

Vizuální vývojové prostředí pro neuronové sítě a jejich aplikace

Visual Development Environment for Neural Networks, and its applications

Bc. Martin Zelinka

Diplomová práce
2011

 Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2010/2011

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Martin ZELINKA**
Osobní číslo: **A08480**
Studijní program: **N 3902 Inženýrská informatika**
Studijní obor: **Informační technologie**

Téma práce: **Vizuální vývojové prostředí pro neuronové sítě
a jejich aplikace**

Zásady pro vypracování:

1. Popište problematiku neuronových sítí.
2. Analyzujte možnosti vizuálního programování neuronových sítí a jejich aplikací.
3. Navrhněte vlastní implementaci vizuálního vývojového prostředí.
4. Demostrujte ukázkovou aplikaci.

Rozsah diplomové práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. **AKAY, Metin. Handbook of neural engineering., 2007. 663 s. ISBN 978047005669.**
2. **ZELINKA, Ivan, et al. Evoluční výpočetní techniky : principy a aplikace. 1. vyd. Praha : BEN – technická literatura, 2008. 536 s. ISBN 80-7300-218-3.**
3. **MACKAY, David. Information theory, inference, and learning algorithms. Cambridge : Cambridge University Press, 2003. 628 s. ISBN 0521642981.**
4. **BERTHOLD, Michael, HAND, David. Intelligent data analysis. 2nd edition. Berlin Heidelberg : Springer, 2003. 514 s. ISBN 978-3-540-43.**
5. **AJITH, Abraham, CRINA, Grosan, WITOLD, Pedrycz. Engineering evolutionary intelligent systems. Berlin, Heidelberg : Springer, 2008. 444 s. ISBN 978-3-540-75.**
6. **ZELINKA, Ivan. Umělá inteligence aneb úvod do neuronových sítí a evolučních algoritmů. 1. vyd. Zlín : Univerzita Tomáše Bati, Fakulta technologická, 2005. 127 s. ISBN 8073182777.**
7. **Hakl, František, Holeňa, Martin: Úvod do teorie neuronových sítí. Vydavatelství ČVUT v Praze, Zikova 4, 163 00 Praha 6. 1998. ISBN 80-01-01716-8.**
8. **ZELINKA, I. Umělá inteligence v problémech globální optimalizace. Praha, BEN-technická literatura, 2002, 192. s. ISBN 80-7300-069-5.**

Vedoucí diplomové práce:

Ing. Erik Král

Ústav průmyslového inženýrství a informačních systémů

Datum zadání diplomové práce:

24. února 2011

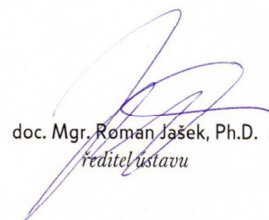
Termín odevzdání diplomové práce:

18. května 2011

Ve Zlíně dne 24. února 2011



prof. Ing. Vladimír Vašek, CSc.
děkan



doc. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

ABSTRAKT

Tato diplomová práce popisuje vytvoření vlastní aplikace pro vizuální programování neuronových sítí. V práci je shrnuta problematika neuronových sítí, popsány jejich principy, zejména vícevrstvé neuronové sítě s algoritmem backpropagation. Dále je shrnuta problematika adaptace neuronových sítí pomocí algoritmu backpropagation, adaptace pomocí evolučního algoritmu a je popsán evoluční algoritmus SOMA. V praktické části je popsána ukázková aplikace vizuálního vývojového prostředí pro neuronové sítě.

Klíčová slova:

Neuronová síť, Backpropagation, Evoluční algoritmy, Vizuální vývojové prostředí, SOMA

ABSTRACT

This master thesis describes how to create custom application for visual programming of neural networks. The paper summarizes the issue of neural networks, describes their principles, in particular, multilayer neural network with backpropagation algorithm. In thesis is also summarized adaptation of neural networks using the backpropagation algorithm, and adaptation using evolutionary algorithm. There is also described evolutionary algorithm SOMA. The practical part describes the proper visual application development environment for neural networks.

Keywords:

Neural network, Backpropagation, Evolutionary algorithms, Visual development environment, SOMA

Rád bych na tomto místě poděkoval vedoucímu diplomové práce Ing. Erikovi Královi, za jeho odborné vedení, cenné rady a připomínky, jež mi ochotně poskytoval při řešení této práce.

Děkuji také svým rodičům a rodině, za jejich podporu během celého mého studia.

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/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 diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/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 diplomovou/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 – diplomovou/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 diplomové/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 diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/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 diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....
podpis diplomanta

OBSAH

ÚVOD.....	9
I TEORETICKÁ ČÁST.....	10
1 ZÁKLADNÍ PRINCIPY A POJMY.....	11
1.1 HISTORIE NEURONOVÝCH SÍTÍ.....	11
1.2 LINEÁRNÍ A NELINEÁRNÍ SEPARABILITA TŘÍD.....	12
1.3 FORMÁLNÍ NEURON.....	13
1.4 PŘENOSOVÉ FUNKCE.....	15
1.5 OBECNÉ SCHÉMA NEURONOVÉ SÍTĚ.....	17
1.6 ADAPTACE NEURONOVÉ SÍTĚ.....	18
2 TYPY NEURONOVÝCH SÍTÍ.....	20
2.1 VÍCEVRSTVÉ NEURONOVÉ SÍTĚ S DOPŘEDNÝM ŠÍŘENÍM SIGNÁLU.....	20
2.1.1 Vícevrstvá neuronová síť s algoritmem backpropagation.....	20
2.2 REKURENTNÍ SÍTĚ.....	23
2.2.1 Hopfieldova síť.....	23
2.3 SÍTĚ SE SAMOORGANIZACÍ.....	24
2.3.1 Kohonenova síť.....	24
3 EVOLUČNÍ ALGORITMY.....	26
3.1 UČENÍ NEURONOVÉ SÍTĚ EVOLUČNÍM ALGORITMEM.....	27
3.2 PRINCIP EVOLUČNÍHO ALGORITMU SOMA.....	28
3.3 STRATEGIE ALLTOALL.....	28
3.4 PARAMETRY A TERMINOLOGIE.....	29
3.4.1 Specimen.....	29
3.4.2 PopSize.....	29
3.4.3 PathLength.....	29
3.4.4 Step.....	30
3.4.5 Prt.....	30
3.4.6 Závislost algoritmu SOMA na řídicích a ukončovacích parametrech.....	30
4 EXISTUJÍCÍ NÁSTROJE PRO PRÁCI S NEURONOVÝMI SÍTĚMI.....	32
4.1 STATSOFT STATISTICA - AUTOMATIZOVANÉ NEURONOVÉ SÍTĚ.....	32
II PRAKTICKÁ ČÁST.....	34
5 POPIS VLASTNÍ IMPLEMENTACE VÝVOJOVÉHO PROSTŘEDÍ PRO NEURONOVÉ SÍTĚ.....	35
5.1 NEURAL STUDIO 1.0.....	35
5.2 STRUKTURA PROGRAMU NEURAL STUDIO 1.0.....	36
5.3 ZÁKLADNÍ TŘÍDY METODY A PROMĚNNÉ.....	37
5.3.1 Třída DDForm.....	37
5.3.2 Třída block.....	37
5.3.3 Třída NeuralNetwork.....	38
5.3.4 Třída Soma.....	38
5.4 POPIS PRVKŮ HLAVNÍHO OKNA PROGRAMU NEURAL STUDIO 1.0.....	39
5.4.1 Ovládací prvky hlavního okna aplikace.....	39

5.4.2	Prvky hlavního menu	40
5.4.3	Panel nástrojů pro práci s návrhovým prostředím.....	41
5.4.4	Panel funkčních bloků.....	41
5.4.5	Panel záložek pro práci s projektem.....	42
5.4.6	Prohlížeč datových tabulek	43
5.4.7	Kontextové menu návrhového plátna.....	44
5.4.8	Kontextové menu bloku	44
5.5	POPIS DIALOGOVÉHO OKNA NEURONOVÉ SÍTĚ	45
5.5.1	Panel funkčních bloků.....	46
5.5.2	Záložky pro práci s neuronovou sítí.....	46
5.5.3	Panel vizualizace neuronové sítě	47
5.5.4	Panel testování neuronové sítě	47
5.6	POPIS FUNKČNÍCH BLOKŮ PROGRAMU NEURAL STUDIO 1.0.....	49
5.6.1	Source blok.....	49
5.6.2	Normalize blok.....	50
5.6.3	Delete columns blok.....	50
5.6.4	Alternate rows blok.....	51
5.6.5	Delete first blok.....	52
5.6.6	Delete last blok.....	52
5.6.7	Average rows blok	53
5.6.8	Split blok	53
5.6.9	Terminator blok.....	54
5.6.10	Neural network blok.....	54
5.6.11	Scope blok.....	57
5.6.12	Save to file blok	59
5.6.13	Mux blok	59
5.6.14	Demux blok.....	60
5.6.15	Input layer blok	60
5.6.16	Hidden layer blok.....	60
5.6.17	Output layer blok.....	61
5.6.18	Expected data blok	61
5.6.19	Evolution algorithm blok	61
5.7	POUŽITÍ APLIKACE NEURAL STUDIO 1.0	62
5.8	TESTOVÁNÍ PROGRAMU NEURAL STUDIO 1.0	64
5.8.1	Učení funkce exkluzivní disjunkce	64
5.8.2	Predikce hodnoty časové řady.....	67
5.9	NEURAL STUDIO 1.0 DEAMON	70
5.10	INSTALACE	71
	ZÁVĚR	72
	ZÁVĚR V ANGLIČTINĚ.....	73
	SEZNAM POUŽITÉ LITERATURY.....	74
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	76
	SEZNAM OBRÁZKŮ	78
	SEZNAM TABULEK.....	81
	SEZNAM PŘÍLOH.....	82

ÚVOD

Umělá neuronová síť je jedním z výpočetních systémů sestávající se z dílčích podsystémů tzv. neuronů. Původním cílem jejich vývoje byla snaha pochopit a modelovat procesy probíhající v jejich biologickém vzoru, tedy v nervovém systému. Neurofyziologické poznatky umožnily sestavit matematické modely, které lze úspěšně použít pro řadu praktických úloh umělé inteligence, jako je například generalizace, klasifikace, filtrace a predikce.

Základní stavební jednotkou umělé neuronové sítě je tzv. formální neuron, který můžeme definovat jako matematickou abstrakci neuronu biologického. Umělou neuronovou sítí pak rozumíme množinu těchto formálních neuronů vzájemně propojených dle určitých pravidel.

Samotné propojení těchto formálních neuronů je základním předpokladem pro realizaci neuronové sítě, nicméně pro určení funkce této sítě je zapotřebí správně upravit tzv. synaptické váhy těchto spojů. Synaptické váhy upravují hodnoty vstupů jednotlivých neuronů a tím ovlivňují chování celé sítě. Procesu úprav těchto vah pro realizaci požadovaného chování říkáme adaptace (učení) neuronové sítě.

Principiálně je neuronová síť systém s mnoha vstupy a mnoha výstupy. Přivedením signálu na její vstupy je vyvolána patřičná odezva na výstupech sítě. Hlavní výhodou neuronových sítí je fakt, že k danému účelu správně naučenou neuronovou sítí poté můžeme použít k řešení dané úlohy, či úloh podobných, tak že odezva na vstupní signály bude odpovídat požadovanému chování. Mezi další výhody pak patří například skutečnost, že neuronová síť je ze svého principu systém paralelní a dynamický (lze kdykoliv změnit její konfiguraci). Neuronové sítě mají ale i své nedostatky a to například výpočetní náročnost, zejména u rozsáhlých sítí.

Tato práce se zabývá především vícevrstevnými neuronovými sítěmi s dopředným šířením signálu a algoritmem učení backpropagation. Dále popisuje evoluční algoritmus Samo-Organizující se Migrační algoritmus (SOMA), který je použit v praktické části práce k učení neuronové sítě. Dále jsou analyzovány možnosti vybraných existujících řešení. Cílem této práce je vyvinout vizuální vývojové prostředí pro neuronové sítě a jejich aplikace, které bude poskytovat uživatelsky přívětivé prostředí pro návrh neuronové sítě a její následné použití.

I. TEORETICKÁ ČÁST

1 ZÁKLADNÍ PRINCIPY A POJMY

1.1 Historie neuronových sítí

Historie vzniku vývoje neuronových sítí sahá do roku 1943, kdy byla vydána práce Warrena McCullocha a jeho studenta Waltera Pittse, která se zabývala neurony a jejich modely. Autoři v práci publikovali model neuronu, který je prakticky používán dodnes [1], a ukázali, že neuronové sítě mohou řešit libovolnou aritmetickou či logickou funkci. Hodnoty parametrů v tomto modelu byly převážně bipolární (z množiny $\{-1,0,1\}$).

Ani přes velký zájem vědců o tuto disciplínu nepřinesla 40. a 50. léta žádný zásadní pokrok v této oblasti. Příkladem výzkumu z tohoto období může být například vývoj prvního neuropočítače Snark (1951). Významný milník vývoje neuronových sítí je rok 1957, kdy Frank Rosenblatt vynalezl tzv. perceptron, který je zobecněním McCullochova a Pittsova modelu neuronu pro reálný číselný obor parametrů. Pro tento model navrhl učící algoritmus, o kterém matematicky dokázal, že pro daná tréninková data nalezne po konečném počtu kroků daný váhový vektor parametrů (pokud existuje), nezávisle na jeho počátečním nastavení.

Přes nesporné úspěchy dosažené v oboru neuronových sítí, zájem o jejich vývoj upadl v roce 1969, kdy autoři Marvin Minsky a Seymour Papert uveřejnili knihu „perceptrons“, ve které vyzdvihovali význam známého faktu nemožnosti řešení nelineárně separabilních problémů (např. tzv. exkluzivní disjunkci XOR) jedním perceptronem. Dospěli k závěru, že daná omezení mohou být sice odstraněna konstrukcí vícevrstvého perceptronu, pro který je nutné nalézt parametry ručně a není možné nalézt algoritmus učení pro tuto strukturu. Jejich kritika zapříčinila útlum zájmu o tento obor až do roku 1983, kdy se agentura DARPA (Defense Advance Research Project Association) začala zajímat o vývoj v této oblasti a investovala do vývoje. V té době se zasloužil o obnovení zájmu o neuronové sítě uznávaný fyzik John Hopfield, který ukázal souvislost některých modelů neuronových sítí s fyzikálními modely magnetických materiálů.

Úspěch na poli výzkumu se dostavil v roce 1986, kdy se nezávisle na sobě podařilo Davidu Rumelhartovi a Yannu LeCunovi odvodit nový algoritmus učení vícevrstvých neuronových sítí, „backpropagation“, čímž vyřešili problém který se M. Minskému a S. Papertovi zdál v 60. letech jako neřešitelný. Jak se později ukázalo, algoritmus byl vlastně znovu objeven, protože byl již znám a publikován v tzv. tichém období (např. Arthur

Bryson a Yu-Chi ho, 1969). V 80. letech také vznikají nové typy sítí jako je Hopfieldova síť, kohonenova síť a Grossenbergova ART síť [2].

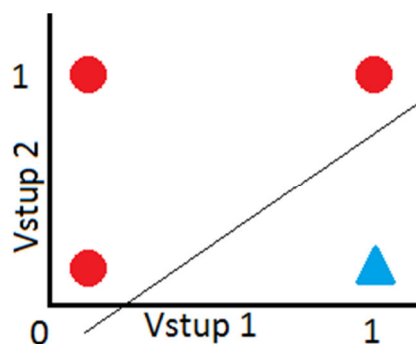
1.2 Lineární a nelineární separabilita tříd

Separabilita znamená zda, či jakou křivkou můžeme vytyčit hranici mezi jednotlivými třídami prvků tak, aby co nejméně příslušných objektů dané třídy bylo touto hranicí (hranicemi) přiřazeno do nesprávné třídy. Rozlišujeme lineárně a nelineárně separabilní třídy, v závislosti na tom, zda lze objekty tříd oddělit křivkou (křivkami) lineární, či nelineární.

Na obrázku (Obr. 1) je znázorněna třída popsaná tabulkou (Tab. 1). Tato třída je lineárně separabilní, protože objekty třídy lze od sebe oddělit lineární křivkou.

Tab. 1. Lineárně separabilní problém.

Vstup 1	Vstup 2	Výstup
0	0	1
1	0	0
0	1	1
1	1	1

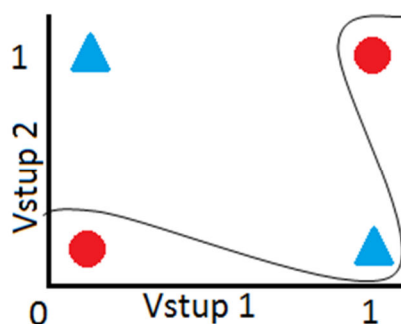


Obr. 1. Lineárně separabilní problém.

Obrázek (Obr. 2) znázorňuje třídu objektů (Problém exkluzivní disjunkce) popsanou tabulkou (Tab. 2). V tomto případě však není možné třídy jednoznačně oddělit lineární křivkou. Tato na první pohled „maličkost“ způsobila úpadek zájmu o neuronové sítě v 60. letech.

Tab. 2. Nelineárně separabilní problém (XOR).

Vstup 1	Vstup 2	Výstup
0	0	0
1	0	1
0	1	1
1	1	0



Obr. 2. Nelineárně separabilní problém.

1.3 Formální neuron

Formální (umělý, technologický) neuron je základní stavební jednotkou neuronové sítě a z teoretického hlediska na něj můžeme nahlížet jako na systém s mnoha vstupy a jedním výstupem (MISO). [3] Je to v podstatě jednoduchá výkonná jednotka, která vynásobí všechny vstupy jejich váhami. Tyto hodnoty posléze sečte a výslednou hodnotu dosadí do přenosové funkce neuronu. Výstup z přenosové funkce je i výstupem z neuronu [1].

Obecně se každý model neuronu principiálně skládá ze dvou částí a to obvodové funkce a funkce přenosové (aktivační).

Funkce obvodová udává, jakým způsobem budou vstupní parametry sítě kombinovány uvnitř neuronu. K nejpoužívanějším obvodovým funkcím patří tzv. vážená lineární kombinace vstupů:

$$u = \left(\sum_{i=1}^n w_i x_i + \theta \right) \quad (1)$$

Kde u je výstupní hodnota obvodové funkce, n je počet vstupů neuronu, x_i je i -tá vstupní hodnota vstupu, w_i je i -tá váha daného vstupu a θ je práh neuronu.

Přenosové funkce mohou mít různý tvar, obecně lze říci, že jsou nelineární, spojitě, nebo diskrétní. Výběr funkce závisí na řešené úloze a může mít zásadní vliv na kvalitu řešení [4]. Vybrané přenosové funkce jsou dále popsány v kapitole přenosové funkce v tabulce (Tab. 3) a (Tab. 4). Funkce ostrá nelinearita je například vhodná pro klasifikaci do dvou tříd, zatímco lineární funkce perceptron se používá pro lineární aproximaci. Nelineární funkce je vhodné použít pro modelování nějaké funkční závislosti.

K nejpoužívanějším přenosovým funkcím patří například tzv. standardní logistická sigmoida:

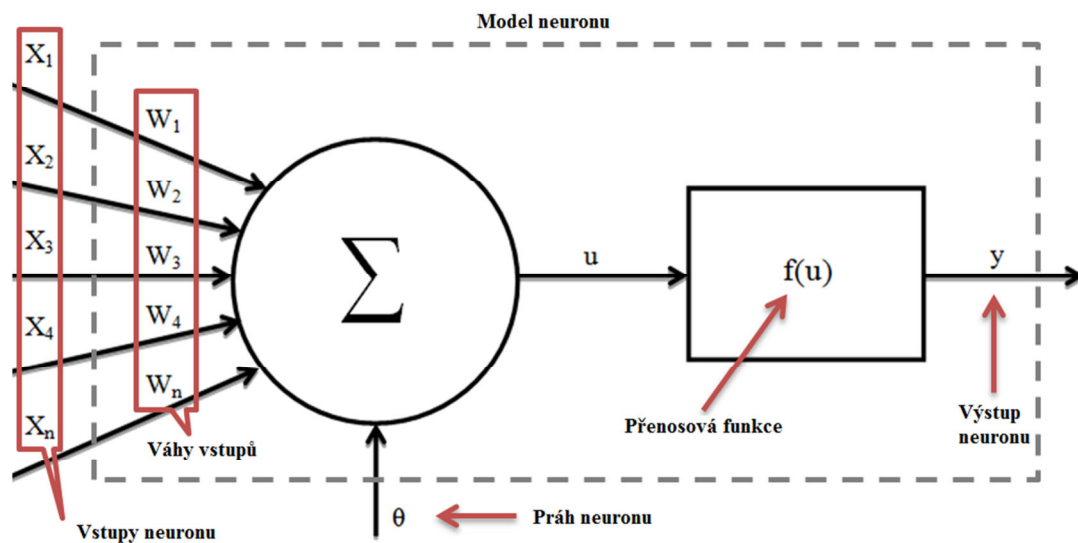
$$f(u) = \left(\frac{1}{1 + e^{-u}} \right) \quad (2)$$

Kde u je vstupní hodnota přenosové funkce a $f(u)$ udává výstupní hodnotu přenosové funkce.

Na obrázku (Obr. 3) je znázorněn jednoduchý model neuronu [5], který zpracovává vstupní údaje podle vztahu:

$$y = f \left(\sum_{i=1}^n w_i x_i + \theta \right) \quad (3)$$

Kde y je výstup neuronu, n je počet vstupů neuronu, x_i je i -tý vstup neuronu, w_i představuje hodnotu i -té váhy vstupu, θ je práh neuronu a f představuje přenosovou funkci neuronu.



