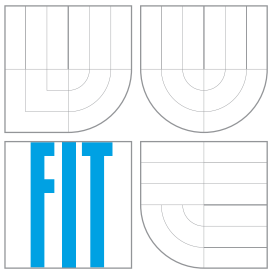
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

VIZUÁLNÍ SIMULÁTOR NEURONOVÝCH SÍTÍ TYPU ASOCIATIVNÍ PAMĚŤ

VISUAL SIMULATOR OF ASSOCIATIVE MEMORY TYPE OF NEURAL NETWORKS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — ŠTĚPÁN MRÁČEK
AUTHOR

VEDOUCÍ PRÁCE — Ing. DAVID MARTINEK
SUPERVISOR

BRNO 2008

Vizuální simulátor neuronových sítí typu asociativní paměť

Visual Simulator of Associative Memory Type of Neural Networks

Vedoucí:
Martinek David, Ing., UITS FIT VUT

Oponent:
Zbořil František V., doc. Ing., CSc., UITS FIT VUT

Zadání:

1. Prostudujte různé typy neuronových sítí a algoritmy pro jejich učení. Zaměřte se na neuronové sítě použitelné jako asociativní paměti.
2. Vyberte si vhodný typ neuronové sítě a navrhněte knihovnu pro práci s tímto typem sítě. Dále navrhněte aplikaci pro vizuální práci s tímto typem sítě, která bude spolupracovat s navrženou knihovnou. Tato aplikace musí být použitelná zejména pro demonstraci algoritmů učení. To znamená, že musí být schopna simulovat tyto algoritmy po jednotlivých krocích a vhodným způsobem graficky reprezentovat stav celé sítě.
3. Navrženou knihovnu a aplikaci implementujte. Vytvořte vhodné demonstrační příklady, které předvedou možnosti vytvořeného systému.
4. Zhodnoťte dosažené výsledky. Diskutujte možnosti dalšího rozšíření.

Část požadovaná pro obhajobu SP:
První dva body zadání

Kategorie:
Umělá inteligence

Implementační jazyk:
Podle vlastního uvážení

Operační systém:
Linux, Windows (je vyžadována přenositelnost)

Literatura:
Podle pokynů vedoucího

Licenční smlouva

Licenční smlouva je uložena v archivu Fakulty informačních technologii Vysokého učení technického v Brně.

Abstrakt

Bakalářská práce se zabývá problematikou neuronových sítí, které lze užít jako asociativní paměti. Teoretická část se zabývá výkladem pojmů neuron, neuronová síť a asociativní paměť. Praktická část navrhuje a popisuje implementaci aplikace sloužící k demonstraci algoritmů učení a odezvy sítě na určitý vstup.

Klíčová slova

Neurnová síť, neuron, umělá inteligence, vizuální simulátor, simulace, asociativní paměť

Abstract

This bachelor thesis is focused on associative neural networks. The theoretical part of the thesis presents an explanation of neuron, neural network and associative memory concepts. The practical part is about designing and implementing an application allowing the demonstration of learning algorithms and the response to specific input.

Keywords

Neural network, neuron, artificial intelligence, visual simulator, simulation, associative memory

Citace

Štěpán Mráček: Vizuální simulátor neuronových sítí typu asociativní paměť, bakalářská práce, Brno, FIT VUT v Brně, 2008

Vizuální simulátor neuronových sítí typu asociativní paměť

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Davida Martinka

.....
Štěpán Mráček
10. srpna 2008

Poděkování

Děkuji vedoucímu práce Ing. Davidu Martinkovi za cenné rady a připomínky.

Obsah

Kapitola 1

Úvod

Jako téma své bakalářské práce jsem si vybral vizuální simulátor neuronových sítí se zaměřením na asociativní paměti. Aplikace, kterou jsem vytvořil slouží především ke studijním účelům a poskytuje uživateli ucelený přehled o chování konkrétní neuronové sítě. Umožňuje interaktivně měnit její vlastnosti a následně sledovat, jak se tyto změny projeví. Spíše než na rychlost výpočtu je zaměřena na to, aby poskytovala přehledné a komplexní informace, díky kterým je snadnější problematiku neuronových sítí pochopit.

V kapitole 2 jsou popsány teoretické základy neuronových sítí. Kapitola 3 je pak blíže zaměřena na sítě, které lze použít jako asociativní paměti. Po teoretickém úvodu následuje kapitola 4 věnovaná analýze problému. Obsahuje návrh knihovny pro práci s neuronovými sítěmi a možné způsoby zobrazení informací o neuronové síti. Na základě těchto znalostí bude navrhnuta aplikace, která by měla být schopna uživateli intuitivní cestou přiblížit fungování asociativních neuronových sítí. Implementaci je věnovaná samostatná kapitola 5. Kapitola 6 je uživatelský návod, kde jsou podrobněji a názorně popsány možnosti aplikace.

Toto téma jsem si vybral proto, že mě zajímá problematika neuronových sítí, strojového učení a umělé inteligence vůbec. Myslím si, že je to velice zajímavá oblast informatiky, ve které spousta věcí a souvislostí teprve čeká na své objevení.

Kapitola 2

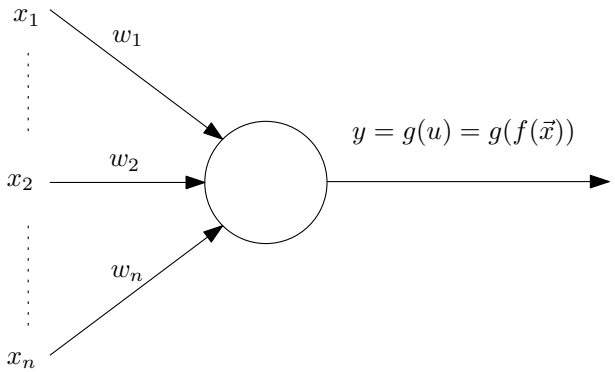
Teoretické souvislosti

V této kapitole bude popsána teorie, kterou při řešení svého projektu používám. Budou vysvětleny pojmy neuron, neuronová síť a asociativní paměť. Při zpracování teoretické části mé zprávy jsem čerpal z publikací Soft Computing and Itelligent System Design [5], Umělá inteligence (4) [9] a přednášek pana doc. Františka Zbořila v akademickém roce 2007/2008 [6].

2.1 Neuron

Neuron, základní jednotka neuronových sítí, je v podstatě abstrakcí biologického neuronu. Vstupem je vektor $\vec{x}$, jehož jednotlivé prvky mohou být buď vstup samotné sítě, nebo výstup jiného neuronu. Zmíněný vektor je bázovou funkcí převeden na skalární hodnotu $f(\vec{x})$ a následně je tato hodnota vstupem pro aktivační funkci g, která produkuje výstup neuronu y. Jednotlivým vstupům je zpravidla přiřazena váha w.

$$y = g(f(\vec{x})) \tag{2.1}$$



Obrázek 2.1: Neuron

Bázová funkce určuje, jakým způsobem je komponováno n složek vstupního vektoru $\vec{x} = (x_1, x_2, \cdots, x_n)$. Každému vstupu x_i odpovídá váha w_i.

Používá se **lineární bázová funkce (LBF)**

$$f(\vec{x}) = \sum_{i=1}^{n} w_i \cdot x_i \tag{2.2}$$

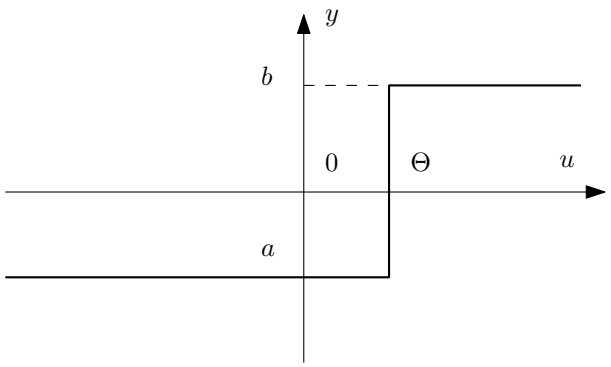
a **radiální bázová funkce (RBF)**

$$f(\vec{x}) = \|\vec{x} - \vec{w}\| = \sqrt{\sum_{i=1}^{n} (x_i - w_i)^2} \tag{2.3}$$

Aktivační funkce je závislá na použité bázové funkci. Při použití LBF se nejčastěji používá skoková aktivační funkce:

$$y = \begin{cases} a & \text{pro } u \leq \Theta \\ b & \text{pro } u > \Theta \end{cases} \tag{2.4}$$

Kde u je hodnota bázové funkce a Θ práh citlivosti neuronu. Hodnoty a a b, které mohou být výstupem neuronu se často upravují na $a = -1$ a $b = 1$ nebo $a = 0$ a $b = 1$.

