VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

VÍCEVRSTVÁ NEURONOVÁ SÍŤ

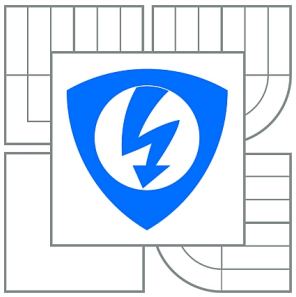
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE PETR KAČER
AUTHOR

BRNO 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION

VÍCEVRSTVÁ NEURONOVÁ SÍŤ

MULTILAYER NEURAL NETWORK

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

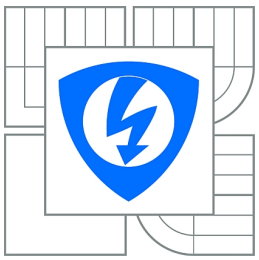
AUTOR PRÁCE — PETR KAČER

AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. VÁCLAV JIRSÍK, CSc.

SUPERVISOR

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Bakalářská práce

bakalářský studijní obor
Automatizační a měřicí technika

Student:	Petr Kačer	***ID:***	138951
Ročník:	3	***Akademický rok:***	2012/2013

NÁZEV TÉMATU:

Vícevrstvá neuronová síť

POKYNY PRO VYPRACOVÁNÍ:

1. Seznamte se s vícevrstvou neuronovou sítí s typem učení backpropagation.
2. Navrhněte a zrealizujte vícevrstvou neuronovou síť s typem učení backpropagation jako výukovou pomůcku vhodnou jak pro názorné předvedení algoritmu učení, tak pro samotné experimentování studentů s neuronovou sítí.
3. V programu vytvořte dostupnou kontextovou nápovědu.
4. Navrhněte a zrealizujte úlohy do laboratorních cvičení předmětu BMPA, které budou demonstrovat vlastnosti naprogramované sítě.
5. Dosažené výsledky zhodnoťte.

DOPORUČENÁ LITERATURA:

Rogers J.: Object-Oriented Neural Networks in C++, Academiic Press, 1997, ISBN 0-12-593115-8
Timothy Masters: Practical Neural Network Rwcipis in C++, Academic Press, 1993, ISBN 0-12-479040-2

Termín zadání:	11.2.2013	***Termín odevzdání:***	27.5.2013

Vedoucí práce: doc. Ing. Václav Jirsík, CSc.
Konzultanti bakalářské práce: Ing. Tomáš Hynčica

doc. Ing. Václav Jirsík, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Bakalářská práce popisuje základy problematiky vícevrstvých neuronových sítí a vysvětluje princip fungování algoritmu backpropagation. Další část práce se zabývá vývojem programu pro učení a testování vícevrstvých neuronových sítí a popisem jeho grafického uživatelského rozhraní a principu ovládání. Poslední část práce je věnována výukovým příkladům a praktickým ukázkám využití vícevrstvé neuronové sítě.

KLÍČOVÁ SLOVA

Neuronová síť, grafické uživatelské rozhraní, backpropagation, design, algoritmus, učení, Qt, vývojové prostředí

ABSTRACT

Bachelor's thesis describes the basics of issue of multilayer neural networks and explains principle of backpropagation algorithm. Next part of thesis is about development of a software for learning and testing multilayer neural networks and describes its graphical user interface. Last part of thesis is dedicated to tutorial examples and practical demonstrations of multilayer neural network usage.

KEYWORDS

Neural network, graphical user interface, backpropagation, design, algorithm, learning, Qt, development environment

KAČER, Petr *Vícevrstvá neuronová síť*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2013. 52 s. Vedoucí práce byl doc. Ing. Václav Jirsík, CSc.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Vícevrstvá neuronová síť“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu bakalářské práce doc. Ing. Václavu Jirsíkovy CSc. a konzultantovi Ing. Tomáši Hynčicovi za jejich cenné rady a připomínky. Dále děkuji své rodině za pomoc při kontrole pravopisu a v neposlední řadě děkuji také všem, kteří se rozhodli si tuto práci přečíst.

Brno ....

(podpis autora)

Obsah

Seznam obrázků

1 ÚVOD

Neuronové sítě jsou mocný nástroj a se vzrůstajícím výkonem dnešní výpočetní techniky, či vývojem čipů hardwarově realizujících neuronové sítě je jejich použití při řešení různých technických problémů velmi aktuální.

Neuronové sítě mohou vykonávat širokou škálu činností. V oboru zpracování signálů se neuronové sítě používají jako filtry šumu nebo filtry pro zvýšení kvality obrazu. Neuronové sítě se velmi často využívají také při předvídání časové řady, například pohybu cen akcií a kurzů měn nebo také při předpovědi počasí, či spotřeby elektrické energie. Pomocí neuronových sítí je také možné s úspěchem klasifikovat data nebo rozpoznávat určité vzory v obraze. Existují však i další možnosti uplatnění neuronových sítí, jsou to například aproximace jinak obtížně matematicky popsatelné funkce, syntéza a rozpoznávání řeči, diagnóza onemocnění na základě zadaných symptomů, řízení, regulace, komprese, dekomprese a šifrování dat [4].

Druhá kapitola vysvětluje základy teorie vícevrstvých neuronových sítí s algoritmem učení backpropagation. Na začátku kapitoly je popsán samotný neuron a jeho přenosové funkce používané ve vícevrstvých neuronových sítích s algoritmem učení backpropagation. Další část kapitoly je věnována topologii vícevrstvé neuronové sítě a několika doporučením k její volbě. V závěru kapitoly je vysvětlen princip fungování algoritmu backpropagation.

Třetí kapitola je výsledkem rešerše a popisuje pětici vybraných programů pro práci s neuronovými sítěmi a poukazuje na jejich hlavní přednosti a nedostatky z hlediska vhodnosti pro výuku, množství funkcí a ergonomie ovládání.

Čtvrtá kapitola řeší hlavní cíl této práce, kterým je vytvoření programového prostředí určeného pro výuku a experimentování s vícevrstvými neuronovými sítěmi a algoritmem učení backpropagation. První část kapitoly se zabývá Qt4 frameworkem a architekturou model-view, jakožto hlavními technologiemi použitými při vývoji. Zbytek kapitoly popisuje samotný program Neural network Creator. První část popisu je věnována jednotlivým modulům / jmenným prostorům a třídám, ze kterých je program sestaven. Ve druhé části popisu je vysvětlena funkce a způsob ovládání programu pomocí jeho grafického uživatelského rozhraní.

Poslední, pátá kapitola, obsahuje osm příkladů pro výuku, vytvořených k doplnění teorie z druhé kapitoly a jako demonstraci možností programu Neural network Creator. První tři příklady prakticky ukazují výpočet dopředného šíření signálu vícevrstvou neuronovou sítí, výpočet zpětného šíření chyby a adaptace vah algoritmem backpropagation a výpočet chyby výstupu neuronové sítě. Zbylých pět příkladů je vypracováno v programu Neural network Creator a jsou na nich předvedeny možnosti využití vícevrstvé neuronové sítě při řešení různých technických problémů.

2 TEORETICKÝ ROZBOR

Tato kapitola vysvětluje základy teorie vícevrstvých neuronových sítí. První část kapitoly se zabývá popisem neuronu, jakožto prvku, ze kterého je složena celá neuronová síť. Vícevrstvá neuronová síť včetně několika doporučení k volbě topologie je popsána v druhé části kapitoly. Poslední část kapitoly je věnována učení neuronové sítě algoritmem backpropagation.

2.1 Neuron

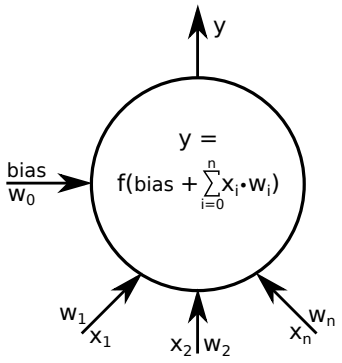
Neuron (obr. 2.1) je základním stavebním kamenem každé neuronové sítě. Každý neuron má vstupy x_1 až x_n opatřených vahami w_1 až w_n. Váhy vstupů mohou být i záporné, čímž je vyjádřen jejich inhibiční charakter [10].

Vážená suma vstupních hodnot představuje vnitřní potenciál neuronu, který lze vypočítat podle vztahu

$$y_{in} = bias + \sum_{i=0}^{n} x_i \cdot w_i \tag{2.1}$$

kde x_1 až x_n je vektor vstupů neuronu, w_1 až w_n je vektor vah těchto vstupů a bias je konstantní vstup neuronu [10].

Pokud k neuronu přidáme vstup o stálé hodnotě 1, můžeme bias včlenit do vztahu jako hodnotu váhy tohoto vstupu. Konečný výstup neuronu potom dostaneme dosazením vnitřního potenciálu neuronu do jeho přenosové funkce [10].

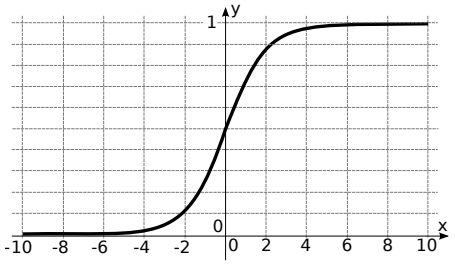


Obr. 2.1: Neuron

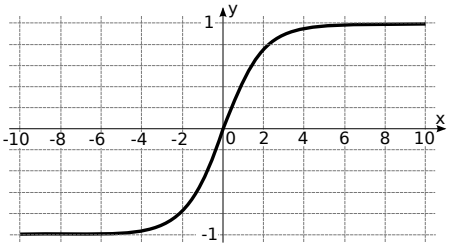
Nejdůležitějšími přenosovými (aktivačními) funkcemi v neuronových sítích typu backpropagation jsou unipolární a bipolární sigmoidní funkce (obr. 2.2). I když se většinou předpokládá použití stejné přenosové funkce pro celou neuronovou síť nebo alespoň její jednotlivé vrstvy, není to pravidlem a každý neuron teoreticky může mít jinou přenosovou funkci [4].

Uvedené sigmoidní aktivační funkce jsou užitečné u neuronových sítí typu backpropagation zejména proto, že je lze snadno zderivovat a dosazení do již zderivované funkce snižuje výpočetní náročnost algoritmu během učení neuronové sítě. Strmost průběhu funkce je dána hodnotou parametru s [4].

Unipolární sigmoidní funkce se používá v sítích, u kterých požadujeme binární výstup, nebo výstup v intervalu 0 až 1. Bipolární sigmoidní funkce si je velmi blízká s unipolární a používá se u sítí s požadovaným výstupem v rozsahu -1 až 1 [4].



$$f(x) = \frac{1}{1 + e^{-sx}} \qquad (2.2)$$



$$f(x) = \frac{1 - e^{-sx}}{1 + e^{-sx}} \qquad (2.3)$$

Obr. 2.2: Sigmoidní aktivační funkce neuronu [10]

2.2 Topologie

Vícevrstvá neuronová síť (obr. 2.3) je síť s jednou nebo více skrytými vrstvami neuronů umístěných mezi vstupní a výstupní vrstvou. Neuron vstupní vrstvy posílá svůj vstup ke všem neuronům vnitřní vrstvy. Výstupy vnitřní vrstvy jsou přivedeny na vstupy každého neuronu vyšší vrstvy a jsou vynásobeny příslušnými vahami [4].

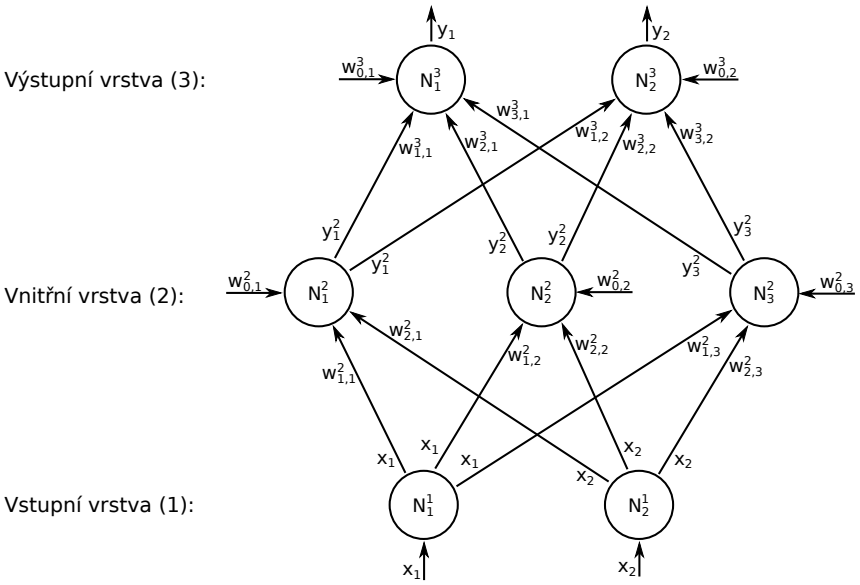
Výstup k-tého neuronu nacházejícího se v n-té skryté nebo výstupní vrstvě vícevrstvé neuronové sítě lze vypočítat podle vzorce

$$y_k^n = f\left(w_{0,k}^n + \sum_{i=1}^{m} y_i^{n-1} \cdot w_{i,k}^n\right) \qquad (2.4)$$

kde $f(x)$ je přenosová funkce neuronu, $w_{0,k}^n$ je bias neuronu a m je počet vah neuronu, kterému taktéž odpovídá počet výstupů nižší vrstvy sítě [10].

Vícevrstvá neuronová síť má potenciál řešit nelineárnější problémy než jednovrstvá, učení takové sítě je ale mnohem složitější. V některých případech je učení vícevrstvé sítě úspěšnější, protože je možné, že se jedná o problém, který nedokáže jednovrstvá síť správně vyřešit [4].

K určení počtu vrstev vícevrstvé neuronové sítě a počtu neuronů v těchto vrstvách neexistuje žádný pevný vztah, topologii neuronové sítě je nutno určit experimentálně. Přesto však existuje několik pravidel a doporučení, která mohou usnadnit volbu její topologie [5].



Obr. 2.3: Značení neuronů, vah, vstupů a výstupů v neuronové síti

Pro řešení daného problému téměř vždy stačí neuronová síť s jednou vnitřní vrstvou. Topologie se dvěma vnitřními vrstvami může být potřebná, když se neuronová síť má naučit funkci mající nespojitosti. Pro použití více jak dvou vnitřních vrstev neexistuje žádný teoretický důvod. Je zde však možnost, že problém bude efektivněji řešitelný pomocí neuronové sítě s více vrstvami o menším počtu neuronů, než pomocí sítě s menším počtem vrstev s neprakticky velkým počtem neuronů [5].

Po zvolení počtu vrstev je třeba zvolit počet neuronů v těchto vrstvách. Volba správného počtu neuronu je velmi důležitá. Při použití malého počtu neuronů nemá neuronová síť kapacitu k naučení daného problému. Naopak při použití příliš velkého počtu neuronů dochází k výraznému prodloužení doby učení a také může nastat problém zvaný přeučení neuronové sítě [5].

K přeučení neuronové sítě dochází, když má tato síť příliš mnoho prostředků ke zpracování informací. Je to stav, kdy se neuronová síť příliš přesně naučí množinu trénovacích dat a to včetně jejích náhodných chyb nebo šumu a ztrácí schopnost generalizace. Přeučená neuronová síť dosahuje výborných výsledků s trénovacími daty. Při použití v reálné aplikaci nebo při práci s s testovací množinou dat jsou však výsledky velmi špatné [5].

Určit počet neuronů můžeme například tak, že začneme s počtem neuronů, který je příliš malý. Pokud tento počet nelze určit, začneme se dvěma neurony. Dále si určíme způsob výpočtu chyby sítě, což je kritérium posuzující, jak dobře je neuronová síť naučená. Potom postupně zvyšujeme počet neuronů a znovu trénujeme a testujeme síť až do doby, než chyba sítě klesne pod přijatelnou mez, nebo už nedochází k žádnému zlepšení [5].

2.3 Učení

Učící algoritmus backpropagation je nejpoužívanějším algoritmem v oblasti učení neuronových sítí (přibližně 80% všech aplikací). Algoritmus lze rozdělit do tří hlavních částí, kterými jsou dopředné šíření vstupního signálu, zpětné šíření chyby a aktualizace váhových hodnot vstupů neuronů. V praxi se tyto části cyklicky opakují, dokud není dosaženo dostatečně malé chyby sítě, mezního počtu iterací nebo jiného kritéria pro zastavení procesu učení [10].

Celkovou chybu neuronové sítě je možné vypočítat, pokud známe skutečné a požadované hodnoty jejích výstupů. Pro hodnotu této chyby potom platí

$$E = \sum_{i=0}^{q} \frac{1}{2} \sum_{k=0}^{n} (y_{i,k} - t_{i,k})^2 \tag{2.5}$$

kde q je počet vzorů trénovací množiny dat, n je počet výstupů neuronové sítě, $y_{i,k}$ je reálný výstup neuronové sítě a $t_{i,k}$ je požadovaná hodnota pro daný výstup neuronové sítě uvedená v trénovacím vzoru [10].

Prvním krokem k adaptaci vah je dopředné šíření signálu přivedeného na vstup neuronové sítě. *"Během dopředného šíření signálu obdrží každý neuron ve vstupní vrstvě vstupní signál a zprostředkuje jeho přenos ke všem neuronům vyšší (vnitřní) vrstvy. Každý neuron ve vnitřní vrstvě opět vypočítá svůj výstup a pošle ho na vstup další vrstvy. Pokud je další vrstva výstupní, tak je její výstup zároveň výstupem neuronové sítě po předložení vstupního vzoru"* ([10], strana 36).

Poté jsou pro každý trénovací vzor porovnány vypočtené hodnoty výstupů s požadovanými a na základě rozdílu těchto hodnot je vypočten faktor δ_k $(k = 1 \dots m)$, který reprezentuje část chyby neuronové sítě šířené z daného neuronu ke všem neuronům nižší vrstvy. Úprava váhových hodnot vstupů neuronů závisí na chybovém faktoru δ_k a hodnotách výstupů neuronů nižší vrstvy [10].

Chybový faktor δ_k pro k-tý neuron výstupní vrstvy n neuronové sítě můžeme vypočítat podle vztahu

$$\delta_k^n = (t_k - y_k) \cdot f' \left(w_{0,k}^n + \sum_{i=1}^{m} y_i^{n-1} \cdot w_{i,k}^n \right) \tag{2.6}$$

kde t_k je požadovaný výstup neuronové sítě, y_k je skutečný výstup neuronové sítě, $f'(x)$ je derivace přenosové funkce neuronu a m je počet výstupů nižší vrstvy [10].

Chybový faktor δ_k pro k-tý neuron n-té vnitřní vrstvy neuronové sítě lze vypočítat podle vztahu

$$\delta_k^n = \sum_{i=1}^{q} (\delta_i^{n+1} \cdot w_{k,i}^{n+1}) \cdot f' \left(w_{0,k}^n + \sum_{i=1}^{m} (y_i^{n-1} \cdot w_{i,k}^n) \right) \tag{2.7}$$

kde q je počet neuronů vyšší vrstvy, δ_i^{n+1} je chybový faktor i-tého neuronu ve vyšší vrstvě, $w_{k,i}^{n+1}$ je váha spoje mezi k-tým neuronem n-té vrstvy a i-tým neuronem vyšší

vrstvy, $f'(x)$ je derivace přenosové funkce neuronu a m je počet výstupů nižší vrstvy, tj. počet vstupů a vah k-tého neuronu n-té vrstvy [10].