Obr. 3. Model neuronu

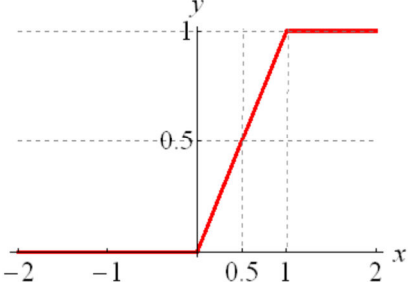
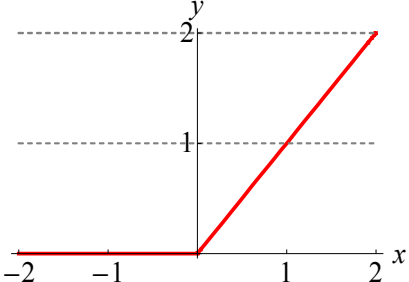
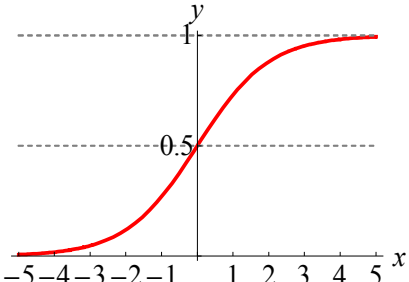
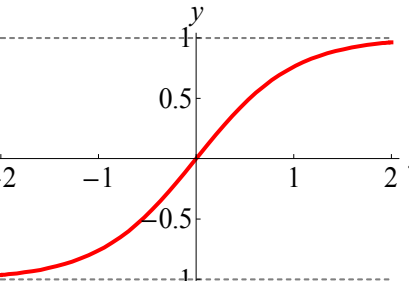
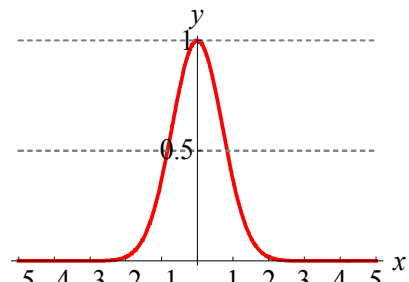
1.4 Přenosové funkce

Závislost stavu neuronu na jeho vstupním potenciálu je definovaná pomocí tzv. přenosové funkce. Tato funkce udává, jaká bude odezva výstupu na vstupní podnět. Volba této funkce je velice důležitá pro správný chod neuronu a celé neuronové sítě a závisí na charakteru řešeného problému. Každá přenosová funkce může mít nulový, či nenulový práh citlivosti. [1] [4] [3]

Tab. 3. Tabulka vybraných přenosových funkcí.

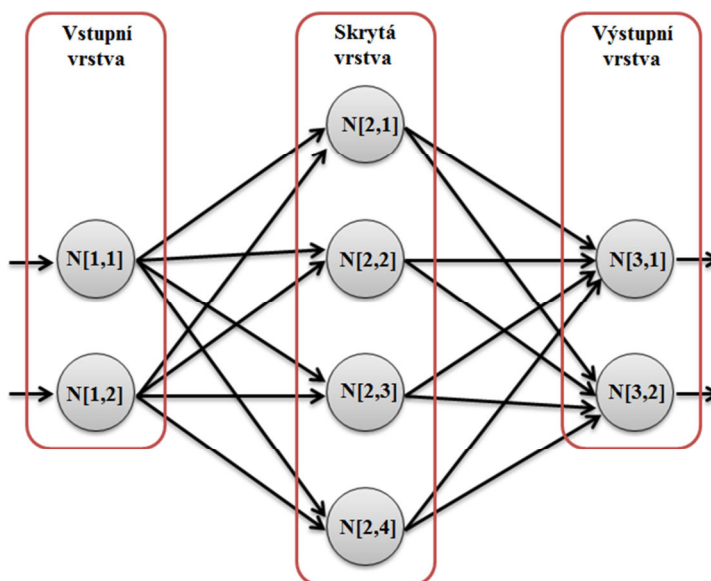
Název funkce	Popis	Graf
Ostrá nelinearita (Signum)	$f(x) = \text{Sign}(x)$	
Ostrá nelinearita	$f(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$	

Tab. 4. Tabulka vybraných přenosových funkcí 1.

Saturovaná lineární funkce	$f(x) = \begin{cases} 1 & x > 1 \\ x & 0 \leq x \leq 1 \\ 0 & x < 0 \end{cases}$	
Lineární funkce perceptron	$f(x) = \begin{cases} ax + b & x \geq 0 \\ 0 & x < 0 \end{cases}$	
Standardní logistická sigmoida	$f(x) = \frac{1}{1 + e^{-x}}$	
Hyperbolický tangens	$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	
Gaussova funkce	$f(x) = \exp[-(x^2)]$	

1.5 Obecné schéma neuronové sítě

Propojením určitých vstupů a výstupů jednotlivých neuronů vzniká neuronová síť. Strukturou (topologií) rozumíme způsob, jakým jsou vzájemně spojeny jednotlivé neurony. Propojení neuronů může být libovolné, v praxi však většinou používáme sítě vícevrstvé, přičemž každá vrstva se skládá z určitého počtu neuronů závislého na řešeném problému, případně technických limitech [6]. První vrstva je pak vrstvou vstupní, poslední vrstva je vrstva výstupní a vrstvy mezilehlé nazýváme vrstvy skryté. Obecně u vícevrstvých sítí je první vrstva jen vrstvou větvící, tzn., že neurony v této vrstvě distribuují vstupní hodnoty beze změny přímo do vrstvy následující. U dopředné neuronové sítě jsou vazby vždy jednosměrné (Neurony vyšších vrstev neovlivňují velikost vnitřních potenciálů neuronů ve vrstvách nižších). Hodnoty vstupního vektoru přivedeného na vstupní vrstvu se šíří neuronovou sítí směrem k výstupní vrstvě, přičemž jsou transformovány jednotlivými neurony, kterými prochází [7]. Schéma jednoduché dopředné třívrstvé sítě je uvedeno na obrázku (Obr. 4).



Obr. 4. Schéma vícevrstvé neuronové sítě.

Kromě vícevrstvých sítí existují také sítě o jedné vrstvě, jako je například Hopfieldova, Kohonenova a mnoho dalších odlišných topologií. Odpověď na otázku vhodného počtu vrstev vícevrstvých sítí přineslo řešení A. N. Kolmogorova třináctého Hilbertova problému (Nemožnost řešit obecně rovnice 7. Stupně složením spojitých funkcí o dvou proměnných), ze kterého vyplývá, že k aproximaci libovolné funkce neuronovou sítí stačí,

aby měla minimálně tři vrstvy s odpovídajícím počtem neuronů v každé vrstvě. Toto řešení však neodpovídá na otázku optimálních počtů neuronů v jednotlivých vrstvách [1].

Pro síť s jednou skrytou vrstvou (dvouvrstvou) se ku příkladu doporučuje [5] počet neuronů:

$$N_{skryt} = \sqrt{N_{vst} N_{výst}} \quad (4)$$

Pro síť se dvěma skrytými vrstvami se například doporučují tyto počty neuronů [5]:

$$N_{skryt-1} = N_{výst} \left(\sqrt[3]{\frac{N_{vst}}{N_{výst}}} \right)^2 \quad (5)$$

$$N_{skryt-2} = N_{výst} \left(\sqrt[3]{\frac{N_{vst}}{N_{výst}}} \right) \quad (6)$$

Kde N_{skryt} určuje počet neuronů skryté vrstvy, N_{vst} počet neuronů vstupní vrstvy a $N_{výst}$ počet neuronů výstupní vrstvy.

1.6 Adaptace neuronové sítě

Neuronová síť je dynamická struktura, v čase se vyvíjí, mění se stav neuronů a adaptují se synaptické váhy. Z tohoto pohledu můžeme rozdělit dynamické chování sítě na tři režimy práce [3]:

1. **Organizační** (Spočívá ve změně struktury sítě).
2. **Adaptivní** (Spočívá ve změně konfigurace sítě – úpravy synaptických vah).
3. **Aktivní** (Spočívá se změnou stavu sítě).

Adaptací (učením sítě) se mění propojení, stav neuronů a adaptují se váhy a to nejčastěji způsobem [8]:

1. **Změnou synaptických vah spojení.**
2. **Změnou prahové hodnoty neuronů.**

Uvažujme učení neuronové sítě takové, při kterém jsou modifikovány váhy synaptických spojení mezi neurony. Předpokládejme, že je možné definovat žádoucí odezvu neuronové sítě a že existuje trénovací množina vzorů k učení.

Pak rozlišujeme tyto typy učení [8]:

1. Učení bez učitele

Tento typ se vyznačuje tím, že neuronové síti jsou předkládány pouze vstupní vzory. Úkolem sítě je správně kategorizovat příbuzné vzory do stejných tříd.

2. Posilující učení

Při tomto typu učení je síti v každém učícím cyklu předána hodnota, zda je odezva na vstupní podnět správná či nikoli. Je zřejmé, že zde přesný trénovací výstup nemusí existovat.

3. Učení s učitelem

Je takové učení, při kterém síti předkládáme ke vzorovým vstupům i očekávaný výstup. Skutečná hodnota výstupu se porovnává s očekávaným výstupem a dle odchylky těchto hodnot aplikujeme změny vah sítě. Zástupcem této kategorie je například algoritmus backpropagation.

4. Účelové učení

Úkolem je najít takový soubor vah, který odpovídá extrémní hodnotě účelové funkce.

5. Soutěžní učení

Je takové, při kterém se před započítím učícího cyklu provede kompetice mezi jednotlivými výkonnými prvky. Vítěz kompetice získává možnost úpravy svých vah.

2 TYPY NEURONOVÝCH SÍTÍ

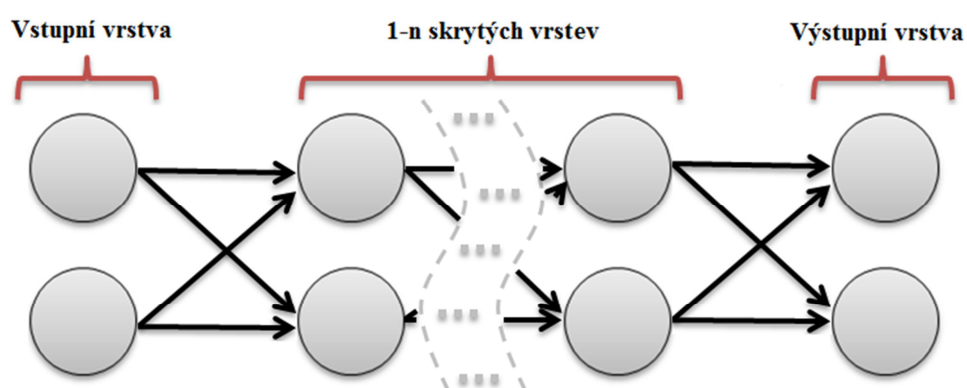
Během doby vývoje neuronových sítí vzniklo velké množství topologií, případně jejich variant. Kromě klasické vícevrstvé topologie existuje mnoho dalších, které nemají charakter vrstevnatých sítí nebo slouží ke specializovaným účelům, proto bych chtěl na tomto místě zmínit základní typy neuronových sítí.

2.1 Vícevrstvé neuronové sítě s dopředným šířením signálu

Tato topologie neuronových sítí se vyznačuje tím, že jsou složeny z vrstev, které jsou mezi sebou propojeny směrem od vstupu k výstupu.

2.1.1 Vícevrstvá neuronová síť s algoritmem backpropagation

Nejznámější a nejpoužívanější typ neuronových sítí, vznikl zobecněním sítě perceptronů. Tyto sítě se skládají minimálně ze tří vrstev [4], jedné vstupní, minimálně jedné skryté vrstvy a jedné vrstvy výstupní (Obr. 5). Každá z vrstev může obsahovat libovolný počet neuronů. K adaptaci synaptických vah se používá algoritmu zpětného šíření chyby (backpropagation).



Obr. 5. Schéma vícevrstvé sítě.

Algoritmus backpropagation

Algoritmus učení zpětným šířením chyby založený na metodě učení s učitelem je vhodný pro učení vrstevnatých sítí s dopředným šířením signálu. Základem metody adaptace je

použití trénovací množiny obsahující požadované vzory (hodnoty excitací vstupní a výstupní vrstvy). Trénovací množinou potom rozumíme množinu uspořádaných dvojic:

$$Tr = \{\{X_1^0, Y_1^0\}\{X_2^0, Y_2^0\} \dots \{X_p^0, Y_p^0\}\} \quad (7)$$

$$X_i^0 = [x_1^0 \ x_2^0 \ \dots \ x_k^0], x_j^0 \in \langle 0,1 \rangle \quad (8)$$

$$Y_i^0 = [y_1^0 \ y_2^0 \ \dots \ y_m^0], y_j^0 \in \langle 0,1 \rangle \quad (9)$$

Kde p je počet trénovacích vzorů, X_i^0 je vektor vstupních hodnot sítě a Y_i^0 je vektor očekávaných hodnot.

Učení sítě může probíhat dvěma způsoby a to tzv.:

1. Offline

Váhy jsou modifikovány až po přivedení všech vzorů na neuronovou síť.

2. Online

Váhy jsou modifikovány bezprostředně po přivedení každého vzoru.

Adaptační algoritmus metody backpropagation (Offline) lze popsat těmito procedurami [6]:

1. Neurony vstupní vrstvy excitujeme vektorem x_i^0
2. Provedeme šíření tohoto signálu dopředným způsobem až k výstupní vrstvě následovně:
 - 2.1. Excitace vstupní vrstvy jsou přivedeny k následující vrstvě, současně jsou upraveny obvodovou funkcí neuronů vyšší vrstvy.
 - 2.2. Každý neuron této vyšší vrstvy je excitován svou přenosovou funkcí.
 - 2.3. Kroky 2.1. a 2.2. jsou opakovány přes všechny skryté vrstvy až k vrstvě výstupní.
3. Porovnáme požadovaný stav Y_i^0 i -tého prvku se skutečnou odezvou sítě Y_i^* . Spočítáme energii za jeden vektor (Odchylku mezi požadovaným stavem a skutečnou odezvou) neuronové sítě:

$$E_k = \frac{1}{2} \sum_{j=1}^m (y_j^0 - y_j^*)^2 \quad (10)$$

Kde n je počet výstupů sítě, y_i^0 je i -tý požadovaný výstup a y_i^* je i -tý skutečný výstup sítě.

4. Kroky 1-3 provedeme všechny dvojice trénovací množiny Tr .
5. Spočítáme globální chybu neuronové sítě jako:

$$E_n = \sum_{i=1}^p E_k \quad (11)$$

6. Pokud je globální chyba nižší, než požadovaná ukončíme algoritmus učení.
7. Aplikujeme úpravy synaptických vah sítě pro každý neuron mimo neurony vstupní vrstvy. Jako velikost změny použijeme hodnotu vypočtenou dle vztahu:

$$\Delta w_i = -\eta \frac{\delta E_n}{\delta w_i} + \mu \Delta w'_i \quad (12)$$

Kde $\eta > 0$ je koeficient učení, $\mu \in \langle 0,1 \rangle$ je koeficient pro omezení velkého skoku změny (momentum), $\Delta w'_i$ je změna váhy z předchozího kroku a $\frac{\delta E_n}{\delta w_i}$ je derivace chyby. Velikost derivace chyby stanovíme jako [6]:

$$\frac{\delta E_n}{\delta w_{ij}} = \delta_{jk} y_j^* (1 - y_j^*) \delta x_i \quad (13)$$

Kde

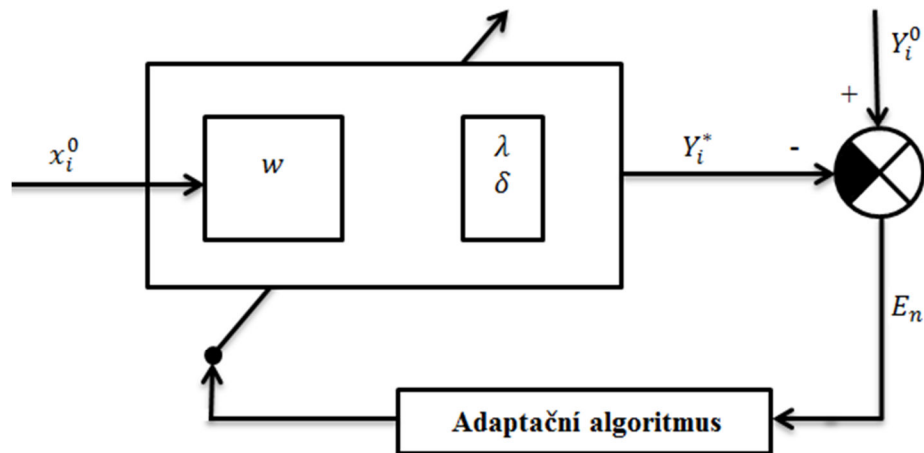
$$\delta_{jk} = \sum_{i=1}^m \delta_{ik} y_i^* (1 - y_i^*) \lambda w_{ij} \quad (14)$$

je rozdílem mezi skutečnou a požadovanou odezvou neuronu j , vzoru k vnější resp. vnitřní vrstvy, kde suma je prováděna od nejvyšší vrstvy, k vrstvě aktuální. Koeficient λ reprezentuje strmost přenosové funkce.

8. Opakujeme učící algoritmus od bodu 1.

Kromě adaptace synaptických vah je dále možno adaptovat i prahy a strmost přenosové funkce (Nehomogenní neuronová síť).

Na obrázku (Obr. 6) je jednoduché schéma adaptační metody řízené velikostí chybové funkce.



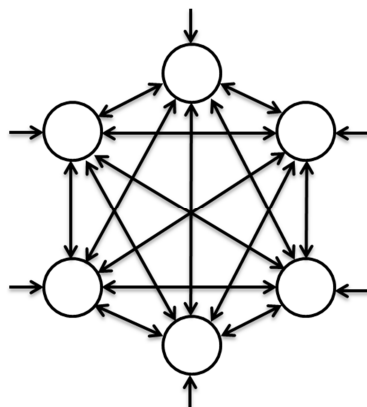
Obr. 6. Schéma adaptační metody backpropagation.

2.2 Rekurentní síť

Tyto síť se vyznačují architekturou, která umožňuje tok informace nejen ve směru od vstupu k výstupu, ale obsahují i zpětné vazby, které přivádí informaci zpět na vstup, či do jiné neuronové vrstvy.

2.2.1 Hopfieldova síť

Hopfieldova síť má jednu vrstvu neuronů s přenosovými funkcemi skokového charakteru. Jedná se o úplně propojenou symetrickou síť, kde každý neuron je propojen se všemi neurony, kromě sebe sama (Obr. 7).



Obr. 7. Hopfieldova síť.

Adaptace sítě probíhá algoritmem učení bez učitele. Během učení se v jednom učícím kole upravují váhy pouze jednoho neuronu. Tato síť se chová jako tzv. autoasociativní paměť, což znamená, že po svém naučení na množině vzorů dokáže rekonstruovat originál z poškozené předlohy.

Učení sítě je práce s maticemi o dimenzi, která je rovna počtu vstupů. Konečná matice vah se získá sečtením dílčích matic [1] [5] vypočtených dle:

$$w_{ij} = \begin{cases} \sum_{s=1}^n x_i^s x_j^s & \forall i \neq j \\ 0 & \forall i = j \end{cases} \quad (15)$$

Kde w_{ij} je hodnota váhy mezi neuronem i a j , x_i^s resp. x_j^s je i -tý resp. j -tý element vzoru S .

Průběh učení probíhá v krocích:

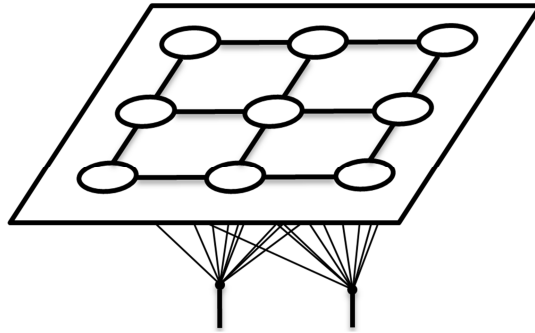
1. Nastavení vah dle rovnice (15) kde ($w_{ij} \in \{-1,1\}$).
2. Opakuj č. 1. Dokud nejsou použity všechny vzory.

2.3 Síť se samoorganizací

Samoorganizující se sítě jsou kategorií neuronových sítí, které využívají ke své adaptaci algoritmus učení bez učitele, čili se samy organizují. Používají se zejména pro analýzu velkého počtu vstupních parametrů, například v technologických procesech, robotice atd.

2.3.1 Kohonenova síť

Tato síť se vyznačuje tím, že obsahuje jedinou vrstvu, která může být reprezentována plošným uspořádáním neuronů (Obr. 8.). V síti jsou neurony uspořádány do určité struktury, nejčastěji do dvourozměrné mřížky. Tímto je přesně definováno okolí neuronu, tedy, s kterými neurony přímo sousedí [9] [1].



Obr. 8. Kohonenova síť.

Ke svému učení nevyžaduje učitele, využívá soutěžní strategii učení, kde při učení neurony soutěží o to, který z nich bude aktivní při daném vstupu. Neuronu, s nejvíce se podobající výstupní hodnotou pro daný vstup jsou poté upraveny váhy, společně s váhami neuronů v jeho nejbližšímu okolí. Zvláštností této sítě je, že neurony postrádají přenosovou funkci, jen se přepočítává vzdálenost vstupního vzoru od realizovaného výstupu, zakódovaném ve váhách neuronu [1].

Algoritmus adaptace se pak skládá z kroků:

1. Předložení vzoru na vstupy neuronové sítě.
2. Výpočet vzdálenosti d pro všechny neurony podle vzorce:

$$d_j = \sum_{i=0}^{N-1} [x_i(t) - w_{ij}(t)]^2 \quad (16)$$

3. Určení vítěze (neuronu s nejmenší vzdáleností od vzoru).
4. Adaptace vah dle vzorce:

$$w_{ij}(t + 1) = w_{ij}(t) + \eta(t)g(x)[x_i - w_{ij}(t)] \quad (17)$$

Kde $\eta > 0$ je koeficient učení a $g(x)$ je tzv. „zvonovitá“ funkce, či jednoduchá obdélníková funkce.

3 EVOLUČNÍ ALGORITMY

Základní myšlenka evolučních algoritmů vychází z evoluční teorie Charlese Darwina přenesená na problematiku současné informatiky a numerické matematiky. Tyto algoritmy představují nový netradiční přístup k hledání optimálního (či suboptimálního) řešení složitých optimalizačních úloh, kde tradiční metody selhávají [10].

Optimalizace je proces, při kterém dochází k určení nejlepšího možného řešení daného problému za daných podmínek a to za pomoci matematického aparátu. Pro nalezení optimálního řešení tedy potřebujeme mít o problému určité informace, které jsou nutné pro vytvoření tzv. účelové funkce. Hodnota této funkce s navrhovanými parametry je poté přímo úměrná kvalitě řešení s těmito parametry. Optimalizace je tedy z matematického hlediska nalezení (nejlépe) globálního extrému této účelové funkce, čili nalezení nejnižší (minimalizační úloha), či nejvyšší (maximalizační úloha) hodnotu této účelové funkce [10].

Hlavní myšlenka evolučních algoritmů spočívá v tom, že na jednotlivé prvky množiny přípustných řešení pohlížíme jako na živé organismy v nějakém umělém životním prostředí. Přitom to, jak si tyto „organismy“ v tomto prostředí vedou, tj. jejich schopnost přežít a schopnost reprodukce, odpovídá tomu, o jak "dobrá" řešení se jedná. Tuto schopnost vyjadřuje tzv. účelová funkce, jejíž hodnota vyjadřuje kvalitu řešení reprezentovaného tímto jedincem. Vlastní hledání pak spočívá ve výběru nějaké počáteční populace těchto „organismů“ a v následné simulaci jejího vývoje pod kontrolou evolučních mechanismů, zahrnující přirozený výběr, dědičnost, reprodukci, mutaci, křížení atd. Jak se tato populace od generace ke generaci vyvíjí, "špatná" řešení (s nevyhovujícím fitness) mají tendenci vymírat, a naopak "dobrá" řešení (s vyhovujícím fitness) se mezi sebou kříží a produkují řešení ještě lepší (z pohledu hodnoty účelové funkce) [10].