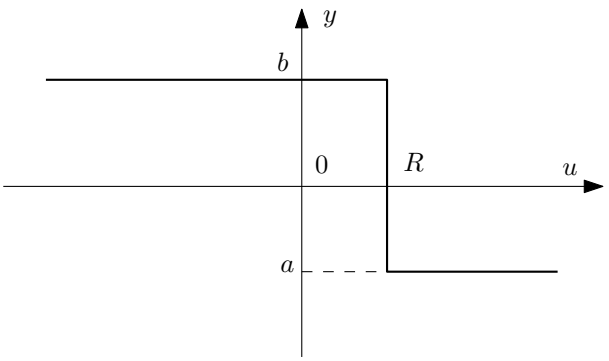


Obrázek 2.2: Skoková aktivační funkce u neuronů s LBF

U neuronů s RBF bývá skoková aktivační funkce ve tvaru:

$$y = \begin{cases} b & \text{pro } u \leq R \\ a & \text{pro } u > R \end{cases} \tag{2.5}$$

Kde a a b často nabývají hodnot $a = 0$ a $b = 1$. Hodnota R bývá označována jako poloměr. Vstupní vektor $\vec{x} = x_1, x_2, \ldots, x_n$ si lze představit jako bod v n-rozměrném prostoru, odpovídající váhy $w_1, w_2, \ldots, w_n$ spolu s poloměrem R jako kulovou plochu. Pokud je tedy vstup $\vec{x}$ uvnitř této kulové plochy, výstup neuronu nabývá hodnoty 1, v opačném případě hodnoty 0.



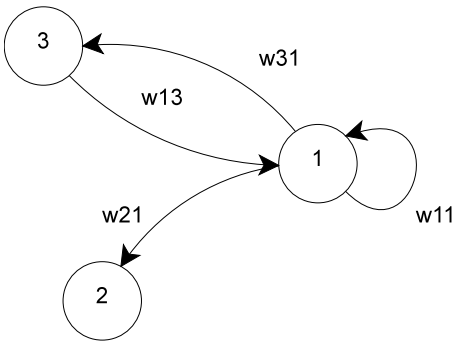
Obrázek 2.3: Skoková aktivační funkce u neuronů s RBF

2.2 Neuronové sítě

Neuronová síť je matematickým modelem biologických neuronových sítí. Tvoří ji soubor vzájemně propojených neuronů, které si prostřednictvím spojů předávají informace. Vstupy jednotlivých neuronů jsou transformovány bázovou a aktivační funkcí a přivedeny na vstup jiného neuronu. Tímto způsobem je postupně transformován celý vstup neuronové sítě. Jak již bylo uvedeno, míra ovlivnění výsledku bázové funkce vstupem $\vec{x}$ neuronu je určena váhou w.

Pro popis celé sítě, přesněji řečeno vah v ní, se často používá maticový zápis. V každém řádku jsou postupně uvedeny váhy vstupů pro všechny neurony v síti. Na hlavní diagonále jsou váhy přímých zpětných vazeb neboli případů, kdy je výstup neuronu opět přiveden na jeho vstup. Následující matice vah odpovídá síti znázorněné na obrázku 2.4.

$$\begin{bmatrix} w_{11} & - & w_{13} \\ w_{21} & - & - \\ w_{31} & - & - \end{bmatrix}$$



Obrázek 2.4: Váhy v neuronové síti

Důležitou vlastností neuronových sítí je schopnost učit se a následně při svém nasazení zobecňovat. Neuronové sítě tak nachází uplatnění v mnoha oblastech. Nejčastěji se užívají při řešení následujících problémů:

- klasifikace

- komprese dat
- predikce
- řízení
- optimalizace
- asociace

Právě poslední zmíněný bod možné aplikace mě zaujal, a proto se mu vrámci své bakalářské práce blíže věnuji.

2.2.1 Učení neuronových sítí

Učení neuronových sítí spočívá v nastavení vah spojů mezi neurony. Při učení se zpravidla používá tzv. trénovací množina, jejíž forma určuje základní způsoby učení:

Učení s učitelem – Při tomto způsobu je každý krok učení informován o požadované odezvě sítě. Trénovací množina T je pak ve tvaru:

$$T = \{(\vec{i}_1, \vec{d}_1), (\vec{i}_2, \vec{d}_2), \ldots, (\vec{i}_P, \vec{d}_P)\} \tag{2.6}$$

kde
P je počet prvků množiny
$\vec{i}_i$ je i-tý vstupní vektor
$\vec{d}_i$ je i-tý požadovaný výstupní vektor

Učení bez učitele – U tohoto způsobu učení tvoří trénovací množinu T pouze jednotlivé vstupní vektory:

$$T = \{\vec{i}_1, \vec{i}_1, \ldots, \vec{i}_P\} \tag{2.7}$$

2.3 Asociativní paměti

Paměť z pohledu elektroniky a informatiky je zařízení, které dokáže načíst, uchovávat a zprostředkovat informace. Jeden ze způsobů klasifikace pamětí je odvozen podle toho, jak je možno přistupovat k uloženým informacím.

RAM - Random Access Memory. V tomto případě je možné si paměť představit jako množinu očíslovaných paměťových buněk. Obsah paměti je zpřístupněn na základě čísla buňky a doba přístupu je na tomto čísle zcela nezávislá.

Sekvenční – U sekvenčních pamětí lze přistupovat k libovolné buňce na základě jejího čísla, avšak doba přístupu není konstantní. Ilustrativním příkladem tohoto typu paměti může být magnetofonová páska.

Asociativní – U asociativních pamětí jsou informace poskytnuty na základě obsahu. Adresa, kterou si lze představit jako vektor, je mapována transformační funkcí Φ na informaci uloženou v paměti. Například fotografie z určité databáze je zpřístupněna na základě jména vyfotografované osoby.

Na asociativními paměti se váží pojmy autoasociace a heteroasociace, které zase úzce souvisí s učením s učitelem a bez učitele.

Autoasociace – Trénovací množinu $T = \{\vec{i}_1, \vec{i}_2, \ldots, \vec{i}_P\}$ tvoří vzory, které chceme uložit do paměti. V kontextu neuronových sítí to budou vstupní vektory. Transformační funkce je pak pro vzory z trénovací množiny identitou:

$$\Phi(\vec{i}_j) = \vec{i}_j \tag{2.8}$$

Pro prvky $\vec{i}_x$, které nejsou součástí trénovací množiny vypadá asociace následovně. Z trénovací množiny bude vybrán prvek $\vec{i}_i$, jehož Hammingova vzdálenost $d = H(\vec{i}_x, \vec{i}_i)$ je minimální. Transformační funkce pak bude mít tvar:

$$\Phi(\vec{i}_x) = \vec{i}_i \tag{2.9}$$

Heteroasociace naopak předpokládá trénovací množinu ve tvaru dvojic vstupů a požadovaných výstupů:

$$T = \{(\vec{i}_1, \vec{d}_1), (\vec{i}_2, \vec{d}_2), \ldots, (\vec{i}_P, \vec{d}_P)\} \tag{2.10}$$

Transformační funkce má tedy tvar:

$$\Phi(\vec{i}_j) = \vec{d}_j \tag{2.11}$$

Pro prvky $\vec{i}_x$, které nejsou součástí trénovací množiny vypadá asociace následovně. Z trénovací množiny bude vybrán prvek $\vec{i}_i$, jehož Hammingova vzdálenost $d = H(\vec{i}_x, \vec{i}_i)$ je minimální. Transformační funkce pak bude mít tvar:

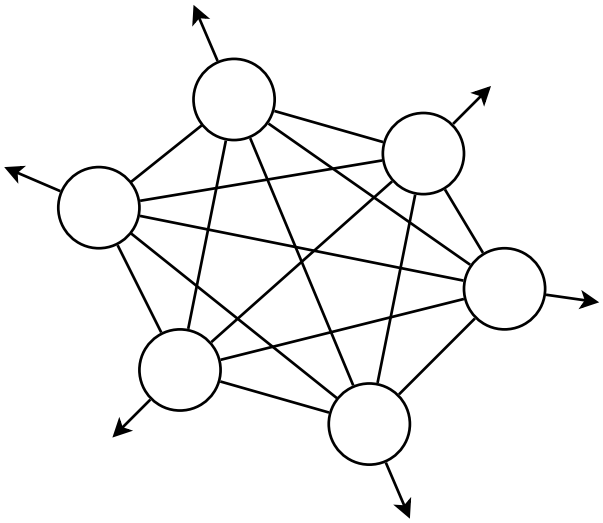
$$\Phi(\vec{i}_x) = \vec{d}_i \tag{2.12}$$

Kapitola 3

Neuronové sítě použitelné jako asociativní paměti

3.1 Hopfieldova neuronová síť

Nejznámějším neuronovou autoasociativní pamětí je Hopfieldova neuronová síť. Jedná se o plně propojenou síť. Vzájemné vazby jsou symetrické (tedy $w_{ij} = w_{ji}$) a přímé zpětné vazby jsou nulové ($w_{ii} = 0$).



