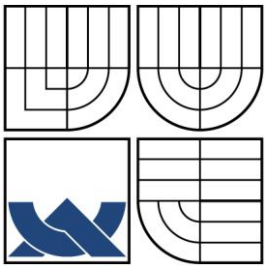
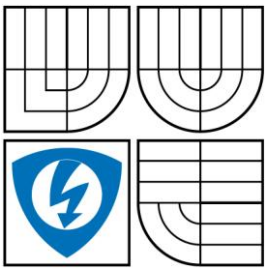


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

VYUŽITÍ NEURONOVÝCH SÍTÍ PRO PREDIKACI SÍŤOVÉHO PROVOZU

NEURAL NETWORKS UTILIZATION FOR NETWORK TRAFFIC PREDICTION

DIPLOMOVÁ PRÁCE
MASTER´S THESIS

AUTOR PRÁCE — BC. RADEK PAVELA
AUTHOR

VEDOUCÍ PRÁCE — ING. JAN KACÁLEK
SUPERVISOR

BRNO 2009

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Radek Pavela	***ID:***	89277
Ročník:	2	***Akademický rok:***	2008/2009

NÁZEV TÉMATU:

Využití neuronových sítí pro predikaci síťového provozu

POKYNY PRO VYPRACOVÁNÍ:

Popište statistické vlastnosti síťového provozu a možnost jeho predikace. Uveďte možné způsoby předpovídání intenzity síťového provozu a změřte se především na možnost predikace pomocí umělých neuronových sítí. Otestujte schopnosti rekuretních neuronových sítí predikovat hodnoty nelineárních časových řad.

DOPORUČENÁ LITERATURA:

[1] BESTAVROS, A., CROVELLA, M. E. Self-Similarity in World Wide Web Traffic. IEEE/ACM Transactions on Networking. 1997, Vol. 5, no. 6, p. 835-846.
[2] HALL, J. MARS, P. The Limitations of Artificial Neural Networks for Traffic Prediction. In Proceedings. Third IEEE Symposium, 1998, p. 8- 12
[3] ANNUNZIATO, M. et al. Evolutionary Feed-forward Neural Networks For Traffic Prediction, In International Congress on Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems, 2003

Termín zadání:	9.2.2009	***Termín odevzdání:***	26.5.2009

Vedoucí práce: Ing. Jan Kacálek

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

V této práci jsou prodiskutovány statické vlastnosti síťového provozu. Dále jsou rozebrány možnosti jeho predikce, se zaměřením na neuronové sítě. Konkrétně tedy hlavně rekurentní neuronové sítě. Trénovací data byla stažena z volně přístupného odkazu. Jde o zachycené pakety provozu LAN sítě z roku 2001. Nejsou nejaktuálnější, ale lze pomocí nich dosáhnout cílů práce. Vstupní data bylo třeba zpracovat do přijatelné podoby. Ve vývojovém prostředí Visual studio 2005 byl vytvořen program na agregaci intenzit těchto dat. Sloučení se jevilo nejvhodnější po intervalech 100 ms. Tím bylo dosaženo vstupního vektoru, který byl rozdělen podle potřeby sítí na trénovací část a testovací část. Jednotlivé typy sítí pracovaly se stejnými vstupními daty, čímž se dosahovalo objektivnějších výsledků.
Z praktického hlediska bylo třeba ověření dvou principů. Principu trénování a principu generalizace. První ze jmenovaných vyžadoval přikládání trénovacích vzorů a ověřování trénování pomocí gradientu a střední chyby. Druhý představoval přiložení neznámých vzorů na neuronovou síť. Sledována byla reakce sítě na tato data. Lze říci, že nejlepším modelem se jevila obecná neuronová síť (LRN). Proto bylo řešení rozvíjeno v tomto směru, kdy následovalo hledání vhodné varianty této rekurentní sítě a její otpimální konfigurace.
Nalezenou variantou je topologie 10-10-1.

Bylo využíváno programu Matlab 7.6, s nástavbou Neural network toolbox 6. Výsledky jsou zpracovány formou grafů a závěrečným zhodnocením. Všechny úspěšné modely a topologie sítí jsou na přiloženém CD. Avšak Neural network toolbox vykazuje určité problémy při jejich importu. Při tvoření této práce nebylo funkce importu sítě prakticky využíváno. Síť lze importovat, ale většinou se jeví jako nenatrénovaná. Neúspěšné modely sítí nejsou v práci prezentovány, neboť by došlo ke zhoršení přehlednosti a orientace.

Klíčová slova: rekurentní neuronová síť, predikce síťového provozu, agregace intenzit provozu, statické vlastnosti, trénovací data, Neural network toolbox,

ABSTRACT

In this master's thesis are discussed static properties of network traffic trace. There are also addressed the possibility of a predication with a focus on neural networks. Specifically, therefore recurrent neural networks. Training data were downloaded from freely accessible on the internet link. This is the captured packej of traffic of LAN network in 2001. They are not the most actual, but it is possible to use them to achieve the objective results of the work. Input data needed to be processed into acceptable form. In the Visual Studio 2005 was created program to aggregate the intensities of these data. The best combining appeared after 100 ms. This was achieved by the input vector, which was divided according to the needs of network training and testing part. The various types of networks operate with the same input data, thereby to make more objective results. In practical terms, it was necessary to verify the two principles. Principle of training and the principle of generalization. The first of the nominated designs require stoking training and verification training by using gradient and mean square error. The second one represents unknown designs application on neural network. It was monitored the response of network to these input data. It can be said that the best model seemed the Layer recurrent neural network (LRN). So, it was a solution developed in this direction, followed by searching the appropriate option of recurrent network and optimal configuration. Found a variant of topology is 10-10-1.

It was used the Matlab 7.6, with an extension of Neural Network toolbox 6. The results are processed in the form of graphs and the final appreciation. All successful models and network topologies are on the enclosed CD. However, Neural Network toolbox reported some problems when importing networks. In creating this work wasn't import of network functions practically used. The network can be imported, but the majority appear to be non-trannin. Unsuccessful models of networks are not presented in this master's thesis, because it would be make a deterioration of clarity and orientation.

Keywords: recurrent neural network, prediction of tradic trace, aggregation of intensit of traffic trace, static properties, training data, Neural Network toolbox,

PAVELA, R. *Využití neuronových sítí pro predikaci síťového provozu.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 83 s. Vedoucí diplomové práce Ing. Jan Kacálek.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Využití neuronových sítí pro predikaci síťového provozu“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne …….. …….....

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu diplomové práce Ing. Janu Kacálkovi za velmi užitečnou metodickou pomoc a cenné rady při zpracování diplomové práce.

V Brně dne …….. ……...……....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

1 Úvod

Lidská společnost je obklopena nejrůznějšími technickými zařízeními a systémy. Už odedávna však existuje také snaha je zdokonalovat a vylepšovat. Avšak v technické praxi se vyskytuje mnoho systémů, které jednoduše vylepšit nelze. Buď z důvodu porušení některého ze základních pravidel jeho funkčnosti, nebo proto, že nelze přesně identifikovat problém. V moderní době je výpočetní technika značně rozšířena ve firmách i v domácnostech. Samozřejmě velká část těchto zařízení je připojena k internetu nebo vzájemně k sobě, čímž vytváří počítačovou síť. Pojmeme-li tuto síť z technického hlediska, jedná se vlastně také o systém. Má také své vlastnosti a určitá pravidla.

Nyní si můžeme opět položit prvotní otázku, jak tento systém vylepšit. Obecně lze použít dva standardní způsoby, na základě nichž provádíme zásahy do systému za účelem vylepšení vlastností. Buď budeme sledovat jeho chování (tzn. určité hodnoty, které jsme obdrželi z jeho výstupu) v minulosti a na základě něj se rozhodneme ke změně. Nebo, což je mnohem účinnější (ale ne vždy možné), se pokusíme predikovat vývoj výstupních hodnot do budoucnosti a tím získat zkušenost, kterak se bude vyvíjet výstup (produkt) za stávajících vstupních podmínek. V našem konkrétním případě počítačové sítě se zabýváme jejím provozem – tzv. traffic trace. Jednotlivá zachycená data nám udávají, kolik je v daný okamžik přeneseno informací. Hlavními aspekty práce je tedy popis statických vlastností síťového provozu, včetně možností a způsobů jeho predikce. Největší pozornost bude věnována právě neuronovým sítím, kde prověříme schopnost učit se (půjde o učení s učitelem) i princip generalizace (zevšeobecňování), tzn. schopnost sítě reagovat správně i na zatím nenaučené vstupy.

Následně bude snaha gradace řešení spočívat ve dvou fázích. Nejprve půjde o hledání nejvhodnějšího modelu sítě pro predikci a následně o hledání nejvhodnější varianty tohoto vybraného modelu.

2 Historický vývoj neuronových sítí

- **1943** – počátek prací zabývajících se neuronovými sítěmi. Vznik velmi **jednoduchého modelu neuronu** (W. McCulloch, W. Pitts)
- **1949** – vydána kniha ***The Organization of Behavior*** (D. Hebb). Otevřela obzory v oblasti aplikace **pravidla pro synapse** neuronů (interface). Šlo o navržení pravidla učení (Hebbovo pravidlo), nastavujícího váhu spoje mezi dvěma neurony podle velikosti jejich aktivit. Stala se základem pro další vědce, zabývající se neuronovými výpočty. Základní myšlenkou jsou podmíněné reflexy, pozorovatelné u živočichů.
- **1951** – Marvin Minsky podnítil vznik neuropočítače **Snark**, jehož hlavním úkolem bylo adaptovat váhy.
- **1957** – zobecnění modelu neuronu na tzv. **perceptron** a následné popsání učícího pravidla (F. Rosenblatt). Práce s reálnými čísly. Jedná se o jednovrstvou síť, obsahující „n“ vstupů a „m“ výstupů.
- **1959** – vytvoření neuronu **adaline** (adaptive linear element), včetně popisu pravidla učení (B. Widrow). Výstupy sítě jsou obecně reálné a jednotlivé prvky realizují lineární funkci.
- **1960** – založení první firmy na aplikaci neurovýpočtů (R. Barron a L. Gilstrap). Studie N. Nilssona ***Learning Machines.*** Podnětem bylo vytvoření společnosti **Memistor Corporation** zaobírající se stavbou neuropočítačů a jejich součástek (založena B. Widrowem v pol. 60. let)
- **1964** – S. Grossberg se stará o vydávání prací, v nichž je matematicky analyzována řada poznatků, týkajících se neuronových sítí (např. samostabilizace, samoorganizace a kompetiční učení). Následně navrhl neurony typu *instar* a *outstar.* Ve spolupráci s G. Carpentrovou se zabýval analýzou sítě ART.
- **1969** – nastává krize a úpadek zájmu o neuronové sítě. Finanční podpora projektu slábne. To vše má za následek vydání publikace ***Perceptrons*** (M. Minsky, S.Pepert), ve které jsou zpochybněny možnosti neuronových sítí. Konkrétně tedy závěry týkající se problému separovatelnosti logické funkce XOR, kterou podle autorů, nelze řešit pomocí perceptronu. To je sice možné pomocí vícevrstvé sítě, avšak v té době ještě nebyly známé postupy řešící adaptaci jednotlivých vah.
- **1969 – 1982** – období stagnace. V důsledku předcházející události se na dlouhých 13let zastavil výzkum neuronových sítí.

- **1982** – návrh tzv. **Kohonenové sítě** (T. Kohonen), rozvíjející myšlenku kompetičního učení. Dále vznik návrhu *učící vektorové kvantizace* (LVQ). V témže roce, díky poznatkům z oblasti magnetických materiálů a z oboru aplikace energetické funkce pro učení a vybavování vznikla tzv. **Hopfieldova síť**(J. Hopfield) fungující na principu autoasociativní paměti. Jde o pevnou strukturu o „n“ neuronech, zapojených cyklicky. Všechny prvky jsou zároveň výstupní.
- **1986** – algoritmus **Backpropagation** (PDP skupina - Parallel Distributed Processing Group, zástupci D. Rumelharten, G. Hintonem, R.Williamsen) – učící algoritmus zpětného šíření chyby. Umožňuje učit vícevrstvou perceptronovou síť a tím řešit zdánlivě neřešitelný problém funkce XOR. Ukázalo se, že závěry M. Minskeho jsou mylné a problém lze řešit (byť jiným způsobem). Jde o nejužívanější algoritmus, který je součástí 80% všech neurosystémů.
- **1987** – První velká konference se zaměřením na neuronové sítě v San Diegu. Přivítala na 1700 účastníků a gradovala založením mezinárodní společnosti pro výzkum ***INNS (International Neural Network Society).*** Následkem bylo masové rozšíření zájmu o obor. Začala se vydávat řada prestižních časopisů a univerzity na celém světě přispívaly ke zkvalitnění vývoje.

