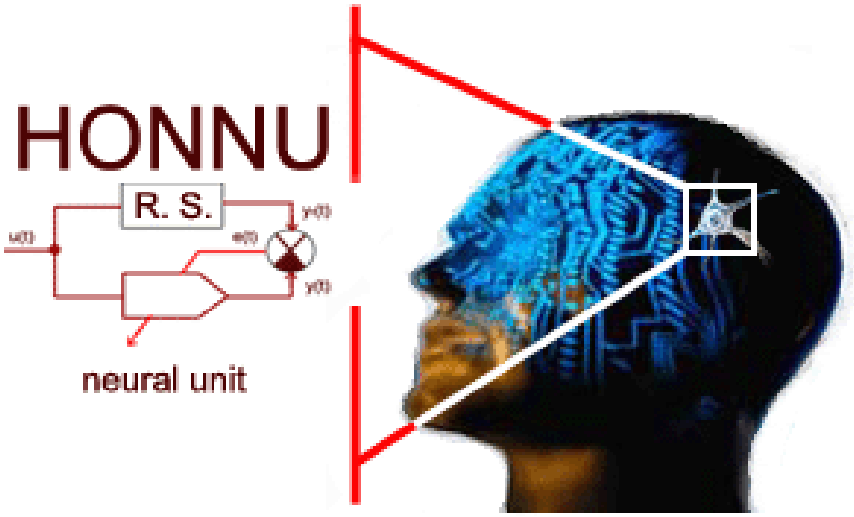
Simulační prostředí pro webové prohlížeče jednotky HONNU



Ing. Miroslav Kopecký

STČ 2007

Obsah

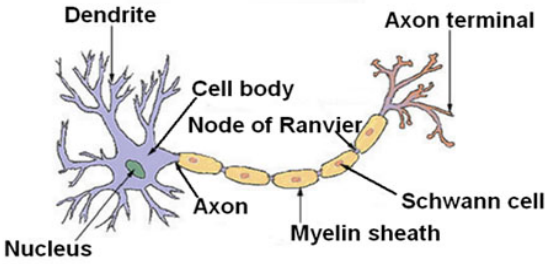
1 ABSTRAKT

Rozšiřitelné webové simulační prostředí zaměřené na neuronové jednotky HONNU. Data pro simulaci mohou být definována v prosředí samotném nebo načteny ze souboru. Zobrazení výsledků je realizováno jak grafickou, tak i textovou formou. Výsledné grafy je možno upravit podle indifiduálních požadavků a nasledně je uložit do grafickeho formatu PNG. Prostředí slouží k lepšímu pochopení problematiky HONNU.

2 ÚVOD

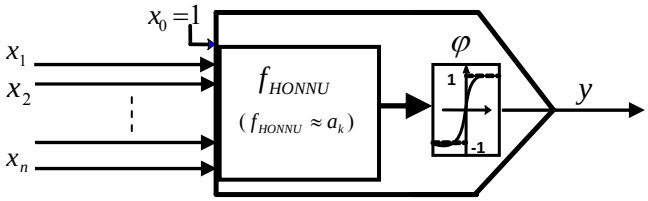
2.1 Jednotky HONNU

Biologické sytémy často slouží jako velká inspirace pro technické, založené na výpočetním aparátu vyznačujícím se velkou schopností učit se. Nekonvenční neuronové jednotky HONNU [1][2][3][4][5] se taktéž snaží lépe matematicky vystihnout a napodobit svůj přírodní protějšek biologický neuron (Obr 1) [12]. Z výpočetního hlediska je nervová soustava paralelní distribuovaný systém, který je schopen řídit složité procesy. Základním cílem vývoje ale stále zůstává bližší pochopení a porozumění procesům souvisejících s rozpoznáváním, učením a schopnosti, které poskytuje paměť.