Obrázek 3.1: Hopfieldova neuronová síť

3.1.1 Odezva sítě

Síť nemá žádný vstup, a proto se při výpočtu odezvy v prvním kroku nastaví výstup všech neuronů na požadovaný vstup:

$$y(0) = \vec{i} \tag{3.1}$$

Odezva sítě se děje ve více krocích. V následujícím vzorci je popsán postup výpočtu kroku $k+1$ neuronu j. Všechny neurony obsahují lineární bázovou funkci u (2.2). Aktivační funkce je skoková (2.4) s drobnou změnou. Pokud je hodnota bázové funkce v kroku stejná jako práh citlivosti neuronu Θ, hodnota výstupu se nemění.

$$y_j(k+1) = \begin{cases} 1 & \text{pro } u_j(k) > \Theta_j \\ -1 & \text{pro } u_j(k) < \Theta_j \\ y_j(k) & \text{pro } u_j(k) = \Theta_j \end{cases} \tag{3.2}$$

V každém kroku se změní výstupní hodnota maximálně jednoho neuronu tudíž i energie celé sítě. Až se energie sítě ustálí, výstupem sítě se stane vektor všech výstupů neuronů.

3.1.2 Učení sítě

Pokud je trénovací množina $T = \{\vec{i}_1, \vec{i}_2, \ldots, \vec{i}_P\}; \vec{i}_i = \{-1, 1\}^n$, pak pro jednotlivé váhy platí:

$$w_{ij} = \frac{1}{P} \sum_{p=1}^{P} i_{pi} \cdot i_{pj} \tag{3.3}$$

Výpočet prahů se řídí následujícím vzorcem:

$$\Theta_i = \frac{1}{2} \sum_{j=1}^{n} w_{ij} \tag{3.4}$$

Práh citlivosti neuronu Θ_i je polovina součtu všech vstupních vah.

3.1.3 Energie

Energetická funkce je důležitou charakteristikou sítě. Každému stavu sítě odpovídá určitá hodnota energie. Lze ji vypočítat pomocí vztahu:

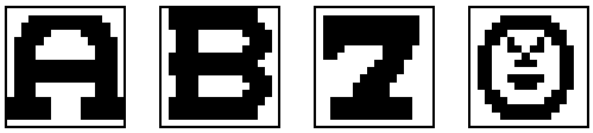
$$E = -\frac{1}{1} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} y_i y_j + \sum_{i=1}^{n} \Theta_i y_i \tag{3.5}$$

Změna energie v každém kroku vybavení není nikdy kladná a konverguje k určité hodnotě – lokálnímu nebo globálnímu minimu ([5], [9], [6]), čímž se řeší optimalizační problém. Proto bývá Hopfieldova síť užívána i k řešení problému obchodního cestujícího [9].

3.1.4 Další vlastnosti

Podstatným problémem Hopfieldovy sítě je její malá kapacita. Počet uložených vzorů P je vždy menší než $\frac{n}{4\ln(n)}$, kde n je počet neuronů v síti.

Jak bylo uvedeno výše, při vybavení stav sítě konverguje do lokálního minima. To však nemusí nutně reprezentovat požadovaný výstup.



Obrázek 3.2: Trénovací množina Hopfieldovy sítě

3.1.5 Příklad

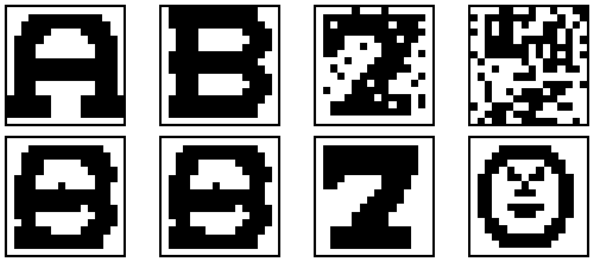
Na obrázku 3.2 je znázorněna trénovací množina Hopfieldovy sítě čítající 4 vzory. Každý jednotlivý pixel obrázku odpovídá jednomu neuronu sítě. Černý pixel odpovídá nastavení výstupu příslušného neuronu na hodnotu 1, bílý bod na hodnotu -1. Obrázek má velikost 16x16 tedy 256 pixelů. Neuronová síť pak obsahuje také 256 neuronů.

Odezva sítě je demonstrována na obrázku 3.3. V prvním řádku jsou jednotlivé vstupy, v druhém řádku jednotlivé odezvy. Trénovací množina i její odpovídající odezvy jsou vybrány záměrně tak, aby byly patrné základní vlastnosti Hopfieldovy sítě.

Odezva pro první dva vzory (písmena A a B) je stejná a neodpovídá žádnému vzoru z trénovací množiny. Tento případ nastává často, pokud jsou si některé vzory trénovací množiny podobné (Jejich Hammingova vzdálenost je malá).

Zobecňující vlastnosti sítě jsou patrné pro třetí vzor. Síti byl předložen částečně poškozený vzor, který byl poté správně rozpoznán.

Čtvrtý vzor je zcela náhodný. Přesto mu byl autoasociací přiřazen vzor z trénovací množiny.



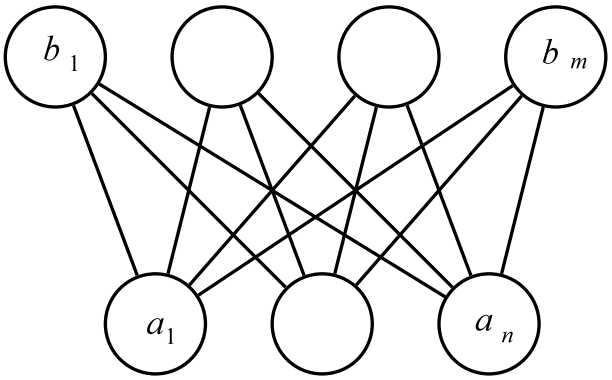
Obrázek 3.3: Odezva Hopfieldovy sítě

3.2 BAM

Síť BAM (Bidirectional associative memory – obousměrná asociativní paměť) je heteroasociativní pamětí. Skládá se ze dvou plně symetricky propojených vrstev. Vazby v rámci stejné vrstvy jsou nulové. Vrstva A obsahuje n neuronů označených $a_1, a_2, \ldots, a_n$, Vrstva B se skládá z m neuronů, které se označují $b_1, b_2, \ldots, b_m$.

3.2.1 Odezva sítě

Všechny neurony v síti obsahují lineární bázovou funkci (2.2) a skokovou aktivační funkci (2.4). Hodnota bázové funkce u_{bj} neuronu b_j z vrstvy B je:



Obrázek 3.4: BAM

$$u_{bj} = \sum_{i=1}^{n} w_{ji} a_i \tag{3.6}$$

Také zde probíhá odezva sítě ve více krocích. Pro výpočet výstupu neuronu v následujícím kroce $k+1$ platí:

$$b_j(k+1) = \begin{cases} 1 & \text{pro } u_{bj}(k) > \Theta_{bj} \\ -1 & \text{pro } u_{bj}(k) < \Theta_{bj} \\ b_j(k) & \text{pro } u_{bj}(k) = \Theta_{bj} \end{cases} \tag{3.7}$$

Obdobně je definována bázová i aktivační funkce neuronů ve vrstvě A:

$$u_{ai} = \sum_{j=1}^{m} w_{ij} b_i \tag{3.8}$$

$$a_i(k+1) = \begin{cases} 1 & \text{pro } u_{ai}(k) > \Theta_{ai} \\ 0 & \text{pro } u_{ai}(k) < \Theta_{ai} \\ b_j(k) & \text{pro } u_{ai}(k) = \Theta_{ai} \end{cases} \tag{3.9}$$

V každém kroku se změní výstupní hodnota maximálně jednoho neuronu tudíž i energie celé sítě. Až se energie sítě ustálí, výstupem sítě se stane vektor všech výstupů neuronů z vrstvy B.

