VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

VIZUÁLNÍ SIMULÁTOR OBECNÝCH NEURONOVÝCH SÍTÍ

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE DAVID HERMAN
AUTHOR

BRNO 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

VIZUÁLNÍ SIMULÁTOR OBECNÝCH NEURONOVÝCH SÍTÍ

VISUAL SIMULATOR OF GENERAL NEURAL NETWORKS

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE DAVID HERMAN
AUTHOR

VEDOUCÍ PRÁCE ING. DAVID MARTINEK
SUPERVISOR

BRNO 2008

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2007/2008

Zadání bakalářské práce

Řešitel: **Herman David**
Obor: Informační technologie
Téma: **Vizuální simulátor obecných neuronových sítí**
Kategorie: Umělá inteligence

Pokyny:

1. Prostudujte různé typy neuronových sítí a algoritmy pro jejich učení.
2. Navrhněte obecnou knihovnu pro práci s neuronovými sítěmi. Dále navrhněte aplikaci pro vizuální práci s různými typy sítí, která bude spolupracovat s navrženou knihovnou. Tato aplikace musí být použitelná zejména pro demonstraci algoritmů učení. To znamená, že musí být schopna simulovat tyto algoritmy po jednotlivých krocích a vhodným způsobem graficky reprezentovat stav celé sítě.
3. Navrženou knihovnu a aplikaci implementujte. Vytvořte vhodné demonstrační příklady, které předvedou možnosti vytvořeného systému.
4. Zhodnoťte dosažené výsledky. Diskutujte možnosti dalšího rozšíření.

Literatura:

- Podle pokynů vedoucího

Při obhajobě semestrální části projektu je požadováno:

- První dva body zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Martinek David, Ing.**, UITS FIT VUT
Datum zadání: 1. listopadu 2007
Datum odevzdání: 14. května 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. **Pan**

Jméno a příjmení: **David Herman**
Id studenta: 79006
Bytem: Palírenská 223/13, 621 00 Brno
Narozen: 06. 07. 1985, Brno
(dále jen "autor")

a

2. **Vysoké učení technické v Brně**

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
   bakalářská práce

Název VŠKP: Vizuální simulátor obecných neuronových sítí
Vedoucí/školitel VŠKP: Martinek David, Ing.
Ústav: Ústav inteligentních systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:
tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☐ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

.....	.....
Nabyvatel	Autor

Abstrakt

Předmětem této bakalářské práce je navržení obecné knihovny pro práci s neuronovými sítěmi. Dále pak implementace vizuálního simulátoru, který bude vhodně graficky reprezentovat po jednotlivých krocích algoritmy učení i aktivní dynamiku sítě. Mezi požadavky rovněž patří platformní nezávislost aplikace.

Klíčová slova

neuron, neuronová síť, backpropagation síť, perceptron síť, Kohonenova síť, obecná knihovna pro práci s neuronovými sítěmi, simulace dynamik neuronové sítě, vizualizace neuronové sítě, formalizmus pro popis neuronových sítí, programování

Abstract

The subject of this bachelor thesis is the design of a general ibrary of neural networks. Another subject is the implementation of a visual simulator, which would represent graphically, in a suitable manner, the algorithm of learning and the active dynamics of the network, in separate steps. This application also has to be platform independent.

Keywords

neuron, neural network, backpropagation network, perceptron network, Kohonen network, general library of neural network, simulation dynamic of neural network, vizualization of neural network, formalism for describe neural network, programming

Citace

David Herman: Vizuální simulátor obecných neuronových sítí, bakalářská práce, Brno, FIT VUT v Brně, 2008

Vizuální simulátor obecných neuronových sítí

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Davida Martinka a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

……
David Herman
12. května 2008

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Davidu Martinkovi za příkladnou metodickou, pedagogickou a odbornou pomoc a za další cenné rady při zpracování mého bakalářského projektu.

© David Herman, 2008.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

Obsah ... 1
1 Úvod ... 3
1.1 Cíle práce ... 4
1.2 Přehled kapitol ... 4
2 Teoretický úvod do neuronových sítí ... 6
2.1 Biologický neuron ... 6
2.2 Umělé neuronové sítě ... 7
3 Analýza a formalizmus ... 10
3.1 Matematický model neuronové sítě ... 10
3.2 Simulace ... 13
3.3 Vizualizace ... 13
3.4 Modulová rozšiřitelnost ... 14
4 Návrh řešení ... 15
4.1 Obecná koncepce ... 15
4.2 Jádro ... 15
4.3 Návrh obecné knihovny ... 16
4.4 Simulace ... 19
4.5 Návrh vizuálního simulátoru ... 20
5 Popis řešení ... 22
5.1 Implementace ... 22
5.2 Ovládání vizualizátoru ... 22
6 Popis implementovaných sítí ... 25
6.1 Síť perceptron ... 25
6.2 Síť backpropagation ... 26
6.3 Kohonenova síť ... 27
7 Závěr ... 30
Literatura ... 32
Seznam příloh ... 33
Příloha A Ukázka tvorby nového modulu ... 34
Implementace nové sítě ... 34
Vytvoření modulu ... 35
Připojení modulu k simulátoru ... 35

1 Úvod

Inteligence již v dnešní době není chápána jen jako vlastnost některých živých organismů. Ruku v ruce s rozvojem výpočetní techniky si lidé kladli otázku, zda je možné i u uměle vytvořených systémů dosahovat reakcí, které bychom kvalifikovali v relaci s živými organismy za projev inteligence. Obligátní otázku, jestli mohou stroje myslet, řešili již během 17. století filosofové jako Pascal, Descartes a Hobbes.

Od 30. let XX. století se objevují snahy o automatizované řešení problémů. Postupně jsou navrhovány metody, postupy a algoritmy napodobující určité hledisko inteligentního chování. Všechny tyto oblasti jsou předmětem relativně mladé vědní disciplíny Umělá inteligence.

Jedním z mnoha odvětví Umělé inteligence je fenomén neuronových sítí, které měli relativně dramatický rozvoj, ať už v podobě do nich vkládaných idejí či pohledu na ně (připomínali tak trochu alchymii). Za počátek vzniku tohoto oboru se považuje publikace Warrnera McCullocha a Waltera Pittse z roku 1943. Jejich práce ukázala možnost aplikovat jednodušší neuronové typy síti na libovolnou logickou nebo aritmetickou funkci. Své pozornosti neunikly neuronové sítě ani u velikánů té doby jako byl například John von Neumann, jenž se jimi inspiroval při návrhu počítačů. Avšak 40. a 50. léta nepřinesla žádný stěžejní pokrok v oblasti neurovýpočtů.

Za průkopnickou práci můžeme považovat vynalezení tzv. perceptronu Frankem Rosenblattem roku 1957, který se stal zobecněním McCullochova a Pittsova modelu neuronu. Rosenblatt též napsal jednu z prvních knih o neurovýpočtech Principles of Neurodynamics. Krátce po objevu perceptronu přišel Bernard Widrow se svými studenty a vyvinul další typ neuronového výpočetního prvku nazvaného ADALINE (ADAptive LINear Element). Mezi 50. a 60. lety dochází obecně k úspěšnému návrhu nových modelů neuronových sítí a jejich implementaci v podobě neuropočítačů.

Tvrdou ránu dynamickému rozvoji neuronových sítí zasadil Marvin Minsky se Seymourem Papertem, kteří uplatnili svůj vliv, aby diskreditovali výzkum neuronových sítí. Minsky a Papert využili pro svoji argumentaci známého faktu, že jeden perceptron není schopný počítat triviální logickou funkci vylučovací disjunkci (XOR). Kampaň Minského a Paperta byla úspěšná, výzkum neuronových sítí nebyl několik let dále dotován a byl označen za neperspektivní.

Renesance zájmu o neuronové sítě přichází v 80. letech. Od roku 1987 založilo mnoho renomovaných univerzit nové výzkumné ústavy zabývající se touto problematikou. Nastolený trend pokračuje až do dnešních dob, kdy se bohužel opět začíná zdát, že široký záběr výzkumu a vložené investice neodpovídají kvalitě dosažených výsledků.

1.1 Cíle práce

Pro snazší výuku a pochopení dějů probíhajících uvnitř neuronových sítí je užitečné mít nástroje, jež vhodným způsobem dokáží simulovat a prezentovat jednotlivé stavy sítí. Tato bakalářská práce si klade za cíl navrhnout obecnou knihovnu pro práci s neuronovými sítěmi a vizuální simulátor, který bude vhodně graficky reprezentovat po jednotlivých krocích algoritmy učení i aktivní dynamiku sítě. Mezi požadavky rovněž patří platformní nezávislost aplikace.

Pojmu obecnost byla dána větší váha, než zadání vyžaduje. Obecnost je chápána hned na několika úrovních. Především se jedná o obecnost samotné knihovny pro práci s neuronovými sítěmi disponujícími takovými prostředky, aby dokázala popsat co nejširší škálu sítí. Dále obecnost vizuálního simulátoru, tedy schopnost zobrazit a simulovat libovolný typ sítě. Rovněž obecnost jako možnost dynamického rozšíření simulátoru o nové typy sítí (modulová rozšiřitelnost). V neposlední řadě obecnost ve smyslu využití sítě odvozené z obecné knihovny i mimo vizuální simulátor. A konečně obecnost jako synonymum jednoduché tvorby nových druhů sítí bez nutnosti hlubších znalostí vizuálního simulátoru.

Tato práce navazuje na Semestrální projekt 2007/2008. V rámci semestrálního projektu byly nastudovány různé typy sítí a představen použitý formalizmus popisující neuronové sítě. Tento formalizmus se stal posléze základem objektového návrhu obecné knihovny pro práci s neuronovými sítěmi. V semestrální práci též byly položeny základní stavební kameny vizuální části simulátoru, bylo navrženo a implementováno jádro a nastíněno řešení samotné komunikace mezi vizualizátorem a obecnou knihovnou.

1.2 Přehled kapitol

Druhá kapitola je stručným teoretickým úvodem do problematiky neuronových sítí. Připravuje živnou půdu pro ty, kteří se s problematikou neuronových sítí doposud nesetkali a chtějí se seznámit se základními principy, ale hlavně pochopit podobnost a inspiraci v biologických systémech. Tato část nevyžaduje předběžné hlubší znalosti. Formalizmus je zde redukován na co nejmenší možnou míru.

Kapitola *Analýza a formalizmus* se snaží objasnit fenomén neuronových sítí. Vede k formalizaci modelu neuronu a neuronové sítě, jejich parametrů a množin vzorů. Ukazuje, že jakýkoliv model neuronové sítě lze popsat pomocí její aktivní, adaptační a organizační dynamiky. Představuje jednoduchý matematický aparát pro definování velké škály sítí z hlediska organizační

dynamiky, jež je fundamentem objektového návrhu obecné knihovny. Taktéž se zabývá analýzou samotné problematiky simulace, vizualizace a modulové rozšiřitelnosti aplikace.