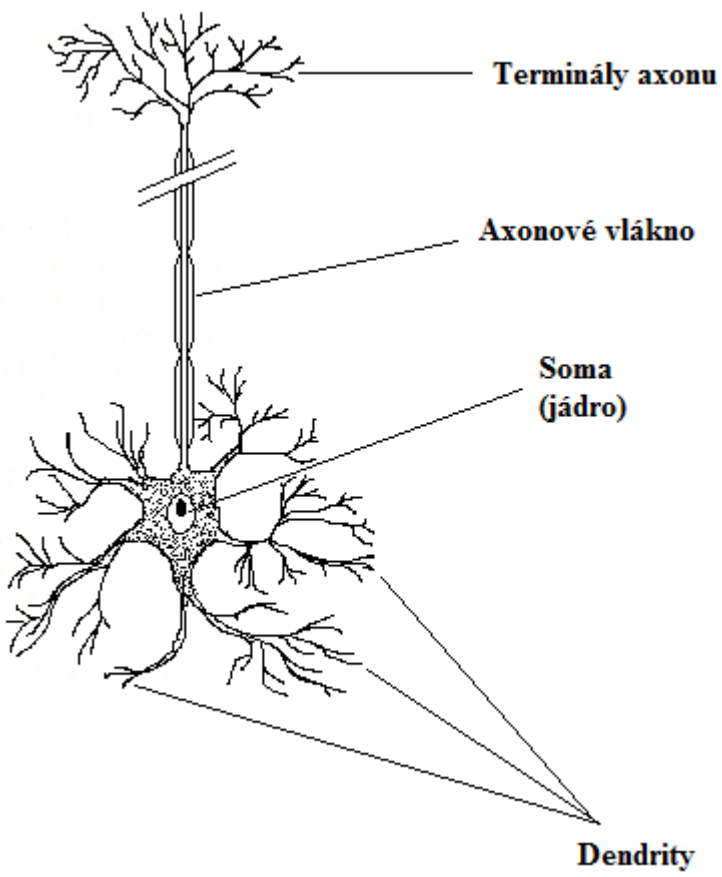
2.1 Biologický neuron

Tou prvotní myšlenkou, která přivedla vědce k tématu zkoumání neuronů a neuronové problematiky obecně, byla touha pochopit činnost mozku. Analyzovat procesy a aktivity v něm obsažené a pokud možno, co nejvěrněji je modelovat. Záhy se objevily první výsledky v podobě zjednodušených matematických modelů, jenž našly své uplatnění při sestavování praktických úkolů z oblasti umělé inteligence. Obor neurofyziologie se tak stal předlohou pro koncipování neuronových sítí.

Lidský mozek má přibližně 100 miliard neuronů (10^{11}). Při popisu biologického neuronu, coby základního stavebního kamene nervové soustavy, si můžeme všimnout jeho stavby, uzpůsobené pro přenos signálů. Obsahuje totiž vstupní a výstupní kanály (Dendrity a Terminály axonu – resp. axony obecně), které v podstatě tvoří v přenosové rozhraní. Vlastním tělem neuronu je pak Soma. Ke komunikaci s dalšími neurony dochází pomocí terminálů, zakončených blánou, jenže se dotýká terminálů jiných neuronů. Tomuto rozhraní se říká Synapse. Rozlišujeme dva druhy Synapse:

- Excitační – stará se o rozvod vzruchu nervovou soustavou
- Inhibiční – zajišťují naopak útlum vzruchu

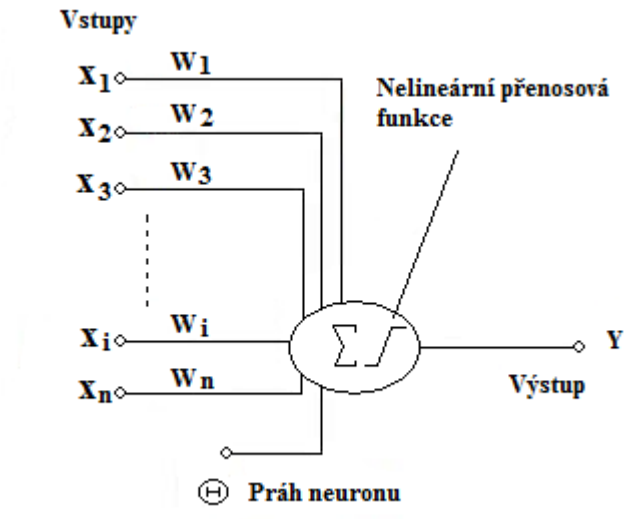
Axon i Soma mají na svém povrchu membránu, která generuje elektrické impulzy, které se přes synaptické brány přenášení z Axonu na Dendrity ostatních neuronů. Odpověď na otázku, kam až informace poputuje, určuje propustnost synaptických bran. Podle ní roste nebo klesá intenzita podráždění Dendrit dalších neuronů. Jde o řetězovou reakci s regulací míry šíření. Činnost neuronů je tedy zaměřena především na sběr, uchování, zpracování a následný přenos informace.



Obr. 2.1: Biologický neuron

2.2 Formální neuron

Pokud vyjádříme biologický neuron matematickým modelem, dostáváme formální neuron. Chování a vlastnosti skutečného prvku lze simulovat funkcemi. Jak můžeme vidět na modelu (formálního) neuronu, je zde n vstupů (X_1 až X_n), které představují dendrity. Ty jsou následně ohodnoceny synaptickými vahami (w_1 až w_n). Může nastat případ, kdy konkrétní hodnota váhy určitého vstupu nabude záporné hodnoty. Tím je simulována Inhibiční Synapse (útlum informace).



Obr. 2.2: Formální neuron

Perceptron pak transformuje vstupy (X_1 až X_n) na výstup Y dle následujícího vzorce:

$$Y = f\left(\sum_{i=1}^{n} Wi\ Xi + \theta\right) = f\left(\sum_{i=1}^{n} Wi\ Xi\right) = f(x) \qquad (1)$$

přičemž θ představuje váhu W_0 na vstupu $X_0 = 1$. Vzorec (1) je lze též nazývat výpočtem vnitřního potenciálu neuronu, který získáme sestavením vážené sumy vstupních hodnot.
Po dosažení prahové hodnoty vnitřního potenciálu (Θ) je indukován výstup neuronu (analogie elektrického impulsu u biologického neuronu). Funkce f je aktivační (přenosová) funkce, která má za následek nelineární nárůst výstupních hodnot při překročení prahového bodu Θ.

Souhrnné vysvětlení symbolů:

- **Xi –** vstup neuronu (každému náleží stav vstupu xi)
- **n –** počet vstupů (počet neuronů v předcházející vrstvě)
- **Y –** výstup neuronu (nabývá hodnot - stavů)
- **Wi –** váha spojení mezi vstupem Xi a výstupem Y
- **b –** bias (váha W_0 = -h)
- **Θ** – práh aktivační funkce neuronu
- **f (x) –** aktivační funkce neuronu (někdy uváděna jako přenosová fce)
- **α** - koeficient učení

3 Aktivační funkce:

Je to matematická funkce, převádející vstupní hodnoty (pod prahem aktivační funkce Θ) na relativně nízké výstupní hodnoty a vstupní hodnoty nad prahem Θ na poměrně vysoké výstupní hodnoty. Zjednodušeně řečeno, je to prahová funkce. Hodnota prahu pak určuje, zda je neuron v aktivním (aktivační fce větší než prahová hodnota) či pasívním stavu (aktivační fce je menší než práh).

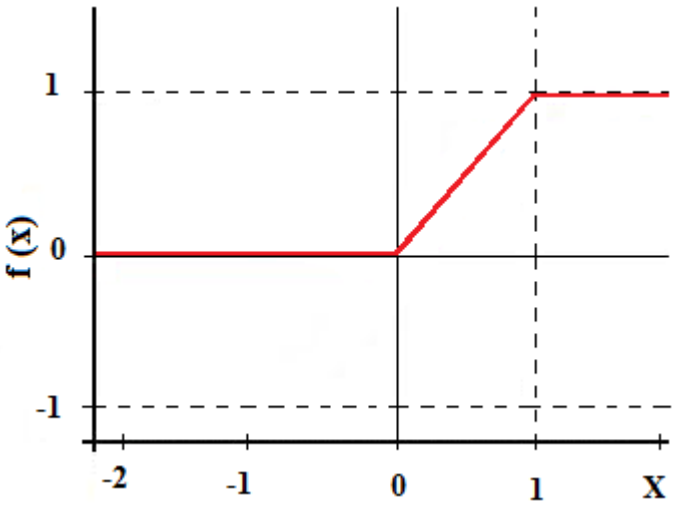
V jednovrstvém perceptronu je jako aktivační funkce použita funkce f = sign (x). Funkce signum není diferencovatelná a proto se u vícevrstvých sítí používají jiné funkce, které se již blíží svým průběhem ale diferencovatelné jsou [2], [12].

Aktivační funkce je možné definovat různými způsoby a nemusí pak tedy ani odpovídat biologické formě neuronu. Nejznámější aktivační funkce jsou:

3.1 Lineární saturovaná funkce

$$f(x) = \begin{cases} 1 \text{ (pokud } x > 1) \\ x \text{ (pokud } 0 \leq x \leq 1) \\ 0 \text{ (pokud } x < 0) \end{cases} \quad (2)$$

Tato funkce omezuje velikost vstupního signálu do rozsahu 0 až +1. Kopíruje vstupní hodnoty na výstupu, pouze při záporných hodnotách vstupu je nulová.

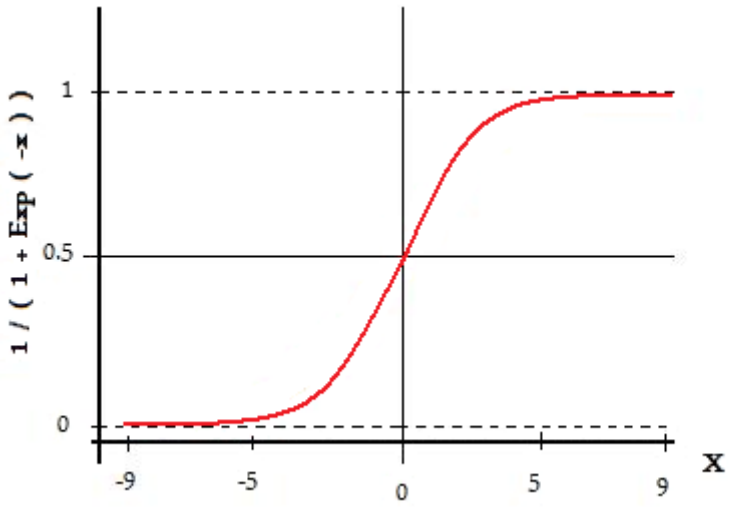


Obr. 3.1: Průběh Lineární saturační funkce neuronu

3.2 Standardní sigmoida

$$f\,(x) = \frac{1}{1+e^{-x}} \tag{3}$$

Hodnoty funkce se blíží k nule v minus nekonečnu a k jedničce v plus nekonečnu. Pokud je vstup nula, náleží mu výstupní hodnota 0,5. Výhodou sigmoidální funkce oproti skokové, je výskyt přijatelné derivace v každém bodě.

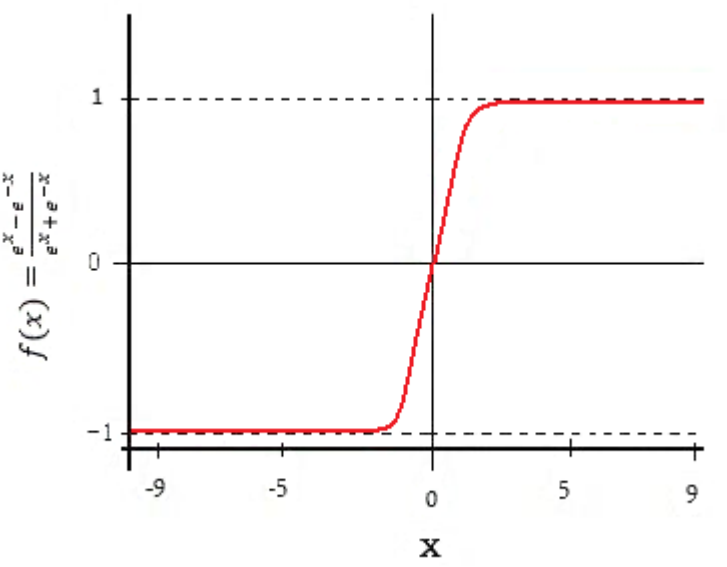


Obr. 3.2: Průběh sigmoidální aktivační funkce neuronu

3.3 Hyperbolický tangens

$$f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{4}$$

Pro minus nekonečno se blíží k minus jedničce, a naopak pro plus nekonečno se blíží k plus jedničce. V nule nabývá rovněž nulové hodnoty.