3.2.2 Učení sítě

Protože se jedná o heteroasociativní paměť, trénovací množinu T tvoří dvojice požadovaných vstupů a výstupů. Konrétně u sítě BAM to je množina dvojic výstupů vrstev A a B:

$$T = \{(\vec{a}_1, \vec{b}_1), (\vec{a}_2, \vec{b}_2), \ldots, (\vec{a}_P, \vec{b}_P)\} \tag{3.10}$$

Pro váhy a prahy platí:

$$w_{ij} = w_{ji} = \frac{1}{P} \sum_{p=1}^{P} a_{pi} b_{pj} \tag{3.11}$$

$$\Theta_{ai} = \frac{1}{2} \sum_{j=1}^{m} w_{ij} \tag{3.12}$$

$$\Theta_{bji} = \frac{1}{2} \sum_{i=1}^{n} w_{ji} \tag{3.13}$$

3.2.3 Energie sítě

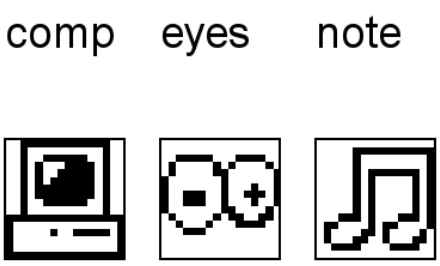
Energetické vlastnosti sítě jsou velmi podobné jako u Hopfielovy sítě. Také zde postupně při výpočtu odezvy klesá hodnota energie do globálního nebo lokálního minima energetické funkce, která má následující tvar:

$$E = -\sum_{i=1}^{n} \sum_{j=1}^{m} w_{ij} a_i b_j + \sum_{i=1}^{n} \Theta_{ai} a_i + \sum_{j=1}^{m} \Theta_{bj} b_j \tag{3.14}$$

Nedostek sítě BAM spočívá také v náklonnosti ke konvergenci k falešným atraktorům a v malé kapacitě stavů, které je schopná si zapamatovat.

3.2.4 Příklad

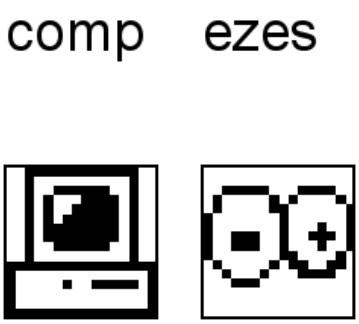
Následující příklad popisuje situaci v síti, která má ve vrstvě A 32 neuronů interpretovaných jako 4 ASCII znaky a ve vrstvě B 256 neuronů, kterým odpovídá černo-bílý obrázek o velikosti 16x16 pixelů. Trénovací množina na obrázku 3.5 obsahuje celkem 3 páry.



Obrázek 3.5: Trénovací množina sítě BAM

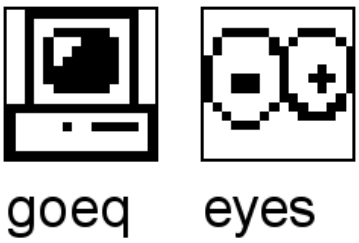
Literatura použitá pro tuto práci uvádí různé způsoby nastavení vsupu sítě a následný výpočet odezvy. Prvním způsobem je možné nastavit požadovaný vstup všem neuronům sítě. V tomto případě je síti předložen poškozený pár a podobně jako u Hopfieldovy autoasociativní paměti si síť vybaví správný pár.

Druhým způsobem lze nastavit pouze jednu vrstvu sítě a pro neurony druhé vrstvy počítat odezvu tak dlouho, dokud se stav neustálí. Tento postup je znázorněn na obrázku 3.6. Síti byl předložen jako vstup čtyřznakový řetězec, který byl přímo součástí trénovací množiny. V druhém případě se jednalo o lehce pozměněný řetězec (záměna „z“ a „y“).



Obrázek 3.6: Odezva sítě BAM

Směr odezvy sítě může být také opačný (znázorněno na obrázku 3.7). Zde je patrné, že první asociace neproběhla v pořádku. Pravděpodobně došlo k atrakci k nějakému lokálnímu minumu energetické funkce, kterému však neodpovídá žádný stav z trénovací množiny. Místo očekávaného „comp“ se objevilo „goeq“.



Obrázek 3.7: Odezva sítě BAM

Kapitola 4

Analýza zadaného problému

V této kapitole bude podrobněji rozebráno zadání. Na základě této analýzy navrhnu způsoby řešení. Ještě předtím však uvedu pár projektů řešící obdobný problém, se kterými jsem se během studia setkal.

4.1 Zadání

Úkolem bylo prostudovat různé typy neuronových sítí a zaměřit se především na ty, které lze využít jako asociativní paměti. Tato teoretická oblast je zahrnuta v kapitolách 2 a 3. Dále bylo nutné navrhnout a implementovat knihovnu a aplikaci pracující s vybraným typem sítí. Pro demonstraci možností aplikace následně vytvořit sadu demonstračních příkladů.

4.2 Nástroje pro práci s neuronovými sítěmi

V této kapitole bude představeno několik nástrojů pro práci s neuronovými sítěmi.

4.2.1 SNNS

SNNS - Stuttgart Neural Network Simulator [4] je komplexní nástroj pro vizualizální simulaci a analýzu neuronových sítí. Původně byl vytvořen na univerzitě ve Stuttgartu, dále byl rozvíjen na univerzitě v Tübingenu. Simulátor je tvořen samotným simulačním jádrem napsaným v jazyce C, grafickým uživatelským rozhraním a řadou dalších analytických nástrojů pro sledování stavu sítě. Uživatlé mají možnost doplnit vlastní aktivační a bázové funkce a stejně tak i algoritmy učení. Standardní distribuce této aplikace už obsahuje řadu předdefinovaných učebních algoritmů, ale žádný z nich se netýká Hopfieldovy a BAM síťě. V současné době (duben 2008) se aplikace nevyvíjí.

4.2.2 PySNNS

PySNNS je součástí projektu SNNS-development [2], jehož cílem je podpora SNNS ve formě záplat, nástrojů a dalších důležitých informací. Poslední verze (únor 2008) byla vydána v červnu 2004.

4.2.3 JavaNNS

Java Neural Network Simulator [3] je následovník SNNS. Po svém předchůdci zdědil simulační jádro, ale získal nové uživatelské rozhraní naprogramované v jazyce Java, které sloužilo jako inspirace při návrhu a implementaci mého zadání.

4.2.4 Neural Networks with Java

Jedná o knihovnu napsanou v jazyce Java [7] pro práci s neuronovými sítěmi. Původně vznikla jako diplomová práce, ale od roku 2004 ji autor dále vyvíjí. Je vhodná především pro práci se sítěmi typu Kohonenovy mapy a Back propagation.

4.2.5 FANN

Fast Artificial Neural Network Library [8] je rozáhlá knihovna napsaná v jazyce C. Nabízí navíc také grafické uživatelské rozraní a podporu volání funkcí z jiných programovacích jazyků, jako například C++, Perl, PHP, Python, Ruby a dokonce Matlab nebo Octave.

4.2.6 Associative Neural Network Library

Knihovna se využívá pro práci s neruronovými sítěmi, které mohou být použity jako asociativní paměti. Je napsána v jazyce C++ a umožňuje vytváření a testování různých druhů asociativních neuronových sítí, většinou variant Hopfieldovy sítě. Poslední verze vyšla v roce 2004 [1].

4.3 Rozdělení problému na podproblémy

Zadaný problém se skládá z několika částí. Je nutné navrhnout a implementovat knihovnu pro práci s neuronovými sítěmi a následně vytvořit aplikaci, která s touto knihovnou bude pracovat. Knihovna bude rozdělena na dvě části. První z nich bude společná část pro všechny typy sítí zahrnující popis sítě z pohledu neuronů a vazeb mezi nimi. Ve druhé části budou popsány algoritmy učení pro jednotlivé typy sítí a jejich další specifické vlastnosti.

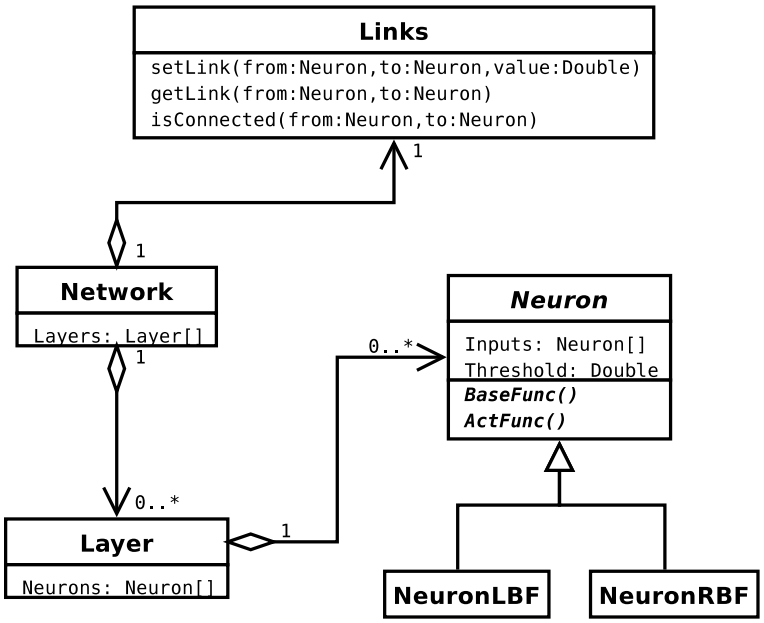
Analýzu a vývoj aplikace bude opět rozdělena na dva podproblémy. Jedna část aplikace bude společná pro všechny typy sítí. Další část pak bude spolupracovat s konkrétním typem sítě. Toto rozdělení zohledňuje fakt, že různé typy sítí požadují různé trénovací množiny. Některé se učí na dvojicích vstupů a požadovaných výstupů, některé pouze na jednotilvých vektorech vstupů.