Posledním krokem prováděným v rámci jedné iterace algoritmu je samotná adaptace vah neuronů. Pro hodnotu změny biasu k-tého neuronu v n-té vrstvě platí

$$w_{0,k}^n = \alpha \cdot \delta_k^n \tag{2.8}$$

kde α je koeficient učení, který ovlivňuje rychlost učení neuronové sítě a δ_k^n je vypočtený chybový faktor neuronu [10].

Hodnotu změny váhy spoje mezi k-tým neuronem n-té vrstvy a i-tým neuronem vrstvy předchozí vypočítáme podle vzorce

$$w_{i,k}^n = \alpha \cdot \delta_k^n \cdot y_i^{n-1} \tag{2.9}$$

kde α je koeficient učení neuronové sítě, δ_k^n je vypočtený chybový faktor neuronu a y_i^{n-1} je výstup i-tého neuronu nižší vrstvy [10].

"Cílem adaptace je minimalizace chyby sítě ve váhovém prostoru. Vzhledem k tomu, že chyba sítě přímo závisí na komplikované nelineární složené funkci vícevrstvé sítě, představuje tento cíl netriviální optimalizační problém. Pro jeho řešení se v základním modelu používá nejjednodušší varianta gradientní metody, která vyžaduje diferencovatelnost chybové funkce" ([10], strana 37).

Aktivační funkce neuronu v neuronových sítích používajících algoritmus adaptace vah backpropagation musí splňovat následující požadavky: musí být spojitá, diferencovatelná a monotóně neklesající [10].

K naučení neuronové sítě je kromě patřičného algoritmu učení nutné mít také množinu trénovacích dat. Každý vzor trénovací množiny popisuje požadovaný výstup neuronové sítě při daném vstupu. Za trénovací množinu T můžeme považovat množinu prvků definovaných dvojicemi vektorů vstupu a výstupu

$$T = \{\{S_1, T_1\}\{S_2, T_2\} \ ... \ \{S_q, T_q\}\} \tag{2.10}$$

$$S_i = [s_1 \ s_2 \ s_3 \ ... \ s_n], \ T_i = [t_1 \ t_2 \ t_3 \ ... \ t_m] \tag{2.11}$$

kde T je množina trénovacích dat, S_i je vektor vstupů neuronové sítě, T_i je vektor požadovaných výstupů neuronové sítě, q je počet vzorů trénovací množiny, n je počet vstupů neuronové sítě a m je počet výstupů neuronové sítě [10].

3 PŘEHLED PROGRAMŮ PRO PRÁCI S NEURONOVÝMI SÍTĚMI

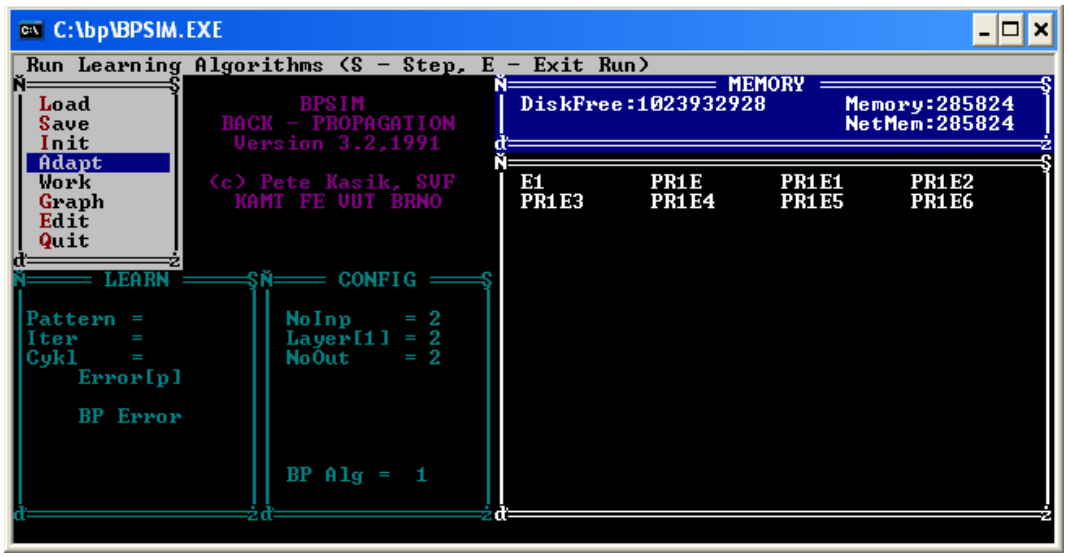
Tato kapitola je výsledkem rešerše a popisuje pětici vybraných programů, zabývajících se problematikou neuronových sítí a umožňujících práci s vícevrstvou neuronovou sítí a poukazuje na jejich hlavní přednosti a nedostatky z hlediska vhodnosti pro výuku, množství funkcí a ergonomie ovládání. Informace získané v průběhu rešerše byly využity při vývoji programu Neural network Creator.

3.1 Program BPSIM

BPSIM (obr. 3.1) je program pro operační systém MS-DOS vytvořený v roce 1991 v rámci diplomové práce na Fakultě elektrotechniky na VUT. Program umožňuje vytvoření neuronové sítě s požadovanou topologií a její inicializaci náhodnými vahami nebo vahami načtenými ze souboru. Množiny trénovacích dat jsou editovány v textové formě s použitím programu Norton Editor.

Po vytvoření trénovací množiny dat lze spustit učení neuronové sítě s použitím některého z nabízených algoritmů, mezi kterými se nachází i algoritmus backpropagation. V průběhu učení je vypisována chyba sítě, kterou je možno po skončení učení zobrazit i graficky, počet cyklů algoritmu a další informace. Program také umožňuje krokování po jednotlivých cyklech algoritmu učení, tj. po jednotlivých vzorech trénovací množiny dat a testování neuronové sítě její aktivací.

Program je svým provedením a funkčností vhodný pro použití ve výuce, nicméně na dnešních strojích není jeho běh optimální a lze o něm říci, že pomalu zastarává.

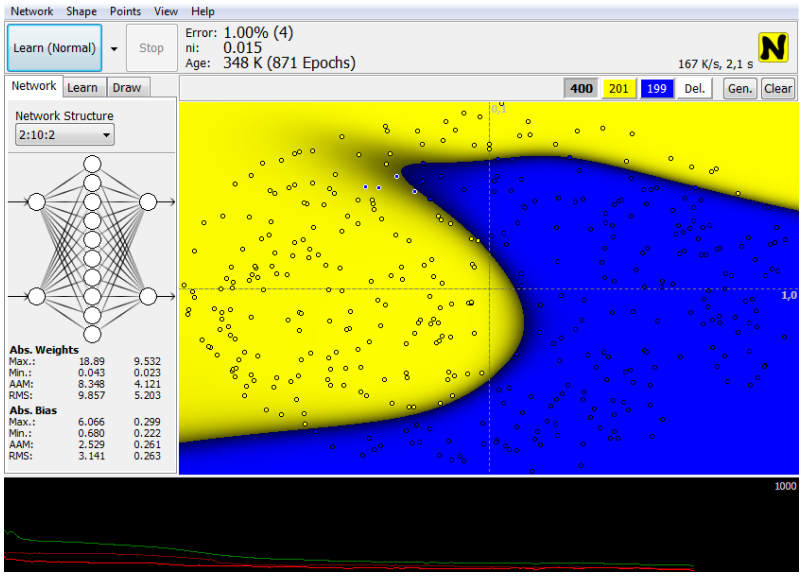


Obr. 3.1: Program BPSIM

Sada programů, jejíž součástí jsou programy BPSIM a Norton Editor obsahuje také program NET_6. Program NET_6 obsahuje funkce pro vykreslování klasifikačních diagramů. Nejprve je nutné vybrat naučenou neuronovou síť. Dále je možné zvolit množinu bodů, které budou následně zobrazeny v klasifikačním diagramu. Klasifikační diagram je možné vykreslit s použitím několika klasifikačních funkcí jako spojité oblasti nebo jen jako jejich hranice.

3.2 Program Sharky Neural Network

Sharky Neural Network (obr. 3.2) je program určený k výuce a experimentování s vícevrstvou neuronovou sítí s algoritmem učení backpropagation, zaměřený na klasifikaci a spustitelný pod operačním systémem Microsoft Windows.



Obr. 3.2: Program Sharky Neural Network

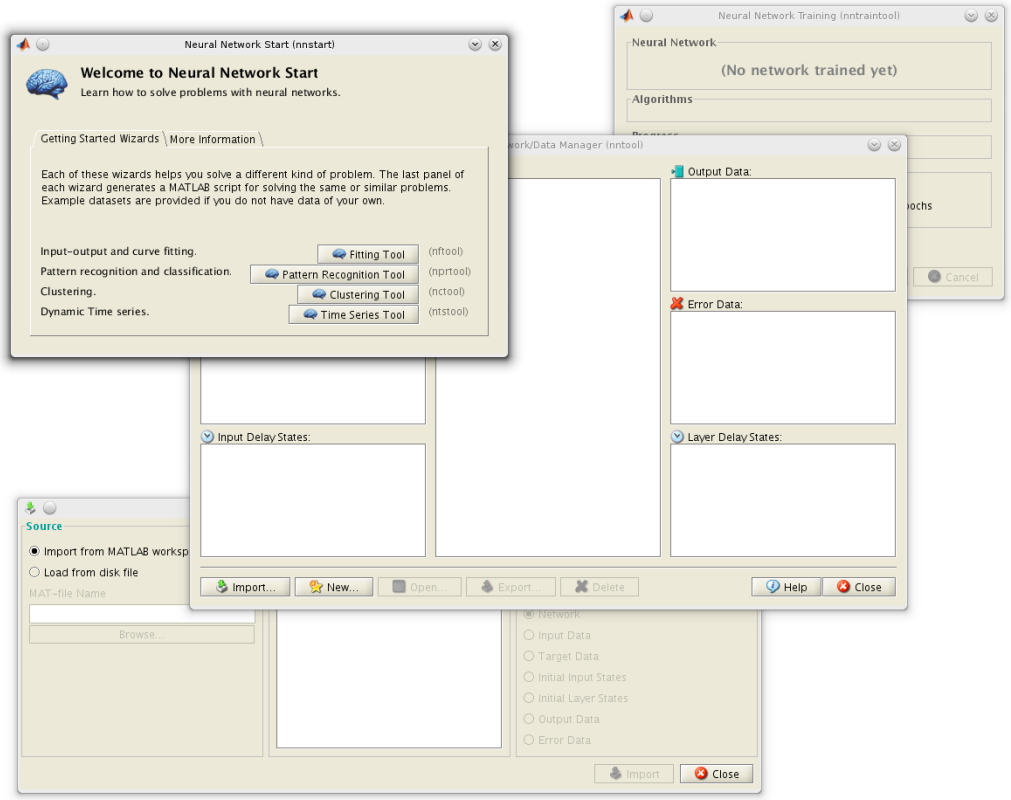
V programu lze vybrat některou z nabízených topologií, pokaždé však pouze síť se dvěma vstupy a dvěma výstupy. Trénovací množinou jsou body patřící do první nebo druhé třídy dat. Body trénovací množiny je možno načíst ze souboru nebo vygenerovat s některým z předdefinovaných rozložení tříd do prostoru. Body jsou zobrazeny v klasifikačním diagramu, kde je lze editovat pomocí myši.

V průběhu učení, které lze spustit pomocí tlačítka *Learn*, je překreslován klasifikační diagram, což nám dovoluje pozorovat změny a pohyby hranic mezi klasifikovanými třídami. V dolní části obrazovky se navíc nachází grafické zobrazení učící křivky neuronové sítě.

Omezující vlastností programu je jeho striktní zaměření na klasifikaci neuronovou sítí pouze se dvěma výstupy (datovými třídami).

3.3 Neural Network Toolbox v Matlabu

Neural Network Toolbox (obr. 3.3) je sadou aplikací nabízející funkce pro modelování komplexních nelineárních systémů, které jsou obtížně analyticky modelovatelné. Neural Network Toolbox umožňuje design, trénování, vizualizaci a simulaci neuronových sítí a obsahuje nástroje pro vytváření neuronových sítí určených pro klasifikaci dat, předvídání časové řady, modelování dynamických systémů, řízení a další aplikace [6].



Obr. 3.3: Neural Network Toolbox v Matlabu

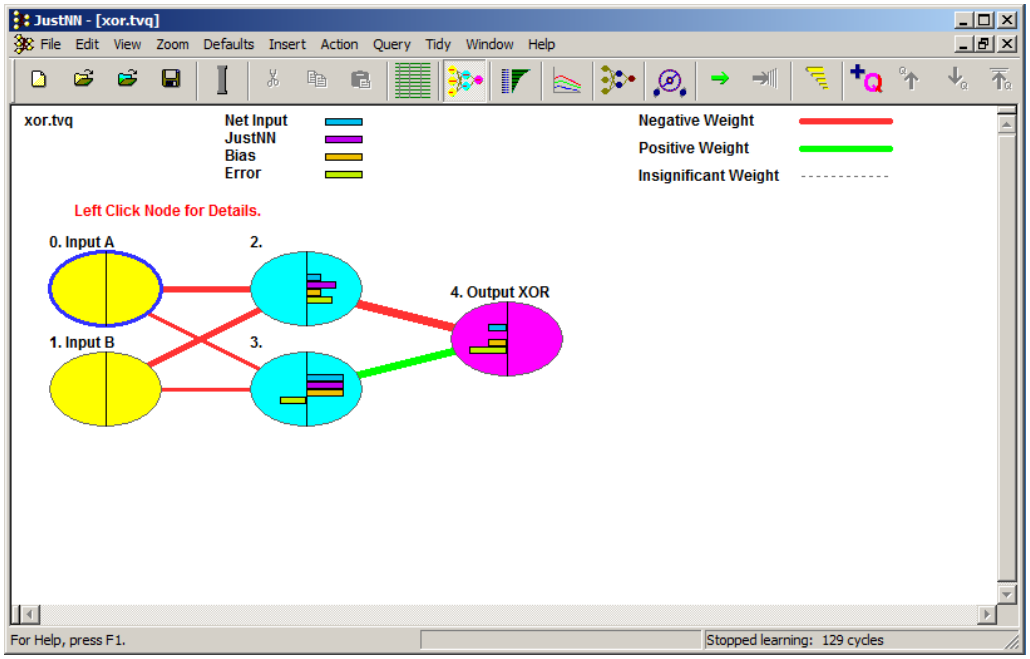
Grafické uživatelské rozhraní toolboxu je řešeno formou dialogových oken. Z úvodního okna pojmenovaného Neural Network Start lze otevřít další dialogová okna umožňující vytvoření neuronové sítě vykonávající určitou specifickou činnost. Správa vytvořených neuronových sítí a použitých datových množin je realizována pomocí okna Neural Network / Data manager (nntool).

Mezi výhody tohoto toolboxu patří množství funkcí a použitelnost v široké škále aplikací, paralelizace a akcelerace výpočtů, možnost použití neuronové sítě v simulinku nebo i možnost použití ostatních funkcí matlabu.

Nevýhodou toolboxu je nutnost znalosti práce s programem Matlab a mírná nepřehlednost způsobená roztříštěním do většího množství dialogových oken.

3.4 Program JustNN

JustNN (obr. 3.4) je freeware program pro práci s vícevrstvými neuronovými sítěmi, určený pro operační systém Windows. Většina funkcí programu je dostupná přes jeho toolbar, pomocí kterého lze také přepínat editaci datových množin, zobrazení topologie neuronové sítě, obrazovkou obsahující údaje o důležitosti vstupů a obrazovkou učení, zobrazující nastavení učení a učící křivku neuronové sítě.



Obr. 3.4: Program JustNN

Editor datových množin je realizován tabulkou, do které je navíc možné importovat data z textového souboru, csv souboru, tabulky v programu Excel a z bitmapového obrázku.

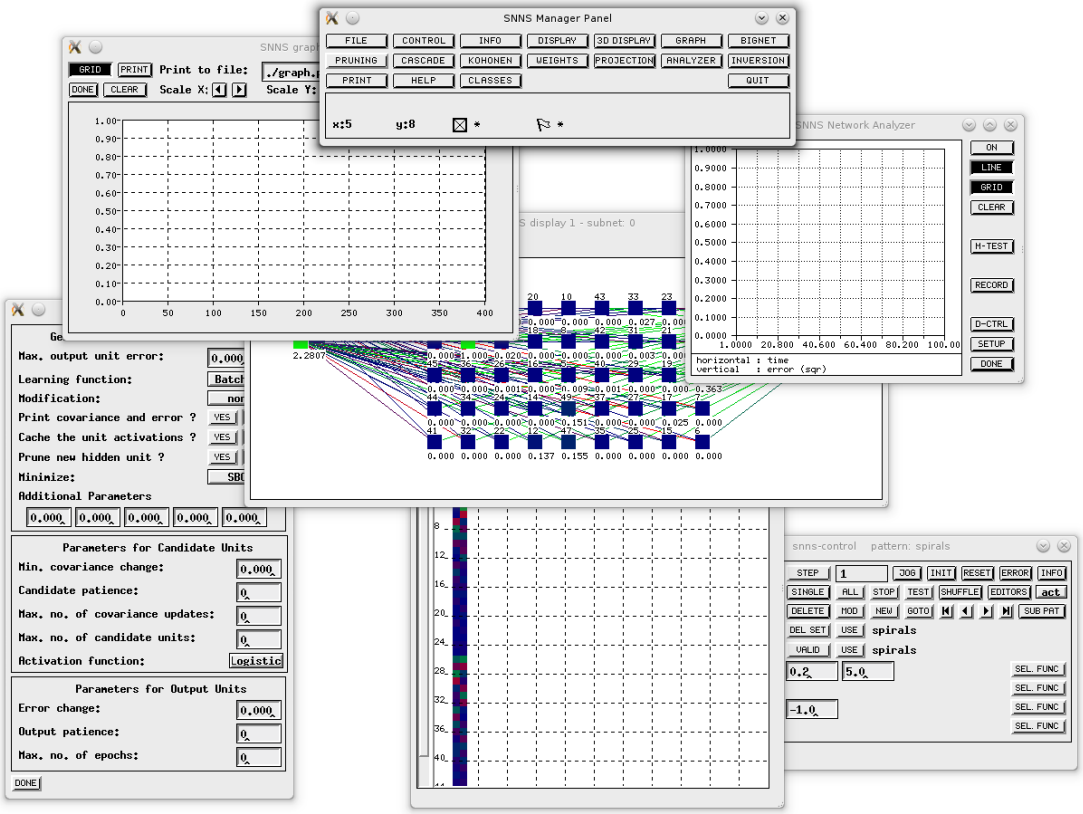
V obrazovce, zobrazující topologii neuronové sítě jsou zakresleny jednotlivé neurony včetně vah, kterými jsou spojeny. Kladné váhy jsou vykresleny zelenou barvou a záporné váhy jsou zakresleny barvou červenou. Konkrétní hodnota váhy je znázorněna tloušťkou čáry.

Další obrazovka zobrazuje důležitosti vstupů neuronové sítě. Důležitost vstupu je vypočítána jako suma absolutních hodnot vah mezi vstupním neuronem a neurony první skryté vrstvy [8].

Výhodou tohoto programu je bezesporu jednoduchost jeho ovládání, při zachování dostatečného množství funkcí. Za nevýhodu lze považovat absenci grafických výstupů, jako je například klasifikační diagram a grafické zobrazení hodnot jednoho výstupu neuronové sítě.

3.5 Stuttgart Neural Network Simulator

Stuttgart Neural Network Simulator (obr. 3.5) je simulátor neuronových sítí vyvinutý na Institutu pro Paralelní a Distribuované systémy na Universitě ve Štutgartu existující od roku 1989. Program je napsán v ANSI-C a jeho grafická část je vytvořena s použitím softwaru X Window System [9].