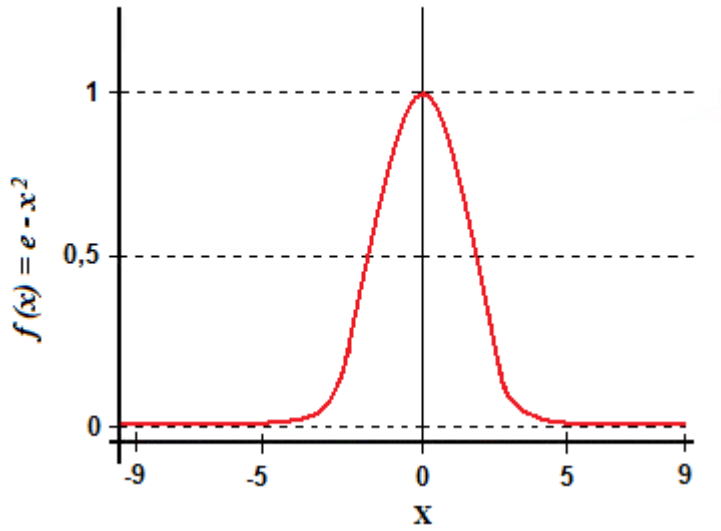


Obr. 3.3: Průběh aktivační funkce hyperbolické tangenty neuronu

3.4 Funkce radiační báze

$$f(x) = e^{-x.^2} \tag{5}$$

V minus nekonečnu se hodnoty blíží k nule. Stejně je tomu i v plus nekonečnu. V nule nabývá funkce hodnoty 1.

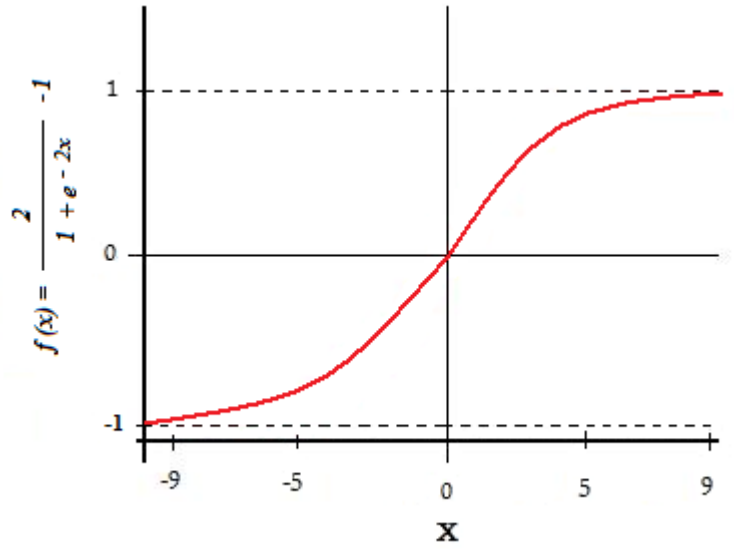


Obr. 3.4: Průběh aktivační funkce radiační báze neuronu

3.5 Tan-sigmoidální funkce

$$f(x) = \frac{2}{1+e^{-2x}} - 1 \tag{5.1}$$

Jde o nejpoužívanější aktivační funkci v oboru neuronových sítí. Vznikla kombinací funkcí tangens a sinoidy. V mínus nekonečnu se její hodnoty blíží k mínus jedné, naopak v plus nekonečnu k plus jedné. Zkráceně se uvádí jako tansig.



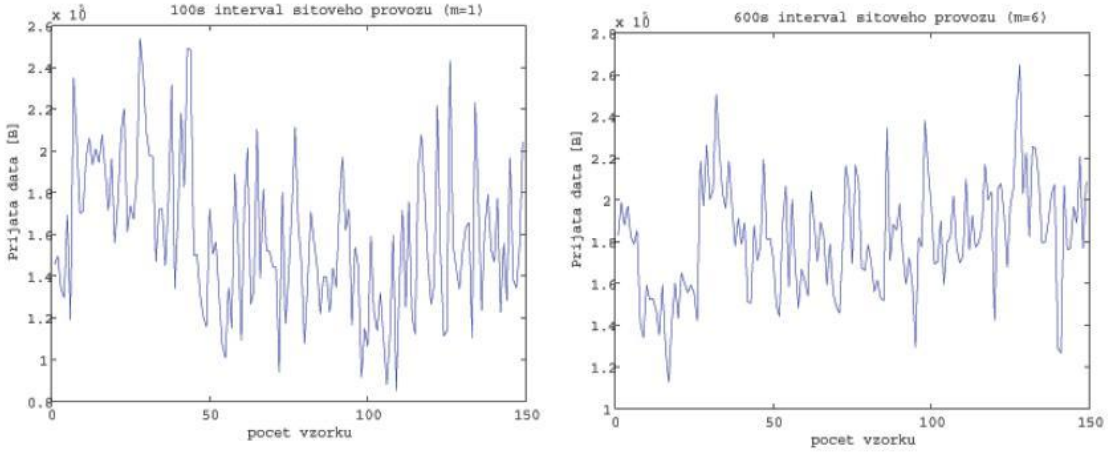
Obr. 3.5: Průběh tan-sigmoidální aktivační funkce neuronu

Velmi často býváme v praxi nuceni řešit problém neschopnosti matematicky popsat daný proces (dostatečnou přesností). Navíc, pokud se sestavení modelu podaří, je třeba brát ohledy na náročnost algoritmizace. Ať už časovou, či programovou.

4 Statické vlastnosti síťového provozu

Definice: Útvar nazveme samopodobným, když jej můžeme rozdělit na konečný počet částí „podobajících se původnímu tvaru (např. trojúhelník, čtverec, Cantorovo diskontinuum, Kochova vločka) [3].

Je dokázáno, že provoz (traffic) v ethernetové LAN síti je statisticky *samo- podobný* . Shlukující proudy takového provozu typicky nabývající na intenzitě samo-podobnosti ("protržení") místo vyhlazování [13]. Studie vlastností sítového provozu jsou podporované statistickou analýzou (vysoké kvality) z obrovského množství packetů v ethernetovém provozu sebraných v letech 1989 až 1992. Ty jsou samozřejmě spojeny s diskusí o podložní matematických a statistických vlastností samo-podobnosti a její vztah s aktuálním chováním sítí. Jde také o presentaci modelu provozu, založeného na samo-podobných náhodných procesech, které jsou jednoduché, přesné a jsou realisticky popsány provozními scénáři s očekávanou dobou trvání



Obr. 4.1: Agregované časové intervaly sítového provozu [5]

Ten prvotní krok učinili pánové Leland a Wilson, kteří byli schopni zaznamenat stovky milionů ethernetových paketů beze ztrát (nezávisle na provozním zatížení), včetně přesně zaznamenaných časových značek**.** Data byla zachycena v mezi srpnem 1989 a únorem 1992 v několika Ethemetových LAN-sítích ve výzkumném centru Bellcore Momstown.
Leland a Wilson prezentovali předběžnou statistickou analýzu těchto jedinečných vysoce kvalitních dat a detailně komentovali momentální situaci - "protržení" napříč extrémně širokému časovému měřítku: Provozní "špičky" trvají delší dobu a vytvářejí "zvlnění", které ve své podstatě "roste".

Tato ***samo-podobnost*** (coby základní vlastnost fraktálu) agregovaného ethernetového LAN provozu je velmi odlišná od konvenčního telefonního provozu a od aktuálních formálních modelů pro pakety provozu (např. model Pisson, či varianty jako Poisson - batch nebo Markov - Modulated Poisson). Vyžadujte nový pohled na modelování provozu a výkonu širokopásmových sítí.

V poslední době se ukazuje, že je možné presentovat princip samo-podobnosti jak na velkém tak i na lokálním měřítku. Existuje hypotéza vysvětlující možnosti samo-podobnosti použitím partikulárních podmnožin široké oblasti provozu: provoz v oblasti World Wide Web (WWW - celosvětový web). Použitím rozsáhlých sad přenosu, které přestavují přes půl milionu žádostí o WWW dokumenty je možné hledat strukturu WWW přenosu. Nejdříve je však třeba dokázat že WWW provoz má vlastnosti charakteristické pro samo-podobnostní modely přenosu.

4.1 Matematické vyjádření samo-podobnosti

Samo-podobná časová řada má tu vlastnost [14], že pokud je agregována novou řadou (kde každý bod tvoří součet originálních bodů), má stejnou autokorelační funkci jako originál. Představme si kovariační stochastický stacionární proces vyjádřený časovou řadou [4]
$X = (X_t ; t = 0, 1, 2,\dots)$ se směrodatnou odchylkou $\sigma^2 = \mathrm{Var}(X_t)$, střední hodnotou $\mu = E(X_t)$ a autokorelační funkcí r(k), $k \geq 0$. Nyní definujeme *m*-agregované řady

$$X^{(m)} = (X_k^{(m)} : k = 1, 2, 3, \dots), \tag{6}$$

coby novou kovariační stacionární časovou posloupnost, která představuje součet vzorků originální řady, nepřesahující velikost *m*.

Následně lze říci, že řada X je samo-podobná, jestliže má stejnou autokorelační funkci

$$r(k) = E\,[(X_t - \mu)(X_{t+k} - \mu)], \tag{6.1}$$

jako řada $X^{(m)}$ pro všechna *m*. Pro všechny $m = 1, 2, 3, \dots$ lze napsat novou kovariační stacionární časovou řadu (s korespondující autokorelační funkcí $r^{(m)}$) kterou získáme průměrováním vzorků originální řady X nepřesahující velikost m.

Takže pro každé $m = 1, 2, 3, \ldots, X^{(m)}$ dostáváme

$$X_k^{(m)} = \frac{1}{m}(X_{km-m+1} + \cdots + X_{km}), k \geq 1, \tag{6.2}$$

To v podstatě znamená, že řady jsou samo-podobně distribuovatelné. Distribuce agregované řady je tedy stejná jako v řadě originální. Ve výsledku můžeme konstatovat, že samo-podobné procesy ukazují svoji velkou míru závislosti. Ta se projevuje zejména na autokorelační funci $r(k) \sim k^{-\beta}$ přičemž $k \to \infty$, a kde $\beta \in (0, 1)$. Taková funkce náleží procesu jehož charakteristika slábnutí má hyperbolický průběh (v porovnání s exponenciálním průběhem slábnutí u standardních modelů sítového provozu).

5 Možnosti predikce síťového provozu

Studie zahrnují různé topologie sítí a různé typy provozů (lineární a nelineární ARMA modely, stavové prostory a modely založené na kanonické korelaci).

Predikce jsou zpravidla porovnávány se dvěma situacemi:

- výsledkem je střední hodnota časové řady
- výsledkem je poslední záznam časové řady

Velké úsilí bylo vynaloženo na rozvoj dynamických kontrolérů pro počítačové sítě. I když byl tento krok důležitý při historickém vývoji sítí, musela mu předcházet zejména schopnost dobře porozumět otevřeným systémům.

Postupem času byly navrženy různé metody pro sestavení modelů sítového provozu, pomocí nichž bylo možné realizovat predikci. Jedná se zejména o FARIMA model (The Fractional Autoregressive Intergrated Moving Average), který vychází z obecnějšího ARIMA modelu. Skládá se z autoregresního procesu a klouzavého průměru. Nevýhodou toho modelu je zejména neschopnost zachytit nelineární síťový provoz. Dále také obtížná volba počátečních parametrů, na které závisí kvalita predikce.

Další možnou metodou je predikce pomocí Teorie chaosu, která využívá hlavně rysu samo-podobnosti síťového provozu. Vlastnosti provozu lze zjistit již ze zadané časové řady a není tak potřeba vytvářet model, jako tomu bylo u předchozí metody. To má za následek zkvalitnění predikce (zejména přesnost a spolehlivost). Jak již bylo naznačeno v úvodu práce, velmi rozšířenou oblastí se stala predikce pomocí Neuronových sítí.

5.1 Umělá neuronová síť UNS

Jak již bylo uvedeno výše, umělé neuronové sítě jsou inspirovány živými biologickými strukturami nervové soustavy. Velmi důležitou částí při sestavování návrhu neuronových sítí je reprezentace dané topologie sítě a vlastní činnosti biologické neuronové sítě.
Analogie je patrná i mezi základními prvky obou typů sítí. Na jedné straně neuron, coby živá buňka. A formální neuron, jenž je matematickým zápisem svého přírodního vzoru, na straně druhé. Rovněž synapse jsou zahrnuty do matematického modelu, a to v podobě vah spojení (podle kladné, či záporné hodnoty rozlišujeme excitační, případně inhibiční charakter). Reprezentace znalostí je řešená právě velikostí hodnot jednotlivých vazeb.