Obr 1 – Struktura typického neuron

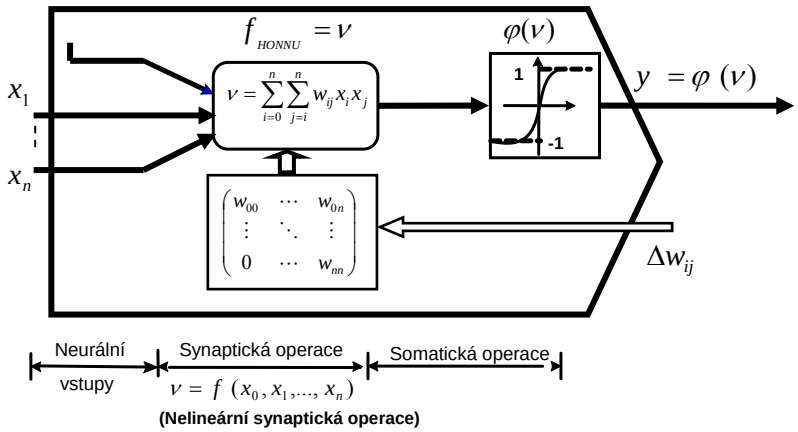
Snaha vytvořit jednoduchou a výpočetně výkonnější matematickou strukturu vedla ke vzniku neuronových jednotek, které bývají označovány jako HONNU (Higher Order Nonlinear Units [1] [5]) (Obr 2 a Obr 3). Tento směr modeluje snahu popsat celý systém co nejjednodušší neurální strukturou a představuje odlišný pohled na matematický zápis. Velkou výhodou nelineárních neuronových jednotek vyšších řádů je minimalizace a přehlednost neurálních parametrů, které se v neuronu typu HONNU vyskytují (váhy a struktura).



Obr 2 – Samostatná neuronová jednotka typu HONNU

Také se zachovává vysoká výpočetní kapacita jednotlivých neuronů. Výhoda je v získávaní jednoduchého matematického popisu systému, což je u konvenčních neuronových sítí znemožněno skladbou neuronové sítě z jednotlivých vrstev a neuronů s nelineární přenosovou (výstupní) funkcí, jejichž počet souvisí s nelinearitou řešeného problému.

HONNU tak díky své jednoduché struktuře poskytují relativně jednoduchý matematický popis celého systému. Předností se stává možnost do jednotek HONNU implementovat apriorní informaci o sledovaném systému, nejen jako počáteční nastavení, ale i jako vhodný typ nelinearity systému pokud je znám (například z odvozených rovnic). Do simulačního prostředí byl implementovány statické jednotky HONNU kvadratické (QNU) a kubické (CNU).



Obr 3 – Kvadratická neuronová jednotka (HONNU [2] [3] [4] [5]) s nelineární synaptickou operací (aktivační funkcí).

2.2 Synaptická operace

Synaptickou operaci (aktivační) je možno vyložit jako funkci, která představuje nelineární agregaci vstupů. Nepředpokládá se, že pouze sčítá jednotlivé vstupy násobené váhovými koeficienty w_{kj} prvního řádu. Rovnici pro synaptickou operaci (aktivační funkci) kvadratické neuronové jednotky můžeme vyjádřit předpisem (1).

$$f_{HONNU} = \nu = \sum_{i=0}^{n}\sum_{j=i}^{n} w_{ij} \cdot x_i \cdot x_j \in R^1 \tag{1}$$

Kde n je počet neurálních vstupů a předpis $x_a = [x_0, x_1, ..., x_n]^T$ představuje vektor, rozšířený o práh $x_0 = 1$. Maticový zápis sumace můžeme zapsat následujícím způsobem (2).

$$\nu = x_a \cdot W_a \cdot x_a^T \in R^1 \tag{2}$$

Kde W_a představuje rozšířenou váhovou matici o koeficient w_0, který představuje váhu prahu x_0. Výstup z neuronu, *y(k)* je vyjádřen nelineární funkcí somatického výstupu jako (3)

$$y_{(k)} = \varphi(\nu(k)) \in R^1 \tag{3}$$

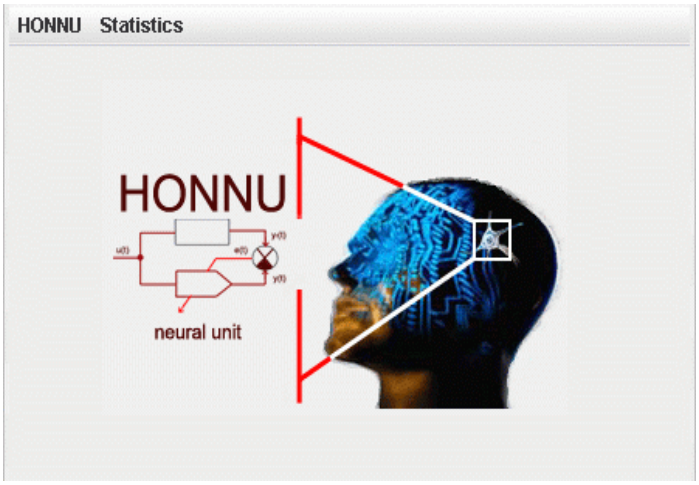
Kde φ představuje somatickou operaci [1] (přenosovou funkci). Pro Kubickou jednotku (CNU) je potup obdobný.

