VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

POROVNÁNÍ KNIHOVEN PRO PRÁCI S UMĚLÝMI NEURONOVÝMI SÍTĚMI

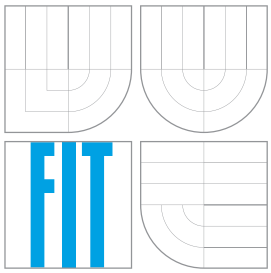
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ZDENĚK DOHNAL

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

POROVNÁNÍ KNIHOVEN PRO PRÁCI S UMĚLÝMI NEURONOVÝMI SÍTĚMI

COMPARISON OF LIBRARIES OF ARTIFICIAL NEURAL NETWORKS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ZDENĚK DOHNAL

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ŠTĚPÁN DALECKÝ

BRNO 2015

Abstrakt

Práce se zabývá porovnáním knihoven pro práci umělými neuronovými sítěmi. Je vysvětlena základní teorie neuronu, neuronových sítí a jejich učení. Pro experimenty je vybrán vícevrstvý perceptron, Kohonenova síť a Hopfieldova síť. Následně jsou zvolena kritéria porovnávání jako je licence, komunita nebo poslední aktualizace, a pro experimenty jsou vybrány úlohy aproximace funkce pro vícevrstvý perceptron, asociace pro Hopfieldovu síť a shlukování pro Kohonenovu síť. Následně jsou implementovány programy s využitím daných knihoven. Závěrem je porovnání výsledků experimentů a kritérií.

Abstract

This thesis is about comparison of libraries of artificial neural networks. Basic theory of neuron, neural networks and their learning algorithms are explained here. Multilayer perceptron, Self organizing map and Hopfield net are chosen for experiments. Criteria of comparison such as licence, community or last actualization are designed. Approximation of function, association and clustering are chosen as task for experiments. After that, there is implementation of applications using chosen libraries. At the end, result of comparison and experiment are evaluated.

Klíčová slova

umělé neuronové sítě, neuron, vícevrstvý perceptron, Kohonenovy sítě, Hopfieldovy sítě, algoritmus zpětného šíření chyby

Keywords

artificial neural networks, neuron, multilayer perceptron, Kohonen maps, Hopfield nets, Backpropagation

Citace

Zdeněk Dohnal: Porovnání knihoven pro práci s umělými neuronovými sítěmi, bakalářská práce, Brno, FIT VUT v Brně, 2015

Porovnání knihoven pro práci s umělými neuronovými sítěmi

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Štěpána Daleckého.

.....
Zdeněk Dohnal
20. května 2015

Poděkování

Rád bych poděkoval panu Ing. Štěpánu Daleckému za vstřícnost, odbornou pomoc a trpělivost při konzultacích bakalářské práce. Dalšími, o kterých bych se zde rád zmínil a poděkoval jim, jsou rodiče a přátelé, kteří mě v této pouti bakalářským studiem podporovali.

© Zdeněk Dohnal, 2015.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

Kapitola 1

Úvod

V moderní době stále více roste důležitost umělé inteligence. Je to perspektivní odvětví, které není dostatečně prozkoumáno, i když její rozvoj by ušetřil člověku velké množství sil. Proto se zvyšuje poptávka po knihovnách, které by práci s umělou inteligencí ulehčily a zjednodušily vývoj aplikací využívající umělou inteligenci. Cílem práce je zjistit, jaké knihovny existují a porovnat je na základě jejich vlastností a zvolených experimentů, za účelem zjištění výkonnosti knihovny.

Umělé neuronové sítě jsou jednou z odvětví umělé inteligence, které si bere za vzor živočišnou nervovou soustavu. Aspektem, který výzkumníky zajímá na nervové soustavě, je schopnost adaptability neboli schopnosti se přizpůsobit. Teorie neuronu a neuronových sítí je důležitá pro tuto práci a proto v první části se práce věnuje základním pojmům a rozdělením neuronových sítí a jejich učením.

Důležitou částí práce je samotný návrh kritérií porovnávání a úloh, se kterými budu experimentovat. U subjektivních kritérií je nutné objasnit, proč je knihovna hodnocena, jak je hodnocena. Úlohy musejí vycházet z navržených sítí, které podporují testované knihovny.

Následně se vyberou knihovny, které se budou porovnávat. Je důležité pro porovnávání, aby porovnávané knihovny podporovaly stejné aktivační funkce a budou testovány se stejnými parametry. Na závěr proběhne porovnání a zhodnocení výsledků.

Kapitola 2

Umělé neuronové sítě

Kapitola se bude zabývat teorií umělých neuronových sítí. Popsána bude návaznost na neurobiologii, jelikož vycházejí z konceptu skutečné buňky v nervové soustavě živočišných druhů, následovat bude popis struktury umělého neuronu.

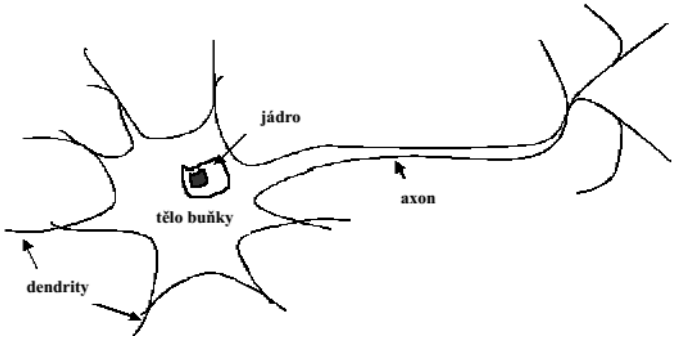
Následně se práce bude zaměřovat na tři druhy učení používané v umělých neuronových sítích, což je korelační, soutěživé a adaptační učení. Zbývající podkapitoly se budou týkat jednotlivých typů umělých neuronových sítím konkrétně vícevrstvého perceptronu, Hopfieldovy sítě a Kohonenovy sítě. Tyto jmenované sítě jsou probrány zvlášť, protože jsou implementovány ve vybraných knihovnách. Další typy sítí jsou zmíněny v poslední části této kapitoly.

2.1 Neuron

Předtím než se budeme zabývat umělými neuronovými sítěmi, je nutné definovat jejich základní prvek, *neuron*. Jedná se o buňka nervové soustavy živého organismu, který přijímá podněty od ostatních neuronů a při překročení určitého prahu se neuron aktivuje. V rámci této práce je nutné funkci a strukturu neuronu pochopit do větší hloubky. Tento biologický neuron je znázorněn na obrázku 2.1. Z pohledu neurobiologie je neuron základním prvkem nervové soustavy, který se skládá z několika částí[13]:

- sóma neboli tělo neuronu,
- dendrity – vstupy,
- axon – výstup,
- synapse – rozhraní mezi axonem prvního neuronu a dendritem druhého neuronu.

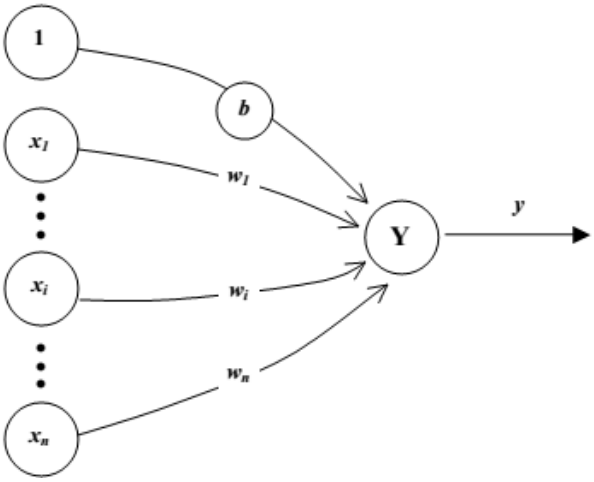
Šíření informace probíhá pomocí membrány, kterou jsou obaleny sóma a axon, která dokáže vytvořit elektrický impuls. Tento impuls je přenášen z axonu na dendrity skrz synapse, které svojí propustností určují míru podráždění dalších neuronů. Pokud takto podrážděný neuron při dosažení určité meze, prahu, sám generuje elektrický impuls a tím dál šíří informaci. Propustnost synapse se s každým průchodem signálu změní.[10]. Tyto vlastnosti a funkce se přenášejí do matematického modelu.



Obrázek 2.1: Biologický model neuronu[8]

Umělý neuron

Umělým neuronem, který je znázorněn na obrázku 2.2, rozumíme matematický model vycházející ze své biologické předlohy z obrázku 2.1. Dendrity jsou modelovány jako vstupní vektor. Propustnost synapse jsou v matematickém modelu váha neuronu a váhy všech synapsí jsou modelovány jako váhový vektor. U každého neuronu je libovolně zvolené x_0 násobeno váhou w_0, která se nazývá *bias* neuronu. Tato váha určuje práh, kdy dojde k aktivaci neuronu. Axon je zastoupen výstupní hodnotou[8].



Obrázek 2.2: Umělý neuron[8]

Bázové funkce

Vnitřní funkci neuronu určuje bázová funkce, která popisuje způsob, jak se pracuje se vstupními vektory a váhovými vektory neuronu. Výstupem funkce je vnitřní potenciál neuronu, který je vstupní hodnotou funkce aktivační. V umělém neuronu se mohou vyskytovat dva druhy bázových funkcí[14], lineární bázová funkce a radiální bázová funkce.

Lineární bázová funkce

Jedná se o základní bázovou funkci, která je založena na váženém součtu vstupního vektoru. Matematicky toto lze zapsat vzorcem 2.1, kde w_i je i-tý prvek váhového vektoru a x_i je i-tý prvek vstupního vektoru. Konstanta n určuje dimenzi vektorů.

$$f(\vec{x}) = \sum_{i=1}^{n} w_i x_i \tag{2.1}$$

Radiální bázová funkce

Radiální bázová funkce je určena významným bodem, který nazýváme střed. Pro tuto funkci platí, že pro argumenty, které jsou ve stejné vzdálenosti od středu, vrací stejné funkční hodnoty[10]. Jedná se výpočet Eukleidovské vzdálenosti vektorů[4]. Matematicky lze funkci zapsat vzorcem 2.2, kde symbol u je vnitřní potenciál, $\vec{x}$ vstupní vektor a $\vec{w}$ váhový vektor. w_i je prvek váhového vektoru a x_i představuje prvek vstupního vektoru, konstanta n je dimenzí vektorů.

$$u = \|\vec{x} - \vec{w}\| = \sqrt{\sum_{i=1}^{n} (x_i - w_i)^2} \tag{2.2}$$

Aktivační funkce

Aktivační funkce neuronu určuje, jaký bude výstup neuronu v závislosti na vnitřním potenciálu. Aktivační funkce jsou rozdělené podle toho, s jakou bázovou funkcí se používají. Pro lineární bázovou funkci je tabulka 2.1[11], pro radiální je tabulka 2.2[12]. V tabulkách jsou jejich vzorce, kde u je vnitřní potenciál a y je výstupem neuronu[8], a grafy.