V komplexním pohledu jde o adaptivní, distributivní, nelineární nástroj, sestavený z mnoha prvků pro zpracování informace [15]. Každý prvek může být propojen různě s ostatními prvky, nebo také sám se sebou, kdy vytváří zpětnou vazbu.

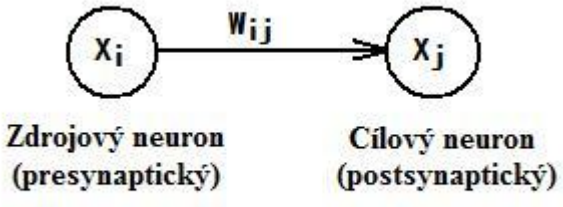
V části věnované aktivačním funkcím jsme zmínili častý výskyt problému, který se týká neschopnosti matematicky popsat danou situaci (s určitou složitostí). Z tohoto pohledu umělá neuronová síť (dále jen UNS) představuje universální funkční aproximátor. To znamená že navržená a podle určitých matematických pravidel „naučená“ UNS je následně činnost výchozího procesu s určitou mírou přesnosti reprodukovat (simulovat) pro různé vstupní signály. [5]

Charakteristika UNS:

- Topologie sítě (organizační etapa)
- Obsažené modely neuronů (organizační etapa)
- Způsob učení sítě (adaptační etapa)
- Způsob vybavování sítě (aktivační etapa)

6 Topologie neuronových sítí

Topologií (strukturou, architekturou) neuronové sítě rozumíme konkrétní umístění jednotlivých neuronů v síti a jejich vzájemné propojení. Každá NS se skládá z formálních neuronů, vzájemně propojených tak, že výstup jednoho neuronu je vstupem do (jednoho nebo více) dalšího neuronu. Předlohu hledejme opět v neurologii, kde terminály axonu biologického neuronu jsou spojeny s dendrity jiných neuronů synaptickými vazbami. Celá struktura tak tvoří libovolný orientovaný graf s vrcholy a orientovanými hranami. Vrcholy grafu jsou tvořeny neurony a orientované hrany jejich propojeními (ohodnocenými vahami).



Obr. 6.1: Spojení dvou neuronů

Podle umístění neuronů lze rozlišovat tyto prvky na nesynaptické (před synapsí) a postsynaptické (po synapsi). Nebo-li zdrojové a cílové. Synaptické váhy jsou označeny symbolem Wij (tzn. od i-tého zdrohového k j-témů cílovému neuronu). Společným rysem všech topologií UNS je většinou jejich vrstevnatá struktura. Jednotlivé neurony jsou organizovány do vrstev. V závislosti na momentálním využití a poloze v neuronové síti můžeme vrstvy (potažmo i neurony v nich obsažené) rozdělovat na:

- vstupní
- pracovní (mezilehlé, skryté, vnitřní)
- výstupní

Informace se šíří prostřednictvím změny stavů neuronů, ležících na cestě mezi vstupními a výstupními neurony. Stav neuronové sítě určují stavy neuronů a konfiguraci sítě určují synaptické váhy.

Vzhledem k časovému chování neuronové sítě (změna stavu neuronů, adaptace vah), lze vymezit tři etapy vývoje [10]:

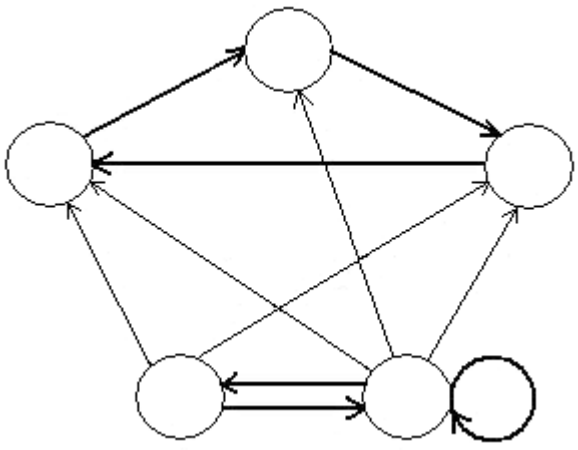
- Organizační etapa (změna topologie)
- Adaptační etapa (změna konfigurace)
- Aktivační etapa (změna stavu)

6.1 Organizační etapa

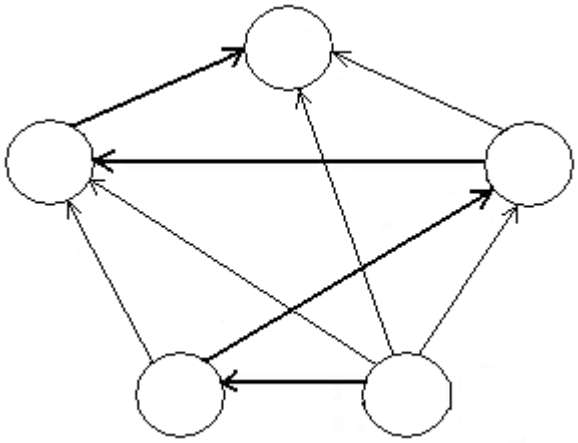
Jak již bylo naznačeno, tato etapa specifikuje architekturu neuronové sítě a její případnou změnu. Ta je v rámci adaptivního režimu realizována rozšířením sítě o další neurony a příslušné spoje. Nicméně, organizační etapa většinou předpokládá pevnou (tj. v čase neměnnou) architekturu neuronové sítě. Podle způsobu toku dat sítí rozlišujeme dva typy architektur:

- Rekurentní (cyklická) síť
- Dopředná (acyklická) síť

V případě rekurentní architektury existuje v síti skupina neuronů, propojená v kruhu. Tzn. řetězcově za sebou, přičemž výstup posledního neuronu z oné skupiny představuje vstup prvního neuronu. Takže dohromady tvoří cyklus. Speciálním případem je tzv. úplná topologie rekurentní sítě, kdy výstup libovolného neuronu je vstupem každého neuronu (představuje největší počet cyklů). Naopak nejjednodušším případem je zpětná vazba neuronu (jehož výstup je zároveň vstupem).



Obr. 6.2: Obecná rekurentní topologie NS s vyznačenými cykly



Obr. 6.3: Obecná dopředná topologie NS

Jak je patrné z obrázků u dopředné neuronové sítě neexistuje cyklus. Neurony jsou uspořádány do vrstev a tvoří následující schéma: Spoje mezi neurony (cesty) vedou pouze z nižších vrstev do vyšších (dopředná struktura). Ale je zde také možné přeskakování vrstev.
Shrneme-li předchozí poznatky, lze říci, že u dopředné sítě je směr toku dat striktně od vstupu k výstupu. Data postupují v síti po vrstvách, nikdy sít neobsahuje zpětnou vazbu.Naopak rekurentní sítě zpětné vazby obsahují (nebo smyčky). Důležité je zde také dynamické chování sítě. Může dojít k tomu, že výstup je do jisté doby neměnný a stabilní, protože přechodový stav trval krátkou dobu. U typů rekurentních sítí je tomu jinak, a výstup lze popisovat jako dynamický děj. [5]

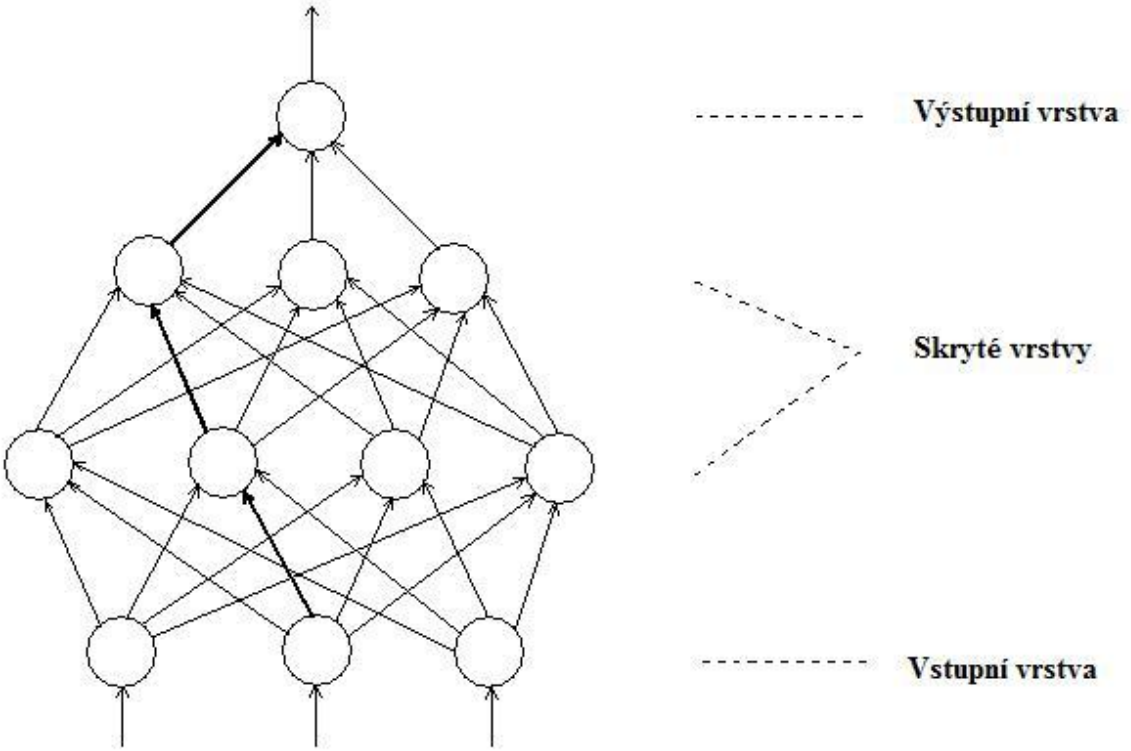
6.1.1 Vícevrstvá perceprtonová síť MLP

MLP (Multilayer Precepton Neutral Network) je nejpoužívanější a nejznámější neuronovou sítí. Základním stavebním kamenem je neuron, nebo-li perceptron. (v roce 1957 jej popsal F.Rosenbaltten, včetně algoritmu učení). Avšak algoritmus učení vícevrstvé preceptorové sítě byl popsán až v roce 1986 Rumelhartem, Hintonem a Wiliamsem. Následně bylo možné vyřešit vyřešit problém log. fce XOR pomocí sítě tří neuronů, což pomocí jednoho neuronu nelze.

Struktura stítě: Jak již bylo řečeno, tato síť obsahuje neurony (percerptrony) , vzájemně propojené tak, že ve výsledku tvoří vrstevnatou síť.
Percerptonu: Jde o speciální typ formálního neuronu – obecného prvku neuronových sítí, kde vnitřní potenciál je počítán jako vážený součet vstupů.

Struktura vícevrstvé sítě se vyznačuje tím, že neurony v jedné vrstvě jsou bezprostředně spojeny s neurony v další vrstvě (následující) [8]. Tím pádem je možné architekturu specifikovat pouze zadáním zjednodušeného formátu struktury. Ten obsahuje čísla oddělená pomlčkou. Přičemž, pozice čísla určuje o kterou vrstvu jde (směr od vstupní k výstupní vrstvě) a hodnota čísla udává počet neuronů v konkrétní vrstvě obsažené.

Vezměme si např. topologii 3 – 4 – 3 – 1. Ze zadaného zápisu můžeme vyčíst, že vstupní vrstva obsahuje 3 neurony, výstupní vrstva 1 neuron. Dále struktura obsahuje dvě mezilehlé skryté vrstvy, o počtech neuronů 4 a3. Detailněji můžeme celou situaci vidět na následujícím obrázku.



