

# Umělé neuronové sítě v PHP

Artificial neural networks in PHP

Petr Juráček

---

Bakalářská práce  
2014



Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky

---

Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

akademický rok: 2013/2014

## ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Petr Juráček**  
Osobní číslo: **A11638**  
Studijní program: **B3902 Inženýrská informatika**  
Studijní obor: **Informační a řídicí technologie**  
Forma studia: **kombinovaná**

Téma práce: **Umělé neuronové sítě v PHP**

Zásady pro vypracování:

1. Seznamte se s umělými neuronovými sítěmi.
2. Seznamte se s možnostmi PHP.
3. Vytvořte různé typy neuronových sítí.
4. Vytvořte funkční příklady s použitím vytvořených neuronových sítí.
5. Funkční příklady zpracujte formou podpůrné webové prezentace pro předmět Umělé neuronové sítě.

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. ZELINKA, Ivan.: Umělá inteligence I. Volume 1. Zlín: Vutium, Brno, 1998, 126 p. ISBN 80-214-1163-5.
2. ŠNOREK M., JIŘINA M.: Neuronové sítě a neuropočítače, ČVUT, 1996, ISBN 80-01-01455-X.
3. BÍLA J.: Umělá inteligence a neuronové sítě v aplikacích, ČVUT, 1996, ISBN 80-01-01275-1.
4. BOSE N.K., LIANG P.: Neural Network Fundamentals with Graphs, Algorithms, and Applications, McGraw-Hill Series in Electrical and Computer Engineering, 1996, ISBN 0-07-006618-3.
5. GUTMANS, Andi, Stig Sather BAKKEN a Derick RETHANS: Mistrovství v PHP 5. Vyd. 1. Brno: CP Books, 2005, 655 s., ISBN 802510799x.
6. VRÁNA, Jakub: 1001 tipů a triků pro PHP. Vyd. 1. Brno: Computer Press, 2010, 456 s. ISBN 9788025129401.
7. ŠÍMA J., NERUDA R.: Teoretické otázky neuronových sítí. Vyd. 1. Praha, 1996: Matfyzpress, 390 s., ISBN 80-85863-18-9

Vedoucí bakalářské práce:

**doc. Ing. Zuzana Komínková Oplatková, Ph.D.**  
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce:

**28. února 2014**

Termín odevzdání bakalářské práce:

**13. června 2014**

Ve Zlíně dne 28. února 2014



prof. Ing. Vladimír Vašek, CSc.  
*děkan*



prof. Ing. Vladimír Vašek, CSc.  
*ředitel ústavu*

## **ABSTRAKT**

Cílem této bakalářské práce je vytvořit webovou aplikaci, která bude sloužit jako podklad k předmětům Umělé neuronové sítě a Metody umělé inteligence. Aplikace obsahuje příklady umělých neuronových sítí. Bakalářská práce je rozdělena na dvě části. Teoretická část obsahuje úvod do umělých neuronových sítí. Praktická část obsahuje návod k webové aplikaci a popis jednotlivých funkcí.

Klíčová slova: umělé neuronové sítě, umělá inteligence.

## **ABSTRACT**

The aim of this bachelor's project is to create web application that will be used for subjects Artificial neural networks and Methods of artificial intelligence. The application includes examples of artificial neural networks. The bachelor's project is divided into two parts. The theoretical part describes the PHP scripting language and an introduction to artificial neural networks. The practical part contains instructions for a Web application and a description of each function.

Keywords: artificial neural networks, artificial intelligence

Zde bych rád poděkoval své vedoucí bakalářské práce, paní doc. Ing. Zuzaně Komínkové Oplatkové, Ph.D. za vedení při mé práci. Dále děkuji rodičům, přátelům a své přítelkyni za podporu při studiu.

**Prohlašuji, že**

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

**Prohlašuji,**

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....  
podpis diplomanta

**OBSAH**

<b>ÚVOD.....</b>	<b>9</b>
<b>I TEORETICKÁ ČÁST.....</b>	<b>10</b>
<b>1 HISTORIE .....</b>	<b>11</b>
1.1 RENESANCE NEURONOVÝCH SÍTÍ.....	11
<b>2 VYUŽITÍ UMĚLÝCH NEURONOVÝCH SÍTÍ.....</b>	<b>12</b>
<b>3 NEURON.....</b>	<b>13</b>
3.1 BIOLOGICKÝ NEURON.....	13
3.1.1 Synapse .....	14
3.2 UMĚLÝ NEURON .....	15
3.2.1 Vstupy neuronu .....	15
3.2.2 Synaptické váhy vstupů.....	15
3.2.3 Práh neuronu .....	16
3.2.4 Vnitřní potenciál neuronu .....	16
3.3 PŘENOSOVÁ FUNKCE NEURONU.....	16
3.3.1 Standardní (logistická) sigmoida.....	17
3.3.2 Hyperbolický tangens.....	17
3.3.3 Perceptron .....	18
3.3.4 Lineární přenos.....	18
3.3.5 Lineární omezený přenos .....	19
3.3.6 Binární přenos (Heavisideův skok).....	19
3.3.7 Přenos RBF (radiální báze) .....	20
<b>4 NEURONOVÁ SÍŤ .....</b>	<b>21</b>
4.1 UČENÍ SÍTĚ .....	21
4.1.1 S učitelem.....	21
4.1.2 Bez učitele .....	21
4.2 TYPY NEURONOVÝCH SÍTÍ .....	22
4.2.1 Dopředná neuronová síť .....	22
4.2.1.1 Učící algoritmus Backpropagation .....	22
4.2.1.2 Perceptron .....	23
4.2.1.3 Vícevrstvá neuronová síť .....	24
4.2.2 Rekurentní neuronová síť .....	24
4.2.2.1 Hopfieldova síť .....	24
4.2.3 Kohonenovy mapy .....	25
<b>II PRAKTICKÁ ČÁST .....</b>	<b>26</b>
<b>5 O APLIKACI.....</b>	<b>27</b>
5.1 VYUŽITÍ KNIHOVNY ANN FOR PHP5 .....	27
5.1.1 Learning rate – učící koeficient.....	27
5.1.2 Počet neuronů ve skrytých vrstvách.....	28
5.1.3 Přenosová funkce .....	28
5.1.4 Učící algoritmus .....	29
5.2 OVLÁDACÍ PRVKY APLIKACE.....	29
5.2.1 Tlačítko: Trénovat síť.....	29
5.2.1.1 Logování vah sítě.....	30

5.2.1.2	Topologie sítě .....	32
5.2.1.3	Index náročnosti.....	32
5.2.2	Tlačítko: Log vah .....	33
5.2.3	Tlačítko: Detail sítě .....	33
5.2.4	Tlačítko: Reset .....	34
5.3	NAUČENÁ NEURONOVÁ SÍŤ .....	34
5.3.1	Informace o síti .....	34
5.3.2	Náhled topologie sítě.....	35
5.3.3	Tester sítě .....	35
5.3.4	Trénovací množina.....	36
5.3.5	Stáhnout síť .....	36
5.4	SEKCE: NEURON.....	37
5.5	SEKCE: LOGICKÉ FUNKCE.....	39
5.6	SEKCE: DETEKCE JAZYKA SÍTÍ.....	40
5.7	SEKCE: PREDIKCE SÍTÍ .....	41
5.8	SEKCE: NAČTENÍ SÍTĚ ZE SOUBORŮ .....	42
5.9	SPRÁVA SOUBORŮ .....	43
5.9.1	Ukládání souborů .....	43
5.9.2	Kontrola a mazání souborů .....	44
5.10	BEZPEČNOST APLIKACE .....	44
<b>ZÁVĚR .....</b>		<b>45</b>
<b>ZÁVĚR V ANGLIČTINĚ.....</b>		<b>46</b>
<b>SEZNAM POUŽITÉ LITERATURY.....</b>		<b>47</b>
<b>SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....</b>		<b>49</b>
<b>SEZNAM OBRÁZKŮ .....</b>		<b>50</b>
<b>SEZNAM TABULEK.....</b>		<b>51</b>



## ÚVOD

Jedním z cílů informatiky je vytvoření umělé inteligence – stroje, jenž by byl schopen samostatného myšlení a uvažování. Proto vznikly umělé neuronové sítě, které si za vzor vzali strukturu biologické neuronové sítě. Umělá neuronová síť je struktura vzájemně propojených umělých (formálních) neuronů. Jednoduše lze říci, že umělá neuronová síť je matematickým modelem biologické neuronové sítě. [2]

Umělé neuronové sítě se v dnešní době používají k řešení mnoha problémů. Mezi nejčastější patří například šifrování, filtrování spamu, rozpoznávání zvuků a obrazů. Neuronových sítí se využívá také například v biometrii při autentizaci otisků prstů, nebo autentizaci oční duhovky [1].

PHP je v dnešní době nejrozšířenějším skriptovacím jazykem pro webové prezentace jakéhokoli rozsahu. Existuje velké množství nejrozličnějších knihoven pro tento jazyk. Jednou z těchto knihoven je i knihovna ANN for PHP5, kterou jsem si zvolil pro zpracování příkladů neuronových sítí.

Elementární příklady umělých neuronových sítí lze vypracovat mnoha způsoby. PHP jsem zvolil z důvodů jednoduchosti přístupu přes internetový prohlížeč nezávisle na platformě. Jednotlivé umělé neuronové sítě jsou interaktivní, což znamená, že uživatel může měnit vstupy do neuronové sítě a následně sledovat změny výstupů neuronové sítě.

## **I. TEORETICKÁ ČÁST**

## 1 HISTORIE

Vytvoření prvních umělých neuronů se datuje do roku 1943, kdy byl vytvořen model prvního umělého neuronu – perceptronu, Warrenem McCullochem a jeho studentem Walterem Pittsem. Perceptron je hojně využíván v modifikované podobě dodnes. V roce 1958 využil tento model F. Rosenblatt, a vytvořil tak první neuronovou síť. Síť byla schopna generovat pouze binární výstupy, tedy 0 a 1 podle toho, jestli vážená suma vstupů překročila práh neuronu. Neexistoval však žádný učicí algoritmus a síť byla schopna řešit pouze lineárně separabilní problémy. Tohoto úskalí využil Rosenblattův spolužák M. Minsky, který tento nedostatek napadl ve své knize „*Perceptron*“ z roku 1969. Zájem o síť tehdy opadnul až do poloviny 80. let. [1]

Od konce 60. let, kdy přestal být o neuronové síti zájem, probíhal jejich výzkum izolovaně mimo USA, protože zde měla kniha „*Perceptron*“ velký vliv. Někteří talentovaní badatelé se ale odradit nenechali, a díky tomu přispěli svými pracemi k rozvoji neuronových sítí do podoby, jak je známe dnes. [2]

### 1.1 Renesance neuronových sítí

Renesance nastala v polovině 80. let, kdy průkopníci D. Rumelhart, G. Hinton a R. Williams vytvořili práci „*Learning Internal Representation by Error Propagation*“, která se zabývá vícevrstevnými neuronovými sítěmi. Tyto sítě jsou již schopny řešit lineárně neseperabilní problémy, což byl pro vývoj umělých neuronových sítí velký průlom. Následně začali vznikat další významné sítě, jako jsou například Hopfieldova, Kohonenova a Grossbergova síť. [1][2]

Umělé neuronové sítě prošly velkým vývojem a jsou úspěšně používány dodnes k řešení nejrůznějších typů problémů, o kterých se zmíním v další kapitole.

## 2 VYUŽITÍ UMĚLÝCH NEURONOVÝCH SÍTÍ

Umělé neuronové sítě mají nespočet využití. Jejich výhoda oproti počítačům spočívá ve schopnosti adaptace na určitý problém. Adaptace plyne z jejich schopnosti se učit, tedy adaptovat se. Následující tabulka názorně popisuje rozdíly mezi počítačem a neuronovou sítí. [1]

NEURONOVÁ SÍŤ	POČÍTAČ
Je určena nastavováním vah, prahů a struktury.	Je programován instrukcemi (if, else, then, switch, go to...)
Paměťové a výkonné prvky jsou uspořádány spolu.	Proces a paměť pro něj jsou separovány.
Paralelismus	Sekvenčnost
Tolerují odchylky od originálních informací	Netolerují odchylky
Samoorganizace během učení	Neměnnost programu.