Algoritmus se obvykle zastaví při dosažení postačující kvality řešení, po předem dané době, po dosažení počtu požadovaných generací, či v případě že rozdíl mezi nejhorším a nejlepším řešením je menší než požadovaný.

3.1 Učení neuronové sítě evolučním algoritmem

Evoluční algoritmy se zdají být velice vhodnou metodou učení neuronové sítě pro svou robustnost (ve smyslu nalezení globálního optima, či uspokojivého suboptimálního řešení). Na rozdíl od ostatních metod nepoužívají ke svému chodu informaci o gradientu chybové funkce, což je velice důležité zejména u sítí se složitou architekturou, navíc se tím snižuje riziko uváznutí v lokálním extrému chybové funkce [11]. Jednou z největších výhod tohoto způsobu učení je všeobecnost evolučních algoritmů, která dovoluje učení v podstatě jakékoliv topologie, přičemž nezáleží na použitých přenosových funkcích a vhodnou volbou vyhodnocovací operace lze dosáhnout libovolné požadované vlastnosti sítě (např. schopnost generalizovat). Další nespornou výhodou může být učení sítě evolučním algoritmem, spojené s evolučním návrhem topologie sítě, spočívajícím v určení optimálního počtu vrstev v síti, optimálních počtů neuronů v jednotlivých vrstvách a existence spojení mezi nimi. Evoluční algoritmy mohou být použity všeobecně k:

1. Evoluční návrh topologie sítě

Použití stochastických metod je v současné době jediným systematickým způsobem návrhu topologie sítě, bohužel tato procedura vyžaduje vygenerovat řádově tisíce různých neuronových sítí, vytrénovat je některým učícím algoritmem a ohodnotit jejich úspěšnost v řešení. Z důvodu vysoké náročnosti této procedury se většinou pro návrh sítě používá kvalifikovaného odhadu řešitele [12].

2. Evoluční optimalizace váhových a prahových koeficientů

Tato metoda adaptace nahrazuje metodu učení backpropagation, která obecně podává dobré výsledky a je i méně náročná. Klasické gradientní metody ale většinou automaticky končí v nejbližším lokálním optimu, které sice pro řešení většinou stačí, někdy je však kladen důraz na efektivnost sítě, je vhodné použít učení evolučním algoritmem [12] které spočívá v:

1. Kódování úlohy

V této části se sestrojí vzorový jedinec, jako uspořádané pole proměnné délky s jednotlivými váhami a dalšími parametry sítě.

2. Vytvoření počáteční populace

Vytvoří se populace náhodně nastavených vzorových jedinců, tak že každý reprezentuje jednu konfiguraci sítě.

3. Ohodnocení populace

Pro každého jedince se zpětně přepíše jeho parametry do neuronové sítě a ohodnotí se jeho úspěšnost.

4. Testování podmínky ukončení

Pokud v populaci existuje jedinec, který podává vyhovující výsledky, evoluci ukončíme.

5. Rekombinace a mutace jedinců populace

Populaci jedinců podrobíme evolučním procesům, jako je rekombinace (migrace), či mutace.

6. Přejít ke kroku č. 3.

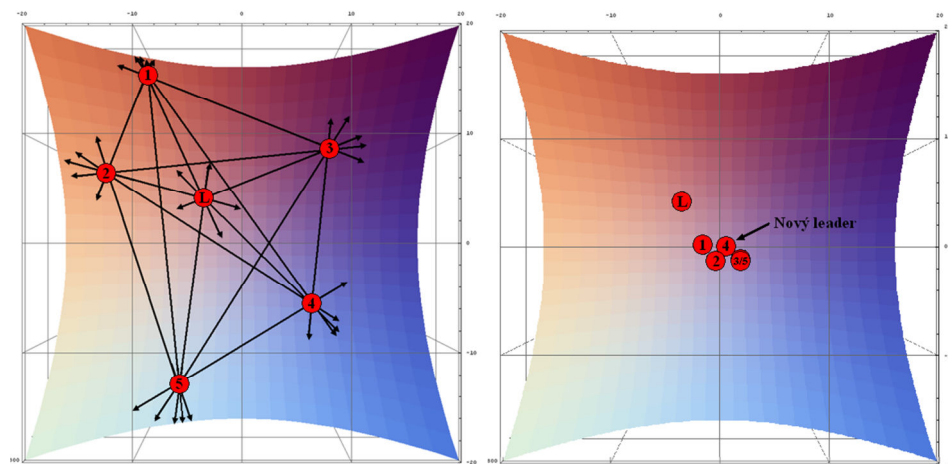
3.2 Princip evolučního algoritmu SOMA

Algoritmus SOMA je algoritmus v mnoha směrech totožný s genetickými algoritmy, jelikož pracuje s populacemi jedinců, přesto se od klasických genetických algoritmů liší, například tím, že za jeho běhu nejsou vytvářeni noví jedinci (generace jedinců).

Algoritmus pohlíží na populaci jako na skupinu inteligentních jedinců, kteří spolupracují na řešení společného problému kooperativním prohledáváním (migrací) prostoru možných řešení [10].

3.3 Strategie AllToAll

Je to strategie, kdy všichni jedinci migrují směrem ke všem ostatním jedincům po diskretních skocích (*step*), dokud je posunutí jedince v prostoru ($t \cdot Step$) menší, než vzdálenost posunutí (*pathLength*). Tuto strategii pro třírozměrný prostor ilustruje obrázek (Obr. 9). Dochází tak k podrobnému prohledávání prostoru možných řešení, ovšem za cenu většího počtu výpočtů [10].



Obr. 9. Princip algoritmu SOMA – AllToAll.

3.4 Parametry a terminologie

3.4.1 Specimen

Jedná se o tzv. vzorového jedince, který obsahuje pro každý parametr minimální a maximální hodnotu. Tyto minimální a maximální hodnoty definuje uživatel s ohledem na fyzikální realizovatelnost a ostatní požadavky na nalezené řešení. Tento jedinec se používá pro vygenerování počáteční populace a vytvoření nového jedince, splňujícího omezení při jejím překročení [10].

3.4.2 PopSize

Dalším uživatelsky definovaným parametrem je velikost populace, která se má použít na prohledávání daného prostoru. Její velikost se odvíjí od složitosti řešeného problému (počet dimenzí, rozlehlost prohledávaného prostoru atd.). Nejnižší přípustná hodnota pro tento parametr je 2, ovšem pro velikosti populace do 10 jedinců by se dal algoritmus přirovnat ke klasickým deterministickým metodám. Proto se jako minimální hodnota doporučuje 10 jedinců. Obecně lze parametr při vysokém počtu optimalizovaných proměnných nastavit na $0,2 * Dim$ až $0,5 * Dim$. Maximální hodnota není omezená, ale měla by být volena s ohledem na velikost prostoru možných řešení a případnou časovou náročnost [10].

3.4.3 PathLength

Tato hodnota určuje, v jaké vzdálenosti od leadera ukončí jedinec svoji migraci. V případě nastavení parametru *PathLength* na hodnotu 1 step, ukončí jedinci svoji migraci na pozici

jedince, ke kterému migrují (V případě, že *Step* dělí *PathLength* celočíselně.). Nastavíme-li tento parametr na hodnotu 2, budou jedinci ukončovat migraci za jedincem, ve dvojnásobné vzdálenosti, která je mezi aktuálním jedincem a jedincem, ke kterému migrují. Minimální hodnota by měla být alespoň 1,1. Maximální hodnota není omezená, ale ve všech simulacích dostačující hodnota byla určena na hodnotu 3 [10].

3.4.4 Step

Je uživatelsky definovaný parametr určující po jak velkých „krocích“ bude migrovat jedinec. Tento parametr tedy určuje, jak podrobně bude prohledávaný prostor prohledán. Nemáme-li o prohledávaném prostoru dostatečné informace je doporučováno nastavit tuto hodnotu co nejnižší. Zvýší se tím pravděpodobnost nalezení globálního extrému. Důležité také je nastavit hodnotu parametru tak, aby nebyla celočíselným násobkem hodnoty *PathLength*. V případě, že by parametr *Step* byl celočíselným násobkem hodnoty *PathLength*, mohlo by dojít ke snížení diverzibility populace tím, že jedinec by mohl v konečném důsledku uváznout na pozici nejlepšího jedince a mohlo by dojít k předčasnému ukončení algoritmu. Doporučená minimální hodnota parametru *Step* je 0,11, je-li ovšem zapotřebí jemnější prohledání prostoru, lze zvolit i nižší. Maximální hodnota je rovna hodnotě *PathLength* [10].

3.4.5 Prt

Jeden z nejcitlivějších parametrů SOMA algoritmu je *Prt*. Tento parametr ovlivňuje směr, kterým se bude jedinec pohybovat. Nabývá libovolné hodnoty z intervalu $< 0,1 >$ a ovlivňuje konstrukci takzvaného perturbačního vektoru, který určuje, zda bude daný parametr jedince perturbován (mutován), či nikoliv. Čím vyšší je *Prt*, tím menší je šance na perturbaci parametru. Je-li parametr *Prt* nastaven na 1, zcela zaniká stochastická složka a algoritmus se chová pouze podle deterministických pravidel, což v určitých situacích může být vhodné pouze pro lokální optimalizaci [10].

3.4.6 Závislost algoritmu SOMA na řídicích a ukončovacích parametrech

Závislost algoritmu na řídicích parametrech můžeme jednoduchou úvahou odvodit ze vzorce pravděpodobnosti nalezení globálního extrému. Tento je pro modifikaci AllToOne definován jako (argumenty účelové funkce uvažujeme jako celočíselné):

$$P_{GE} = \left(\frac{\frac{(PopSize-1)*PathLength*Migrations}{Step}}{L^{Dim}} \right) \quad (18)$$

Kde L^{Dim} je počet všech možných hodnot účelové funkce.

Ze vzorce můžeme vypočítat, že pravděpodobnost nalezení extrému roste s rostoucími parametry $PopSize$, $PathLength$, nebo $Migrations$, či klesajícím parametrem $Step$. Pravděpodobnost nalezení extrému pro verzi AllToAll je definována jako:

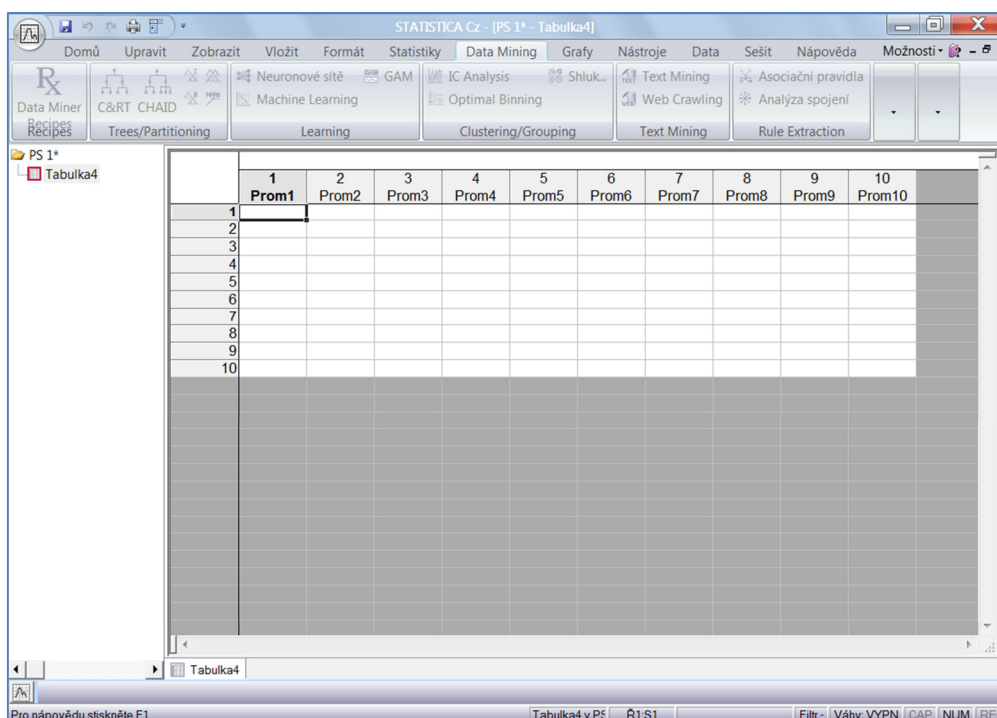
$$P_{GE} = \left(\frac{\frac{PopSize*(PopSize-1)*PathLength*Migrations}{Step}}{L^{Dim}} \right) \quad (19)$$

Ze vzorce je vidět, že pravděpodobnost nalezení extrému u verze AllToAll se zvýší $PopSize$ krát [10].

4 EXITUJÍCÍ NÁSTROJE PRO PRÁCI S NEURONOVÝMI SÍTĚMI

4.1 StatSoft STATISTICA - Automatizované neuronové sítě

StatSoft STATISTICA - Automatizované neuronové sítě je pokročilá aplikace pro použití neuronových sítí (Obr. 10) v praxi. Systém nabízí velké množství prostředků pro práci s neuronovými sítěmi a je určen nejen pro specialisty v oboru neuronových sítí, ale i pro nové uživatele, pro které je k dispozici nástroj, který uživatele provede všemi kroky potřebnými pro vytváření neuronových sítí [13].



Obr. 10. Hlavní okno StatSoft STATISTICA.

Vlastnosti:

- **Snadnost použití**

Aplikace obsahuje nástroj automatický vyhledávač sítě, který provede uživatele krok po kroku postupem vytvoření skupiny rozdílných sítí a výběrem nejvhodnější sítě. Významnou část času během návrhu neuronové sítě aplikace stráví výběrem vhodných proměnných a následně optimalizováním architektury sítě za pomoci heuristického vyhledávání.

- **Integrovaný pre a post-processing**
Pre a post-processing obsahující výběr dat, kódování jmenovitých hodnot, škálování, normalizaci a nahrazování chybějících hodnot.
- **Implementace různých typů neuronových sítí**
Aplikace podporuje několik tříd neuronových sítí pro řešení problémů reálného světa včetně vícerozměrných perceptronů, RBF sítí a Samoorganizovaných map.
- **Optimalizované učící algoritmy**
Jsou implementovány pokročilé učící algoritmy a poskytnuta plná kontrola nad všemi aspekty, které ovlivňují chování neuronové sítě.
- **Volitelné aktivační funkce**
K použití je připravena řada specializovaných aktivačních funkcí například: exponenciální, hyperbolického tangentu a logistické sigmoidy.
- **Plná integrace v rámci systému STATISTICA**
Systém je integrován se systémem STATISTICA, tudíž je k dispozici velký výběr nástrojů pro přípravu dat, analýzy atd..
- **Grafická a statistická zpětná vazba**
Pro vizualizaci dat zobrazuje bodové grafy a 3D grafy povrchů odezvy a sestavuje statistiky, které napomáhají uživateli porozumění chování sítě.

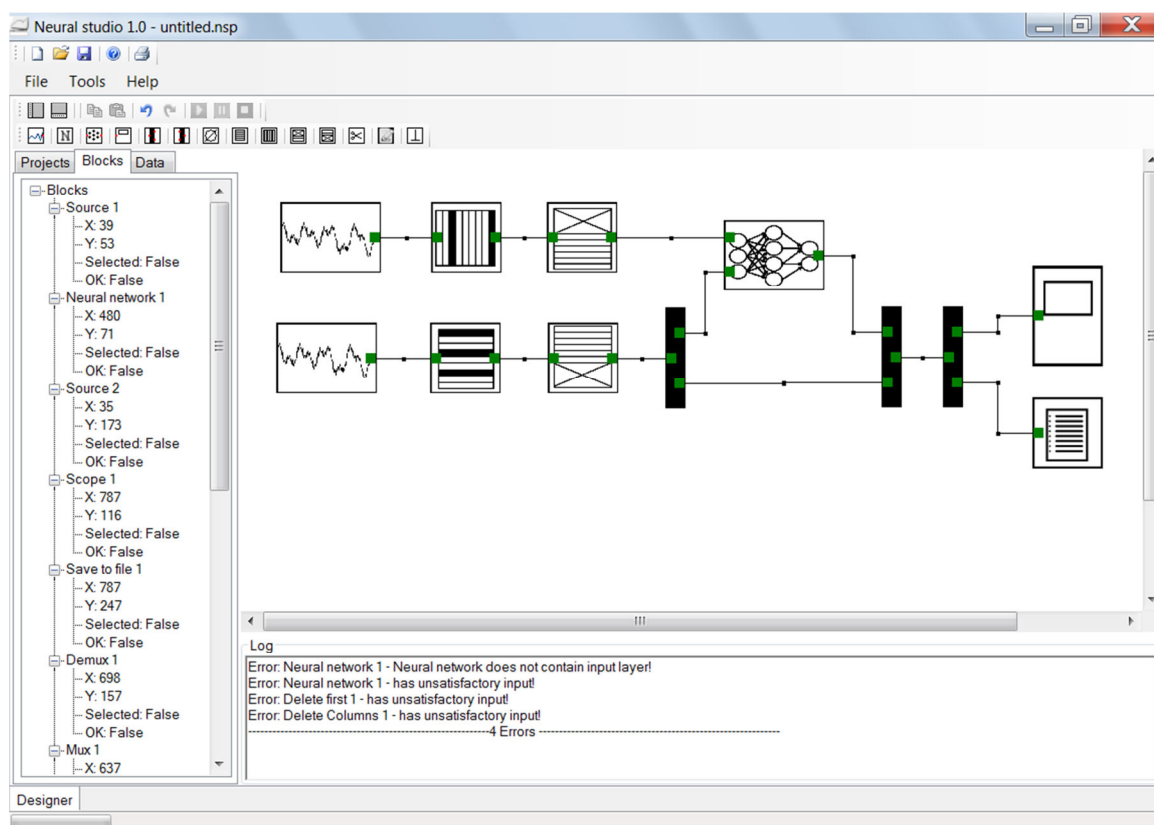
II. PRAKTICKÁ ČÁST

5 POPIS VLASTNÍ IMPLEMENTACE VÝVOJOVÉHO PROSTŘEDÍ PRO NEURONOVÉ SÍTĚ

5.1 Neural studio 1.0

Neural studio 1.0 je vizuální vývojové prostředí pro neuronové sítě, poskytující uživateli vícevrstvou dopřednou neuronovou síť s algoritmy učení backpropagation, učením neuronové sítě evolučním algoritmem a rozšířenými nástroji pro zpracování dat.

Toto prostředí uživateli umožňuje grafický návrh schématu požadovaného řešení pomocí funkčních bloků a signálových cest a prezentaci výstupních dat ve formě tabulek, grafů, nebo simulace.



Obr. 11. Vývojové prostředí Neural studio 1.0.

Výhodami tohoto řešení je například:

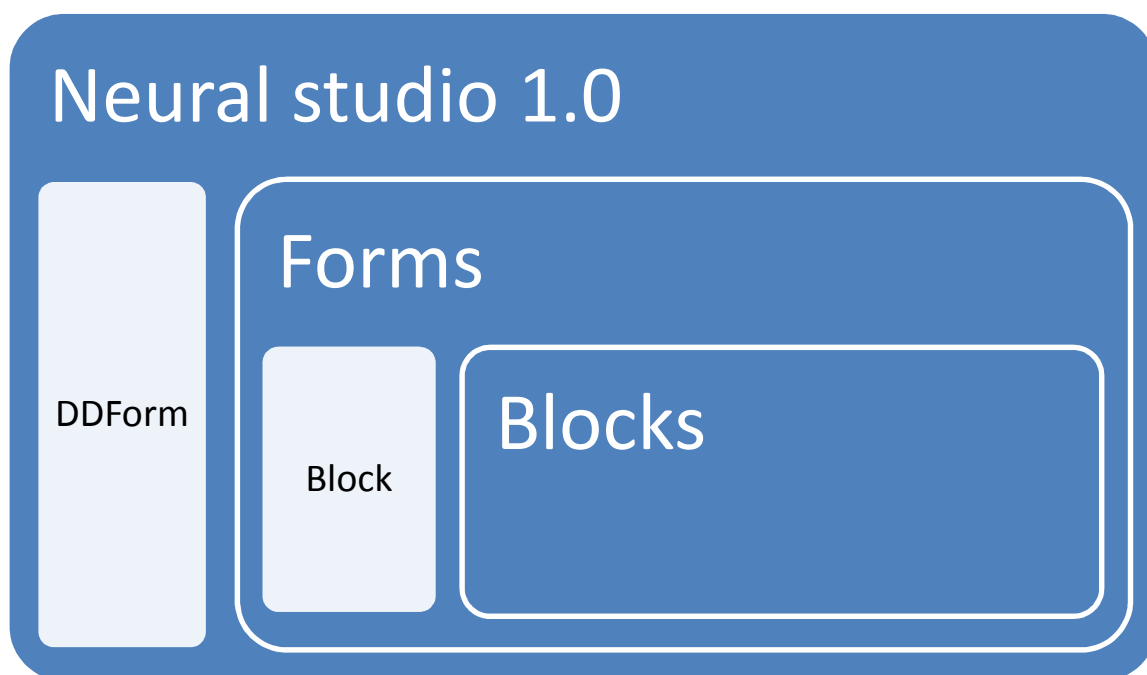
- Jednoduchý a rychlý návrh řešení.
- Aplikace libovolného uživatelsky definovaného řešení.
- Přehlednost prováděných operací s daty.
- Snadná modifikovatelnost řešení.

Program byl úspěšně testován na operačních systémech:

- Microsoft Windows 7.
- Windows XP Professional.

5.2 Struktura programu Neural studio 1.0

Struktura programu byla rozvržena do dvou vrstev, které zajišťují v programu odlišné funkce. První vrstvu reprezentuje vrstva obsluhy vývojového formuláře, která je tvořena vlastní třídou *DDForm*, odvozenou od třídy *Form* která je definována v souboru *DDForm.cs*. Tato odvozená třída obsahuje všechny základní vizuální prvky návrhového formuláře a implementuje obsluhy událostí společné pro všechny její potomky, jako jsou obsluhy událostí Drag & Drop, ukládání historie atd. a obsahuje metody potřebné pro práci s vývojovým formulářem. Druhou vrstvou je práce se samotnými funkčními bloky, které jsou definovány třídou *Block*, odvozenou od třídy *PictureBox*, která je definována v souboru *block.cs*. Tato třída obsahuje všechny metody pro práci s blokem, jako je kontrola správného připojení bloku, obsluhy událostí atd.. Dále obsahuje virtuální metody pro případnou individuální implementaci v každém bloku. Tuto strukturu ilustruje obrázek (Obr. 12).



