadat povinné parametry, protože na jejich základě se tvoří Kohonenovy mapy. Okno se zobrazuje při přidávání a upravování pacientů.

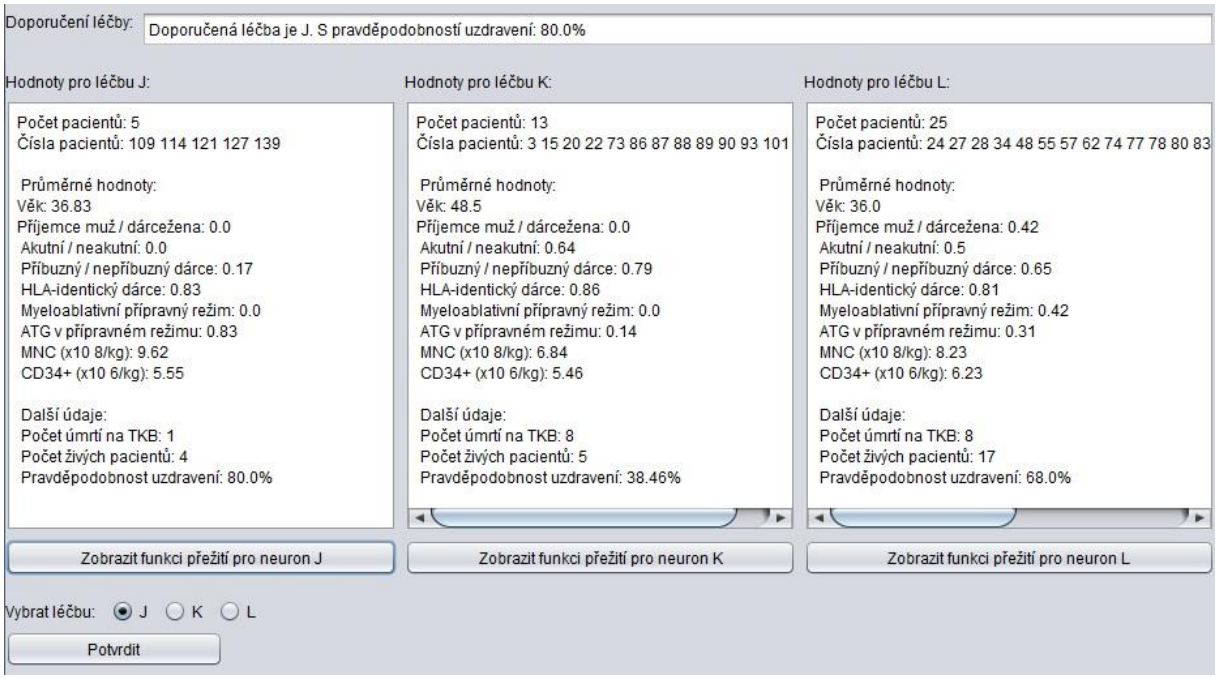
6.3.1 Přidání pacienta

Přidání pacienta se provádí tak, že se klikne na **Přidat** v hlavním okně a poté se zobrazí nové okno pro zadání hodnot (Obrázek 21), pouze místo tlačítka **Upravit** je zde tlačítko **Přidat**). V okně se zadají hodnoty pacienta a zvolí se typ léčby.

Zvolení léčby (GvHD profylaxe na obrázku 21) obsahuje následující položky:

- (J) GvHD profylaxe CsA.
- (K) GvHD profylaxe CsA+MTX.
- (L) GvHD profylaxe CsA+MMF.
- Žádná.