2.3 Algoritmus učení

Pro učení jednotek HONNU je použito gradientové učení metodou backpropagation [1] [5] . Pomocí tohoto algoritmu jsou zajištěny adaptabilní přírůstky vah v reálném čase.

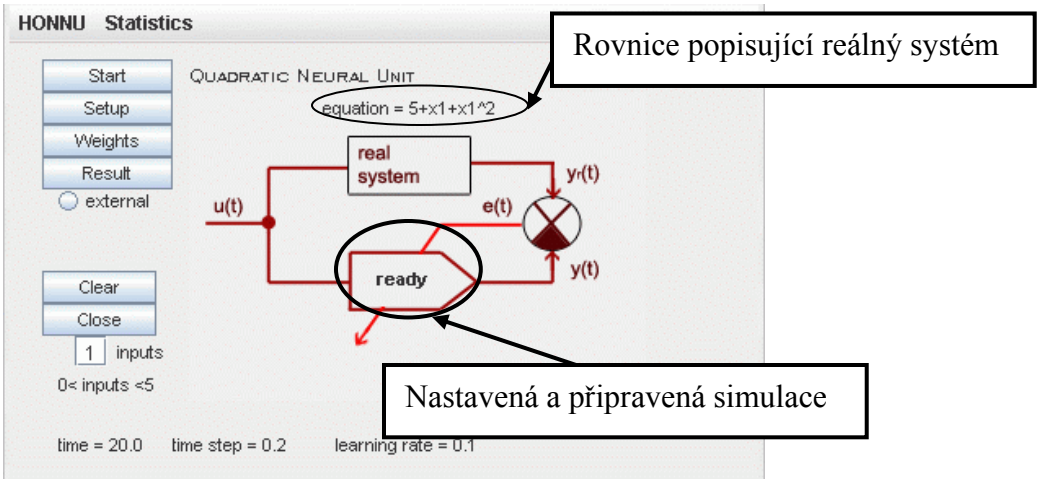
3 SIMULAČNÍ PROSTŘEDÍ

Simulační prostředí je vytvořeno pomocí programovacího jazyku JAVA [9][15] a je vloženo do webové stránky, zde je prezentováno *jAppletem*. Prvotní návrh byl vytvořen v jazyce UML [10] [18], takto získané diagramy následně doplňují projektovou dokumentaci. *jApplet* poskytuje interaktivní ovládání (viz. Obr 4). Jako zdrojový signál pro simulaci byla zvolena funkce sinus, která dosahovala při všech simulacích nejlepších výsledků díky periodické změně signálů.



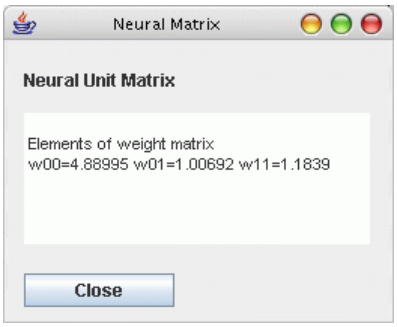
Obr 4 – Hlavní menu simulačního prostředí

Pomocí ovládacího panelu je možné aktivovat jednotlivé implementované moduly. Data mohou být zadávána v prostředí samotném nebo načtena pomocí souboru. Soubor musí obsahovat data v podporovaném formátu. Celkové ovládaní prostředí bylo snaha vytvořit intuitivně.