Obr. 12. Struktura programu Neural studio 1.0.

5.3 Základní třídy metody a proměnné

5.3.1 Třída DDForm

Tato třída se nachází v souboru *DDForm.cs* a je to derivovaná třída od třídy *Form*. Tuto třídu rozšiřuje o další vizuální prvky společné pro všechny návrhové formuláře. Dále implementuje metody pro práci s formulářem, jako jsou:

- *refrshTreeViewDatatables*
- *iDragEnter*
- *DrawLine*
- *Redraw*
- *check*
- *saveFile*
- *BinarySerialize*
- *BinaryDeserialize*

5.3.2 Třída block

Tato třída se nachází v souboru *block.cs* a reprezentuje základní třídu každého funkčního bloku. Implementuje v sobě funkce společné všem blokům odvozeným, některé metody definuje jako virtuální, pro případné individuální implementace speciálních bloků. Tato třída a všechny třídy odvozené musí implementovat rozhraní *ISerializable* pro možnost použití serializace bloku do souboru. Nejdůležitější metody této třídy jsou:

- *propageteToOutputForCheck*

Tato metoda je volána během kontroly zapojení schématu směrem od zdroje dat k ukončovacím prvku. Každému bloku připojenému na jeho výstupy pošle tzv. paket a schéma tabulky pro kontrolu korektnosti dat. Zjištěné nedostatky odesílá do textového pole *tbLog*.

- *pushData*

Metoda odešle na všechny svoje výstupy zpracované data.

- *processData*

Virtuální metoda, která se implementuje pro každý blok individuálně a určuje samotnou operaci, která je s daty provedena.

- *inputCheck*

Virtuální metoda, která se implementuje individuálně pro každý blok a je volána při kontrole zapojení schématu metodou *propageteToOutputForCheck*. V této metodě se

upravuje paket a schéma tabulky tak jakoby data prošla operací prováděnou blokem a tento paket se dále pošle na všechny výstupy.

- *deleteBlock* (Smaže blok z návrhového plátna)
- *copy* (Kopíruje blok)
- *paste* (Vloží blok ze schránky)

5.3.3 Třída NeuralNetwork

Tato třída reprezentuje objektově orientovanou vícevrstvou dopřednou neuronovou síť s algoritmem učení backpropagation. Jedná se o upravenou neuronovou síť dostupnou z [14]. Mezi nejdůležitější metody této třídy patří:

- *Initialize* (Vytvoří systém vrstev neuronů a definuje jejich spojení)
- *BackPropogation_TrainingSession* (Provede jedno trénovací kolo algoritmem backpropagation)
- *PreparePerceptionLayerForPulse* (Vystaví na vstupy vstupní vzor)
- *Pulse* (Přepočítá váhy všech neuronů)
- *CalculateErrors* (Spočítá Chybu sítě)
- *CalculateAndAppendTransformation* (Aplikuje úpravy synaptických vah)

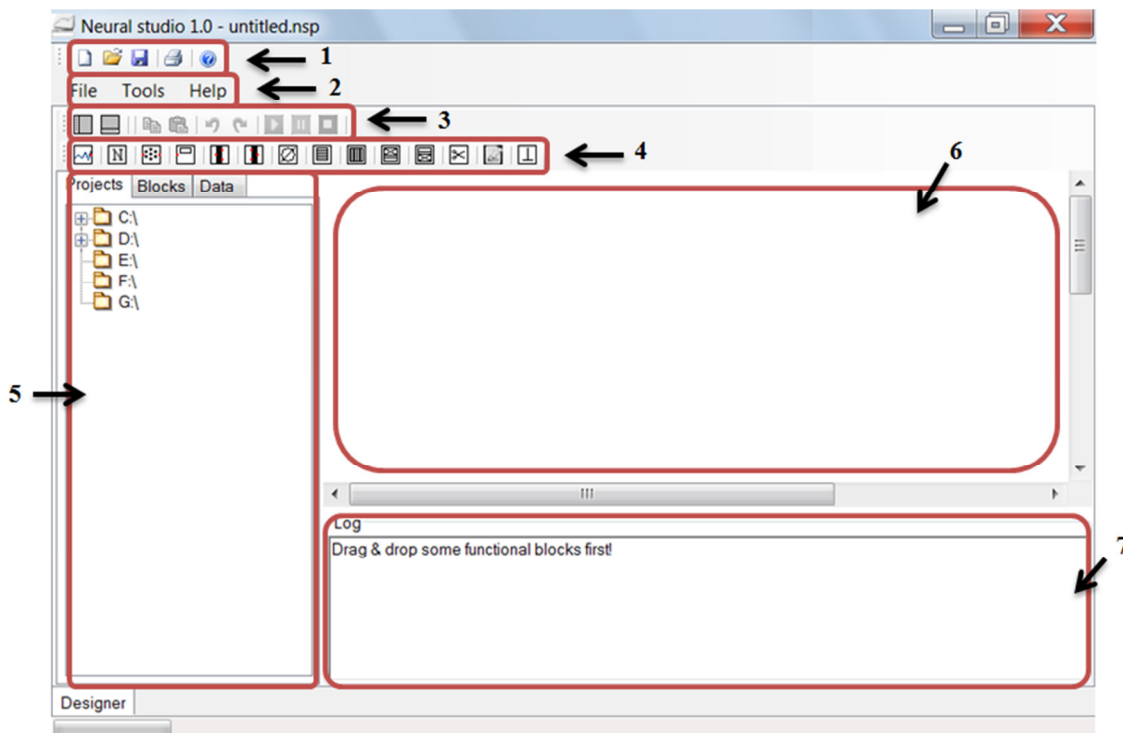
5.3.4 Třída Soma

V této třídě je definován evoluční algoritmus SOMA All2All, který se používá pro učení neuronové sítě. Základní metody této třídy jsou:

- *All2All_OneAfterAnother* (Implementuje evoluční algoritmus SOMA All2All)
- *Run* (V této metodě jsou parametry jednotlivých řešení přepsány do neuronové sítě a ohodnoceny)

5.4 Popis prvků hlavního okna programu Neural studio 1.0

Po spuštění programu neural studio 1.0 se zobrazí hlavní okno aplikace. Toto okno je rozčleněno na několik částí, které jsou vyobrazeny na obrázku (Obr. 13).



Obr. 13. Hlavní okno aplikace Neural Studio 1.0.

5.4.1 Ovládací prvky hlavního okna aplikace

1. Panel nástrojů pro práci s projektem

Tento panel uživateli umožňuje rychlý přístup k operacím načítání, ukládání a tvorby nového projektu, tisku schématu a dále nápovědu k aplikaci.

2. Hlavní menu

V tomto menu se nachází operace načítání, ukládání a tvorby nového projektu, tisku dále přístup k pěti posledním otevřeným projektům a přístup k nápovědě.

3. Panel nástrojů pro práci s návrhovým prostředím

Tento panel obsahuje nástroje pro práci s vývojovým prostředím, jako je zobrazení (skrytí) přídatných panelů, pohyb v historii prováděných akcí a spuštění projektu. Panel je podrobněji popsán dále.

4. Panel funkčních bloků

Na tomto panelu jsou umístěny tlačítka reprezentující dané funkční bloky. Z tohoto panelu se jednotlivé funkční bloky přidají na návrhové plátno operací Drag & Drop. Panel je podrobněji popsán dále.

5. Panel záložek pro práci s projektem

Tento panel obsahuje tři záložky pro práci s projektem a je podrobněji popsán dále.

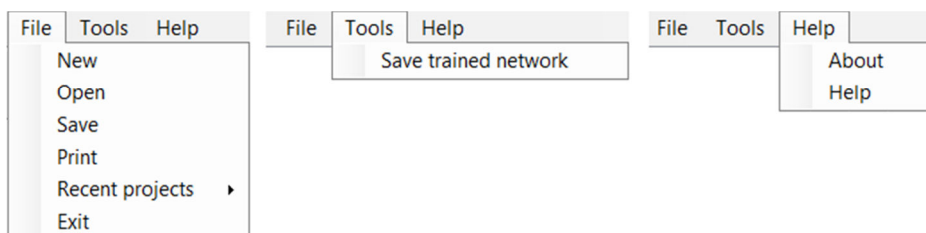
6. Návrhové plátno

Návrhové plátno je oblast v aplikaci, na které probíhá návrh struktury daného funkčního schématu.

7. Panel stavu projektu

V tomto panelu se uživateli zobrazují informace o nalezených nedostatcích v navrhovaném schématu.

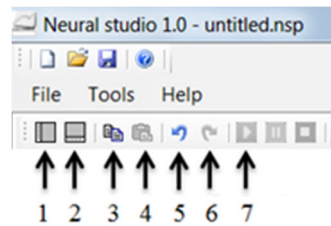
5.4.2 Prvky hlavního menu



Obr. 14. Hlavní menu aplikace.

Toto menu (Obr. 14) nabízí uživateli přístup ke správě projektu, jako je vytvoření nového projektu, otevření uloženého projektu, či uložení současného projektu do souboru. Dále v tomto menu lze vytisknout schéma současného projektu, uložit nastavení neuronové sítě do souboru, získat nápovědu o aplikaci a také zavřít aplikaci.

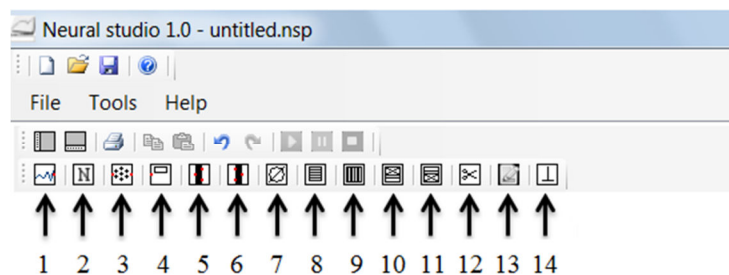
5.4.3 Panel nástrojů pro práci s návrhovým prostředím



Obr. 15. Popis panelu nástrojů pro práci s návrhovým prostředím.

1. Zobrazení / skrytí panelu záložek pro práci s projektem.
2. Zobrazení / skrytí panelu stavu projektu.
3. Kopírování označeného bloku.
4. Vložení kopírovaného bloku.
5. Pohyb zpět v historii prováděných akcí.
6. Pohyb vpřed v historii prováděných akcí.
7. Spuštění projektu.

5.4.4 Panel funkčních bloků

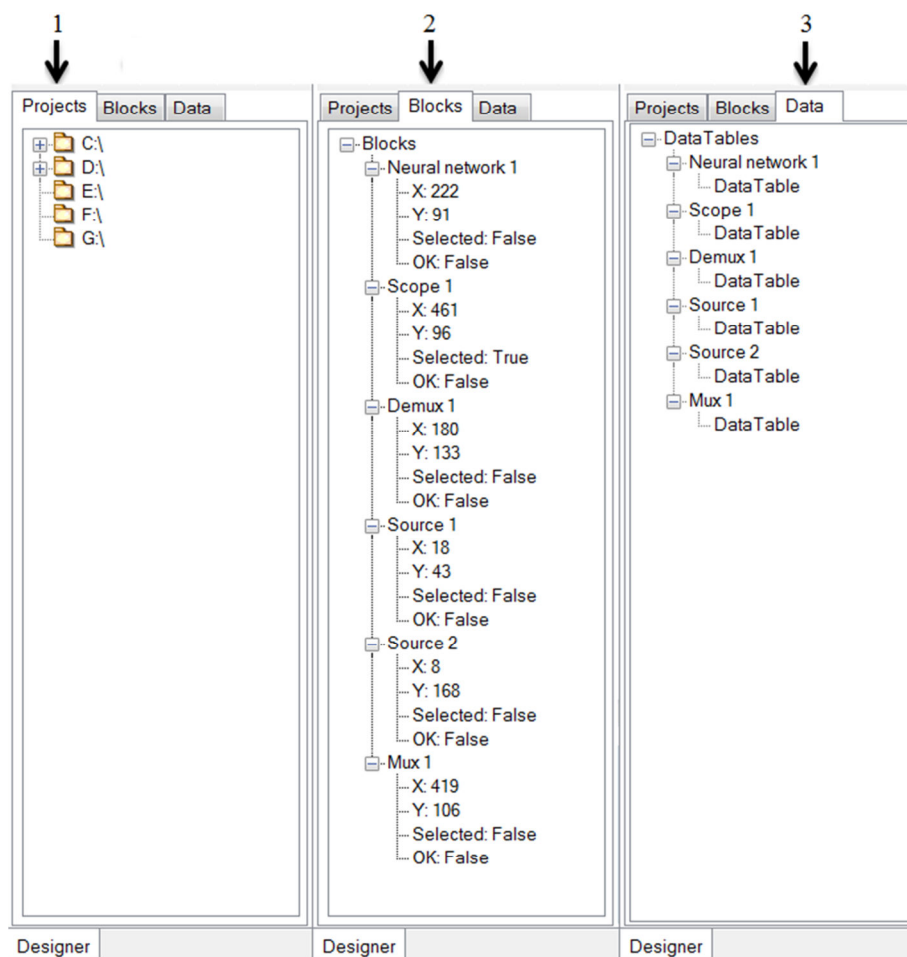


Obr. 16. Popis panelu funkčních bloků.

Následující popis se vztahuje k obrázku (Obr. 16), samotné funkce těchto bloků budou popsány dále.

1. Source blok.
2. Normalize blok.
3. Neural network blok.
4. Scope blok.
5. Demux blok.
6. Mux blok.
7. Average blok.
8. Alternate blok.
9. Delete columns blok.
10. Delete first blok.
11. Delete last blok.
12. Split blok.
13. Text file blok.
14. Terminator blok.

5.4.5 Panel záložek pro práci s projektem



Obr. 17. Panel záložek pro práci s projektem.

1. Záložka souborového prohlížeče

Tento prvek slouží pro rychlou orientaci v souborovém systému uživatele a s jeho pomocí je možné načíst uložený projekt (soubory .nsp), či načíst zdrojová data do projektu. (označením souboru .xls / .xlsx se do projektu přidá blok source)

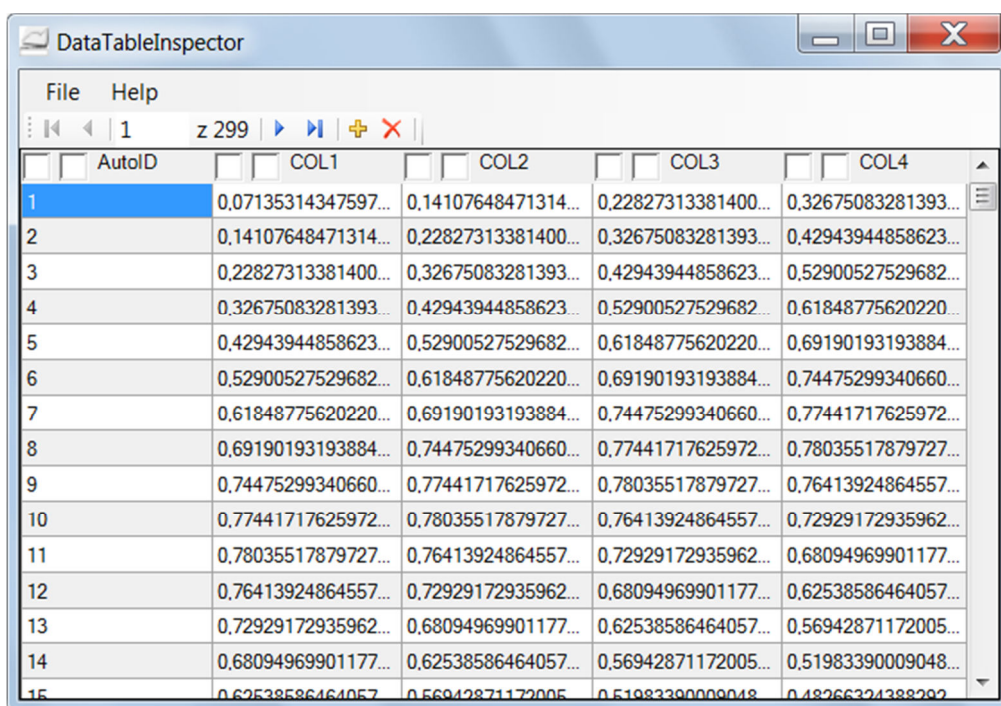
2. Záložka seznamu použitých bloků

Tato záložka slouží pro rychlou orientaci mezi bloky. Nabízí seznam všech bloků použitých na vývojovém plátně a možnost vyvolat označením dialogové okno daného bloku.

3. Záložka dat jednotlivých bloků

Tento prvek poskytuje kontrolu nad datovými tabulkami daných bloků. Po označení datové tabulky daného bloku se zobrazí okno prohlížeče datových tabulek, který uživateli umožňuje data prohlížet, kopírovat a ukládat.

5.4.6 Prohlížeč datových tabulek

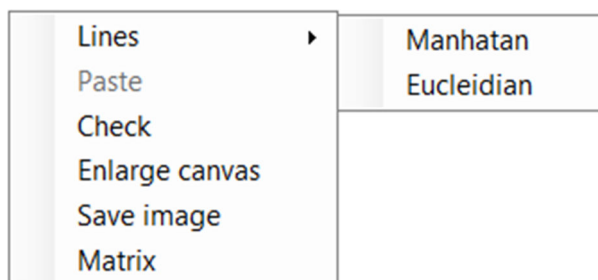


AutoID	COL1	COL2	COL3	COL4
1	0,07135314347597...	0,14107648471314...	0,22827313381400...	0,32675083281393...
2	0,14107648471314...	0,22827313381400...	0,32675083281393...	0,42943944858623...
3	0,22827313381400...	0,32675083281393...	0,42943944858623...	0,52900527529682...
4	0,32675083281393...	0,42943944858623...	0,52900527529682...	0,61848775620220...
5	0,42943944858623...	0,52900527529682...	0,61848775620220...	0,69190193193884...
6	0,52900527529682...	0,61848775620220...	0,69190193193884...	0,74475299340660...
7	0,61848775620220...	0,69190193193884...	0,74475299340660...	0,77441717625972...
8	0,69190193193884...	0,74475299340660...	0,77441717625972...	0,78035517879727...
9	0,74475299340660...	0,77441717625972...	0,78035517879727...	0,76413924864557...
10	0,77441717625972...	0,78035517879727...	0,76413924864557...	0,72929172935962...
11	0,78035517879727...	0,76413924864557...	0,72929172935962...	0,68094969901177...
12	0,76413924864557...	0,72929172935962...	0,68094969901177...	0,62538586464057...
13	0,72929172935962...	0,68094969901177...	0,62538586464057...	0,56942871172005...
14	0,68094969901177...	0,62538586464057...	0,56942871172005...	0,5198339009048...
15	0,62538586464057...	0,56942871172005...	0,5198339009048...	0,48266324388292...

Obr. 18. Prohlížeč datových tabulek.

Této formulář (Obr. 18) slouží pro zobrazení dat na jednotlivých blocích a lze jej vyvolat označením požadované datové tabulky k zobrazení ze záložky dat jednotlivých bloků.

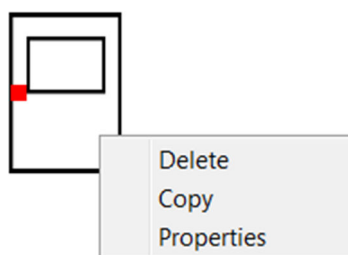
5.4.7 Kontextové menu návrhového plátna



Obr. 19. Kontextové menu návrhového plátna.

V tomto menu (Obr. 19) může uživatel změnit typ spojovacích čar návrhového plátna, vložit kopírovaný blok, explicitně vyvolat proceduru kontroly správnosti zapojení a zvětšit kreslicí plátno. Dále lze uložit navržené schéma jako obrázek (Nekomprimovaný rastrový obraz .bmp / Komprimovaný obraz JPEG) a zobrazit na návrhovém plátně orientační mřížku.

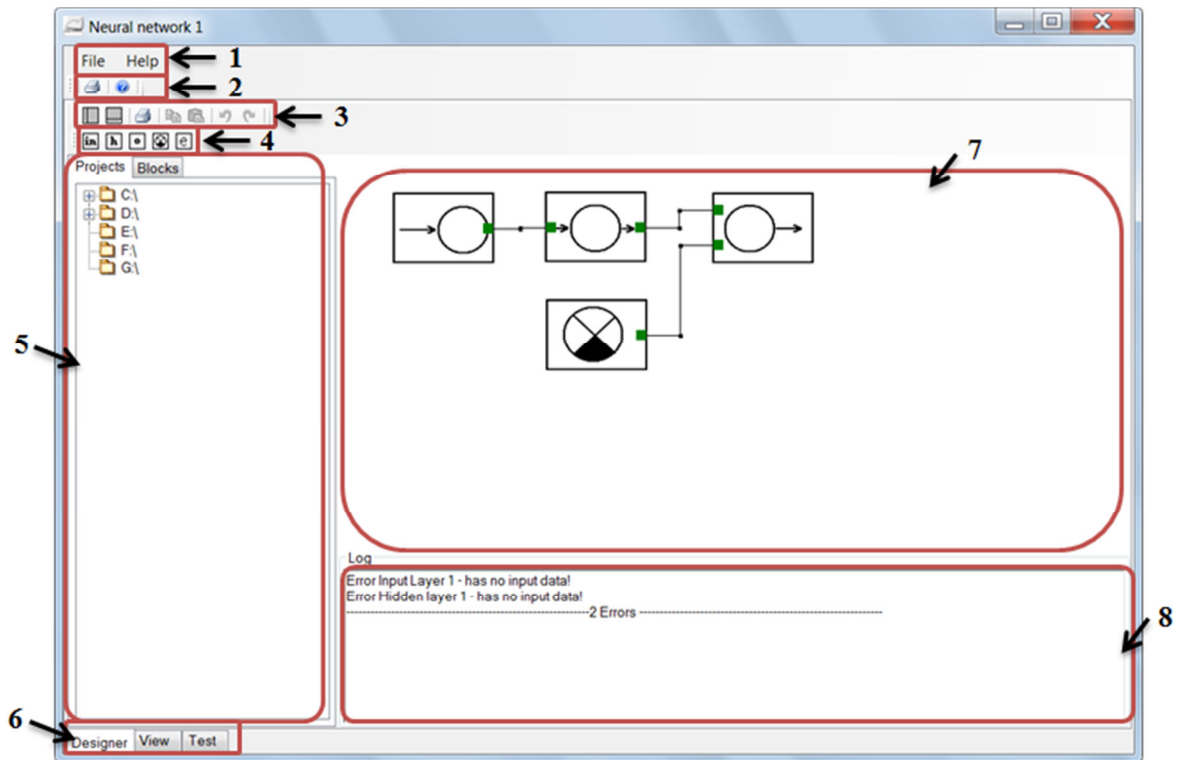
5.4.8 Kontextové menu bloku



Obr. 20. Kontextové menu bloku.