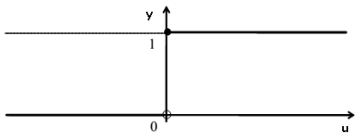
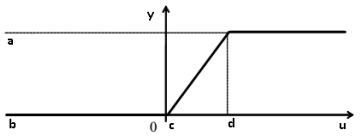
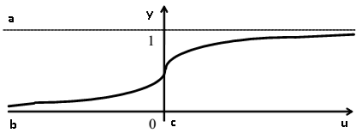
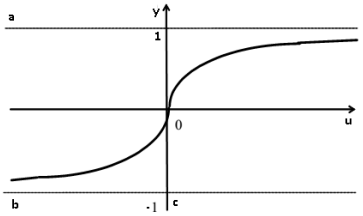
Matematický zápis umělého neuronu

Nyní byly vysvětleny všechny součásti umělého neuronu, takže jeho funkci lze matematicky vyjádřit pomocí vzorce 2.3[13]:

$$y = g(u) = g(f(\vec{x})) \tag{2.3}$$

, kde ve vzorci je:

- symbol y je výstup neuronu,
- funkce g je aktivační funkcí, u vnitřní potenciál neuronu,
- funkce f je bázová funkce,
- vektor $\vec{x}$ je vstupním vektorem.

Aktivační funkce	Vzorec	Graf
Skoková	$y = \left\{ \begin{array}{rl} 1, & u \geq \theta \\ y_{old}, & u = \theta \\ 0, & u \leq \theta \end{array} \right\}$	
Rampová	$y = \left\{ \begin{array}{rl} a, & u < c \\ b, & u > d \\ a + \frac{(b-a)(u-c)}{d-c}, & c \leq u \leq d \end{array} \right\}$	
Sigmoida	$y = a + \frac{b-a}{1+e^{-\lambda u+c}}$	
Hyperbolický tangens	$y = \tanh(\lambda u)$	

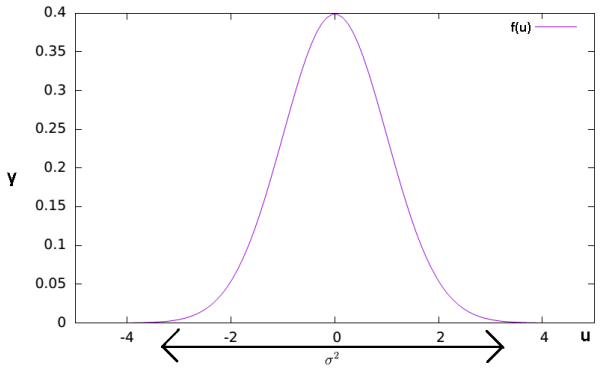
Tabulka 2.1: Aktivační funkce pro LBF

Učení neuronu

Před samotným učením neuronu je nutné vysvětlit pojmy *učení s učitelem* a *učení bez učitele*. U těchto učení rozlišujeme soubor trénovacích dat, se kterým neuron pracuje ve fázi učení, a soubor testovacích dat, na kterém neuron řeší daný problém za použití naučených znalostí. Při učení s učitelem je přítomen učitel, který dokáže určit, zda systém pracuje správně, dokáže určit požadovaný výstupní vektor nebo dokáže zjistit velikost chyby. Tímto učitelem je požadovaný výstupní vektor, který je spojen ve dvojici se vstupním vektorem[13]. Zapsáno formálně vzorcem 2.4, kde T je trénovací množina, $\vec{x}$ je vstupní vektor a $\vec{y}$ požadovaný výstupní vektor. Protikladem je *učení bez učitele*, kdy učení musí spoléhat vztahy a informace získané zkoumáním trénovacích dat.

$$T = (\vec{x}, \vec{y}) \tag{2.4}$$

Samotný neuron se musí před nasazením na řešení problému naučit, jak tento problém řešit. Zavádí se dvě možné označení neuronu, podle kterých se odlišují i algoritmy učení pro jeden neuron. Jedná se o *perceptron* a *Adaline*[13]. V případě perceptronu to je *Rosenblattovo*

Aktivační funkce	Vzorec	Graf
Gaussova	$y = e^{-\left(\frac{u}{\sigma}\right)^2}$	

Tabulka 2.2: Aktivační funkce pro RBF

pravidlo učení perceptronu popsané vzorcem 2.5[13], které bude vysvětleno. Existují další úpravy algoritmu učení perceptronu, ale pro popis principu je dostačující tento základní. Učení Adalin je popsáno v literatuře[13].

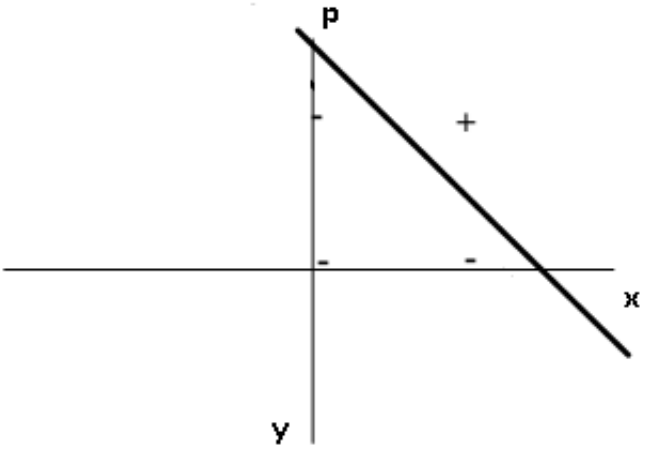
$$\begin{aligned} &\vec{w}(0) = libovolné \\ &\vec{w}(k) = \vec{w}(k-1) + \gamma\left(d(k) - y(k)\right)\vec{x}(k) \quad k = 1, 2, \ldots \\ &\left(\vec{x}(k), d(k)\right) \in T_x \end{aligned} \tag{2.5}$$

Rosenblattovo pravidlo pro učení perceptronu je učení s učitelem, kdy se váhy mění tehdy, když se liší požadovaný a aktuální výstupní vektor. Učení probíhá do té doby, dokud není dosaženo požadované přesnosti. Toto učení je matematicky popsáno vzorcem 2.5, kde T_x je trénovací množina, kam patří dvojice $\vec{x}(k)$, která zastupuje vstupní vektor v kroku učení k a $d(k)$ požadovaný výstupní vektor v kroku učení k. Symbol $\vec{w}(0)$ představuje váhový vektor na začátku učení, k značí krok učení, γ je koeficient učení, který je možné volit libovolně, $y(k)$ je skutečný výstupní vektor v kroku k.

Takto naučený perceptron se schopen řešit úlohy, které se dají vyřešit proložením prostoru přímkou. Takové problémy nazýváme lineárně separovatelné. Mezi takovéto úlohy patří logické funkce `AND` a `OR`. Pro tyto funkce perceptron proloží prostor přímkou, která prostor rozdělí na dvě poloroviny, pomocí kterých pak dokáže rozhodovat. Tento princip je znázorněn na obrázku 2.3.

Důvodem, proč funkci perceptronu pro takovéto úlohy lze znázornit přímkou, je schopnost zápisu lineární bázové funkce ve tvaru, který odpovídá obecné rovnici přímky z analytické geometrie. Matematický zápis odpovídá vzorci 2.6, kde w_i prvek váhového vektoru a x_i prvek vstupního vektoru, $w_0 x_0$ je součin prahu w_0 a libovolné hodnoty x_0. U úloh `AND` a `OR` se používá vstupní vektor o dvou dimenzích, takže po rozvoji součtu ve výsledku získává vzorec tvar obecné rovnice přímky.

$$w_0 x_0 + \sum_{i=1}^{i} w_i x_i = 0 \tag{2.6}$$



Obrázek 2.3: Znázornění funkce perceptronu[8]

2.2 Umělé neuronové sítě

Umělé neuronové sítě jsou struktury o více než jednom neuronu a tyto neurony mezi sebou jsou propojeny. Samotná neuronová síť je deterministická, což znamená, že neobsahuje pravděpodobnost. Matematicky lze zapsat neuronovou síť vzorcem 2.7, kde platí, že vstupní vektor $\vec{x}$ dokáže převést na výstupní vektor $\vec{y}$ pomocí funkce $f(\vec{x})$, která představuje funkci neuronové sítě.

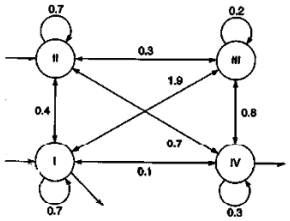
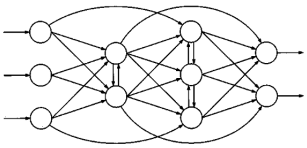
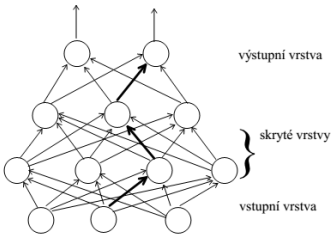
$$\vec{y} = f(\vec{x}) \tag{2.7}$$

Takové sítě dokáží řešit složitější úlohy než samotný neuron, ale daná síť musí projít složitější fází učení. Druhů učení existuje několik, a proto je možno neuronové sítě podle nich dělit. V této části jsou vyjmenovány všechny zmíněná dělení až na učení, kterým je věnována další podkapitola. Kritéria jsou[13]:

- *podle architektury sítě*,
- *podle proměnlivosti topologie sítě*,
- *podle použití*,
- *podle implementace*,
- *podle výpočtu*.

Dělení dle architektury

Vybrané architektury jsou v tabulce 2.3. Mezi další druhy patří plně propojené sítě, které jsou podobné jako plně propojené symetrické, ale vztahy mezi dvěma neurony jsou reprezentovány dvěma spojeními o různých vahách, sítě acyklické, které neobsahují ve své architektuře žádnou smyčku, a sítě rekurentní, které obsahují alespoň jednu smyčku[13].

Neuronová síť	Graf	Popis
Plně propojená symetrická[13]		Tato síť má vlastnost, že váha mezi neuronem i a j je stejná jako váha mezi neuronem j a i. Jinak je strukturou stejná jako plně propojená síť.
Vrstvová[13]		Tyto sítě jsou specifické tím, že neurony nejsou propojeny s předešlými vrstvami. Neurony jsou uspořádány do vrstev. Zvláštním případem jsou jednovrstvé sítě.
Dopředná[13]		Dopřednou sítí rozumíme síť, kde z jedné vrstvy neuronů existují spojení pouze do bezprostředně následující.

Tabulka 2.3: Architektury vybraných druhů sítí

Dělení podle použití

V této části je prostor věnován použitím, podle kterých jsou implementovány úlohy pro porovnávání knihoven. Druhy použití, které jsem vybral, jsou asociace, aproximace funkce a shlukování.