Tabulka 1: Rozdíly mezi neuronovou sítí a počítačem

Využití neuronových sítí na obecných případech [1]:

- Identifikace různých signálů
- Klasifikace objektů
- Predikce – předpovídání dle minulých hodnot
- Řešení matematických funkcí
- Filtrace
- Optimalizace

Konkrétní příklady využití [3]:

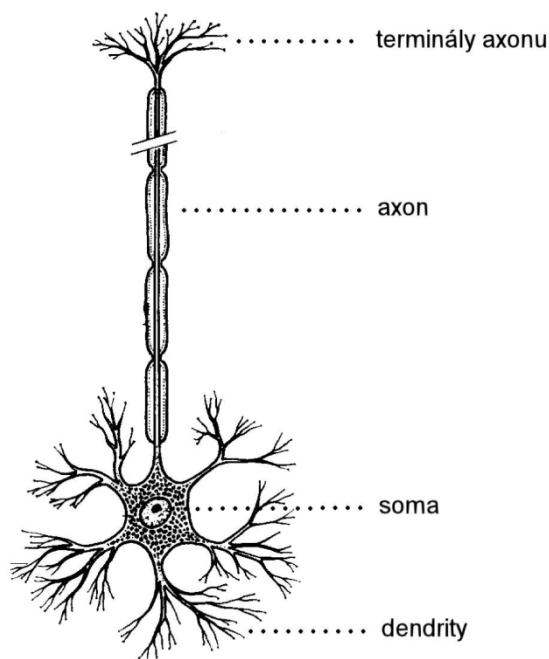
- Predikce spotřeby energie
- Detekce rakoviny z několika výsledků vyšetření
- Řízení letového provozu
- Filtrování nevyžádané pošty
- Predikce na akciovém trhu
- Inteligentní router v síti – učí se nejrychlejší trasy
- Predikce prodeje produktů

### 3 NEURON

#### 3.1 Biologický neuron

Biologický neuron, neboli také nervová buňka, je základní funkční jednotkou nervové tkáně. Tyto buňky jsou schopné přijmout, vést a zpracovat speciální signál. Neurony přijímají signály z vnitřního i vnějšího prostředí a reagují na ně výstupními signály. [2]

Mozková kůra člověka může být tvořena 13 až 15 miliardami neuronů, z nichž každý může být propojen s dalšími až 5000 neurony. [2]



Obrázek 1: Biologický neuron [2]

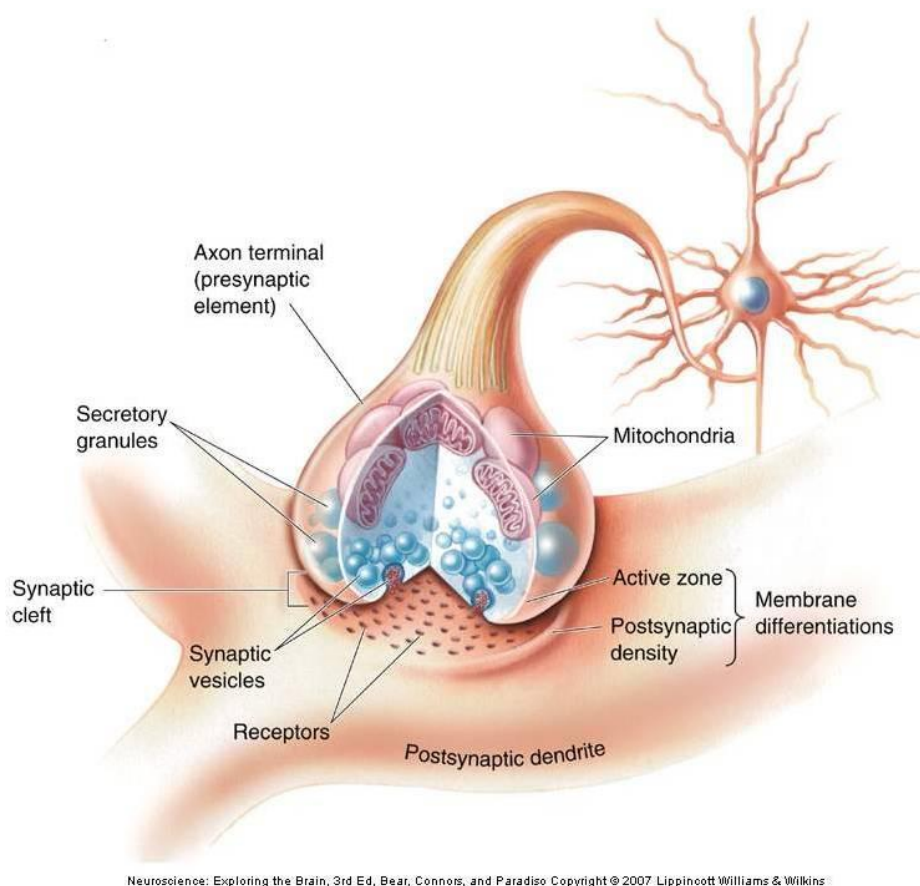
Části biologického neuronu:

- terminály axonu
- axon – odstředivý výběžek neuronu
- soma – vlastní tělo neuronu
- dendrity – dostředivé výběžky neuronu

### 3.1.1 Synapse

Neurony spolu komunikují pomocí specializovaných struktur, tzv. synapsí. Terminály axonu při podráždění uvolňují neurotransmitery do synaptické štěrby. Neurotransmitter je zpravidla nízkomolekulární chemická látka, která přirozeným způsobem vzniká v nervové soustavě živočichů a slouží v ní k přenášení vzruchů. Synaptická štěrbina je prostor mezi terminály axonu a dendrity dalšího neuronu. Mezi axonem a dendrity probíhají přenosy chemické a uvnitř neuronu přenosy elektrické. [3]

Elektrické vazby jsou vývojově starší oproti chemickým. Chemické se vyskytují převážně u savců. [4]



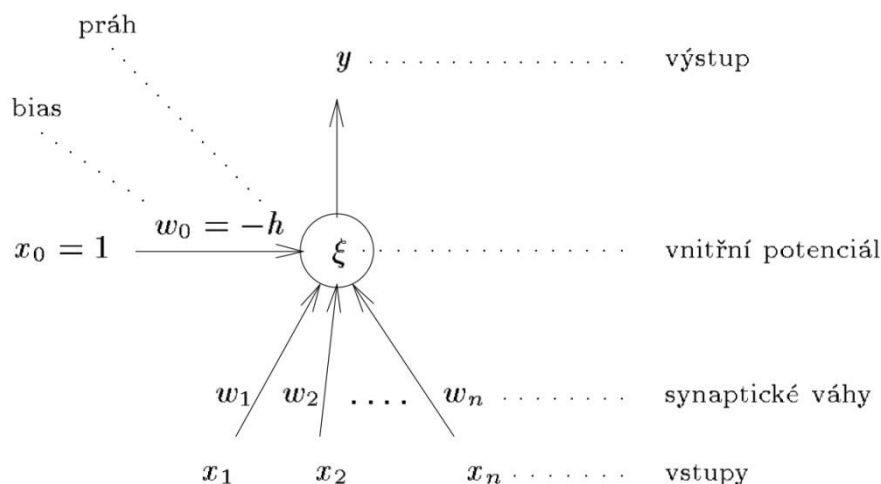
Obrázek 2: Synapse

Neurotransmitery dělíme na [1][2]:

- krátkodobě působící – přenos
- dlouhodobě působící – paměť

### 3.2 Umělý neuron

Neboli také formální neuron, je získán přeformulováním neurofyzilogického neuronu do matematické řeči [2]. Ve skutečnosti je umělý neuron mnohem jednodušší než biologický neuron. [1]



Obrázek 3: Umělý (formální) neuron [2]

Vstupy  $x_1 - x_n$  představují v matematickém modelu dendrity biologického neuronu. Propustnost vstupů určují synaptické váhy  $w_1 - w_n$  [2].

Umělý neuron je schopen pracovat ve dvou režimech [4]:

- excitační – vybuzující (šíření vzruchu v neuronové síti)
- inhibiční – tlumící (utlumení vzruchu v neuronové síti)

#### 3.2.1 Vstupy neuronu

Pomocí vstupů získává neuron informaci jako reálné číslo. Tyto vstupy mohou pocházet z vnějšího prostředí nebo také jako výstupy z jiných neuronů [2].

#### 3.2.2 Synaptické váhy vstupů

Každý vstup má určitou důležitost. Tuto důležitost vstupům přiřazují jejich váhy, které mají zpravidla reálnou hodnotu. Chceme-li docílit správného řešení problému pomocí

neuronové sítě, máme nekonečně mnoho řešení, jak adaptovat váhy pro vstupy do jednotlivých neuronů, abychom dosáhli potřebného výsledku [2].

### 3.2.3 Práh neuronu

Pokud aktivita neuronu překročí tento práh, přepne se neuron do aktivního stavu a vyšle signál na výstup. Dokud není práh překročen, je neuron pasivní a na jeho výstupu není žádná hodnota. [2]

### 3.2.4 Vnitřní potenciál neuronu

Nazývá se též aktivační funkce neuronu nebo přenos neuronu. Vnitřní potenciál neuronu vyjadřuje zvážená suma vstupů  $x_1 - x_n$  [4]. Můžeme jej vyjádřit vztahem

$$a = \sum_{i=1}^n w_i x_i + b w_b \quad (1)$$

Dokud je vnitřní potenciál  $a$  menší než prahová hodnota  $h$ , je neuron v pasivním stavu. Dosažením prahové hodnoty  $h$ , se neuron aktivuje. [2]

## 3.3 Přenosová funkce neuronu

Přenosovou funkci získáme z aktivační funkce vztahem [4]:

$$f(a) = TF \left( \sum_{i=1}^n w_i x_i + b w_b \right) \quad (2)$$

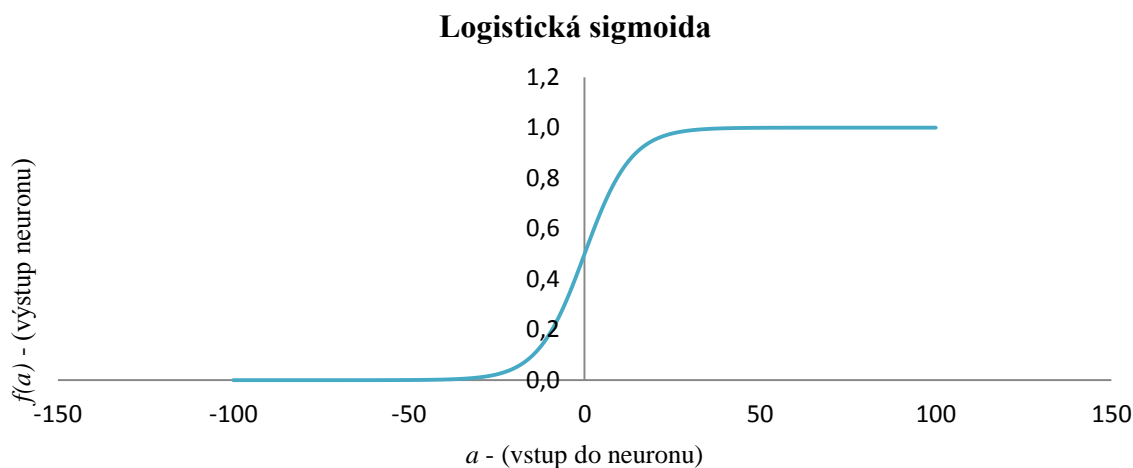
Neurony mohou být binární nebo spojité a mohou mít různé typy přenosových funkcí, které si následně popíšeme [5].