Obr. 3.5: Stuttgart Neural Network Simulator

Program se skládá ze čtyř hlavních komponent: jádro simulátoru, grafické uživatelské rozhraní, dávkový simulátor a kompilátor neuronových sítí. Jádro simulátoru pracuje s vnitřními datovými strukturami neuronových sítí a provádí s nimi veškeré operace. Grafické uživatelské rozhraní pracující nad jádrem simulátoru, poskytuje grafické znázornění neuronových sítí a umožňuje ovládání běžící simulace [9].

Program kromě několika variant implementace vícevrstvé neuronové sítě a algoritmu backpropagation obsahuje velké množství jiných typů neuronových sítí a jejich algoritmů učení. Program také nabízí širokou škálu možností analýzy neuronových sítí včetně grafických výstupů.

Mezi největší nevýhody programu lze jednoznačně zařadit ne příliš intuitiví a dnes již zastaralé grafické rozhraní programu.

3.6 Porovnání uvedených programů

Až dosud se pro výuku používal program BPSIM, který je napsán pro operační systém MS-DOS, proto jeho výkon na dnešních strojích již není optimální a program navíc permanentně vytěžuje jedno jádro procesoru. Uživatelské rozhraní programu je i přes svou zastaralost dobře a přehledně zpracované a spolu s celkovou funkčností programu stále tvoří prostředí poměrně vhodné pro výuku neuronových sítí.

Z ostatních uvedených programů je pro výuku a experimentování nejvhodnější program Sharky Neural Network, jehož použití je však limitováno tím, že je zaměřen pouze na klasifikaci dat do dvou tříd. Dalším vhodným kandidátem pro použití ve výuce je program JustNN. Program obsahuje dostatečné množství funkcí a je dobře ovladatelný, postrádá však grafické znázornění výstupu neuronové sítě, jakým je například klasifikační diagram, či zobrazení jednoho výstupu neuronové sítě. Programy Stuttgart Neural Network Simulator a Neural Network toolbox v Matlabu implementují velké množství funkcí, které je však pro výuku nadbytečné. Ovládání těchto programů je díky tomu a také díky jejich roztříštěnosti do velkého množství dialogových oken mnohem složitější.

4 PROGRAM NEURAL NETWORK CREATOR

Tato kapitola je věnována programu Neural network Creator, který vznikl v rámci této bakalářské práce. První část kapitoly popisuje Qt4 framework, jakožto hlavní technologii použitou při vývoji programu. Druhá část vysvětluje architekturu model-view, na které je postaven jak Qt4 framework, tak i samotný program Neural network Creator. Třetí část kapitoly řeší strukturu programu z programátorského hlediska. Poslední kapitoly je věnována popisu grafického uživatelského rozhraní programu.

4.1 Použité technologie

Program je napsán v jazyce C++ s použitím Qt4 frameworku a vývojového prostředí QtCreator. Qt4 je multiplatformní framework pro tvorbu aplikací s grafickým uživatelským rozhraním. Qt4 framwork je napsán v jazyce C++, ale je použitelný i v jiných jazycích, jako jsou například Java, Python, Ruby a další. Mezi výhody tohoto frameworku patří již výše zmíněná přenositelnost mezi většinou používaných platforem (Windows, Linux, Mac a další), velmi dobře zpracovaná dokumentace, vysoká produktivita práce, skvělé nástroje pro vývoj (QtCreator), otevřenost zdrojových kódů nebo i jeho oblíbenost u programátorů.

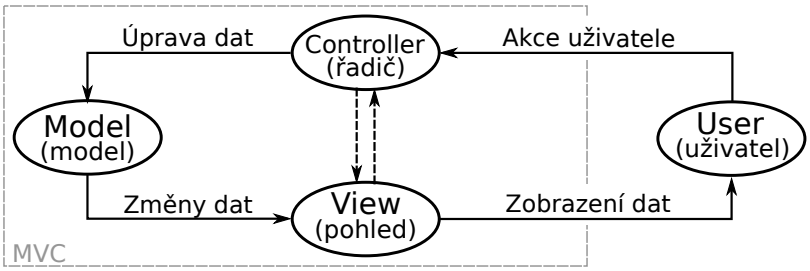
Qt4 framework se skládá z modulů, které lze rozdělit do čtyř skupin. Do první skupiny patří moduly pro všeobecný vývoj softwaru, druhá skupina obsahuje moduly pro práci s nástroji Qt4 frameworku a třetí a čtvrtá skupina obsahují dodatečné, platformně závislé moduly pro vývojáře pro Windows a pro Linux [1].

Pro vývoj programu byly použity moduly QtCore, QtGui, QtXml, QWebKit, QtOpenGL a QtTest. Modul QtCore obsahuje třídy obstarávající základní programu, která nesouvisí s jeho grafickým rozhraním. Funkčnost grafického rozhraní obstarává modul QtGui, který je ve své podstatě rozšířením modulu QtCore o potřebné třídy. Modul QtXml je použit při načítání a ukládání dat projektu do xml souborů. Modul QWebKit obsahuje třídy pro zobrazování a editaci webového obsahu a v programu je použit k zobrazení nápovědy, vytvořené ve formě webové stránky. Modul QtOpenGL je skupina tříd obstarávajících podporu OpenGL v Qt4 aplikacích. Poslední použitý modul je modul QtTest obsahující nástroje pro vývoj unit testů programu, včetně testů jeho grafického uživatelského rozhraní. Při vývoji nebyl použit žádný modul závislý na konkrétní platformě [1].

Qt4 framework je použit při vývoji mnoha známých projektů, jako jsou například multimediální přehrávač VLC, linuxové desktopové prostředí KDE 4, komunikační program Skype, 3D kreslící program Autodesk Maya, Google Earth, program na kreslení plošných spojů Eagle, virtualizační program VirtualBox a dalších.

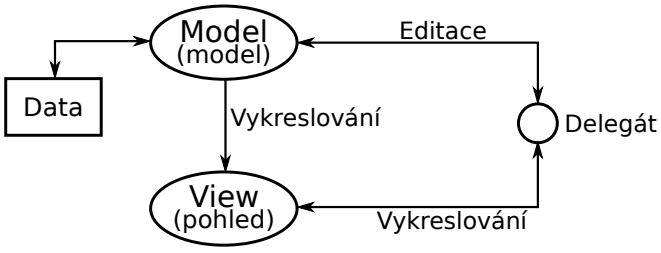
4.2 Architektura model-view

Model-view-controller je architektura často používaná pro tvorbu programů s grafickým uživatelským rozhraním. Architektura obsahuje tři hlavní objekty (obr. 4.1). Prvním objektem je model (aplikační objekt), druhým objektem je pohled (view), který se stará o grafickou reprezentaci dat uživateli a posledním objektem je řadič (controller), který říká, jakým způsobem reaguje uživatelské rozhraní na akce vykonané uživatelem [3].



Obr. 4.1: Architektura model-view-controller [3][2]

Pokud zkombinujeme objekty view (pohled) a controller (řadič), vznikne architektura model-view (obr. 4.2). V této architektuře jsou data stále oddělena od způsobu jejich reprezentace uživateli. Získáme však jednodušší framework založený na stejném principu jako model-view-controller. Toto oddělení dat od způsobu jejich reprezentace nám umožňuje vytvořit nové způsoby zobrazení a editace dat, aniž bychom museli měnit datové struktury modelů [7].



Obr. 4.2: Architektura model-view [7]

V Qt4 frameworku model komunikuje se zdrojem dat a poskytuje rozhraní pro ostatní komponenty architektury. Způsob komunikace modelu se zdrojem dat závisí na typu zdroje dat a konkrétní implementaci modelu. Pohled získává z modelu indexy, které slouží jako reference na jednotlivé datové položky uložené ve zdroji

dat. Předáváním těchto indexů do modelu může pohled získat data ze zdroje dat. Ve standardních objektech typu pohled, delegát vykresluje položky dat. Pokud dojde ke změně datové položky, tak delegát přímo upozorní model s použitím indexů modelu a model změní příslušná data ve zdroji dat [3].

Obecně tedy lze říci, že model-view třídy nacházející se v Qt4 frameworku lze rozdělit do tří skupin: na modely, pohledy a delegáty. Pro každý z těchto objektů existují abstraktní třídy obsahující společné rozhraní a v některých případech také základní implementaci funkčnosti. U abstraktních tříd se k dosažení plné škály funkčnosti požadované ostatními komponentami předpokládá využití mechanismu dědičnosti. Modely, pohledy a delegáti mezi sebou komunikují pomocí systému signálů a slotů [3].

4.3 Struktura programu

Program je rozdělen do několika částí (modulů), z nich každý náleží do svého jmenného prostoru. Názvy jmenných prostorů byly voleny s ohledem na činnost modulu a třídy programu byly do těchto jmenných prostorů umístěny na základě povahy činnosti, kterou vykonávají.

Prvním jmenným prostorem je jmenný prostor *Application*. V tomto jmenném prostoru se nacházejí všechny třídy grafického uživatelského rozhraní s výjimkou dialogových oken. Patři sem třída hlavního okna *(MainWindow)* a třídy hlavních editačních widgetů *(DatasetEditWidget, TopologyWidget, LearningWidget, DatasetTestWidget, OutputGraphWidget)* sloužící jako pohledy pro modely položek projektu ze jmenného prostoru *ProjectData* v architektuře model-view, tentokrát na úrovni programu. Dále se zde nacházejí třídy grafických rozhraní uvítací obrazovky a obrazovky nápovědy *(WelcomeWidget, HelpWidget)* a také třídy menších částí grafického uživatelského rozhraní *(LabelButton, LayerEditWidget, NetParamWidget, OpenedListItem, Plot1D, Plot2D, Plot3D a PlotCls)*.

Druhým jmenným prostorem, který se taktéž zabývá grafickým rozhraním je jmenný prostor *Dialog*, ve kterém se nacházejí všechny třídy dialogových oken. Patří mezi ně dialog zobrazený při vytváření nového projektu *(NewProjectDialog)* nebo nové položky projektu *(FileNameDialog)*, dialog určený k editaci nastavení programu *(SettingsDialog)* a dialog zobrazující informace o programu *(About Dialog)*.

Třetí jmenný prostor, nesoucí název *NeuralNetwork* obsahuje třídy řešící problematiku vícevrstvé neuronové sítě a jejího učení. Jsou zde dvě abstraktní třídy *(AbstractMlnNet, AbstractLrnAlg)* sloužící jako rozhraní (interface) pro všechny současné i budoucí implementace vícevrstvé neuronové sítě a jejích algoritmů učení. Dále třída *MlnNetSt*, která je jednovláknovou implementací vícevrstvé neuronové sítě, a třída

BpAlgSt, která je jednovláknovou implementací učícího algoritmu backpropagation. Třída *LrnEngine* se stará o spouštění učícího algoritmu ve vedlejším vlákně, aby bylo možné v průběhu učení vykreslovat učící křivku neuronové sítě a celý program zůstal ovladatelný. Do tohoto jmenného prostoru patří také třída *Dataset*, která slouží jako datová struktura pro ukládání a editaci množin trénovacích či testovacích dat a třída *Neuron* realizující funkčnost jednoho neuronu.

S třídami z jmenného prostoru *NeuralNetwork* pracují datové modely ze jmenného prostoru *ProjectData*. Model *DatasetEditModel* dědí třídu *Dataset* a doplňuje ji o další metody potřebné ke zobrazení a editaci v něm uložených dat. Podobně tak i model *TopologyEditModel* dědí třídu *MlnNetSt*, takže kromě metod pro správu topologie, obsahuje také neuronovou síť samotnou, a to včetně všech vah vstupů, biasů a přenosových funkcí neuronů a jejich strmostí. Další z modelů, *LearningConfigModel*, se stará o nastavení a spouštění procesu učení a kromě nastavených parametrů učení obsahuje také učící křivku neuronové sítě získanou v průběhu učení.

K otestování neuronové sítě pomocí její aktivace a porovnání skutečných výstupů s požadovanými slouží model *DatasetTestModel*. Neuronovou síť lze otestovat také pomocí grafů jejích výstupů nebo pomocí klasifikačního diagramu, k tomu slouží model pojmenovaný *GraphTestModel*. Poslední tři výše uvedené modely (*LearningConfigModel*, *DatasetTestModel*, *GraphTestModel*) pracují s modely *DatasetEditModel* a *TopologyEditModel* jako s externími datovými zdroji.

Všech pět datových modelů *(DatasetEditModel, TopologyEditModel, LearningConfigModel, DatasetTestModel, GraphTestModel)* dědí společného předka BaseModel. Ten implementuje základní funkčnost společnou pro všechny datové modely a zajišťuje komunikaci s modelem projektu *(Project)*, ve kterém jsou všechny datové modely editačních obrazovek uloženy.

Model *Project* umožňuje přejmenovávání a mazání modelů položek projektu v něm uložených a samozřejmě také vytváření nových modelů. Program umožňuje otevření více projektů najednou, modely projektů jsou uloženy v modelu *Workspace*. Model *Workspace* zajišťuje podporu pro vytváření nových projektů a otevírání a zavírání stávajících. *Workspace* také implementuje *QAbstractItemModel*, a slouží jako model pro stromové zobrazení projektu *(QTreeView)*.

Čtvrtý jmenný prostor nazvaný *Parser* obsahuje třídy *DatasetMdlParser, TopologyMdlParser, LrnConfMdlParser, DatasetMdlParser, GraphTestMdlParser a ProjectParser*, díky kterým lze modely položek projektu i samotný model projektu uložit do xml souboru a opětovně ho z něj načíst. Všechny třídy v tomto jmenném prostoru jsou realizovány s využitím návrhového vzoru singleton.

Poslední jmenný prostor *Test* obsahuje třídy unit testů *(BpAlgStTest, DatasetTest, LrnEngineTest, MlnNetStTest, NeuronTest)* k algoritmické části programu ze jmenného prostoru *NeuralNetwork*.

Zbylé třídy *(Settings)* a soubory zdrojových kódů *(Globaldef, NeuralNetCreator.cpp)* nejsou přiřazeny do žádného jmenného prostoru. Třída *Settings* se stará o zápis a načítání nastavení programu ze systémových registrů (Windows) nebo z konfiguračního souboru (Linux). Knihovna *Globaldef* se skládá z globálních definic typů a několika pomocných funkcí. Soubor *NeuralNetCreator.cpp* obsahuje funkci main, ve které dochází k vytvoření instance programu a jeho spuštění.

4.4 Grafické uživatelské rozhraní programu

Grafické uživatelské rozhraní programu bylo navrženo tak, aby bylo snadno pochopitelné pro uživatele orientujícího se v problematice, kterou se program zabývá, a to až do takové míry, že by tento uživatel neměl mít potřebu číst návod k použití a po spuštění programu by s ním měl bez většího problému pracovat. Grafické rozhraní programu by mělo vyhovovat i pokročilému uživateli, který by s ním měl být schopen pracovat pohodlně, rychle a efektivně.

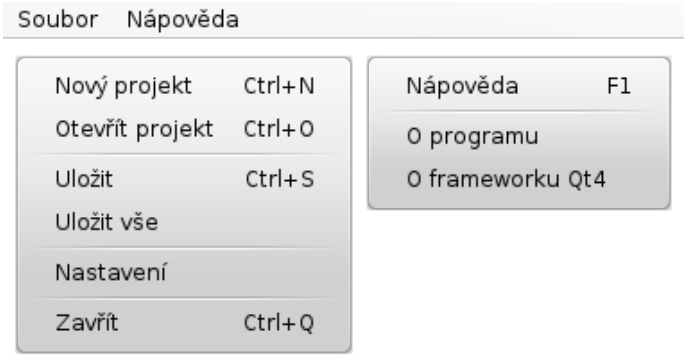
Jako inspirace pro rozložení ovládacích prvků a způsob práce s programem posloužilo vývojové prostředí QtCreator, ve kterém je program vyvíjen. Za povšimnutí na jeho grafickém rozhraní stojí především levý ovládací panel, který zobrazuje, ve které editační obrazovce se uživatel momentálně nachází nebo kterou činnost zrovna vykonává a umožňuje uživateli se mezi těmito obrazovkami a činnostmi pohybovat.

Grafické rozhraní programu Neural network Creator se skládá z výše zmíněného levého panelu, vedle kterého je ve všech obrazovkách s výjimkou uvítací obrazovky a obrazovky nápovědy umístěno stromové zobrazení otevřených projektů, pod kterým se navíc nachází seznam momentálně otevřených souborů. Zbytek okna programu zaplňuje obrazovka pro práci s aktuálně otevřeným modelem (souborem).

4.4.1 Hlavní menu programu

Hlavní menu (obr. 4.3) se skládá z nabídek *Soubor* a *Nápověda*. K vytvoření nebo otevření projektu slouží akce *Nový projekt* a *Otevřít projekt* z nabídky *Soubor*. V této nabídce jsou dále akce *Uložit* a *Uložit vše*, které umožňují uložení momentálně otevřeného modelu nebo uložení všech neuložených modelů. Akce *Nastavení* otevře dialog nastavení programu. Akce *Zavřít* zavře otevřené projekty a ukončí program, pokud jsou některé datové modely v projektu neuložené, tak bude uživatel tázán, zdali se mají uložit.

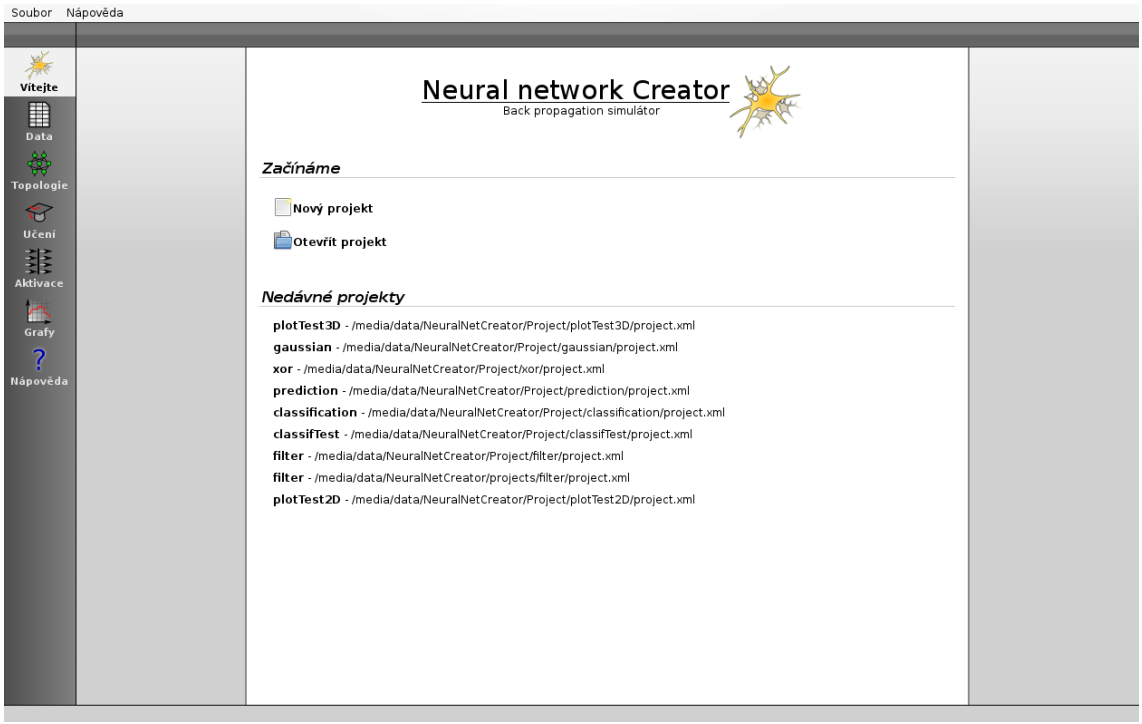
Pomocí akce *Nápověda* z nabídky *Nápověda* je možné přejít na obrazovku nápovědy. Na obrazovku nápovědy lze přejít také pomocí levého ovládacího panelu. Kliknutí na akci *O programu* zobrazí dialog z informacemi o programu a kliknutí na akci *O frameworku Qt4* zobrazí informace o použité verzi Qt4 frameworku.