Obr. 6.4: Příklad topologie vícevrstvé neuronové sítě (3 - 4 - 3 - 1)

6.1.2Volba vrstev

Při návrhu topologie celé sítě je jedním ze stěžejních úkonů volba počtu skrytých vrstev. Universální řešení neexistuje, každá praktická situace si žádá jinou topologii k dosažení uspokojivých výsledků. Nicméně, zkušenosti ukazují [9], [11], že vyhovující aproximace

chování modelované soustavy lze dosáhnout v mnoha případech pomocí dvouvrstvé neuronové sítě. Pro stanovení počtu neuronu ve skryté vrstvě můžeme psát:

$$p = \sqrt{mn}\,, \tag{7}$$

nebo také přesněji

$$p \geq m + n, \tag{7.1}$$

kdy n - počet terminálů, m - počet výstupních neuronů

V případě sítě se dvěma skrytými vrstvami lze volit počty neuronů pomocí vztahů:

$$p_1 = mr^2, \quad p_2 = mr, \tag{7.2}$$

přičemž

$$r = \sqrt[3]{n/m}, \tag{7.3}$$

Nicméně, je třeba dávat pozor na tzv. „přetrénování“ sítě. To se může stát, v případě nevhodně zvoleného velkého počtu vrstev [9]. Projevem takového chování je znatelně zvýšená doba učení a špatná schopnost generalizace, vlivem nadměrného množství trénovacích dat. Často se však můžeme setkat s tím, že uvedené vztahy nepřinášejí pro své aplikace dobré výsledky a tak jsme odkázáni na metodu pokus/omyl, kdy zkoušíme měnit parametry sítě víceméně náhodně. MLP pracuje s reálnými hodnotami, tzn. všechny vstupy, výstupy, váhy, stavy i potenciály jsou obecně reálnými čísly. Aby síť fungovala správně, musí mít vhodně nastaveny váhy. Jejich nastavováním se zabývá učení sítě [1].

6.2 Adaptační etapa

Určuje počáteční konfiguraci sítě a způsob změny jednotlivých vah spojení mezi neurony (v čase). Výčet všech možných konfigurací sítě tvoří váhový prostor sítě. Nejdříve je třeba nastavit váhy spojů na určité hodnoty (třeba náhodně). Následně při načtení konkrétní konfigurace se hodnoty vah přizpůsobí na potřebné hodnoty. Předpokládáme spojitý model se spojitým vývojem konfigurace neuronové sítě v čase (váhy tvoří spojitou časově závislou funkci, obvykle určena diferenciální rovnicí). Čas adaptace je však diskrétní. Účelem adaptační etapy je nalezení takové konfigurace sítě z váhového prostoru, která by v aktivační etapě řešila předepsanou funkci. Neuronová síť organizuje tréninkové vzory a analyzuje jejich souhrnné vlastnosti.

6.2.1 Učení

Speciálním případem je učení. Jde o vytváření konfigurace pro danou funkci (programování) na míru, respektive o nastavování vah spojení w_{ij} [7] tak aby sít vytvářela správnou odezvu na vstupní signál.

Základem je tzv. Trénovací množina (posloupnost) vstup/výstup, určující požadovanou funkci. Snažíme se tedy, aby odchylka mezi aktuálními a požadovanými výstupy sítě byla minimální [1]. Tuto odchylku můžeme definovat jako:

$$E = \sum_k E_k \,, \tag{8}$$

kde index *k* značí probíhání přes všechny trénovací vzory a E_k chybu odpovídající *k*-tému trénovacímu vzoru. Pro ni lze dále přesněji napsat

$$E_k = \frac{1}{2}\sum_j \left(y_j - d_{kj}\right), \tag{8.1}$$

přičemž index *j* označuje probíhání přes výstupní neurony a d_{kj} představuje *j*-tý element požadovaného výstupu *k*-tého trénovacího vzoru. Pro optimalizaci chyby se v základním modelu sítě používá gradientní metoda. Ta spočívá v nastavení všech vah (včetně biasů) na malé náhodné hodnoty se střední hodnotou v okolí nuly. Následně jsou předkládány jednotlivé trénovací vzory a pro každý je spočítána chyba pomocí vybavovací fáze (8.1). Tato chyba se akumuluje. Takto opakujeme postup pro všechny trénovací vzory a nakonec dostáváme výslednou chybu pomocí vztahu (8). Změna vah tedy bude:

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij} \,, \tag{8.2}$$

kde

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}, \tag{8.3}$$

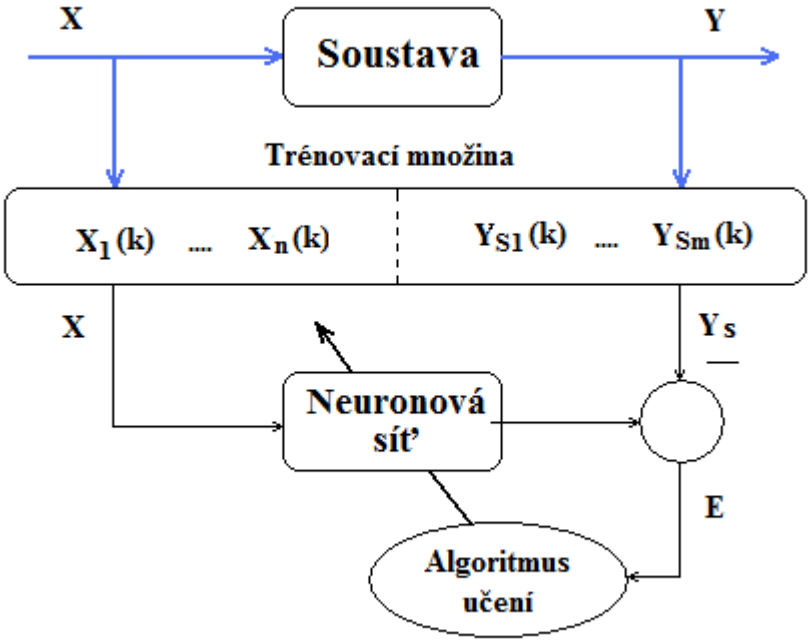
přičemž η je parametr učení ($0 \leq \eta \leq 1$).

6.2.2 Trénování NS – Back Propagation (BP)

Někdy se tento algoritmus nazývá též *zobecněné delta pravidlo.* Byl navržen v roce 1986 Rumelhartem, Hinonem, a Williamsem, avšak podobné návrhy byly již v letech 1976 nebo 1982.

Tento algoritmus spočívá v opakovaném připojování souboru vzorů chování sítě (trénavací množiny TRM) na vstupy a následném hledání optimálních parametrů a nastavení sítě, při kterých bude definovaná funkce rozdílu výstupu modelované soustavy a výstupu neuronové sítě (chyba E) minimální. Jedná se zejména o vektor vah i strmostí aktivačních funkcí γ. Trénovací množinu lze získat dvěma způsoby. Buď z matematického popisu sledovaného procesu (numerická simulace, modelování či řízení systémů), nebo experimentálně měřením vstupních a výstupních hodnot modelovaného technologického procesu [9]. Pokud jde o statickou soustavu je popisem algebraická rovnice. Jedná-li se dynamickou soustavu, je popisem diferenční či diferenciální rovnice. Výsledkem jejich řešení je trénovací množina.

Epocha – jeden průchod trénovací množiny neuronovou sítí.



Obr. 6.5: Schéma modelu trénovaní sítě

Osnova algoritmu BP:

1. Inicializace vah
2. Přiložení trénovací množiny (vstupy tvoří žádaný výstup)
3. Výpočet skutečného výstupu
4. Adaptace vah – učení

k	Vstupy				Výstupy			
1	$X_1(1)$	$X_2(1)$	...	$X_n(1)$	$Y_1(1)$	$Y_2(1)$	...	$Y_m(1)$
2	$X_1(2)$	$X_2(2)$	...	$X_n(2)$	$Y_1(2)$	$Y_2(2)$	...	$Y_m(2)$
...	...				...			
N	$X_1(N)$	$X_2(N)$	...	$X_n(N)$	$Y_1(N)$	$Y_2(N)$	...	$Y_m(N)$

Tab. 1: Trénovací data

Podle způsobu aplikace trénovacích dat lze rozlišit dva režimy:
Trénování off-line – sestavení kompletní trénovací množiny a její následná aplikace na NS
Trénování on-line – okamžitá aplikace získaného trénovacího vzoru (zejména při adaptivním řízení)

Učení s učitelem (supervised):

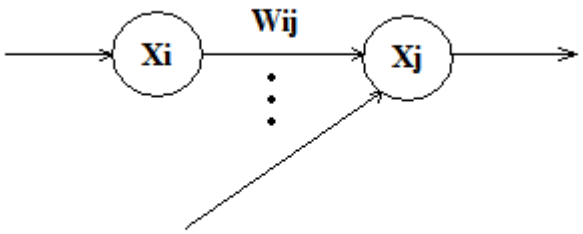
Variantě adaptační etapy [6], obsahující mechanismus, jenž informuje systém o správných výstupech náležících příslušným vstupům, říkáme učení s učitelem. Klasifikované učení – mechanismus hodnotí aktuální skutečné odpovědi (výstupy) známkami doplněnými místo výstupu sítě.

Učení bez učitele (unsupervised):

Není zde žádný kontrolující mechanismus jako v předcházejícím případě. Trénovací množina navíc obsahuje jen vstupy (z důvodu absence kontrolovacího mechanismu). Jde o tzv. samoorganizaci.

Hebbovo pravidlo učení

(Hebb rule, Hebbian learning) Je pravidlo učení v neuronových sítích: jestliže jsou dva neurony aktivní ve stejnou dobu a současně jsou spojeny synapsemi, pak síla spojení synapsemi bude posilována. Změna hodnoty synaptické váhy je přímo uměrná součinu vstupních hodnot aktivovaných neuronů [7].



Obr. 6.6: Hebbovo pravidlo učení

Charakteristická rovnice uvedeného případu je

$$w_{ij}(t+1) = w_{ij}(t) + \alpha(X_i X_j), \tag{8.4}$$

kde α je tzv. Koeficient učení.

6.3 Aktivační etapa

Specifikuje počáteční stav sítě a způsob jeho změny v čase při konkrétní topologii a konfiguraci [6]. Nejprve se nastaví stavy vstupních neronů na tzv. vstupy sítě (zbylé neurony jsou v počátečním stavu). Jak již víme, všechny možné vstupy (stavy sítě) tvoří vstupní prostor (resp. stavový prostor). Většinou předpokládáme diskrétní čas systému, kdy na počátku je síť v bodu 0. Její stav se mění v bodech 1, 2, 3, 4, atd. V každém časovém bodu se mění stav jednoho nebo více neuronů, které jsou vybrány podle pravidla aktivní dynamiky.

Dle způsobu aktualizace změny stavu neuronů rozlišujeme modely:

- Synchronní (aktualizace stavů je řízena centrálně)
- Asynchronní (stav se mění nezávisle na sobě)

Důležitá je také funkce neruronové sítě, jejíž podoba závisí na použité konfiguraci a topologii sítě, coby neměnných parametrech. Jde o přiřazování výstupních hodnot vstupním hodnotám. Aktivační etapa neuronové sítě také určuje aktivační funkci jejíž předpis je

většinou pro všechny nevstupní neurony stejný (viz aktivační funkce, kapitola 3). Zjednodušeně řečeno, na vstupy sítě přiložíme vzor a ten pak postupně šíříme (transformujeme) dopředně přes váhy neuronů v jednotlivých vrstvách s využitím aktivačních funkcí perceptronů až v k výstupům sítě [1]. Pro počítání hodnot potenciálů od první k poslední vrstvě lze psát

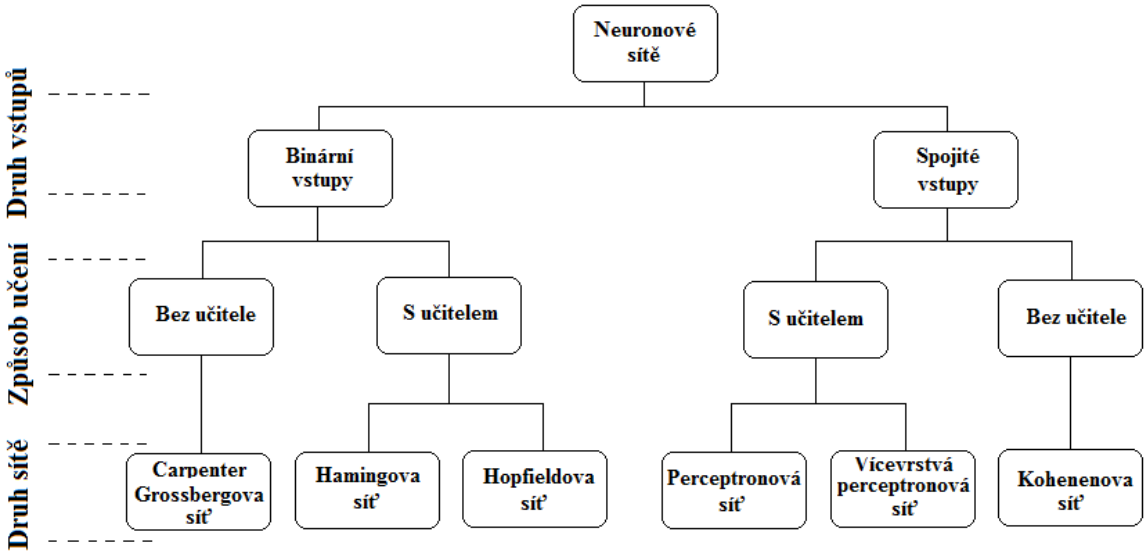
$$\xi_j = \sum_i w_{ij} x_i \,, \tag{9}$$

kde index i značí procházení jednotlivými vrstvami (pro aktuální je třeba vypočítat vždy tu předchozí, pro první vrstvu se užije tatéž vrstva), w_{ij} značí váhy mezi i-tým a j-tým neuronem, i a j jsou tedy indexy neuronů sousedících vrstev. Tím dostáváme hodnoty vstupů pro další vrstvu, kterou můžeme vypočíst dle vztahu

$$X_j = f(\xi_j), \tag{9.1}$$

kde f je aktivační funkce – sigmoida (obecně lez zvolit libovolnou diferencovatelnou funkci). Nárůst sigmoidy v okolí počátku určuje parametr strmosti λ. Většinou bývá pevný v rámci celé sítě, jeho hodnota se pohybuje blízko jedné.