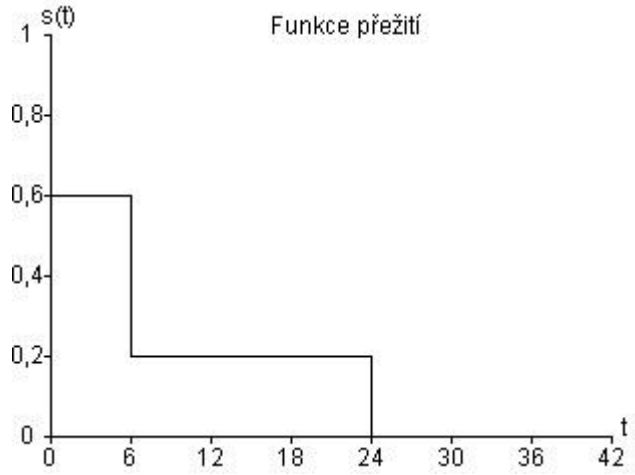
Pokud se vybere jeden z typů léčby J, K, L, stačí pouze použít **Přidat** a pacient se do aplikace vloží. Zvolením typu léčby žádná a použitím **Přidat** se objeví nové okno, které doporučí vhodný typ léčby (Obrázek 22).



Obrázek 22: Okno s doporučenou léčbou (vlastní)

Při spouštění okna se v pozadí aplikace vytvoří Kohonenovy mapy pro jednotlivé léčby a nový pacient se přidá do každé z nich. Díky tomu je možné vidět průměrné hodnoty z jednotlivých léčebných metod, do kterých se pacient přiřadil. Kromě průměrných hodnot jsou zde i údaje o počtu pacientů, čísla pacientů, počet úmrtí na nemoc, počet živých pacientů

a jakou pravděpodobnost má pacient na uzdravení. Pro léčby je možné si nechat zobrazit graf funkce přežití pomocí **Zobrazit funkci přežití pro J** (Obrázek 23). Není ovšem nutné vybrat doporučenou léčbu. Veškeré údaje jsou pouze informativní. Po vybrání léčby ať už doporučené, nebo jiné, stačí daný typ léčby zvolit, stisknout **Potvrdit** a pacient se přidá.



Obrázek 23: Graf funkce přežití v aplikaci (vlastní)

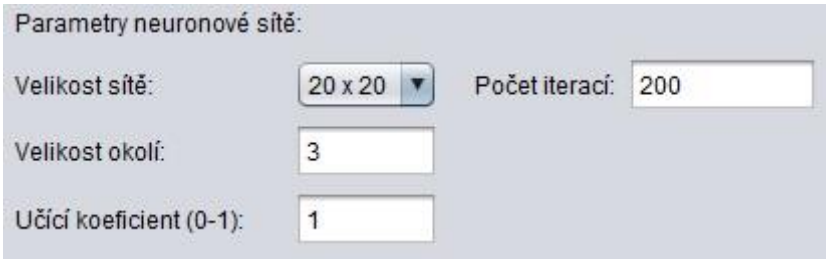
6.3.2 Úprava pacienta

Aby bylo možné upravovat pacienta, tak se nejdříve musí nějaký vybrat z tabulky pacientů. Pokud je vybrán pacient, který se má upravovat, stačí stisknout **Upravit pacienta** v hlavním okně (Obrázek 16). Stejně jako u přidávání pacienta se zobrazí nové okno pro nastavení hodnot (Obrázek 21). V okně jsou předem nastaveny původní hodnoty zvoleného pacienta. Pro dokončení úprav stačí pouze potvrdit pomocí **Upravit**.

6.3.3 Smazání pacienta

V první řadě je nutné z tabulky pacientů vybrat pacienta, který je určen pro smazání. Vybraného pacienta je možné odstranit pomocí **Odebrat pacienta** v hlavním okně.

6.4 Nastavení parametrů neuronové sítě



Obrázek 24: Nastavení parametrů neuronové sítě (vlastní)

Parametry pro tvorbu Kohonenovy mapy se nastavují v pravé dolní části (Obrázek 24) hlavního okna. Nejprve se vybere velikost sítě, tedy kolik bude síť obsahovat neuronů. Možnosti velikosti sítě jsou:

- 20 x 20.
- 21 x 21.
- 22 x 22.
- 23 x 23.
- 24 x 24.
- 25 x 25.