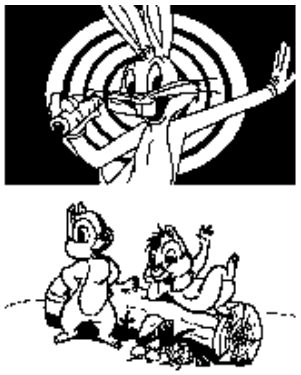
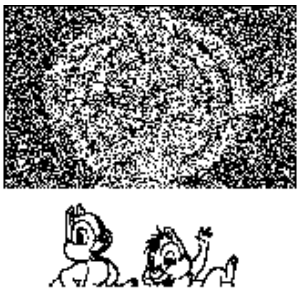
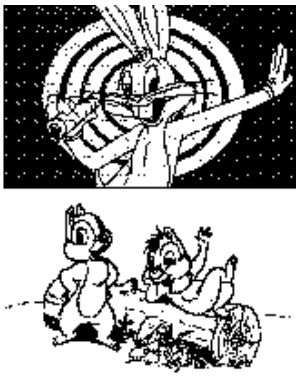
Asociace

Sítě určené k tomuto použití po předložení vstupního vzorku spustí generování specifického výstupního vzoru. Při asociaci existují dva druhy úloh. Prvním je *auto-asociace*, kdy na vstupu úlohy se předpokládá poškozený vzorek a rekonstruuje ho. Síť nastavuje váhy reprezentující mapování vstupních vzorků na výstupní. Jde o jednu vrstvu, kde každý neuron představuje jednu dimenzi vstupu a při rekonstrukci se opakovaně upravují výstupy neuronů, dokud se výstupní vzorek nepřestane měnit. Tento princip lze znázornit vzorcem 2.8, kde $f(\vec{x})$ je funkce se vstupním vektorem $\vec{x}$. Její funkce je znázorněna obrázky v tabulce 2.4.

$$f(\vec{x}) = \vec{x} \tag{2.8}$$

Druhým druhem úloh asociace je *hetero-asociace* , kdy výstupem je libovolný vzor, který je asociován se vstupním vzorkem. Stejně jako u auto-asociace jde o úlohu nastavování vah reprezentující mapování vstupních vzorků na výstupní, ale u úloh hetero-asociace jsou vrstvy dvě, když druhá generuje odpovídající výstupní vzorek[13]. Matematicky lze tento princip vyjádřit vzorcem 2.9, kde $f(\vec{x})$ je funkce se vstupním vektorem $\vec{x}$ a $\vec{y}$ je vektor asociovaných vzorků, které jsou přiřazené ke vstupnímu vektoru.

$$f(\vec{x}) = \vec{y} \tag{2.9}$$

Původní	Poškozený	Rekonstruovaný
		

Tabulka 2.4: Příklad autoasociace obrazu

Aproximace funkce

Sítě pro aproximaci funkce jsou sítě, které zjišťují nebo vytvářejí funkci, která generuje přibližně stejné výstupy z daných vstupů z trénovacích dat. Tyto úlohy existují, protože mnoho výpočetních modelů je znázorněno jako mapovací funkce numerických vstupů na numerické výstupy. Tyto vstupy a výstupy jsou známé z trénovacích dat, ale funkce, která popisuje proces generování výstupu, známá není[13].

Shlukování

Tyto sítě pro shlukování seskupují vektory s určitou podobností do předem neznámých shluků. V úlohách jsou dostupné soubor vzorků a vztahy, které mohou být odvozeny ze vzorků. Mechanismy shlukování jsou většinou postaveny na měření vzdálenosti. Každý objekt v úloze je reprezentován vektorem vlastností a podobné objekty mají podobné hodnoty vlastností. Tyto podobné objekty se shromažďují do shluků podle podobných vlastností.[13].

Další druhy použití

Mezi další použití patří *klasifikace*[13], kdy sítě pro toto použití přiřadí vektoru do určité předdefinované třídy. Jiným je *vektorová kvantizace*[13], kdy sítě provádějí kompresi dat.

Existují také sítě pro použití k predikci, řízení výrobních procesu, optimalizací a prohledávání stavových prostorů[13].

Dělení dle druhu učení

V následující části bude probráno rozdělení umělých neuronových sítí podle algoritmu učení, kterým se síť učí. Dále podrobněji bude sekce zaměřena na nejznámější druh adaptivního učení, *algoritmus zpětného šíření chyby*. V případě soutěživého učení se jedná o *jednoduché soutěživé učení* a u korelačního učení princip *Hebbova učení*[12].

Adaptivní učení

Adaptivní učení je učením s učitelem, protože ke vstupním vektorům jsou k dispozici požadované výstupní vektory. Vstupní vektory se přikládají na vstup sítě a výstup sítě se porovnává s požadovaným. Získají se odchylky, které potom ovlivňují úpravu vah neuronů. Tento princip vychází z reálného světa, kdy živočichové jsou schopni učit se z vlastních chyb[13]. Mezi adaptivní učení patří algoritmus zpětného šíření chyby a několik jeho úprav jako algoritmus přizpůsobivého zpětného šíření chyby *(RPROP)*[9].

Soutěživé učení

Soutěživým učením myslíme učení bez učitele, které probíhá tak, že na vstup sítě se přikládají trénovací vektory, síť reaguje na vstup excitací vítězného neuronu, u kterého se upraví váhový vektor. V dalších modifikacích se mohou měnit i váhy okolních sousedů. To má za příčinu posun neuronů v prostoru blíže k trénovacím vektorům, což se dobře využívá u shlukovacích úloh. V *Kohonenových samoorganizovacích mapách*, popsané v části 2.3.3n, je použita heuristika s úpravou vah sousedů. Zde je popsán algoritmus jednoduchého soutěživého učení, upravený algoritmus pro Kohonenovy mapy je v sekci 2.3.3. Pro vnitřní potenciál neuronu j, který v tomto učení představuje vzdálenost od váhového vektoru od vstupního trénovacího vektoru, platí vzorec 2.10[13]:

$$u_j\left(k-1\right) = \|\vec{i_p} - \vec{w_j}\left(k-1\right)\| = \sqrt{\sum_{i=1}^{n}\left(i_{pi} - w_{ji}\left(k-1\right)\right)^2} \tag{2.10}$$

Ve vzorci $u_j\left(k-1\right)$ je vnitřní potenciál j neuronu v kroku učení $k-1$, $\vec{i_p}$ je vstupní trénovací vektor na indexu p v trénovací množině, $\vec{w_j}$ je váhový vektor neuronu j. Vzdálenost se pak vypočítá jako odmocnina ze sumy druhých mocnin rozdílů prvků vektorů i a w. Potom se zjistí vítězný neuron dle vzorce 2.11[13].

$$\|\vec{i_p} - \vec{w_j^*}\left(k-1\right)\| \leq \|\vec{i_p} - \vec{w_j}\left(k-1\right)\|, \quad j \in \{1,\ldots,m\} \tag{2.11}$$

Ve vzorci je symbol $\vec{i_p}$ je vstupní trénovací vektor v trénovací množině na indexu p, $\vec{w_j}$ je váhový vektor neuronu j, $\vec{w_j^*}$ je váhový vektor vítězného neuronu j^*, k je krokem učení a m je celkový počet neuronů. Pro změnu vah vítězného neuronu platí vztah 2.12[11].

$$\vec{w}_{j^*}\left(k\right) = \vec{w}_{j^*}\left(k-1\right) + \gamma\left(k\right)\left(\vec{i_p} - \vec{w}_{j^*}\left(k-1\right)\right), \tag{2.12}$$

Ve vzorci $\vec{w}_{j^*}$ je váhový vektor, který obsahuje váhu vítězného neuronu j^*. Symbol k je krokem učení. γ je koeficient učení. Učení probíhá, dokud váhy nedosáhnou ustáleného stavu.

Korelační učení

Korelační učení je učení s učitelem, kdy se na vstupní vektory aplikuje Hebbův princip, který je založen na myšlence, že váha mezi dvěma neurony je úměrná součinu jejich výstupů. Matematický zápis tohoto principu je vzorec 2.13[10].

$$\Delta w_{ij} = \gamma y_i y_j, \tag{2.13}$$

Ze vzorce se vyvozuje, že současná kladná aktivita váhu zvyšuje a opačná váhu snižuje. Ve vzorci 2.13 značí Δw_{ij} změnu váhy mezi neurony i, j, γ je koeficient učení a y_i, y_j jsou výstupy neuronů i, j. Učení končí po průchodu trénovacími daty, kdy konečná váha w_{ij} mezi neurony i, j je rovna součtu změn vah za všechny kroky učení k. Počáteční váhy jsou nulové. V praxi před učením sítě existuje tréninková množina dvojic $s : t$. Potom vstupní vektor $\vec{s}$ tvoří sloupcovou matici a výstupní vektor $\vec{t}$ řádkovou matici. Součinem těchto matic vznikne váhová matice W. V trénovací množině je P dvojic, což se musí projevit na váhové matici, která ukládá informace o vztazích mezi dvojicemi. Vztah pro výpočet váhové matice je zapsán vzorcem 2.14 [8].

$$W = \sum_{p=1}^{P} s^T(p)\, t(p) \tag{2.14}$$

Ve vzorci $s^T(p)$ je sloupcová matice vstupního vektoru pro dvojici z trénovací množiny na indexu p a $t(p)$ je sloupcová matice výstupního vektoru pro dvojici z trénovací množiny na indexu p.

Ostatní dělení sítí

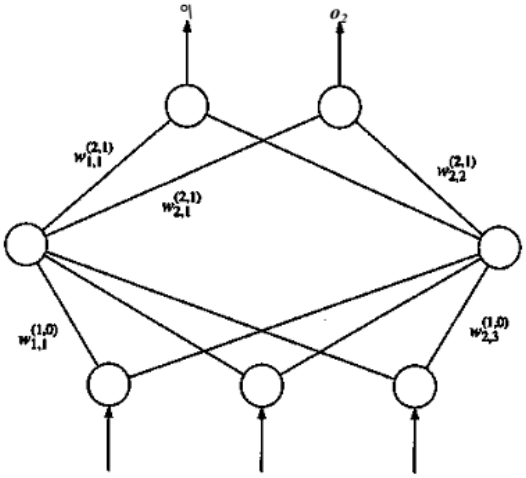
Umělé neuronové sítě se také dělí podle dalších kritérií, ale tyto kritéria nejsou podstatné při řešení této práce, ale v případě zájmu čtenáře jsou zde zmíněny spolu s odkazem na literaturu, ve které lze dohledat potřebné detaily. Sítě se tedy mohou dělit i podle *proměnlivosti topologie sítě*[13], podle *typu implementace*[13] nebo podle *způsobu výpočtu*[13], jak síť získá výstup.

2.3 Vybrané umělé neuronové sítě

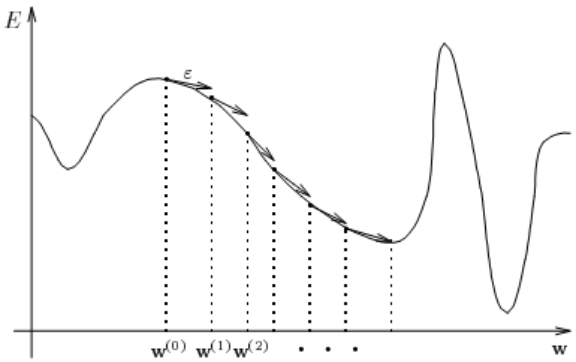
V této části jsou popsány vybrané umělé neuronové sítě. Jsou to *vícevrstvý perceptron*, *Hopfieldova síť* a *Kohonenova síť*. Ostatní sítě jsou zmíněny na konci části společně s odkazem na literaturu.