### 3.3.1 Standardní (logistická) sigmoida

Přenosová funkce:

$$f(a) = \frac{1}{1 + e^{-\lambda a}} \quad (3)$$

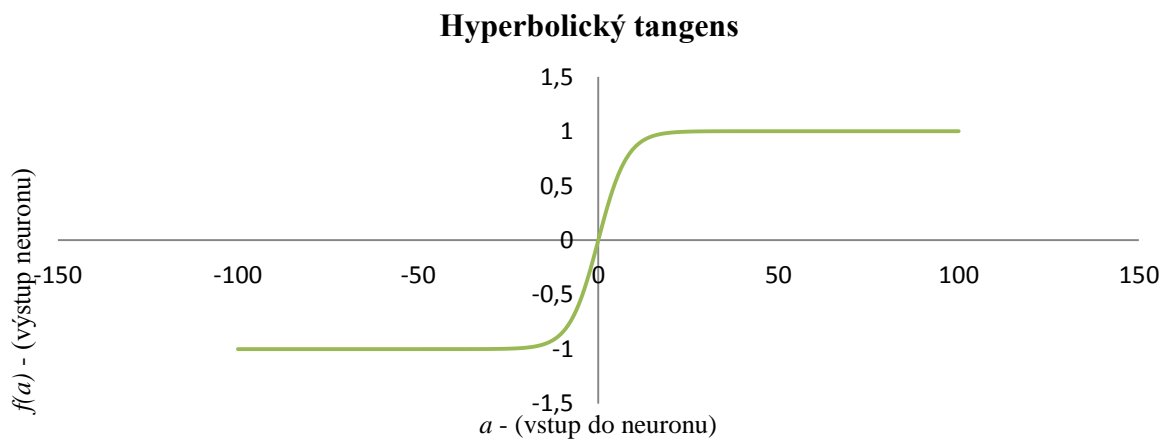


Obrázek 4: Přenosová funkce - logistická sigmoida

### 3.3.2 Hyperbolický tangens

Přenosová funkce:

$$f(a) = \frac{1 - e^{-ka}}{1 + e^{-ka}} \quad (4)$$

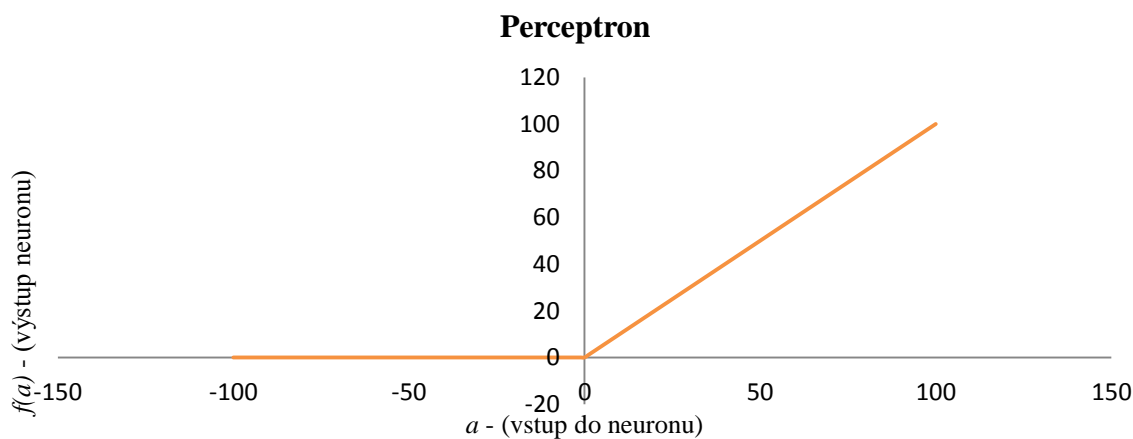


Obrázek 5: Přenosová funkce - hyperbolický tangens

### 3.3.3 Perceptron

Přenosová funkce:

$$\forall a > 0: f(a) = a \quad \wedge \quad \forall a \leq 0: f(a) = 0 \quad (5)$$



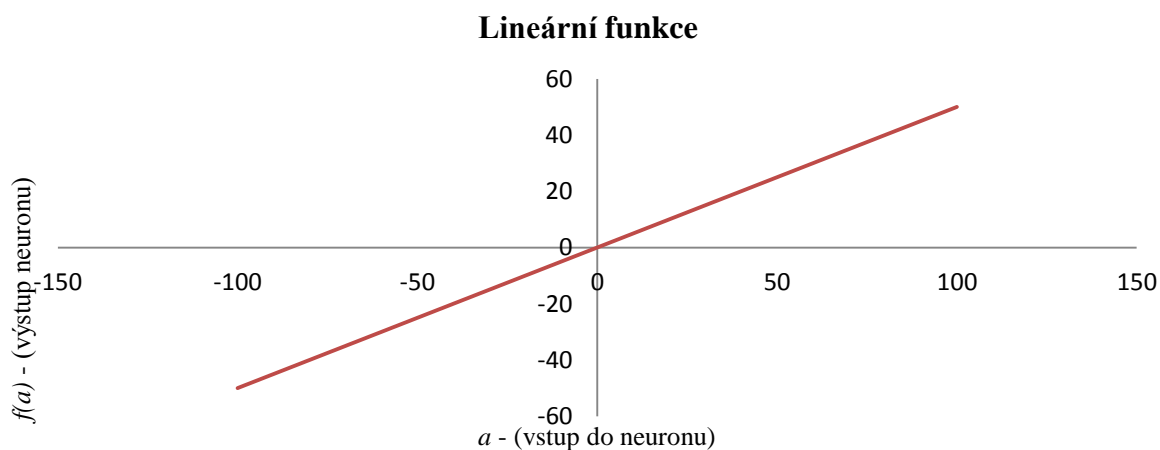
Obrázek 6: Přenosová funkce – perceptron

### 3.3.4 Lineární přenos

Přenosová funkce:

$$f(a) = k \cdot a \quad (6)$$

kde  $k$  je směrnice.

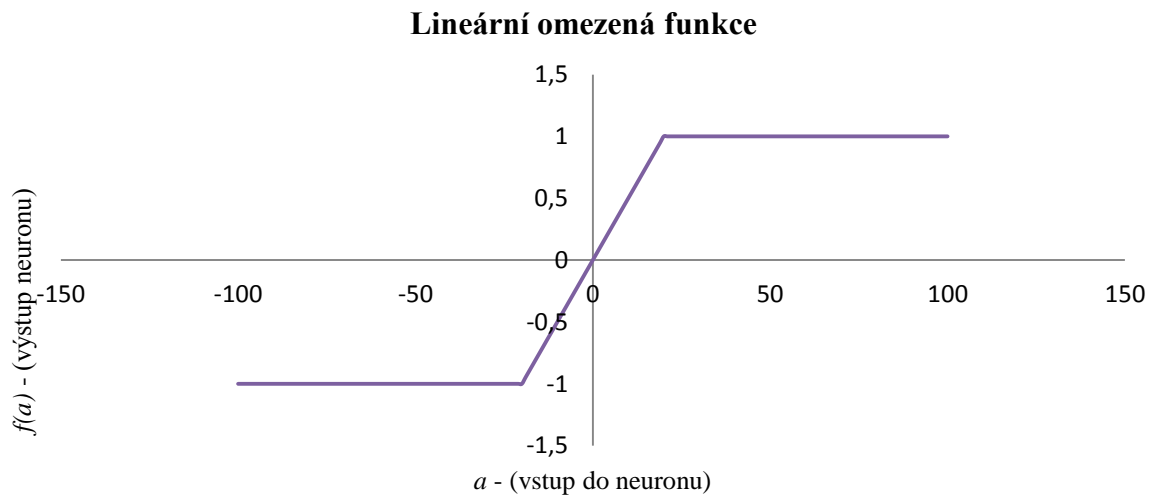


Obrázek 7: Přenosová funkce - lineární funkce

### 3.3.5 Lineární omezený přenos

Přenosová funkce:

$$\begin{aligned} \forall a \in \langle -h_1, h_1 \rangle: f(a) &= k \cdot a \quad \wedge \\ \forall a > h_1: f(a) &= \gamma \quad \wedge \\ \forall a < -h_1: f(a) &= -\gamma \end{aligned} \quad (7)$$



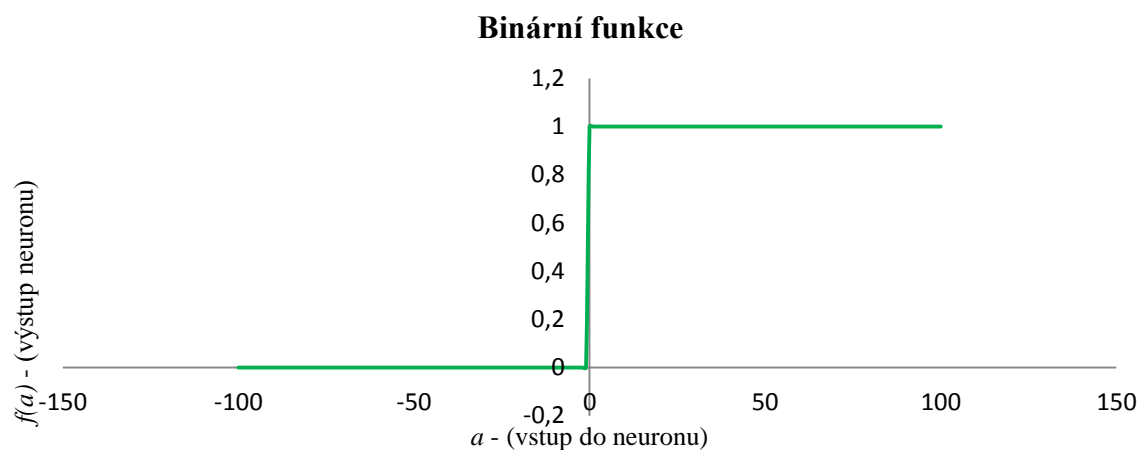
Obrázek 8: Přenosová funkce - lineární omezená funkce

### 3.3.6 Binární přenos (Heavisideův skok)

Rozlišujeme unipolární a bipolární, podle toho, zda může výstup umělého neuronu dosahovat kladných i záporných hodnot. Pokud výstup dosahuje kladných i záporných výstupů, je přenos bipolární. [4]

Přenosová funkce:

$$\forall a > 0: f(a) = 1 \quad \wedge \quad \forall a \leq 0: f(a) = 0(-1) \quad (8)$$

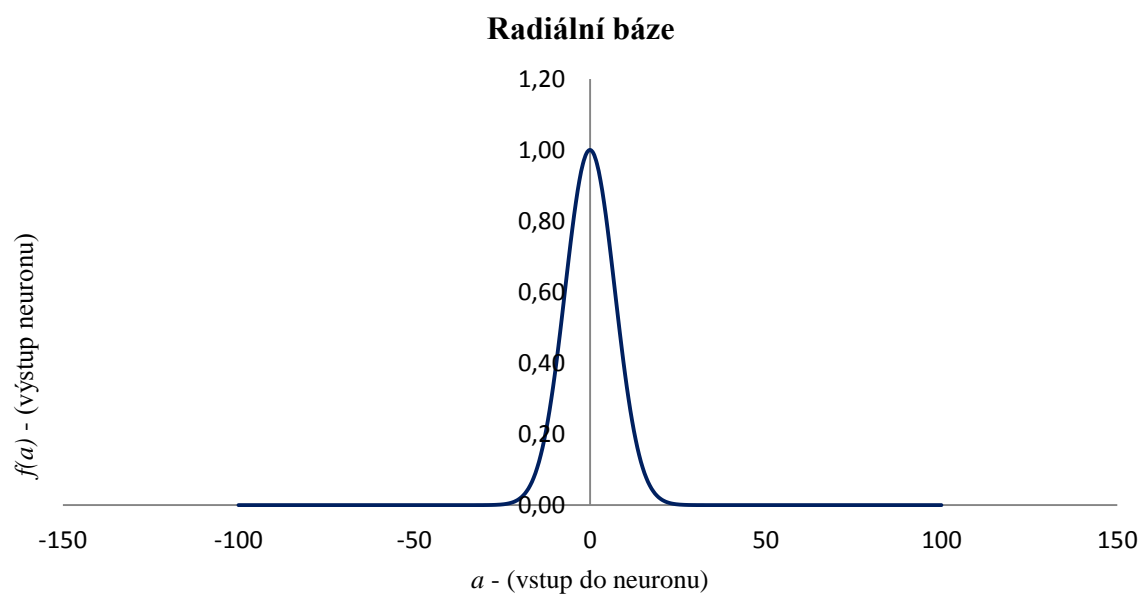


Obrázek 9: Přenosová funkce - binární funkce

### 3.3.7 Přenos RBF (radiální báze)

Přenosová funkce:

$$f(a) = e^{-ka^2} \quad (9)$$



Obrázek 10: Přenosová funkce - radiální báze

## 4 NEURONOVÁ SÍŤ

Neuronová síť je propojení dvou nebo více formálních neuronů. Výstup jednoho neuronu je zpravidla kombinací více vstupů. Tento princip vychází z biologických neuronových sítí, kde jsou neurony propojeny pomocí terminálů axonu s dendrity jiného neuronu. Počet neuronů a jejich vzájemné propojení určuje topologii (architekturu) neuronové sítě. [2]