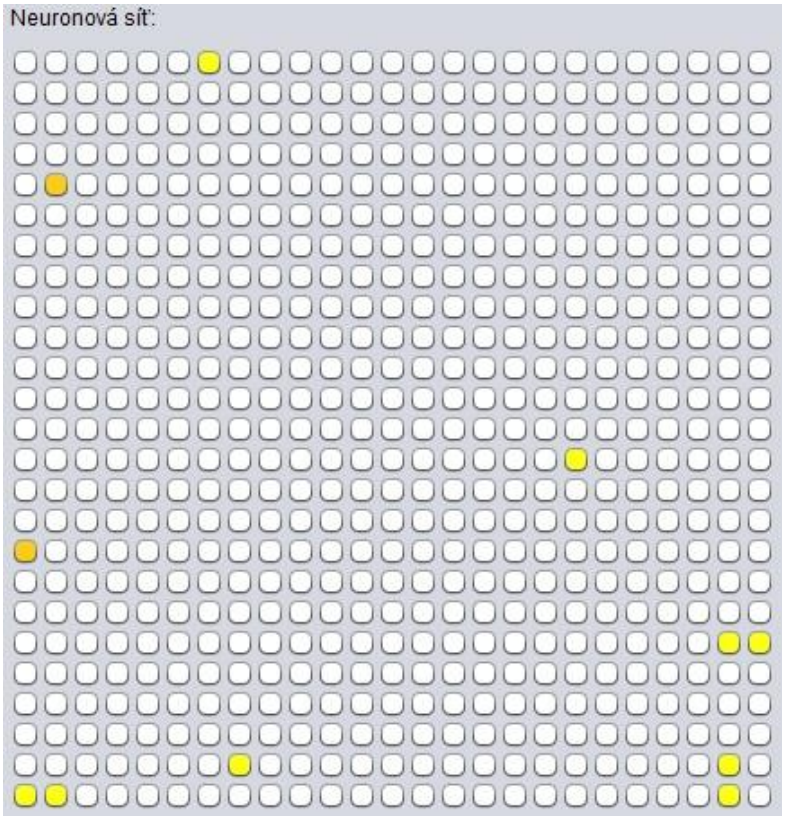
Velikost okolí udává, jak velkou oblast vkládaný vzor ovlivní. Hodnota je rovna poloměru celkové oblasti. Musí to být celé číslo a nesmí být záporné. Zadávaná velikost nemá žádné horní omezení, proto je možné vložit například hodnotu 1 000. Použitím takové hodnoty by se docílilo toho, že by každý vkládaný vzor ovlivňoval celou vytvořenou mapu. Okolí klesá lineárně a to vždy o hodnotu 1. Hodnotu pro okolí je tedy potřeba zvolit vhodně pro vybranou velikost sítě. Doporučená hodnota okolí je 3 pro nejmenší okolí. Větším sítím (např. 25 x 25) je možné zvolit větší velikosti velikost okolí (např. 4, 5).

Učící koeficient ovlivňuje rychlost učení sítě a nabývá hodnot od 0 do 1. Z počátku je doporučeno nastavit ho na hodnotu blízké jedné.

Poslední parametrem pro nastavení sítě je počet iterací. Ten se volí právě podle nastavení předchozích parametrů. Pokud se nastaví malé okolí sítě a malý koeficient učení, tak bude potřeba daleko více iterací, než při učícím koeficientu nastaveným na hodnotu 1, a větším okolí.

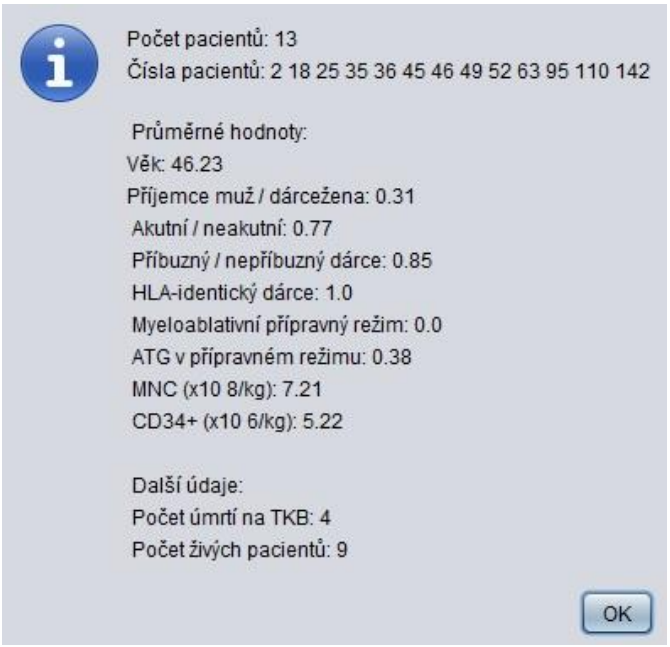
6.5 Vytvoření a zobrazení Kohonenovy mapy