Ve čtvrté kapitole nazvané *Návrh řešení* jsou uplatněny poznatky z předchozích dvou kapitol. Podrobně rozebírá stěžejní prvky při realizaci vizuálního simulátoru, prezentuje návrh samotného aplikačního jádra a obecné knihovny pro práci s neuronovými sítěmi. Nemalá část je též věnována konceptu pro snadnou simulaci aktivní či adaptivní dynamiky sítě.

Pátá kapitola se věnuje popisu řešení. Zdůvodňuje zvolení jazyka C++ a multiplatformního toolkitu Qt pro implementaci. Mimo to je rovněž stručnou uživatelskou příručkou k ovládání vizuálního simulátoru.

Spolu s obecnou knihovnou a vizuálním simulátorem byly implementovány tři ukázkové moduly neuronových sítí (perceptron, backpropagation a Kohonen), které demonstrují funkčnost navržené koncepce. Kapitola šestá je lehkým teoretickým úvodem do těchto sítí a představuje jejich konkrétní implementaci.

Poslední kapitola obsahuje zhodnocení dosažených výsledků a diskutuje možnosti dalších rozšíření simulátoru majících za cíl ještě názornější a důvtipnější prezentaci dějů probíhajících uvnitř neuronových sítí.

2 Teoretický úvod do neuronových sítí

Prvotní hnací silou tvorby modelů neuronových sítí byla snaha pochopit, jakým způsobem pracuje lidský mozek. Ze získaných neurofyziologických poznatků byly vytvořeny zjednodušené matematické modely, které lze použít při řešení úloh z Umělé inteligence. Většina dnešních aplikací umělých neuronových sítí využívá jejich selektivních a generalizačních vlastností. S postupem času byly umělé neuronové sítě dále rozvíjeny bez ohledu na jejich původní záměr. Přesto je velmi užitečné poukázat na analogii s biologickými systémy a využít ji při samotném matematickém popisu umělého neuronu. Další podrobnosti v [1], [2], [3] a [4].

2.1 Biologický neuron

Základním funkčním a histologickým prvkem nervové tkáně je nervová buňka neboli neuron. Jedná se o velice specializované buňky schopné přijmout, vést, zpracovat a odpovědět na vnější signály. Kombinace velkého množství těchto elementárních prvků s danými vlastnostmi pak dávají schopnost organismu reagovat na vnější i vnitřní podněty a pamatovat si. Jen mozková kůra člověka je odhadem tvořena 13 až 15 miliardami těchto buněk, přičemž každá z nich může být spojena až s 5 000 dalšími. Neuronovou buňku poprvé popsal roku 1835 J.E. Purkyně.

2.1.1 Tělo neuronu

Části neuronu, kde se nachází jádro neuronové buňky, říkáme tělo neuronu. Tělo neuronu se člení na perikaryon, neurocyt a somatu. Jádro neuronu je poměrně velké, zpravidla kulovitého nebo oválného tvaru a je spojeno s okolím pomocí tzv. výběžků.

2.1.2 Výběžky neuronu

Neuron je uzpůsoben pro přenos signálu výběžky, jež jsou dvojího typu. Jedná se o krátké dendrity a dlouhé neurity neboli axony. Z axonu obvykle odbočují větve, tzv. terminály, zakončené blánou. Blána má funkci pojítka s trny, výběžky dendritů jiných neuronů. K šíření vzruchu dochází pak díky unikátnímu mezineuronovému rozhraní, synapsi.

2.1.3 Synapse neuronu

Komunikace neuronů je postavena na vysoce specializovaných strukturách zvaných synapse. Akční potenciál šířící se po povrchu neuronu vyvolá uvolnění specifických látek, mediátorů, do synaptické štěrbiny (prostor mezi dvěma neurony). Díky mediátoru může dojít ke vzniku dalšího akčního

potenciálu šířícího se na příslušný neuron. Z funkčního hlediska lze synaptická propojení rozdělit na exhibiční, které podporují šíření vzruchu, a inhibiční, jež naopak vzruch tlumí. Po každém průchodu vzruchu se synaptické váhy mění, rekonfigurují, což je předpokladem paměťové schopnosti neuronové buňky. Také organizace propojení mezi samotnými neurony sítě doznává během života sítě neustálé změny. Ve fázi učení se vytvářejí nové synaptické stopy, v případě zapomínání se ruší. Synaptická propustnost, respektive její míra, je význačnou informací během existence neuronové sítě.

2.2 Umělé neuronové sítě

Umělé neuronové sítě se skládají z umělých funkčních bloků, neuronů, které jsou vzájemně propojeny. Obecně je výstup jednoho neuronu vstupem více neuronů, a to podobně jako jsou terminály axonu biologického neuronu přes synaptické váhy spojeny s dendrity jiných neuronů. Počet neuronů v síti a jejich vzájemné propojení je jedním z charakteristických vlastností sítě a určuje její topologii (architekturu). Při vnějším pohledu můžeme uvnitř sítě nalézt shluky neuronů neboli vrstvy, což jsou skupiny neuronů mající stejné umístění z hlediska topologie sítě. Těmto vrstvám dle funkčního významu dáváme přívlastky vstupní, skrytá (pracovní) a výstupní. Obdobně se nazývají neurony patřící do daného typu vrstvy (vstupní neuron, skrytý neuron a výstupní neuron). Šíření a zpracování informace uvnitř sítě je umožněno změnou stavů neuronů ležících na cestě, po níž se informace přenáší. Stavy všech neuronů v síti a jejich synaptických vah definují tzv. konfiguraci neuronové sítě.

Neuronová síť jako celek se v čase svého života mění. V této souvislosti lze rozdělit celkovou dynamiku sítě do tří podskupin a uvažovat tak tři funkční režimy sítě: organizační (změna topologie), aktivní (změna stavu) a adaptivní (změna konfigurace). Konkretizací jednotlivých dynamik získáme různé modely neuronových sítí vhodné pro řešení určitého typu úloh. Další podrobnosti v [1] a [2].

2.2.1 Organizační dynamika

Organizační dynamika sítě určuje topologii sítě a její případnou změnu. Topologie se zpravidla mění jen v rámci adaptivního režimu a to tím způsobem, že síť je v případě potřeby rozšířena o další neurony či neuronové spoje. Avšak ve většině situacích má organizační dynamika sítě spíše statický charakter.

V zásadě mluvíme o dvou typech architektury: cyklická (rekurentní) a acyklická (dopředná) topologie. Cyklická architektura se vyznačuje existencí kruhového spojení v rámci jedné skupiny neuronů. Nejjednodušším příkladem tohoto uskupení budiž zpětnovazební neuron, kde výstup neuronu je zároveň jeho vstupem. U dopředných sítích naopak všechny cesty vedou jedním směrem

a můžeme u nich neurony vždy disjunktně rozdělit do tzv. vrstev. Typickým zástupcem takového uspořádání je vícevrstvá neuronová síť.

2.2.2 Aktivní dynamika

Aktivní dynamika popisuje nejen počáteční stav sítě, ale také způsob změny v čase při pevně dané konfiguraci a topologii. V aktivním režimu se zpočátku nastaví stavy neuronů uvnitř vstupní vrstvy na vstupní vektor sítě. Všechny možné vstupní vektory reprezentují tzv. vstupní prostor neuronové sítě. Po inicializaci probíhá vlastní výpočet. Protože výpočetní prostředky, na kterých se sítě modelují, jsou výpočetně časově diskrétní systémy, předpokládá se i v aktivní dynamice sítě diskrétní čas. Avšak obecně má síť vlastnosti spojitého modelu a její stav můžeme tedy charakterizovat spojitou funkcí.

V každém časovém kroku se uplatní aktivní pravidlo dynamiky. Pokud je vybrán pouze jeden neuron měnící svůj stav, mluvíme o sekvenčním výpočtu, jinak o výpočtu paralelním. Modely sítě lze rozdělit ještě dle toho, zda mění svůj stav nezávisle (asynchronní modely) anebo jsou řízeny centrálně (synchronní modely). Stavy výstupních neuronů sítě představují výsledek výpočtu. Obecně je funkce neuronové sítě dána aktivní dynamikou, jejíž rovnice závisí na neměnné konfiguraci a topologii v aktivním režimu.

Aktivní dynamika sítě též definuje funkci jednoho neuronu. Pakliže je matematický předpis pro všechny neurony v síti stejný, hovoříme o tzv. homogenní neuronové síti. Homogenní struktury jsou z hlediska uplatnění majoritními.

2.2.3 Adaptivní dynamika

Definice adaptivní dynamiky sítě vychází z počáteční konfigurace a ze způsobu, kterým dochází ke změnám vah neuronů v čase. Cílem adaptivní dynamiky je nalézt takovou konfiguraci sítě ve váhovém prostoru, jež bude v aktivním režimu realizovat chtěnou funkci. Řekneme-li, že aktivní režim sítě se používá pro vlastní výpočet funkce na daný vstup, pak adaptivní režim sítě slouží k naučení se této funkce. Učení sítě může být i pro menší úlohy časově náročné. Představuje většinou složitý nelineární optimalizační problém. Vyžadovaná funkce sítě je zpravidla zadána tréninkovou množinou. Samotné učení probíhá předkládáním vstupních vzorů síti, vyhodnocením těchto vstupů a následnou rekonfigurací sítě ve snaze dosáhnout lepší odezvy. Existují dva typy adaptace, a to adaptace s pomocí učitele a adaptace bez učitele.

Učení s učitelem je úzce podobné klasickému učení žáka ve škole, kde učitel plní funkci arbitra rozhodujícího o správnosti výsledku dodaného řešení. V některých případech učitel hodnotí kvalitu momentální odezvy sítě na daný vzorový vstup pomocí známky, pak mluvíme o tzv. klasifikovaném učení. Typickým představitelem tohoto typu sítí je například síť zpětného šíření.

Učení bez učitele patří k jinému typu adaptace. V případě takového tréninku obsahuje tréninková množina vzorů jen vstupy sítě. Neuronová síť v adaptivním režimu sama organizuje tréninkové vzory a hledá jejich souborné vlastnosti. Právě proto jsou tyto sítě taktéž nazývány sítěmi samoorganizujícími. Příkladem samoorganizující struktury je Kohonenova síť.

3 Analýza a formalizmus

Navržení obecné knihovny pro práci s neuronovými sítěmi s sebou nutně nese požadavek na studium široké palety odlišných typů sítí. V této studii je následně třeba nalézt společné prvky a vytvořit co nejvíce obecný formalizmus pro jejich popis. V závěru kapitoly se nachází rovněž analýza problémů spjatých se simulací, grafickou reprezentací a modulovou rozšiřitelností aplikace.