2.3.1 Vícevrstvý perceptron

Vícevrstvým perceptronem rozumíme z hlediska struktury dopřednou vrstvovou síť, která je učena algoritmem zpětného šíření chyby. Toto učení a algoritmus je popsán dále. Tento algoritmus ale obsahuje nevýhodu, že nemusí konvergovat, takže nedosáhne správných hodnot, ale pak po naučení výpočet probíhá v určitém počtu kroků. Síť je tedy složená z perceptronů, které mají lineární bázovou funkci a aktivační funkci (většinou sigmoidu). Může se využívat v úlohách zabývající se klasifikací, aproximací funkce nebo predikce [13].



Obrázek 2.4: Příklad vícevrstvého perceptronu[13]



Obrázek 2.5: Gradientní metoda[10]

Algoritmus zpětného šíření chyby

Při algoritmu zpětného šíření chyby[1] se nejdříve vypočítá výstupní vektor sítě, který vznikne přiložením vstupního vektoru k síti a jeho zpracováním. Poté se výstupní vektor porovná s požadovaným výstupním vektorem, s jehož pomocí se vypočítá chyba učení. Tato chyba se pak šíří neurony směrem k vstupní vrstvě, a zároveň se mění podle této chyby váhy jednotlivých neuronů. Učení se tedy vlastně skládá z průchodů sítí směrem k výstupní vrstvě, kdy je stav sítě určený a neměnný, a z průchodů směrem ke vstupní vrstvě, kdy se šíří chyba a upravují se váhy neuronů. Proto se algoritmus nazývá zpětným šířením chyby[12].

Samotná úprava vah je založená na principu gradientu, protože chybová funkce je funkcí vícerozměrnou. Je to kvůli mnohorozměrnému váhovému vektoru. Abychom získali co nejnižší chybu, je nutné najít konfiguraci váhového vektoru, kdy je chybová funkce v globálním minimu. V počáteční konfiguraci váhového vektoru $w^{(0)}$ tedy vytvoříme gradient a ve směru tohoto vektoru dolů se posuneme o konstantu rychlosti učení γ, čímž získáme nové váhy[10]. Princip je znázorněn na obrázku 2.5. Z tohoto principu vychází vztah 2.15[12], kde γ je koeficient učení násobené parciální derivací chybové funkce z celé tréninkové množiny podle váhových vektorů, $Delta^l w_{ji}$ je změna váhy mezi neurony j, i, $\frac{\partial E}{\partial w_i}$ je parciální derivací chyby

[1] angl. Backpropagation

podle váhy w_i na neuronu i.

$$\Delta^l w_{ji} = -\gamma \frac{\partial E}{\partial w_i} \tag{2.15}$$

Chyba E se vypočítá dle vzorce 2.16, kde $y^*(k)$ je požadovaný výstupní vektor, $y(k)$ je skutečný výstupní vektor v kroku učení k, $\vec{w}(k-1)$ je váhový vektor v kroku k a k je krok učení.

$$E(\vec{w}(k-1)) = \frac{1}{2}(y^*(k) - y(k))^2 \tag{2.16}$$

Z principu algoritmu vychází, že chybu jedné vrstvy ovlivňují vrstva předcházející. Proto existují dva vzorce pro výpočet vah. První vzorec 2.17[12] je pro neurony na poslední vrstvě, kde ve vzorci $Delta^L w_{ji}$ je změna váhy mezi neurony ji, kdy i je na poslední vrstvě L. y_j^* je požadovaný výstup neuronu j, ${}^L y_j$ je výstup neuronu j na poslední vrstvě. ${}^L x_i$ je vstup do neuronu i. γ je koeficient učení. L je počet vrstev, ve vzorci ale představuje poslední vrstvu.

$$\Delta^L w_{ji} = \gamma \left(y_j^* - {}^L y_j\right) y_j \left(1 - {}^L y_j\right)^L x_i, \tag{2.17}$$

Druhý vzorec je pro předcházející vrstvy zapsaný vzorcem 2.18[12], kde n je počet neuronů ve vrstvě, l je číslo aktuální vrstvy sítě. Ostatní symboly jsou popsané v předcházejícím vzorci 2.17.

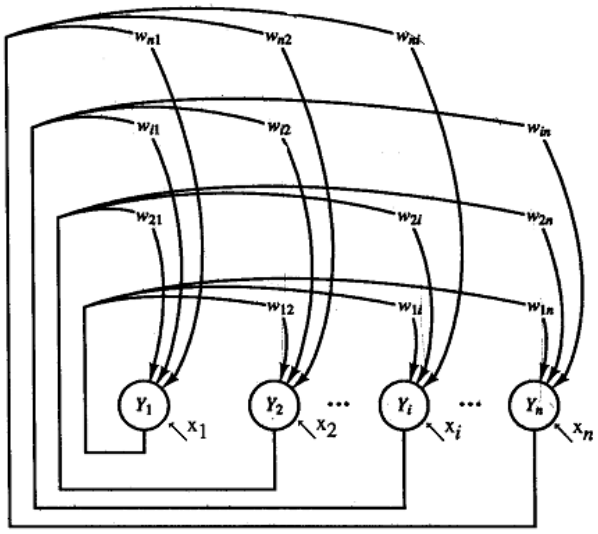
$$\Delta^L w_{ji} = \gamma \sum_{k=1}^{n_{l+1}} \left({}^{l+1}\delta_k^{l+1} w_{kj}\right)^l y_j \left(1 - {}^l y_j\right)^l x_i \tag{2.18}$$

, kde ve vzorci 2.18 se aplikuje myšlenka, že chybu aktuálního neuronu j ovlivňují neurony z následující vrstvy. Proto se ve vzorci nachází suma nad všemi neurony (n je počet neuronů ve vrstvě $l+1$) součinů částečných chyb a vah mezi neuronem j a k z vrstvy $l+1$. Nakonec je nutné zmínit hlavní nevýhodu algoritmu zpětného šíření chyby, kterou je zastavení algoritmu na lokální minimu, ale má dosáhnout globálního minima. Tuto situaci lze řešit přidáním dalších skrytých neuronů[12]. Konec učení nastává v okamžiku, kdy celková chyba dosáhne přijatelné hodnoty vůči požadovanému výstupnímu vektoru nebo dosáhne požadovaného počtu iterací.

2.3.2 Hopfieldova síť

Hopfieldovou sítí (obrázek 2.6) nazýváme síť, která je plně propojená a symetrická, ale vazby neuronů samy na sebe jsou nulové. Patří mezi iterační auto-asociativní paměti, což znamená, že dokáže rekonstruovat obraz se šumem na obraz bez šumu a pracuje v iteracích. Síť se využívá k asociačním úlohám. Jejich učení je postaveno na *Hebbově principu*, neboli v síti se vytvoří dvojice neuronů, které když jsou oba aktivní, zvyšuje se váha mezi nimi. Toto učení je s učitelem, ale v případě auto-asociativních pamětí jsou vstupní a výstupní vektor identické. V Hopfieldově síti jsou neurony inicializovány buď binárně $\{0, 1\}$ nebo bipolárně $\{-1, 1\}$[8].

$$W = \{w_{ij}\} \qquad w_{ij} = \sum_{p=1}^{P} s_i(p)\, s_j(p) \quad i \neq j, \tag{2.19}$$



Obrázek 2.6: Diskrétní Hopfieldova síť[11]

Ve vzorci W je váhová matice, w_{ij} je váha mezi neurony i, j, P je počet trénovacích dvojic, p index v trénovací množině, $s_i\,(p)$ je vstupní vektor do neuronu i z trénovací množiny na indexu p a $s_j\,(p)$ je výstupní vektor z neuronu j z trénovací množiny na indexu p.

Pro popis funkce této sítě při asociaci je zavedena *energetická funkce*, která každému stavu sítě přiřazuje potencionální energii. Tato funkce je nerostoucí pro daný *stav systému* (množina aktivací všech neuronů), zdola ohraničená a pokud je nalezena, vždy konverguje ke stabilní množině aktivací neuronu.

Z této energetické funkce lze vyvodit nevýhodu Hopfieldovy sítě. Oproti vícevrstvému perceptronu fáze učení má předem daný počet kroků, ale samotná minimalizace energetické funkce k dosažení stability probíhá v iteracích a konvergovat nemusí. Z toho vyplývá, že podobně jak u algoritmu zpětného šíření chyby se cíl učení nachází v lokálním minimu funkce, v tomto případě energetické funkce. Takže okolo lokálního minima se nachází tzv. *oblast atrakce*, ve které všechny vstupní vektory jsou převedeny na výstupní vektor v lokálním minimu. Kvůli těmto oblastem při učení pak vznikají *nepravé vzory tzv. fantómy*, kterým neodpovídá žádný tréninkový vzor[8].

Další nevýhodou této sítě je její malá paměť, kdy Hopfield experimentálně zjistil, že počet binárních vzorů, které si síť dokáže zapamatovat a rekonstruovat s žádanou přesností, je možné zapsat vzorcem 2.20 [8], kde P je počet vzorů a n je počet neuronů v síti. Tento vzorec platí pro síť s bipolárními vzory.

$$P \approx \frac{n}{2\log_2 n}, \tag{2.20}$$

2.3.3 Kohonenova síť

Před vysvětlením Kohonenovy sítě je nutné znát *síť Maxnet* a *jednoduchou soutěživou síť*, ze kterých vychází Kohonenova síť. Síť Maxnet je plně propojená symetrická síť, kde váhy mezi neurony jsou vždy stejné a záporné, kdežto zpětnovazebné váhy jsou ohodnoceny hodnotou +1. Výstupy neuronů z sítě Maxnet se počítají synchronně dle vztahu 2.21[13].

$$y_j(k) = max(0, u_j(k)) = max\left(0, \sum_{i=1}^{m} w_{ji} y_i(k-1)\right), \tag{2.21}$$

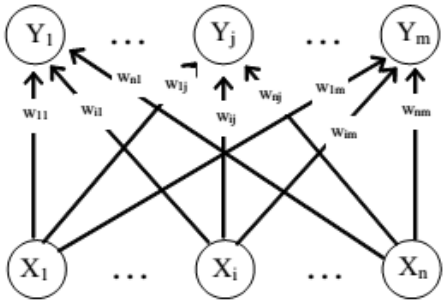
Ve vzorci $u_j(k)$ je vnitřní potenciál neuronu j aktuálního kroku výpočtu, w označuje váhu mezi dvěma neurony i, j. Symbol $y_i(k-1)$ představuje výstup neuronu i z předcházejícího kroku výpočtu. Krok výpočtu je označeno k. Z principu sítě Maxnet a vzorce 2.21 vyplývá, že tento vzorec se v každém kroku vypočítá zvlášť pro každý neuron. Součet ve vzorci znázorňuje sčítání hodnot na spojeních vycházejících z neuronu. Součin, který představuje hodnotu na zpětné vazbě, bude kladný a ostatní součiny budou záporné, protože reprezentují hodnoty ze spojení s ostatními neurony. Výsledek se porovná s nulou a neurony, kde výsledek sumy byl menší než nula, se vyřadí z výpočtu. Výpočet končí, až má nenulovou hodnotu pouze jediný vítězný neuron[13].

Jednoduchá soutěživá síť vychází ze sítě Maxnet, kdy k síti připojí vstupy o váhách w_{ij} a algoritmus učení je založen na základním soutěživém učení a pracuje jako síť Maxnet podle vzorce 2.21. Tím se získá vítězný neuron a ten upraví váhy.