Pokud jsou nastaveny správně parametry neuronové sítě (6.4), tak mapa se vytvoří pomocí **Vytvořit Kohonenovy mapy**. Mapy se vytvoří a následně zobrazí pomocí pole tlačítek (Obrázek 25). Neurony jsou barevně rozlišeny, podle toho kolik obsahují pacientů. Nejčetnější neurony jsou označeny oranžovou barvou a ostatní žlutou.



Obrázek 25: Zobrazení Kohonenovy mapy v aplikaci (vlastní)

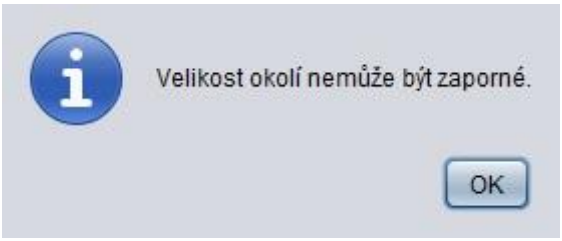
K získání informací o neuronu obsahující pacienty stačí kliknout na barevný neuron a informace se zobrazí v dialogovém okně (Obrázek 26). V něm jsou obsaženy informace o počtu pacientů, čísla pacientů, průměrné hodnoty pacientů, počet živých pacientů a počet těch, kteří zemřeli.



Obrázek 26: Detailní informace o neuronu (vlastní)

6.6 Ošetření špatně zadaných údajů

Při nesprávném zadání hodnot se zobrazí chybová hláška, v které je napsáno, jaký parametr byl špatně zadán a proč (Obrázek 27).



Obrázek 27: Chybová hláška (vlastní)

7 VÝSLEDKY ZPRACOVÁNÍ REÁLNÝCH DAT

Před zpracováním statistické analýzy je důležité si něco málo říci o leukémii.

7.1 Leukémie

Leukémie je rakovinové onemocnění bílých krvinek. Tyto bílé krvinky se vytvářejí v kostní dřeni. Při leukémii dochází k vytváření velkého množství bílých krvinek, které bývají nezralé, a tedy nedokáží provádět svoji funkci tak, jak by u zdravého jedince měly. Jelikož se jich vytváří mnoho, vzniká problém s tvorbou normálních krvinek, protože ty špatné se tvoří na úkor normálních. Při této nemoci dochází k poškození běžných funkcí krve, jako jsou imunita, krevní srážení a přenos kyslíku. Jedinec trpící touto nemocí trpí únavou, tachykardií způsobenou chudokrevností, časté bolesti hlavy a kloubů, má zvýšenou šanci k infekci a krvácení [6].

V zásadě rozlišujeme dva druhy leukémie, akutní a chronickou. Akutní leukémie má velmi rychlý průběh, a pokud se neposkytne včas léčba, vede k brzké smrti. Velké množství vytvořených bílých krvinek se ukládá do orgánů (např. mozkových blan) a tyto orgány postupně odumírají. Akutní leukémie se dále dělí na *akutní lymfoblastickou leukémii*, která se vyskytuje hlavně u dětí, a *akutní myeloidní leukémii*, která je častěji v pozdějším věku. U těchto typů může docházet k zvětšení jater, horečce, otoku lymfatických uzlin a dalším problémům. Chronická leukémie má pomalejší spád a vyznačuje se většími stabilními úseky. Také se dělí dále na dva druhy. *Chronická lymfatická leukémie* je nejčastěji u lidí nad 45 let a *chronická myeloidní leukémie* se objevuje o něco dříve. Zde mohou být příznaky jako únava, hubnutí, ztráta chuti a další. Včasné objevení tohoto typu leukémie má velký vliv na léčbu [6].