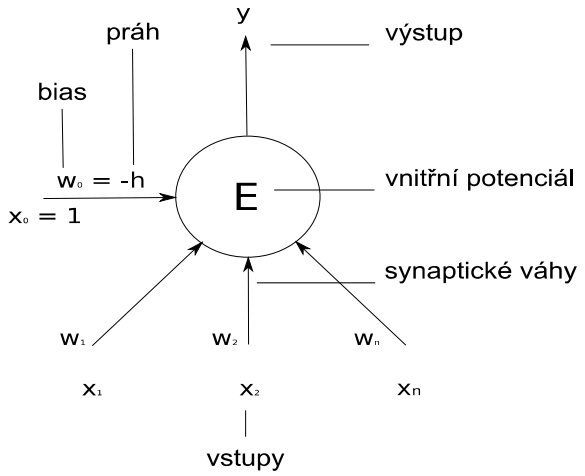
3.1 Matematický model neuronové sítě

Na definování široké škály neuronových sítí musíme mít dostatečně flexibilní matematický aparát. Formalizace neuronové sítě představuje v konečném důsledku jednoduchý matematický popis samotného modelu, a to z hlediska aktivní, adaptivní a organizační dynamiky. Alfou a omegou modelu je pojem formální neuron (viz [2]).

3.1.1 Formální neuron

Fundamentem matematického modelu neuronové sítě bude formální neuron. Formální neuron staví na zjednodušeném matematickém popisu funkce neurofyziologického neuronu. Strukturu formálního neuronu (dále jen neuronu) ilustruje obrázek 1. Neuron má n obecně reálných vstupů $x_1 \ldots x_n$, které jsou synonymem pro dendrity v biologickém neuronu. Každý vstup neuronu je ohodnocen reálnou synaptickou váhou $w_1 \ldots w_n$, jež definuje její propustnost. Inhibiční charakter synaptické vazby vyjadřuje záporná hodnota.

Vnitřní potenciál neuronu představuje *vážená suma* dílčích vstupních hodnot. Jestliže hodnota vnitřního potenciálu přesáhne tzv. *prahovou hodnotu h*, indukuje se výstup neuronu. Toto chování modeluje elektrický impuls axonu u biologického neuronu. Jednoduchou úpravou lze docílit toho, že vlastní práh neuronu budeme chápat jako zápornou váhu dalšího formálního vstupu s konstantní jednotkovou hodnotou. Aktivační (přenosová) funkce neuronu způsobuje nelineární nárůst výstupní hodnoty neuronu při dosažení prahové hodnoty potenciálu. Existují různé typy přenosových funkcí. V této práci se setkáme s těmito *sigmoidními aktivačními funkcemi*: *ostrá nelinearita*, *standardní sigmoida* a *hyperbolický tangens*.



Obrázek 1: Formální neuron

3.1.2 Formalizace neuronových sítí

Formalizace neuronových sítí není horkou novinkou v oblasti neurovýpočtů. Nicméně nalezení jednoduchého aparátu k popisu různých variací neuronových sítí může být obtížné. Jako základ formalizace byla použita zjednodušená verze autorů Rogerse a Satyadase (lze nalézt v [5] a [6]). Ta dosahuje solidní komplexnosti a je zároveň i dostatečně snadnou a vhodnou volbou pro použití v kontextu s návrhem obecné knihovny.

Nechť *NN* je neuronová síť, *S* množina vzorů (použita pro trénování, testování či ostatní operace se sítí), *P* parametry sítě a *T* síťovou topologii, pak neuronovou síť definujeme uspořádanou trojicí:

$$NN = (S, P, T) \quad . \tag{1}$$

3.1.2.1 Formalizace množiny vzorů

Součástí formalizace neuronové sítě nemusí být nutně formalizace množiny vzorů *S*. Přesto je vhodné ji do formalizace sítě integrovat pro jednodušší definici trénovací operace a samotné aktivní dynamiky sítě. V opačném případě by bylo začlenění pravidel týkajících se práce s trénovacími vzory daného typu sítě obtížné.

Formalizace množiny vzorů je velmi jednoduchá. Buď *S* množinou všech vzorů, *I* množinou všech vstupních vzorů a *O* množinou jim odpovídajících výstupních vzorů. Pak *S* definujeme dvouprvkovou množinou:

$$S = \{I, O\} \quad . \tag{2}$$

Množinu vstupních vzorů definujeme:

$$I=\{p_{k,j}\} \quad , \tag{3}$$

kde k je indexem daného výstupního vzoru a j indexem jeho komponenty.

Množinu výstupních vzorů definujeme:

$$O=\{o_{k,j}\} \quad , \tag{4}$$

kde k je indexem daného výstupního vzoru a j indexem jeho komponenty.

Poznamenejme, že samoorganizující sítě (např. Kohonenova síť) nevyžadují přítomnost výstupní množiny vzorů. V takovém případě tato množina obsahuje nulový počet prvků.

3.1.2.2 Formalizace parametrů sítě

Množina parametrů sítě P zahrnuje různá charakteristická kritéria sítě aplikovaná při učení, testování nebo jiných operacích. Parametry sítě jsou zpravidla konstanty, nicméně se může jednat o funkce závislé na čase či jiných vlastnostech sítě.

$$P=\{p_1, p_2, \dots, p_n\} \tag{5}$$

3.1.2.3 Formalizace topologie neuronových sítí

Architekturu neuronové sítě T definujeme jako uspořádanou dvojici prvků fyzické struktury sítě F a interních vazeb L mezi uzly sítě.

$$T=(F, L) \tag{6}$$

Strukturou sítě F míníme množinu shluků nebo vrstev v síti.

$$F=\{c_1, c_2, \dots, c_n\} \tag{7}$$

Shluk je množinou uzlů n, kde každý z nich je identifikován samotnou vrstvou i, ve které se nachází a pozicí v ní j.

$$c_i=\{n_{i,j}\} \tag{8}$$

Struktura sítě F popisuje pouze uzly, z nichž se síť skládá. Druhou částí definice topologie sítě musí být jednoznačné určení vazeb, kterými jsou vzájemně uzly propojeny mezi sebou.

$$L=\{w_{i,j\rightarrow k,l}\} \tag{9}$$

Vazba (propojení) w je určena vrstvou i a pozicí j, kde se nachází startovní uzel, a vrstvou k a pozicí l uzlu koncového.

Všimněme si, že použitá formalizace nepopisuje aktivní ani adaptivní dynamiku sítě. Specifikuje pouze architekturu sítě, tedy to, jak jsou spolu propojeny jednotlivé uzly. Aktivní a adaptivní dynamika se uvádí zvlášť.

3.2 Simulace

Modelování a simulace patří bezesporu k efektivním způsobům zcela běžně aplikovaných ve fázi návrhu či testování rozličných problémů. Při simulaci složitých funkčních bloků se často využívá hierarchického přístupu k modelování, který umožňuje na různých úrovních abstrakce zjišťovat nebo ověřovat jeho funkce a parametry. Na nejvyšší úrovni abstrakce modelujeme chování systému jako celku, na nižších úrovních pak jeho dílčí bloky a jejich vzájemné interakce.

Podobné pojetí, tedy samotné možnosti simulace neuronových sítí ve vizuálním simulátoru, může vycházet právě z těchto obecných principů. Modelovaným systémem zde myslíme neuronovou síť, dílčími funkčními bloky následně chápeme její uzly a vazební propojení. Žádoucí funkcionalitou simulace je eventualita nastavení úrovně abstrakce nad modelovaným systémem. Logicky se nabízí rozčlenění do těchto disjunktních úrovní: jeden iterační krok přes množinu vzorů sítě, změna parametrů sítě, změna stavu uzlu sítě a změna váhy vazebního propojení mezi uzly.

Důležité je, že při uvedeném přístupu řešení této funkcionality není rozdílu mezi simulací aktivní či adaptivní dynamiky sítě (obě způsobují změny stavů bloků na zvolené abstrakční úrovni) a nevyžadují tedy užití rozdílných simulačních mechanismů. Tento poznatek lze s výhodou praktikovat u samotného návrhu řešení, ale hlavně dovoluje za předpokladu aplikace vhodných programových technik dosáhnout vysoké obecnosti.

3.3 Vizualizace

Vizuální reprezentace sítě je nedílnou součástí vizuálního simulátoru. Neuronové sítě můžeme vyobrazovat mnoha způsoby. Některé z nich (například klasifikace uzlů do tříd, vizualizace založená na inhibičních vazbách) jsou charakteristické pouze pro specifickou skupinu neuronových sítí a nekorespondují proto s obecnou funkcionalitou simulátoru. Další restrikci ve výběru přináší slučitelnost vizualizace se simulací, neboli krokováním aktivní a adaptivní dynamiky sítě.

Nejlépe splňuje disjunkci uvedených požadavků vyobrazení fyzické architektury (topologie) sítě. Ze své podstaty dokáže graficky reprezentovat jakýkoliv druh sítě a vhodně ilustrovat všechny dynamiky sítě v korelaci se simulací. Možné problémy mohou nastat u zobrazení acyklických sítí, ale i ty lze řešit autonomně „inteligencí“ vizualizačního algoritmu.

Nicméně samotné znázornění architektury sítě pomocí grafických prvků nemusí být jediným řešením. Například topologii vícevrstvých sítí lze vhodně vyobrazit stromovou strukturou. Kladem stromové reprezentace je vnitřní logické rozdělení sítě do samosprávných celků (síť, shluky neuronů, neurony, kolekce vstupních a výstupních vazebních propojení atd.). Nesmíme opomenout ani zcela odlišnou formu, a to matematický popis sítě vycházející z představeného formalizmu.

Vizualizace není doménou jen samotné sítě, ale patří k ní vyobrazení množiny vzorů. Taktéž v tomto případě můžeme nalézt více možností, jak ji zobrazovat. Avšak zde je vazba na neuronovou síť vzdálená. Opěrným bodem ke zvolení toho či onoho způsobu bude naše chápání dat, tedy jejich informační charakter. Bohužel sémantický význam množiny vzorů si protiřečí s obecností simulátoru. Univerzální pohled na data (množinu vzorů) má genezi v jejich vektorovém pojetí. Ačkoli prezentovat množinu vzorů ve formě vstupních a výstupních vektorů nemusí být z hlediska názornosti ideální, jedná se o generalizující řešení.

3.4 Modulová rozšiřitelnost

Při deklarované obecnosti vizuálního simulátoru se přímo vybízí navrhnout jej tak, aby byl schopný pracovat s později definovanými neuronovými sítěmi. Takovou rozšiřitelnost lze postavit na modulové architektuře simulátoru. Jádro simulátoru je následně chápáno spíše jako patice pro zasouvání jednotlivých modulů vytvořených podle předem daných pravidel komunikačního rozhraní. Zároveň poskytuje globální nástroje pro práci se sítí (vizualizace, simulace, ukládání apod.).