V následujících kapitolách budou postupně rozebrány všechny podproblémy. Bude navržena obecná knihovna, dále knihovna pro práci s různými typy sítí, aplikace a rozhraní mezi knihovnou a aplikací.

4.4 Obecná knihovna

Pro návrh i následnou implementaci zadaného systému byl použit objektově orientovaný přístup. Intuitivně jednotlivé prvky neuronové sítě – síť samotná, vrstvy v síti, vazby mezi neurony i jednotlivé neurony – odpovídají budoucím třídám a objektům. Síť se skládá z jednotlivých vrstev a vrstvy z neuronů navzájem propojených vazbami. Vazba může existovat mezi libovolnými dvěma neurony. Existuje dokonce případ, kdy je výstup přiveden

na vstup stejného neuronu. Tato skutečnost musí být v návrhu také zahrnuta. V teoretické části již bylo zmíněno, že neurony se liší podle způsobu převádo svého vstupu na výstup. Bázová i aktivační funkce může být počítána různým způsobem. Například neurony s lineární bázovou funkcí při výpočtu výstupu nejprve provedou vážený součet. Pokud je tato suma větší než zadaný práh, jejich výstupem je zpravidla hodnota 1. V opačném případě je pak jejich výstupem -1 nebo 0. I toto musí být v knihovně podchyceno.



Obrázek 4.1: Diagram tříd knihovny

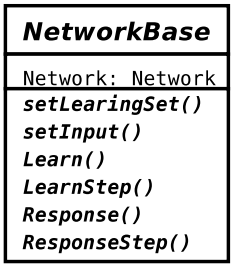
Výsledek této analýzy je znázorněn v diagramu tříd na obrázku 4.1. Nejsou v něm sice zahrnuty všechny atributy a metody, avšak poskytuje ucelený pohled na hierarchii tříd v obecné knihovně. Třída popisující neuron je navržena jako abstraktní. Konkrétní výpočet bázové funkce `baseFunc()` a aktivační funkce `actFunc()` je ponechán na potomcích této třídy. V diagramu jsou naznačeny neurony s lineární a radiální bázovou funkcí.

Váhy jednotlivých spojů mezi neurony jsou uloženy v samostatné třídě `Links`, která v podstatě implementuje asociativní pole. Klíčem je dvojice neuronů, hodnotou pak konkrétní váha. Kromě toho obsahují jednotlivé neurony také seznam všech svých vstupů. Implementace knihovny tento aspekt zohledňuje a například při odstranění spoje ve třídě `Links` bude odebrán odpovídající záznam v seznamu vstupů. Ačkoliv se může zdát, že je takové ukládání dat zbytečně redundantní, rychlý přístup k seznamu vstupů aktuálního neuronu je využíván například při výpočtu bázové funkce.

4.5 Konkrétní typy sítí

Jednotlivé neuronové sítě se liší podle způsobu nastavení jejich vstupu, výstupem, algoritmy učení sítě a podle výpočtu odezvy na určitý vstup. Bude navrženo obecné rozhraní, které bude šablonou pro jednotnou práci se všemi typy sítí. Protože bylo požadováno, aby knihovna i aplikace mohly sloužit k studijním a demonstračním účelům, bude vhodné rozdělit učení sítě i odezvu na jednotlivé kroky tak, aby poskytly uživateli ucelený přehled

o chování sítě.



Obrázek 4.2: Třída zapouzdřující konkrétní chování sítě

Abstraktní třída navržená na základě předcházející analýzy je na obrázku 4.2. Obsahuje dříve zmíněnou třídu popisující obecnou neuronovou síť a také abstraktní metody, které slouží pro jednotnou práci se sítí. Abstraktní třída se chová jako stavový automat. Na základě volaných metod mění svůj vnitřní stav. Pro větší přehlednost nejsou v diagramu zobrazeny parametry a návratový typ jednotlivých metod. V následujícím přehledu je krátce popsána sémantika jednotlivých metod.

setLearningSet() Nastaví trénovací množinu.

setInput() Sítě mohou různými způsoby nastavovat svůj vstup. Například u Hopfiedovy sítě se nastavuje výstup všech neuronů, u sítě BAM se nastavuje výstup neuronů v konkrétní vrstvě.

Learn() Na základě vložené trénovací množiny se provede učení sítě.

LearnStep() Provede jeden krok učení.

Response() Vrací odezvu sítě, která je závislá na předloženém vstupu.

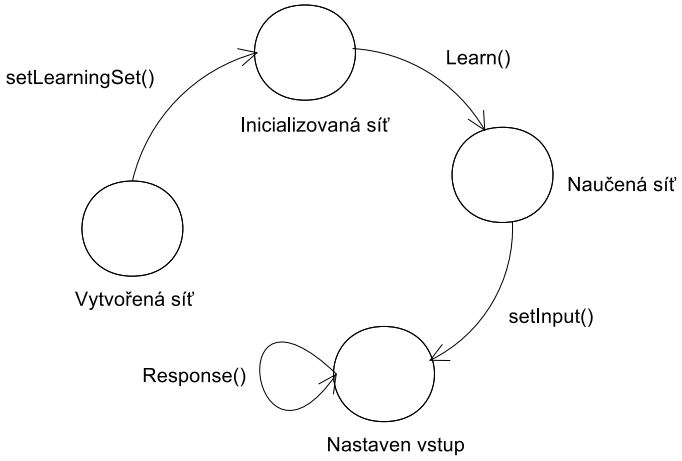
ResponseStep() Odezva se pro sítě s více neurony počítá zpravidla tak, že je postupně výstup jednoho neuronu přiveden na vstup jiného. Zde je spočítána hodnota bázové a aktivační funkce a výsledek předán dále. Pro zkoumání průběhu výpočtu odezvy slouží právě tato metoda.

Návrh možné implementace metod `Learn()` a `LearnStep()` (respektive `Response()` a `ResponseStep()`) je naznačen v následujícím pseudokódu:

```
Learn()
{
  nastav potrebne parametry pro prubeh uceni;

  while (sit neni naucena)
  {
    LearnStep();
  }
}
```

Na obrázku 4.3 je popsána změna vnitřního stavu sítě v závislosti na volaných metodách.



Obrázek 4.3: Reakce na volání jednotilvých metod

4.6 Vizualizace neuronových sítí

V této kapitole bude popsáno, jak reprezentovat topologii, aktuální stav učení a vybavování neuronových sítí na obrazovce.

4.6.1 Topologie

Grafické znázornění topologie sítě bylo znázorněno na obrázcích 3.1 a 3.4 v kapitole 3. Jednotlivé neurony lze zobrazit jako malé kružnice a spoje mezi nimi jako orientované úsečky. V závislosti na typu sítě lze neurony uspořádat do jednotlivých vrstev (například síť BAM) nebo do kruhu (Hopfieldova síť).

Při zobrazení aktuálního stavu sítě je také znázorněna síla vah, práh citlivosti neuronů a jejich aktuální výstup. Všechny tyto vlastnosti lze vyjádřit převedením do barevného prostoru. Slabé váze nebo nízké hodnotě prahové citlivosti odpovídá modrá barva. Červené odpovídá vysoká hodnota váhy respektive prahové citlivosti. Takto lze postupně využít celé barvené spektrum. Barva orientované úsečky určuje sílu váhy, barva obrysu kružnice určuje práh citlivosti.

Jako místo pro vyjádření hodnoty výstupu byla zvolena výplň kružnice znázorňující neuron. Neurony se skokovou aktivační funkcí zpravidla produkují diskrétní hodnoty 1 nebo -1. Výstupu 1 odpovídá černá barva výplně, výstupu -1 bílá barva. Pokud je aktivační funkce spojitá, barva výplně bude vyjádřena v odstínech šedi.

4.6.2 Vstupy a výstupy vtrstev i celé sítě

Další možností, jak poskytnout přehled o stavu sítě, je sledovat vstup nebo výstup konkrétní vrstvy. Tu lze interpretovat jako:

Text – vždy 8 neuronů si lze představit jako bajt, ve kterém je na konkrétním bitu hodnota 1 při výstupu větším nebo rovném nule. Bit bude nabývat hodnoty 0, pokud je výstup menší než 0. Hodnota této osmice v rozsahu 0-255 je převedena na znak z ASCII tabulky.