V tomto menu (Obr. 20) jsou k dispozici funkce pro odstranění označeného bloku, dále kopírování označeného bloku a vyvolání dialogového okna bloku.

5.5 Popis dialogového okna neuronové sítě

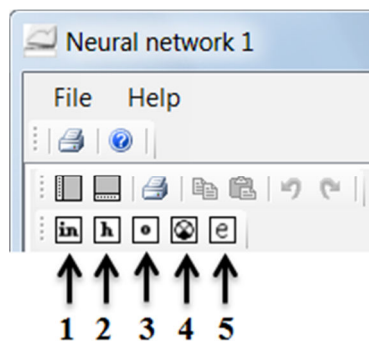


Obr. 21. Dialogové okno neuronové sítě.

1. Hlavní menu.
2. Panel rychlého přístupu.
3. Panel pro práci s dialogovým oknem.
4. Panel funkčních bloků.
5. Záložky pro práci s bloky.
6. Záložky pro práci s neuronovou sítí.
7. Návrhové plátno neuronové sítě.
8. Panel stavu neuronové sítě.

Na tomto dialogovém okně (Obr. 21) na záložce *Designer* uživatel navrhne vnitřní strukturu neuronové sítě a to analogicky pomocí funkčních bloků a signálových cest s pomocí bloků input layer, hidden layer, output layer, expected data a evolution algorithm.

5.5.1 Panel funkčních bloků

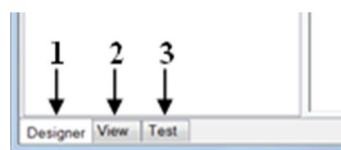


Obr. 22. Panel funkčních bloků neuronové sítě.

1. Input layer (vstupní vrstva).
2. Hidden layer (skrytá vrstva).
3. Output layer (Výstupní vrstva).
4. Expected data (Předpokládané výstupy).
5. Evolution algorithm (učení sítě evolučním algoritmem).

Jednotlivé bloky uživatel umístí na návrhové plátno pomocí operace Drag & Drop z panelu funkčních bloků.

5.5.2 Záložky pro práci s neuronovou sítí,



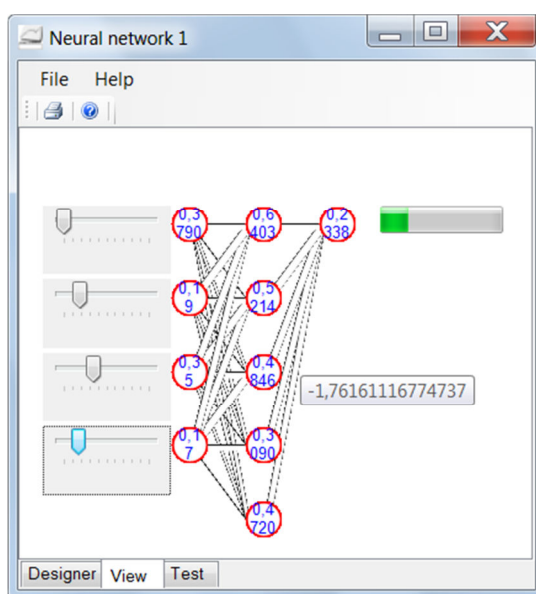
Obr. 23. Záložky pro práci s neuronovou sítí.

Tyto záložky (Obr. 23) slouží k přepínání mezi různými panely tohoto okna, a to:

1. Panel návrhu struktury neuronové sítě.
2. Panel vizualizace neuronové sítě.
3. Panel testování neuronové sítě.

5.5.3 Panel vizualizace neuronové sítě

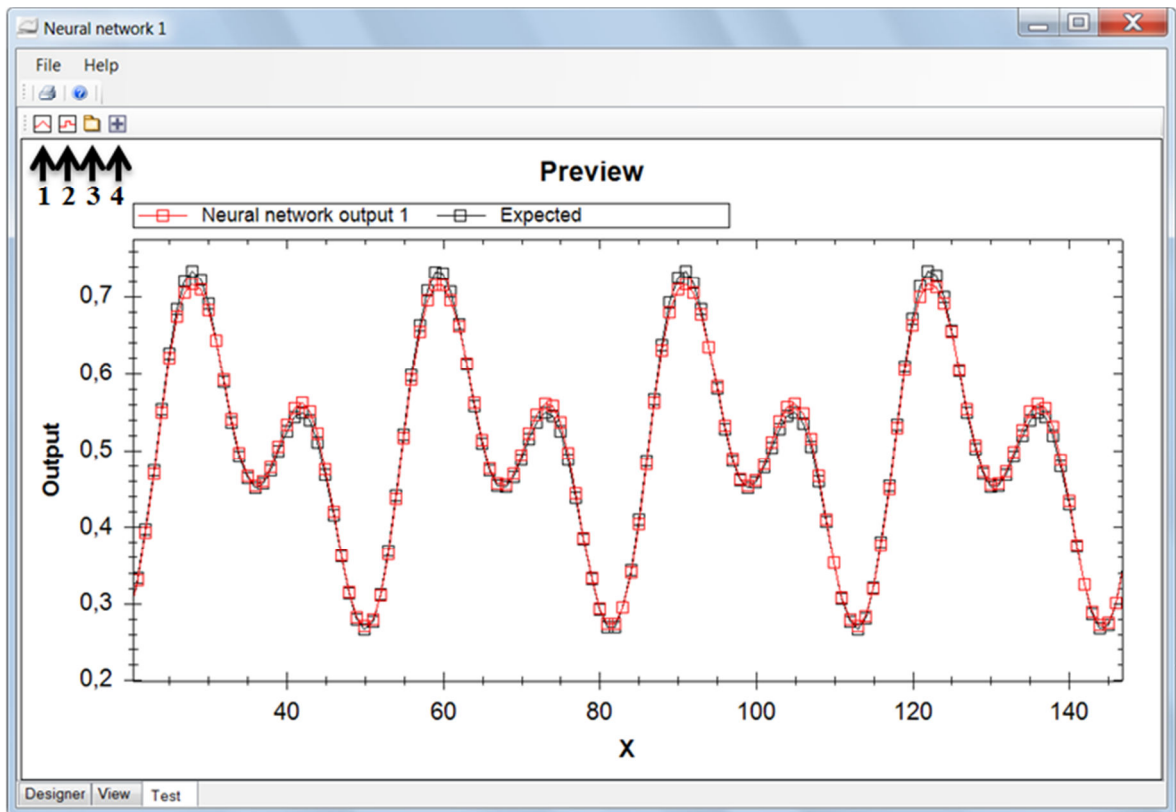
Po úspěšném naučení sítě je na tomto panelu (Obr. 24) k dispozici možnost vizuálně zobrazit strukturu neuronové sítě, manuálně nastavit vstupy pomocí trackbarů, (či poklepáním na dané vyobrazení neuronu a ručním vyplnění hodnoty) a v reálném čase sledovat odezvu neuronové sítě na změny vstupních dat. Dále může na tomto zobrazení uživatel sledovat změny výstupů jednotlivých neuronů a odečíst váhy jednotlivých spojů (ponecháním kurzoru nad danou spojnici).



Obr. 24. Záložka view dialogového okna neural network.

5.5.4 Panel testování neuronové sítě

Tento panel (Obr. 25) umožňuje uživateli po naučení sítě přivést na tuto síť libovolná data a srovnat výstup neuronové sítě s očekávanými daty vizuálně.



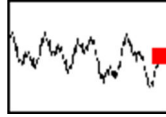
Obr. 25. Záložka test (funkce $\cos(x) \cdot \sin(3x)$)

1. First-Order Hold zobrazení.
2. Zero-Order Hold zobrazení.
3. Tlačítko pro načtení dat a vyhodnocení neuronovou sítí.
4. Libovolná data (například očekávaný výstup k vykreslení).

Na (Obr. 25) je výstup neuronové sítě natréované pro predikci funkce $f(x) = \cos(x) \cdot \sin(3x)$. K učení sítě byl použit algoritmus SOMA All2All. Průběh vykreslený červenou čarou je výstup neuronové sítě a průběh vykreslený černou čarou jsou přesná data.

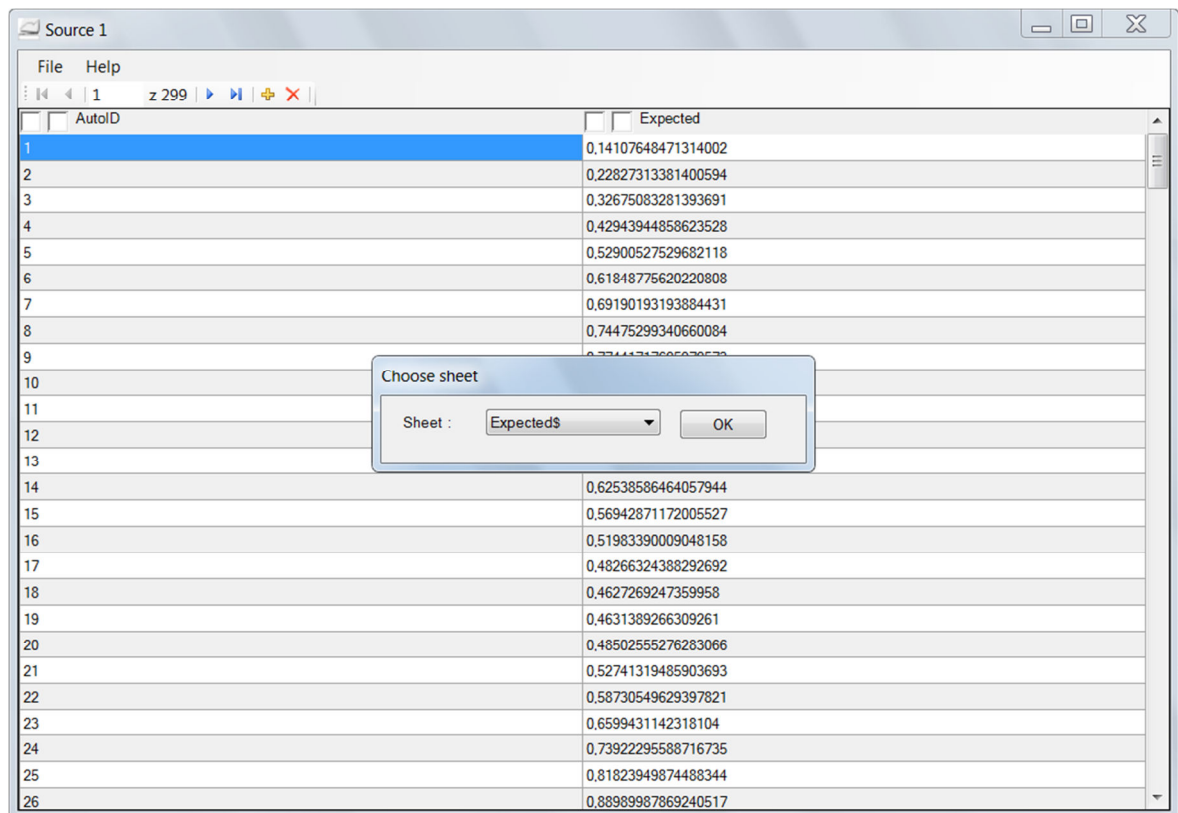
5.6 Popis funkčních bloků programu Neural studio 1.0

5.6.1 Source blok

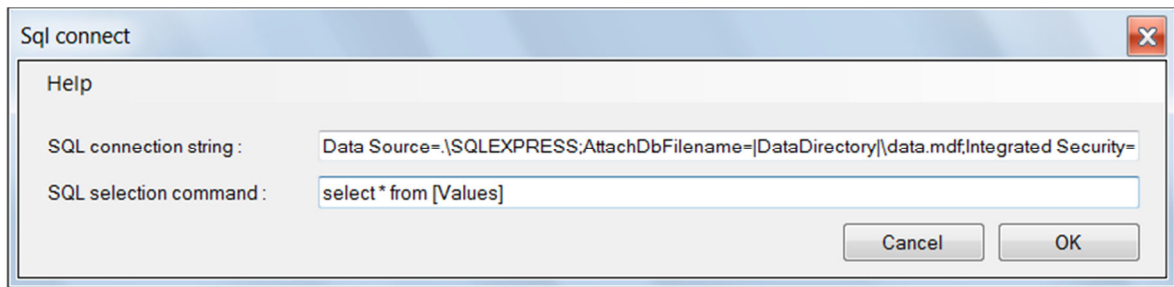


Obr. 26. Source blok.

Tento blok (Obr. 26) slouží pro vstup dat do vývojového prostředí a to z tabulek programu Microsoft Excel s příponami .xls, nebo .xlsx nebo z SQL databáze. Po vyvolání dialogového okna prvku (poklepnutím na něj, či z kontextového menu -> Properties) se zobrazí formulář (Obr. 27) pro načtení dat. Data načteme volbou File -> Open, vybereme soubor k otevření a zvolíme požadovaný list. Chceme-li data načíst z SQL databáze, zvolíme z menu File -> SQL, vyplníme pole connection string a selection string sloužící k připojení k databázi a potvrdíme tlačítkem OK (Obr. 28).

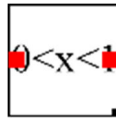


Obr. 27. Dialogové okno prvku source.



Obr. 28. Formulář připojení SQL.

5.6.2 Normalize blok

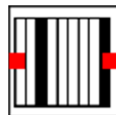


Obr. 29. Normalize blok.

Tímto blokem (Obr. 29) může uživatel normalizovat data na interval $\langle 0,1 \rangle$ a to podle vzorce (20), kde y je nenormalizovaná hodnota, min , je nejnižší hodnota obsažená v tabulce dat, max je nejvyšší hodnota obsažená v tabulce dat a y' je normalizovaná hodnota. Tento blok, je nezbytný při použití dat z jiného intervalu než $\langle 0,1 \rangle$, protože neuronová síť očekává vstupy v intervalu $\langle 0,1 \rangle$.

$$y' = \frac{y - min}{max - min} \quad (20)$$

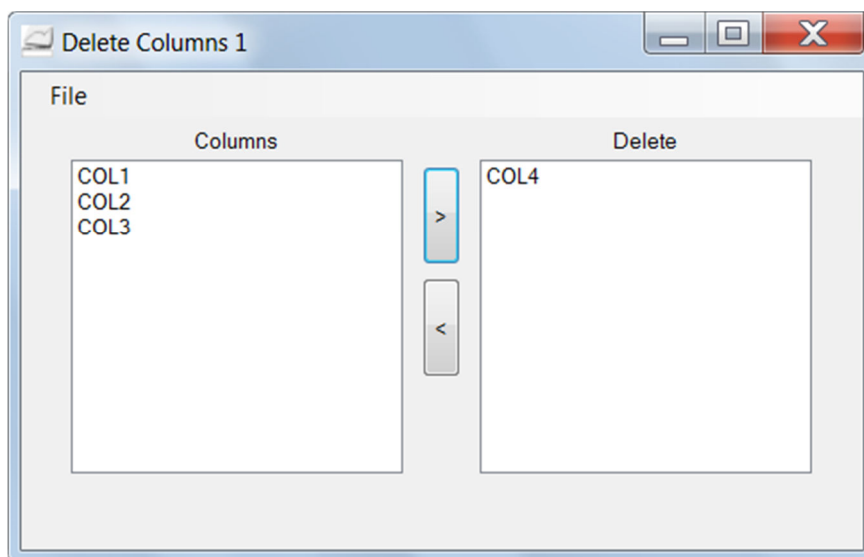
5.6.3 Delete columns blok



Obr. 30. Delete columns blok.

Blok delete columns (Obr. 30) slouží k odstranění přebytečných sloupců v datech. Po vyvolání dialogového okna (Obr. 31) tohoto prvku, může uživatel zvolit tlačítka s šipkami sloupce k odstranění tak, že sloupce, které mají být zachovány, zanechá v levém sloupci

(Columns) a sloupce k odstranění označením a stlačením tlačítka (>) přesune do pravého sloupce (Delete).



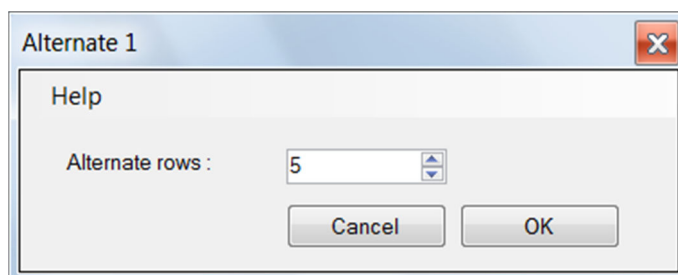
Obr. 31. Dialogové okno prvku delete columns.

5.6.4 Alternate rows blok



Obr. 32. Alternate rows blok.

Tímto blokem (Obr. 32) může uživatel snížit počet řádků dat tak, že do signálové cesty vsune blok alternate rows, který na výstup bloku pošle jen každý n-tý řádek. Tuto hodnotu n lze nastavit na dialogovém okně bloku (Obr. 33) změnou hodnoty alternate rows. Tento blok může být užitečný, pokud uživatel pracuje s detailními daty a nepotřebuje pro další zpracování tak podrobná data.



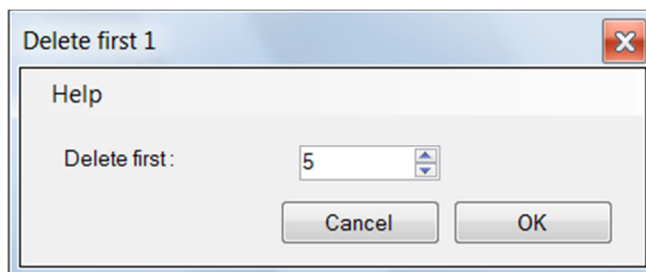
Obr. 33. Dialogové okno alternate rows.

5.6.5 Delete first blok



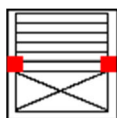
Obr. 34. Delete first blok.

Pomocí tohoto bloku (Obr. 34) může uživatel z dat odstranit n počátečních řádků a to nastavením hodnoty Delete first na dialogovém okně prvku.



Obr. 35. Dialogové okno bloku delete first.

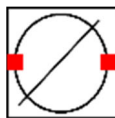
5.6.6 Delete last blok



Obr. 36. Delete last blok.

Pomocí tohoto bloku (Obr. 36) může uživatel z dat odstranit posledních n řádků a to nastavením hodnoty Delete last na dialogovém okně prvku, toto dialogové okno je podobné dialogovému oknu bloku delete first.

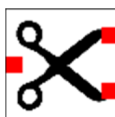
5.6.7 Average rows blok



Obr. 37. Average rows.

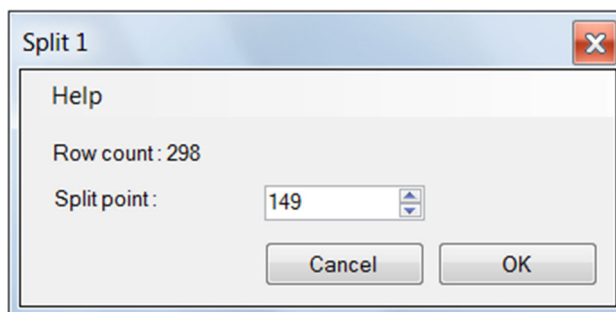
Tímto blokem (Obr. 37) může uživatel snížit počet řádků dat tak, že do signálové cesty vsune blok average rows, který na výstup bloku vystaví vždy průměr z n řádků. Tuto hodnotu n lze nastavit na dialogovém okně bloku změnou hodnoty average rows. Tento blok může být alternativou k bloku alternate rows.

5.6.8 Split blok



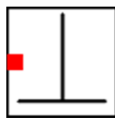
Obr. 38. Split blok.

Tento blok (Obr. 38) slouží k rozdělení vstupních dat na dvě části a to v bodě, který se číselně nastaví na dialogovém okně bloku (Obr. 39). Výsledkem je na výstupu č. 1. (číslováno shora) tabulka dat s řádky s čísly od 1 do hodnoty nastavené na dialogovém okně a na výstupu č. 2. druhá část těchto dat.



Obr. 39. Dialogové okno bloku split.

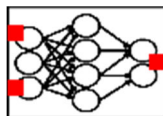
5.6.9 Terminator blok



Obr. 40. Terminator blok.

Tento blok (Obr. 40) slouží ke správnému ukončení signálové cesty v případě, kdy uživatel nepožaduje výstup na zobrazovací jednotku, či textového souboru.

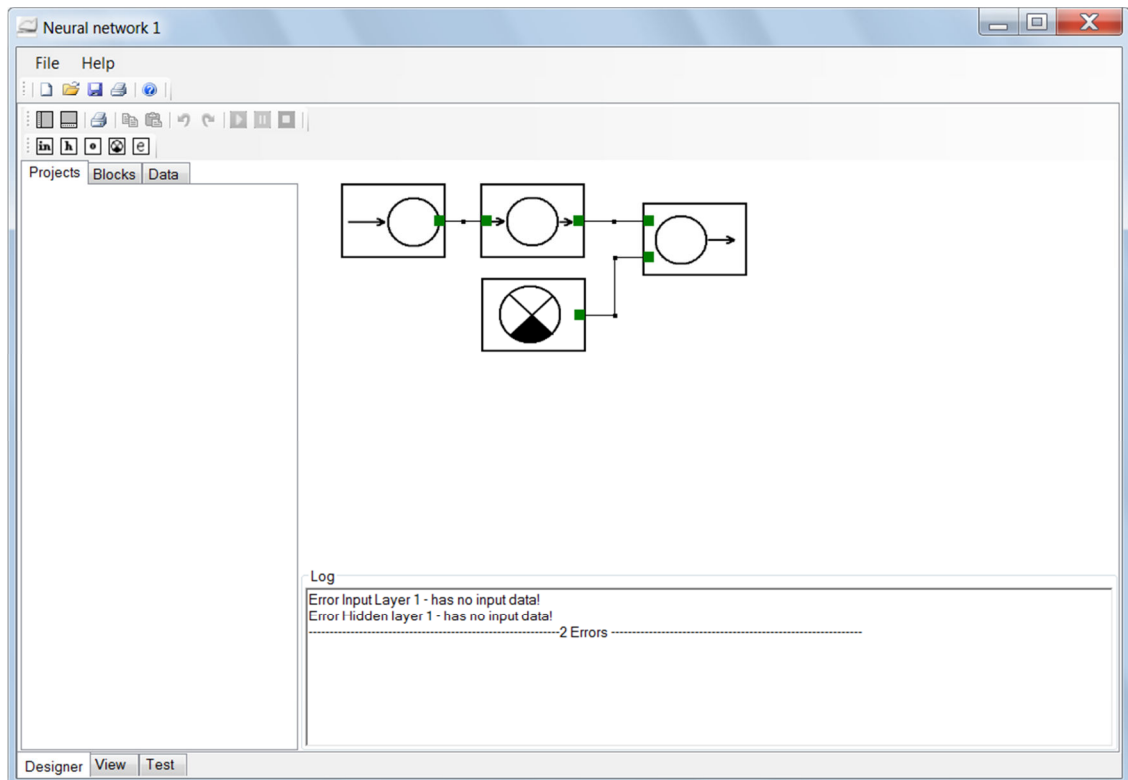
5.6.10 Neural network blok



Obr. 41. Neural network blok.