Důležité je zmínit základní typy léčby nemoci, a to chemoterapii, ozařování a transplantaci kostní dřeně. Chemoterapie je stále nejvíce podstatnou léčbou, i přesto, že má dost nežádoucích účinků. Pacientovi jsou podávány léky tzv. *cytostatika*. Cytostatik je velké množství, které se různě kombinují a berou v různých fázích. Tyto léky účinkují na dělené buňky. Podstata spočívá v tom, že napadají špatné buňky a poškozují je. Nežádoucí vliv je, že bohužel poškozují i buňky zdravé, ale ty se dokáží zregenerovat v kratší době, než nádorové buňky. Nežádoucími účinky jsou zvracení, poškození buněk kostní dřeně, poškození močového měchýře, nebo záněty ústní dutiny. Tyto účinky nastávají právě kvůli poškozování zdravých buněk. Přesto, že se buňky regenerují, tak stále nefungují jako u zdravého člověka. Další léčbou je léčba pomocí ozařování. Pacienti jsou ozařováni pomocí rentgenového, nebo gama záření. Tato léčba účinkuje na zrychleně dělicí se nádorové buňky, které poškodí, nebo

je dokonce zničí. Při této léčbě jsou také poškozovány zdravé buňky, ale pokud se ozařování aplikuje ve vhodných dávkách, tak se zdravé buňky uzdraví. Nežádoucím účinkem terapie může být případný problém s kůží, nebo sliznicí. V poslední řadě je tu transplantace kostní dřeně. U této metody se nemocnému člověku se transplantují buňky. U transplantace je důležité, aby u příjemce i dárce byly hlavní tkáňové znaky totožné a aby bylo tělo zbaveno všech nádorových buněk. Existují tři druhy transplantace. První z nich je *alogenní*, kde kostní dřen daruje příbuzný, či nepříbuzný dárce. Druhy druh je *syngenní transplantace*, která se používá u jedno-vaječných dvojčat. Poslední druh je *autologní transplantace*, kde nemocný je sám svým dárcem. Při transplantaci mohou také nastat nepříznivé komplikace, pokud tělo příjemce nepřijme nové buňky. Před začátkem samotné transplantace se provádí přípravná léčba tzv. *profylaktická léčba*. U této nemoci je důležitá doba do *relapsu*. To je doba, kdy se člověku znovu vrátí nemoc [6].

7.2 Výběr dat

K dispozici jsou údaje o pacientech trpící touto nemocí. Z těchto údajů se vyberou pouze takové, podle kterých pacienti budou tvořit následně skupiny. Tyto údaje jsou uvedeny v tabulce 2 a popisují transplantaci a léčbu. Další informace jako datum nástupu leukémie, datum poslední kontroly, délka sledování a podobné údaje, nejsou pro vybrání léčby podstatné. Příklad pacientů a jejich hodnoty jsou zobrazeny v tabulce 2.

Tabulka 2: Hodnoty pacientů

	Pacient 1	Pacient 2
Příjemce muž / dárce žena	0	0
Věk příjemce	24	44
Diagnóza akutní leukémie	1	1
Příbuzný dárce	1	1
HLA-identický dárce	1	1
Myeloablativní přípravný režim	1	0
ATG v přípravném režimu	0	1
MNC (x10 8/kg)	5,03	8,01
CD34+ (x10 6/kg)	6,70	4,55
Úmrtí po TKB	1	1
Typ léčby	K	L

U pacientů byly použity tři léčebné metody profylaxe: J, K, L. Metodou J se léčilo 56 pacientů, metodou K 38 pacientů a metoda L se použila u 56 pacientů. Nově příchozí pacient se zařadí do všech tří léčebných metod. Následně se vytvoří Kohonenovy mapy pro dané typy léčby a nalezne se u každé neuron, do kterého by se nový pacient přidal. Pro ukázku se použijí hodnoty z následující tabulky:

Tabulka 3: Tabulka pacientů shluknutých v jednotlivých léčbách

	Neuron v léčbě J	Neuron v léčbě K	Neuron v léčbě L
Nezemřeli	3	12	13
Zemřeli	8	20	12
Celkem	11	32	25

Pro zjištění pravděpodobnosti uzdravení pacienta se dá použít jednoduchý poměr nezemřelých pacientů a celkového počtu pacientů. Výsledek vynásobený 100, pak udává pravděpodobnost v procentech.