Černo-bílý obrázek – Pixelům odpovídají jednotlivé neurony. Obdobně jako v předcházejícím bodě je hodnota výstupu převedena na binární hodnotu. Černému pixelu pak odpovídá hodnota výstupu větší nebo rovna nule, bílému hodnota menší než nula.

Obrázek v odstínech šedi – Opět jsou neurony rozděleny po skupinách po osmi. Číselná hodnota v rozsahu 0-255 je však převedena na intenzitu konkrétního pixelu.

4.6.3 Další možnosti

Další možnosti jsou specifické pro konkrétní typy sítí. Například u Hopfieldovy sítě a BAM lze sledovat vývoj hodnoty energetické funkce při odezvě na určitý vstup. Může tak být předveden záporný nebo nulový přírůstek energie v každém kroku odezvy.

4.7 Návrh aplikace

Jak bylo uvedeno v části o nástrojích pro práci s neuronovými sítěmi, inspirací pro návrh uživatelského rozhraní byla aplikace JavaNNS [3]. Hlavní okno aplikace je koncipováno jako MDI (Multiple Document Interface). V tomto okně je tedy možné zobrazit další podokna, která slouží pro ovládání a zobrazení informací o síti. V následujícím výčtu je souhrn komponent, které byly navrženy a poté v aplikaci implementovány:

Kontrolní panel – Komponenta slouží pro ovládání celé sítě. Umožňuje nahrát trénovací množinu, sledovat učení sítě i její odezvu, diagnostikovat stav sítě a nastavovat, jakým způsobem bude interpretován vstup a výstup sítě.

Informační panel – Komponenta, která informuje uživatele o nejdůležitějších vlastnostech sítě: Název sítě se kterou uživatel právě pracuje a počet neuronů celkem i v jednotlivých vrstvách.

Interpretace vstupu a výstupu – Podle nastavení v kontrolním panelu se zde interpretuje aktuální vstup a výstup sítě.

Graf – Pomocí této komponenty je možné zobrazovat posloupnost číselných hodnot jako čárový graf. Graf nalezne uplatnění například při zobrazení aktuální energie při počítaní odezvy sítě.

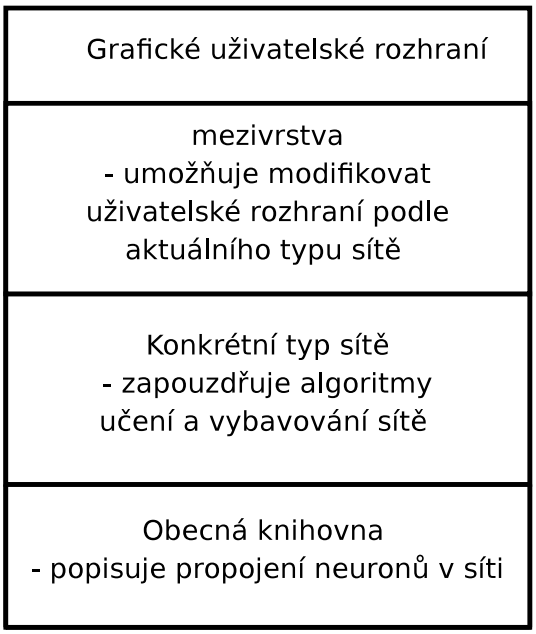
Znázornění topologie – V kapitole 4.6.1 o vizualizaci topologie sítě bylo popsány vlastnosti, které tato komponenta poskytuje. Byla navržena jako interaktivní, tudíž neposkytuje pouze informace o aktuálním stavu, ale umožňuje uživateli také měnit tento stav. Například pomocí přetažení kurzoru při podržení tlačítka myši z jednoho neuronu na druhý vytvoří spoj mezi neurony. Podobně při klepnutí tlačítka myši na neuron vyvolá dialogové okno, které uživateli umožní měnit vlastnosti neuronu – práh citlivosti a aktuální hodnotu výstupu.

4.8 Vrstva mezi aplikací a neuronovou sítí

Vrstva slouží k propojení knihovny a aplikace. Poskytuje aplikaci přístup k instanci neuronové sítě. Tvoří v podstatě unifikované rozhraní umožňující aplikaci jednotnou práci se

všemi druhy sítí. Je navržena tak, aby umožňovala modifikovat grafické uživatelské rozhraní aplikace podle potřeby aktuálního typu sítě, se kterou uživatel pracuje. Například implementuje vlastní komponenty pro učení a odezvu sítě.

Architektura celé navržené aplikace je na obrázku 4.4. Mezivrsta zapouzdřuje celou knihovnu.



Obrázek 4.4: Architektura navržené aplikace

Kapitola 5

Implementace

Kapitola je věnována postupu při implementaci aplikace. Nejprve bude objasněn přístup při výběru programovacího jazyka a následně popsány implementační podrobnosti zajímavých částí aplikace. Především se zaměřím na způsob, jakým je možné aplikaci rozšířit o další typy sítí.

5.1 Výběr programovacího jazyka

Jedním z hlavních požadavků zadání byla přenositelnost aplikace, především mezi operačními systémy Microsoft Windows a GNU/Linux. Další požadavek na zvolený jazyk pramení z návrhu – musí být objektově orientovaný. Dále byla požadována možnost tvořit netriviální grafické uživatelské rozhraní (GUI) buď přímo pomocí prostředků jazyka, nebo pomocí externí knihovny. Nakonec byl zvolen jazyk Java a jeho GUI toolkit Swing, protože splňuje kladené požadavky.

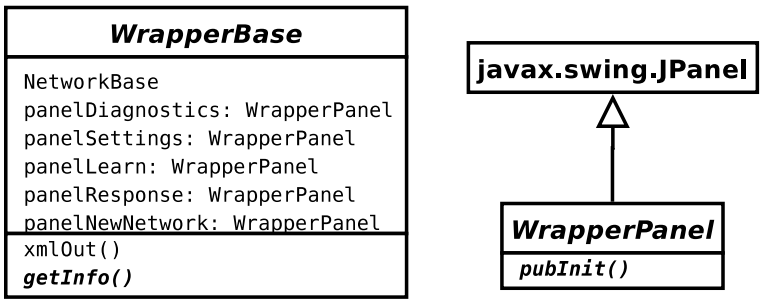
Nevýhodou může být nižší rychlost výsledné aplikace, protože Java je jazyk interpretovaný ve virtuálním stroji. Aplikace však nebude sloužit k rozsáhlým náročným výpočtům, nýbrž k vizualizaci pro studijní účely, kde požadavek na rychlost není tak kritický.

5.2 Hlavní okno aplikace

Třída `MainWindow` popisující hlavní okno aplikace je implementována jako singleton. Její konstruktor je definován jako privátní. Přístup k hlavnímu oknu lze získat volaním statické metody `getInstance()`. Používá se při ní pozdní inicializace (lazy initialization) – objekt se vytvoří až při prvním zavolání `getInstance()`.

Jednotlivé prvky uživatelského rozhraní jsou při svém vytvoření registrovány v hlavním okně. Toho lze využít především v situacích, kdy akce vyvolaná v jedné komponentě mění obsah jiné. Například stisknutí tlačítka pro výpočet odezvy sítě aktualizuje zobrazení výstupu. Zavoláním metody `MainWindow.getInstance()` se zpřístupní rozhraní hlavního okna, odkud je možné přistoupit ke komponentě zobrazující výstup sítě a její obsah aktualizovat.

Výhoda návrhového vzoru singleton a registrace jednotlivých kompoment spočívá ve zpřehlednění kódu. Reference na ostatní prvky jsou globálně přístupné skrze hlavní okno aplikace, a proto není potřeba je uchovavat ve všech komponentách.



Obrázek 5.1: Rozhraní zásuvných modulů

5.3 Systém zásuvných modulů

Architektura aplikace byla popsána v kapitole věnující se návrhu (4.8). Tato část je zaměřena na konkrétní implementaci rozhraní zásuvných modulů a postup využitý pro jejich nahrávání. Na obrázku 5.1 je vyobrazen diagram tříd rozhraní zásuvného modulu. Všechny použité zásuvné moduly musí být odvozeny od abstraktní třídy `WrapperBase`, která poskytuje přístup k neuronové sítí (atribut `NetworkBase`). Především však může obsahovat konkrétní implementace prvků uživatelského rozhraní. Pro reprezentaci těchto uživatelských prvků byla vytvořena abstraktní třída `WrapperPanel`. Jednotlivé prvky musí být odvozeny právě od této třídy.