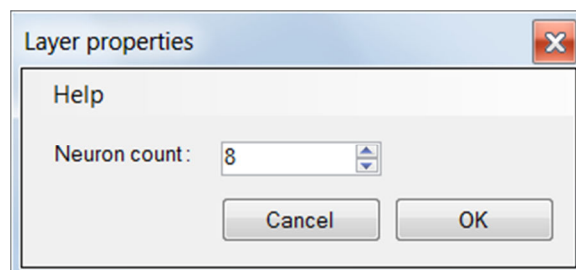
Tento blok (Obr. 41) realizuje neuronovou síť. Uživatel přivede na vstup č. 1. (číslováno shora) data, která požaduje, aby se neuronová síť naučila a na vstup č. 2. data, která očekává na výstupu neuronové sítě.

Dále je nutné, aby uživatel navrhl strukturu neuronové sítě na dialogovém okně na záložce design (Obr. 42) tohoto bloku, nastavil parametry skrytých vrstev (hidden layer blok) (Obr. 43), a parametry bloku expected data (Obr. 44).

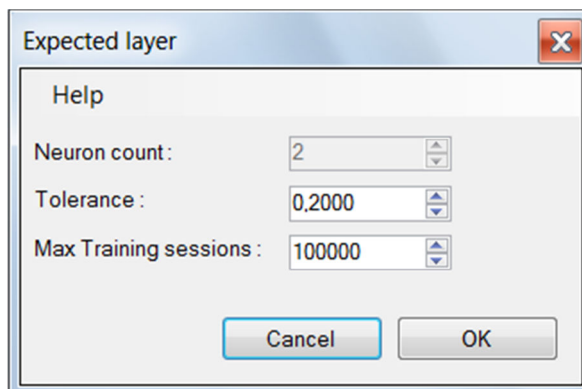


Obr. 42. Dialogové okno prvku neural network

Na obrázku (Obr. 42.) je zobrazena neuronová síť s jednou vstupní vrstvou, jednou skrytou vrstvou a jednou výstupní vrstvou, na kterou jsou napojeny data ke srovnání, která jsou načtena ze vstupu č. 2. bloku neural network. Struktura sítě se navrhuje obdobně jako sestavení celého výpočetního schématu a to přidáním bloků na plátno přetažením bloku (Drag & Drop) z panelu nástrojů.

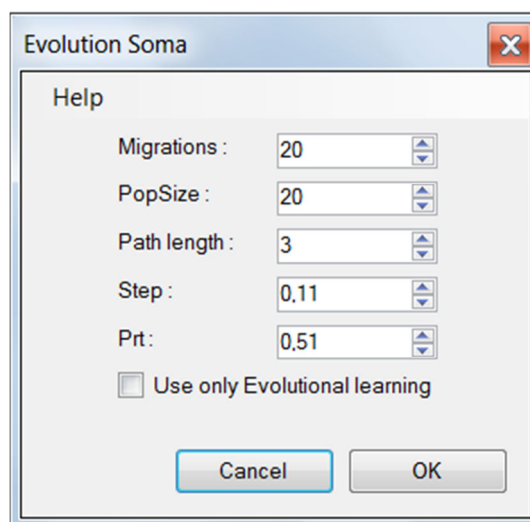


Obr. 43. Dialogové okno nastavení skryté vrstvy.



Obr. 44. Dialogové okno bloku expected data.

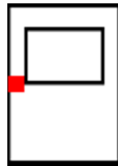
Parametrem skryté vrstvy je počet obsažených neuronů a parametry bloku expected data jsou maximální odchylka výstupních dat neuronové sítě od požadovaných dat (tolerance [%]) a maximální počet iterací algoritmu backpropagation. Parametry vstupních a výstupních dat se nastaví automaticky dle přiložených dat na vstupy tohoto bloku. Pokud uživatel na plátno vloží blok evolution algorithm, provede se po ukončení učícího procesu pomocí algoritmu backpropagation učení neuronové sítě algoritmem SOMA. Řídící parametry tohoto algoritmu se nastavují na dialogovém okně (Obr. 45) tohoto bloku a zvolí-li uživatel možnost „Use only evolutionary learning“, učení sítě algoritmem backpropagation se neprovede a síť bude trénována pouze evolučním algoritmem SOMA All2All.



Obr. 45. Dialogové okno bloku evolution algorithm.

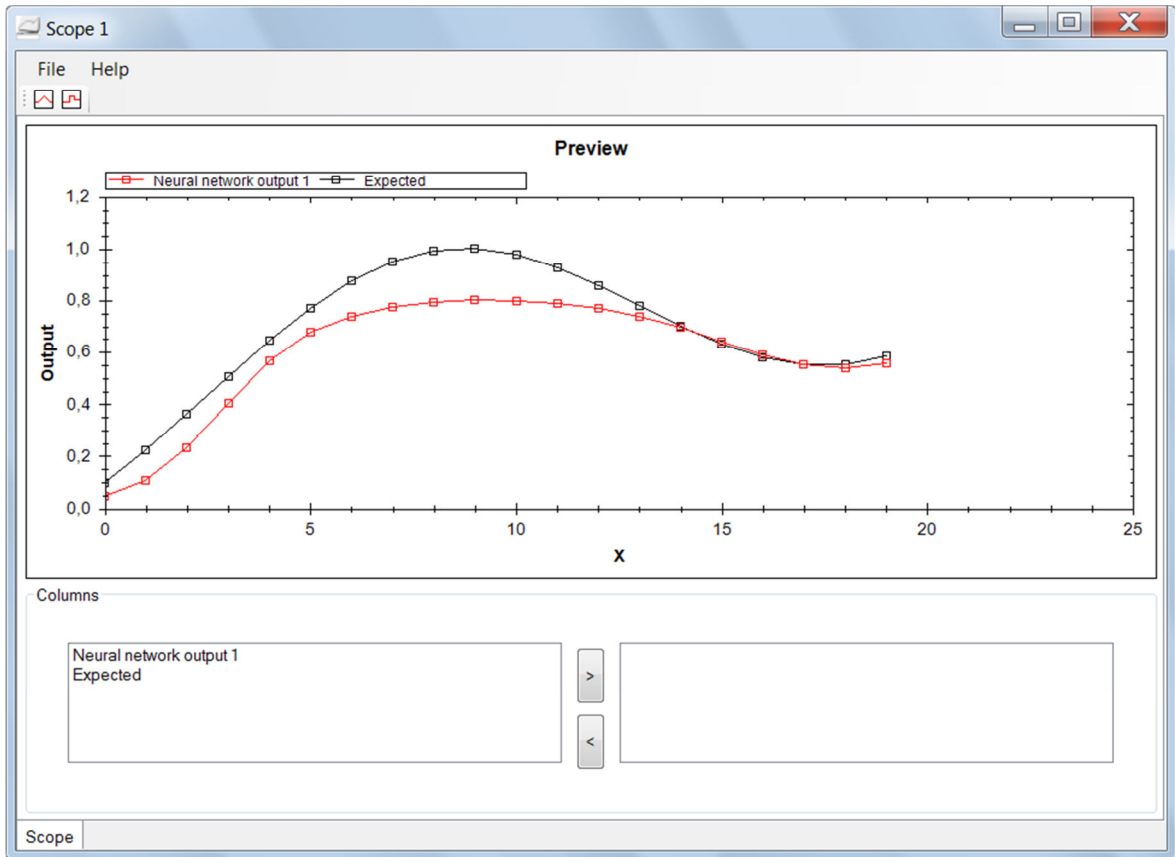
Po úspěšném spuštění projektu je na tomto bloku na záložce test (Obr. 25) k dispozici možnost přivést na vstupy neuronové sítě libovolná data (se stejným počtem sloupců jako původní vzorová data) a vygenerovat grafický výstup odezvy sítě na tyto data. Dále je možno do tohoto grafu vyobrazit další data, např. srovnávací data odpovídající požadovanému výstupu.

5.6.11 Scope blok

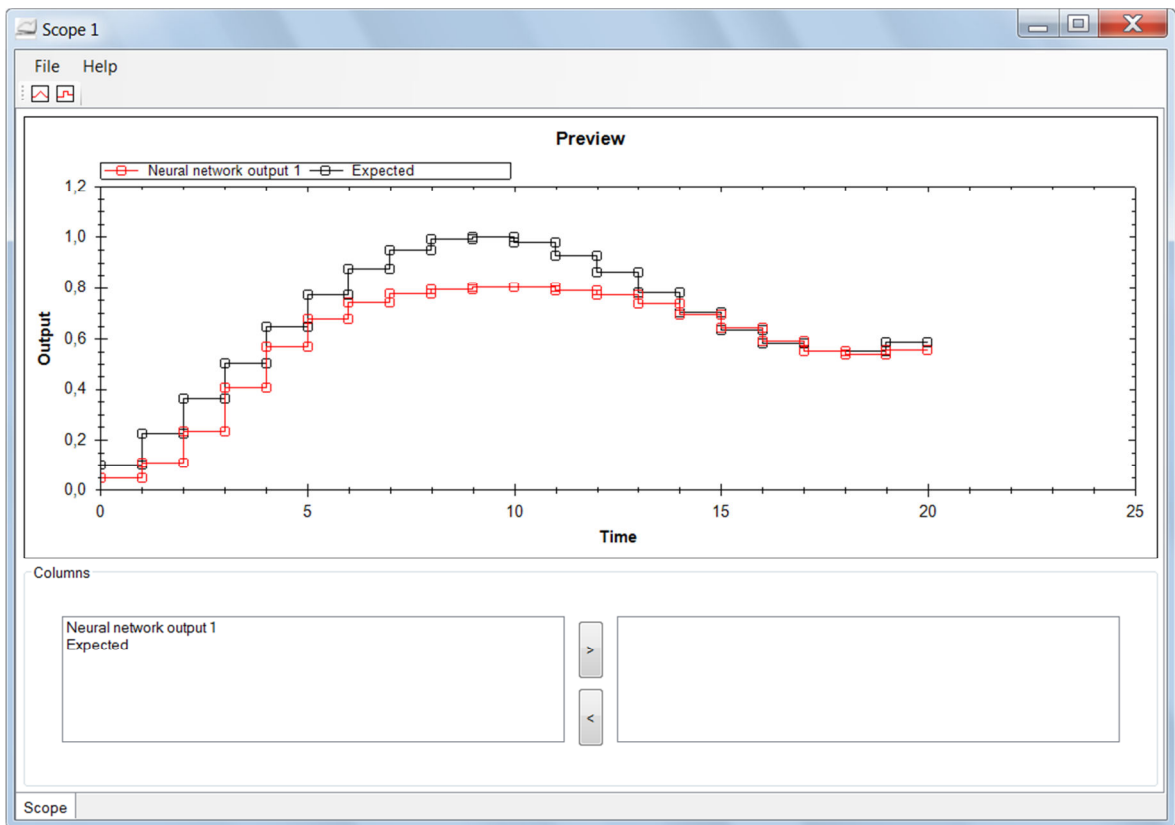


Obr. 46. Scope blok.

Blok scope (Obr. 46) slouží jako zobrazovací jednotka dat. Po úspěšném spuštění projektu můžeme na jeho dialogovém okně (Obr. 47) zobrazit data přivedená na jeho vstup a to buď ve formě lineárních spojovacích čar (First-Order Hold), či ve formě nelineárních čar (Zero-Order Hold). Na obrázku (Obr. 47) je červenou barvou vyobrazen výstup neuronové sítě a černou barvou přesná data pro funkci $f(x) = \cos(x) + \sin(3x)$. Na obrázku (Obr. 48) je tentýž výstup zobrazen ve formě nelineárních čar. K vykreslování grafů v této aplikaci bylo využito komponenty zedgraph [15].



Obr. 47. Dialogové okno bloku scope.



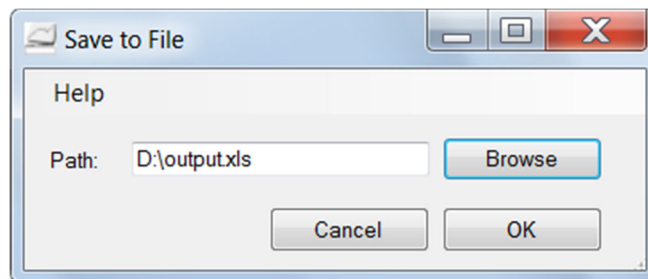
Obr. 48. Scope – Zero-OrderHold zobrazení.

5.6.12 Save to file blok



Obr. 49. Save to file blok.

Přivedením dat signálovou cestou na vstup tohoto bloku (Obr. 49) může uživatel tyto data uložit do tabulky programu Microsoft Excel a to do formátu .xls. Výstupní soubor uživatel musí před spuštěním projektu definovat na dialogovém okně tohoto bloku (Obr. 50).



Obr. 50. Dialogové okno bloku save to file.

5.6.13 Mux blok



Obr. 51. Mux blok.

Tento blok (Obr. 51) převezme data ze svých vstupů a odešle je postupně na jeden výstup. Jedná se o obdobu multiplexoru, tento blok nemá další nastavení a slouží zejména ke spojování jednotlivých datových řad dat k finálnímu zobrazení.

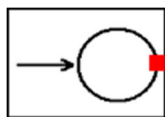
5.6.14 Demux blok



Obr. 52. Demux blok.

Tímto blokem (Obr. 52) lze data přivedená na vstup postupně odeslat na výstupy tohoto bloku, jedná se o obdobu demultiplexoru.

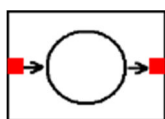
5.6.15 Input layer blok



Obr. 53. Input layer blok.

Tento blok (Obr. 53) dostupný z dialogového okna bloku neural network reprezentuje vstupní vrstvu neuronové sítě, parametry tohoto bloku jsou nastavovány automaticky, dle přivedených dat na vstup č. 1. daného bloku neural network.

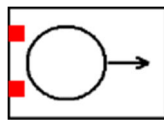
5.6.16 Hidden layer blok



Obr. 54. Hidden layer blok.

Tento blok (Obr. 54) dostupný z dialogového okna bloku neural network reprezentuje skrytou vrstvu neuronové sítě, jediným řídicím parametrem je počet neuronů ve vrstvě, který může uživatel nastavit na dialogovém okně tohoto bloku.

5.6.17 Output layer blok



Obr. 55. Output layer blok.

Tento blok (Obr. 55) dostupný z dialogového okna bloku neural network reprezentuje výstupní vrstvu neuronové sítě, parametry tohoto bloku jsou nastavovány automaticky, dle přivedených dat na jeho vstup č. 2. (číslováno shora). Na vstup č. 1. musí být připojen výstup bloku hidden layer.

5.6.18 Expected data blok



Obr. 56. Expected data blok.

Tento blok (Obr. 56) dostupný z dialogového okna bloku neural network posílá požadované výstupní data ze vstupu č. 2. Bloku neural network na výstupní vrstvu ke srovnání s výstupem z neuronové sítě. Dialogové okno je zobrazeno na obrázku (Obr. 44) a umožňuje nastavení maximální odchylky požadovaných dat od výstupních dat z neuronové sítě a maximální počet iterací algoritmu backpropagation.

5.6.19 Evolution algorithm blok



Obr. 57. Evolution algorithm blok.

Tento blok (Obr. 57) je dostupný z dialogového okna bloku neural network a zajišťuje učení sítě evolučním algoritmem SOMA All2All. Umístí-li uživatel tento blok na plátno návrhu neuronové sítě, bude po naučení neuronové sítě algoritmem backpropagation

vygenerováno evolučním algoritmem PopSize-1 identických neuronových sítí s náhodně nastavenými váhami, a jedna síť v předešlém kroku naučená algoritmem backpropagation. Po dokončení evolučního procesu budou nastavení neuronové sítě podávající nejlepší výsledky přepsány do neuronové sítě bloku neural network. Zvolí-li uživatel v dialogovém okně tohoto bloku „*Use only evolutionary learning*“ učení sítě algoritmem backpropagation se neprovede a síť bude trénována pouze evolučním algoritmem SOMA.

5.7 Použití aplikace Neural studio 1.0

Aplikace neural studio 1.0 je určena pro zpracování dat neuronovou sítí dle požadavků uživatele. Základní myšlenkou této aplikace je grafický návrh postupu zpracování vstupních dat pomocí funkčních bloků. Postup při tvorbě uživatelského řešení daného problému je následující:

1. Umístění požadovaných bloků na návrhové plátno

Požadované bloky přesuneme na návrhové plátno z panelu funkčních bloků na návrhové plátno operací Drag & Drop.

2. Konstrukce signálových cest

Připravené bloky spojíme vždy od výstupu jednoho bloku na vstup druhého bloku kliknutím na požadovaný výstup a druhým kliknutím na požadovaný vstup. Signálová cesta od každého bloku source musí být vždy správně uzavřena a to blokem scope, save to file, nebo terminator. Odstranění signálové cesty provedeme vyvoláním kontextového menu vstupu/výstupu (stlačením pravého tlačítka myši v oblasti vstupu/výstupu) a volbou *Delete*.

3. Načtení vstupních dat

Pro každý blok source na návrhovém plátně provedeme načtení dat z dialogového okna tohoto bloku pro tabulky programu excel volbou z menu „*Open*“, označením požadovaného souboru a výběrem listu. Při načítání dat z databáze SQL zvolíme z menu volbu „*SQL*“, vyplníme pole pro připojení k SQL serveru a pole výběru tabulky.

4. Nastavení parametrů jednotlivých bloků

Pro každý blok zkontrolujeme nastavení jeho funkce na dialogovém okně daného bloku.

5. Návrh struktury neuronové sítě

Otevřeme dialogové okno neuronové sítě, navrhne její strukturu přetažením požadovaných bloků z panelu funkčních bloků neuronové sítě a konstrukcí signálových cest. Každá neuronová síť musí mít právě jednu vstupní vrstvu, jednu či více skrytých vrstev (algoritmus backpropagation je naprogramován pro maximálně 3 vrstvy) a jednu výstupní vrstvu. Dále musí být k výstupní vrstvě na vstup č. 2. připojen blok s očekávanými výstupy sítě. U všech bloků skrytých vrstev nastavíme počty neuronů v těchto vrstvách a nastavíme požadovanou maximální odchylku výstupních hodnot od požadovaných na bloku *expected data*. Pokud požadujeme následné učení sítě evolučním algoritmem SOMA, přesuneme na návrhové plátno blok *evolution algorithm* a nastavíme jej na jeho dialogovém okně. V případě, že na tomto dialogovém okně zvolíme možnost „*Use only evolutionary learning*“ provede se učení sítě pouze evolučním algoritmem.

6. Spuštění projektu

V případě, že jsou všechny parametry bloků nastaveny správně, můžeme spustit projekt z hlavního okna aplikace stlačením tlačítka *run*. V opačném případě dbáme pokynů zobrazených v panelu stavu projektu, případně překontrolujeme zapojení, či nastavení bloků.

7. Verifikace výstupních dat

Zkontrolujeme, zdali výstupní data odpovídají očekávanému výsledku. Pokud ano můžeme využít testovacího nástroje neuronové sítě na dialogovém okně neuronové sítě na panelu vizualizace sítě tlačítkem pro načtení dat a vyhodnocení neuronovou sítí a vybereme data, která chceme, aby neuronová síť vyhodnotila. Tyto data musí mít stejný počet sloupců jako použitá trénovací vstupní data neuronové sítě. V grafu na panelu se zobrazí výstup neuronové sítě použitím těchto dat. Do tohoto grafu můžeme dále vykreslovat další datové řady bez použití neuronové sítě tlačítkem *Libovolná data* z panelu nástrojů. Pro další kontrolu můžeme využít vizualizace neuronové sítě ze záložky *View*.

8. Uložení projektu

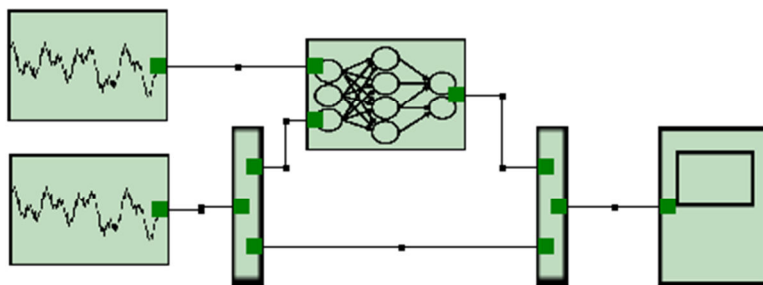
Projekt uložíme například z hlavního menu aplikace (*Save*).

9. Uložení parametrů neuronové sítě

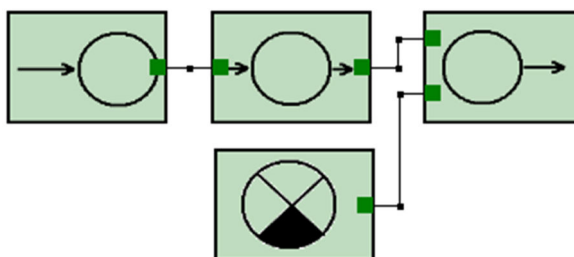
Nastavení sítě pro doplněk *Neural studio 1.0 Daemon* uložíme z menu *tools* -> *Save trained network*.

5.8 Testování programu Neural studio 1.0

V rámci ověření funkce aplikace bylo provedeno základní testování neuronové sítě na naučení funkce exkluzivní disjunkce a predikce hodnoty časové řady. V obou případech bylo dosaženo uspokojivého výsledku a z toho usuzují, že aplikace je schopná sloužit svému účelu. Pro oba testy bylo použito schématu zobrazeném na obrázku (Obr. 58) a třívrstvé sítě na obrázku (Obr. 59).



Obr. 58. Testovací schéma.



Obr. 59. Testovací struktura sítě.

5.8.1 Učení funkce exkluzivní disjunkce

Pro tento test byl připraven soubor dat funkce XOR, který je popsán v tabulce (Tab. 5).

Tab. 5. Trénovací množina pro učení XOR.