Kohonenova síť, znázorněná obrázkem 2.7, je jednoduchou soutěživou sítí, ale s upraveným algoritmem učení, kdy vítězný neuron podle vzorce 2.22 upraví váhy i sousedům ve vzdálenosti d. Pro změnu vah vítězného neuronu a jeho sousedů platí vztah 2.22[11].

$$\vec{w}_{j^*d}(k) = \vec{w}_{j^*d}(k-1) + \gamma(k)\left(\vec{i}_p - \vec{w}_{j^*d}(k-1)\right), \tag{2.22}$$

Ve vzorci $\vec{w}_{j^*d}$ je váhový vektor, který obsahuje váhu vítězného neuronu j^* a váhy jeho sousedů. Symbol d představuje vzdálenost, do které bude sousedy ovlivňovat. Symbol k je krokem učení. γ je koeficient učení. Učení probíhá, dokud váhy nedosáhnou ustáleného stavu.



Obrázek 2.7: Kohonenova mapa[8]

Počet neuronů v síti při shlukování odpovídá počtu atributů objektu. Tato síť se používá především pro úlohy shlukování, na které se také v práci budou knihovny s Kohonenovou sítí porovnávat.

2.3.4 Další sítě

Existují další sítě, které lze dohledat v literatuře. Jsou to Adaline síť, Madaline síť, sítě s proměnlivou topologií[13], sítě s kaskádovou architekturou[10], sítě s časovým zpožděním[12], sítě s radiální bázovou funkcí[12], RCE síť[13], síť Counterpropagation[8], síť LVQ, síť ART, hetero-asociativní paměti BAM a SDM nebo Boltzmannův stroj[13].

Kapitola 3

Návrh porovnávání a experimentů

Cílem této kapitoly je navržení kritérií, podle kterých se budou dále porovnávat vybrané knihovny umělých neuronových sítí, a experimentů, které se budou provádět na programech vytvořených za použití knihoven neuronových sítí. Tyto kritéria a experimenty musejí být zvoleny vhodně tak, aby podle jejich výsledků bylo možné posoudit, kterou knihovnu je v jistých případech vhodné použít.

3.1 Kritéria porovnávání

Jako první z kritérií jsem vybral *výpočetní čas*, protože rychlost provádění výpočtu je důležitá vlastnost programu. Což znamená, že čím rychleji se program provede, tím v tomto ohledu kvalitnější implementaci sítě knihovna poskytuje.

Dalším je *počet řádků zdrojového kódu*, který nám udává, kolik řádků ve zdrojovém kódu zabere implementace dané úlohy a sítě použité v úloze. Každý program napsaný s danou knihovnou se může lišit v počtu řádků od programu s druhou knihovnou.

Následuje *kvalita dokumentace a API*, která je subjektivním hodnocením, protože tyto kritéria ovlivňují to, jak snadno se učí programátor danou knihovnu používat a jak intuitivně se mu s ní pracuje. Pro tyto dvě kritéria jsem použil hodnocení o třech stupních. Prvním a nejlepším stupněm je hodnocení *vynikající*. U dokumentace je tímto stupněm ohodnocena kvalitní dokumentace, která obsahuje i příklady použití, a API, kde jsou popsány jak funkce, tak jejich parametry. Prostředním stupněm je hodnocení *dobré*. Knihovny, které mají alespoň nějakou dokumentaci nebo popis API, jsou hodnoceny tímto stupněm. Nejhorším stupněm je hodnocení *nedostatečné*, kdy u knihovny dokumentace a popis API chybí, je nesrozumitelné nebo obsahuje chyby.

Důležitým hlediskem k porovnání je *jazyk*, podle kterého si programátor vybírá knihovnu v jazyce, který ovládá nebo který potřebuje ke své vlastní aplikaci, kterou píše v určitém jazyku. Některé knihovny nabízejí možnost vazeb na další programovací jazyky než jen na jazyk, ve kterém je knihovna implementována.

Při instalaci knihovny je důležitým kritériem *platforma*, na které programátor pracuje nebo pro kterou aplikaci vyvíjí. Pokud je knihovna podporována na více platformách, tím širší je její možnost využití.

Další kritérium souvisí s udržováním knihovny, což je *poslední aktualizace knihovny*, která vypovídá o tom, zda se knihovna udržuje a vyvíjí. Zaznamenává se měsíc a rok poslední aktualizace, podle kterých se následně knihovna porovnává.

S udržováním knihovny také souvisí kritérium *komunita*. Toto kritérium se zabývá tím,

zda existuje okolo té knihovny komunita, která by uživateli v případě problému poradila nebo udržovala knihovnu. Za nejlepší možnost v tomto kritériu považuji diskuzní fórum. Základem je kontaktní email na tvůrce knihovny.

Důležitým právním kritériem je *licence*, která programátorovi určuje, jak může s kódem manipulovat. S knihovnami s volnější licencí lze používat bez větších problémů ohledně autorských práv.

Následně se budou porovnávat *podporované umělé neuronové sítě*. V tomto hledisku jde o zjištění možností vybrané knihovny, přičemž čím více sítí knihovna implementuje, tím univerzálnější může být.

Posledním hlediskem je *podpora vícevláknového zpracování*. Toto kritérium sleduje, jestli knihovna podporuje využití více vláken při výpočtu za účelem zrychlení. Ty knihovny, které podporují vícevláknové zpracování, jsou v tomto kritériu hodnoceny jako lepší.

Většina výše zmíněných kritériích jsou vlastnosti, které je možno porovnat bez experimentování. Ale pro výpočetní čas a délku zdrojového kódu jsem navrhl tyto následující experimenty s užitím umělé neuronové sítě pro aproximaci funkce, shlukování a asociaci.

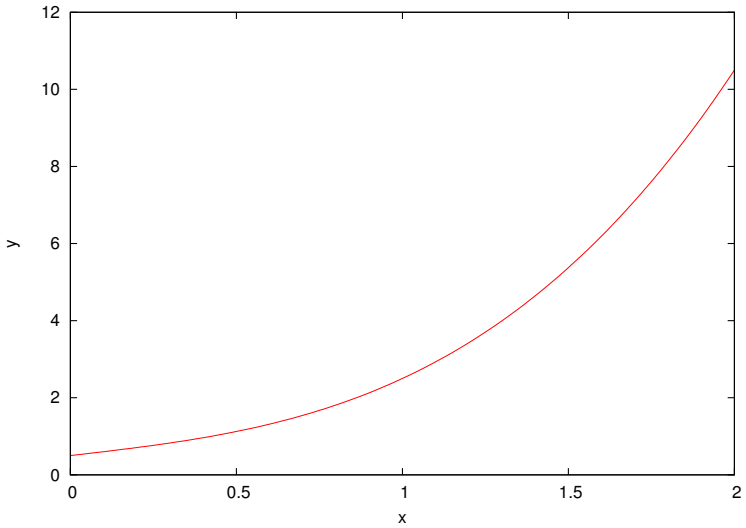
3.2 Aproximace funkce

V úloze jsem zvolil tři funkce, které v experimentu neuronová síť bude aproximovat. Jsou to funkce, které jsou matematicky zapsané ve vzorcích 3.1, 3.2 a 3.3. Tyto funkce jsou znázorněny grafy 3.1 a 3.2. Funkce a hodnoty intervalů, na kterých bude síť funkce aproximovat, jsou z článku [5].

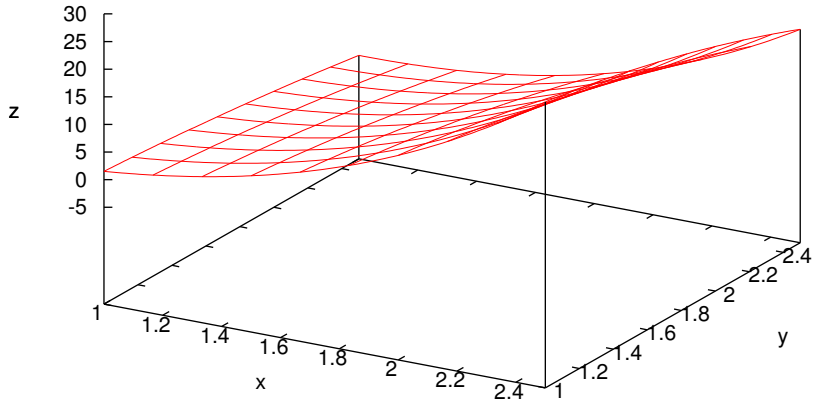
$$f(x) = x^3 + x + 0.5 \quad x \in R, 0 \leq x \leq 2 \tag{3.1}$$

$$f(x, y) = 2x^3 - xy^2 + 0.5 \quad x, y \in R, 1 \leq x \leq 2.5, \quad 1 \leq y \leq 2.5 \tag{3.2}$$

$$f(x, y, z) = x^2 + y^2 + z^2 \quad x, y, z \in R, 1 \leq x, z \leq 1.5, \quad 2 \leq y \leq 2.5 \tag{3.3}$$



Obrázek 3.1: Funkce f(x)



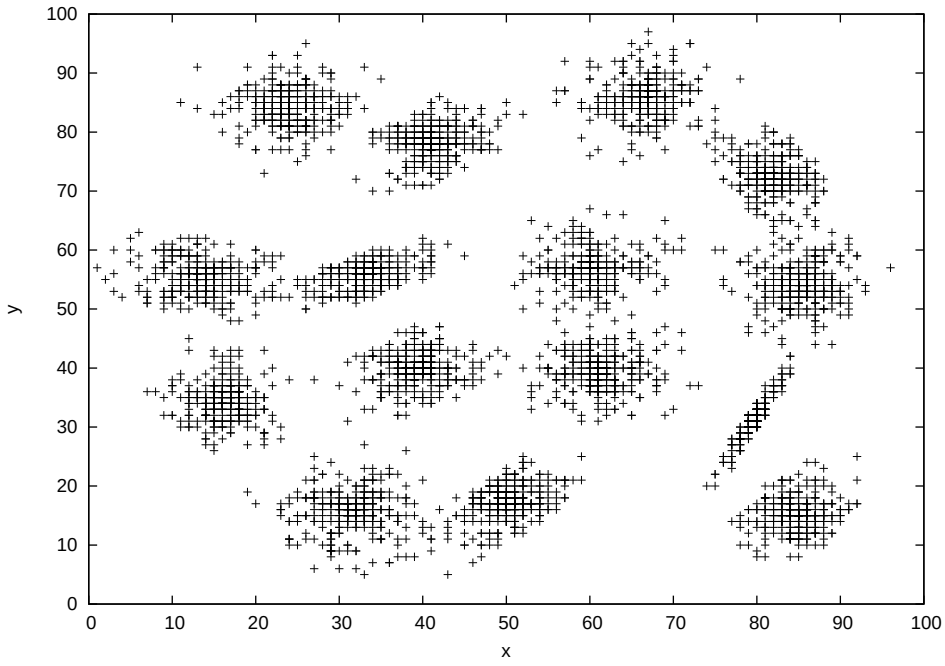
Obrázek 3.2: Funkce f(x,y)

V úloze se použije vícevrstvý perceptron, který bude obsahovat jednu skrytou vrstvu s deseti neurony. Bázová funkce těchto neuronů je lineární a za aktivační funkci byla zvolena funkce sigmoidní. Koeficient učení je nastaven na 0.2, počet cyklů učení 100000 a požadovaná přesnost je 0.0001. Neuronová síť bude aproximovat funkce zadanou pomocí vstupních hodnot z trénovací množiny. Tyto hodnoty jsou získány vzorkováním příslušného intervalu funkce, kde bude síť aproximovat, a jejich následném dělení konstantou 100 za účelem lepší aproximace. V experimentu se budou knihovny porovnávat na základě času potřebnému k učení. Vzorkování je nastaveno na hodnotu 150 pro $f(x)$, 25 pro $f(x, y)$ a 10 pro $f(x, y, z)$. Z tohoto vzorkování vzniknou množiny vstupních dat o velikosti $T(f(x)) = 150$, $T(f(x, y)) = 625$ a $T(f(x, y, z)) = 1100$.