4 Návrh řešení

Vizuální simulátor neuronových sítí je nutně aplikací řešící širokou paletu dílčích problémů. Obecná knihovna pro práci s neuronovými sítěmi, samotná simulace, komunikace s vizuálním simulátorem, vizualizace, ukládání a načítání dílčích částí (množiny vzorů, sítě, projekty), modulová rozšiřitelnost a uživatelské rozhraní jsou jen kapkami po dešti. Proto návrh řešení hraje klíčovou roli v celém projektu a vymezuje hranice jeho možné funkcionality.

Poznamenejme, že návrh řešení vychází z provedené analýzy. Kapitola se postupně zaměřuje na stěžejní, případně zajímavé prvky problematiky a představuje jejich konkrétní podobu.

4.1 Obecná koncepce

Projekt jako celek můžeme rozdělit do více samostatných částí. Jedná se o grafické rozhraní (vizualizátor), obecnou knihovnu pro práci s neuronovými sítěmi, jednotlivé moduly (neuronové sítě) a jádro. Blokové schéma aplikace ilustruje obrázek 2. Jádro je společnou komponentou vizualizéru i obecné knihovny. Na obecné knihovně jsou pak postaveny konkrétní typy neuronových sítí a spolu s dodaným obecným rozhraním se distribuují ve formě modulů vizuálnímu simulátoru.

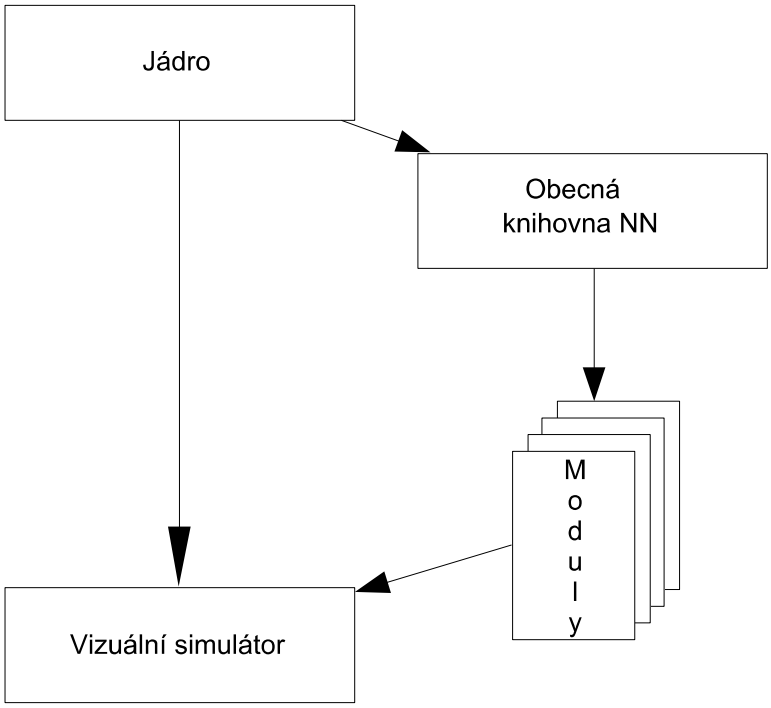
4.2 Jádro

Jádro plní účel bázového rozhraní jak obecné knihovny, tak vizuálního simulátoru. Rolí jádra je vytvořit společnou mezivrstvu, na níž komunikuje simulátor s knihovnou, a systémově řešit společné požadavky obou částí. Mezi ně patří například deklarace univerzálního rozhraní pro přístup k objektu, generování jedinečných identifikátorů objektů, schopnost transformace objektu do stromové struktury, jež je následně využita pro generování XML atd. Mimo to poskytuje vyšší úroveň abstrakce a zapouzdřuje implicitní jmenovanou funkcionalitu.

Obecný přístup k atributům objektu deklarovaný jádrem dává do vínku mocný nástroj dovolující práci s datovými složkami z vnějšku bez znalosti jeho vnitřní struktury. Na této bázi staví samotné rozhraní datové komunikace vizualizátoru s prvky neuronové sítě (modulu). Obecnost tohoto konceptu lze pojmout i tak, že ty části vizualizátoru, které jsou schopny prezentovat/měnit atributy prvků neuronové sítě prakticky neznají jejich skutečný charakter.

Na principu ne nepodobnému generickému rozhraní pro práci s atributy buduje i obecné rozhraní pro práci s potomky objektu. Možnost přístupu na jakýkoliv podobjekt právě

zpracovávaného objektu má stěžejní význam u transformace aktuálního dění do hierarchické struktury.



Obrázek 2: Obecná koncepce aplikace

Hierarchickou strukturu můžeme posléze uplatnit ve stromové grafické ilustraci sítě, použít jako mezistupeň při ukládání do XML či díky paměťovým nárokům uzlů stromu najde využití u tvorby historie. Samozřejmostí je též reverzní chování ve smyslu ustanovení objektu z hierarchické struktury.

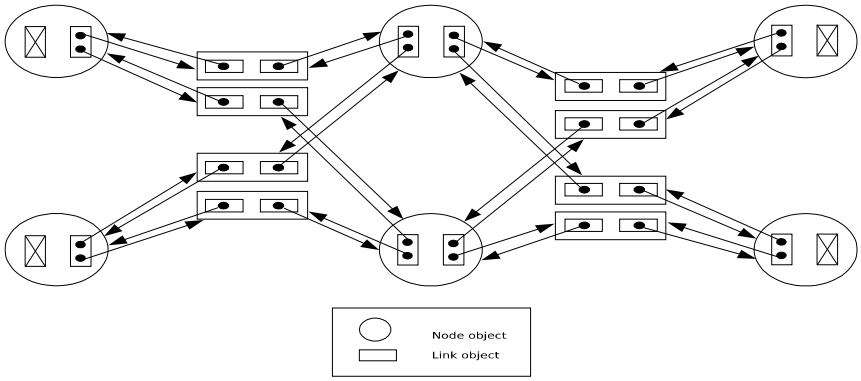
4.3 Návrh obecné knihovny

Rozhodující faktor kvality obecné knihovny neuronových sítí tkví ve výběru formalizmu popisujícího jejich architekturu. Koncepce obecné knihovny nesmí degradovat obecnost použitého formalizmu a tím uměle snižovat početnost množiny sítí jím popsatelných. Dále by měl návrh reflektovat rychlostní požadavky aktivní a adaptivní dynamiky neuronové sítě. Kromě poskytnutí prostředků pro manipulací s fyzickými strukturami neuronových sítí (uzly, vazební propojení) by knihovna nad nimi měla rovněž vytvářet logickou a funkční abstrakci (shluky, kolekce shluků, síť), neboli poskytovat abstraktní komponenty. Inspiraci lze nalézt v [5] a [7].

Poznamenejme, že abstraktními komponentami sítě jsou myšleny právě ty prvky, jež v korespondenci s biologickou neuronovou sítí nemají skutečnou reprezentaci. Mezi ně můžeme zařadit například samotnou síť (parametry, adaptační a aktivní dynamika), kolekci shluků, shluky uzlů atd.

4.3.1 Fyzická struktura sítě

Fyzická struktura sítě vychází z užitého formalizmu s přihlédnutím k rychlostním nárokům, stejně tak i z respektování vhodného objektového návrhu a snadné použitelnosti knihovny ze dvou bázových tříd: uzel (*BNode*) a vazba (*BLink*). Každý z uzlů má kolekci vstupních a výstupních vazeb, přes které je vzájemně propojen s ostatními uzly dle topologie sítě (viz obr. 3).



Obrázek 3: Fyzická struktura sítě

Komponenty *BNode* a *BLink* jsou pouze prvky schopné vytvořit fyzickou strukturu sítě a definovat základní rozhraní pro práci s nimi. Potřebná funkcionalita je posléze dodána na vyšší úrovni abstrakce podle jejich charakteru v rámci neuronové sítě.

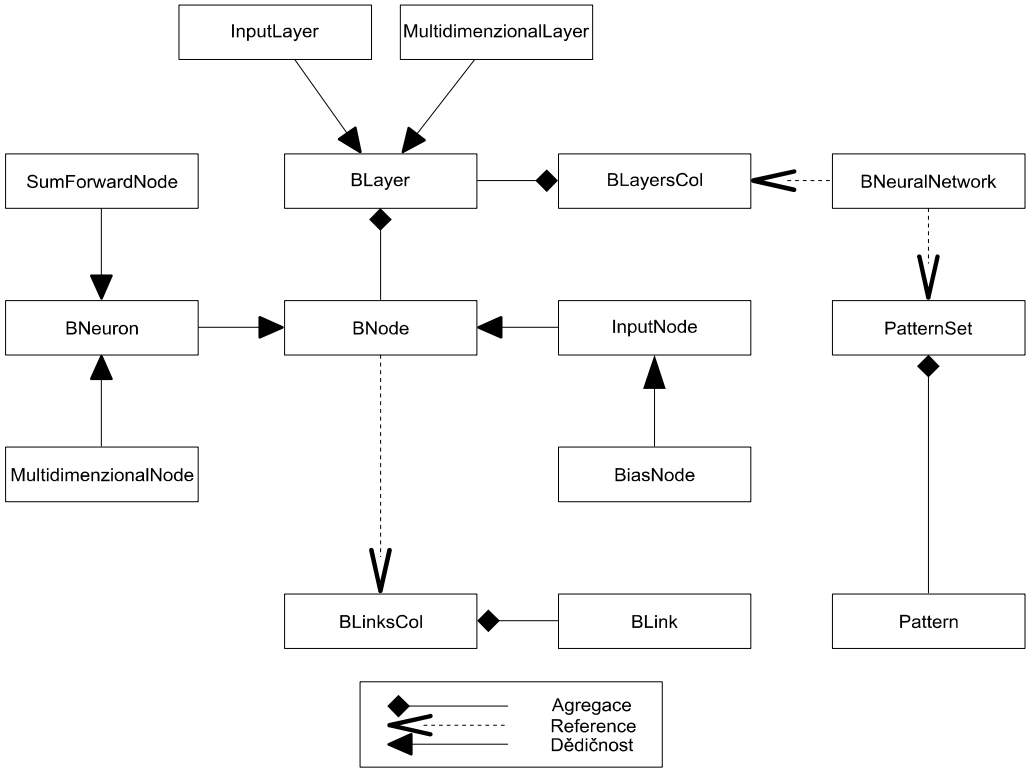
4.3.1.1 Bias neuronu

Bias má implicitně povahu atributu neuronu. Toto pojetí ovšem není nápadité pro popsání aktivní či adaptivní dynamiky sítě. Obecně je arbitrem obou procesů sama síť, která upravuje váhy uzlů dle vnitřních algoritmů, ne samotný uzel. V konečném důsledku by to znamenalo v dynamikách sítě dvojí přístup k uzlům, tedy komplikovalo zápis samotných algoritmů a způsobovalo další zbytečnou režii při běhu sítě.