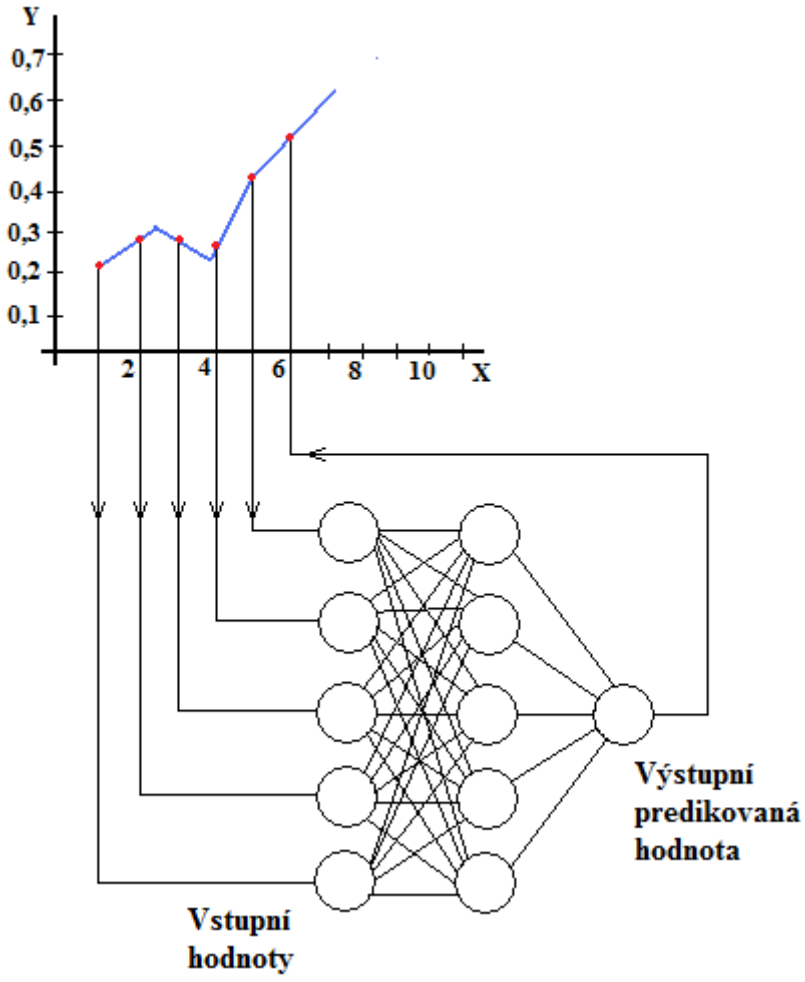
6.4 Rozčlenění neuronových sítí



Obr. 6.7: Modely neuronových sítí

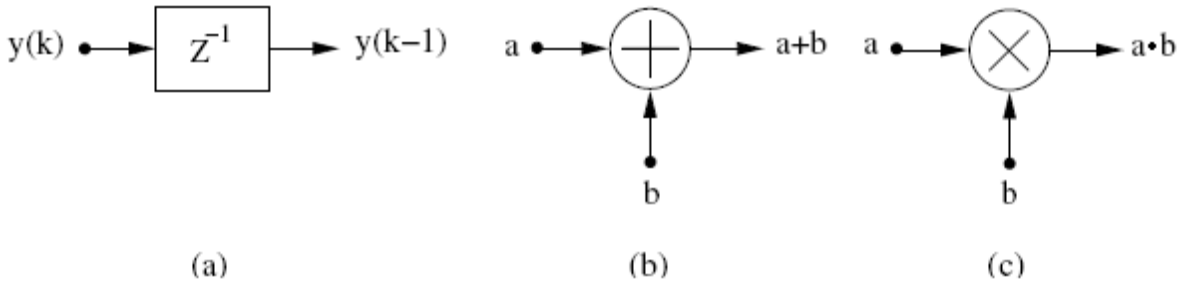
7 Praktická realizace

Jak již bylo předesláno, lze rozlišit dva případy učení. V pokračování této práce bude používán (testován) typ Učení s učitelem. To znamená, že v praxi využíváme zachycená data, odpovídající učitému síťovému provozu (zachycené pakety) a pokusíme se jimi natrénovat neuronovou síť. Nejprve však musíme data získat a upravit do přijatelné podoby. Na japonských www stránkách (http://mawi.wide.ad.jp) jsou data k dispozici volně ke stažení. Pocházejí sice „až“ z roku 2001, nicméně pro naše testovací účely a pro aplikaci predikce, jsou ideální. Ukázka takových dat je v příloze B. Po jejich stažení je třeba tyto zachycené pakety agregovat do časových intervalů, abychom s nimi byli schopni dále pracovat. Následuje postup, který je popsán již v teoretické části (viz kapitola 6.2). Uvedenou situaci dokumentuje následující obrázek.



Obr. 7.1: Realizace predikce pomocí neuronové sítě

Základními bloky všech diskrétních systémů pro predikci jsou sčítačky, násobičky, zpožďovací členy a v případě nelineárních systémů také paměťové členy průchodu nulou (zero-memory nonlinearites). Konkrétní způsob jejich vzájemného propojení určuje architekturu prediktoru.



Obr. 7.2: Základní bloky predikčních systémů: a) zpožďovací člen b) sčítačka c) násobička

Lineární struktury jsou jádrem oboru digitálního zpracování signálu a lze je dále dělit podle typu impulzní charakteristiky odezvy na:

Systémy s konečnou impulzní odezvou FIR (finite impulse response)

Systémy s nekonečnou impulzní odezvou IIR (infinite impulse response)

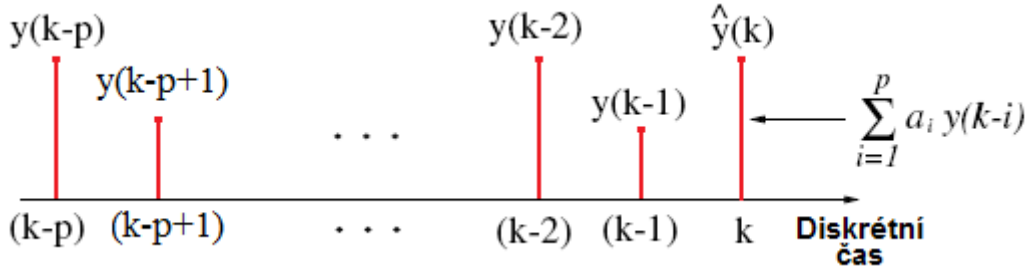
FIR filtry všeobecně nevyužívají ke své činnosti zpětnou vazbu na rozdíl do IIR filtrů, jenž díky ní redukují počet parametrů nezbytných pro svou funkci. Přítomnost zpětné vazby u IIR filtrů je opatřením pro zlepšení stability.

V oboru modelování signálů jsou FIR filtry více známé jako průměrně proměnlivé struktury (moving average -MA) a IIR filtry jako autoregressní (AR) nebo také autoregressní struktury se střední změnou (ARMA). Tu nejzákladnější verzi nelineárního struktury bychom mohli popsat jako vložení nelineární operace do výstupní fáze filtru FIR nebo IIR. Podobně lze popsat další nelineární struktury (NAR, NMA nebo NARMA).

Dopředné neuronové sítě (bez zpětné vazby) mohou představovat FIR/MA/NMA filtry naproti tomu rekurentní neuronové sítě zahrnují IIR/AR/NAR/NARMA filtry.

Takové filtry nacházejí okamžité uplatnění v oboru predikce diskrétního náhodného signálu, produkovaného obecně nelineárním systémem. Navíc jsou velmi úzce spjaty s jednotlivými neurony neuronových sítí.

Vezměme si konkrétní případ. {Y(k)} představuje reálný diskrétní náhodný signál, *k* je index bodu časové řady.



Obr. 7.3: Základní koncept lineární predikce

Princip predikce diskrétního signálu (Obr. 7.1) tvoří základ lineárního predikčního kódování (LPC), z něhož vychází spousta dalších kompresních technik. Hodnota signálu *y(k)* je predikována na základě součtu hodnot minulých. V našem případě y(k-1),y(k-2),…,y(k-p). Pomocí koeficientů a_i, kde i = 1, 2, 3,…, p získáme ŷ(k). Chybu predikce e(k) lze pak vypočíst:

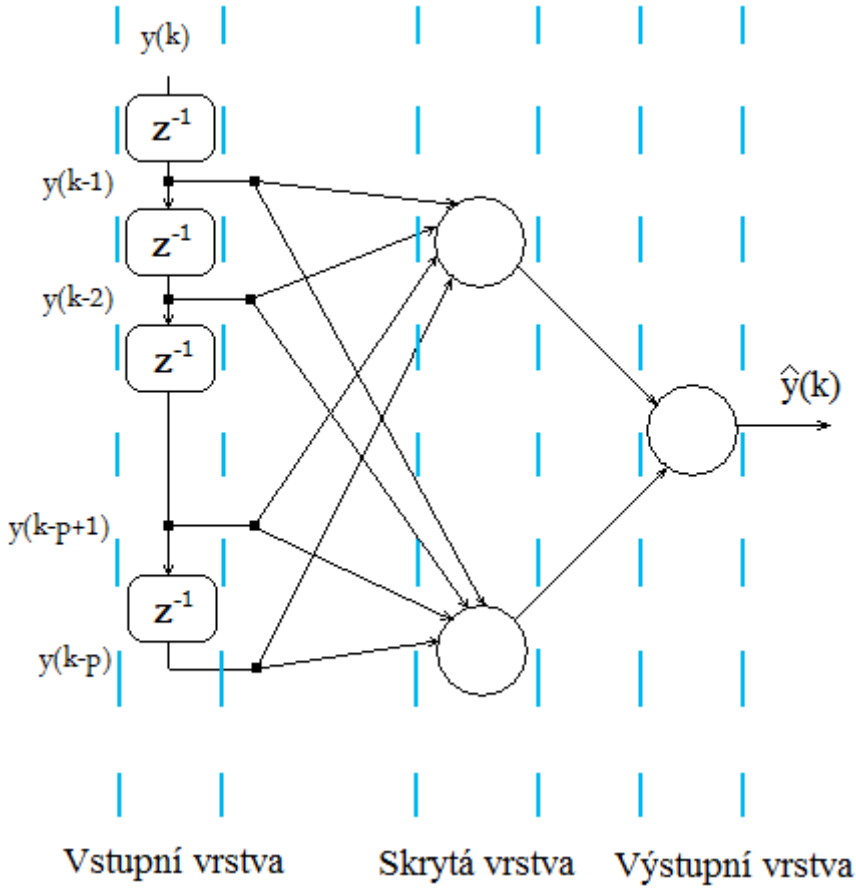
$$e(k) = y(k) - \hat{y}(k) = y(k) - \sum_{i=1}^{p} a_i y(k-i) \qquad (10)$$

Velikost koeficientů a_i závisí na minimalizaci některé z chybových funkcí - nejčastěji na střední kvadratické chybě $E[e^2(k)]$, kde E je statistický operátor a {y(k)} představuje stálé spektrum s nulovou střední hodnotou.

Dopředná neuronová síť (Feedforward neural network):

Jak již bylo předesláno, dopředná neuronová síť se skládá z neuronů, tvořící vrstvy. Jejich vzájemné propojení je pouze v přilehlých vrstvách (nikoli v jedné vrstvě navzájem), a bez zpětné vazby (Obr. 7.4). Těchto vrstev může být různé množství, ale platí, že první vrstva je vstupní, pak následuje jedna nebo několik vrstev skrytých a nakonec vrstva výstupní.