Při vytvoření určitého okna aplikace zjistí, zda právě používaný zásuvný modul nedisponuje vlastním prvkem. Například při vytvoření okna pro ovládaní neuronové sítě (`NetControl`) se zkontroluje, jestli zásuvný modul definuje prvky `panelLearn`, `panelResonse`, `panelDiagnostics` nebo `panelSettings`. Pokud definuje zmíněné prvky, namísto inicializace nativních komponent dojde k zavolání inicializace panelu, který je součástí zásuvného modulu.

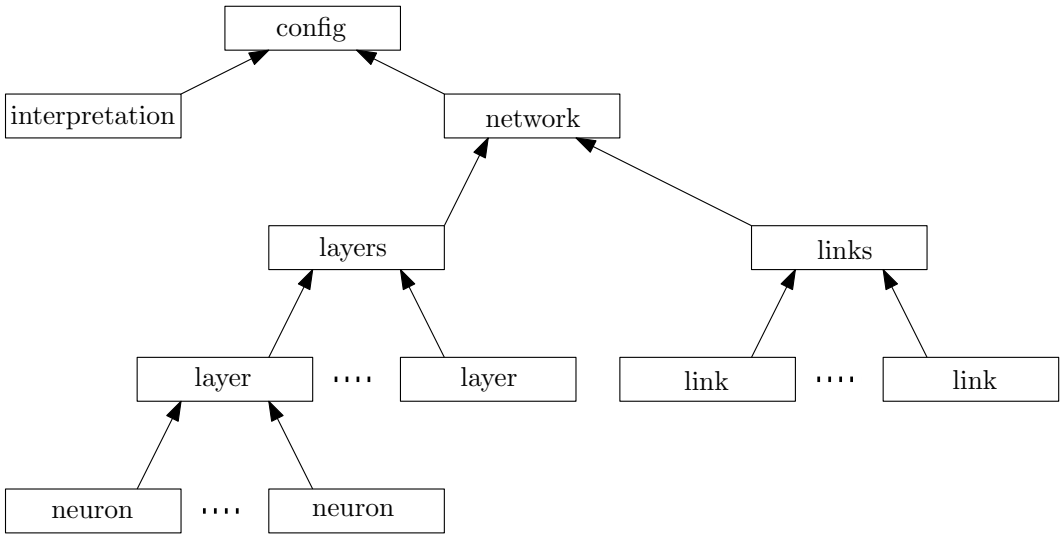
Situace je popsána v následujícím pseudokódu. V kódu je zohledněn pouze panel pro učení sítě. Postup u ostatních panelů je analogický.

```
// konstruktor
public NetControl()
{
    ...
    // test, jestli zásuvný modul obsahuje vlastní definici prvku
    if (actualWrapper.panelLearn != null)
    {
        // pokud ano
        this.panelLearn = actualWrapper.panelLearn;
        this.panelLearn.pubInit();
    }
    else
    {
        // jinak se použije výchozí
        ... standardní inicializace ...
    }
}
```

Za zmínku stojí také metoda abstraktní metoda `getInfo()`, kterou například využívá prvek pro zobrazení informací o síti (4.7).

5.4 Ukládání stavu sítě

Pro uložení stavu sítě a tedy i celé aplikace jsem se rozhodl na základě doporučení mého vedoucího použít formát XML. K tomuto účelu slouží metoda `xmlOut()` třídy `WrapperBase`. Kořenovým elementem XML souboru, který popisuje uložený stav je element `config`.



Obrázek 5.2: Struktura souboru uchovávajícího stav sítě

Popis jednotlivých elementů z obrázku 5.2 je v následujícím výčtu:

interpretation – Uchovává nastavení interpretace sítě.

network – Zapouzdřuje další elementy popisující uspořádání sítě.

layers – Obsahuje elementy `layer` popisující vrstvy.

layer – Obsahuje elementy `neuron` odpovídající jednotlivým neuronům.

neuron – Atributy elementu přesně definují typ, vlastnosti a identifikaci neuronu.

links – Obsahuje definice vazeb mezi neurony

link – Odpovídá konkrétní vazbě mezi neurony.

Při načítání stavu sítě je nejprve zkontrolována syntaxe XML souboru. Pokud je v pořádku, vytvoří se nová instance třídy `Network`. Poté jsou postupně do nové sítě přidávány jednotlivé vrstvy a neurony. Každý element `neuron` obsahuje atribut `id`, který jej jednoznačně identifikuje. Element `link` reprezentující vazbu obsahuje vždy atributy `from` a `to` obsahující reference na zdrojový a cílový neuron vazby.

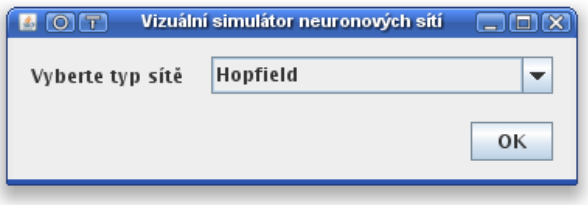
Kapitola 6

Uživatelský návod

V této kapitole jsou názorně popsány jednotlivé komponenty uživatelského rozhraní. Jednotlivé zásuvné moduly si funkčnost aplikace přizpůsobují tak, aby bylo možné se sítí optimálně pracovat. Obecné vlastnosti, jako je umístění komponent v hlavní nabídce a jejich účel, se však nemění.

6.1 Popis aplikace

Ihned po spuštění se zobrazí okno, které vyzve uživatele k vybrání jednoho z dostupných zásuvných modulů.



Po zvolení zásuvného modulu se zobrazí hlavní okno aplikace. Hlavní nabídka obsahuje následující položky:

- **Soubor**

  Vybrat zásuvný modul – Zavře hlavní okno a zobrazí výzvu ke zvolení jiného zásuvného modulu.

  Nahrát – Umožní nahrát dříve uloženou síť.

  Uložit – Uloží stav sítě na disk.

  Konec – Ukončí aplikaci.

- **Síť**

  Nová síť – Zobrazí se dialogové okno, kde uživatel v závislosti na použitém zásuvném modulu nastaví vlastnosti nové sítě, která se poté vytvoří.

  Přidat Neuron – Přidá do existující sítě nový neuron.

- **Okna** – Jsou podrobněji popsána v kapitole 6.1.1.

  Informační okno – Obecné informace o aktuální síti.

  Ovládací panel – Učení, odezva, diagnostika a nastavení sítě.

  Energie sítě – Graf zobrazující časový průběh energie sítě.

  Vstup / Výstup sítě – Interpretace vstupu a výstupu sítě.

  Topologie sítě – Grafická reprezentace sítě.

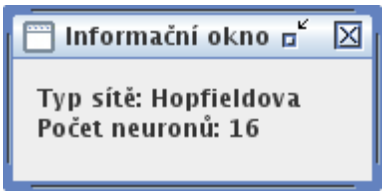
- **Nástroje**

  Vytvořit vektory – Tvrorba trénovacích množin, vstupů a diagnostických dat pro sítě.

- **Nápověda** – Nápověda k aplikaci.

6.1.1 Podrobný popis komponent

Informační okno

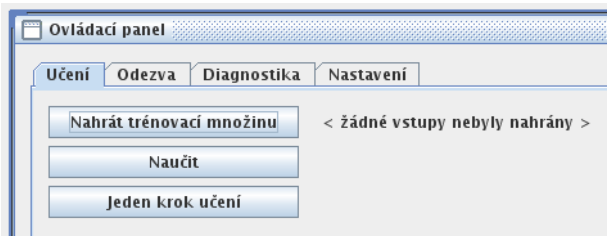


Zobrazuje obecné informace o aktuální síti, závislé na použitém zásuvném modulu. Zpravidla se uživatel dozví informace o velikosti vstupu a výstupu sítě, počtu neuronů apod.

Ovládací panel

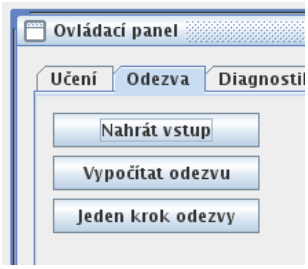
Obsahuje čtyři záložky jejichž přesný obsah je závislý na zvoleném zásuvném modulu. Například u Hopfieldovy sítě je následující:

- **Učení**



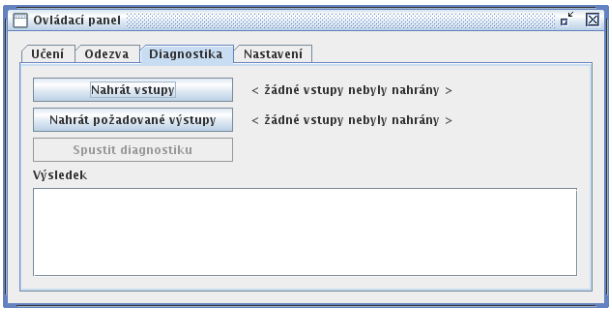
  Učení sítě probíhá v následujících krocích. Pomocí tlačítka „Nahrát trénovací množinu“ uživatel vybere trénovací množinu sítě. Učení pak lze provézt plně automaticky pomocí tlačítka „Naučit“ nebo lze krokovat pomocí tlačítka „Jeden krok učení“.