Uvedené neduhy eliminuje vnímání biasu jako obyčejné vazby pramenící obecně z uměle vytvořeného neuronu s jednotkovým výstupem. Všimněme si, že tento koncept nakonec kopíruje uvedenou transformaci prahové hodnoty ve formálním modelu neuronu.

4.3.2 Koncept obecné knihovny

Koncept obecné knihovny musí podobně jako koncept fyzické struktury sítě vycházet z formalizmu pro popis sítí, ale navíc by měl být v konzistenci s logickým postavením jednotlivých prvků v síti.



Obrázek 4: Koncept obecné knihovny pro práci s neuronovými sítěmi

Obecná knihovna (viz obr. 4) pro práci s neuronovými sítěmi poskytuje tyto abstraktní komponenty jakožto generické třídy: *BNeuralNetwork*, *BLayersCol* a *BLayer*. *BNeuralNetwork* je bázovým rozhraním pro práci se sítí. Definuje organizační, adaptivní a aktivní dynamiku sítě. Třída má referenci na komponentu kolekci shluků (tříd *BLayer*) *BLayersCol* a referenci na komponentu množiny vzorů *PatternSet*. Vrstvy jsou rovněž kolekcemi již fyzických komponent *BNode*. *PatternSet* je stejně tak jako *BLayer* kolekcí, ovšem prvků *Pattern*. *Pattern* pro síť reprezentuje jeden vzor složený ze vstupního a výstupního vektoru.

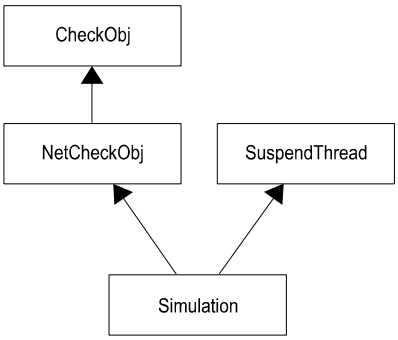
Mezi požadavky na obecnou knihovnu patří poskytnutí komfortního rozhraní pro definici nových typů sítí. A právě proto jsou implicitně součástí obecné knihovny dceřiné komponenty bázových tříd, které přináší vyšší abstrakt u specifického použití. Patří mezi ně *SumForwardNode*, *MultidimensionalNode, BInputNode, InputLayer* a *MultidimensionalLayer*.

SumForwardNode a *MultidimensionalNode* jsou potomky komponenty *BNeuron* (pracovní uzel), *InputNode* má za předka *BNode*. *SumForwardNode* je bázovou třídou pro uzly vícevrstvých neuronových sítí (backpropagation síť). *MultidimensionalNode* ve spojení s *MultidimensionaLayer*

naopak nabízí snadnou možnost, jak vytvořit neuronovou vrstvu libovolné dimenze. Skutečnost, že každá neuronová síť má vstupní vrstvu reflektuje dvojice tříd *InputNode* a *InputLayer* sloužící k tomuto účelu.

4.4 Simulace

Systémové řešení krokování aktivní a adaptivní dynamiky pramení z faktu, že obě způsobují změny stavů prvků v síti. Zároveň musí skutečná realizace odrážet požadavky na jednoduché použití. Jinak řečeno schopnost krokování se by se z pohledu uživatele obecné knihovny, který definuje novou síť, měla jevit jako implicitní.



Obrázek 5: Diagram dědičnosti simulační třídy

Obecně je mechanismus „krokování“ (na této úrovni se ještě o krokování nejedná) postaven na tzv. kontrolních bodech v zajímavých částech dynamik. Zpravidla jsou kontrolní body součástí metod nastavujících atributy zkoumaného prvku. Kontrolním bodem míníme ve skutečnosti statickou metodou třídy *CheckObj*, jež má jako parametr ukazatel na prvek, který kontrolní bod vyvolal, a ohodnocení úrovně kontrolního bodu. Komponenta *CheckObj* posléze provede dle povolenosti dané úrovně kontrolních bodů zavolání virtuální metody definující akci na kontrolní bod. Poznamenejme, že všechny kontrolní body jsou na podmíněný překlad, a proto lze danou síť snadno této funkcionality zbavit.

Ve vztahu s obecnou knihovnou je třída *NetCheckObj*. Její smysl spočívá v možnosti uplatnit implementované sítě i mimo vizuální simulátor při zachování systému kontrolních bodů. Příkladem může být vypisování atributů prvků sítě ve vlastní konzolové aplikaci.

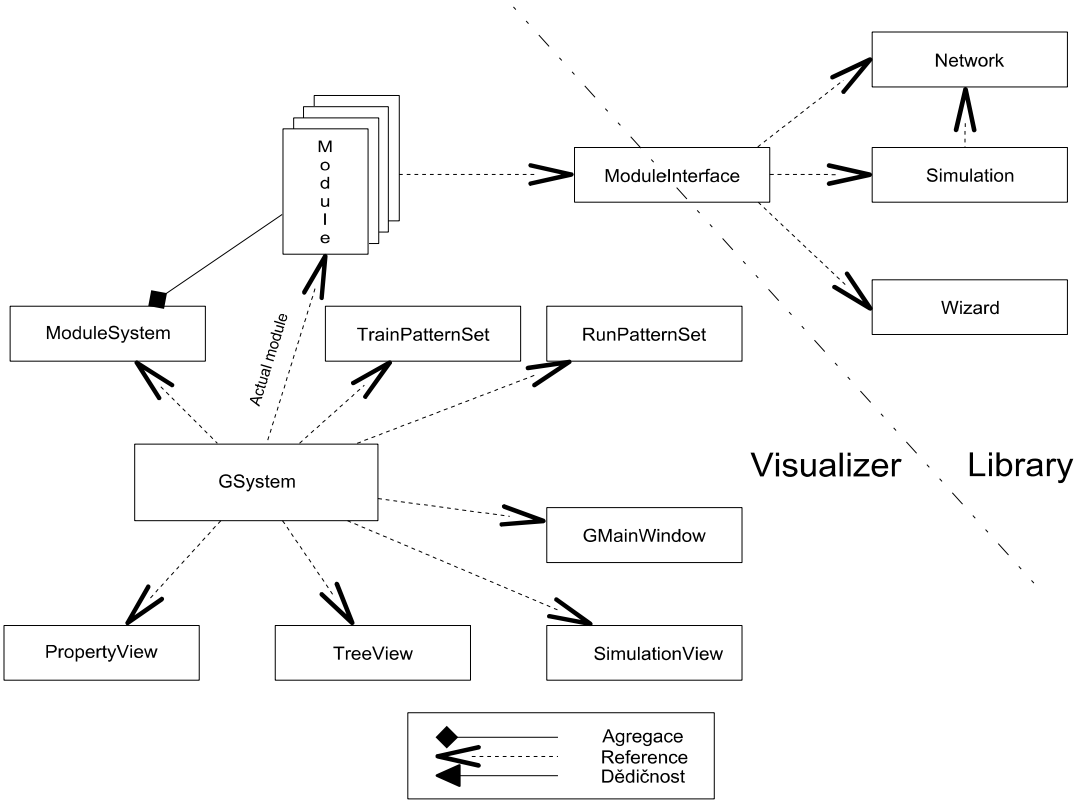
Naplnění významu slova krokování je učiněno až u komponenty *Simulation*, která dědí z třídy *NetCheckObj* a *SuspendThread* (viz obr. 5). *SuspendThread* jest třídou poskytující vlákno se schopností samouspání se. Funkcionalita krokování vybrané dynamiky sítě v konečném důsledku vypadá tak, že třída *Simulation* spustí vybranou akci v samostatném vlákně. Redefinicí virtuální metody určující akci na kontrolní bod bázové třídy *CheckObj* zajistíme uspání vlákna, a tedy

pozastavení aktuální činnosti. Další krok v simulaci znamená pouze probuzení uspaného vlákna. Vláknový přístup rovněž dovoluje aktuální činnost kdykoliv ukončit.

Výsledkem redefinice metody akce na kontrolní bod není samozřejmě pouze uspání vlákna. Mimo jiné dojde k nastavení příznaku pro překreslení oken, aby byla provedena vizuální reprezentace kroku atd. Možností akcí na kontrolní body najdeme celou řadu, kupříkladu snadné vytvoření historie, což jen podtrhuje transparentnost tohoto řešení.

4.5 Návrh vizuálního simulátoru

Jádro vizuálního simulátoru (viz obr. 6) tvoří třída *GSystem*. Jejím smyslem je být můstkem mezi modulovým rozhraním, respektive aktuálně používanou sítí, a okenní části aplikace. Poskytuje metody pro ukládání a načítaní jednotlivých prvků a zahrnuje v sobě reference na množiny vzorů pro síť.



Obrázek 6: Koncept vizuálního simulátoru

4.5.1 Okenní část vizualizátoru

Vizualizátor má koncepci SDI aplikace. Samotné okno představuje třída *GMainWindow*. *GTree* widget slouží pro stromovou reprezentaci sítě a je svázán přes události s třídou *GpropretyWidget*.

Widget *GPropertyWidget* zobrazuje atributy aktuálně vybraného prvku sítě. Obě tato okna textově vizualizují struktury neuronové sítě a jsou konfigurovatelná pomocí XML. O grafické vyobrazení vnitřní topologie sítě se stará třída *GVisualWidget*.

Množiny vzorů má aplikace dvě. Jedna se uplatní při adaptivní dynamice sítě a druhá při aktivní. Síti je předána reference na příslušnou množinu vzorů podle vybrané akce. Množiny vzorů jsou zobrazeny v tabulkové formě oknem *GPatternsWidget*, která dovoluje taktéž jejich přímou editaci.

4.5.2 Modulová architektura vizualizátoru

Vizuální simulátor lze připodobnit k patici, do které zasouváme jednotlivé moduly představující neuronové sítě, a tím rozšiřujeme jeho funkcionalitu. Jádro modulové architektury tvoří komponenta *ModuleSystem*, což je de facto kolekce načtených modulů (tříd *Module*) vizuálním simulátorem. Třída *Module* má referenci na třídu *ModuleInterface* neboli na obecné rozhraní pro práci s modulem sítě vyexportovaným dynamickou knihovnou.