V neuronové síti mohou být tři typy neuronů [4]:

- vstupní
- skryté
- výstupní

Neuronová síť se vyvíjí v čase. Mohou se měnit váhy jednotlivých neuronů, ale také jejich vzájemné propojení [2]. Tyto změny v síti určují aktuální pracovní režim sítě:

- Organizační – změna topologie sítě
- Adaptivní – změna konfigurace a adaptace vah neuronů
- Aktivní – změna stavu jednotlivých neuronů

U biologických neuronových sítí nejsou pracovní režimy rozlišovány, jelikož všechny změny v síti probíhají současně [2].

### 4.1 Učení sítě

#### 4.1.1 S učitelem

Síť se učí pomocí vzorové trénovací množiny. Máme vždy sadu vstupů a k nim přiřazené výstupy, které síti předkládáme. Pomocí těchto sad se síť přibližuje požadovanému výstupu. Je zde tedy využita zpětná vazba „učitele“, který síti řekne, jestli je výstup správný nebo ne. V případě že ne, provádí se korekce vah, dokud není výstup sítě optimální. [5]

#### 4.1.2 Bez učitele

Při učení bez učitele nepředkládáme síti známý výstup, čili neporovnáváme aktuální výstup sítě s referenčním výstupem. Síť se rozhoduje sama a sama si ze vstupních vzorů vytváří skupiny podle podobnosti a následně reaguje na typického zástupce. [5]

## 4.2 Typy neuronových sítí

### 4.2.1 Dopředná neuronová síť

#### 4.2.1.1 Učící algoritmus Backpropagation

Backpropagation je zkratkou anglického spojení „*backward propagation errors*“, což by se dalo přeložit jako „*zpětné šíření chyb*“. Je to nejpoužívanější učící algoritmus pro vícevrstvé dopředné neuronové sítě. Používá se přibližně v 80% všech aplikací neuronových sítí [2]. Backpropagation vyžaduje, aby aktivační funkce byla diferencovatelná. Nastavení vah probíhá opačným směrem, než se šíří vstupní signál sítě.

Pro naučení sítě požaduje tento učící algoritmus sadu požadovaných výstupů se vstupy k nim přiřazenými. Množina těchto sad se nazývá trénovací množina. Učení sítě by se dalo přirovnat k učení malého dítěte. Pokud chceme dítě naučit rozpoznat například psa, musíme mu předkládat různé rasy psů a říci mu, že se jedná o psa. Tím jsme mu předložili vstup (konkrétní rasu psa) a výstup v podobě klasifikace (pes). [1]

Algoritmus Backpropagation se skládá ze dvou fází:

#### 1. Aktivační

Při inicializaci sítě se volí váhy náhodně v rozmezí  $\langle -0,5; 0,5 \rangle$ . Lze k tomu použít například generátor náhodných čísel.

Sítí se předloží vstupní vektor složený z jednotlivých vstupů. Tento vektor se dále šíří sítí. Ve vstupní vrstvě je rozvětven a každá větev je ohodnocena svou vahou. Touto vahou se vždy násobí vstupní hodnota do neuronu. Uvnitř neuronu se takto váhami vynásobené vstupy sečtou a použijí se jako argument přenosové funkce. Funkční hodnota přenosové funkce pro tuto sumu zvážených vstupů je zároveň i výstupní hodnotou neuronu, která se dále šíří sítí jako vstup do dalšího neuronu, nebo, jedná-li se o neuron výstupní vrstvy, na výstup sítě. [1]

#### 2. Adaptační

Tato fáze je vlastní algoritmus Backpropagation. Získaný výstupní vektor z první (aktivační) fáze se porovná s vektorem požadovaných výstupů sítě. Rozdíl mezi těmito dvěma vektory je dále použit pro výpočet nových vah neuronů. Nejprve se upraví váhy u spojů vstupujících do výstupní vrstvy, poté v nižší vrstvě, a tak se

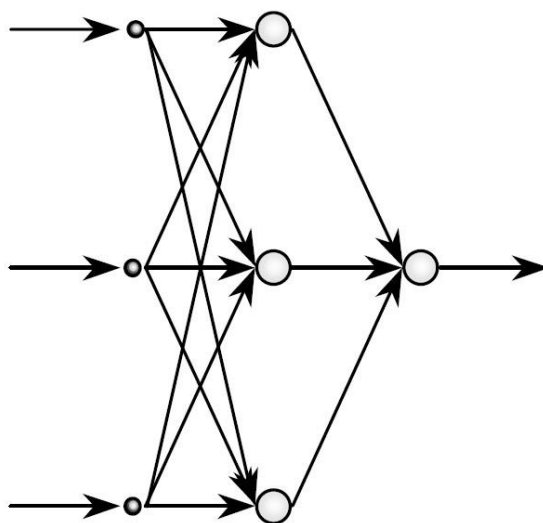
dále pokračuje, než dosáhneme vstupní vrstvy. Dosažením vstupní vrstvy je tato fáze ukončena a pokračujeme opět fází první. [1]

*„Při každém porovnávání výstupní odezvy s požadovaným originálem se daný rozdíl uchová v paměťové proměnné a sumarizuje se s dalšími postupně získanými rozdíly. Takto získané číslo za celou trénovací množinu (epochu) se nazývá globální chyba. Tato globální chyba je po každé epoše kontrolována s chybou, kterou zadal uživatel a pokud je nižší, než chyba zadaná, pak je síť naučena a učení končí. Algoritmus tedy hledá globální minimum chybové funkce.“ [1, s. 36]*

#### 4.2.1.2 Perceptron

*„Perceptron je síť s jednou pevnou (nepřestavitelnou) vstupní vrstvou a s výstupní vrstvou, jejíž váhy se mohou měnit. Její neurony jsou schopny funkční transformace ze vstupu na výstup díky měnitelnosti prahů a vah ve výstupní vrstvě.“ [1, s. 29]*

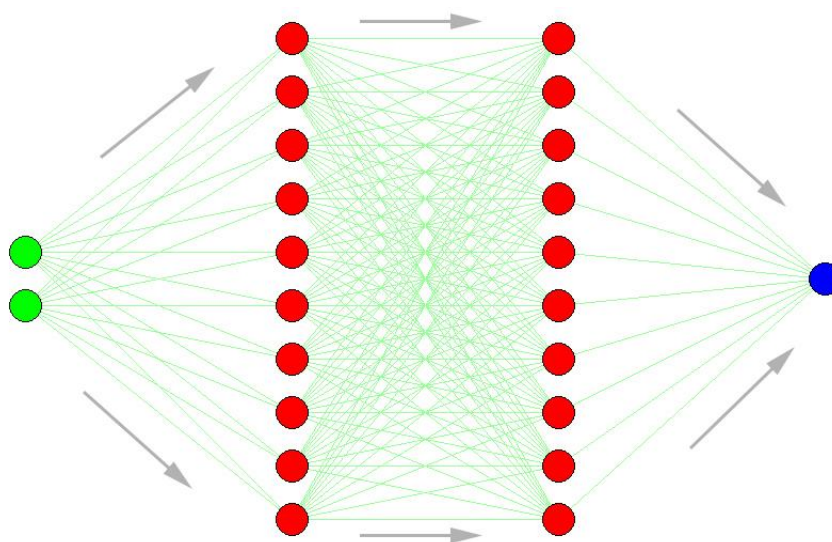
Podíváme-li se na perceptron z pohledu měnitelnosti vah, můžeme síť považovat za jednovrstvou neuronovou síť. Na obrázku (Obrázek 11) vidíme jednovrstvou síť typu perceptron. Tato síť nemá mezi vstupy a neurony skryté vrstvy přestavitelné váhy. První vrstva pouze provádí konstantní transformaci ze vstupu [1].



Obrázek 11: Schéma perceptronu [1]

#### 4.2.1.3 Vícevrstvá neuronová síť

„Vícevrstvá síť je složená z vrstev, které jsou mezi sebou propojeny směrem ze vstupu na výstup. Jako učící algoritmus se obvykle používá Backpropagation, který při adaptační fázi přenastavuje váhy sítě tak, aby se odezva sítě co nejvíce blížila požadované hodnotě.“ [1, s. 35]



Obrázek 12: Vícevrstvá neuronová síť

Vícevrstvá neuronová síť nemusí mít vždy stejný počet neuronů ve všech skrytých vrstvách, jako je tomu na obrázku (Obrázek 12). Topologie může být různorodá, například skryté vrstvy můžou tvořit trojúhelník.

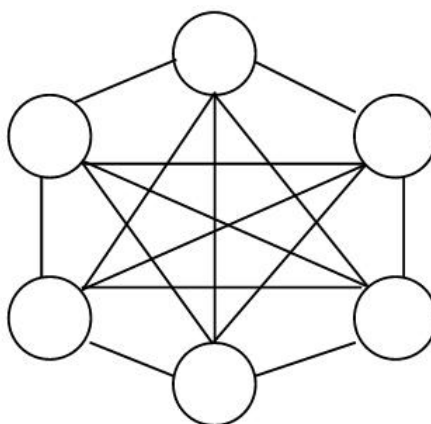
#### 4.2.2 Rekurentní neuronová síť

##### 4.2.2.1 Hopfieldova síť

„Hopfieldova síť je síť s jednou vrstvou a algoritmem učení bez učitele. Přenosové funkce jsou skokového charakteru.“ [1, s. 43]

Zakladatelem této sítě je fyzik John Hopfield. Tato síť se hojně používá pro rekonstrukci neúplných, šumem poškozených obrazů. Své uplatnění nalezne také v řešení optimalizačních problémů nebo jako autoasociativní paměť. Síť se skládá z navzájem oboustranně propojených neuronů. [6]



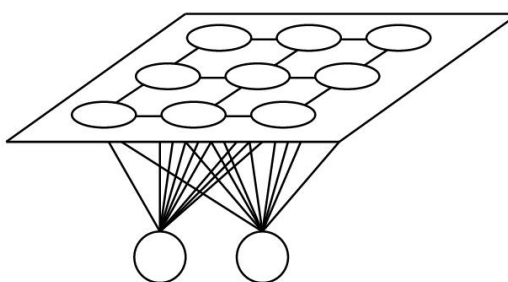


Obrázek 13: Hopfieldova síť [6]

#### 4.2.3 Kohonenovy mapy

*„Kohonenova síť nevyžaduje ke svému učení učitele. Je jednovrstvá s tím, že tato vrstva může být uspořádána v řádku nebo v ploše.“ [1, s. 53]*

Základní myšlenkou tohoto typu sítí je nalezení prostorové reprezentace složitých datových struktur. Účelem je, aby třídy si podobných vektorů byly reprezentovány neurony, které jsou si v dané topologii blízké. Analogii lze nalézt například i v lidském mozku. Každý konec mozkové kůry reaguje na jiný typ frekvencí. Jeden konec zpracovává nízké frekvence a opačný konec zpracovává frekvence vysoké. Díky tomu lze mnohorozměrová data zobrazit v jednodušším prostoru. [6]



Obrázek 14: Kohonenova mapa [6]

## **II. PRAKTICKÁ ČÁST**

## 5 O APLIKACI

Webová aplikace využívající knihovny umělých neuronových sítí pro PHP, je tvořena několika sekcemi. Jednotlivé sekce simulují určitý problém, který je řešen pomocí umělé neuronové sítě. Aplikace poskytuje možnost exportování konfiguračních souborů neuronové sítě, což umožňuje síť opět rekonstruovat bez nutnosti opětovného nastavování parametrů umělé neuronové sítě. Celá aplikace je dostupná na internetu [19, dočasné umístění].