Vstupní vrstva se skládá ze vstupních terminálu (uzlů), jenže mají za úkol přivádět signály z okolí do sítě. V podstatě jde o pasivní prvky s jednotkovým přenosem. Rovněž slouží k rozdělení vstupních signálů pro aktivní neurony další vrstvy. Celý systém je zakončen vrstvou výstupní, která přenáší signály z neuronové sítě do okolí. Takže např. dvouvrstvá síť je tvořena jednou skrytou vrstvou a jednou výstupní vrstvou [9]. Neurony zpravidla bývají stejného druhu (nebo je jen malý počet druhů). V rámci jedné vrstvy jsou však neurony stejného druhu.



Obr. 7.4: Vícevrstvá dopředná neuronová síť

Při volbě počtu skrytých vrstev je třeba mít se na pozoru. Jelikož příliš velké množství skrytých vrstev (ale i neuronů v nich obsažených) může vést k přetrénování celé sítě (viz kap. 6.2.1). Proto je v praxi velmi častým řešením použití metody „pokus-omyl“, kdy v podstatě zkoušíme různé varianty a porovnáváme výsledky. Souvisí to s již zmíněnou variabilitou koncepcí řešených úloh.

Trénování dopředné neuronové sítě:¨

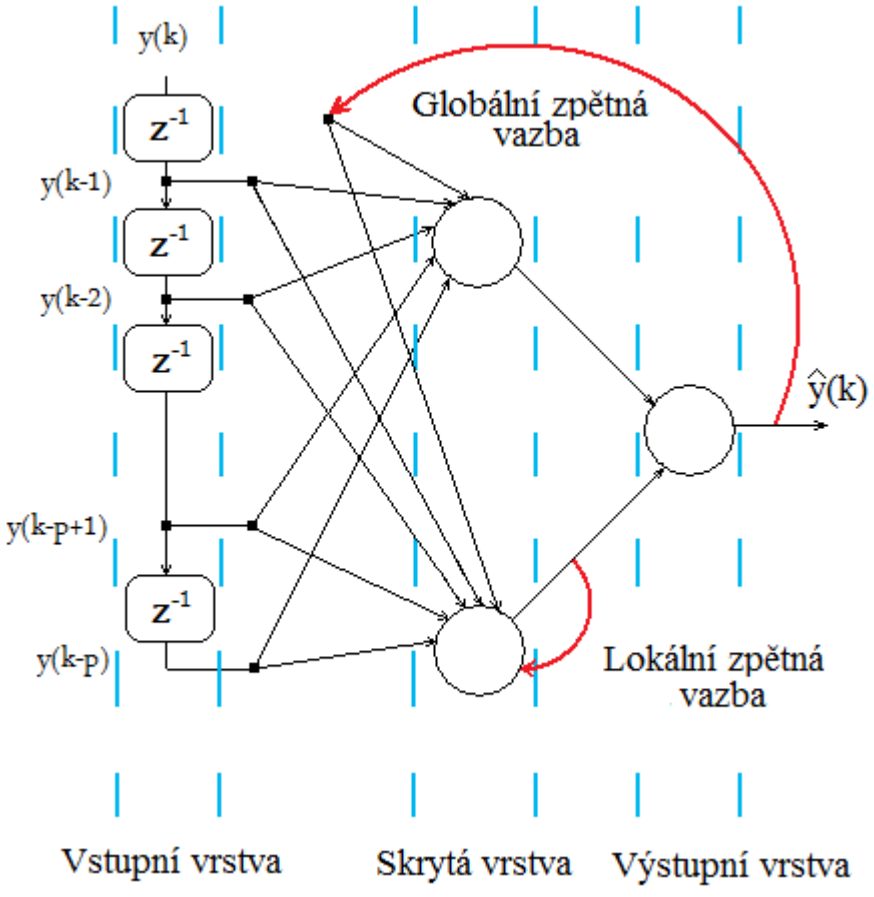
Abychom byli schopni dosáhnout s DNS přijatelných výsledků, musíme využít její nejsilnější stránky - schopnosti učit se. Schéma učení DNS je shodné s obecným principem učení sítí probraným v kapitole 6.2.2. Jedná se o algoritmus backpropagation (zpětné šíření chyby). Připomeňme si alespoň základní princip této metody.

Na vstup NS je opakovaně připojována sada vzorů chování sítě. Tu získáme buď matematickým popisem sítě, nebo prakticky z naměřených hodnot. Následně je síť trénována touto trénovací množinou (tzn. učí se, jakým vstupům náleží jaké adaptované váhy a strmosti

aktivačních funkcí). Snažíme se přitom, aby odchylka mezi skutečnou budoucí a vypočtenou (predikovanou) hodnotou byla co nejmenší. Je-li v požadovaném rozmezí, považujeme síť za naučenou a můžeme přiložit konkrétní data ze sítě, se kterou chceme dále pracovat. Na základě takto získaných výsledků lze stanovit závěry, potažmo učinit opatření za účelem zlepšení charakteristik sledované sítě.

Rekurentní neuronová síť (recurrent neural network):

Rekurentní neuronová síť na rozdíl od dopředné disponuje zpětnými vazbami (jednou nebo více). Ty je možné dělit na lokální nebo globální, podle toho, zda přemosťují jednu či více vrstev. Jednotlivé modely sítí, které budou dále testovány se vyznačují právě určitým druhem zpětné vazby. Např. pro síť NARX je charakteristická globální zpětná vazba, pro Elmanovu síť je to zase vazba lokální. Tyto vazby jsou velmi důležité, zavádějí do modelu zpoždění, a s jejich pomocí lze lépe predikovat.

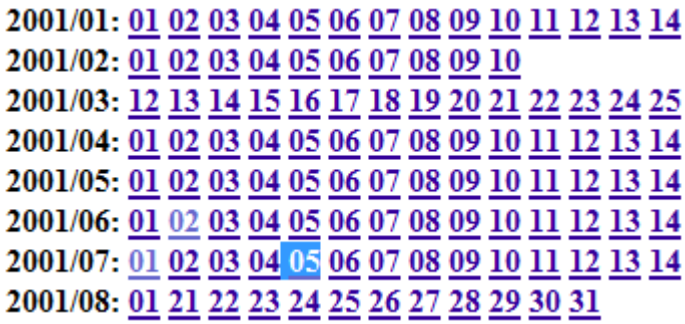


Obr. 7.5: Rekurentní neuronová síť s globální a lokální zpětnou vazbou

8 Předzpracování dat (agregace intenzit provozu)

Data jsou pořízena z adresy http://mawi.wide.ad.jp, kde je celá databáze zachycených paketů v jednotlivých režimech i časech síťového provozu. V našem případě jde o režim provozu protokolu IPv6. Lze si vybrat přesné období. Pro splnění naší úlohy byla použita data 2001/07:05, což znamená dopolední provoz v rozmezí:

Počáteční čas: úterý, 5. července, 09:00:01, 2001
Konečný čas: úterý, 5. července, 11:49:56, 2001



Obr. 8.1: Přehled databáze zachycených paketů

Nyní máme data stažena, ale je třeba je ještě upravit do přijatelné podoby. Toho dosáhneme tzv. agregací intenzit provozu, kdy v podstatě sčítáme tyto intenzity v rámci každého intervalu. Takže původní informace se nezmění, jen bude reprezentována v přijatelnější podobě. K tomuto účelu bylo použito vývojové prostředí Visual Studio 2005 pomocí něhož jsme agregovali dané intenzity v intervalu 100ms (příloha C.1).

```
1.301584 8 7 2227 119 0          1.1 11887
1.307440 1 125 80 2756 1         1.2 19985
1.307441 90 22 2429 25 21        1.3 11476
1.308416 5 6 3434 80 0           1.4 12856
1.309392 1 125 80 2756 1         1.5 10435
1.311344 5 6 3434 80 0           1.6 23995
1.317201 55 10 1053 20 0         1.7 17640
1.317202 22 96 25 1279 8         1.8 31874
1.319152 22 96 25 1279 8         1.9 19214
```

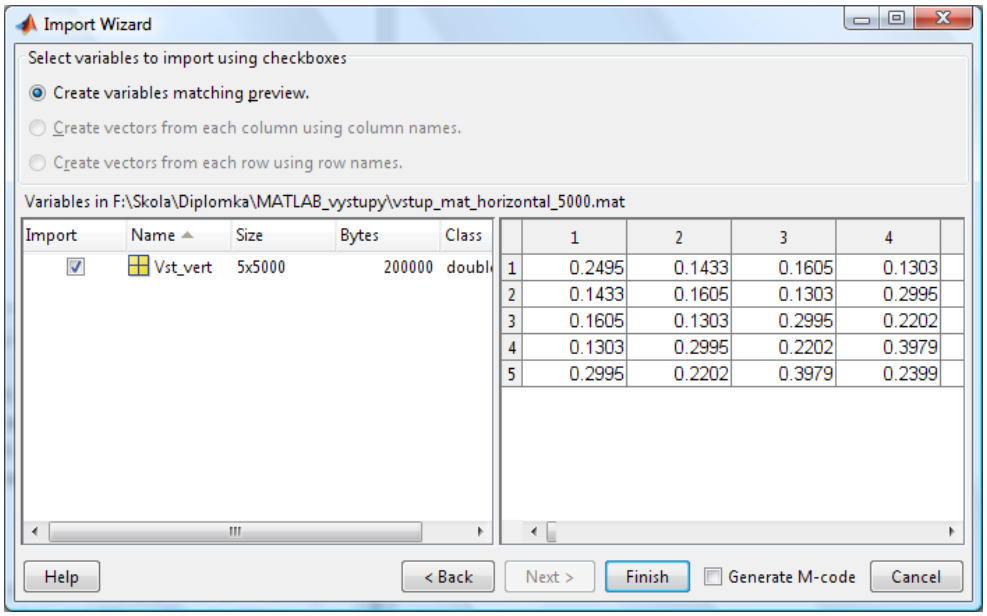
Obr. 8.2: Ukázka procesu agregace dat

Před samotným použitím agregovaných dat v oboru neuronových sítích, je velmi vhodné je znormovat. Protože nás zajímá hlavně průběh intenzit (vychází se z principu samo-podobnosti), než jejich skutečná hodnota. Navíc neuronové sítě lépe zvládají vybavovací fázi.

9 Zpracování dat – vytváření trénovacích vzorů

Data je třeba zpracovat do přijatelné podoby. Prozatím máme 30 000 vzorků zachycených dat. Jedná se o matici matice s rozměrem 2 x 30 000 (v každém vzorku je informace o času a o intenzitě agregovaných dat). Nás však zajímá pouze hodnota intenzity. Tento požadavek vstupní hodnoty zužuje na vektor o rozměru 1 x 30 000. Nyní je třeba uvědomit si, vzorky jsou vybírány oknem, které se posouvá vždy po jednom vzorku, ačkoli jeho velikost zůstává stejná (v našem případě 5 vzorků). Takže musíme vstupní vektor upravit do matice 5 x X. V tomto případě je možné vytvořit maximální rozměr 5 x 6000. Avšak je třeba tato celková data rozdělit na trénovací část a testovací část, abychom mohli síť trénovat ale také testovat. A ověřit tak princip generalizace, což je správná reakce sítě na neznámé vstupy.

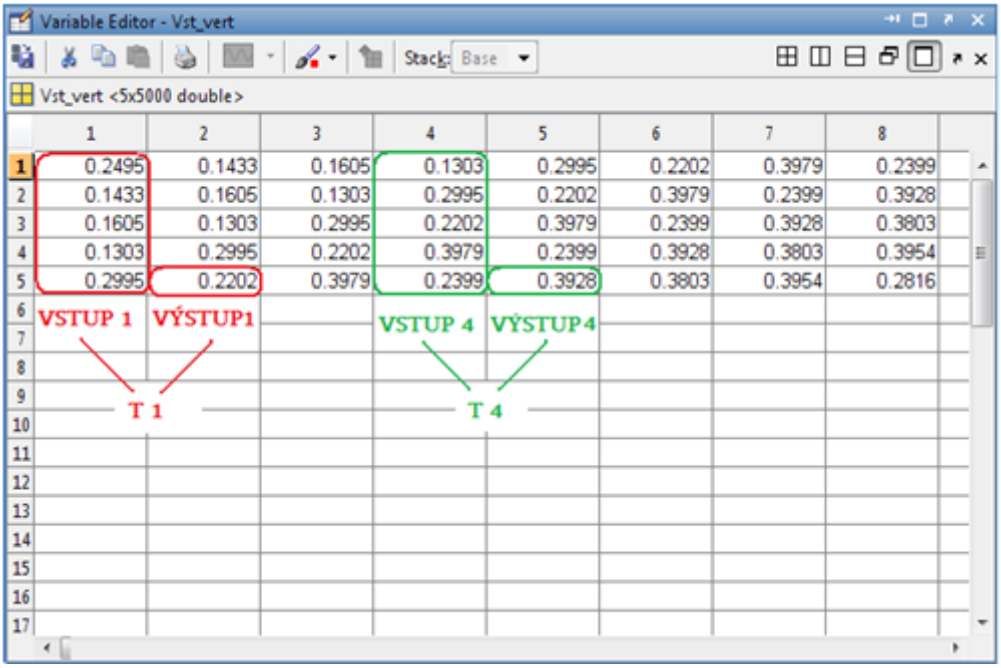
Nejprve tedy načteme vstupní trénovací data o rozměru 5 x 5000. Proměnnou jsme si pojmenovali třeba Vst_vert (Obr. 9.1).