Obr. 4.3: Hlavní menu programu

4.4.2 Uvítací obrazovka

Uvítací obrazovka (obr. 4.4) je navržena takovým způsobem, aby umožnila uživateli co nejrychleji začít pracovat s programem. Pomocí příslušných tlačítek lze založit nový projekt nebo otevřít projekt stávající. Pokud byl projekt nedávno otevřen, je možné ho znovu otevřít kliknutím levého tlačítka myši na jeho jméno v seznamu nedávno otevřených projektů.



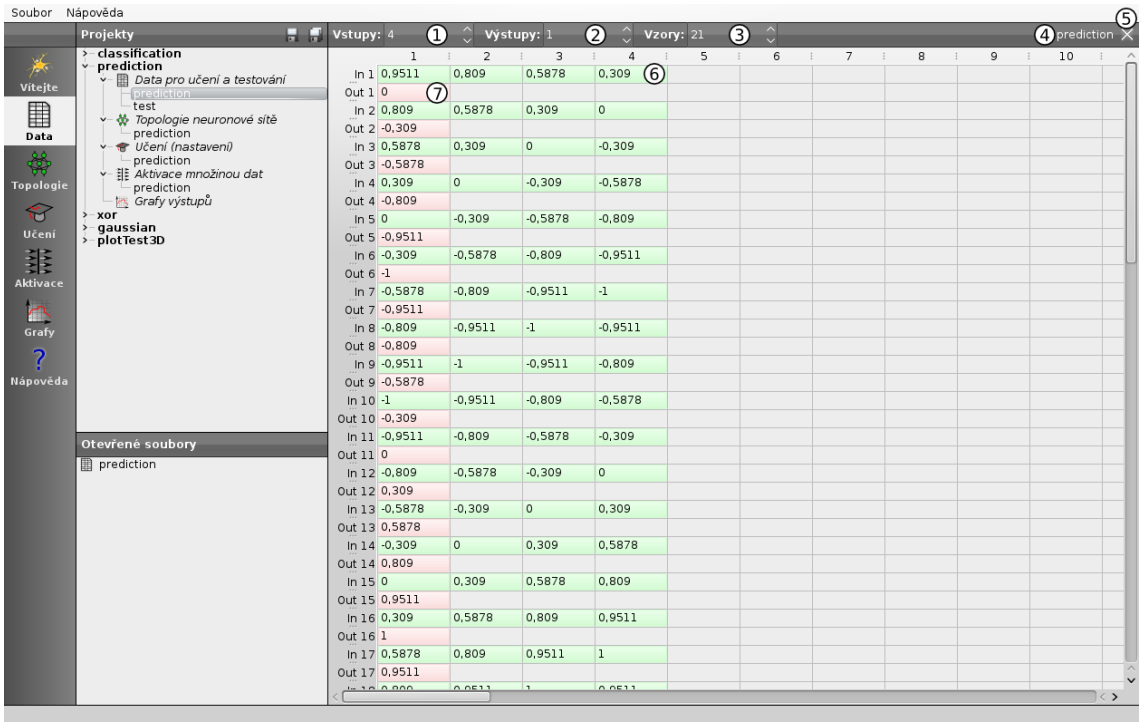
Obr. 4.4: Uvítací obrazovka programu

Po vytvoření nového, nebo otevření stávajícího projektu program automaticky přejde na obrazovku editace trénovacích a testovacích datových množin. K otevření žádného souboru (modelu) však zatím nedojde, místo toho bude zobrazena obrazovka s nápisem 'Žádný soubor nebyl otevřen'. K otevření souboru a jeho editaci je třeba kliknout na jeho název ve stromovém zobrazení projektu.

Do uvítací obrazovky se lze kdykoli vrátit pomocí tlačítka *Vítejte* umístěného na levém panelu. Pomocí tohoto panelu je možno se pohybovat mezi jednotlivými obrazovkami programu. Tlačítka na tomto panelu jsou seřazena podle předpokládaného průběhu tvorby neuronové sítě, který začíná tvorbou trénovacích a testovacích dat a pokračuje přes volbu topologie, učení a končí u testování naučené neuronové sítě.

4.4.3 Editor trénovacích a testovacích dat

Editor datových množin (obr. 4.5) je realizován tabulkou, kde jsou jednotlivé vzory umístěny v řádcích. Název řádku udává, zdali jde o vstup nebo výstup a o kolikátý vzor se jedná. Například pro vstupy neuronové sítě v druhém vzoru platí název řádku *In 2* a pro výstupy platí název *Out 2*. Číslo sloupce znamená číslo vstupu nebo výstupu neuronové sítě.



Obr. 4.5: Editace trénovacích a testovacích dat

Na horním panelu je třeba nastavit počet vstupů a výstupů neuronové sítě, pro kterou bude trénovací či testovací množina dat určena a počet vzorů v této množině. Buňky, které neodpovídají nastavenému počtu vstupů, výstupů a vzorů v množině dat jsou zašedlé, a není možné je editovat.

Mezi levým ovládacím panelem a tabulkou pro editaci datové množiny se nachází stromové zobrazení otevřených projektů a seznam otevřených souborů. Otevřený projekt lze v jeho stromovém zobrazení editovat pomocí kontextového menu, vyvolaného kliknutím pravého tlačítka myši na některou z položek. Podle úrovně a typu položky bude zobrazeno příslušné kontextové menu.

Seznam otevřených souborů je umístěn pod stromovým zobrazením otevřených projektů a zobrazuje názvy momentálně otevřených souborů (modelů) ve všech projektech. Kliknutím na některý ze zobrazených názvů lze přejít k editaci příslušného modelu. Neuložené modely jsou v tomto seznamu označeny hvězdičkou.

Stromové zobrazení otevřených projektů a seznam otevřených souborů jsou společné ovládací prvky pro všechny editační obrazovky, tj. pro všechny obrazovky programu kromě uvítací obrazovky a obrazovky nápovědy.

Seznam ovládacích prvků:

1. Počet vstupů neuronové sítě, pro kterou je množina dat určena
2. Počet výstupů neuronové sítě, pro kterou je množina dat určena
3. Počet vzorů v množině dat
4. Název právě editované množiny dat
5. Tlačítko ukončující editaci otevřené množiny dat
6. Vstupní vektor vzoru (zelené podbarvení)
7. Výstupní vektor vzoru (červené podbarvení)

4.4.4 Editor topologie

Editor topologie (obr. 4.6) je složen ze tří částí. Hlavní částí je samotná editace topologie, kde jsou graficky znázorněny všechny existující vrstvy neuronové sítě. Novou vrstvu, která nahradí současnou výstupní vrstvu lze přidat pomocí tlačítka *Přidat* na horním panelu (4). Tlačítko *Duplikovat* (5) vytvoří duplikát označené vrstvy a tlačítko *Odstranit* (6), označenou vrstvu smaže. Vrstvu (7) lze označit kliknutím levého tlačítka myši.

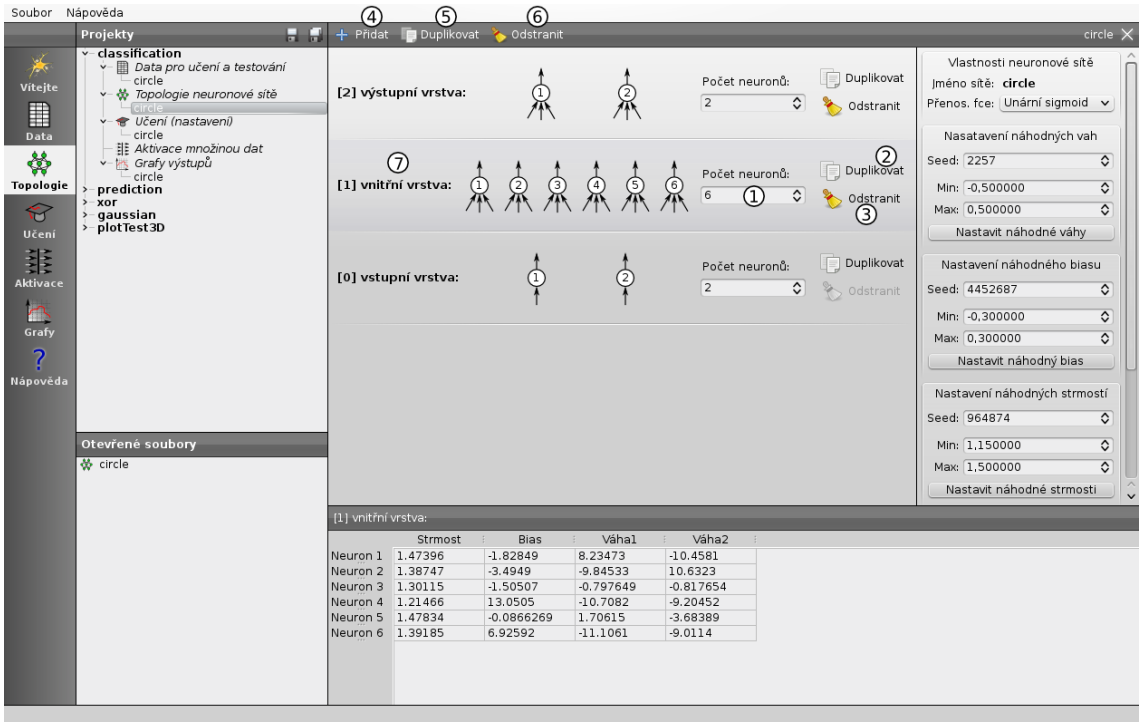
Počty neuronů v těchto vrstvách je možné měnit změnou hodnoty ve vstupním poli s názvem *Počet neuronů* (1). Dále je možné vrstvu duplikovat nebo smazat, nezávisle na tom, zdali je označená, pomocí tlačítek *Duplikovat* (2) a *Odtranit* (3).

Pod samotnou editací topologie se nachází tabulka zobrazující jednotlivé váhy, biasy a strmosti přenosové funkce neuronů v označené vrstvě. Protože vstupní vrstva

pouze přeposílá vstupy neuronové sítě na vstupy neuronů v další vrstvě, tak nemá žádnou přenosovou funkci, její bias je nulový a váhy všech vstupů mají hodnotu 1.

Na pravé straně obrazovky se nachází panel, ve kterém je možné nastavit přenosovou funkci neuronů v neuronové síti, provést náhodnou inicializaci vah, biasů a strmostí přenosových funkcí neuronů. Panel také zobrazuje informace o topologii neuronové sítě, jako je počet vstupů, výstupů, vrstev, neuronů a vah. Stejný panel se nachází i v obrazovce zabývající se učením neuronové sítě (obr. 4.7 a 4.8).

Náhodné hodnoty vah, biasů a strmostí jsou generovány na základě zadaného seedu. To znamená, že pokud bude zadaná stejná hodnota seedu a stejný rozsah náhodných čísel, bude pro stejnou topologii vygenerována shodná sekvence pseudonáhodných čísel.



Obr. 4.6: Editace topologie neuronové sítě

Seznam ovládacích prvků:

1. Počet neuronů ve vrstvě
2. Duplikovat vrstvu
3. Odstranit vrstvu
4. Přidání vrstvy do topologie
5. Duplikovat označenou vrstvu
6. Odstranit označenou vrstvu
7. Grafické znázornění vrstvy

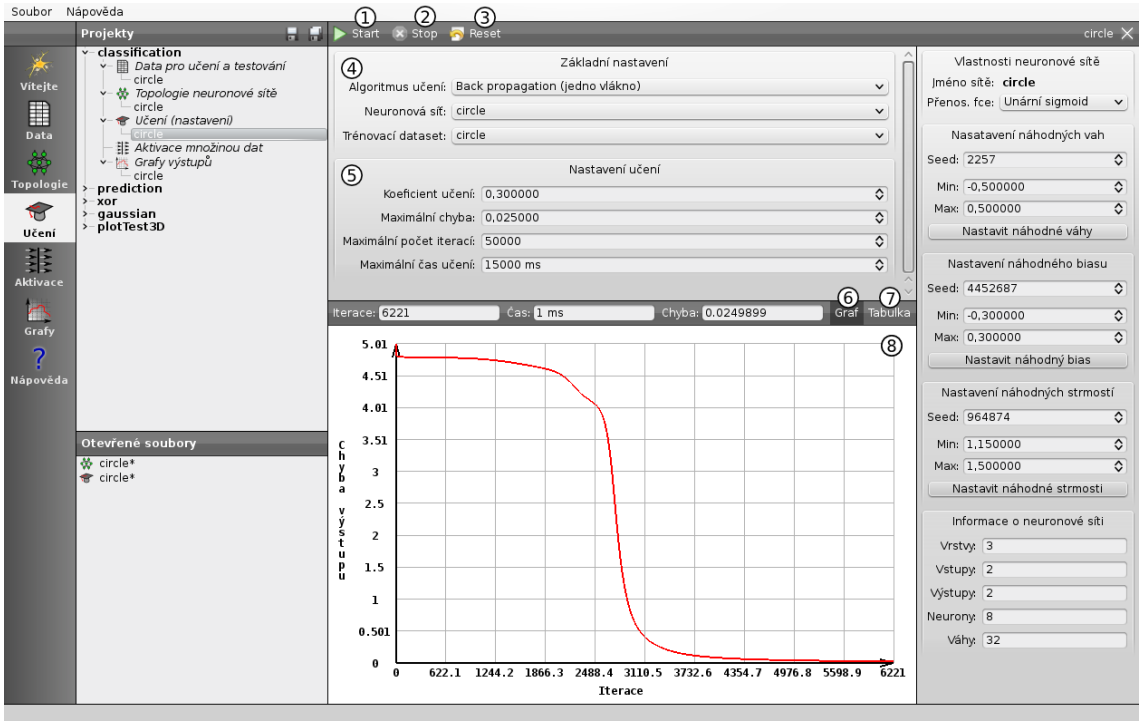
4.4.5 Učení neuronové sítě

Pomocí obrazovky učení (obr. 4.7 a 4.8) můžeme nastavit a ovládat proces učení neuronové sítě. Nejprve je nutné zvolit neuronovou síť, která bude trénována, a k ní příslušnou množinu trénovacích dat. Dále je možné zvolit implementaci algoritmu učení, nastavit koeficient učení, maximální počet iterací algoritmu, maximální čas učení a chybu sítě, při které bude učení zastaveno.

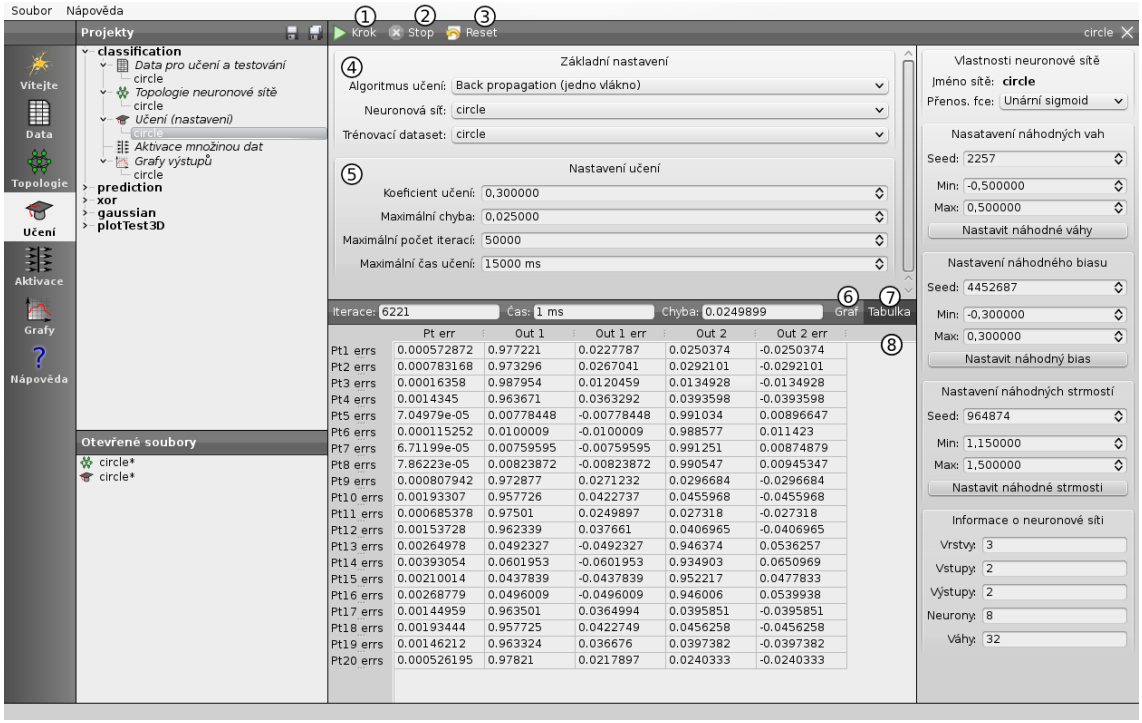
Tlačítky umístěnými na horním panelu lze ovládat proces učení. Pokud je zvolena neuronová síť a trénovací množina dat, tak je možné spustit učení pomocí tlačítka *Start* (1). Běžící učení lze zastavit tlačítkem *Stop* (2). Tlačítko *Reset* (3) vyčistí učící křivku neuronové sítě a inicializuje tuto síť náhodnými vahami, biasy a strmostmi přenosových funkcí neuronů.

Pod nastavením parametrů učení (4) a (5) se nachází učící křivka neuronové sítě, která se v průběhu učení pravidelně aktualizuje. Do grafu je možné přidávat jiné průběhy učení pomocí kontextového menu, které se zobrazí po kliknutí pravého tlačítka myši na plochu grafu.

Panel umístěný přímo nad grafem chyby sítě zobrazuje aktuální iteraci učení, uplynulou dobu učení a celkovou chybu sítě. Na jeho pravém konci se nacházejí tlačítka *Graf* (6) a *Tabulka* (7) pro volbu zobrazení učící křivky nebo tabulky chyb jednotlivých trénovacích vzorů.



Obr. 4.7: Učení neuronové sítě - učící křivka



Obr. 4.8: Učení neuronové sítě - tabulka chyb vzorů

Po kliknutí na tlačítko *Tabulka* (7) se zobrazí tabulka chyb jednotlivých trénovacích vzorů a hodnot jednotlivých výstupů a jejich chyb. Program taktéž přejde do režimu krokování.

Tlačítko *Start* změní název na *Krok* a spouští učení vždy pouze na jednu iteraci přes všechny trénovací vzory. Po ukončení procesu učení, tzn. jedné iterace (krokování) nebo i více iterací (běžné učení), dojde k přepočítání tabulky s využitím neuronové sítě se změněnými vahami. Tabulka se rovněž přepočítá při resetu učení, kde taktéž dochází ke změně vah v neuronové síti.