7.3 Použití poměru šancí

V této části se budou porovnávat šance na přežití vybraného neuronu vzhledem k všem ostatním pacientům dané léčby.

7.3.1 Léčba J

Nejprve se otestuje léčebná metoda J. Kontingenční tabulka pro léčbu J:

Tabulka 4: Kontigenční tabulka léčby J

	Neuron	Ostatní	Σ
Nezemřeli	3	30	33
Zemřeli	8	15	23
Σ	11	45	56

Poměr šancí léčby J:

$$OR = \frac{3 \cdot 15}{8 \cdot 30} = 0{,}188$$

Testovací kritérium léčby J:

$$|u| = \left| \frac{\ln \frac{3 \cdot 15}{8 \cdot 30}}{\sqrt{\frac{1}{3} + \frac{1}{8} + \frac{1}{30} + \frac{1}{15}}} \right| = |-2{,}463| < 1{,}96$$

Zde se zamítne nulová hypotéza. To znamená, že data nejsou statisticky odlišné.

7.3.2 Léčba K

Kontingenční tabulka léčby K:

Tabulka 5: Kontingenční tabulka léčby K

	Neuron	Ostatní	Σ
Nezemřeli	12	1	13
Zemřeli	20	5	25
Σ	32	6	38

Poměr šancí léčby K:

$$OR = \frac{12 \cdot 5}{20 \cdot 1} = 3$$

Testovací kritérium léčby K:

$$|u| = \left| \frac{\ln \frac{12 \cdot 5}{20 \cdot 1}}{\sqrt{\frac{1}{12} + \frac{1}{20} + \frac{1}{1} + \frac{1}{5}}} \right| = 0{,}951 < 1{,}96$$

Zde se nezamítne nulová hypotéza. Z toho vyplývá, že data jsou statisticky odlišné.

7.3.3 Léčba L

Poměr šancí léčby L:

$$OR = \frac{13 \cdot 11}{12 \cdot 20} = 0{,}596$$

Kontingenční tabulka pro L:

Tabulka 6: Kontigenční tabulka léčby L

	Neuron	Ostatní	Σ
Nezemřeli	13	20	33
Zemřeli	12	11	23
Σ	25	31	56

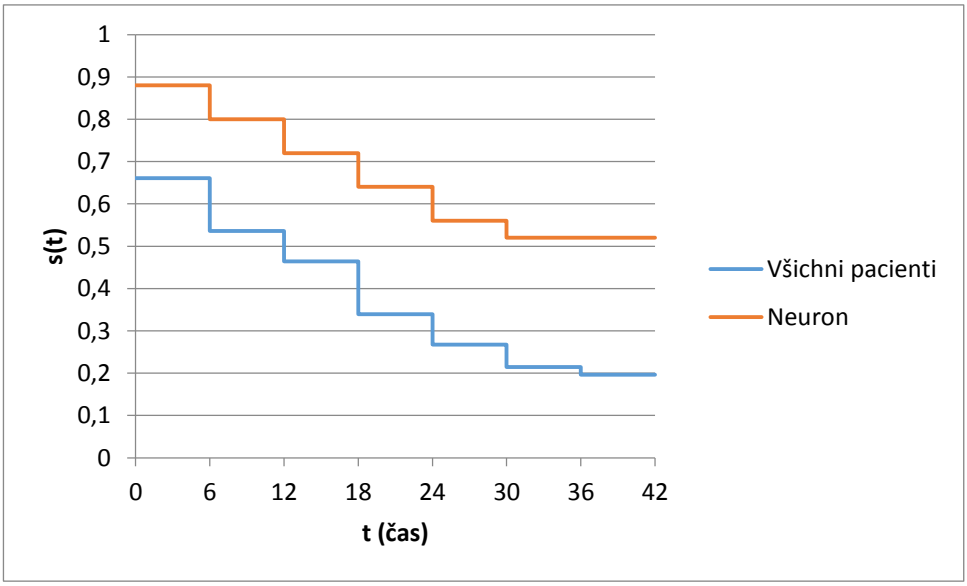
Testovací kritérium léčby L:

$$|u| = \left| \frac{\ln \frac{13 \cdot 11}{12 \cdot 20}}{\sqrt{\frac{1}{13} + \frac{1}{12} + \frac{1}{11} + \frac{1}{20}}} \right| = |-0{,}944| < 1{,}96$$

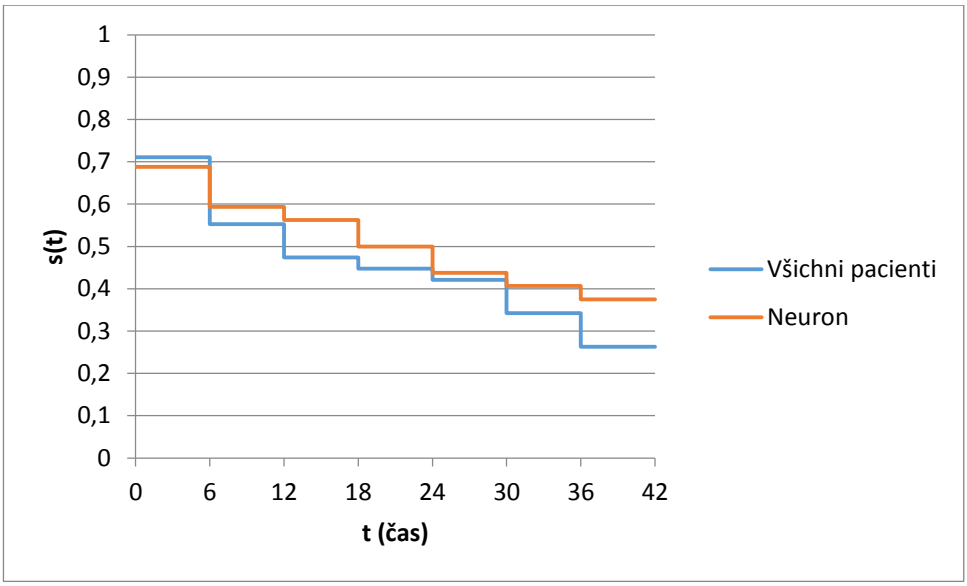
Opět se zde nezamítne nulová hypotéza a data tedy jsou statisticky odlišné. Tato léčba se zdá vzhledem k výsledkům z ostatních léčebných metod jako nejlepší varianta pro nového pacienta.