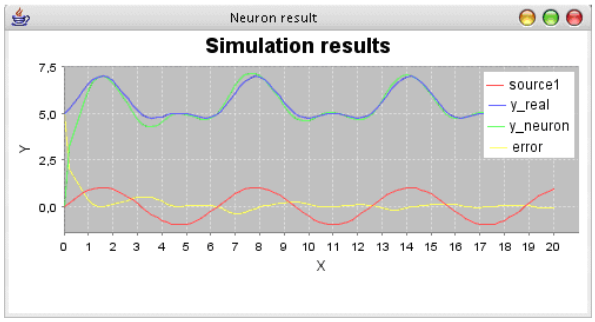


Obr 5 – Nastavená simulace

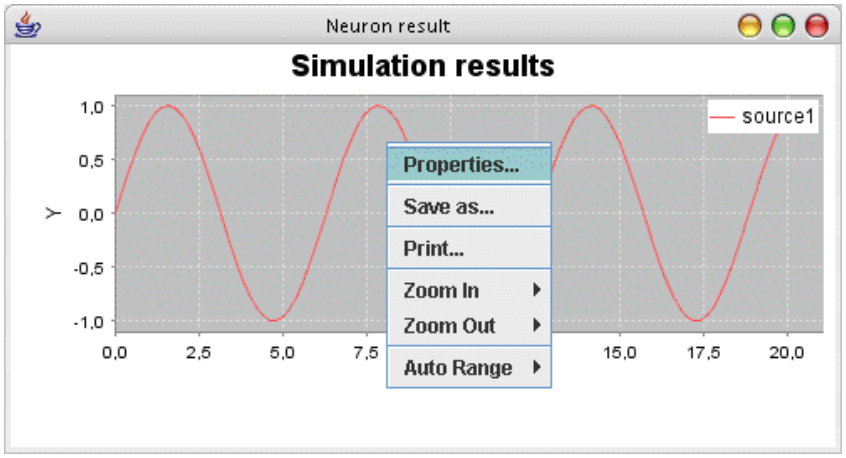
Dosažené výsledky identifikace je možno zobrazit v textové (Obr 6) i grafické podobě . Grafické výsledky (Obr 67) mohou být dále modifikovány (např. popisky nebo barvy viz. Obr 8) a posléze uloženy do formátu PNG.



Obr 6 – Zobrazení členů váhové matice



Obr 7 – Grafický výsledek simulace



Obr 8 – Menu pro práci s grafy

Matematické výpočty v rámci prostředí během celé simulace jsou řešeny s podporou knihovny JEP [6][7] (Java Math Expression Parser). Díky ní je možno procovat a vyhodnocovat rovnice v symbolickém zápisu.

Pro grafickou prezentaci výsledků je použito knihovny JFreeChart [16][17]. Knihovna pomáhá vytvořit prostředí, ve kterém je možno pracovat z grafy obdobným způsobem jako ve známém prostředí Matlab/Simulink.

Obě knihovny jsou dostupné zdarma a jejich vývoj probíhá pod záštitou veřejné licence.

4 ZÁVĚREČNÉ ZHODNOCENÍ

Použití neuronových sítí si získává v dnešní době stále větší oblibu i v oborech, kde jejich použití nebylo v počátku typické. Způsoby využití a implementace můžeme nalézt ve většině průmyslových odvětvích. Samotné téma představuje pro řadu lidí velkou neznámou. Důvodem může být složitá představa biologického protějšku. I přes to, že matematický popis problému není složitý jako přírodní realizace, stále přetrvává nedůvěra a nejednoznačnost. Také toto je jeden z důvodů, proč navržené simulační prostředí jako svůj první modul realizuje umělé neuronové jednotky HONNU a tak nabízí možnost jejich použití pomocí www prohlížečů.

Na počátku vzniku simulačního prostředí bylo potřeba řešit problémy, které se týkaly schopnosti řešit rovnice v symbolickém zápisu. Tyto rovnice měly obsahovat víc než jednu proměnnou, která se bude v průběhu simulací vyhodnocovat. Další neméně složitou otázku představovala reprezentace dosažených výsledků, které byly získány v průběhu simulace. Zobrazení mělo být uživatelsky pochopitelné a přitom poskytovat jednoznačný přehled o celé simulaci. Závěrečnou otázku představovala samotná realizace neuronových jednotek a navrch datových modelů, které se budou během celé simulace užívat, tak aby nebyla omezena dostupnost a přístupnost po síti internet.

Po překonání a úspěšném vyřešení těchto otázek vzniklo simulační prostředí, které nabízí uživateli možnost vyzkoušet si aproximaci systémů za použití implementovaných jednotek HONNU (QNU a CNU). Jednotky je možno modifikovat a nastavit tak, aby vyhovovaly potřebám. Uživatel je schopen si dosažené výsledky uložit v grafické podobě, kde si může modifikovat vzhledy jednotlivých grafů. Prostředí mu také poskytuje možnost nastavení pomocí dat, která má uložena na svém lokální datové jednotce.