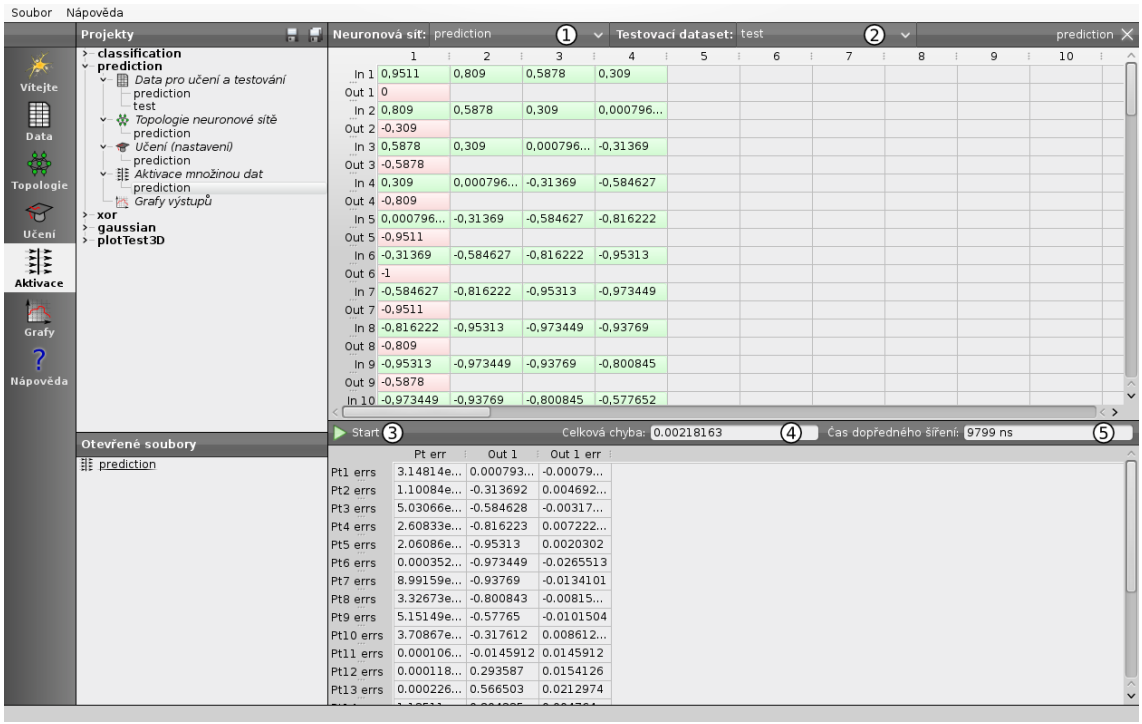
Seznam ovládacích prvků:

1. Start učení nebo krokování učení
2. Zastavení běžícího učení
3. Reset učení a neuronové sítě
4. Výběr algoritmu učení, neuronové sítě a množiny dat
5. Nastavení parametrů učení
6. Tlačítko pro zobrazení učící křivky
7. Tlačítko pro zobrazení tabulky chyb vzorů
8. Učící křivka sítě nebo tabulka chyb vzorů

4.4.6 Testování sítě množinou dat

Tato obrazovka, zobrazená na obrázku 4.9, umožňuje otestování neuronové sítě množinou testovacích dat, stačí pouze zvolit naučenou neuronovou síť (1) a příslušnou množinu testovacích dat (2), která je po jejím vybrání automaticky zobrazena v tabulce, nacházející se v horní části obrazovky. Po zvolení příslušné testovací množiny dat je automaticky vypočítána tabulka chyb umístěná ve spodní části obrazovky.

Řádky této tabulky odpovídají jednotlivým vzorům z datové množiny. V prvním sloupci jsou zobrazeny chyby těchto jednotlivých vzorů. V dalších sloupcích tabulky se nacházejí hodnoty jednotlivých výstupů neuronové sítě a absolutní hodnoty jejich chyb. Tabulku chyb je možné přepočítat pomocí tlačítka *Start* (3).



Obr. 4.9: Testování neuronové sítě množinou dat

Seznam ovládacích prvků:

1. Výběr neuronové sítě k otestování
2. Výběr testovací množiny dat
3. Tlačítko na přepočítání tabulky
4. Celková chyba neuronové sítě
5. Průměrná vybavovací doba v nanosekundách

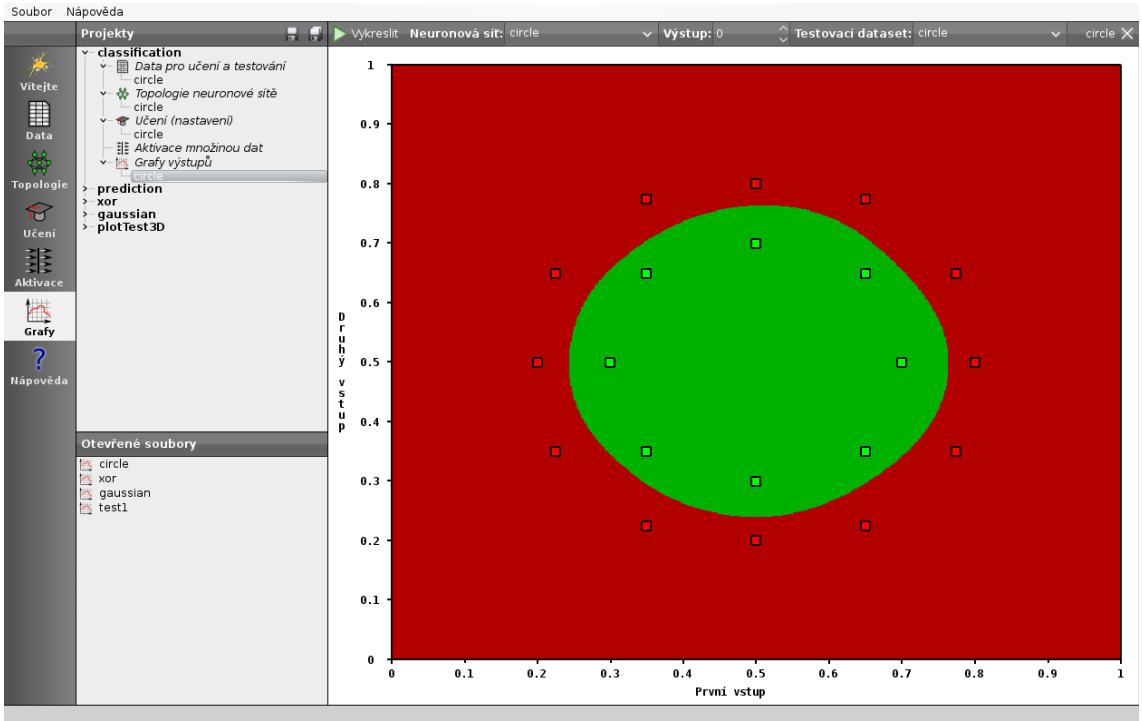
4.4.7 Výstupní graf neuronové sítě

Tato obrazovka nám umožňuje nechat si vykreslit různé druhy výstupních grafů neuronové sítě. Vykreslování grafů je řešeno pomocí OpenGL. K vykreslení grafu je nejprve nutné zvolit neuronovou síť, jejíž výstupní graf chceme vykreslit. Dále pak zvolit, pro jaký výstup neuronové sítě se má graf vykreslit.

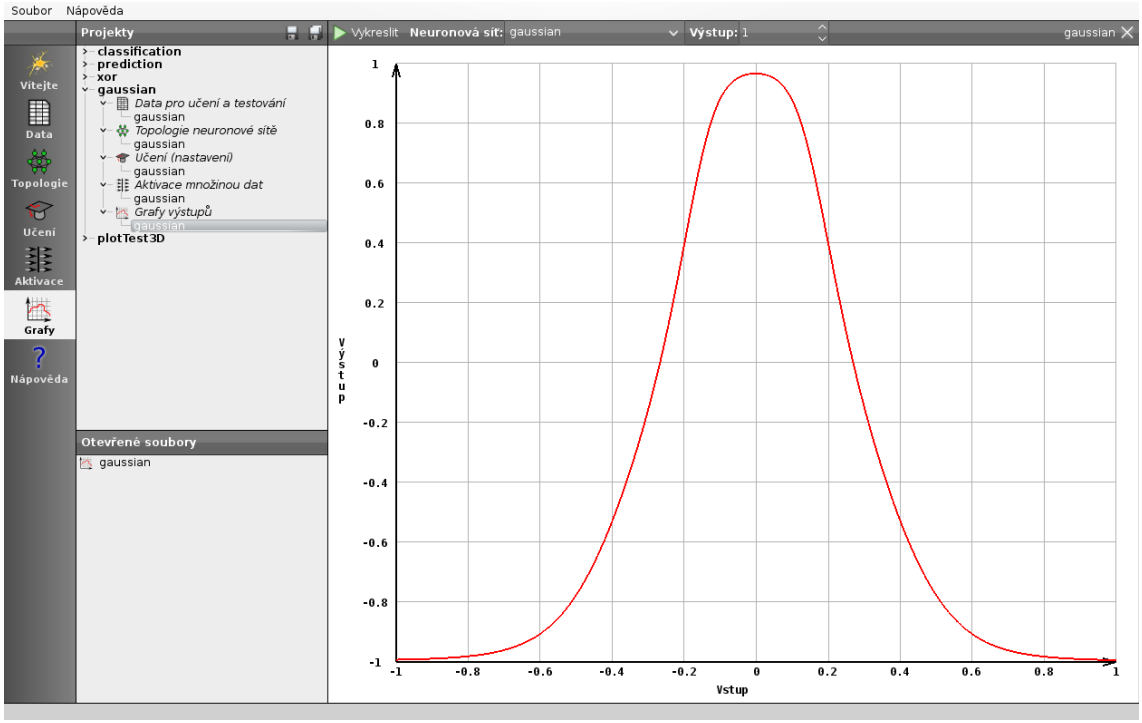
Při zvolení nultého výstupu bude vykreslen klasifikační diagram. Typ grafu zobrazující vybraný výstup neuronové sítě je závislý na počtu vstupů neuronové sítě. Vykreslování grafů je možné pouze pro neuronové sítě s jedním, dvěma a třemi výstupy. Požadovaný graf bude vykreslen po stisknutí tlačítka *Vykreslit* na horním ovládacím panelu.

Prvním typem grafu je klasifikační diagram (obr. 4.10). Každému výstupu neuronové sítě je přidělena jedna barva. Potom je pro každý bod grafu pomocí klasifikační funkce určen výstup s nejvyšším významem a bod je podle toho obarven příslušnou barvou, vznikne tak plocha rozdělená do barevných oblastí.

Program používá klasifikační funkci hledající výstup s nejvyšší hodnotou. U tohoto typu grafu je možné na horním panelu zvolit množinu dat, jejíž body budou v grafu následně zobrazeny. Pro vykreslení klasifikačního diagramu musí mít neuronová síť právě dva vstupy.



Obr. 4.10: Klasifikační diagram



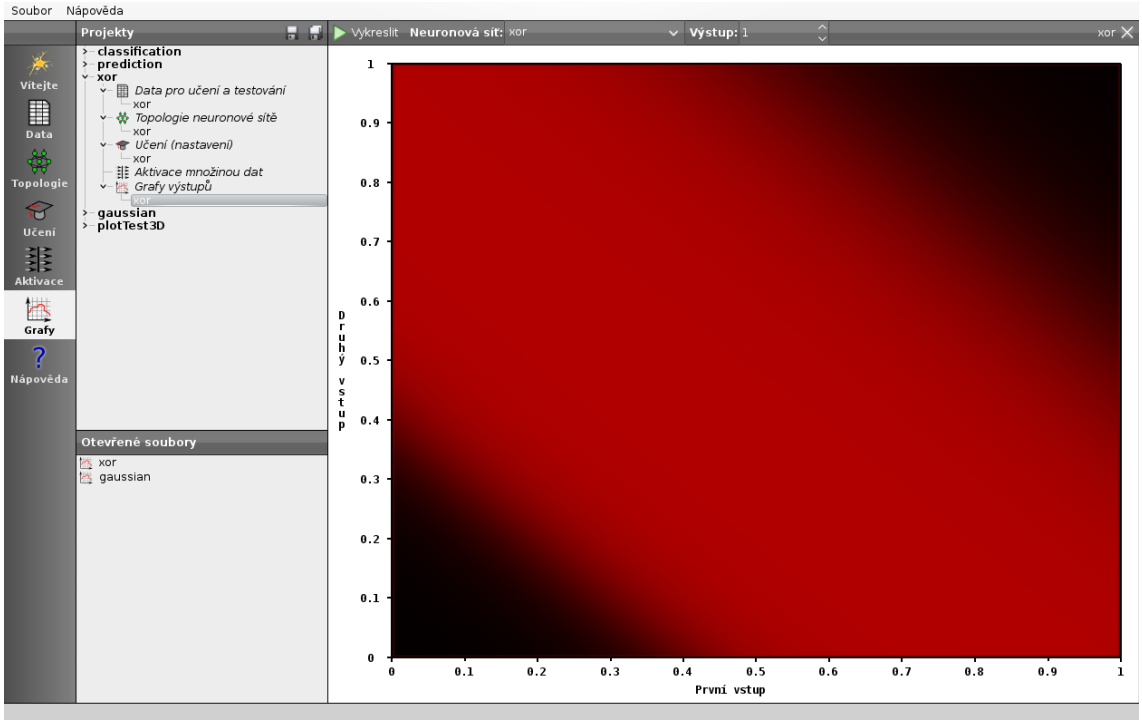
Obr. 4.11: Závislost výstupu na jednom vstupu

Pokud je zvolen konkrétní výstup a neuronová síť má jen jeden vstup, bude vykreslen graf závislosti hodnoty výstupu na hodnotě vstupu (obr. 4.11). Hodnoty na osách grafu jsou v rozsahu od 0 do 1 nebo od -1 do 1 podle zvolené přenosové funkce neuronů.

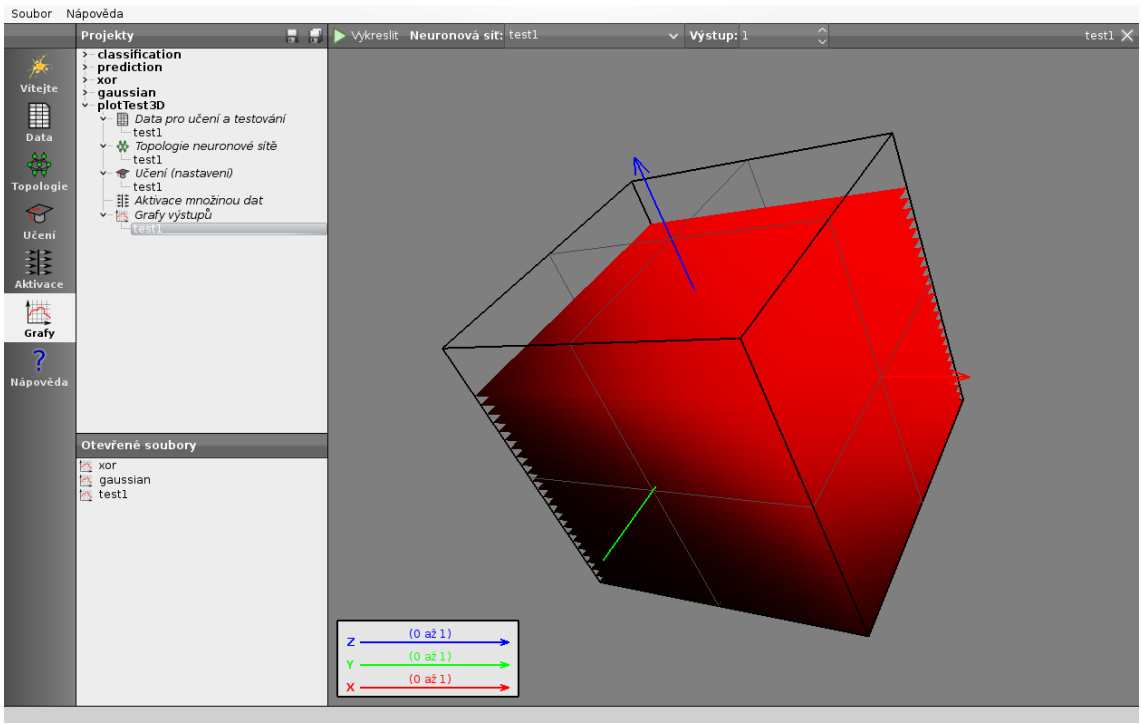
Při zvolení neuronové sítě se dvěma vstupy je vykreslen graf (obr. 4.12), kde na vodorovné ose je hodnota prvního vstupu, na svislé ose je hodnota druhého vstupu a barva plochy znamená hodnotu zvoleného výstupu. Černá barva znamená hodnotu 0, červená barva znamená +1 a modrá -1.

Pro neuronovou síť se třemi vstupy je vykreslen trojrozměrný graf (obr. 4.13), na jehož osách leží hodnoty jejích vstupů. Krychli grafu otáčíme pohybem myši při stisknutém levém tlačítku. Řez krychlí provádíme pohybem myši ve vertikálním směru při stisknutém pravém tlačítku. Zobrazení uvedeme do původního nastavení pomocí příslušné akce v kontextovém menu zobrazitelném dvojklikem pravého tlačítka myši na plochu grafu.

Pomocí kontextového můžeme každý graf uložit do souboru jako obrázek typu *png*. Grafické zobrazení jednoho výstupu u neuronové sítě s jedním vstupem je navíc možné uložit jako soubor *csv* a otevřít ho například v Excelu.



Obr. 4.12: Závislost výstupu na dvou vstupech

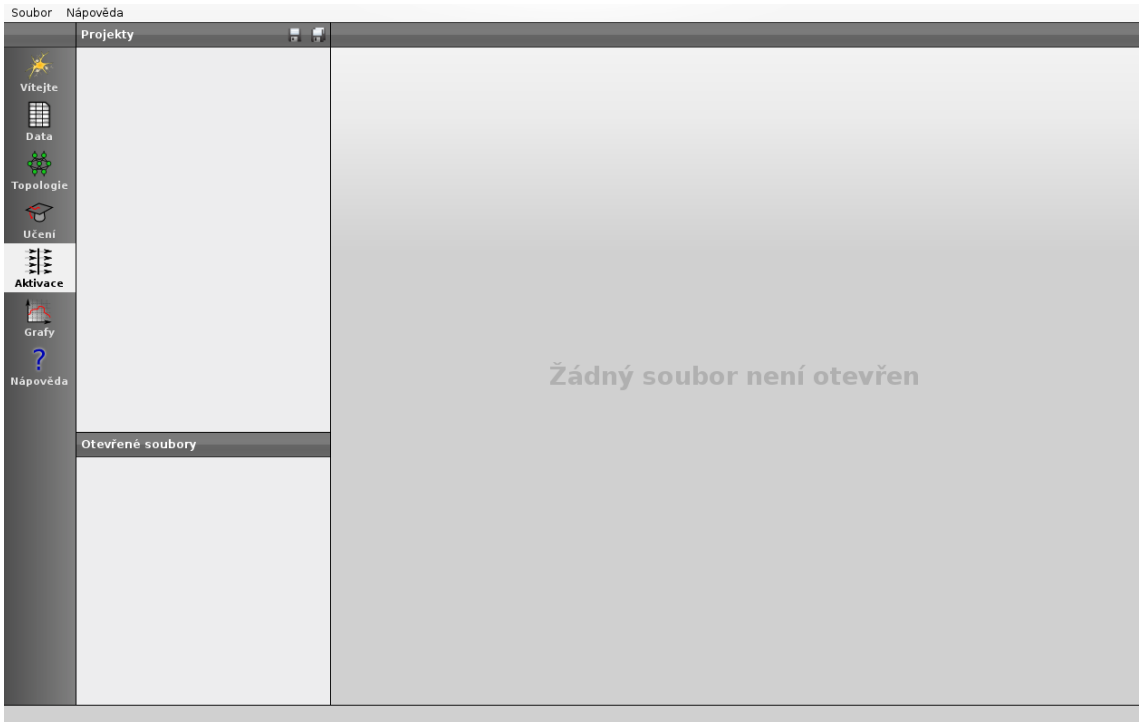


Obr. 4.13: Závislost výstupu na třech vstupech

4.4.8 Program bez otevřených souborů a projektů

Obrazovka s nápisem *Žádný soubor není otevřen* (obr. 4.14) bude zobrazena v případech, že se uživatel kliknutím na levý panel dostane na obrazovku určenou k editaci některého typu modelu (souboru) v projektu, od kterého ale není žádný otevřen. V takovém případě je třeba nějaký model (soubor) otevřít nebo v projektu vytvořit pomocí kontextového menu zobrazeného po kliknutí pravého tlačítka myši na název kategorie ve stromu projektu.