7.3.4 Graf funkce přežití

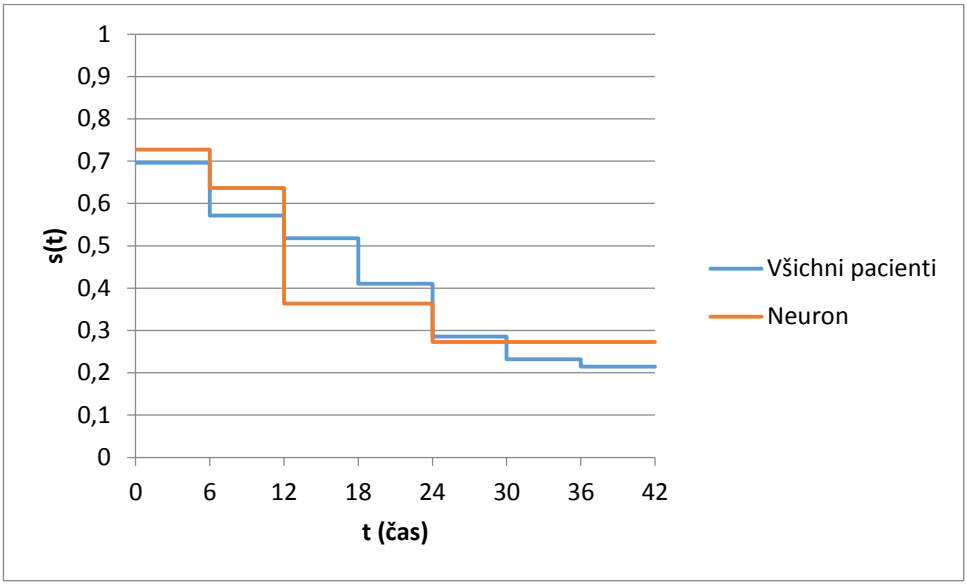
Základní princip funkce přežití je popsán v teoretické části (4.1.1). V grafech na ose x je vynesena doba v měsících a na ose y pravděpodobnost přežití pacienta. Porovnává se zde funkce přežití zvoleného neuronu vůči funkci přežití celé léčby (Obrázky 27, 28, 29). Z obrázků je patrné že v léčbě L se funkce přežití neuronu výrazně liší oproti zbylým pacientům. V ostatním léčbách není příliš velký rozdíl.



Obrázek 28: Graf přežití léčby L (vlastní)



Obrázek 29: Graf přežití léčby K (vlastní)



Obrázek 30: Graf přežití léčby J (vlastní)

ZÁVĚR

Cílem práce bylo naprogramovat aplikaci pro analýzu úspěšnosti léčby pomocí Kohenenových map. Po nastudování problematiky neuronových sítí jsem v práci popsal postup jak tyto mapy vytvořit. Problematika tvorby Kohonenovy mapy je popsána od samotného začátku.

Aplikaci je naprogramovaná v jazyce Java a umožňuje vytvářet Kohonenovy mapy na základě údajů o pacientech. Veškeré výpočty a mezivýpočty se zapisují do souboru, kde jsou dobře popsány. Aplikace také dokáže modifikovat pacienty a následně je ukládat do souboru. Z tohoto souboru se po spuštění aplikace nahrají zpět.

Při tvorbě aplikace jsem se snažil ošetřit veškeré chyby, které mohou nastat při zadávání hodnot. Myslím si, že aplikace funguje podle mých představ. Pro to, aby aplikace mohla být použita v praxi, bylo by nutné doplnit pár kritérií, podle kterých se vybírala vhodná léčba.

Po případném konzultování s odborníkem a přidáním více kritérií pro výběr léčby by aplikace mohla sloužit doktorům jako informační nástroj.

POUŽITÁ LITERATURA

[1] ŘEZANKOVÁ, Hana, Dušan HÚSEK a Václav SNÁŠEL. *Shluková analýza dat*. Praha: Professional Publishing, 2007. ISBN 978-80-86946-26-9.

[2] ŠÍMA, Jiří a Roman NERUDA. *Teoretické otázky neuronových sítí*. Praha: Matfyzpress, 1996. ISBN 80-85863-18-9.

[3] VOJÁČEK, Antonín. *Samoučící se neuronová síť – SOM, Kohonenovy mapy*. Plzeň: Západočeská Univerzita v Plzni, 2006. Dostupné z: http://www.kiv.zcu.cz/studies/predmety/uir/NS/Samouc_NN2.pdf

[4] ŠNOREK, Miroslav. *Neuronové sítě a neuropočítače*. Dot. 1. vyd. Praha: České vysoké učení technické, 1998. ISBN 80-01-01455-x.

[5] KVASNIČKA, Vladimír. *Úvod do teórie neurónových sietí*. Bratislava: IRIS, 1997. ISBN 80-887-7830-1.

[6] *Sme Primár: Leukémia* [online]. Bratislava: Petit Press, 2004 [cit. 2017-05-10]. Dostupné z: https://primar.sme.sk/c/4117019/leukemia.html

[7] ZVÁRA, Karel a Josef ŠTĚPÁN. *Pravděpodobnost a matematická statistika*. Vyd. 4. Praha: Matfyzpress, 2006. ISBN 80-867-3271-1.