3.3 Shlukování

V úloze bude síť shlukovat dvoudimenzionální vektory, kde jeden vektor modeluje souřadnice bodu v rovině. Vektor představuje pozici bodu v rovině o rozloze 100×100. Příklad vstupních dat je znázorněn na obrázku 3.3. Tyto data jsou převzata z [2] a [3], která jsou dělena hodnotou 10000.

Pro experiment shlukování jsem vybral Kohonenovu síť. Experiment se provádí pro různé vstupní data, přičemž má síť dva vstupní neurony a šestnáct výstupních neuronů, které mají radiální bázovou funkci a Gaussovu aktivační funkci. Koeficient učení je nastaven na hodnotu 0.3, počet cyklů učení je roven 1000 a počáteční velikost okolí je 1. V experimentu se provede porovnání podle výstupních grafů, která knihovna dokáže rozeznat shluky dat s větší přesností, a času potřebnému k učení.

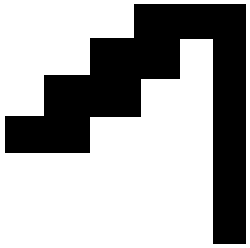
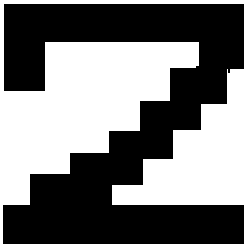
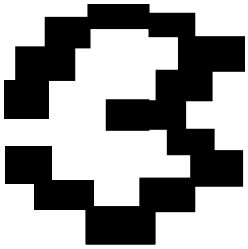
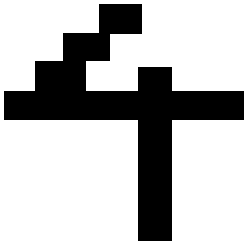
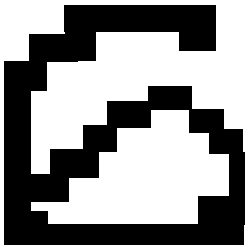
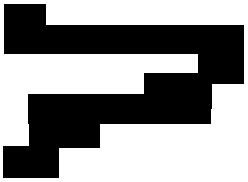


Obrázek 3.3: Úloha shlukování

3.4 Asociace

V úloze obsahuje trénovací množina sedm původních obrázků o velikosti 10×10 pixelů. Na každém obrázku je jedno číslo od jedné do sedmi. Tyto obrázky jsou reprezentovány maticemi. Pomocí těchto matic proběhne učení. Poté na vstup sítě přicházejí matice modelující obrázky se šumem, které síť zrekonstruuje. Přibližné nákresy vstupních obrázků jsou v tabulce 3.1.

Experiment k asociaci využívá Hopfieldovu síť se 100 neurony, které mají aktivační funkci signum, která je podobná skokové funkci s jediným rozdílem, že místo 0/1 vrací hodnoty $-1/1$. Podle vzorce 2.20 síť se dokáže naučit sedm obrazů, ale tyto obrazy musejí mít dostatečně velkou Hammingovu vzdálenost, která udává, v kolika bitech se obrazy liší. Podle tohoto zjištění jsem navrhl vstupní vzory. V experimentování se používá různě poškozených vstupních obrázků, které si na základě procentuální hodnoty šumu sám program vyrobí. Porovnávat se budou počty chyb v závislosti na šumu a času potřebném k učení sítě.

Vzor 1	Vzor 2	Vzor 3
		
Vzor 4	Vzor 5	Vzor 6
		
Vzor 7		
		

Tabulka 3.1: Vzory pro autoasociaci

Kapitola 4

Vybrané knihovny

V kapitole se práce zabývá knihovnami, které jsem vybral k porovnávání a experimentům. Jsou to knihovny založené na jazycích C/C++, které z podstaty svého implementačního jazyka pracují rychleji než ostatní knihovny. Mezi tyto vybrané knihovny patří *FANN: Fast Artificial Neural Network Library*[1], *OpenNN: Open Neural Networks Library*[2], *OpenANN: Open Artificial Neural Networks Library*[3], *Neu: The Neu Framework*[4], *KNNL: Kohonen Neural Network Library*[5], *Libann: Artificial Neural Network Library*[6] a *Annie: Artificial Neural Network Library*[7]. Z knihoven, které jsou napsány v jiných jazycích, je možné zmínit knihovnu *Torch7* založenou na jazyku Lua, knihovnu *Encog* ke strojovému učení napsanou v jazyku Java, ze skriptovacího jazyka Python knihovny *PyBrain* a *PyLearn*, .NET knihovnu *AForge.NET* nebo knihovnu *ANN* pro jazyk PHP5.

Vybrané knihovny jsou následně uspořádány do tabulek 4.1, 4.2, 4.3, 4.4, 4.5, 4.6 a 4.7. Jednotlivé položky tabulky jsou kritéria, které byly navrženy v předchozí kapitole a podle kterých se knihovny budou následně porovnávat.

Instalace knihoven je ve většině knihoven možná podle návodu dostupném na webových stránkách nebo v souborech `README`. Některé knihovny ale vyžadují instalaci přídavných programů a knihoven. Tyto dodatečné balíčky jsou většinou zmíněny v manuálech ke knihovnám. V tomto ohledu nastal problém u knihovny *Neu*, která kromě programů zmíněných v příručce vyžadovala balíček *Editline*.

U zastaralých knihoven *Libann* a *Annie* byly problémy se zdrojovým kódem, který jsem částečně upravoval, aby bylo možné použít alespoň jednotlivé moduly knihovny. Jednalo se o moduly realizující *Hopfieldovu* a *Kohonenovu* síť, které jiná knihovna explicitně neimplementuje jako strukturu. Obě knihovny obsahují i vícevrstvý perceptron, ale pro dostatečný počet novějších knihoven a kvůli problémům s instalací se pro tyto knihovny vícevrstvý perceptron porovnávat nebude.

[1] http://leenissen.dk/fann/wp/download/
[2] http://www.intelnics.com/opennn/download
[3] https://github.com/OpenANN/OpenANN
[4] http://neu.andrometa.net/
[5] http://sourceforge.net/projects/knnl/
[6] http://www.nongnu.org/libann/index.html
[7] http://sourceforge.net/projects/annie/

4.1 FANN

FANN	
Jazyky	C, C++, Python
Platforma	Windows, Linux, Unix, OS X
Licence	GNU LGPL
Neuronové sítě	vícevrstvý perceptron
Komunita	fórum
Více vláken	ne
Dokumentace	dobrá
API	vynikající
Aktualizace	leden 2012

Tabulka 4.1: FANN

Knihovna *FANN* je knihovnou zaměřenou na neuronové sítě, speciálně na vícevrstvý perceptron. Podporuje kromě klasického algoritmu zpětného šíření chyby i jeho upravené verze např. *RPROP*[6], *Quickprop*[13]. Z aktivačních funkcí nabízí lineární, skokovou, sigmoidu, Gaussovu a jiné. Pro tuto knihovnu existuje několik grafických rozhraní jako FANNTool nebo NeuralView.

4.2 OpenNN

OpenNN	
Jazyky	C++
Platforma	Windows, Linux
Licence	GNU LGPL
Neuronové sítě	vícevrstvý perceptron
Komunita	email
Více vláken	ne
Dokumentace	dobrá
API	vynikající
Aktualizace	únor 2015

Tabulka 4.2: OpenNN

Knihovna *OpenNN* od skupiny Intelnics je zaměřená na vícevrstvý perceptron. Kromě algoritmu zpětného šíření chyby podporuje další metody učení, např. *ConjugateGradient*[13]. Knihovna implementuje skokovou aktivační funkci, sigmoidu, hyperbolický tangens a lineární aktivační funkci. Instalace se doporučuje využitím frameworku *Qt*.

4.3 Neu

Neu	
Jazyky	C++
Platforma	Linux, Mac OS
Licence	BSD
Neuronové sítě	vícevrstvý perceptron
Komunita	email
Více vláken	ne
Dokumentace	nedostačující
API	dobrá
Aktualizace	únor 2015

Tabulka 4.3: The Neu Framework

Framework *Neu* je založen na standardu `C++11`. Jeho účelem je vytváření aplikací umělé inteligence, modelování a simulace, databáze nebo vytváření překladačů. Implementuje aktivační funkce sigmoidy, lineární funkce a funkce signum. Pro instalaci je třeba mít nainstalované balíčky editline, GMP, MPFR, Boost, Flex, Bison, LLVM a Clang.

4.4 OpenANN

OpenANN	
Jazyky	C++, Python
Platforma	Linux, Mac OS
Licence	GNU GPL 3
Neuronové sítě	vícevrstvý perceptron
Komunita	email
Více vláken	ne
Dokumentace	vynikající
API	dobrá
Aktualizace	únor 2015

Tabulka 4.4: OpenANN

K instalaci knihovny *OpenANN* je nutné nainstalovat knihovnu *Eigen*, další balíčky je třeba instalovat podle nutnosti. Pro vygenerování dokumentace je vyžadován *doxygen* a *Graphviz* nebo pro vazby na jazyk `Python` balíčky *Numpy* a *Cython*. Další případně požadované balíčky jsou zmíněny v dokumentaci knihovny.

4.5 KNNL

KNNL	
Jazyky	C++
Platforma	Windows, Linux
Licence	BSD
Neuronové sítě	Kohonenova síť
Komunita	email
Více vláken	ne
Dokumentace	dobrá
API	dobrá
Aktualizace	duben 2013

Tabulka 4.5: KNNL

Knihovna pro Kohonenovu síť ke své funkci potřebuje knihovnu *Boost*. Implementuje jak algoritmus učení jednoduché soutěživé sítě, tak Kohonenovy sítě s různými topologiemi(liší se tím, kolik neuronů vedle vítězného neuronu bude ovlivněno). Není třeba instalovat, stačí připojit hlavičkové soubory.

4.6 Libann

Libann	
Jazyky	C++
Platforma	Linux
Licence	GNU GPL
Neuronové sítě	Kohonenova síť, Hopfieldova síť
Komunita	email
Více vláken	ne
Dokumentace	nedostačující
API	dobrá
Aktualizace	květen 2003

Tabulka 4.6: Libann

Starší knihovna *Libann* implementuje kromě testovaných dvou sítí také vícevrstvý perceptron a Boltzmannův stroj. Pro aktivační funkce obsahuje skokovou funkci a funkci signum, proto pro Kohonenovů síť bylo třeba napsat vlastní Gaussovu aktivační funkci a upravit koncovou podmínku učení na počet iterací.