Pokud není otevřen žádný projekt, je třeba nějaký otevřít nebo vytvořit nový. To lze udělat pomocí hlavního menu programu, tlačítka *Nový projekt* na uvítací obrazovce nebo pomocí kontextového menu ve stromovém zobrazení projektů.

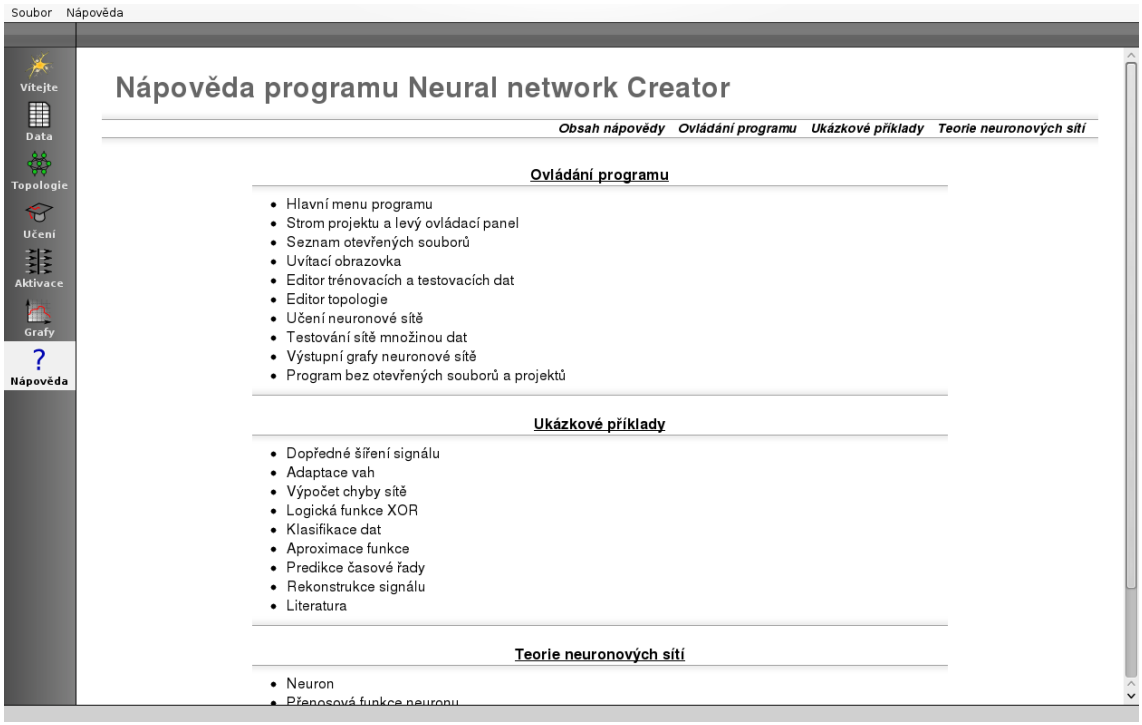


Obr. 4.14: Program bez otevřených souborů a projektů

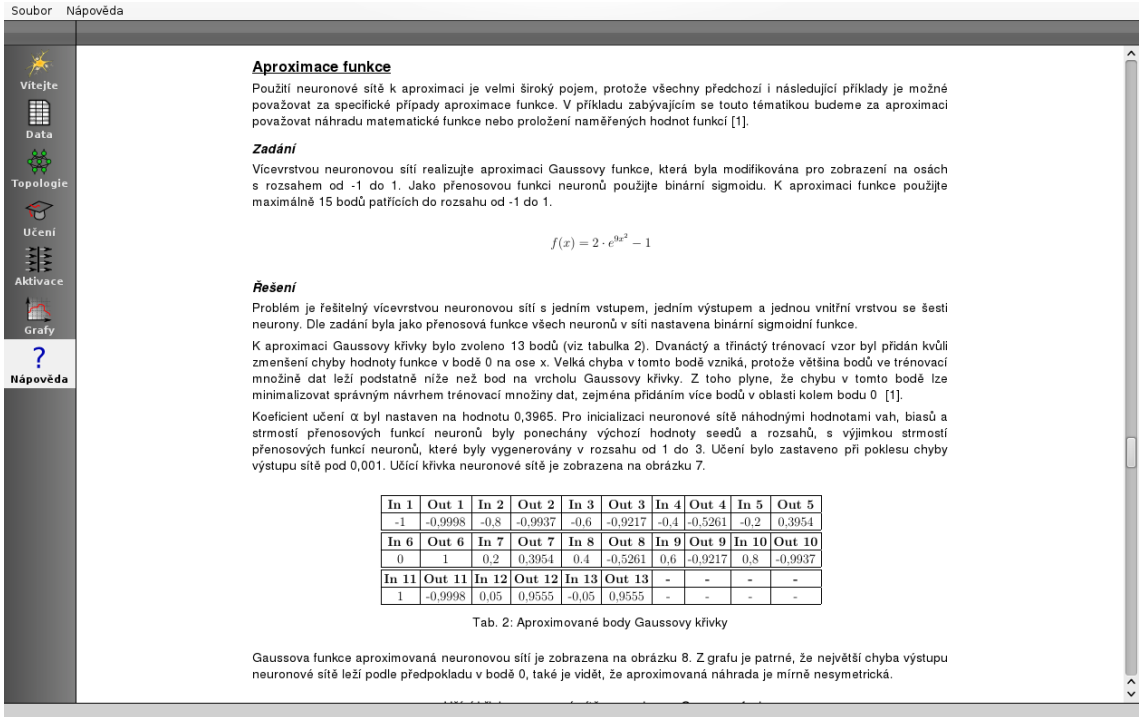
4.4.9 Nápověda

Nápověda (obr. 4.15 a 4.16) je realizována formou webové stránky zobrazené v okně programu. Hlavní část nápovědy je návod k obsluze programu s popisem jeho grafického uživatelského rozhraní. Další části nápovědy mají výukový charakter. Obsahují základy teorie neuronových sítí (neuron, vícevrstvá neuronová síť, algoritmus backpropagation) a několik výukových příkladů včetně vypracovaného řešení.

Jednotlivé ovládací prvky grafického rozhraní programu jsou popsány s využitím tooltipů, které se zobrazí při zastavení kurzoru nad některým z těchto prvků.



Obr. 4.15: Nápověda programu - obsah



Obr. 4.16: Nápověda programu - výukový příklad

5 VÝUKOVÉ PŘÍKLADY

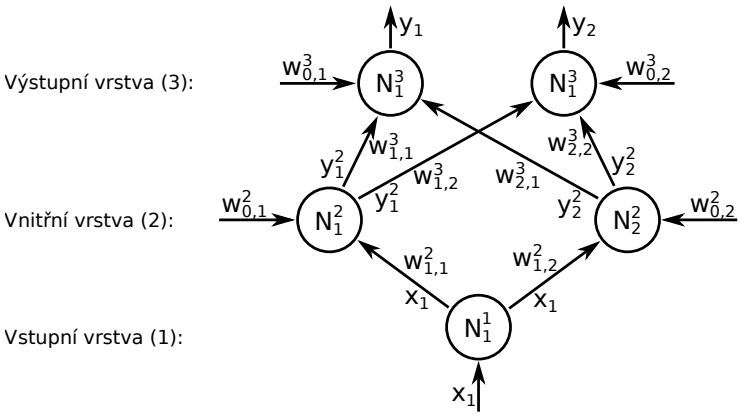
Tato kapitola je věnována příkladům do výuky. První příklad ukazuje aktivaci neuronové sítě zadaným vstupem. Další příklady obsahují ukázky jedné iterace algoritmu backpropagation a výpočtu chyby sítě. Ostatní příklady jsou vypracovány v programu Neural network Creator a prakticky demonstrují možnosti neuronové sítě při klasifikaci, aproximaci, predikci a filtraci šumu.

5.1 Dopředné šíření signálu

Tento příklad ukazuje proces aktivace neuronové sítě, tj. postupné šíření vstupního signálu od vstupní vrstvy k výstupní.

Zadání

Vypočítejte hodnoty výstupů vícevrstvé neuronové sítě s jedním vstupem, dvěma výstupy a topologií znázorněnou na obrázku 5.1.



Obr. 5.1: Neuronová síť s topologií 1-2-2

Hodnoty vah a biasů neuronů v síti jsou

$$w^2_{0,1} = 0{,}561 \qquad w^3_{1,1} = -1{,}852$$
$$w^2_{1,1} = -2{,}327 \qquad w^3_{1,2} = 0{,}025$$
$$w^2_{0,2} = 1{,}239 \qquad w^3_{0,2} = 0{,}956$$
$$w^2_{1,2} = 0{,}121 \qquad w^3_{2,1} = -3{,}726$$
$$w^3_{0,1} = 3{,}653 \qquad w^3_{2,2} = 0{,}884$$

Přenosová funkce neuronů je unipolární sigmoida

$$f(x) = \frac{1}{1 + e^{-x}} \tag{5.1}$$

Na vstup neuronové sítě byl přiveden signál o hodnotě

$$x_1 = 0{,}223$$

Řešení

Neuron v první vrstvě pouze přeposílá n-tý vstup neuronové sítě na všechny n-té vstupy neuronů v první skryté vrstvě, takže platí

$$y_1^1 = x_1 \tag{5.2}$$

Při výpočtu hodnot výstupů první skryté vrstvy nejprve vypočítáme vnitřní potenciály neuronů N_1^2 a N_2^2

$$y_{1,in}^2 = w_{0,1}^2 + w_{1,1}^2 \cdot x_1 \tag{5.3}$$
$$y_{1,in}^2 = 0{,}561 - 2{,}327 \cdot 0{,}223 = 0{,}042$$
$$y_{2,in}^2 = w_{0,2}^2 + w_{1,2}^2 \cdot x_1 \tag{5.4}$$
$$y_{2,in}^2 = 1{,}239 + 0{,}121 \cdot 0{,}223 = 1{,}266$$

Takto získané vnitřní potenciály neuronů dosadíme do jejich přenosových funkcí, čímž získáme jejich konečné výstupy

$$y_1^2 = f(0{,}042) = \frac{1}{1 + e^{-0{,}042}} = 0{,}51$$
$$y_2^2 = f(1{,}266) = \frac{1}{1 + e^{-1{,}266}} = 0{,}78$$

K získání výstupu celé neuronové sítě postupujeme obdobným způsobem, na vstup výstupní vrstvy přivedeme signál, získaný z výstupů vrstvy předchozí. Nejprve vypočítáme vnitřní potenciály neuronů

$$y_{1,in}^3 = w_{0,1}^3 + w_{1,1}^3 \cdot y_1^2 + w_{2,1}^3 \cdot y_2^2 \tag{5.5}$$
$$y_{1,in}^3 = 3{,}653 - 1{,}852 \cdot 0{,}51 + 0{,}025 \cdot 0{,}78 = 2{,}728$$
$$y_{2,in}^3 = w_{0,2}^3 + w_{1,2}^3 \cdot y_1^2 + w_{2,2}^3 \cdot y_2^2 \tag{5.6}$$
$$y_{2,in}^3 = 0{,}956 - 3{,}726 \cdot 0{,}51 + 0{,}884 \cdot 0{,}78 = -0{,}255$$

Výstupy celé neuronové sítě potom jsou

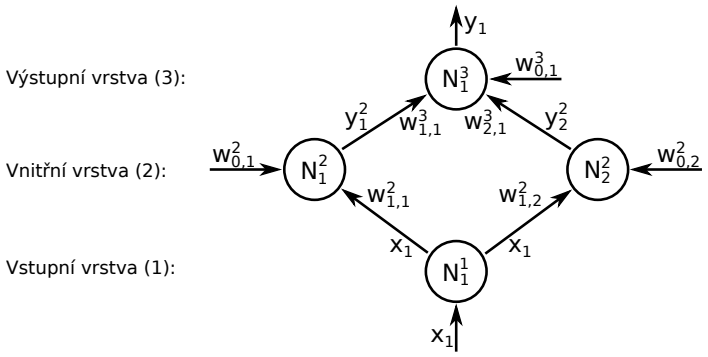
$$y_1^3 = y_1 = f(2{,}728) = \frac{1}{1 + e^{-2{,}728}} = 0{,}939$$
$$y_2^3 = y_2 = f(-0{,}255) = \frac{1}{1 + e^{0{,}255}} = 0{,}437$$

5.2 Adaptace vah

Tento příklad prakticky demonstruje jednu iteraci algoritmu backpropagation na jednom trénovacím vzoru, a to včetně dopředného šíření vstupního signálu a adaptace vah po výpočtu chybového faktoru δ_k.

Zadání

Vypočtěte změny hodnot vah neuronů v neuronové síti na obrázku 5.2 po jedné iteraci algoritmu backpropagation.



Obr. 5.2: Neuronová síť s topologií 1-2-1

Počáteční hodnoty vah a biasů neuronů v síti jsou

$$w^2_{0,1} = 0{,}561 \qquad w^3_{0,1} = 0{,}746$$
$$w^2_{1,1} = 0{,}992 \qquad w^3_{1,1} = 0{,}398$$
$$w^2_{0,2} = 0{,}147 \qquad w^3_{2,1} = 0{,}812$$
$$w^2_{1,2} = 0{,}228$$

Počítejte s hodnotou koeficientu učení

$$\alpha = 2{,}5$$

Množina trénovacích dat

$$T = \{S_1, T_1\} \tag{5.7}$$
$$S_1 = [0{,}95],\ T_1 = [0{,}05]$$

Přenosová funkce neuronů a její derivace

$$f(x) = \frac{1}{1+e^{-x}} \tag{5.8}$$

$$f'(x) = \frac{e^{-x}}{(1+e^{-x})^2} \tag{5.9}$$

Řešení

Pro výpočet chybových faktorů pomocí zpětného šíření chyby je nejdříve nutné vypočítat hodnoty výstupů neuronů vnitřní i výstupní vrstvy. Nejdříve vypočítáme výstupy vnitřní vrstvy

$$y_1^2 = f(w_{0,1}^2 + w_{1,1}^2 \cdot x_1) = f(0{,}561 + 0{,}992 \cdot 0{,}95) = f(1{,}503) = 0{,}818 \tag{5.10}$$

$$y_2^2 = f(w_{0,2}^2 + w_{1,2}^2 \cdot x_1) = f(0{,}147 + 0{,}228 \cdot 0{,}95) = f(0{,}364) = 0{,}59 \tag{5.11}$$

Pro zjištění výstupu sítě dosadíme vypočtené hodnoty výstupů vnitřní vrstvy na vstupy neuronu výstupní vrstvy

$$y_1 = f(w_{0,1}^3 + w_{1,1}^3 \cdot y_1^2 + w_{2,1}^3 \cdot y_2^2) \tag{5.12}$$

$$y_1 = f(0{,}746 + 0{,}398 \cdot 0{,}818 + 0{,}812 \cdot 0{,}59) = f(1{,}551) = 0{,}825$$

Nyní již můžeme přejít k výpočtu chybových faktorů. Nejprve vypočítáme chybový faktor pro neuron ve výstupní vrstvě

$$\delta_1^3 = (t_1 - y_1) \cdot f'(w_{0,1}^3 + w_{1,1}^3 \cdot y_1^2 + w_{2,1}^3 \cdot y_2^2) \tag{5.13}$$

$$\delta_1^3 = (0{,}05 - 0{,}825) \cdot f'(0{,}746 + 0{,}398 \cdot 0{,}818 + 0{,}812 \cdot 0{,}59)$$

$$\delta_1^3 = -0{,}775 \cdot f'(1{,}551) = -0{,}775 \cdot 0{,}144 = -0{,}112$$

Dále s využitím chybového faktoru výstupní vrstvy zpětně šíříme chybu a vypočítáme chybové faktory neuronů ve vnitřní vrstvě

$$\delta_1^2 = (\delta_1^3 \cdot w_{1,1}^3) \cdot f'(w_{0,1}^2 + w_{1,1}^2 \cdot x_1) \tag{5.14}$$

$$\delta_1^2 = (-0{,}112 \cdot 0{,}992) \cdot f'(0{,}561 + 0{,}992 \cdot 0{,}95) = -0{,}017$$

$$\delta_2^2 = (\delta_1^3 \cdot w_{2,1}^3) \cdot f'(w_{0,2}^2 + w_{1,2}^2 \cdot x_1) \tag{5.15}$$

$$\delta_2^2 = (-0{,}112 \cdot 0{,}228) \cdot f'(0{,}147 + 0{,}228 \cdot 0{,}95) = -0{,}0063$$

Když známe chybové faktory neuronů, tak můžeme vypočítat hodnoty přírůstků jejich vah a biasů. Rychlost změny vah je závislá na koeficientu učení α

$$\Delta w_{0,1}^2 = \alpha \cdot \delta_1^2 = 2{,}5 \cdot -0{,}017 = -0{,}043 \tag{5.16}$$

$$\Delta w_{1,1}^2 = \alpha \cdot \delta_1^2 \cdot x_1 = 2{,}5 \cdot -0{,}017 \cdot 0{,}95 = -0{,}0404 \tag{5.17}$$

$$\Delta w_{0,2}^2 = \alpha \cdot \delta_2^2 = 2{,}5 \cdot -0{,}0063 = -0{,}016 \tag{5.18}$$

$$\Delta w_{1,2}^2 = \alpha \cdot \delta_2^2 \cdot x_1 = 2{,}5 \cdot -0{,}0063 \cdot 0{,}95 = 0{,}015 \tag{5.19}$$

$$\Delta w_{0,1}^3 = \alpha \cdot \delta_1^3 = 2{,}5 \cdot -0{,}112 = -0{,}28 \tag{5.20}$$

$$\Delta w_{1,1}^3 = \alpha \cdot \delta_1^3 \cdot y_1^2 = 2{,}5 \cdot -0{,}112 \cdot 0{,}818 = -0{,}229 \tag{5.21}$$

$$\Delta w_{2,1}^3 = \alpha \cdot \delta_1^3 \cdot y_2^2 = 2{,}5 \cdot -0{,}112 \cdot 0{,}59 = -0{,}165 \tag{5.22}$$

5.3 Výpočet chyby sítě

Chyba výstupu neuronové sítě slouží jako kritérium, které nám dovoluje posoudit, jak dobře je neuronová síť naučená. Umožňuje nám sledovat průběh učení neuronové sítě a následně kontrolu její funkčnosti při aktivaci testovací množinou dat.