Dynamická knihovna, respektive třída *ModuleInterface,* definuje přístup k síti (*Network*), simulační komponentě sítě (*Simulation*) a průvodci pro vytvoření sítě (*Wizard*). Simulační komponenta je nutně součástí dynamické knihovny kvůli řešení mechanismu krokování přes statické metody tzv. kontrolní body. Ty jsou totiž schopny v případě neexistence simulačního objektu tento objekt autonomně vytvořit a to právě v paměťovém prostoru knihovny.

5 Popis řešení

Podkapitola *Implementace* souhrnně popisuje, v čem byl celý projekt implementován. Druhá část *Ovládání vizualizátoru* má podobu stručné uživatelské příručky zaměřující se na základní práci se samotným vizuálním simulátorem. Ukázány jsou možnosti simulace, práce s trénovacími vzory a práce s neuronovými sítěmi.

5.1 Implementace

Celý projekt je implementován v jazyce C++ s využitím GUI toolkitu Qt. Tato kombinace byla zvolena z důvodu platformní nezávislosti a efektivnosti výsledného řešení. Implementace kopíruje nastíněný návrh řešení.

Qt poskytuje celou řadu tříd nemající grafický charakter a stává se tak pomalu platformně nezávislým rozhraním pro tvorbu aplikací. I vizuální simulátor těží z těchto výhod. Například vláknové řešení krokování, konkrétně třída *SuspendThread* dědí z třídy *QThread* a dodefinovává suspendovatelné vlákno. Podobně třída *QLibrary* uplatněná při načítání modulů sítí či třídy poskytnuté Qt knihovnou pro práci s XML formátem ulehčují splnění požadavku platformní nezávislosti.

Formát XML byl zvolen jako výchozí pro ukládání stavu sítě, množin vzorů, projektu a dílčích konfiguračních souborů. Důvodem je jeho průhlednost, samopopisnost a přirozená hierarchičnost. Přímá podpora XML ve všech částech je deklarována jádrem.

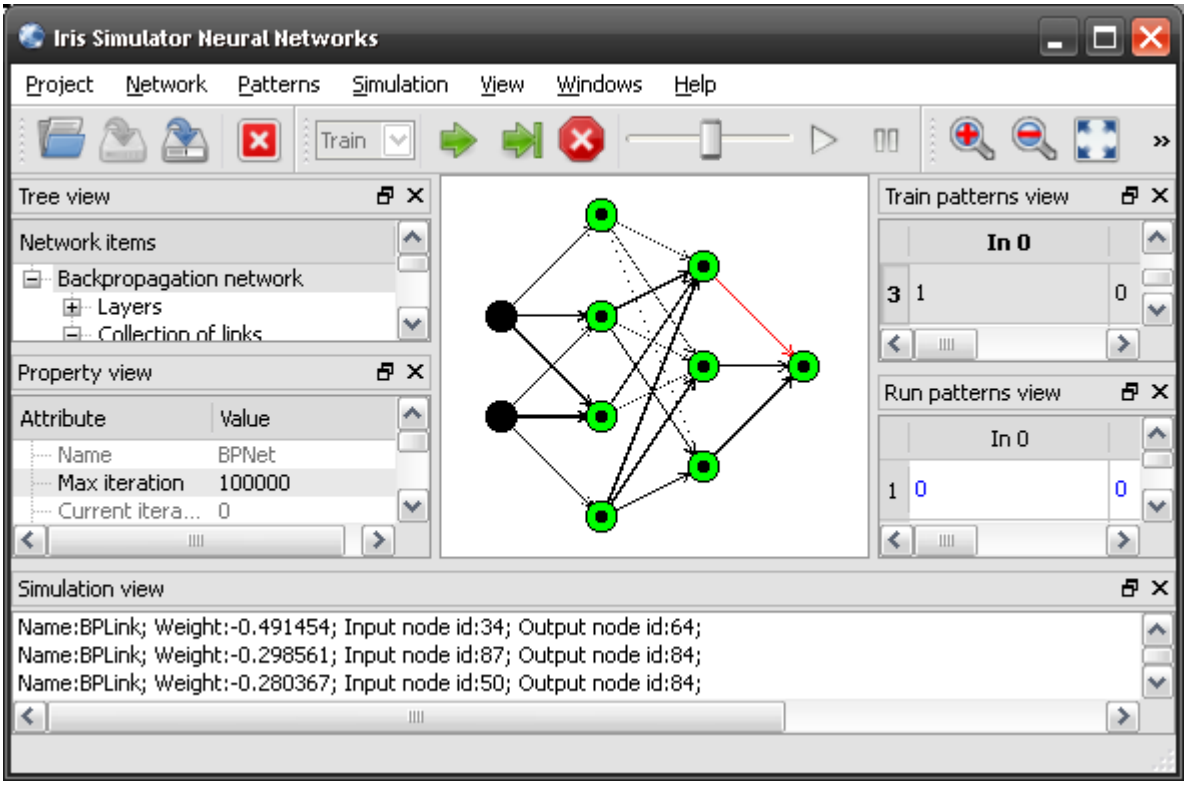
Za zmínku stojí i snadnost vytvoření dalších jazykových mutací vizuálního simulátoru. Toolkit Qt pamatuje na tento úkon aplikací Linguist, jež při dodržení jistých pravidel při programování ulehčuje jazykový překlad. Vizuální simulátor tyto zásady ctí.

5.2 Ovládání vizualizátoru

Filozofie uživatelského rozhraní vychází z pojetí sítě, množiny vzorů pro běh i pro trénování a projektu, jakožto čtyř samostatných celků. Proto každému z nich patří jedinečná nabídka akcí dovolujících jeho uložení, načtení, případně jiné specifické operace.

5.2.1 Práce s projektem

Projektem je kompozice neuronové sítě, množiny vzorů pro běh a množiny vzorů pro trénování. Do projektového souboru se uchovávají cesty k dílčím souborům, tj. neuronové sítě, trénovací množiny vzorů a množiny vzorů aktivní dynamiky. Nejsou-li dílčí části uloženy a byly užity, volají se postupně dialogová okna pro jejich uložení. V projektovém souboru jsou rovněž zaznamenány nastavení týkající se vizualizace a kontrolních bodů. Poznamenejme, že projektem mohou být také pouze množiny vzorů bez samotné neuronové sítě. Otevřením projektového souboru se načtou všechny jeho dílčí části, přičemž na neplatné cesty k souborům je uživatel upozorněn varovnou hláškou.



Obrázek 7: Uživatelské rozhraní aplikace

5.2.2 Práce se sítí

Akce proveditelné se samotnou sítí jsou zprostředkovány nabídkou „*Network*“. Síť můžeme načíst, uložit a uložit jako. Novou síť definujeme přes k ní dodaného průvodce vyplněním všech povinných údajů.

5.2.3 Práce s množinami vzorů

Množiny vzorů aplikace poskytuje dvě. Jedna je použita pro trénování sítě a vizualizuje ji okno „*Train patterns view*“. Druhá se aplikuje při aktivaci aktivní dynamiky a zobrazuje ji okno „*Run patterns view*“. Každou množinu vzorů lze jednotlivě uložit a načíst. Práce s oběma vzory je stejná.

K vytvoření chtěné množiny vzorů slouží průvodce. Průvodce si vyžádá velikost vstupních a výstupních vektorů trénovacích vzorů (množina vzorů je vždy homogenní) a jejich počet. Implicitně mají všechny složky vektorů hodnotu 0. Hodnoty se následně editují přímo v příslušné tabulkové vizualizaci.

Kliknutím na *Patterns->Use train as run* dojde ke kopii trénovací množiny vzorů do množiny vzorů pro aktivní dynamiku sítě.

5.2.4 Simulace

Simulační panel se skládá ze dvou částí oddělených čárovým separátorem. První z nich má řídící postavení. Položka v ovládacím prvku *combobox* definuje dynamiku, která bude u sítě při stisku aktivačního tlačítka zahájena, nebo která je právě aktuální. Výběr máme mezi adaptivní (*train*) a aktivní (*run*) dynamikou. Tento ovládací prvek zešedne a stane se tak nepřístupným, pakliže jedna z dynamik byla již aktivována. Tlačítko s názvem „*Run*“ spustí provádění vybrané dynamiky bez krokování. Sousedící ovládací prvek „*Next step*“ má stejnou funkcionalitu jako tlačítko „*Run*“ s tím rozdílem, že uvede v činnost krokovací mechanismy. Obě akce můžeme kombinovat za běhu, jsou vůči sobě transparentní. Kdykoliv lze dynamiku spuštěnou přes tlačítko „*Run*“ pozastavit a dále krokovat kliknutím na tlačítko „*Next step*“, obdobně i naopak. K úplnému přerušení simulace slouží tlačítko „*Terminate*“. Druhá část řídí automatické krokování. Posuvníkem se nastavuje frekvence kroků, tlačítkem „*Play*“ se auto-krokování aktivuje, tlačítkem „*Stop*“ provedeme jeho deaktivaci.

Přes nabídku *Simulation->Settings* se zpřístupní okno nastavení aktivity úrovní kontrolních bodů. V tomto okně můžeme povolit typy prvků čí změn sítě, přes které chceme krokování vést. Implicitně jsou uplatněny všechny úrovně (kontrolní body) sítě. Mezi samotnými úrovněmi není žádná hierarchická nebo logická závislost. Proto lze mít například aktivované pouze krokování přes samotné vazební propojení uzlů sítě.

Poznamenejme, že v korelaci s nastavitelnými parametry sítě a jejich správnou kombinací jednoduše dosáhneme jakékoliv konfigurace sítě, kterou chceme zkoumat. Příkladem nám budiž stav sítě po stém iteračním kroku.

6 Popis implementovaných sítí

Pro demonstraci funkčnosti a obecnosti řešení vizuálního simulátoru a knihovny je součástí projektu implementace tří sítí ve formě samostatných modulů. Jedná se o prostou jednovrstvou síť perceptron, typického představitele vícevrstvých sítí, síť backpropagation a jednoho zástupce z řad samoorganizujících struktur Kohonenovu síť.

V podkapitolách se nachází stručný výčet jejich charakteristických vlastností a popis možností nastavení parametrů sítí na konkrétních implementacích. Podrobnější popis dynamik sítí lze nalézt v [1], [2], [3] a [4].

6.1 Síť perceptron

Síť perceptron se řadí mezi nejjednodušší neuronové sítě a v publikacích s touto tématikou lze její postavení připodobnit k programu „*Hello world*“. Proto byl implementován modul poskytující tuto legendární síť.

6.1.1 Princip a struktura

Za typický perceptron je považována neuronová síť s jedním pracovním neuronem propojeným se všemi vstupy sítě. Signál přenášený vstupními uzly může být binární (tj. má hodnotu 0 nebo 1), nebo bipolární (tj. má hodnotu -1, 0 nebo 1). Výstup perceptronu určuje sigmoidní aktivační funkce upravená *ostrá nelinearita.*

Váhové hodnoty jsou adaptovány podle adaptačního pravidla perceptronu tak, aby diference mezi reálným a chtěným výstupem byla co nejmenší. Aktualizaci podléhají pouze ty váhové hodnoty, které neprodukují chtěné výstupy.