- **Odezva**



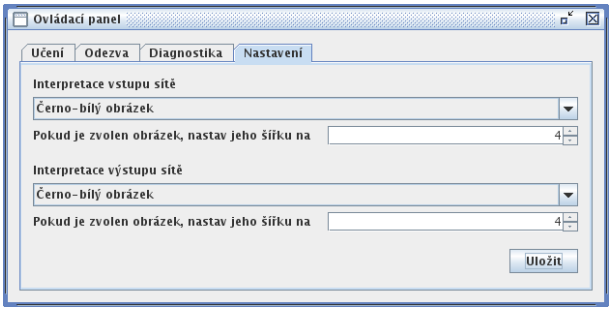
Tlačítko „Nahrát vstup“ slouží k nastavení vstupu sítě. Následně je možné odezvu sítě krokovat pomocí tlačítka „Jeden krok odezvy“ nebo získat najednou tlačítkem „Vypočítat odezvu“.

- **Diagnostika**



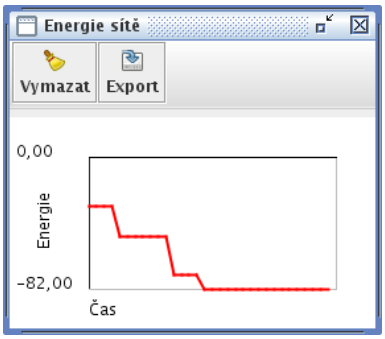
Diagnostika se spouští tlačítkem „Spustit diagnostiku“. Ještě předtím je však nutné nahrát vstupy a požadované výstupy. Síti jsou postupně předkládány jednotlivé vstupy, počítány výstupy a ty pak srovnávány s definovanými požadovanými výstupy. Diagnostika počítá poměr úspěšně rozpoznaných vzorů k celku.

- **Nastavení**



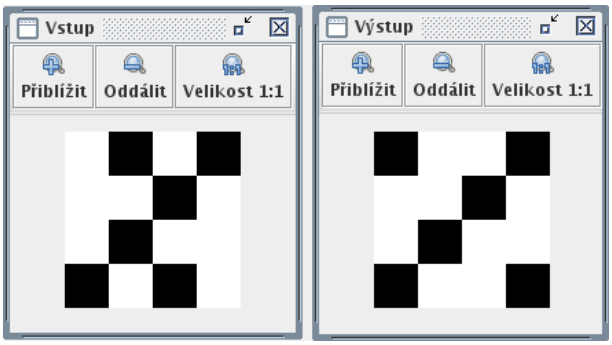
V tomto panelu je obsaženo nastavení interpretace vstupů a výstupů sítě.

Energie sítě



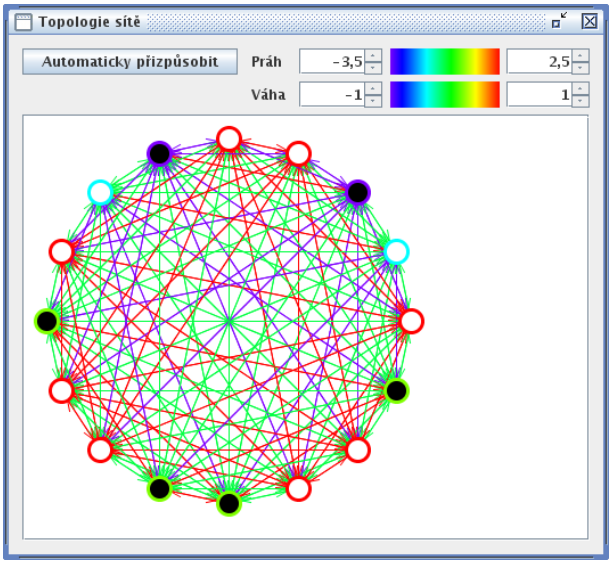
Komponenta zobrazuje časový průběh energie sítě. Slouží především k demonstraci záporného nebo nulového přírůstku energie v každém kroku odezvy u Hopfieldovy sítě a sítě BAM. Umožňuje také export zobrazovaných hodnot do textového formátu, kde jsou jednotlivé hodnoty odděleny mezerami. Tento výstup může být posléze použit například programem `gnuplot`.

Interpretace vstupu a výstupu



V závislosti na nastavení v Ovládacím panelu je zde interpretován vstup a výstup sítě.

Topologie sítě



Komponenta umožňuje nejen zobrazovat stav sítě, ale také jej interaktivně měnit. Klepnutím na jakýkoliv neuron se zobrazí dialogové okno, kde je možné měnit aktuální jeho výstup, práh citlivosti nebo neuron ze sítě zcela odstranit. Zmáčknutím tlačítka myši a přetažením kurzoru mezi dvěma neurony lze vytvořit spojení mezi neurony. Pokud již spoj dříve existoval, uživatel u něj může změnit hodnotu váhy nebo spoj odstranit.

Velikosti vah spojů i prahy citlivosti jsou převedeny do barevného spektra. Vysoké hodnotě váhy (respektive citlivosti) odpovídá červená barva, nízké hodnotě váhy (respektive citlivosti) pak modrá barva. Tlačítkem „Automaticky přizpůsobit“ je možné mapování číselných hodnot do barevného spektra optimálně přizpůsobit aktuálnímu stavu sítě.

Kapitola 7

Závěr

V této práci byla po teoretickém úvodě, který se zabýval výkladem pojmů jako neuron, neuronová síť a asociativní paměť, navrhnuta knihovna a aplikace sloužící pro simulaci chování neuronových sítí, které mohou být použity jako asociativní paměti. Knihovna i aplikace byly následně implementovány. Byl kladen důraz na to, aby bylo možné vzniklý systém dále udržovat a v budoucnu jej rozvíjet. Aplikaci lze tedy pomocí systému zásuvných modulů doplnit o nové neuronové sítě i specifické prvky uživatelského rozhraní.

Vzniklý systém slouží především ke studijním účelům. Uživatel může vytvářet pro neuronové sítě vlastní trénovací množiny a sledovat stav sítě v průběhu učení i odezvy na definovaný vstup. Díky použití platformy Java je aplikace snadno přenositelná. Úspěšně byla testována v prostředí operačních systémů Microsoft Windows XP a GNU/Linux.

V budoucnu lze aplikaci rozšířit o nové typy sítí. Rozhraní aplikace umožňuje vytváření zásuvných modulů, které podle potřeb modifikují uživatelské rozraní a definují síť, algoritmy učení a další vlastnosti. Rovněž se nabízí možnost doplnit aplikaci o podporu spolupráce s jinými již existujícími nástroji pro práci s neuronovými sítěmi.

Literatura

[1] Alexey Dekhtyarenko; Dmitry Gorodnichy. Associative Neural Network Library. `http://www.perceptual-vision.com/memory/pinn/`, naposledy navštíveno 30. 4. 2008.

[2] Alexey Dmitriev; Daniel Sanchez; André El-Ama; Patrick Kursawe. SNNS-development. `http://developer.berlios.de/projects/snns-dev/`, naposledy navštíveno 30. 4. 2008.

[3] Andreas Zell. Java Neural Network Simulator. `http://www-ra.informatik.uni-tuebingen.de/software/JavaNNS/welcome_e.html`, naposledy navštíveno 30. 4. 2008.

[4] Andreas Zell. SNNS. `http://www-ra.informatik.uni-tuebingen.de/software/snns/welcome_e.html`, naposledy navštíveno 30. 4. 2008.

[5] Fakhreddine O. Karray; Clarence de Silva. *Soft Computing and Intelligent System Design*. Pearson Education Limited, Hardback, 2004. ISBN 0-321-11617-8.

[6] František Zbořil. Přednášky předmětu Soft Computing v akademickém roce 2007/2008 na FIT VUT v Brně. `http://www.fit.vutbr.cz/study/courses/SFC/`, naposledy navštíveno 30. 4. 2008.

[7] Jochen Frőhlich. Neural Networks with Java. `http://www.nnwj.de/`, naposledy navštíveno 30. 4. 2008.

[8] Steffen Nissen. FANN. `http://leenissen.dk/fann/`, naposledy navštíveno 30. 4. 2008.

[9] Vladimír Mařík; Olga Štěpánková; Jiří Lažanský a kol. *Umělá inteligence (4)*. Academia, Praha, 2003. ISBN 80-200-1044-0.