Zadání

Vypočítejte chybu výstupu vícevrstvé neuronové sítě se třemi výstupy. Chybu lze vypočítat podle vzorce [10]

$$E = \sum_{i=0}^{q} \frac{1}{2} \sum_{k=0}^{n} (y_{i,k} - t_{i,k})^2 \tag{5.23}$$

Požadované výstupy neuronové sítě pro první a druhý trénovací vzor jsou

$$T_1 = [1, 0, 0],\ T_2 = [0, 0, 1]$$

Reálné výstupy neuronové sítě pro první a druhý trénovací vzor jsou

$$Y_1 = [0.923, 0.092, 0.125],\ Y_2 = [0.139, 0.035, 0.896]$$

Řešení

Nejprve vypočítáme parciální chybu pro první trénovací vzor

$$E_1 = \frac{1}{2} \sum_{k=0}^{n} (y_{i,k} - t_{i,k})^2 = \frac{1}{2} \left((Y_{1,1} - T_{1,1})^2 + (Y_{1,2} - T_{1,2})^2 + (Y_{1,3} - T_{1,3})^2 \right) \tag{5.24}$$

$$E_1 = \frac{1}{2} \left((0{,}923 - 1)^2 + (0{,}092 - 0)^2 + (0{,}125 - 0)^2 \right) = 0{,}015$$

Dále vypočítáme parciální chybu pro druhý trénovací vzor

$$E_2 = \frac{1}{2} \sum_{k=0}^{n} (y_{i,k} - t_{i,k})^2 = \frac{1}{2} \left((Y_{2,1} - T_{2,1})^2 + (Y_{2,2} - T_{2,2})^2 + (Y_{2,3} - T_{2,3})^2 \right) \tag{5.25}$$

$$E_2 = \frac{1}{2} \left((0{,}139 - 0)^2 + (0{,}035 - 0)^2 + (0{,}896 - 1)^2 \right) = 0{,}0157$$

Celkovou chybu neuronové sítě získáme sumací parciálních chyb vypočtených pro jednotlivé trénovací vzory

$$E = \sum_{i=0}^{q} E_i = E_1 + E_2 \tag{5.26}$$

$$E = 0{,}015 + 0{,}0157 = 0{,}0307$$

5.4 Logická funkce XOR

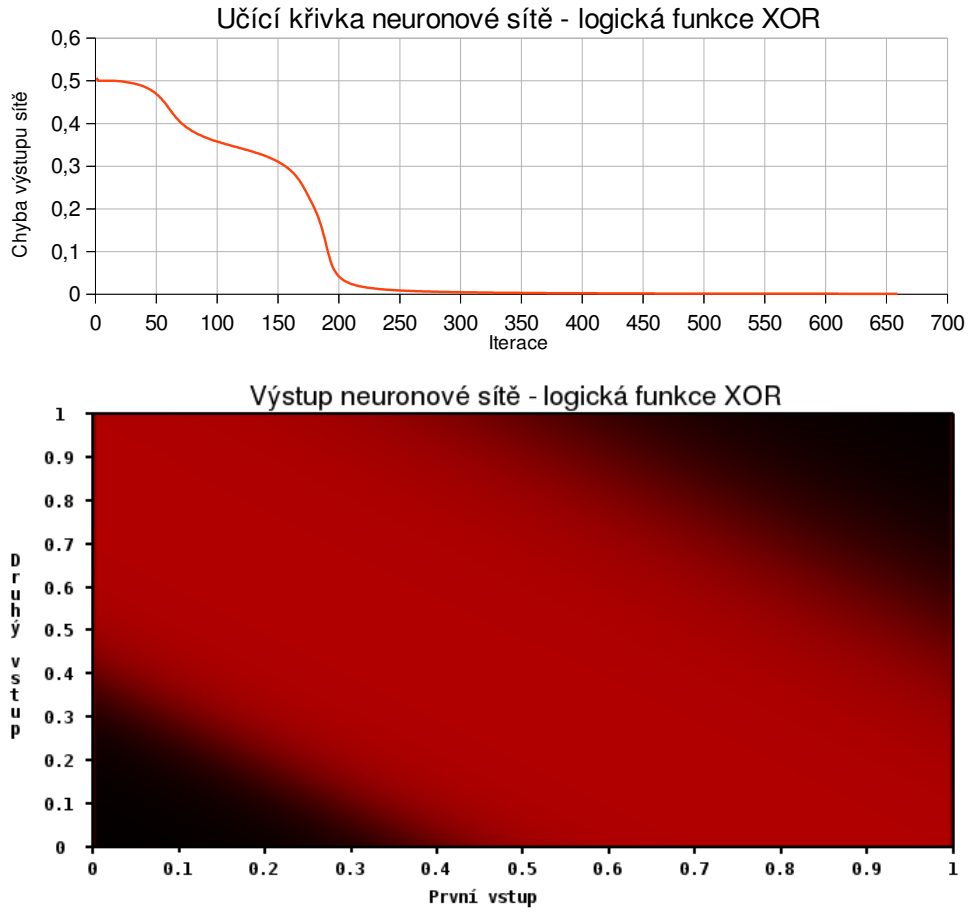
Tento příklad byl původně vypracován jako jeden z testů implementace algoritmu backpropagation v programu Neural network Creator. Při jeho vypracovávání by se měl uživatel seznámit s ovládáním programu.

Zadání

Realizujte logickou funkci XOR pomocí vícevrstvé neuronové sítě, a pro tuto síť zobrazte graf jejího výstupu.

Řešení

Pro řešení problému byla zvolena neuronová síť se dvěma vstupy, jedním výstupem a jednou vnitřní vrstvou o čtyřech neuronech. Experimentálně určená hodnota koeficientu učení $\alpha = 6$ zajišťuje dostatečně rychlý průběh učení, který však stále zůstává monotónní (obr. 5.3). Pro generování náhodných vah, biasů a strmostí přenosových funkcí neuronů byly ponechány výchozí hodnoty seedů a rozsahů.



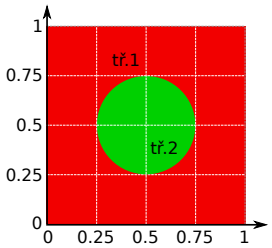
Obr. 5.3: Učící křivka neuronové sítě a graf výstupu naučené sítě

5.5 Klasifikace dat

Klasifikace dat je oblast, ve které jsou dnes neuronové sítě nejvíce používány. Klasifikací dat může být binární rozhodování typu ano / ne nebo třídění dat do několika typových tříd. Třídění dat podle příznaků je možno použít například v oblasti počítačového vidění, k rozpoznávání objektů s již určenými hodnotami příznaků [5].

Zadání

Navrhněte topologii neuronové sítě a její trénovací množinu dat tak, aby neuronová síť klasifikovala vstupy do dvou tříd podle obrázku 5.4. Jednotlivé klasifikační třídy odpovídají jednotlivým výstupům neuronové sítě.



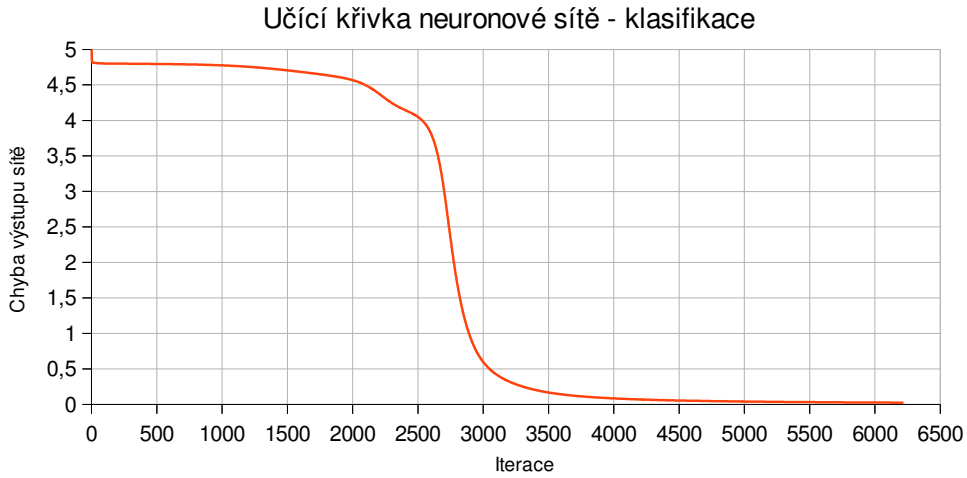
Obr. 5.4: Požadovaný klasifikační diagram

Řešení

K realizaci požadované klasifikační funkčnosti byla zvolena neuronová síť se dvěma vstupy, dvěma výstupy a jednou vnitřní vrstvou o šesti neuronech, tato síť byla trénována množinou dat uvedenou v tabulce 5.1, její body jsou také zobrazeny v klasifikačním diagramu.

	1	**2**		**1**	**2**		**1**	**2**		**1**	**2**
In 1	0,5	0,2	**In 6**	0,7	0,5	**In 11**	0,775	0,35	**In 16**	0,35	0,65
Out 1	1	0	**Out 6**	0	1	**Out 11**	1	0	**Out 16**	0	1
In 2	0,5	0,8	**In 7**	0,3	0,5	**In 12**	0,65	0,225	**In 17**	0,775	0,65
Out 2	1	0	**Out 7**	0	1	**Out 12**	1	0	**Out 17**	1	0
In 3	0,2	0,5	**In 8**	0,5	0,3	**In 13**	0,65	0,35	**In 18**	0,225	0,35
Out 3	1	0	**Out 8**	0	1	**Out 13**	0	1	**Out 18**	1	0
In 4	0,8	0,5	**In 9**	0,225	0,65	**In 14**	0,65	0,65	**In 19**	0,65	0,775
Out 4	1	0	**Out 9**	1	0	**Out 14**	0	1	**Out 19**	1	0
In 5	0,5	0,7	**In 10**	0,35	0,775	**In 15**	0,35	0,35	**In 20**	0,35	0,225
Out 5	0	1	**Out 10**	1	0	**Out 15**	0	1	**Out 20**	1	0

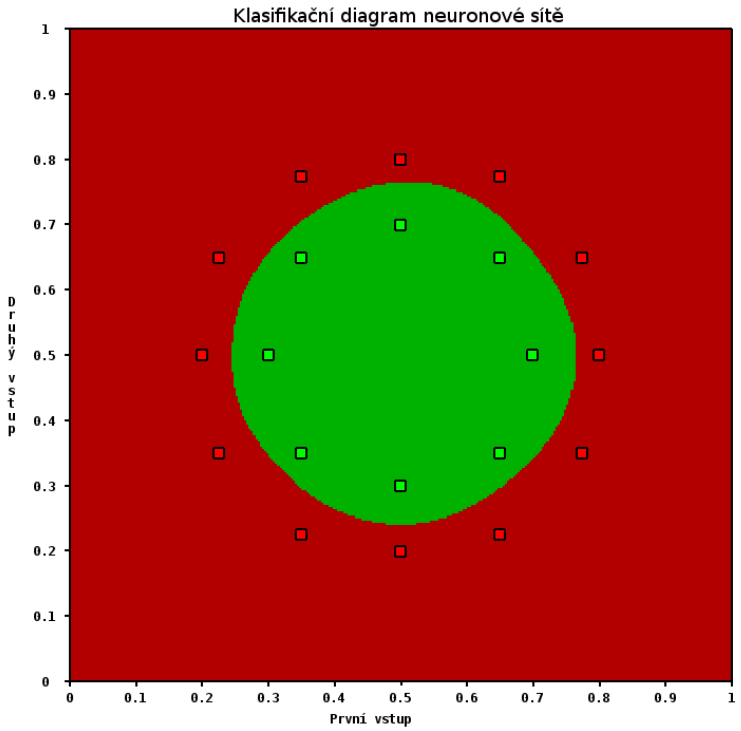
Tab. 5.1: Množina trénovacích dat pro příklad klasifikace dat



Obr. 5.5: Učící křivka neuronové sítě - klasifikace

Dále byl zvolen koeficient učení $\alpha = 0{,}35$. Seed pro generování náhodných vah neuronů byl nastaven na hodnotu 2257. Strmosti přenosových funkcí neuronů byly vygenerovány v rozsahu od 1,15 od 1,5. Ostatní seedy a rozsahy byly ponechány na výchozích hodnotách. Učení bylo zastaveno při poklesu chyby výstupu sítě pod 0,025. Učící křivka je zobrazena na obrázku 5.5.

Po dokončení učení byl pro neuronovou síť vykreslen klasifikační diagram, který je zobrazený na obrázku 5.6. Lepšího výsledku by bylo možné dosáhnout například naučením neuronové sítě na hrubší trénovací množinu a jejím následným doučením na jemnější trénovací množinu.



Obr. 5.6: Výsledný klasifikační diagram

5.6 Aproximace funkce

Použití neuronové sítě k aproximaci je velmi široký pojem, protože všechny předchozí i následující příklady je možné považovat za specifické případy aproximace funkce. V příkladu zabývajícím se touto tématikou budeme za aproximaci považovat náhradu matematické funkce nebo proložení naměřených hodnot funkcí [5].

Zadání

Vícevrstvou neuronovou sítí realizujte aproximaci Gaussovy funkce která byla modifikována pro zobrazení na osách s rozsahem od -1 do 1

$$f(x) = 2 \cdot e^{9x^2} - 1 \tag{5.27}$$

Jako přenosovou funkci neuronů použijte bipolární sigmoidu. K aproximaci funkce použijte maximálně 15 bodů patřících do rozsahu od -1 do 1.

Řešení

Problém je řešitelný vícevrstvou neuronovou sítí s jedním vstupem, jedním výstupem a jednou vnitřní vrstvou se šesti neurony. Dle zadání byla jako přenosová funkce všech neuronů v síti nastavena bipolární sigmoidní funkce.

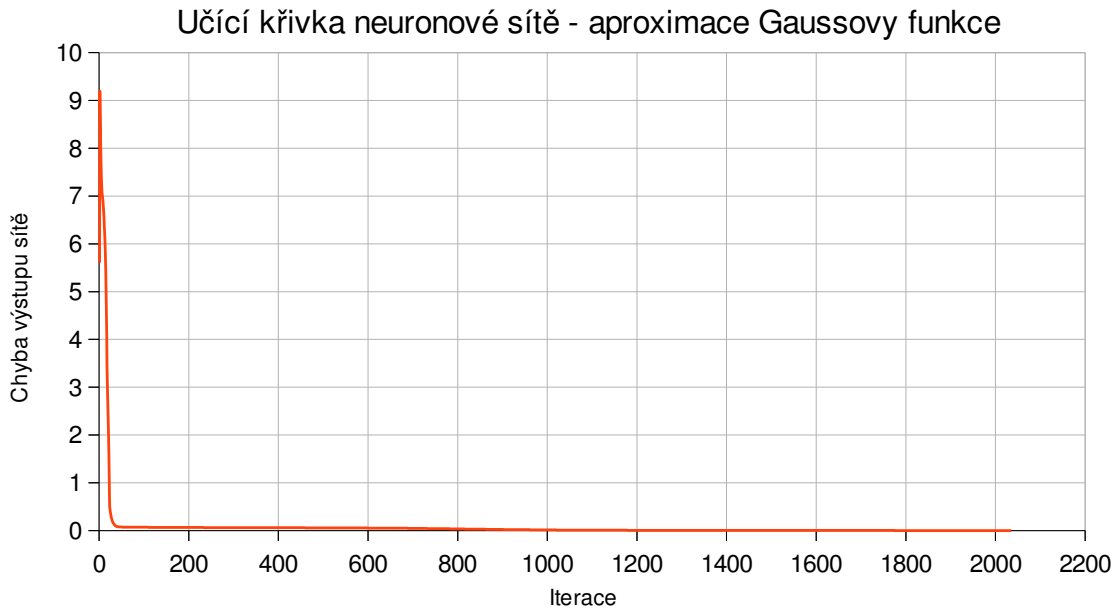
K aproximaci Gaussovy křivky bylo zvoleno 13 bodů (tab. 5.2). Dvanáctý a třináctý trénovací vzor byl přidán kvůli zmenšení chyby hodnoty funkce v bodě 0 na ose x. Velká chyba v tomto bodě vzniká, protože většina bodů ve trénovací množině dat leží podstatně níže než bod na vrcholu Gaussovy křivky. Z toho plyne, že chybu v tomto bodě lze minimalizovat správným návrhem trénovací množiny dat, zejména přidáním většího počtu bodů v oblasti kolem bodu 0 [5].

Koeficient učení α byl nastaven na hodnotu 0,3965. Pro inicializaci neuronové sítě náhodnými hodnotami vah, biasů a strmostí přenosových funkcí neuronů byly ponechány výchozí hodnoty seedů a rozsahů, s výjimkou strmostí přenosových funkcí neuronů, které byly vygenerovány v rozsahu od 1 do 3. Učení bylo zastaveno při poklesu chyby výstupu sítě pod 0,001. Učící křivka neuronové sítě je zobrazena na obrázku 5.7.

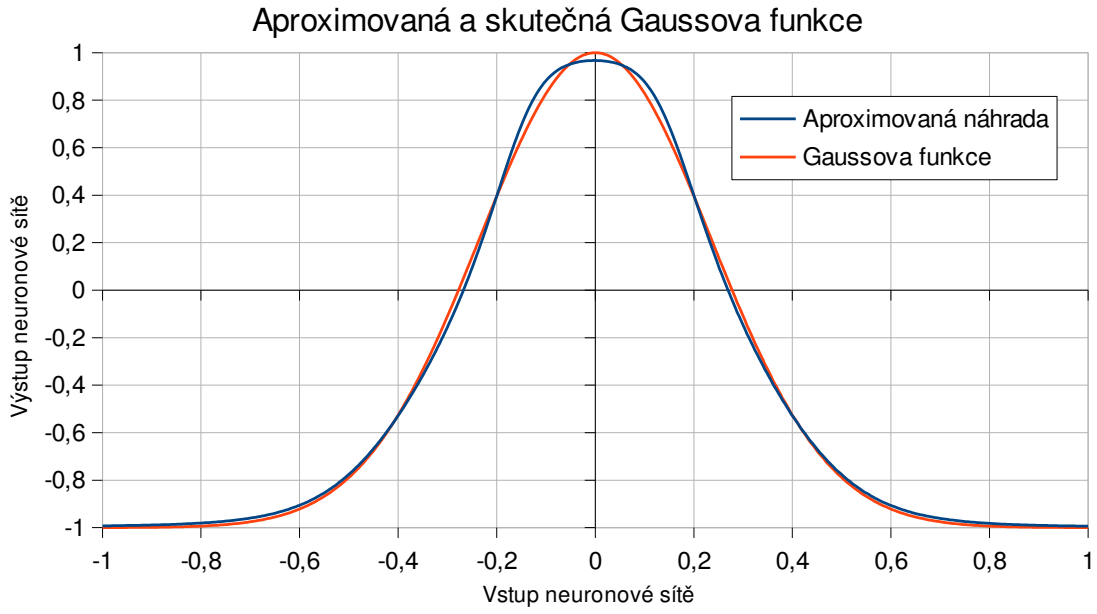
In 1	**Out 1**	**In 2**	**Out 2**	**In 3**	**Out 3**	**In 4**	**Out 4**	**In 5**	**Out 5**
-1	-0,9998	-0,8	-0,9937	-0,6	-0,9217	-0,4	-0,5261	-0,2	0,3954
In 6	**Out 6**	**In 7**	**Out 7**	**In 8**	**Out 8**	**In 9**	**Out 9**	**In 10**	**Out 10**
0	1	0,2	0,3954	0.4	-0,5261	0,6	-0,9217	0,8	-0,9937
In 11	**Out 11**	**In 12**	**Out 12**	**In 13**	**Out 13**	**-**	**-**	**-**	**-**
1	-0,9998	0,05	0,9555	-0,05	0,9555	-	-	-	-

Tab. 5.2: Aproximované body Gaussovy křivky

Gaussova funkce aproximovaná neuronovou sítí je zobrazena na obrázku 5.8. Z grafu je patrné, že největší chyba výstupu neuronové sítě leží podle předpokladu v bodě 0, také je vidět, že aproximovaná náhrada je mírně nesymetrická.