### 5.1 Využití knihovny ANN for PHP5

Projekt je založen na práci Eddy Younga z roku 2002. Zdrojového kódu se ujal Thomas Wien z německého Düsseldorfu a provedl řadu změn a vylepšení, díky kterým je projekt použitelný pro řešení řady situací.

Rozhodl jsem se vytvořit aplikaci využívající umělé neuronové sítě v PHP, z důvodu dostupnosti z jakékoliv platformy. Další výhodou PHP je také to, že uživatel nemusí instalovat žádný software a vystačí si pouze s internetovým prohlížečem.

Knihovna spadá pod licenci BSD 2-Clause License. Kompletní formulace licence je k dispozici na internetu [18], stejně jako kompletní balíček knihovny [17].

Knihovna je dostačující pro účel demonstrace funkce umělých neuronových sítí, nicméně některé postupy a algoritmy jsou zjednodušeny kvůli ušetření výpočetního času. V následujících podkapitolách uvádím upřesnění těchto zjednodušení.

#### 5.1.1 Learning rate – učící koeficient

Koeficient je přepočítáván dynamicky v průběhu učícího algoritmu a jeho zjednodušení spočívá v tom, že se dynamicky zmenšuje nebo zvětšuje pouze o hodnotu 0,5. Průběh změn koeficientu napříč učícím algoritmem lze exportovat a sledovat jako hodnotu označenou zkratkou *E*.

### 5.1.2 Počet neuronů ve skrytých vrstvách

Při vytváření objektu umělé neuronové sítě požaduje konstruktor tři celočíselné hodnoty:

- počet skrytých vrstev
- počet neuronů v jedné skryté vrstvě
- počet výstupů

Nelze však definovat počet neuronů ve vrstvě pro každou vrstvu zvlášť. Proto nastavení topologie umělé neuronové sítě není dostatečně flexibilní.

```
1 $network = new Network(2, 4, 1);
```

Uvedený zdrojový kód vytvoří objekt sítě, která bude mít:

- 2 skryté vrstvy
- 4 neurony v jedné skryté vrstvě
- 1 výstup ze sítě

Je tedy patrné, že nelze podrobněji nastavit topologii sítě.

Výchozí zdrojový kód také neposkytuje možnost vytvořit síť se skrytou vrstvou obsahující pouze jeden neuron. Zásahem do zdrojového kódu knihovny jsem tento nedostatek odstranil, aby bylo možné nastavit skryté vrstvě pouze jeden neuron. Toto řešení se následně ukázalo jako špatné, protože aplikace nepracovala správně s jedním neuronem a skripty způsobovaly chyby. Omezení knihovny pouze na dva a více neuronů ve skryté vrstvě tedy má své opodstatnění.

### 5.1.3 Přenosová funkce

Umělé neurony mohou mít různé přenosové funkce (viz kapitola 3.3). Knihovna však využívá pouze jeden typ a tím je logistická sigmoida. V případě potřeby dále výstupní hodnotu upravuje. Příkladem je síť fungující binárně, tedy binární vstupy a binární výstupy. Přenosová funkce typu logistická sigmoida nám na výstupu generuje reálné neceločíselné hodnoty. Abychom získali námi požadované binární hodnoty, knihovna reálné hodnoty binarizuje pomocí funkce nazvané *threshold*, což znamená, že reálnou hodnotu převede na hodnotu logická 0 nebo 1.

```
1      public static function threshold($floatValue) {  
2          return ($floatValue > 0.5) ? 1 : 0;  
3      }
```

Funkce *threshold* požaduje jako parametr reálné číslo, které binarizuje. Pokud je reálné číslo ostře větší než 0,5 funkce vrátí hodnotu 1. Pokud je menší nebo rovno 0,5 funkce vrátí hodnotu 0.

### 5.1.4 Učící algoritmus

Každá síť vytvořena touto knihovnou je schopna se učit pouze algoritmem Backpropagation. Více o tomto učícím algoritmu v kapitole 4.2.1.1.

## 5.2 Ovládací prvky aplikace

Pro aplikaci jsem zvolil jednotnou úpravu ovládacích prvků, aby se ovládala co nejjednodušeji a bylo na první pohled zřejmé, co který ovládací prvek obsluhuje.

Každá sekce aplikace obsahuje vpravo nahoře panel s ovládacími tlačítky, která jsou podobná pro všechny sekce.



Obrázek 15: Ovládací panel

### 5.2.1 Tlačítko: Trénovat síť

Tímto tlačítkem uživatel zobrazí dialogové okno pro nastavení trénovací množiny sítě. V dialogovém okně, které může obsahovat záložky pro jednodušší manipulaci, lze také nastavovat různé parametry. Například spouštět logování vah jednotlivých neuronů při učení sítě a nastavit topologii sítě.



Obrázek 16: Dialogové okno - Trénovat síť

#### 5.2.1.1 Logování vah sítě

Logování vah umožňuje uživateli logovat váhy do souboru, který lze následně stáhnout v CSV formátu, tedy otevřít v programu Microsoft Excel, nebo pouze analyzovat tyto váhy přímo v aplikaci. Logování vah lze vypnout ručně pomocí zatržítka v dialogovém okně trénování sítě pod záložkou „Rozšířené“, nebo se vypne automaticky při předpokládané vysoké výpočetní náročnosti. Pokud lze v trénovací množině přidávat a odebírat počet elementů trénovací množiny, má tento počet také vliv na logování vah. Překročí-li tento počet určitou hranici, logování vah se automaticky vypne, bez možnosti ručního zapnutí, pokud nebude snížena počet elementů trénovací množiny.

	E	H0 N0 W0	H0 N0 W1	H0 N0 W2	H0 N1 W0	H0 N1 W1	H0 N1 W2	O N0 W0	O N0 W1	O N0 W2
1	1.5	0.64108	0.28679	1.44802	-0.35104	2.01357	-1.45222	-1.6673	0.46671	1.35888
2	1.5	0.6369	0.29804	1.44384	-0.34068	2.03931	-1.44186	-1.66491	0.43052	1.36391
3	1.5	0.63258	0.3092	1.43952	-0.33044	2.06555	-1.43162	-1.66246	0.39371	1.36911
4	0.5	0.63005	0.32073	1.45937	-0.31745	2.09204	-1.40077	-1.74458	0.33415	1.27117
5	1	0.62539	0.33171	1.45472	-0.30695	2.11918	-1.39027	-1.74107	0.29634	1.27767
6	1.5	0.6098	0.3138	1.44361	-0.32356	2.09264	-1.41559	-1.654	0.35271	1.37047
7	1.5	0.6067	0.30706	1.44547	-0.3109	2.09344	-1.41179	-1.6602	0.35382	1.3605
8	1.5	0.59103	0.30706	1.45198	-0.32747	2.09344	-1.41056	-1.64873	0.35072	1.3653
9	1.5	0.57544	0.30706	1.45843	-0.3437	2.09344	-1.40898	-1.63824	0.34725	1.36916
10	0.5	0.57108	0.31837	1.47602	-0.33356	2.12047	-1.38042	-1.71106	0.29074	1.28177

Obrázek 17: Obrázek tabulky logování vah z aplikace

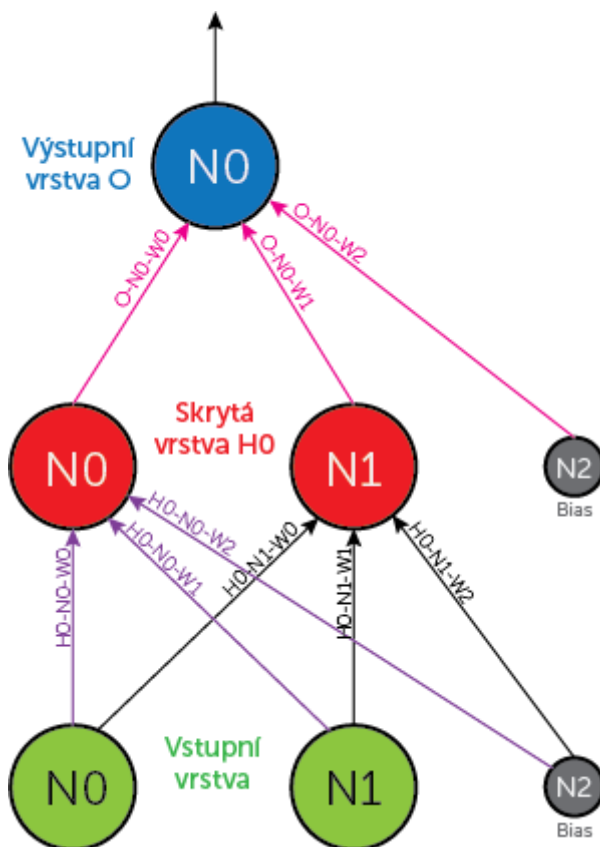
Záhlaví tabulky popisuje ve zkratkách, jaké hodnoty ve sloupci najdeme.

- První neoznačený sloupec obsahuje pouze číslování řádků, pro lepší orientaci.
- **E** je tzv. learning rate, v překladu učící koeficient, který je popsán v kapitole 5.1.1

Každý následující sloupec je popsán již třemi zkratkami pod sebou:

- **H0** – vychází z anglického *hidden layer*, což znamená skrytá vrstva a číslo popisující, o kterou skrytou vrstvu se jedná.
- **N0 – N1** – zkratka neuronu a číslo popisující pořadí neuronu v dané vrstvě
- **W0 – W2** – vychází z anglického *weight*, což znamená váha a číslo popisující o kterou váhu se jedná
- **O** – vychází z anglického *output*, což znamená výstup

Následující obrázek znázorňuje graficky, kterou váhu který sloupec vyjadřuje z tabulky na obrázku (Obrázek 17). Pro jednodušší orientaci jsem pole záhlaví tabulky vyznačil stejnými barvami, jako příslušné sady šipek na následujícím obrázku.



Obrázek 18: Schéma - znázornění označení vah v logovací tabulce

### 5.2.1.2 Topologie sítě

Topologii sítě lze nastavit pomocí posuvníků v dialogovém okně trénování sítě pod záložkou „Rozšířené“. Uživatel může změnit počet skrytých vrstev a také počet neuronů v jedné skryté vrstvě. Knihovna bohužel není dostatečně flexibilní a umožňuje nastavit pouze stejný počet neuronů ve všech skrytých vrstvách. Nikoliv však počet neuronů pro každou skrytou vrstvu zvlášť. Ve výchozím stavu knihovna poskytovala možnost nastavit tento počet neuronů na minimální hranici dvou neuronů. Zásahem do zdrojového kódu jsem umožnil vytvořit i síť s jedním neuronem ve skryté vrstvě, ale z důvodů nestabilního chování aplikace jsem tuto možnost nastavil opět do výchozího nastavení.