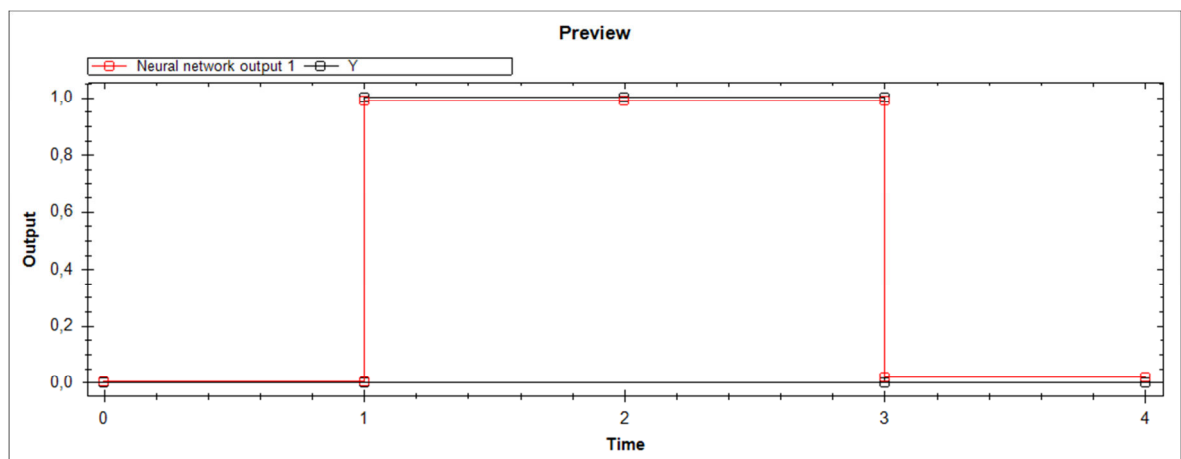
A	B	Y
0	0	0
1	0	1
0	1	1
1	1	0

Neuronové síti byla předložena trénovací množina o čtyřech vzorech. Každý vzor obsahoval hodnotu A, hodnotu B a požadovanou výstupní hodnotu neuronové sítě Y. Tato trénovací množina je k dispozici na přiloženém CD.

Nastavení neuronové sítě je popsáno v tabulce (Tab. 6) Na obrázku (Obr. 60) je výsledek naučení sítě. Černou čarou je vykreslena požadovaná odezva sítě a červenou čarou skutečná odezva sítě.

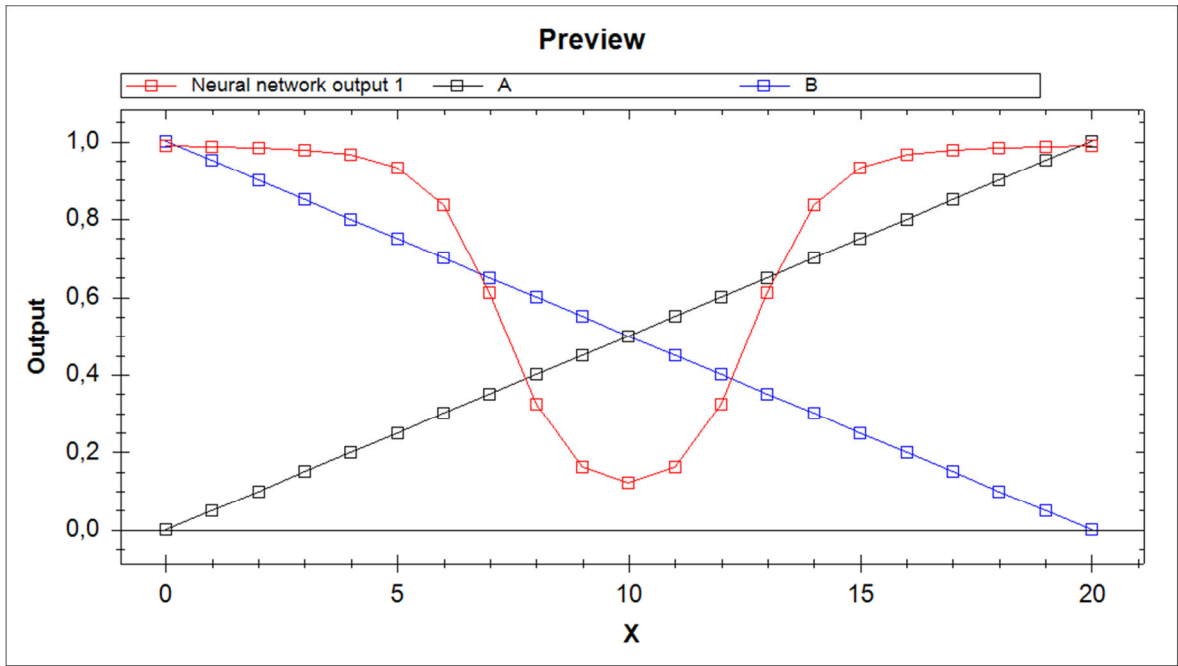
Tab. 6. Nastavení Neuronové sítě (XOR).

Počet sítě	vrstev	Počty neuronů [vst,skrytá,výst]	Evoluční algoritmus	Max. odchylka od vzoru
3		2-3-1	Ne	2%

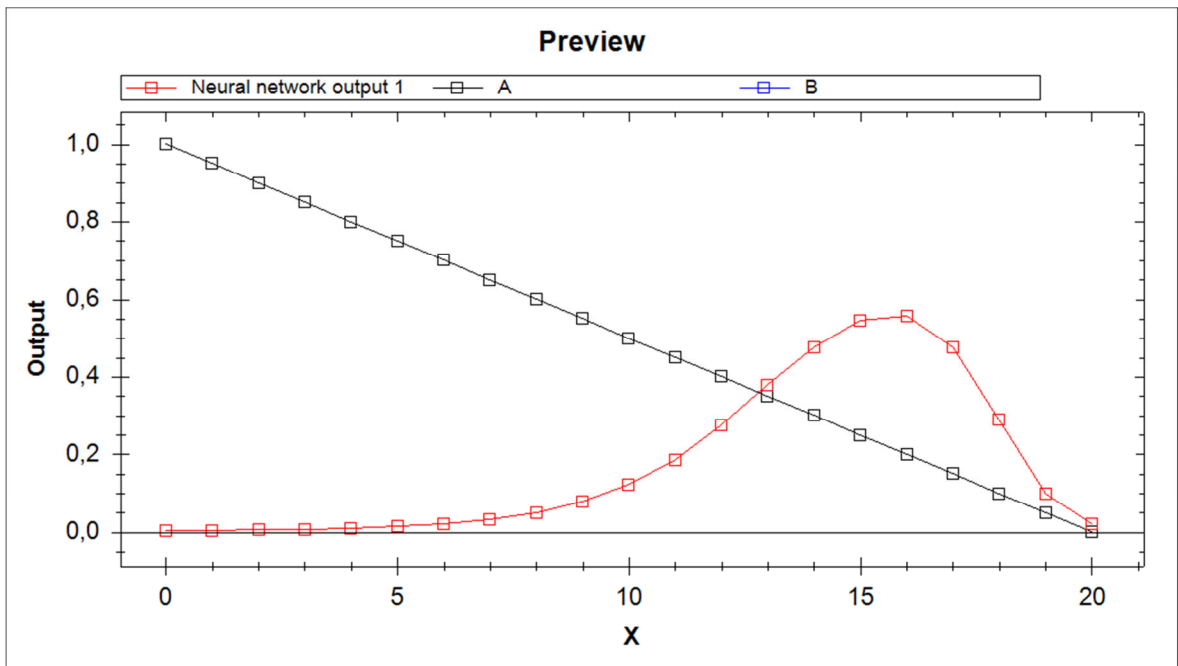


Obr. 60. Výsledek učení funkce XOR.

Verifikace proběhla na dvou souborech dat o 21 vzorech. V prvním souboru dat (Obr. 61) hodnota parametru „A“ rostla, zatímco parametru „B“ klesala. V druhém souboru dat (Obr. 62) se hodnoty obou parametrů snižovaly. Tyto soubory dat jsou k dispozici na přiloženém CD.



Obr. 61. Verifikace funkce XOR na souboru dat č. 1.



Obr. 62. Verifikace funkce XOR na souboru dat č. 2.

5.8.2 Predikce hodnoty časové řady

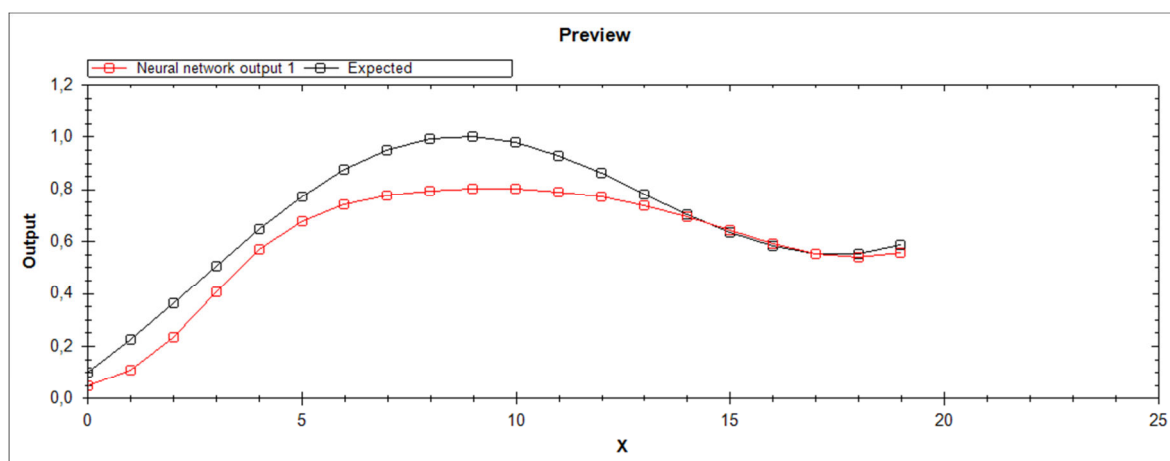
Pro tento test byl připraven soubor dat funkce $f(x) = \cos x + \sin 3x$ normalizované na interval $\langle 0,1 \rangle$, vzorkované s periodou 0,1 z intervalu $\langle 15,-15 \rangle$

Neuronové síti byla předložena trénovací množina o prvních 20-ti vzorech. Každý vzor obsahoval 4 po sobě jdoucí hodnoty funkce $f(x) = \cos x + \sin 3x$ ze souboru dat a pátou následující hodnotu funkce, jako požadovanou výstupní hodnotu neuronové sítě.

Nastavení neuronové sítě je popsáno v tabulce (Tab. 7) Na obrázku (Obr. 63) je výsledek naučení sítě na těchto 20-ti vstupních vzorech. Černou čarou je vykreslena požadovaná odezva sítě a červenou čarou skutečná odezva sítě.

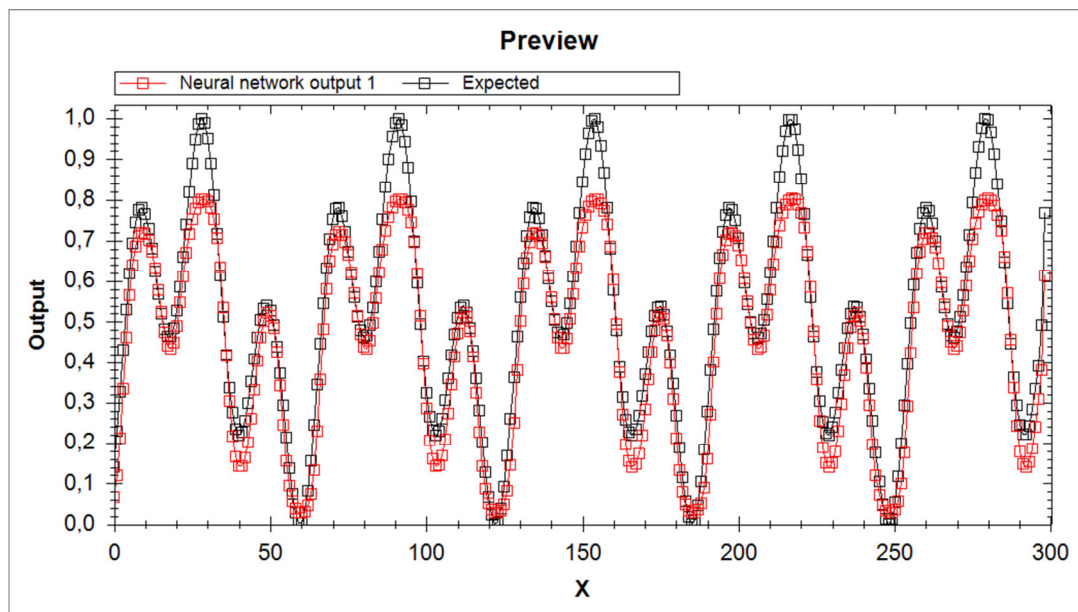
Tab. 7. Nastavení Neuronové sítě (Predikce).

Počet sítě	vrstev	Počty neuronů [vst,skrytá,výst]	Evoluční algoritmus	Max. odchylka od vzoru
3		4,5,1	Ne	20%



Obr. 63. Výsledek predikce na trénovací množině.

Na obrázku (Obr. 64) je znázorněn test na celém souboru dat. Černou čarou je vykreslena požadovaná odezva sítě a červenou čarou skutečná odezva sítě.



Obr. 64. Výsledek predikce na celém souboru dat.

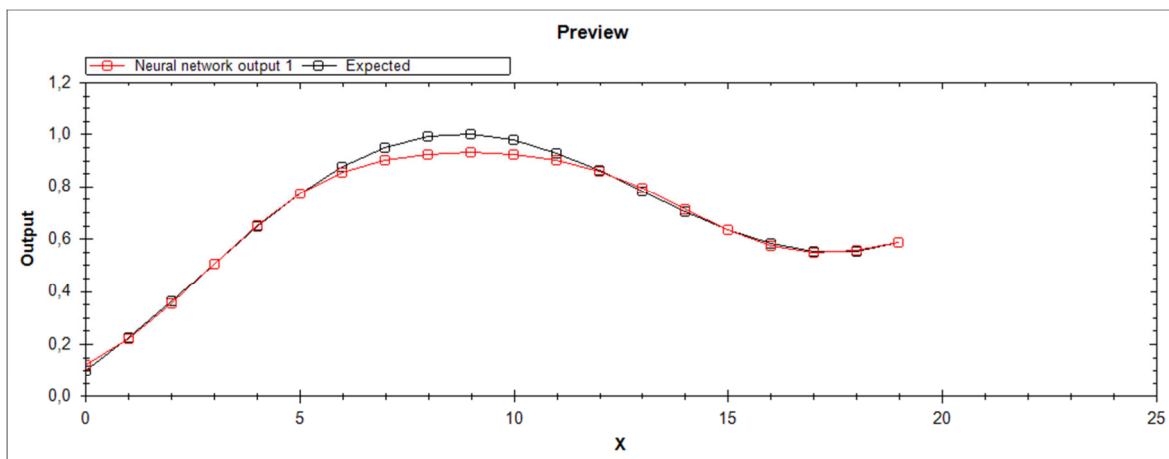
Dále byla uskutečněna predikce výše zmíněné funkce s nastavením popsaném v tabulce (Tab. 8) a (Tab. 9). Na obrázku (Obr. 65) je výsledek naučení sítě na těchto 20-ti vstupních vzorech. Černou čarou je vykreslena požadovaná odezva sítě a červenou čarou skutečná odezva sítě.

Tab. 8. Nastavení Neuronové sítě (Predikce + SOMA).

Počet sítě	vrstev	Počty neuronů [vst,skrytá,výst]	Evoluční algoritmus	Max. odchylka od vzoru
3		4,5,1	Ano	10%

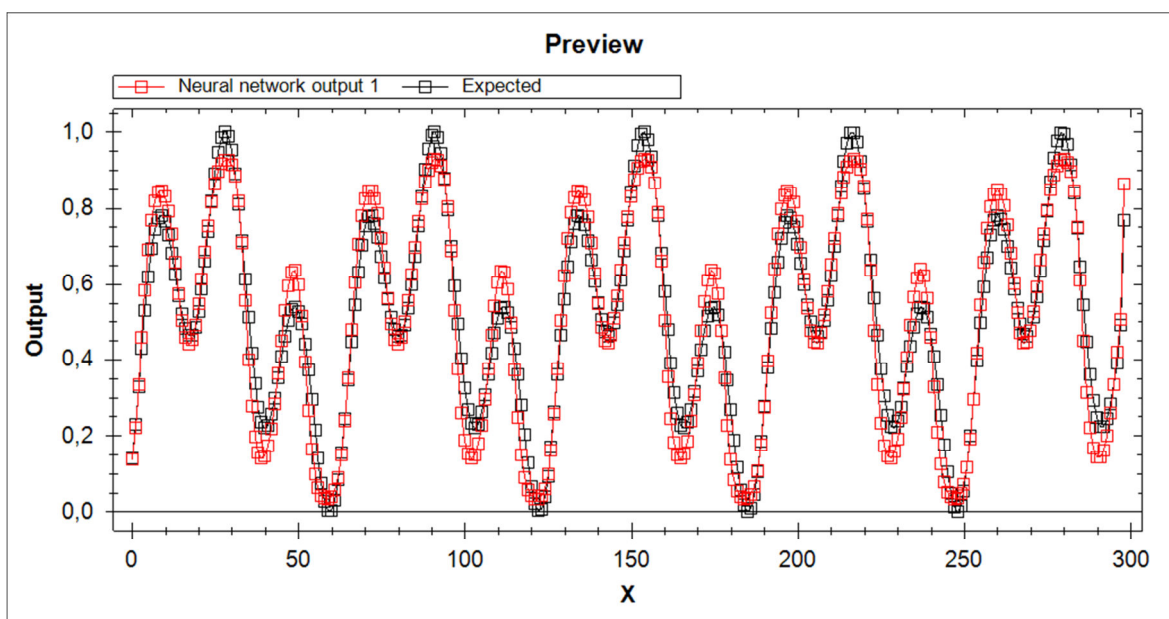
Tab. 9. Nastavení Algoritmu SOMA.

Migrations	20
PopSize	30
Path length	3
Step	0,11
Prt	0,51
Use only evolutionary learning	Ano



Obr. 65. Výsledek predikce na trénovací množině.

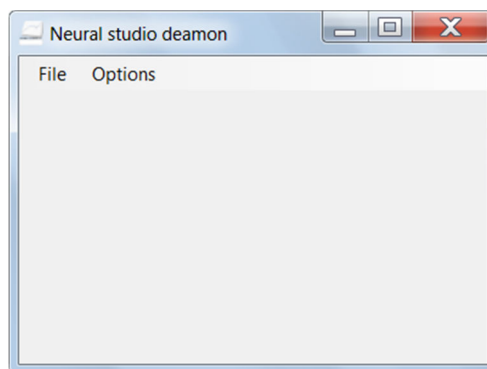
Na obrázku (Obr. 66) je znázorněn test na celém souboru dat. Černou čarou je vykreslena požadovaná odezva sítě a červenou čarou skutečná odezva sítě.



Obr. 66. Výsledek predikce na celém souboru dat.

5.9 Neural studio 1.0 Daemon

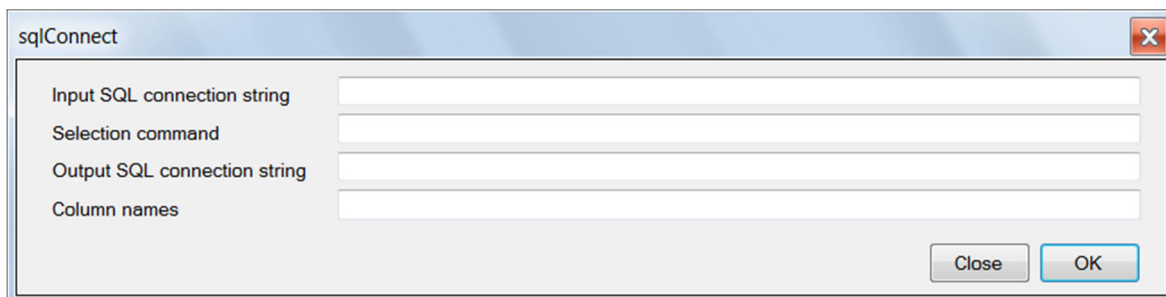
Tato aplikace (Obr. 67) byla vytvořena pro jednoduché použití neuronové sítě v praktických úlohách a umožňuje neuronovou síť vytvořenou a naučenou ve vývojovém prostředí Neural studio 1.0 přenést do této jednoduché aplikace, která může být spuštěna například na vzdáleném serveru a poskytovat tuto neuronovou síť k výpočtům.



Obr. 67. Neural studio 1.0 Daemon.

Postup použití aplikace Neural studio 1.0 Daemon:

1. Naučenou neuronovou síť uložíme v programu neural studio 1.0 z menu *tools* volbou *save trained network*.
2. Volbou menu *file* -> *open* v programu Neural studio 1.0 Daemon otevřeme uložený soubor z předešlého kroku.
3. Volbou menu *options* -> *Refresh rate* v programu Neural studio 1.0 Daemon zvolíme periodu (v milisekundách) odečítání dat z vstupní databáze.
4. Vyplníme formulář (Obr. 68) z menu *file* -> *start* v programu Neural studio 1.0 Daemon tak, že do prvního pole vyplníme connection string pro připojení k databázi ze které budeme data odebírat. Do druhého pole vyplníme SQL příkaz, který vrátí poslední řádek z požadované vstupní tabulky v databázi, který musí mít stejný počet sloupců (typu float), jako počet vstupů vytrénované sítě.
Do třetího pole vyplníme connection string pro připojení k databázi do které budeme vyhodnocená data odesílat. Tato databáze musí obsahovat tabulku s názvem „Output“ ve které musí být stejný počet sloupců (typu float) jako výstupů naučené sítě. Do čtvrtého pole vyplníme jména sloupců v tabulce “Output” oddělené čarkami.
5. Volbou OK spustíme.



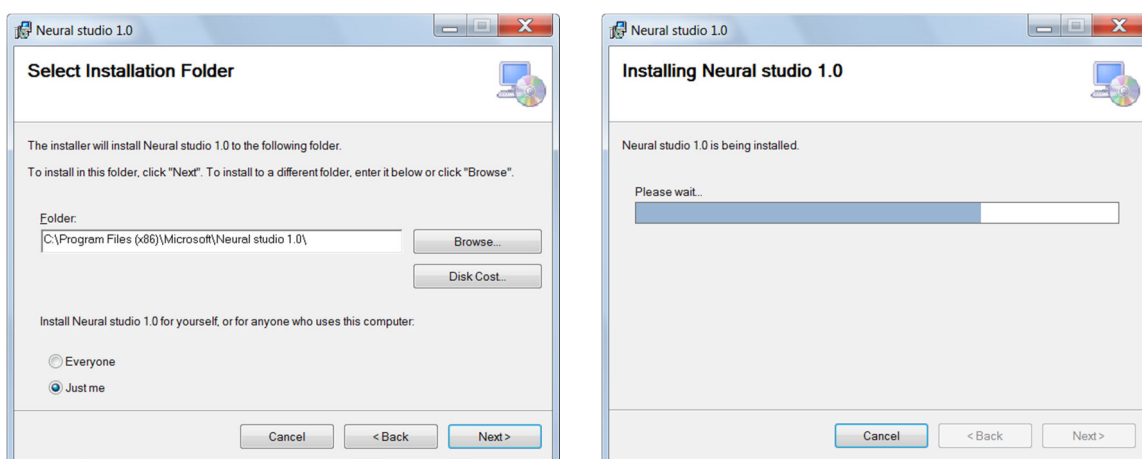
Obr. 68. Sql připojení k serveru - Neural studio 1.0 Deamon.