4.7 Annie

Annie	
Jazyky	C++
Platforma	Linux
Licence	GNU LGPL 2
Neuronové sítě	Hopfieldova síť
Komunita	email
Více vláken	ne
Dokumentace	dobrá
API	dobrá
Aktualizace	červen 2004

Tabulka 4.7: Annie

Knihovna *Annie* je zastaralejší knihovnou implementující Hopfieldovu síť, Kohonenovu síť a vícevrstvý perceptron. Kohonenova síť je bohužel nefunkční a pro vícevrstvý perceptron existují novější knihovny. Z aktivačních funkcí podporuje Gaussovu aktivační funkci, sigmoidu nebo funkci signum a další.

Kapitola 5

Porovnání knihoven

V této závěrečné kapitole provedu porovnání knihoven podle parametrů navržených v předchozích kapitolách. Nejdříve se tabulkově porovnají vlastnosti jednotlivých knihoven a potom následují výsledky experimentů vyjádřené pomocí grafů. Závěrem bude rozhodnutí, která knihovna je pro danou síť lepší a zdůvodnění tohoto výsledku.

5.1 Knihovny s Hopfieldovou sítí

Pro porovnávání byly vybrány knihovny *Libann* a *Annie*, které budou řešit úlohu autoasociace pomocí Hopfieldovy sítě. Porovnání podle nominálních parametrů jsou znázorněná tabulkou 5.1.

	Libann	**Annie**
Jazyk	C++	C++
OS	Linux	Linux
Licence	GNU GPL	GNU LGPL 2
Komunita	email	email
Vícevláknový	ne	ne
Dokumentace	nedostačující	dobrá
API	dobrá	dobrá
Aktualizace	květen 2003	červen 2004

Tabulka 5.1: Porovnání knihoven Libann a Annie

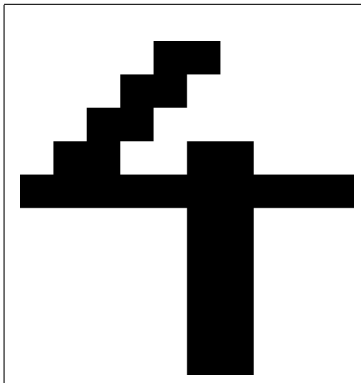
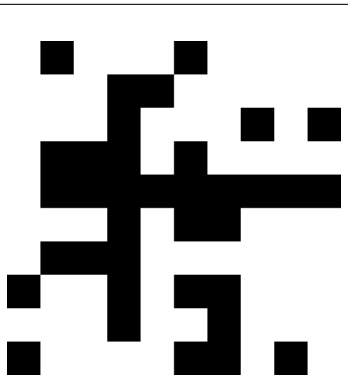
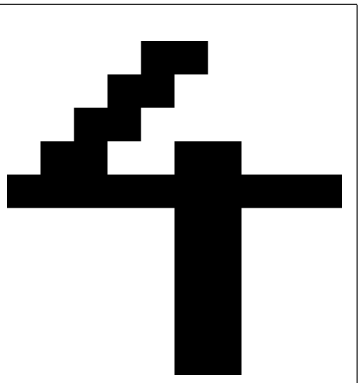
Knihovny se liší v licenci, která u knihovny *Libann* přikazuje volné šíření aplikací, které knihovnu využívají, kdežto *Annie* s licencí `GNU LGPL 2` může být využívána programem s licencí proprietární. Ohledně kvality dokumentací a API *Libann* sice má webovou stránku s příklady, ale jsou zmatečné a příliš stručné, takže uživatel musí informace hledat ve zdrojových kódech nebo si vygenerovat manuální *Doxygen* dokumentaci ve složce s knihovnou. V *Annie* webové stránky přímo obsahují popis API za pomocí *Doxygenu*, což považuji za dobrou dokumentaci a lépe se mi s touto knihovnou pracovalo. Proto dokumentaci *Libannu* hodnotím jako nedostatečnou a *Annie* jako dobrou. Vzhledem ke kvalitě API je jeho popis a intuitivnost v obou knihovnách obdobná. Obě knihovny jsou zastaralé, poslední aktualizace byly v roce 2003, respektive 2004. Nelze je podle instalačních pokynů nainstalovat a obsahují chybné nebo zastaralé konstrukce, takže pro experiment musely být vyjmuty a opraveny konkrétní moduly. Novější knihovnou je *Annie*.

Výkonnostní parametry knihoven jsem porovnal pomocí experimentu, kdy jsem měřil čas učení sítě. Výsledky po dvaceti spuštěních programu jsou znázorněny tabulkou 5.2. Dle experimentu je v tomto aspektu lepší knihovna *Libann*, protože se naučí rychleji.

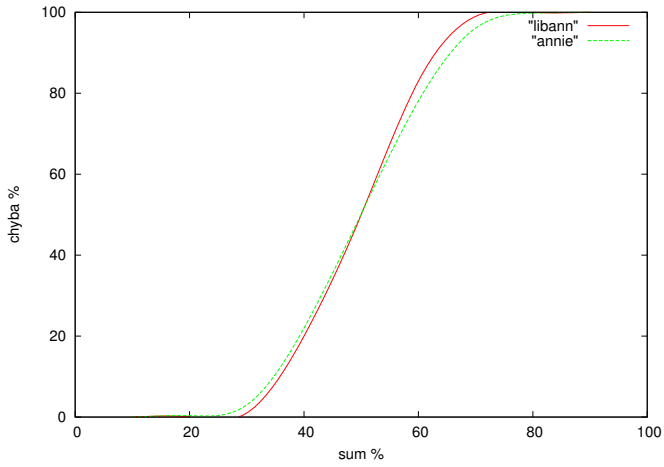
	Libann	**Annie**
Čas učení[s]	0.000619	0.0010649

Tabulka 5.2: Výsledky testů výkonnosti

V druhém experimentu jsem sledoval vliv šumu na procentuální chybovost rekonstrukce obrazu za účelem pozorování, do jaké míry může být poškozen vstupní obraz, aby ho program bez chyby rekonstruoval. Na obrázcích 5.3 je znázorněna práce programu a na grafu 5.1 je zmíněná závislost chyby na šumu.

Původní	Poškozený	Rekonstruovaný
		

Tabulka 5.3: Příklad práce programu



Obrázek 5.1: Závislost chyby na šumu

Na základě grafu lze odvodit, že oba programy dávají téměř stejné výsledky, i když by měly dávat výsledky stejné. Tato skutečnost může být daná odchylkou měření.

5.2 Knihovny s Kohonenovou sítí

Kohonenovu síť implementují knihovny *KNNL* a *Libann*. Pro tuto síť jsem k testování vybral problém shlukování. Porovnání podle nominálních parametrů jsou znázorněná tabulkou 5.4.

	KNNL	**Libann**
Jazyk	C++	C++
OS	Windows, Linux	Linux
Licence	BSD	GNU GPL
Komunita	email	email
Vícevláknový	ne	ne
Dokumentace	dobrá	nedostačující
API	dobrá	dobrá
Aktualizace	duben 2013	červen 2004

Tabulka 5.4: Porovnání knihoven KNNL a Libann

Podle tabulky s porovnáním (5.4) knihovna *KNNL* je přenositelná i na operační systém Windows, proto je v tomto hledisku lepší. Licence knihovny *Libann* je omezenější jako licence knihovny *KNNL*, tedy knihovnu *KNNL* v tomto hledisku považuji za lepší. Knihovna *KNNL* je novější, využívá knihovny *Boost* a obsahuje více topologií, jak může vítězný neuron ovlivňovat své okolí.

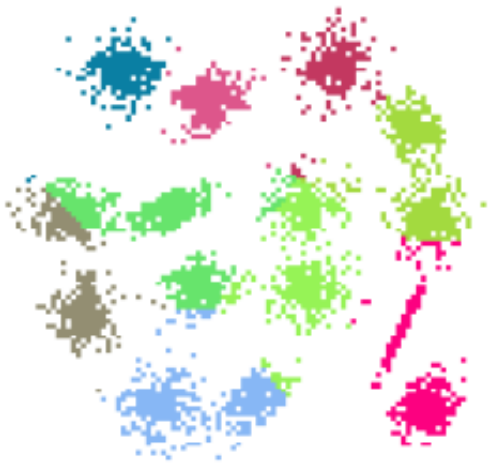
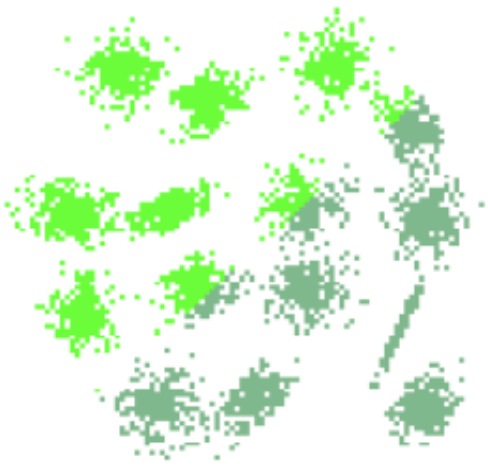
V experimentu za účelem porovnání výkonnosti program s knihovnou *Libann* dokázal naučit síť za čas kratší než program s knihovnou *KNNL*. Výsledné průměrné časy učení jsou v tabulce 5.5.

	Libann	**KNNL**
Čas učení[s]	4.30	138.5

Tabulka 5.5: Výsledky testů výkonnosti

Na obrázku 5.6 je výstup programu s knihovnou*KNNL*. Na těchto konkrétních datech program nalezl devět shluků. Výsledek shlukování je uspokojivý.

Výstup programu s knihovnou *Libann* je znázorněn obrázkem 5.6. Tento výstup není uspokojivý jako výsledek shlukování, i když nastavení parametrů, aktivačních funkcí a počtu iterací jsou stejné jako pro program s knihovnou *KNNL*. Příčinu této chyby se nepodařilo zjistit, pravděpodobně se ale jedná o chybu v samotné knihovně.

KNNL	Libann
	

Tabulka 5.6: Výstupy experimentu se shlukováním

5.3 Knihovny s vícevrstvým perceptronem

Knihovny *FANN*, *OpenANN*, *OpenNN* a *Neu* implementují vícevrstvý perceptron, pro který jsem zvolil úlohu aproximace tří odlišných funkcí. Porovnání podle nominálních parametrů jsou znázorněná tabulkou 5.7.