Předností vytvořeného simulačního prostředí je jednoduchost, nastavení a spuštění požadované simulace. Výsledky své práce si může uživatel pro další použití uložit. Také není omezen místem, kde je možno prostředí spustit. To je navrženo tak, aby bylo spustitelné na velké většině konvenčních výpočetních systémů.

Při návrhu prostředí byl brán ohled na možnost dalšího rozšíření. Z toto důvodu byl komentován rozsáhlý zdrojový kód, aby orientace v něm nezpůsobovala programátorovi značné problémy. Dalším přínosem pro vývoj se stávají UML diagramy, které poskytují pohled o vzájemné provázanosti jednotlivých částí. Samotné začlení dalších knihoven nebo funkcí pak je jednodušší a přehlednější. Díky použitým nástrojům je prostředí otevřeno novým systémům a prostředkům, které se mohou v následujícím období objevit. Také je možno používat jednotlivé knihovny nebo již implementované řešení napříč celým projektem. Tento fakt byl zohledněn již při samotném jejich návrhu a výsledkem se staly třídy, které nejsou do jisté míry závislé na místě použití. U všech tříd jsou značené vstupy a výstup a tento postoj je dobré zachovávat i nadále při dalším rozšiřování celého prostředí.

5 LITERATURA

[1] M.M. Gupta, J. Liang, and N. Homma, : *Static and Dynamic Neural Networks: From Fundamentals to Advanced Theory*, IEEE Press and Wiley-Interscience, published by John Wiley & Sons, Inc., 2003

[2] S. Redlapalli, K. –Y. Song and M. M. Gupta, *: Development Of Quadratic Neural Unit with Applications To Pattern Classification,* Fourth International Symposium on Uncertainty Modeling and Analysis ISUMA 2003, IEEE Computer Society, 2003, Maryland USA, ISBN 0-7695-1997-0, p.p. 141-146.

[3] K. -Y. Song, S. Redlapalli and M. M. Gupta, : Cubic Neural Units for Control Applications, Fourth International Symposium on Uncertainty Modeling and Analysis ISUMA 2003, IEEE Computer Society, 2003, Maryland USA, ISBN 0-7695-1997-0, p.p.324-329.

[4] Bukovsky I., S. Redlapalli and M. M. Gupta : Quadratic and Cubic Neural Units for Identification and Fast State Feedback Control of Unknown Non-Linear Dynamic Systems, Fourth International Symposium on Uncertainty Modeling and Analysis ISUMA 2003, IEEE Computer Society, 2003, Maryland USA, ISBN 0-7695-1997-0, p.p.330-334

[5] Bukovsky, I. : Development of Higher-Order Nonlinear Neural Units as a Tool for Approximation, Identification and Control of Complex Nonlinear Dynamic Systems and Study of Their Application Prospects for Nonlinear Dynamics of Cardiovascular System, Final report from scientific research under NATO Science Fellowships at the INTELLIGENT SYSTEM RESEARCH LABORATORY at the University of Saskatchewan in Canada from April to October 2003 partially supported by Internal Grant of Czech Technical University (IGS #CTU0304112)

[6] JEP – Java Math Expression Parser: http://www.singularsys.com/jep/, 2006

[7] DJep symbolic differentation and matricies for the Java Equation Parse (JEP): http://www.singsurf.org/djep/, 2006

[8] JMatLink Connect MATLAB and JAVA: http://jmatlink.sourceforge.net, 2006

[9] NetBeans: http://www.netbeans.org, 2006

[10] VisualParadigm for UML 5.2: http://www.visual-paradigm.com/product/vpuml/, 2006

[11] Bíla Jiří : Umělá inteligence a neuronové sítě v aplikacích, CVUT, 1998

[12] Wikipedia: http://en.wikipedia.org/wiki/Artificial_neuron, 11/2006

[13] McCulloch, W. and Pitts, W. (1943). *A logical calculus of the ideas immanent in nervous activity.* Bulletin of Mathematical Biophysics, 7:115 - 133.

[14] Brian Cole , Robert Eckstein, James Elliott, Marc Loy, Dave Wood: Java Swing, 2nd Edition, O'Reilly, 2002

[15] Rogers Cadenhead, Laura Lemay: Sams Teach Yourself Java™ 2 in 21 Days, Fourth Edition, Sams, 2004

[16] David Gilbert: The JFreeChart Class Library – Developer Guide, version 0.9.21, 2004

[17] JFree: http://www.jfree.org, 2006

[18] Kim Hamilton, Russell Miles: Learning UML 2.0, O'Reilly, 2006