Spuštění této služby periodicky přečte poslední záznam ze vstupní tabulky a odezvu sítě na tyto data odešle do výstupní tabulky. Což může být užitečné pro aplikace neuronové sítě v reálném čase.

5.10 Instalace

Aplikaci neural studio 1.0 je možno nainstalovat spuštěním souboru Setup.msi z příloženého CD. Instalace aplikace je realizována pomocí „průvodce instalací“, který uživatele provede celou procedurou instalace. Během instalace může být uživatel vyzván k instalaci scházejících doplňků potřebných pro běh aplikace a souhlasu s licenčním ujednáním třetích stran.

Postup instalace je vyobrazen na obrázku (Obr. 69).



Obr. 69. Postup instalace programu neural studio 1.0.

ZÁVĚR

Cílem této diplomové práce bylo vyvinout vizuální vývojové prostředí pro neuronové sítě a jejich aplikace poskytující uživateli možnost rychlého a jednoduchého návrhu neuronových sítí a umožnit jejich použití při řešení reálných úloh.

V rámci této diplomové práce byla popsána problematika neuronových sítí, která by měla čtenáře seznámit se základními principy neuronových sítí, jako je jejich struktura, či možnostmi jejich učení. Dále byl popsán evoluční algoritmus SOMA, použitý v praktické části jako alternativní způsob adaptace synaptických vah neuronové sítě.

Pro úspěšnou realizaci této práce bylo nezbytné pochopení principů neuronových sítí současně s možnostmi a principy programování univerzálního vizuálního vývojového prostředí pro zpracování dat. Výsledkem tohoto spojení je aplikace Neural studio 1.0, která uživateli umožňuje grafický návrh vlastního řešení úlohy, za pomoci funkčních bloků a signálových cest.

Tato aplikace poskytuje uživateli vizuální prostředí pro práci s daty, s možností použití vícevrstvé neuronové sítě s učícím algoritmem backpropagation a s možností učení sítě evolučním algoritmem. Toto prostředí umožňuje použití třinácti různých funkčních bloků pro práci s daty a jeden blok reprezentující vícevrstvou dopřednou neuronovou síť.

V rámci ověření funkčnosti aplikace bylo provedeno základní testování na učení neuronové sítě funkce exkluzivní disjunkce a učení predikce hodnoty časové řady. V obou případech bylo dosaženo uspokojivých výsledků a z toho usuzují, že aplikace, může být použita pro řešení uživatelských úloh.

K aplikaci byl vytvořen automatický instalátor a nápověda, která by měla uživateli usnadnit její použití.

Tato práce je základem pro další vývoj, zejména pro rozšíření funkcí poskytovaných vývojovým prostředím, implementaci dalších typů neuronových sítí a funkčních bloků pro zpracování dat.

ZÁVĚR V ANGLIČTINĚ

The aim of this master thesis was to develop a visual development environment for neural networks and their applications providing to users a quick and simple design of neural networks and allow their use to solve real problems.

In the thesis were described problems of neural networks, which should familiarize readers with the basic principles of neural networks, such as their structure or learning capabilities. Furthermore, the evolutionary algorithm SOMA was described, which is used in practical part as alternative method of adaptation of synaptic weights of neural networks.

For the successful implementation of this work was necessary to understand the principles of neural networks together with the possibilities and principles of programming universal visual environment for data processing. The result is application Neural Studio 1.0, which allows to users custom graphic design solutions, using functional blocks and signal paths.

This application provides to user a visual environment for working with data, with the possibility of using multilayer neural network with a backpropagation learning algorithm, and the possibility to learning by evolutionary algorithm. This environment allows use of thirteen different functional blocks for working with data and one block representing forward multilayer neural network.

The verification of application functionality was performed on the learning simple function (exclusive disjunction), and learning prediction of time series. In both cases were achieved satisfactory results.

For the application were developed automatic installer and help, that should make it easier for user to use.

This work is the basis for further development, particularly for extending the functionality provided by the development environment, the implementation of other types of neural networks and functional blocks for data processing.

SEZNAM POUŽITÉ LITERATURY

- [1] ZELINKA, Ivan. *Umělá inteligence : aneb úvod do neuronových sítí a evolučních algoritmů*. druhé vydání. Zlín : Univerzita Tomáše Bati, Fakulta technologická, 2005. 127 s. ISBN 80-7318-277-7.
- [2] ŠÍMA, Jiří; NERUDA, Roman. *Teoretické otázky neuronových sítí*. Praha : Matfyzpress, 1996. 389 s. ISBN 80-85863-19-9.
- [3] HAMMER, Miloš. *Metody umělé inteligence v diagnostice elektrických strojů*. Praha : BEN - technická literatura, 2009. 400 s. ISBN 978-80-7300-231-2.
- [4] TUČKOVÁ, Jana. *Úvod do teorie a aplikací umělých neuronových sítí*. Praha : České vysoké učení technické v Praze, 2005. 103 s. ISBN 80-01-02800-3.
- [5] ŠNOREK, Miroslav. *Neuronové sítě a neuropočítače*. Praha : České vysoké učení technické v Praze, 2004. 156 s. ISBN 80-01-02549-7.
- [6] POKORNÝ, Miroslav. *Umělá inteligence v modelování a řízení*. Praha : BEN - technická literatura, 1996. 178 s. ISBN 80-901984-4-9.
- [7] BÍLA, Jiří. *Umělá inteligence a neuronové sítě v aplikacích*. Praha : České vysoké učení technické v Praze, 1996. 115 s. ISBN 80-01-01275-1.
- [8] BISKUP, Roman. *Možnosti neuronových sítí*. Praha, 2009. 170 s. Dizertační práce. Česká zemědělská univerzita v Praze.
- [9] KOTRLA, Jakub. *Prostorové mapy pro agenty imitující lidské chování*. Praha, 2009. 75 s. Diplomová práce. Univerzita Karlova v Praze.
- [10] ZELINKA, Martin. *Implementace evolučních algoritmů při řešení problémů globální optimalizace s rozhraním v prostředí Microsoft Excel*. Zlín, 2008. 72 s. Bakalářská práce. Univerzita Tomáše Bati ve Zlíně.
- [11] SLUŠNÝ, Stanislav. *Evoluční učící algoritmy pro dopředné neuronové sítě*. Praha, 2005. 50 s. Diplomová práce. Univerzita Karlova v Praze.
- [12] KVASNIČKA, Vladimír, et al. *Úvod do teórie neurónových sietí* [online]. [s.l.] : [s.n.], [200?] [cit. 2011-04-11]. Dostupné z WWW: <http://ics.upjs.sk/~novotnyr/home/skola/neuronove_siete/nn_ivasnicka/Uvod%20do%20NS.pdf>.
- [13] *StatSoft* [online]. 2011 [cit. 2011-05-1]. STATISTICA Automatizované neuronové sítě Cz. Dostupné z WWW: <<http://www.statsoft.cz/produkty/4-neuronove-site/20-statistica-automatizovane-neuronove-site-cz/detail/>>.

- [14] *C# Corner* [online]. 2006 [cit. 2011-04-25]. C# Artificial Intelligence (AI) Programming: A Basic Object Oriented (OOP) Framework for Neural Networks. Dostupné z WWW: <http://www.c-sharpcorner.com/UploadFile/rmcochran/AI_OOP_NeuralNet06192006090112AM/AI_OOP_NeuralNet.aspx>.
- [15] *The code project* [online]. 2007 [cit. 2010-09-5]. A flexible charting library for .NET. Dostupné z WWW: <<http://www.codeproject.com/KB/graphics/zedgraph.aspx>>.
- [16] *The code project* [online]. 2005 [cit. 2010-08-1]. .NET Shape Control. Dostupné z WWW: <<http://www.codeproject.com/KB/miscctrl/DotNetShapeControl.aspx>>.
- [17] *The code project* [online]. 2007 [cit. 2010-08-15]. C# Line Control with Rotate Option. Dostupné z WWW: <<http://www.codeproject.com/KB/miscctrl/csLineControlWithRotateOp.aspx>>.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

Δ	Rozdíl dvou hodnot.
θ	Práh neuronu
\forall	Matematická značka „pro každé“
<i>Dim</i>	Dimenze.
<i>e</i>	Exponenciální funkce
<i>f</i>	Funkce.
$N_{Výst}$	Počet neuronů výstupní vrstvy
N_{skryt}	Počet neuronů skryté vrstvy
N_{vst}	Počet neuronů vstupní vrstvy
<i>Sign</i>	Znaménková funkce
<i>t</i>	Číslo určující aktuální vzdálenost jedince od jeho výchozí polohy.
<i>Tr</i>	Trénovací množina
w_i	Hodnota i-té synaptické váhy neuronu
x_i	Velikost vstupní hodnoty i-tého vstupu neuronu
X_i^0	i-tý vektor vstupů trénovací množiny
Y_i^*	i-tá hodnota vektoru výstupů neuronové sítě
Y_i^0	i-tý vektor výstupů trénovací množiny
AllToAll	Strategie algoritmu SOMA.
ART	Typ neuronové sítě (Adaptive – Resonance Theory)
fitness	Hodnota účelové funkce.
Leader	Jedinec s nejvíce vyhovující hodnotou účelové funkce.
Migrations	Parametr algoritmu SOMA. Počet migračních kol.
MISO	Systém s mnoha vstupy a jedním výstupem
PathLength	Parametr algoritmu SOMA. Délka cesty jedince.

PopSize	Parametr algoritmu SOMA. Velikost populace.
Prt	Parametr algoritmu SOMA. Perturbace.
SOMA	Samo-Organizující se Migrační Algoritmus.
step	Parametr algoritmu SOMA.
XOR	Funkce exkluzivní disjunkce (eXclusive OR)

SEZNAM OBRÁZKŮ

Obr. 1. Lineárně separabilní problém.	12
Obr. 2. Nelineárně separabilní problém.	13
Obr. 3. Model neuronu.	15
Obr. 4. Schéma vícevrstvé neuronové sítě.	17
Obr. 5. Schéma vícevrstvé sítě.	20
Obr. 6. Schéma adaptační metody backpropagation.	23
Obr. 7. Hopfieldova síť.	23
Obr. 8. Kohonenova síť.	25
Obr. 9. Princip algoritmu SOMA – AllToAll.	29
Obr. 10. Hlavní okno StatSoft STATISTICA.	32
Obr. 11. Vývojové prostředí Neural studio 1.0.	35
Obr. 12. Struktura programu Neural studio 1.0.	36
Obr. 13. Hlavní okno aplikace Neural Studio 1.0.	39
Obr. 14. Hlavní menu aplikace.	40
Obr. 15. Popis panelu nástrojů pro práci s návrhovým prostředím.	41
Obr. 16. Popis panelu funkčních bloků.	41
Obr. 17. Panel záložek pro práci s projektem.	42
Obr. 18. Prohlížeč datových tabulek.	43
Obr. 19. Kontextové menu návrhového plátna.	44
Obr. 20. Kontextové menu bloku.	44
Obr. 21. Dialogové okno neuronové sítě.	45
Obr. 22. Panel funkčních bloků neuronové sítě.	46
Obr. 23. Záložky pro práci s neuronovou sítí.	46
Obr. 24. Záložka view dialogového okna neural network.	47
Obr. 25. Záložka test (funkce $\cos(x) \cdot \sin(3x)$).	48
Obr. 26. Source blok.	49
Obr. 27. Dialogové okno prvku source.	49
Obr. 28. Formulář připojení SQL.	50
Obr. 29. Normalize blok.	50
Obr. 30. Delete columns blok.	50
Obr. 31. Dialogové okno prvku delete columns.	51
Obr. 32. Alternate rows blok.	51

Obr. 33. Dialogové okno alternate rows.	51
Obr. 34. Delete first blok.	52
Obr. 35. Dialogové okno bloku delete first.	52
Obr. 36. Delete last blok.	52
Obr. 37. Average rows.	53
Obr. 38. Split blok.	53
Obr. 39. Dialogové okno bloku split.	53
Obr. 40. Terminator blok.	54
Obr. 41. Neural network blok.	54
Obr. 42. Dialogové okno prvku neural network.	55
Obr. 43. Dialogové okno nastavení skryté vrstvy.	55
Obr. 44. Dialogové okno bloku expected data.	56
Obr. 45. Dialogové okno bloku evolution algorithm.	56
Obr. 46. Scope blok.	57
Obr. 47. Dialogové okno bloku scope.	58
Obr. 48. Scope – Zero-OrderHold zobrazení.	58
Obr. 49. Save to file blok.	59
Obr. 50. Dialogové okno bloku save to file.	59
Obr. 51. Mux blok.	59
Obr. 52. Demux blok.	60
Obr. 53. Input layer blok.	60
Obr. 54. Hidden layer blok.	60
Obr. 55. Output layer blok.	61
Obr. 56. Expected data blok.	61
Obr. 57. Evolution algorithm blok.	61
Obr. 58. Testovací schéma.	64
Obr. 59. Testovací struktura sítě.	64
Obr. 60. Výsledek učení funkce XOR.	65
Obr. 61. Verifikace funkce XOR na souboru dat č. 1.	66
Obr. 62. Verifikace funkce XOR na souboru dat č. 2.	66
Obr. 63. Výsledek predikce na trénovací množině.	67
Obr. 64. Výsledek predikce na celém souboru dat.	68
Obr. 65. Výsledek predikce na trénovací množině.	69

Obr. 66. Výsledek predikce na celém souboru dat.	69
Obr. 67. Neural studio 1.0 Deamon.	70
Obr. 68. Sql připojení k serveru - Neural studio 1.0 Deamon.	71
Obr. 69. Postup instalace programu neural studio 1.0.	71

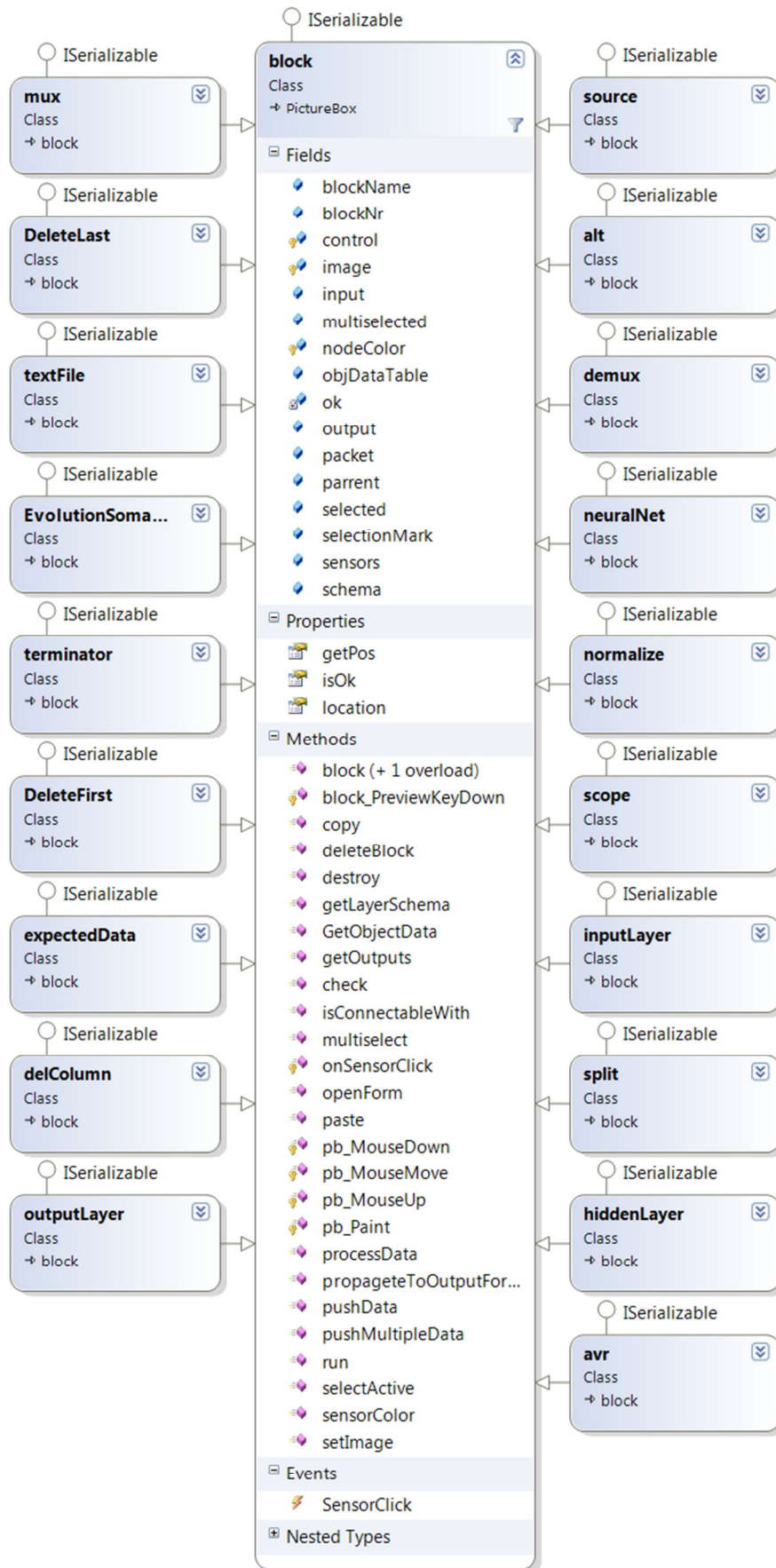
SEZNAM TABULEK

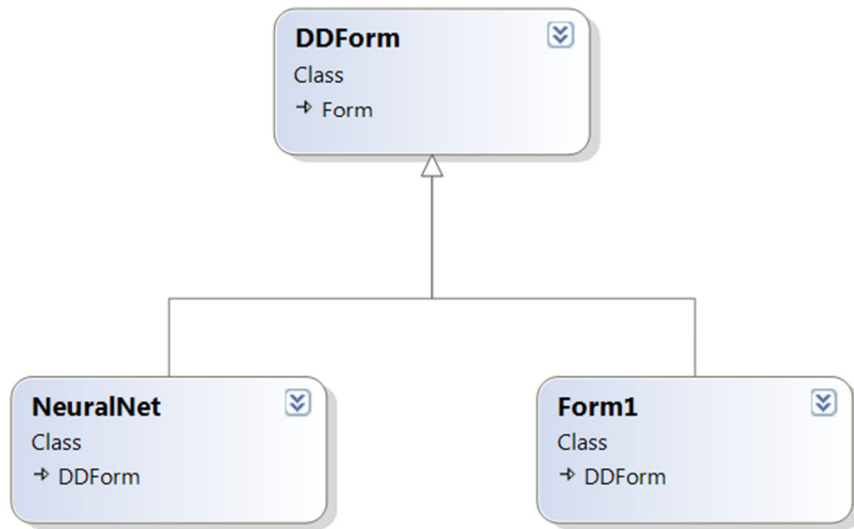
Tab. 1. Lineárně separabilní problém.	12
Tab. 2. Nelineárně separabilní problém (XOR).....	13
Tab. 3. Tabulka vybraných přenosových funkcí.....	15
Tab. 4. Tabulka vybraných přenosových funkcí 1.....	16
Tab. 5. Trénovací množina pro učení XOR.....	64
Tab. 6. Nastavení Neuronové sítě (XOR).....	65
Tab. 7. Nastavení Neuronové sítě (Predikce).	67
Tab. 8. Nastavení Neuronové sítě (Predikce + SOMA).	68
Tab. 9. Nastavení Algoritmu SOMA.....	68

SEZNAM PŘÍLOH

- P I Diagramy vybraných tříd
- P II AlgoritmusSOMA All2All v jazyce C#
- P III Tisk z programu Neural studio 1.0

PŘÍLOHA P I: DIAGRAMY VYBRANÝCH TŘÍD





PŘÍLOHA P II: ALGORITMUS SOMA ALL2ALL V JAZYCE C#

```
double Step, PathLenght, Prt, MinDiv, LeaderCost=Double.MaxValue;
int Migrations, m , PopSize, loop = 0, leaders=0, LeaderIndex, Dim = V_HigherBounds.Length;
double[] V_LowerBounds=new double[Dim/2], V_HigherBounds=new double[Dim/2];
double[] V_CostValues = new double[PopSize], V_TmpInd = new double[Dim], V_StartInd = new
double[Dim];
double[,] M_Pop = new double[PopSize, Dim];
bool[] V_Prt = new bool[Dim];
IOBase.ioCommunication IO;
Random RandomNumber = new Random();

for (int i = 0; i < PopSize; ++i) // Tvorba počáteční populace
{
    for (int j = 0; j < Dim; ++j)
    {
        M_Pop[i, j] = Math.Round(V_LowerBounds[j] + (double)(RandomNumber.NextDouble() *
(V_HigherBounds[j] - V_LowerBounds[j])), IO.rounding);
    }
    double[] V_TMP = new double[Dim];

    for (int q = 0; q < Dim; q++)
    {
        V_TMP[q] = M_Pop[i, q];
    }

    double Cost = IO.clasificate(V_TMP, Dim);

    V_CostValues[i] = Cost;

    if (Cost < LeaderCost)
    {
        LeaderCost = Cost;
        LeaderIndex = i;
    }
}

for (m = 0; m < Migrations; ++m) //migrace
{
    for (int i = 0; i < PopSize; ++i) //cela populace migruje
    {
        for (int n = 0; n < PopSize; n++) //ke všem jedincum
        {
            if (n != i) //kromě sama sebe
            {

                if (i != LeaderIndex)//leader se migrace neúčastní
                {
                    for (int j = 0; j < Dim; ++j)
                    {
                        V_StartInd[j] =M_Pop[i, j];
                    }

                    for (double t = 0; t < PathLenght; t += Step) )
                    {
                        bool zeros = true;

                        while (zeros == true) //Generování perturbačního vektoru
                        {
                            for (int j = 0; j < Dim; ++j)
                            {

                                if (RandomNumber.NextDouble()< Prt)
                                {
                                    V_Prt[j] = true;
                                    zeros = false;
                                }
                                else
                                {
                                    V_Prt[j] = false;
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
for (int j = 0; j < Dim; ++j) //tvorba jedince
{
```


PŘÍLOHA P III: TISK Z PROGRAMU NEURAL STUDIO 1.0