	FANN	**Neu**	**OpenNN**	**OpenANN**
Jazyk	C, C++ aj.	C++	C++	C++, Python
OS	Win, Linux aj.	Linux, Mac	Win, Linux	Linux, Mac
Licence	GNU LGPL	BSD	GNU LGPL	GNU GPL 3
Komunita	fórum	email	email	email
Vícevláknový	ne	ne	ne	ano
Dokumentace	dobrá	nedostačující	dobrá	vynikající
API	vynikající	dobrá	vynikající	dobrá
Aktualizace	leden 2012	leden 2015	únor 2015	únor 2015

Tabulka 5.7: Porovnání knihoven FANN, Neu, OpenNN a OpenANN

Z tabulky 5.7 vyplývá, že v rámci podporovaných programovacích jazyků je vhodná knihovna *FANN*, která také je přenositelná na více platforem než ostatní knihovny. *OpenANN* je vytvořena pod licencí `GPL`, která je omezenější jako licence `LGPL`. V případě problémů s knihovnou je v případě *FANN* možné se dotázat na fóru, u knihoven *OpenNN*, *Neu* a *OpenANN* lze se dotázat autora pomocí emailové adresy. *OpenANN* jako jediná z testovaných knihoven podporuje vícevláknové zpracování. Z hlediska kvality dokumentace je kvalitně zpracována knihovna *OpenANN*, která se kromě popsiu knihovny a jejích funkcí zabývá i související teorií. Ostatní knihovny mají dokumentaci průměrnou, případně obsahuje několik chyb. V případě knihovny *Neu* je dokumentaci modulu neuronových sítí věnována pouze jedna strana manuálu, většinu informací o použití neuronových sítí je nutno

nastudovat z přiložených příkladů. Kvalita API je výborná u knihoven *FANN* a *OpenNN*, které implementují řadu pomocných funkcí, které nejsou přístupné v ostatních knihovnách. Knihovny (s výjimkou *FANN*) jsou aktuální a udržované.

Výsledky měření výkonnosti jsou zpracovány v tabulce 5.8. Hodnoty v tabulce jsou průměrné hodnoty z dvaceti běhů programu. Z výsledků je zřejmé, že nejrychleji se naučí program s knihovnou *FANN*, jenže jeho výsledky jsou velmi nepřesné. Tato nepřesnost je pravděpodobně způsobena implementací knihovny.

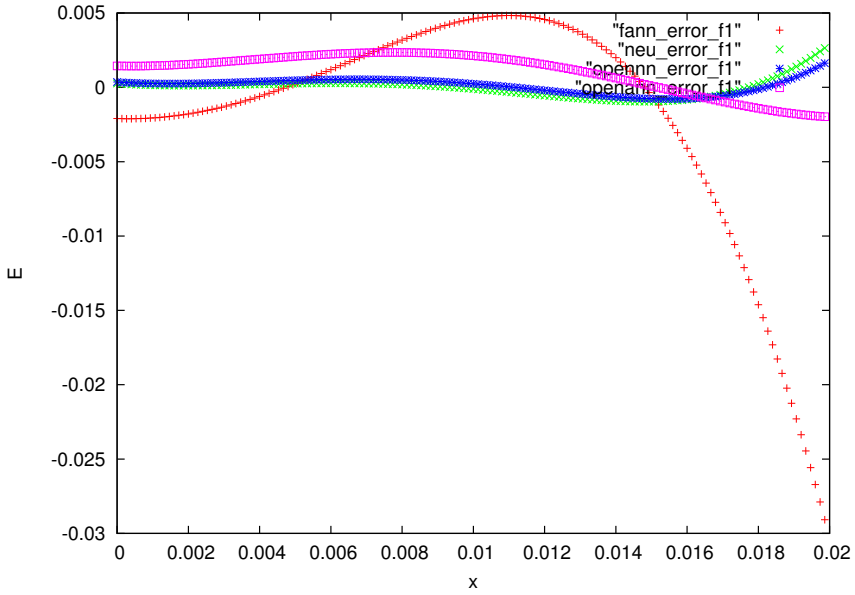
	Čas učení[s] f(x)	Čas učení[s] f(x,y)	Čas učení[s] f(x,y,z)
FANN	1.612	0.753	0.062
Neu	32.744	147.549	280.22
OpenNN	35.177	126.187	250.12
OpenANN	43.712	192.842	312.888

Tabulka 5.8: Výsledky testů výkonnosti

V grafu 5.2 je znázorněna chyba při aproximaci pomocí zvolených knihoven. Jelikož nelze vykreslit graf funkce o třech proměnných, vyjadřuji přesnost jako průměrnou chybu na jeden vzorek vstupních dat podle tabulky 5.9.

	FANN	Neu	OpenNN	OpenANN
Chyba/bod	0.02	0.00035	0.00044	0.00035

Tabulka 5.9: Průměrná chyba ve funkci f(x,y,z)



Obrázek 5.2: Funkce chyby aproximace funkce f(x)

Kapitola 6

Závěr

Cílem práce bylo provést rešerši ohledně existujících knihoven pro práci s umělými neuronovými sítěmi, tyto knihovny porovnat a provést s nimi experimenty. Za tímto účelem bylo nezbytné se seznámit se základní teorií ohledně neuronu, základní jednotky sítí. Práce se v této části zabývá biologickým neuronem, ze kterého umělý neuron vychází, a mapování jeho funkcí na funkce umělého neuronu. Jsou vyjmenovány dostupné bázové funkce a aktivační funkce jako je skoková aktivační funkce nebo sigmoida. Následně se zabývá učením neuronu.

Práce se dále zabývá teorií umělých neuronových sítí, vysvětluje tři základní druhy učení sítí, které jsou adaptivní (algoritmus zpětného šíření chyby), soutěživé (Kohonenovo učení) a korelační (Hebbovo učení). Podle možností knihoven jsem vybral tři druhy sítí, na kterých jsem prováděl experimenty. Byl to vícevrstvý perceptron, Kohonenova síť a Hopfieldova síť.

Podle dostupných informací o knihovnách jsem navrhl vlastnosti, podle kterých se knihovny mezi sebou porovnávaly. Byly to nominální vlastnosti jako licence, podporované platformy, komunita nebo poslední aktualizace. Dle vybraných sítí jsem navrhl vhodné úlohy, na kterých jsem experimentoval. Pro vícevrstvý perceptron to byla úloha aproximace tří funkcí, pomocí Kohonenovy sítě jsem shlukoval body v rovině a Hopfieldovu síť jsem použil v úloze auto-asociace sedmi obrazů.

Pro porovnávání knihoven s Hopfieldovou sítí jsem vybral knihovny *Libann* a *Annie*. Porovnávání knihoven implementující Kohonenovu síť jsem provedl na knihovnách *Libann* a *KNNL*. Na experimenty s vícevrstvým perceptronem byly vybrány knihovna *FANN*, *Neu*, *OpenNN* a *OpenANN*.

Z výsledků porovnání a experimentů je zřejmé, že knihovny Hopfieldovu síť a Kohonenovu síť většinou implicitně neimplementují, uživatelé tyto sítě pravděpodobně vytvářejí manuálně. Lze to usuzovat ze stáří knihoven, které tyto sítě implementovaly, přičemž dokonce Kohonenova síť v knihovně *Libann* nebyla funkční. Většina knihoven se zaměřuje na vícevrstvý perceptron. Tyto knihovny jsou udržované a dále se vyvíjejí.

Práce nastínila, které knihovny jsou nyní volně na internetu a je možné je použít. Čtenáři dává doporučení, které knihovny by měl v danou situaci a problém, použít.

Literatura

[1] WANG, Ryue. Hopfield Network. *Harvey Mudd College* [online]. 2013 [cit. 2014-12-15]. Dostupné z: http://fourier.eng.hmc.edu/e161/lectures/nn/node5.html

[2] FRÄNTI, Pasi a Olli VIRMAJOKI. Iterative shrinking method for clustering problems. *Pattern Recognition* [online]. 2006, vol. 39, issue 5, s. 761-775 [cit. 2015-05-18]. DOI: 10.1016/j.patcog.2005.09.012. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S0031320305003778

[3] KÄRKKÄINEN, Ismo a Pasi FRÄNTI. *Dynamic local search algorithm for the clustering problem* [online]. Joensuu: University of Joensuu, 2002 [cit. 2015-05-18]. ISBN 95-245-8143-4; ISSN 0789-7316. Dostupné z: http://cs.uef.fi/sipu/pub/A-2002-6.pdf. Výzkumná zpráva. University of Joensuu.

[4] FIŠNAROVÁ, Simona. LDF MENDELU. *Euklidovský prostor. Funkce dvou proměnných: základní pojmy, limita a spojitost: prezentace* [online]. Brno, 2014, 25 s. [cit. 2014-12-13]. Dostupné z: http://user.mendelu.cz/fisnarov/imt/prednasky/funkce.pdf

[5] NGUYEN-THIEN, T. a T. TRAN-CONG. Approximation of functions and their derivatives: A neural network implementation with applications. *Applied Mathematical Modelling* [online]. 1999, vol. 23, issue 9, s. 687-704 [cit. 2015-05-14]. DOI: 10.1016/S0307-904X(99)00006-2. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S0307904X99000062

[6] FAHLMAN, Scott E. a Christian LEBIERE. *The Cascade-Correlation learnign architecture* [online]. Pittsburgh, 1990 [cit. 2014-12-09]. Dostupné z: http://www.cs.iastate.edu/~honavar/fahlman.pdf. Technical report. Carnegie Mellon University.

[7] VOJÁČEK, Antonín. *Samoučící se neuronová síť - SOM, Kohonenovy mapy: učební text* [online]. 2006, 8 s. [cit. 3.12.2014]. Dostupné z: http://www.kiv.zcu.cz/studies/predmety/uir/NS/Samouc_NN2.pdf

[8] VOLNÁ, Eva. *Neuronové sítě 1: skripta* [online]. Ostrava: Ostravská universita v Ostravě, 2008 [cit. 2014-12-03]. Dostupné z: http://www1.osu.cz/~volna/Neuronove_site_skripta.pdf

[9] RIEDMILLER a Heinrich BRAUN. A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. In: IEEE NEURAL

NETWORKS COUNCIL, Institute of Electrical and Electronics Engineers. *1993 IEEE International Conference on Neural Networks, San Francisco, California, March 28-April 1, 1993*. New York: IEEE, 1993, s. 586-591. ISBN 0780309995. Dostupné z: http://deeplearning.cs.cmu.edu/pdfs/Rprop.pdf

[10] ŠÍMA, Jiří. *Teoretické otázky neuronových sítí*. Vyd. 1. Praha: MATFYZ press, 1996, 390 s. ISBN 80-858-6318-9.

[11] FAUSETT, Laurene V. *Fundamentals of neural networks: architectures, algorithms, and applications*. Englewood Cliffs, NJ: Prentice-Hall, c1994, xvi, 461 p. ISBN 01-333-4186-0.

[12] ALIEV, R a R ALIEV. *Soft computing and its applications*. Singapore: World Scientific, 2001, xiv, 444 s. ISBN 98-102-4700-1.

[13] MEHROTRA, Kishan. *Elements of artificial neural networks*. Cambridge, Mass.: MIT Press, c1997, xiv, 344 s. ISBN 978-0-262-13328-9.

[14] RUSSELL, Stuart J a Peter NORVIG. *Artificial intelligence: a modern approach*. 3rd ed. Upper Saddle River: Prentice Hall, 2010, xviii, 1132 s. Prentice Hall series in artificial intelligence. ISBN 978-0-13-604259-4.

Příloha A

Obsah CD

- src
  - libann
  - annie
  - knnl
  - fann
  - neu
  - opennn
  - openann
- tex
- lib
- BP.pdf