6.1.2 Popis implementace

Modul perceptronové sítě není omezující v počtu pracovních neuronů. Chování adaptační dynamiky si potom můžeme představit jako více sítí pracujících ve stejný okamžik nad jednou trénovací množinou vzorů. Síť má heterogenní charakter (každý neuron lze nastavit samostatně).

6.1.2.1 Parametry sítě

Jediným nastavitelným atributem sítě je maximální počet cyklů přes trénovaní množinu vzorů „*Max iteration*“, jenž může nabývat kladných celých hodnot. Po dosažení tohoto počtu iterací dojde k automatickému ukončení vykonávané činnosti.

6.1.2.2 Parametry neuronu

Důležitým editovatelným atributem pracovního neuronu je velikost jeho prahové hodnoty. Výsledná prahová hodnota se rovná součinu váhy k *bias* uzlu a výstupu *bias* uzlu (atributy „*Bias weight*“, „*Bias value*“). Dále lze nastavit učící konstantu „*Learning rate*“, aktuální výstup neuronu „*Output*“ a aktuální chybu neuronu vůči vstupnímu vektoru „*Error*“. Poznamenejme, že výstupem pracovního neuronu smí být pouze hodnota -1, nebo 1. Výchozí hodnota „*Learning rate*“ se generuje náhodně v intervalu <0.0, 1.0>. Váhy neuronu jsou inicializovány náhodnou hodnotou z intervalu <-1.0, 1.0>.

6.2 Síť backpropagation

K nejrozšířenějším a nejznámějším modelům neuronové sítě patří vícevrstvá neuronová síť používající algoritmus zpětného šíření chyby v adaptační dynamice. Model této sítě vychází ze zobecnění sítě perceptron.

6.2.1 Princip a struktura

Vícevrstvá síť je tvořena minimálně třemi vrstvami neuronů: vstupní, výstupní a nejméně jednou vnitřní (skrytou) vrstvou. Vždy mezi dvěma sousedícími vrstvami se nachází tzv. *úplné propojení neuronů*, což znamená, že každý neuron nižší vrstvy je spojen se všemi neurony vrstvy vyšší.

Samotný adaptační algoritmus se skládá ze tří etap: dopředné (*feedforward*) šíření vstupního signálu tréninkového vzoru, zpětné šíření chyby a aktualizace váhových hodnot spojení. V první etapě obdrží všechny neurony vstupní vrstvy vstupní signál a zprostředkují jej ke všem neuronům vnitřní vrstvy. Následně každý neuron vnitřní vrstvy vypočítá svou aktivaci a pošle tento signál všem neuronům ve výstupní vrstvě. Opět každý neuron výstupní vrstvy spočte svou aktivaci, která odpovídá skutečnému výstupu po předložení vstupního vzoru. Během adaptace neuronové sítě jsou porovnávány vypočtené aktivace s výstupními hodnotami. Na základě tohoto porovnání se spočte chyba neuronové sítě. Část této chyby se šíří zpětně z výchozího neuronu ke všem neuronům předcházející vrstvy, jež jsou s tímto neuronem spojeny. Úprava váhových hodnot na spojení mezi neurony pak přímo závisí na šířené chybě. Cílem adaptace je minimalizace chyby sítě ve váhovém prostoru.

Aktivační funkce neuronů musí splňovat následující vlastnosti: musí být spojité, diferencovatelné a monotónně neklesající. Mezi nejčastěji používané aktivační funkce patří proto *standardní sigmoida* a *hyperbolický tangens*.

6.2.2 Popis implementace

Implementace neuronové sítě se zpětným šířením chyby není omezena na počet skrytých vrstev, ani na počet neuronů v každé vrstvě. První vrstva je vždy chápana jako vrstva vstupní, poslední má význam vrstvy výstupní. Rovněž síť backpropagation má heterogenní charakter.

6.2.2.1 Parametry sítě

Parametry sítě backpropagation odpovídají parametrům u sítě perceptron. Novým editovatelným atributem je velikost dovolené chyby sítě „*Error allowed*“ (implicitně nastavena na hodnotu 0.005). Dovolená chyba sítě, podobně jako maximální počet iterací přes trénovací vzory, říká, kdy bude adaptační dynamika sítě ukončena.

6.2.2.2 Parametry neuronu

Parametry uzlu sítě backpropagation mají navíc ve srovnání s pracovním uzlem sítě perceptron atribut „*Activation function*“ a „*Delta*“. Atribut s názvem „*Activation function*“ určuje, jakou aktivační funkci bude uzel používat. Lze si vybrat mezi funkcí *standardní sigmoida* (defaultní) a *hyperbolický tangens*. Atribut „*Delta*“ je uplatněn u adaptační dynamiky při etapě zpětného šíření chyby a představuje tuto hodnotu vypočtenou pro daný uzel. Učící koeficient má uzel nastaven při inicializaci náhodně v rozmezí intervalu <0.1, 0.9>. Váhy neuronu mají výchozí hodnotu náhodně z otevřeného intervalu <-0.5, 0.5>.

6.3 Kohonenova síť

Samoorganizující neuronové sítě jsou využívány pro rozlišení, rozpoznávání a třídění neznámých číslicových signálů. Mezi hlavní reprezentanty těchto sítí patří Kohonenovy mapy. Velkou výhodou těchto sítí je schopnost nalézt a rozpoznat společné znaky nebo odlišnosti vstupních dat. Proto jsou Kohonenovy mapy přímo předurčeny k aplikacím jako zpracování řeči, úprava zvuku, přepis ručně psaného textu na tištěný, automatické třídění apod.

6.3.1 Princip a struktura

Základ tvoří uspořádaná struktura neuronů. Zpravidla má tato struktura podobu dvojrozměrné matice. Tvar uspořádání neuronů ovlivňuje volbu tzv. okolí neuronu, jež vymezuje jeho sousedy (blíže

postavené neurony). V maticovém uspořádání neuronů je velikost okolí rovna počtu řad neuronů od centrálního neuronu.

Adaptivní dynamika sítě staví na výběru jediného neuronu s nejmenší vzdáleností hodnot vah od hodnot vstupního vektoru. Způsoby výpočtu vzdálenosti mohou být různé. Nejčastěji se však uplatňuje Euklidova vzdálenost. Vítězný neuron posléze adaptuje své váhy s cílem se co nejvíce přiblížit vstupnímu vektoru. Dojde rovněž k adaptaci vah neuronů v jeho okolí s jistým útlumem. Výsledný stav adaptace vypadá tak, že ve struktuře neuronů vznikne několik významných shluků, mezi nimiž se výrazně liší hodnoty vah neuronů. Počet shluků by měl být shodný s počtem odlišných parametrů, které Kohonenova síť dokázala ve vstupní množině vzorů detekovat.

6.3.2 Popis implementace

Modul Kohonenovy sítě poskytuje širokou škálu rozšíření. V první řadě je Kohonenova síť implementována jako multidimenzionální. To znamená, že dovoluje vytvořit síť libovolného rozměru, ne jen běžné jednorozměrné či dvojrozměrné struktury. Stejně jako v předchozích případech se jedná o síť heterogenní, a tedy u každého neuronu máme možnost nastavit parametry zvlášť.

Při aktivní dynamice síť vrací jako výstup na daný vstupní vektor identifikační číslo vítězného neuronu. Tato hodnota je ve vizuální simulátoru přímo zobrazována, přesto lze díky ní snadno získat skutečnou hodnotu výstupního neuronu (standardně bývá nastavena na hodnotu 1 pro vítězný uzel a na hodnotu 0 pro všechny ostatní) .

6.3.2.1 Parametry sítě

Mezi nastavitelné parametry sítě patří režim učení, konkrétně atribut se jménem „*Train mode fast*“. Pokud má tento parametr hodnotu *false*, aplikuje se adaptace na všechny neurony v síti s utlumením. Velikost utlumení je přímo úměrná vzdálenosti adaptovaného neuronu od vítěze neboli vzdálenější uzel mění své váhy s menší razancí. Pakliže je „*Train mode fast*“ nastaven na hodnotu *true*, nedochází k adaptaci všech uzlů sítě, ale pouze těch, jež jsou v momentálním sousedství vítězného neuronu. Tlumení závislé na vzdálenosti se tu projevuje rovněž, ale velikost okolí představuje hranici, za níž již k adaptaci vah uzlů nikdy nedojde.

Dalším parametrem sítě je velikost okolí „*Base radius*“. Počáteční hodnota odpovídá polovině rozměru největší dimenze. Velikost okolí neuronu se s počtem iterací v adaptivním režimu postupně snižuje. Stejně tak se s počtem iterací mění i učící koeficient neuronu „*Learning rate*“.

Koeficienty ovlivňující velikost změny okolí a změny učícího koeficientu ku počtu iterací jsou dva. První z nich nese název „*Damping rate*“. Tento koeficient je aplikován ve vzorci pro výpočet dočasné proměnné, jež se uplatní dále při kalkulaci nové učící konstanty, a pro výpočet nové velikosti sousedství uzlu.

Dočasná proměnná se počítá dle vztahu

$$hlpVar = \exp\left(dampingRate \,.\, currentIteration^2 / maxIteration^2\right) \quad . \tag{10}$$

Nová hodnota velikosti okolí uzlu je dána vztahem

$$R = R_0 \,.\, hlpVar \quad , \tag{11}$$

kde R_0 reprezentuje počáteční velikost okolí (atribut „*Initial radius*“).

Učící koeficient „*Learning rate*“ se počítá jako

$$learningRate = learningRateCoefficient_0 \,.\, hlpVar \quad , \tag{12}$$

kde atribut „*Learning rate coefficient*“ má implicitně po inicializaci sítě hodnotu 0.9.

6.3.2.2 Parametry neuronu

Kromě standardních atributů (identifikační číslo, výstup, učící konstanta atd.) mají neurony v Kohonenově síti koordináty určující jejich umístění, typ funkce pro výpočet vzdálenosti a samotnou hodnotu vypočtené vzdálenosti. Díky heterogennosti sítě můžeme každému neuronu v ní nastavit jiný typ funkce pro výpočet distance. Volit lze mezi Euklidovou vzdáleností (implicitní), sumou druhých mocnin rozdílu dílčích složek vektorů vzdálenosti a úhlem. Váhy neuronu mají výchozí hodnotu náhodně v intervalu <0.0, 1.0>.

7 Závěr

Cílem bakalářské práce bylo vytvořit vizuální simulátor spolu s obecnou knihovnou neuronových sítí. Vizuální simulátor měl vhodně graficky prezentovat adaptivní a aktivní dynamiky sítě. Rovněž bylo třeba dodržet při výběru implementačních technik jejich platformní nezávislost.