### 5.2.1.3 Index náročnosti

Učení sítě může být v některých případech výpočetně velmi náročný. Webové servery mají vždy nastaven maximální čas vykonávání jednoho skriptu, po němž vykonávání skriptu



server ukončí. Tento čas bývá ve většině případů nastaven na 30 sekund. Samotné učení může být v některých případech tak výpočetně náročné, že síť nebude vytrénována úplně, ale pouze z určité části. Pokud by k tomu ještě byla logována váha každého vstupu pro každý neuron, skript by mohl způsobit výjimku a učení by skončilo neúspěšně.

Při programování a testování jsem zjistil rizikové situace, díky kterým aplikace způsobovala výjimky kvůli výpočetní náročnosti. Těmito situacím jsem se snažil zabránit. Zavedl jsem takzvaný index náročnosti, který se přepočítává při každé změně nastavení sítě JavaScriptem ještě před potvrzením všech nastavení a odesláním formuláře na server. Index náročnosti mění barvu podle rizikovosti nastavení od zelené, přes oranžovou až po červenou, při které již nelze síť učit.

Na hodnotu indexu náročnosti má vliv nastavení topologie sítě. Jeho hodnota je rovna součinu počtu neuronů ve skryté vrstvě a počtu skrytých vrstev. Pokud je hodnota indexu náročnosti kritická, tedy index má červenou barvu, je automaticky vypnuto logování vah bez možnosti ručního zapnutí, pokud nebude přenastavena topologie sítě tak, aby se hodnota indexu náročnosti nepohybovala za kritickou hranicí.

### **5.2.2 Tlačítko: Log vah**

Chce-li uživatel po naučení sítě analyzovat váhy a jejich změny v průběhu učícího algoritmu, může si zobrazit log vah tímto tlačítkem. Otevře se dialogové okno, které obsahuje výpis přímo z CSV souboru se záznamem vah. Pokud bylo učení výpočetně náročnější, může zobrazení trvat delší dobu. V těchto případech je lepší si CSV soubor raději stáhnout a prohlédnout například v softwaru Microsoft Excel.

### **5.2.3 Tlačítko: Detail sítě**

Tímto tlačítkem otevřeme dialogové okno obsahující tabulku s podrobným výpisem konfigurace sítě. Tento výpis je zobrazen metodou pro výpis podrobných informací o síti implementovanou v samotné knihovně.

#### 5.2.4 Tlačítko: Reset

Tlačítkem „Reset“ odstraníme aktuálně naučenou síť, nebo také odstraníme změny v učícím dialogu ještě před odesláním celého formuláře.

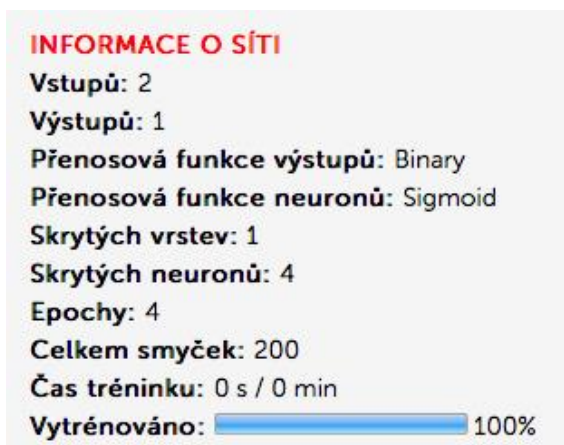
### 5.3 Naučená neuronová síť

Po ukončení učícího algoritmu aplikace vypíše spoustu informací o dané síti. Nyní popíšu jednotlivé části, které můžeme na obrazovce vidět.

#### 5.3.1 Informace o síti

Tato část nám vypisuje nejdůležitější informace o aktuální síti.

- Vstupů – počet vstupů
- Výstupů – počet výstupů
- Přenosová funkce výstupů – typ výstupních hodnot
- Přenosová funkce neuronů - typ přenosové funkce jednotlivých neuronů
- Skrytých vrstev – počet skrytých vrstev
- Skrytých neuronů – počet skrytých neuronů v jedné skryté vrstvě
- Epochy – počet epoch
- Celkem smyček – celkový počet smyček při učení
- Čas tréninku – doba, kterou trval učící proces na serveru
- Vytrenováno – procentuální vyjádření vytrenovanosti



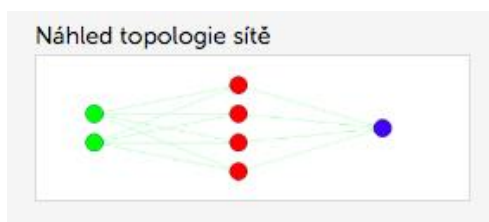
Obrázek 19: Informace o vytrenované síti

### 5.3.2 Náhled topologie sítě

Obrázek, který má uživatel k dispozici v náhledu, je vygenerován grafickou knihovnou přesně podle topologie sítě. Pro každou trénovanou síť je vygenerován originál pro její topologii. Pro zvětšený náhled po kliknutí na miniaturu je použit plug-in pro JavaScriptovou knihovnu jQuery. Plug-in se nazývá Lightbox a je určen pro efektní zobrazování obrázků a fotografií. Lightbox je open source projekt (projekt s otevřeným zdrojovým kódem), tzn. je volně ke stažení [16].

Popis náhledu topologie:

- Zelená – vstupy tvořící vstupní vektor
- Červená – jednotlivé neurony skryté vrstvy
- Modrá – výstupní neurony



Obrázek 20: Náhled topologie sítě

### 5.3.3 Tester sítě

Tester sítě slouží pro zadávání vstupů do vytrénované sítě a sledování odezvy na tyto vstupy. Zadané vstupní hodnoty potvrdíme tlačítkem „Použít“. Tester funguje na základě technologie AJAX, kdy jsou data na server odesílána na pozadí aplikace, a stejně tak přijímána odezva zpět ze serveru, a tím pádem není nutno načítat a vykreslovat kompletně celou stránku znovu.

Obrázek 21: Tester sítě

### 5.3.4 Trénovací množina

Tabulka s přehledem zadané trénovací množiny. Slouží pro porovnání s výstupy testeru sítě. Uživatel tak může vidět výstup ze sítě a porovnat s touto tabulkou a tím i ověřit správnost výstupních hodnot proti trénovací množině, kterou zadal.

Vstup A	Vstup B	Výstup Y
0	0	0
0	1	1
1	0	1
1	1	0

Obrázek 22: Trénovací množina

### 5.3.5 Stáhnout síť

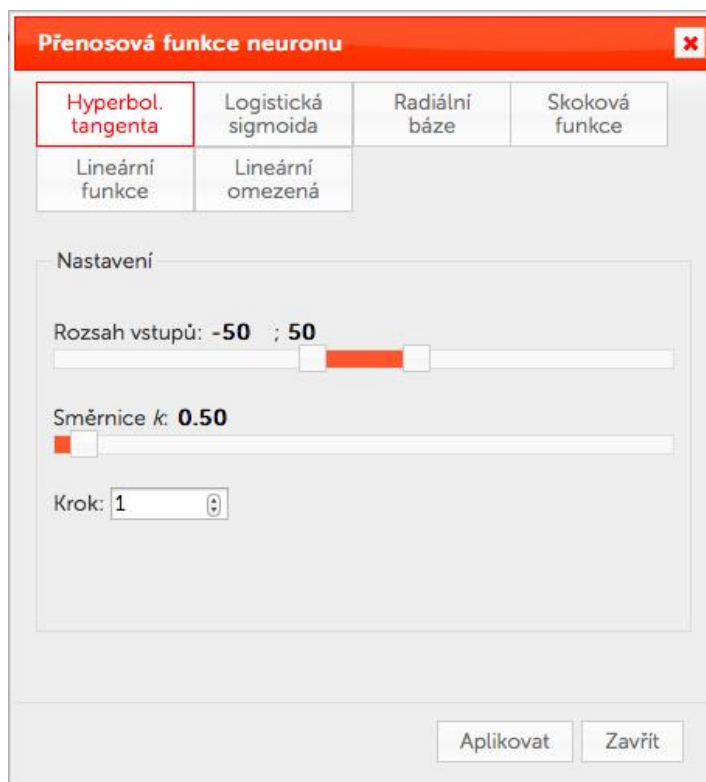
Kliknutím na tento odkaz si může uživatel stáhnout ZIP soubor se všemi potřebnými soubory k rekonstrukci sítě. Není tedy potřeba si pamatovat nastavení jednotlivých parametrů neuronové sítě, protože ji lze pomocí těchto souborů rekonstruovat v sekci „Síť ze souborů“.

[Stáhnout síť](#) (66.2 kB zip soubor)

Obrázek 23: Odkaz stažení ZIP souboru k síti

## 5.4 Sekce: Neuron

V této sekci uživatel nalezne simulaci aktivačních funkcí neuronů. Zde zatím není využita knihovna pro umělé neuronové sítě. Pouze si zde uživatel může nasimulovat jednotlivé typy aktivačních funkcí a analyzovat jejich změny na základě změn různých parametrů, jako je například směrnice.



Obrázek 24: Neuron - dialog pro nastavení neuronu

Nastavení přenosové funkce neuronu provádíme v dialogu zobrazeném na obrázku (Obrázek 24). Podle zvolené přenosové funkce se nám zobrazují volitelné parametry pro tu danou funkci.

- Rozsah vstupů – udává interval, ve kterém se pohybují vstupní hodnoty do neuronu.
- Směrnice – udává hodnotu směrnice  $k$ .
- Krok – hodnota kroku při získávání funkčních hodnot přenosové funkce (při nastavení hodnoty na 0,1 aplikace vytvoří množinu od spodní hranice intervalu po kroku 0,1 – například pokud je dolní hranice intervalu -50, potom bude množina vstupních hodnot začínat hodnotami -50; -49,9; -49,8; -49,7 atd. až po horní hranici

intervalu. Postupným dosazením do přenosové funkce neuronu získáme data potřebná pro vykreslení grafu přenosové funkce).

- **Mez** – udává mezní hodnotu u některých typů přenosových funkcí (u skokové přenosové funkce udává mez na ose vstupních hodnot, čili hodnotu vstupu, kde se binární výstup překlápí z 0 na 1 a u lineární omezené funkce udává mez na ose funkčních hodnot).



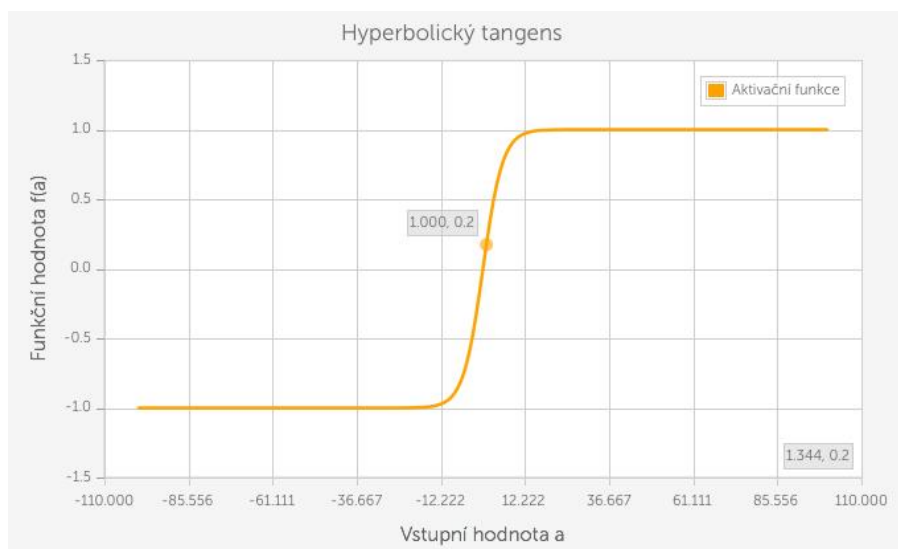
**Přenosová funkce**  
**Typ přenosové funkce:** Hyperbolický tangens  
**Interval vstupů:** [-100; 100]  
**Krok:** 1  
**Směrnice k:** 0.35

Obrázek 25: Informace o aktuální přenosové funkci

Po potvrzení nastavení tlačítkem „*Aplikovat*“ se nám vykreslí graf a vypíše informace o dané přenosové funkci.