Obr. 5.7: Učící křivka neuronové sítě - aproximace Gaussovy funkce



Obr. 5.8: Gaussova funkce aproximovaná neuronovou sítí

5.7 Predikce časové řady

Potřeba předvídat budoucí hodnoty v časové řadě je velice běžný problém, ať už se jedná o předpověď počasí nebo předpověď pohybů cen akcií či jiných ekonomických dějů. Využití neuronové sítě má tu výhodu, že nemusíme volit žádný matematický model predikce. Stačí nám jen předložit neuronové síti množinu dat a ona sama si zvolí svůj vlastní model, který většinou odvede dobrou práci [5].

Zadání

Navrhněte, naučte a otestujte vícevrstvou neuronovou síť, která bude na základě předchozích čtyř bodů předpovídat následující bod sinusového průběhu

$$f(x) = sin(\pi x) \tag{5.28}$$

Řešení

Neuronová síť zvolená pro řešení zadaného úkolu má čtyři vstupy, jeden výstup a vnitřní vrstvu o třech neuronech. Vstupy sítě reprezentují známou čtveřici bodů sinusové funkce, podle které se bude předpovídat hodnota pátého, již neznámého bodu.

Trénovací množina dat zobrazená v tabulce 5.3 je sestavena z pětic bodů ležících na zadané funkci tak, že první čtyři body představují vstup sítě a pátý bod její výstup. Postupným posunem po ose X o hodnotu 0,1 bylo vytvořeno všech 21 trénovacích vzorů.

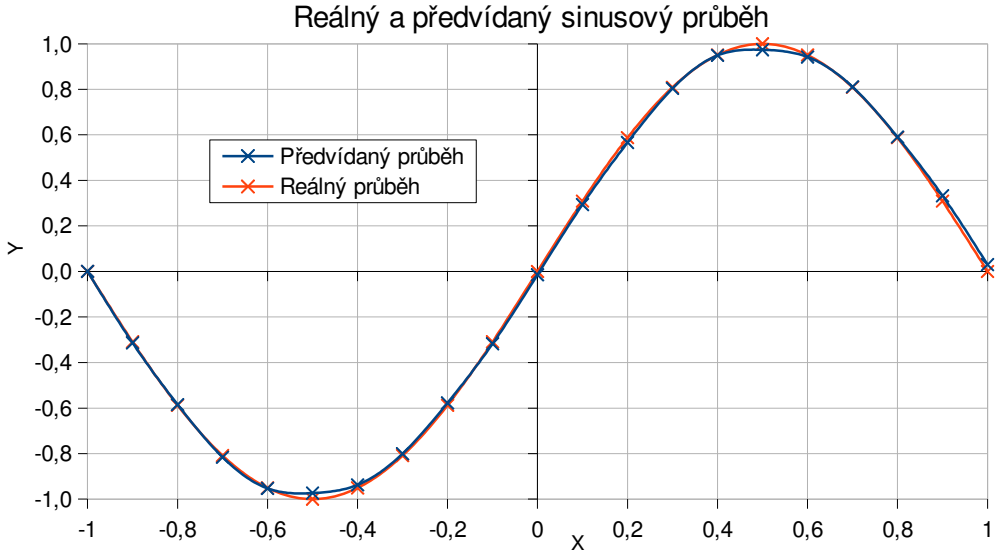
	1	**2**	**3**	**4**		**1**
In 1	0,9511	0,8090	0,5878	0,3090	**Out 1**	0,0000
In 2	0,8090	0,5878	0,3090	0,0000	**Out 2**	-0,3090
In 3	0,5878	0,3090	0,0000	-0,3090	**Out 3**	-0,5878
In 4	0,3090	0,0000	-0,3090	-0,5878	**Out 4**	-0,8090
In 5	0,0000	-0,3090	-0,5878	-0,8090	**Out 5**	-0,9511
In 6	-0,3090	-0,5878	-0,8090	-0,9511	**Out 6**	-1,0000
In 7	-0,5878	-0,8090	-0,9511	-1,0000	**Out 7**	-0,9511
In 8	-0,8090	-0,9511	-1,0000	-0,9511	**Out 8**	-0,8090
In 9	-0,9511	-1,0000	-0,9511	-0,8090	**Out 9**	-0,5878
In 10	-1,0000	-0,9511	-0,8090	-0,5878	**Out 10**	-0,3090
In 11	-0,9511	-0,8090	-0,5878	-0,3090	**Out 11**	0,0000
In 12	-0,8090	-0,5878	-0,3090	0,0000	**Out 12**	0,3090
In 13	-0,5878	-0,3090	0,0000	0,3090	**Out 13**	0,5878
In 14	-0,3090	0,0000	0,3090	0,5878	**Out 14**	0,8090
In 15	0,0000	0,3090	0,5878	0,8090	**Out 15**	0,9511
In 16	0,3090	0,5878	0,8090	0,9511	**Out 16**	1,0000
In 17	0,5878	0,8090	0,9511	1,0000	**Out 17**	0,9511
In 18	0,8090	0,9511	1,0000	0,9511	**Out 18**	0,8090
In 19	0,9511	1,0000	0,9511	0,8090	**Out 19**	0,5878
In 20	1,0000	0,9511	0,8090	0,5878	**Out 20**	0,3090
In 21	0,9511	0,8090	0,5878	0,3090	**Out 21**	0,0000

Tab. 5.3: Trénovací data pro předvídání sinusového průběhu

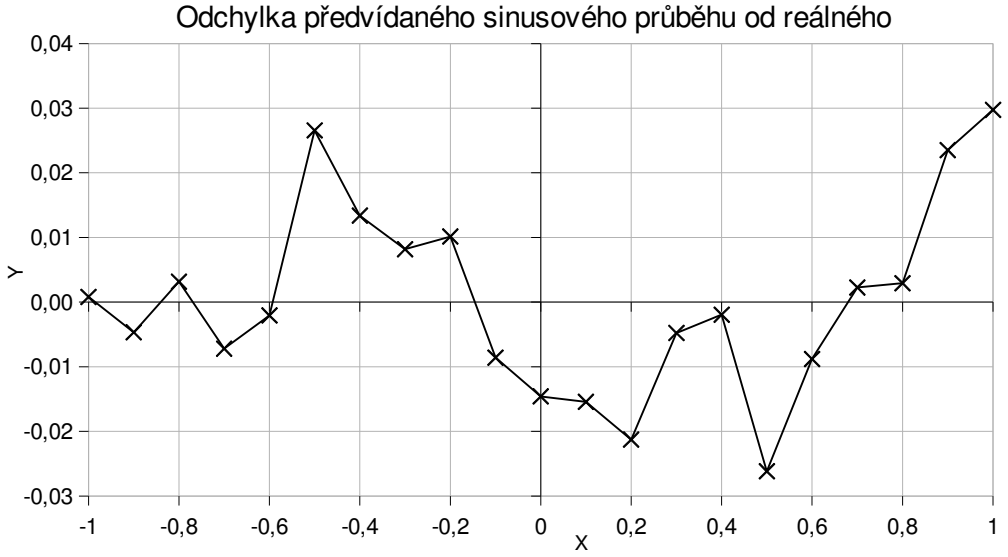
Jako přenosová funkce neuronů v neuronové síti byla nastavena bipolární sigmoida. Koeficient učení α byl nastaven na hodnotu 1. Ostatní volby byly ponechány na výchozích hodnotách.

Naučená neuronová síť byla otestována předvídáním jedné periody sinusové funkce. Body získané předchozím předvídáním byly zpětně předkládány neuronové síti, která s jejich využitím předvídala další body. Průběh sinusové funkce získaný předvídáním a průběh ideální, jsou zobrazeny v grafu na obrázku 5.9.

Na obrázku 5.10 je zobrazen rozdíl hodnot předvídaného a reálného průběhu, který se vzrůstajícím počtem předvídaných bodů stoupá.



Obr. 5.9: Reálný a předvídaný sinusový průběh



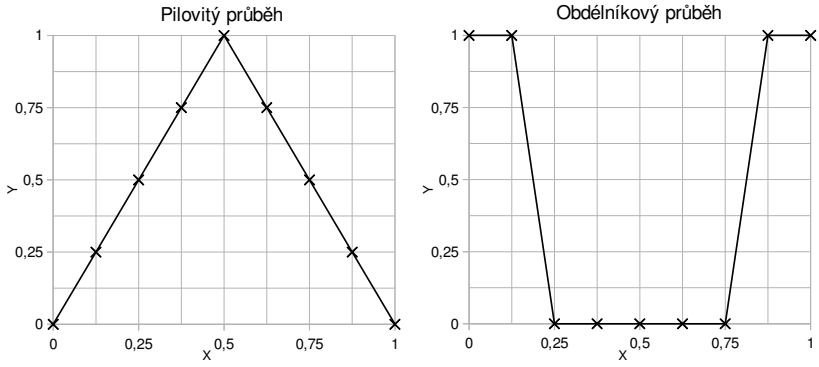
Obr. 5.10: Odchylka předvídaného sinusového průběhu od reálného

5.8 Rekonstrukce signálu

V tomto případě je neuronová síť navržena tak, aby reprodukovala svůj vstup jako výstup. Tento přístup má spoustu využití, z nichž mezi nejdůležitější patří filtry zvyšující kvalitu obrazu a filtrace šumu ze signálů [5].

Zadání

Navrhněte a vytvořte jednoduchý filtr pro rekonstrukci signálů na obrázku 5.11. Každý ze signálů popište devíti vzorky. Vytvořte pouze jeden filtr, který bude pracovat s oběma signály a tento filtr otestujte poškozenými signály dle vaší úvahy.



Obr. 5.11: Zadané signály pro tvorbu filtru

Řešení

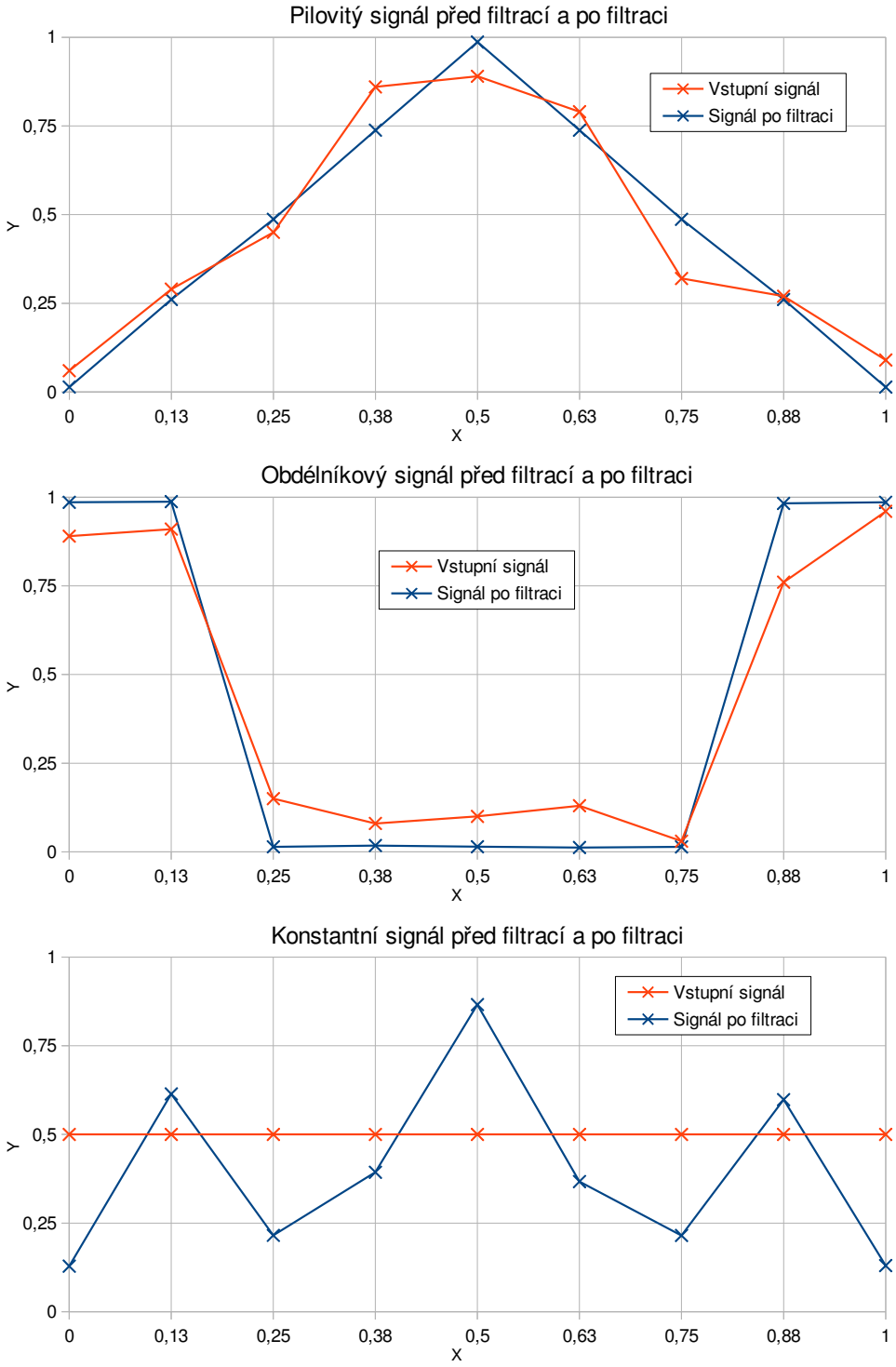
K vypracování úlohy byla zvolena neuronová síť s jednou vnitřní vrstvou se čtyřmi neurony, devíti vstupy a devíti výstupy. Počet vstupů a výstupů neuronové sítě odpovídá počtu vzorků zadaného signálu.

Neuronová síť byla trénována tak, aby reprodukovala svůj vstup jako svůj výstup. Byly vytvořeny dva trénovací vzory, přičemž hodnoty vstupů i výstupů v těchto vzorech jsou shodné a odpovídají hodnotám bodů na zadaných průbězích.

Před začátkem učení byla provedena inicializace neuronové sítě náhodnými hodnotami vah, biasů a strmostí přenosových funkcí neuronů s použití výchozího nastavení programu. Učení bylo ukončeno při dosažení maximální chyby sítě o hodnotě 0,001. Koeficient učení α byl ponechán na přednastavené hodnotě 0,5.

Naučená neuronová síť byla otestována poškozenými signály podobného tvaru jako signály zadané (obr. 5.11). Z grafů na obrázku 5.12 je patrné, že pro pilovitý i obdélníkový průběh došlo ke značnému zlepšení. Naopak u konstantního signálu došlo ke značnému zkreslení. Toto zkreslení je způsobeno tím, že neuronová síť je naučená pouze na filtraci pilovitého a obdélníkového signálu a konstantní signál

se snaží v různých bodech upravit na signál obdélníkový nebo pilovitý. Tomuto problému lze předejít přidáním dalších trénovacích vzorů, které neuronovou síť naučí filtrovat konstantní signál.



Obr. 5.12: Signály před filtrací a po filtraci neuronovou sítí

6 ZÁVĚR

Hlavním cílem práce bylo vytvoření programu pro výuku, který umožňuje návrh, učení a testování vícevrstvých neuronových sítí s algoritmem učení backpropagation. Dalším cílem práce je souhrn teorie použité při vývoji tohoto programu a tvorba příkladů do výuky.

V první kapitole jsem provedl rozbor teorie nutné pro vývoj programu. V první části teoretického rozboru jsem popsal funkci jednoho neuronu a jeho nejběžněji používané přenosové funkce při učení pomocí algoritmu backpropagation. V rozboru jsem dále vysvětlil funkci vícevrstvé neuronové sítě a princip fungování algoritmu backpropagation.

Pro účely výuky a jako náhradu staršího programu BPSIM jsem v rámci této bakalářské práce vytvořil program Neural network Creator. Rozvržení grafického uživatelského rozhraní a princip jeho ovládání je inspirován vývojovým prostředím QtCreator. Vzniklo tak dostatečně komplexní, moderní a přitom stále velmi přehledné grafické rozhraní, které je vhodné jak pro výuku a experimentování, tak i pro praktické použití.

Na rozdíl od ostatních programů (viz rešerše v kapitole 3), má Neural network Creator přehlednou správu projektů, díky které může uživatel pracovat i s větším počtem datových množin a neuronových sítí. Program také nabízí možnosti grafického zobrazení výstupu pro neuronovou síť s jedním až třemi vstupy a klasifikačního diagramu pro neuronovou síť se dvěma vstupy. Součástí programu je také rozsáhlá nápověda, která obsahuje teoretické základy potřebné pro práci s programem, návod k ovládání programu a výukové příklady včetně jejich řešení.

Výukové příklady jsem navrhl pro účely výuky se zaměřením na praktické ověření teoretických znalostí vícevrstvých neuronových sítí a jako ukázku možností využití programu. První tři příklady doplňují teorii a ukazují výpočet výstupu neuronové sítě, adaptaci vah pomocí algoritmu backpropagation a výpočet chyby výstupu sítě. Zbylých pět příkladů bylo vypracováno v programu Neural network Creator a demonstruje možnosti použití vícevrstvé neuronové sítě.

V budoucnosti plánuji vytvoření webových stránek, kde bude možné stáhnout aktuální verzi programu, včetně jeho dokumentace. Zdrojové kódy programu budou zveřejněny ve formě git repozitáře. Plánován je také překlad nápovědy do anglického jazyka a vytvoření instalátoru programu pro operační systém Windows a instalačních balíčků pro vybrané distribuce Linuxu.

V plánu je také využití technologií OpenCl nebo CUDA pro zrychlení učení, přidání dalších typů grafů umožňujících zobrazit výstupy neuronových sítí určených k predikci časové řady či klasifikaci dat a implementace většího množství automatických testů programu.

LITERATURA

[1] All Modules. DIGIA. *Qt Project* [online]. [cit. 2013-04-28]. Dostupné z: http://qt-project.org/doc/qt-4.8/modules.html

[2] BALAZS, Zoltan. A Practical Use of the MVC Pattern. *CodeProject - For those who code* [online]. 8.1.2007 [cit. 2013-04-27]. Dostupné z: http://www.codeproject.com/Articles/17068/A-Practical-Use-of-the-MVC-Pattern

[3] BERNARD, Borek. Úvod do architektury MVC. *Zdroják / o tvorbě webových stránek a aplikací* [online]. 7.5.2009 [cit. 2013-04-27]. Dostupné z: http://www.zdrojak.cz/clanky/uvod-do-architektury-mvc

[4] FAUSETT, Laurene. *Fundamentals of Neural Networks: Architectures, Algorithms And Applications*. Prentice-Hall, 1994. ISBN 978-0-13-334186-7.

[5] MASTERS, Timothy. *Practical Neural Network Recipes in C++*. San Francisco: Morgan Kaufmann, 1993. ISBN 978-0-12-479040-7.

[6] MATHWORKS. *Neural Network Toolbox - MATLAB* [online]. [cit. 2013-05-10]. Dostupné z: http://www.mathworks.com/products/neural-network/

[7] Model/View Programming. DIGIA. *Qt Project* [online]. [cit. 2013-04-27]. Dostupné z: http://qt-project.org/doc/qt-4.8/model-view-programming.html

[8] NEURAL PLANNER SOFTWARE. *JustNN Help* [online]. [cit. 2013-05-10]. Dostupné z: http://www.justnn.com/application/window_2_3.htm

[9] UNIVERSITY OF STUTTGART. *Stuttgart Neural Network Simulator: User Manual, Version 4.1* [online]. 1995 [cit. 2013-05-10]. Dostupné z: http://www.ra.cs.uni-tuebingen.de/SNNS/UserManual/UserManual.html

[10] VOLNÁ, Eva. *Neuronové sítě 1*. Ostrava, 2002. Učební texty. Ostravská Universita, Přírodovědecká fakulta.