Obecnost knihovny pro práci s neuronovými sítěmi určuje výběr matematického formalizmu, jenž sítě popisuje. Obecná knihovna vychází ze zjednodušeného formalizmu pánů Rogerse a Satyadase, který přináší rozumný kompromis mezi složitostí a mohutnosti množiny sítí jim popsatelných.

Požadavek na simulaci neboli krokování dílčích dynamik v korelaci s obecností knihovny s sebou nutně nesl restrikci ve výběru vhodného mechanismu poskytujícího tuto funkcionalitu. Jako elegantní se ukázalo spojení vláknové koncepce s detekcí změn stavů prvků sítě. Mezi výhody patří univerzálnost a jednoduchost použití. Koncept je užitečný i mimo vizuální simulátor (ve vlastních aplikacích) a díky podmíněnému překladu tzv. kontrolních bodů máme možnost stále získat „čisté“ řešení.

Vizuální simulátor má modulovou architekturu. De facto je prvkem deklarujícím základní generické uživatelské prostředí pro práci se sítěmi. Nové neuronové sítě jsou distribuovány ve formě modulů, s nimiž umí vizuální simulátor komunikovat. Zdůrazněme, že náročnost vytvoření nového modulu neuronové sítě se limitně blíží ke složitosti překódování dynamik sítě do jazyka C++ (není nutno nikterak deformovat dynamiky sítě pro simulaci). Tento krok navíc výrazně usnadňují abstraktní komponenty obecné knihovny.

Genezí komunikace vizuálního simulátoru a neuronové sítě je obecné jádro, z něhož obě části těží. Nemálo vizuálních složek simulátoru přistupuje k prvkům sítě bez chápání jejich logického významu. Jádro rovněž implicitně deklaruje podporu XML, obecně hierarchické struktury snadno transformovatelné do kteréhokoliv formátu.

Pro demonstraci možností vizuálního simulátoru a obecné knihovny byly implementovány ve formě modulů tři typy neuronových sítí: perceptron síť, backpropagation síť a Kohonenova síť. Každá z nich má velké množství nastavitelných atributů ve snaze nabídnout co nejširší konfigurovatelnost a názornost jejich dynamik.

Vizuální simulátor ani obecná knihovna pro práci s neuronovými sítěmi svou celkovou koncepcí nedegradují množinu popsatelných sítí zvoleným formalizmem, tedy možnosti rozšíření o nové sítě jsou značné.

Konstruktivním a snadno realizovatelným doplňkem vizuálního simulátoru může být grafické znázornění časového průběhu jakéhokoliv atributu vybraného prvku sítě - historie. Spolu s historií se nabízí i implementování funkcionality dovolující zpětné krokování v dynamikách sítě. Modulová architektura dává naopak do vínku možnost distribuovat se sítí pro ni specifickou vizuální reprezentaci. Příkladem nám budiž Kohonenova struktura a vizuální vyobrazení rozdělení jejich uzlů do shluků (třídní klasifikace). Samozřejmě se nemusí jednat pouze o vizualizační prvky, ale také o různá rozšíření ulehčující tvorbu množin vzorů (normalizace vstupů, práce s obrázky apod.).

Rovněž zajímavé, nicméně už podstatně složitější doplnění by bylo spojení s evolučními technikami optimalizace parametrů neuronových sítí a zakomponování fuzzy logiky (viz [7], [8] a [9]).

Literatura

[1] Kuba, M., Neuronové sítě [online]. [cit.2008-05-10]
Dostupné na URL: <http://tf.czu.cz/~votruba/WAN/Neuronove%20site.pdf>.

[2] Šmída, J., Neruda, R., Teoretické otázky neuronových sítí. Matfyzpress, Praha, 1996.
ISBN 8085863189.

[3] Volná., E., Neuronové sítě 1. Skripta Ostravské univerzity, Ostrava, 1998.

[4] Zbořil, F., V., Soft Computing, přednášky na VUT FIT v Brně, [cit.2008-05-10]
Dostupné na URL: <http://www.fit.vutbr.cz/study/course-l.php?id=88>

[5] Rogers, J., Object-Oriented Neural Network in C++. Academic Press, New York, 1997.
ISBN 0125931158.

[6] Rogers., J., S., Satyadas., A., Applying a Neural Network Formalization. Department of Computer Science, The University of Alabama, Tuscaloosa, Alabama, 1994.

[7] Masters, T., Practical Neural Network Recipes in C++. Academic Press, New York, 1993.
ISBN 0124790402 .

[8] Volná., E., Neuronové sítě 2. Skripta Ostravské univerzity, Ostrava, 2002.

[9] Rao, V., C++ neural networks and fuzzy logic. MISPress, New York, 1995. ISBN 1558515526.

Seznam příloh

Příloha A. Ukázka tvorby nového modulu neuronové sítě.

Příloha B. CD obsahující elektronickou verzi technické zprávy, uživatelskou příručku, ukázky a zdrojové soubory programu.

Příloha A Ukázka tvorby nového modulu

Tato příloha na triviálním příkladu principiálně ukáže, jak můžeme vytvořit nový modul neuronové sítě připojitelný k vizuálnímu simulátoru. Neklade si za cíl být podrobným průvodcem, nicméně představené techniky jsou obecně praktikovatelné.

Tvorbu nového modulu dělíme na dvě etapy. První z nich se týká vytvoření samotné neuronové sítě, náplní druhé je tuto síť modulově zapouzdřit. Poznamenejme, že v první etapě se soustředíme jen na implementaci samotné sítě a není třeba řešit ostatní úkony. Laicky lze říci, že stačí pouze přepsat dynamiky sítě do programovacího jazyka C++ s využitím struktur poskytnutých obecnou knihovnou, jež tuto práci navíc výrazně ulehčují. Ve druhé fázi posléze vytvoříme modulové rozhraní, to znamená nakonfigurování uživatelského průvodce pro ustanovení této sítě a redefinici virtuálních metod ve třídě *BModuleInterface*. Pak modul přeložíme jako dynamickou knihovnu a připojíme jej k vizuálnímu simulátoru.

Všimněme si, že nové moduly lze tvořit s minimální znalostí vnitřního systému a věnovat se vskutku pouze neuronové síti samotné. Koncept, který byl za tímto účelem navržen, nabízí vysokou flexibilitu a nikterak nenutí programově přizpůsobit dynamiky sítě simulaci (krokování).

Příkladem nám budiž triviální úprava adaptivní dynamiky sítě typu perceptron. Nová síť pro jednoduchost bude odvozena již z hotového modulu pro práci s tímto typem sítě. Původní implementace adaptivní dynamiky prochází trénovací množinu vzorů stále v neměnném cyklickém pořadí a to i v případně špatné klasifikace výstupního neuronu. Naším cílem bude nepatrně odlišné chování. Budeme chtít, aby při špatné klasifikaci byl průchod trénovací množinou vzorů přerušen a začalo se v následném cyklu prvním trénovacím vzorem.

Implementace nové sítě

V první fázi vytvoříme novou třídu reprezentující naši síť, jež bude potomkem třídy *PerceptronNet*. Tuto třídu nazvěme například *MyPerceptronNet*. Díky tomu, že naše třída je dceřinou třídou hotové sítě, bude našim jediným úkolem redefinice virtuální metody *learn()*. Bylo by zbytečné popisovat zde celou adaptivní dynamiku sítě, proto pouze naznačme místo modifikace původního algoritmu. Požadovaného chování dosáhneme podmíněným voláním (v závislosti na chybné klasifikaci) metody *PatternSet::toStart()* v místě ukončení cyklu procházejícím všechny neurony výstupní vrstvy. Tato metoda provede restart indexu na aktuálně používaný vzor.

Druhá fáze implementace sítě znamená definovat obecné rozhraní pro přístup k ní. Obecné rozhraní je dáno virtuálními metodami třídy *BObject* a skládá se ze dvou částí. První z nich zajišťuje

přístup k dětem objektu (virtuální metody s prefixem *boChild* v názvu), druhá k samotným atributům prvku (virtuální metody s prefixem *boAttr* v názvu).

U našeho příkladu stačí redefinovat pouze přístup k atributům sítě, protože rodičovská třída *PerceptronNet* již metody pro práci s potomky definovány má. Mohlo by se zdát, že redefinice metod pro přístup k atributům sítě nemá důvodu, jelikož jsme žádné nové atributy nepřidali. To je pravdou. Nicméně k atributům sítě patří rovněž její systémový název a ten je nutné změnit, jinak by docházelo k duplicitě a v konečném důsledku k používání špatného konstruktoru (například při načítání z XML). Nekritická chyba by vytvořila síť jiné funkcionality, kritická by způsobila neustanovení sítě s vypsáním chybové hlášky. Kopií metody *boAttr()* z originálního vzoru a změnou atributu systémového názvu (atribut *sys_object_name)* na *MyPerceptroNet* máme novou síť kompletně implementovánu.

Vytvoření modulu

Zpřístupnění sítě jako modulu pro vizuální simulátor je především implementací uživatelského průvodce k ustanovení nové sítě. Avšak tvorba průvodců není cílem této kapitoly. Budeme předpokládat, že průvodce máme hotového a zaměříme se pouze na stěžejní části.

Modulové rozhraní popisuje třída *BModuleInterface*. Pakliže chceme síť distribuovat ve formě modulu, musíme vytvořit potomka této třídy a redefinovat virtuální metody *getNameButton()*, *createEmptyNet()* a *getWizard()*. První z jmenovaných metod vrací název tlačítka jako *std::string* zobrazeného ve vizuálním simulátoru (*return „MyPercpetronNet";*). Druhá slouží pro získání samotného ukazatele na nově vytvořenou síť (*return new MyPerceptronNet();*). Poslední vrací referenci na uživatelského průvodce k vytvoření sítě (*return new MyWizard();*).

O samotný export modulového rozhraní se stará funkce *getInterface()*. Návratová hodnota funkce představuje referenci na námi definované modulové rozhraní neuronové sítě (*return new MyModuleInterface();*).

Připojení modulu k simulátoru

Hotový modul přeložíme jako dynamickou knihovnu. Připojení této knihovny k vizuálnímu simulátoru provedeme přidáním cesty k ní do konfiguračního souboru aplikace *config.xml*. Součástí modulu je rovněž konfigurační soubor ve formátu XML říkající, jak se budou prvky sítě zobrazovat v textové podobě, určující chybové hlášky a definující hodnoty atributů, kterých mohou nabývat. Konfigurační soubor modulu není nezbytně nutný, ovšem jeho nedodáním se bude síť zobrazovat v „syrové" formě a nebude dovolena změna atributů prvku sítě.