Pro vykreslení grafu je použit plug-in pro JavaScriptovou knihovnu jQuery. Plug-in má název jqPlot a je k dispozici volně ke stažení [15] – open source projekt (projekt s otevřeným zdrojovým kódem).

Vykreslený graf je interaktivní. Pokud kurzorem najedeme na plochu grafu, vpravo dole se nám zobrazují aktuální souřadnice kurzoru. Chceme-li odečíst souřadnice konkrétního bodu grafu, umístíme kurzor na linku grafu. Najedeme-li kurzorem na vykreslený bod (tj. bod, který obsahuje množina vypočtených funkčních hodnot), tento bod se nám zvětší a vedle něj se zobrazí rámeček se souřadnicemi. Přibližné souřadnice ostatních nevykreslených bodů lze odečítat pomocí zobrazovaných souřadnic kurzoru vpravo dole.



Obrázek 26: Graf přenosové funkce vykreslený aplikací

Tabulku vykreslených bodů nalezneme pod tlačítkem „*Funkční hodnoty*“ v ovládacím panelu vpravo nahoře. Tato tabulka slouží jako zdroj hodnot pro vykreslení grafu.

$x$	$f(x)$
-50.00	-1.000
-49.00	-1.000
-48.00	-1.000
-47.00	-1.000
-46.00	-1.000
-45.00	-1.000
-44.00	-1.000
-43.00	-1.000
-42.00	-1.000
-41.00	-1.000

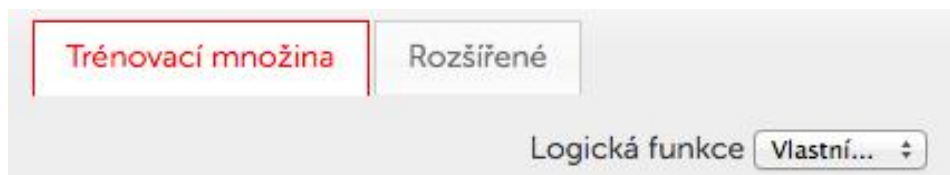
Obrázek 27: Tabulka vstupních a funkčních hodnot přenosové funkce

## 5.5 Sekce: Logické funkce

Zde si může uživatel vytvořit síť s binárními vstupy a binárními výstupy. Síť bude po ukončení učícího algoritmu schopna pracovat jako logický prvek.

Při nastavení trénovací množiny máme možnost vybrat jednu z předdefinovaných logických funkcí, na kterou bude síť vytrénována. Trénovací množina se nám při výběru předdefinované logické funkce předvyplní a uzamkne se možnost editace. Pokud chceme

vytvořit vlastní trénovací množinu, zvolíme v roletkovém menu možnost „*Vlastní...*“. Jednotlivé kombinace binárních vstupů jsou již vyplněny pro usnadnění ovládání.



Obrázek 28: Logická funkce - trénovací množina

Trénovací množina může obsahovat pouze binární hodnoty 0 a 1, ale síť bude po vytrénování schopna reagovat na reálná čísla ve vstupním vektoru.

Více o ovládání aplikace v kapitole 5.2.

## 5.6 Sekce: Detekce jazyka sítí

Tato síť je schopna rozpoznat, v jakém jazyce je vstupní slovo nebo fráze. Je to typický příklad na klasifikaci neuronovou sítí. Při trénování musíme zadat slovo nebo frázi a přiřadit jí jazyk. Jednotlivé jazyky tvoří třídy klasifikační množiny. Slovo nebo fráze je vstupem a jazyk výstupem sítě.

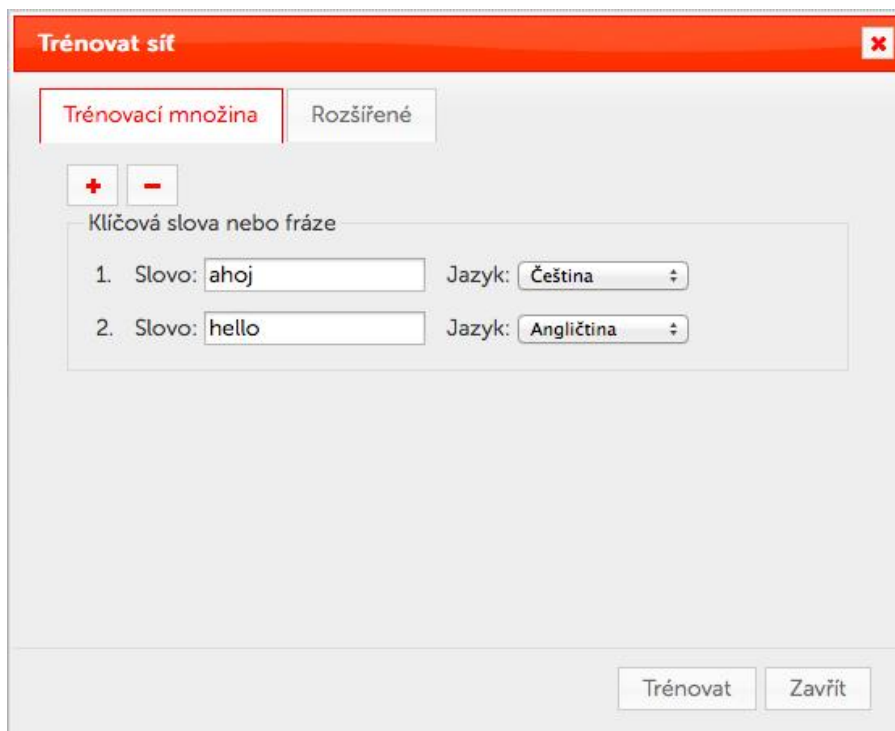
Vstup zadaný jako řetězec znaků je přepočítán na číselné hodnoty podle ASCII kódů jednotlivých znaků, a použit jako prvek vstupního vektoru sítě. Výstup sítě je binární. Máme-li například dva jazyky, tak jazyk první má binární hodnotu 10 a jazyk druhý má hodnotu 01.

V případě tří jazyků:

1. Jazyk – dvojková hodnota 100
2. Jazyk – dvojková hodnota 010
3. Jazyk – dvojková hodnota 001

Pro pohodlnější a snazší odečtení výstupu je v knihovně implementována metoda, která již při učení sítě těmto binárním hodnotám přiřadí námi definovaný slovní název, který jsme zadali do trénovací množiny (např. „angličtina“, „čeština“ atd.).





Obrázek 29: Dialog pro trénování detekční sítě

V dialogu pro zadání trénovací množiny můžeme volit i počet vstupních slov nebo frází pomocí tlačítek „+“ a „-“. Čím podobnější slova nebo fráze pro různé jazyky zadáme, tím je proces učení výpočetně náročnější (např. pozdrav u němčiny a angličtiny – hello a hallo).

Více o ovládání aplikace v kapitole 5.2.

## 5.7 Sekce: Predikce sítí

Jedná se o síť schopnou předpovídat určitou veličinu podle jiných zadaných veličin. Předpovídání probíhá na základě naměřených minulých dat, která se použijí jako jednotlivé elementy do trénovací množiny. Tato síť simuluje předpověď návštěvnosti horského střediska podle zadané teploty a výšky sněhové pokrývky.

Predikční síť je velmi náchylná na pořadí hodnot v trénovací množině a jejich vzájemné vychýlení. Nelze si vymyslet náhodně jakékoliv hodnoty a očekávat od sítě rozumné předpovědi. U této konkrétní sítě pro předpověď návštěvnosti horského střediska je potřeba mít na paměti, že teplota i výška sněhové pokrývky se v čase vyvíjí. Nelze tedy například

zadat po sobě jdoucí hodnoty sněhové pokrývky 150 cm, 0 cm, 300 cm, 75 cm, jelikož tento vývoj výšky sněhové pokrývky není reálný. Z toho také vyplývá, že je nutno data měřit a uchovávat pravidelně v dostatečně blízkých časových úsecích, abychom se vyhnuli výše uvedené posloupnosti hodnot. Čím více dat o systému máme, tím přesnější předpovědi můžeme očekávat.

Obrázek 30: Dialog pro trénování predikční sítě

Každá hodnota trénovací množiny musí být z intervalu, který je uveden vždy nad sloupcem v závorce (Obrázek 30). Počet řádků je možno měnit tlačítky „+“ a „-“.

Více o ovládání aplikace v kapitole 5.2.

## 5.8 Sekce: Načtení sítě ze souborů

Chceme-li rekonstruovat některou ze sítí, kterou jsme již měli vytvořenou, můžeme zde načíst soubory, které jsou u každé sítě k dispozici ke stažení v ZIP souboru. Každý typ sítě má různý počet a typ souborů potřebný pro inicializaci sítě ze souborů. Je možno nahrávat pouze soubory ve formátu \*.dat.

Obrázek 31: Rekonstrukce sítě pomocí stažených souborů

## 5.9 Správa souborů

Výhodou i nevýhodou knihovny je, že si potřebuje pro své vnitřní procesy uchovávat data v souborech, které si sama vytváří. Tyto soubory jsou ve velké většině případů malé (řádově desítky kB), ale například CSV soubor pro logování vah může dosahovat velikosti i několik MB, pokud je učení náročné a trvá delší dobu. Bylo tedy nutné vyřešit efektivním způsobem správu těchto souborů, aby zbytečně nebyly uchovávány již nepotřebné soubory. K jedné síti jsou ukládány tři typy souborů:

- DAT – soubory s konfiguracemi sítě
- CSV – log vah
- PNG – náhled topologie sítě
- ZIP – komprimovaný balíček všech souborů k síti

### 5.9.1 Ukládání souborů

Pro ukládání souborů jsou vyčleněny dvě složky. První složka je na soubory s daty k síti, která obsahuje další podsložky rozdělené podle účelu sítě. Druhá složka je pro ukládání náhledu topologie sítě, čili pro obrázky typu PNG.

Samotné ukládání těchto souborů probíhá při učení sítě. Vždy před začátkem trénovacího algoritmu je vygenerován unikátní identifikátor (UID), používaný pro tento konkrétní zpracovávající skript. Toto UID se přiřadí za jméno každého ukládaného souboru. UID tedy spojuje tyto soubory k sobě a vytváří tak skupinu souborů jedné sítě. Tento identifikátor je také použit u ZIP souboru, který obsahuje sadu všech potřebných souborů k síti.

### 5.9.2 Kontrola a mazání souborů

Pro manipulaci s již existujícími soubory jsem vytvořil novou třídu nazvanou `FileHandler`, která obstarává kontrolu a mazání souborů.

Při zakládání objektu třídy `FileHandler` konstruktor požaduje jeden parametr a tím je počet hodin, po kterém je soubor smazán. Je tedy možno tento parametr měnit při vytváření objektu třídy. V aplikaci je nastaven na hodnotu 24 hodin.

V této třídě jsou implementovány dvě metody:

```
1      private function getCriticalDate() {...}
2      public function deleteOldFiles($path) {...}
```

Metoda na řádce č. 1 je volána přímo v konstruktoru třídy. Její návratovou hodnotou je datum. Soubory starší než toto datum jsou metodou na řádce č. 2 smazány. Metoda pro mazání souborů požaduje jako parametr cestu ke složce, kde jsou uloženy soubory. Projde i všechny podsložky a porovná datum vytvoření každého souboru s datem z metody na řádce č. 1. V případě, že je soubor starší než toto datum, je smazán.

## 5.10 Bezpečnost aplikace

Z důvodů možnosti nahrávat vlastní soubory na server, jsem musel aplikaci ošetřit proti případným útokům. Soubory vkládané na server mohou mít maximální velikost 100 kB. Dalším ošetřením je kontrola přípony jednotlivých vkládaných souborů. Přípustná je pouze přípona „*dat*“, jelikož je to jediný formát potřebný ke správnému načtení sítě.

Pro bezpečný chod aplikace jsem také ošetřoval uživatelsky zadávané vstupy. Pokud aplikace ze vstupního pole očekává číslo, tak si také ověří, že jde skutečně o číslo. Pokud ne upozorní uživatele na nekorektní vstup. Jde-li o vstup typu řetězec, aplikuji na něj funkci `htmlspecialchars()`, která v řetězci nahradí všechny rizikové znaky (<, >, „, &, \$, %, atp.) HTML entitami.

## ZÁVĚR

V době, kdy autor začal vypracovávat tuto bakalářskou práci, o umělých neuronových sítích měl jenom základní povědomí, protože tato problematika je probírána až ve 4. ročníku. Autor se věnuje již pár let webovým technologiím, a proto zvolil spojení webových technologií a neuronových sítí.

Na počátku, když autor hledal knihovnu, která by byla vyhovující pro tuto aplikaci, očekával, že existuje pro PHP knihovna, díky které bude možné vytvořit jakýkoliv typ sítě s jakoukoliv topologií. Bohužel díky tomu, že webová aplikace pracuje systémem klient-server, a každý požadavek na server má předem vymezený čas zpracování, není možné některé rozsáhlejší neuronové sítě vůbec trénovat, jelikož čas vymezený serverem ke zpracování není dostačující, a proto některé principy byly tvůrci knihovny zjednodušeny, aby byl ušetřen výpočetní čas. Nicméně na ukázkou základních příkladů sítí je tato knihovna více než postačující.

Knihovna disponuje metodami, jako je vykreslení topologie sítě jako obrázku ve formátu PNG, a také detailní výpis informací o síti spolu s nastavením serveru týkajícím se času vykonávání jednoho skriptu. Tyto metody jsou velmi užitečné, protože díky nim uživatel přesně vidí konfiguraci sítě a také může sledovat, jak je která síť výpočetně náročná.

Knihovna má dvě velká negativa. Tím prvním je využití pouze jednoho učícího algoritmu. Využívá pouze algoritmus Backpropagation, který je ale na druhou stranu nejvíce používaným učícím algoritmem, takže pro účely této práce postačující. Druhým je možnost nastavit počet neuronů ve skryté vrstvě pouze pro všechny skryté vrstvy současně, takže není možno určit počet neuronů pro každou skrytou vrstvu zvlášť. Všechny skryté vrstvy tedy musí mít stejný počet neuronů, což je omezující. Dalším negativem se zdála být možnost nastavit pouze dva a více neuronů do skryté vrstvy. Proto autor provedl drobné úpravy ve zdrojovém kódu samotné knihovny, aby byla možnost nastavit i jeden neuron. Po testování jednoho neuronu musel tuto úpravu změnit, protože aplikace způsobovala výjimky a nebyla stabilní.

Díky použité JavaScriptové knihovně jQuery, využití jejích plug-inů, a použití technologie AJAX, je výsledná webová aplikace interaktivní a uživatelsky přívětivá.

## ZÁVĚR V ANGLIČTINĚ

While the author has started to develop this bachelor thesis on artificial neural networks only had a basic understanding, because this issue is discussed only in the 4th grade. The author has devoted a few years Web technologies, and therefore opted for the connection of web technology and neural networks.

At the beginning, when the author was looking for a library that would be suitable for this application, expecting that exists for PHP library that makes possible to create any type of network with any topology. Unfortunately, due to the fact that the Web application works client-server system, and each server has a requirement for a pre-defined processing time can not be any larger neural networks to train at all, because the time allowed by the server for processing is not sufficient, and therefore some of the principles were the creators of the library simplified to spare computing time. However, the demonstration of the basic examples of networks is that the library is more than sufficient.

The library provides methods such as network topology rendering as a PNG image, and also a detailed list of information about the network, along with server settings relating to the execution time of a script. These methods are very useful because they can see exactly user network configuration and also can see how that network is computationally intensive.

The library has two big negatives. The first is the use of only one learning algorithm. Backpropagation algorithm uses only that but most used the learning algorithm, so the purpose of this work is sufficient. The second is the ability to set the number of neurons in the hidden layer for all hidden layers simultaneously, so it is not possible to determine the number of neurons for each hidden layer separately. All hidden layer must have the same number of neurons, which is restrictive. Another downside seemed to be the only possibility to set two or more neurons in the hidden layer. Therefore, the authors introduced minor modifications in the source code library to be able to set even one neuron. After testing a neuron had to change this adjustment because the application causing the exception and not stable.

Thanks to the jQuery JavaScript library, making use of the plug-ins and use of AJAX technology, the resulting web application is interactive and user friendly.

## SEZNAM POUŽITÉ LITERATURY

- [1] ZELINKA, Ivan. *Umělá inteligence I: Neuronové sítě a genetické algoritmy*. 1. vyd. Brno: VUT v Brně, 1998, 126 s. ISBN 80-214-1163-5.
- [2] ŠÍMA, Jiří. *Teoretické otázky neuronových sítí*. Vyd. 1. Praha: MATFYZ press, 1996, 390 s. ISBN 80-858-6318-9.
- [3] ANN - Artificial Neural Network for PHP 5: Neural Networks [online]. [cit. 2014-05-23]. Dostupné z: [http://ann.thwien.de/index.php/Neural\\_Networks](http://ann.thwien.de/index.php/Neural_Networks)
- [4] OPLATKOVÁ, Zuzana. *Neuronové sítě: Úvod do umělé inteligence - 1*. Zlín, 2012.
- [5] Neuronová síť. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2007, 2013-3-12 [cit. 2014-05-23]. Dostupné z: [http://cs.wikipedia.org/wiki/Neuronov%C3%A1\\_s%C3%AD%C5%A5](http://cs.wikipedia.org/wiki/Neuronov%C3%A1_s%C3%AD%C5%A5)
- [6] VONDRÁK, Ivo. *Umělá inteligence a neuronové sítě*. Ostrava: VŠB, 1995, 139 s. ISBN 80-707-8259-5.
- [7] VRÁNA, Jakub. *1001 tipů a triků pro PHP*. Vyd. 1. Brno: Computer Press, 2010, 456 s. ISBN 978-80-251-2940-1.
- [8] GUTMANS, Andi, Stig Sæther BAKKEN a Derick RETHANS. *Mistrovství v PHP 5*. Vyd. 1. Překlad Bogdan Kiszka. Brno: CP Books, 2005, 655 s. ISBN 80-251-0799-X.
- [9] *JQuery: kuchařka programátora*. Vyd. 1. Brno: Computer Press, 2010, 436 s. ISBN 978-80-251-3152-7.
- [10] MARGORÍN, Marián. *JQuery bez předchozích znalostí*. Vyd. 1. Brno: Computer Press, 2011, 253 s. ISBN 978-80-251-3379-8.
- [11] CROFT, Jeff, Ian LLOYD a Dan RUBIN. *Mistrovství v CSS: pokročilé techniky pro webové designéry a vývojáře*. Vyd. 1. Překlad Josef Bábík. Brno: Computer Press, 2007, 409 s. ISBN 978-80-251-1705-7.
- [12] CASTRO, Elizabeth a Bruce HYSLOP. *HTML5 a CSS3: názorný průvodce tvorbou WWW stránek*. 1. vyd. Brno: Computer Press, 2012, 439 s. ISBN 978-80-251-3733-8.
- [13] HOGAN, Brian P. *HTML5 a CSS3: výukový kurz webového vývojáře*. Vyd. 1. Brno: Computer Press, 2011, 272 s. ISBN 978-80-251-3576-1.

- [14] KEOGH, James Edward a Mario GIANNINI. *OOP bez předchozích znalostí: průvodce pro samouky*. Vyd. 1. Brno: Computer Press, 2006, 222 s. ISBN 80-251-0973-9.
- [15] *JqPlot: JqPlot Charts and Graphs for jQuery* [online]. 2009, 27.3.2013 [cit. 2014-05-29]. Dostupné z: <http://www.jqplot.com/>
- [16] *Lightbox: Lightbox 2* [online]. [cit. 2014-05-29]. Dostupné z: <http://www.lokeshdhakar.com/projects/lightbox2/>
- [17] ANN for PHP5. *ANN- Artificial Neural Network for PHP5: Download* [online]. 2002, 13.12.2012 [cit. 2014-05-30]. Dostupné z: <http://ann.thwien.de/index.php/Download>
- [18] Open SOurce. *Open Source Initiative: The BSD 2-Clause License* [online]. 1998, 9.4.2014 [cit. 2014-05-30]. Dostupné z: <http://opensource.org/licenses/BSD-2-Clause>
- [19] *Umělé neuronové sítě v PHP* [online]. 2014, 6.6.2014 [cit. 2014-06-03]. Dostupné z: <http://ann.netprog.cz>



**SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK**

$f(a)$	funkční hodnota aktivační funkce pro vstupní hodnotu $x$
$x_i$	vstupní hodnota $i$ do neuronu
$k$	Směrnice
$w_i$	váha pro vstupní hodnotu $x_i$
$e$	Eulerovo číslo ( $\approx 2,71828$ )
$a$	vnitřní potenciál neuronu
$\forall x > 0: f(a) = a$	pro všechna $x$ větší než 0 platí, že $f(x)$ se rovná $x$
$\wedge$	a současně platí
$\forall a \leq 0: f(a) = 0$	pro všechna $x$ menší nebo rovny 0 platí, že $f(x)$ se rovná 0
$\forall a \in \langle -h_1, h_1 \rangle$	pro všechna $x$ patřící do intervalu od $-h_1$ do $h_1$ platí
$E$	Learning rate – učící koeficient
$H0-Hn$	skrytá vrstva 0 až $n$
$N0-Nn$	neuron 0 až $n$
$W0-Wn$	váha vstupu 0 až $n$

**SEZNAM OBRÁZKŮ**

Obrázek 1: Biologický neuron [2] .....	13
Obrázek 2: Synapse .....	14
Obrázek 3: Umělý (formální) neuron [2].....	15
Obrázek 4: Přenosová funkce - logistická sigmoida.....	17
Obrázek 5: Přenosová funkce - hyperbolický tangens.....	17
Obrázek 6: Přenosová funkce – perceptron .....	18
Obrázek 7: Přenosová funkce - lineární funkce.....	18
Obrázek 8: Přenosová funkce - lineární omezená funkce.....	19
Obrázek 9: Přenosová funkce - binární funkce.....	20
Obrázek 10: Přenosová funkce - radiální báze .....	20
Obrázek 11: Schéma perceptronu [1] .....	23
Obrázek 12: Vícevrstvá neuronová síť .....	24
Obrázek 13: Hopfieldova síť [6].....	25
Obrázek 14: Kohonenova mapa [6] .....	25
Obrázek 15: Ovládací panel.....	29
Obrázek 16: Dialogové okno - Trénovat síť .....	30
Obrázek 17: Obrázek tabulky logování vah z aplikace .....	31
Obrázek 18: Schéma - znázornění označení vah v logovací tabulce.....	32
Obrázek 19: Informace o vytrénované síti.....	34
Obrázek 20: Náhled topologie sítě.....	35
Obrázek 21: Tester sítě .....	36
Obrázek 22: Trénovací množina.....	36
Obrázek 23: Odkaz stažení ZIP souboru k síti .....	36
Obrázek 24: Neuron - dialog pro nastavení neuronu.....	37
Obrázek 25: Informace o aktuální přenosové funkci.....	38
Obrázek 26: Graf přenosové funkce vykreslený aplikací.....	39
Obrázek 27: Tabulka vstupních a funkčních hodnot přenosové funkce.....	39
Obrázek 28: Logická funkce - trénovací množina.....	40
Obrázek 29: Dialog pro trénování detekční sítě .....	41
Obrázek 30: Dialog pro trénování predikční sítě.....	42
Obrázek 31: Rekonstrukce sítě pomocí stažených souborů.....	43

**SEZNAM TABULEK**

Tabulka 1: Rozdíly mezi neuronovou sítí a počítačem.....	12
--	----