Obr. 9.1: Importování trénovacích dat (vstupů)

Je třeba uvědomit si, že neuronovou síť budeme trénovat s učitelem – tzn. musíme přikládat ke konkrétním pěti vstupním hodnotám také jednu konkrétní výstupní hodnotu (predikovaná hodnota). Takže je třeba ještě vytvořit trénovací výstupní vektor. Posun okna na vstupu bude znamenat současnou inkrementaci výstupu. Pokud tedy zobrazíme vstupní matici, je možné v ní sledovat hodnoty výstupu, což je vždy nová hodnota okna v dalším

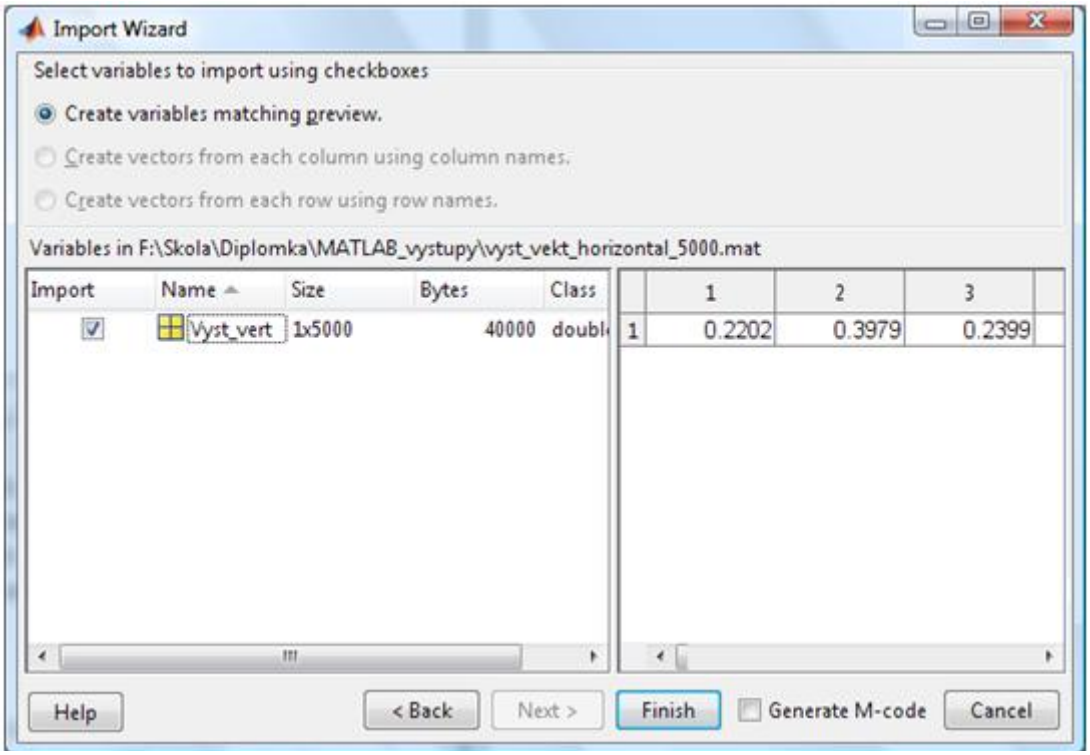
kroku. Společně vytvářejí trénovací vzory T1, T2, …Tn, kde n je počet trénovacích vzorů. Postup se opakuje. Pro názornost je situace zakreslena do obrázku…



	1	2	3	4	5	6	7	8
1	0.2495	0.1433	0.1605	0.1303	0.2995	0.2202	0.3979	0.2399
2	0.1433	0.1605	0.1303	0.2995	0.2202	0.3979	0.2399	0.3928
3	0.1605	0.1303	0.2995	0.2202	0.3979	0.2399	0.3928	0.3803
4	0.1303	0.2995	0.2202	0.3979	0.2399	0.3928	0.3803	0.3954
5	0.2995	0.2202	0.3979	0.2399	0.3928	0.3803	0.3954	0.2816

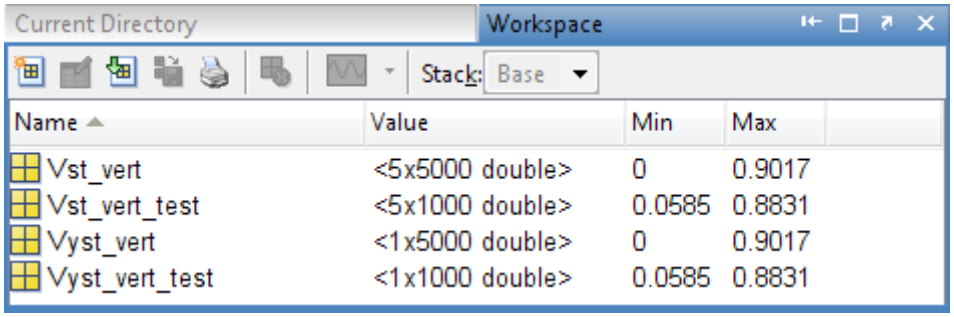
Obr. 9.2: Zobrazení trénovacích vzorů

Jak jsme již zmínili, je vhodné rozdělit si data na trénovací a testovací část. Jelikož jsme zvolili rozměr trénovací části 5 x 5000, na testovací část využijeme matici 5 x 1000. Je samozřejmě nutné aby trénovací a testovací data měla stejnou podobu. Matice vstupů 5 x X a vektor výstupu 1 x X, kde X je počet vzorů, musí mít stejný počet sloupců v rámci jednotlivých režimů (trénování/testování), protože společně tvoří jeden trénovací vzor.



Obr. 9.3: Import trénovacích dat (výstupů)

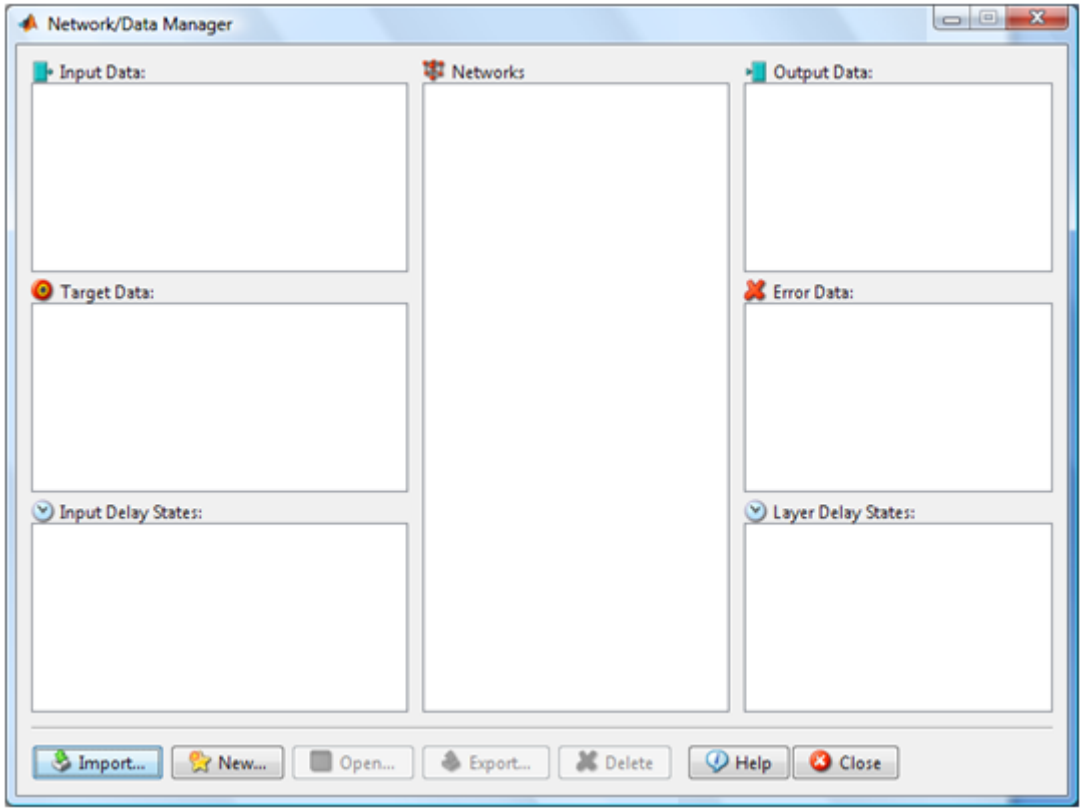
Nyní máme tedy připravena všechna data v požadovaném formátu a může započít trénování a testování jednotlivých síťí. Na obr. X, lez vidět jejich výčet ve Workspace Matlabu. Přehledně jsou zobrazeny informace o rozměrech (value) a limitech intenzit daných proměnných (Min, Max).



Obr. 9.4: Přehled všech importovaných dat

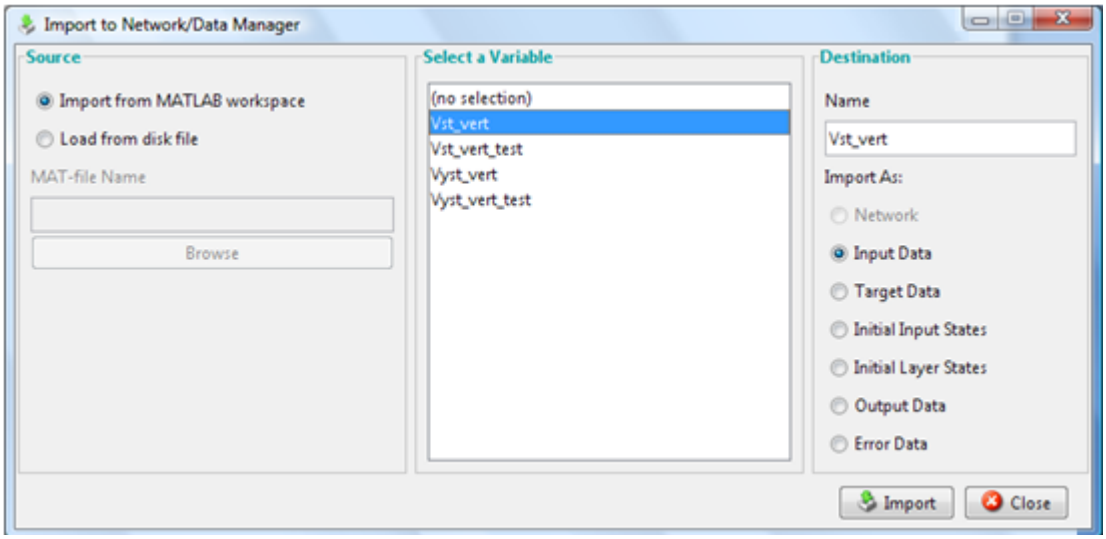
10 Vytvoření nové sítě

Při práci se sítěmi je vhodné využívat toolbox neuronových sítí (Neural Networks Toolbox). Je implementován přímo v matlabu a volá se příkazem „nntool“ z hlavního okna. Nejprve musíme určit data (Trénovací vzory) a způsob jakým s nimi budeme pracovat (druh neuronové sítě). Začneme tedy klepnutím na tlačítko import.



Obr. 10.1: Grafická podoba Network/Data Manažeru

Otevře se nám nové okno kde musíme definovat nejprve vstupní (inputs) a pak i výstupní (targets) data. To lze provést buď importem z workspace matlabu, nebo novým načtením ze souborového systému počítače. V pravé straně aportovacího okna je třeba definovat o jaká data se jedná, zda vstupní či výstupní. A samozřejmě klepneme na tlačítko import po každé operaci. Opticky se nic nestane, ale správnost ověříme tak, že se přepneme do základního okna nntoolu, jež máme na pozadí. Tam můžeme zkontrolovat zda byla data přidána.



Obr. 10.2: Import jednotlivých dat do Neural network toolboxu

Tímto jsme stanovili trénovací vzory. Nyní je třeba ještě vytvořit samotnou neuronovou síť. V základním okně nntoolu tedy klepneme na tlačítko „New“ a objeví se okno, kde můžeme definovat nejrůznější parametry naší sítě.

Nejprve samozřejmě zvolíme název naší sítě (Nepovinné, defaultně nastaveno na Network1)

Network Type: výběr druhu neuronové sítě

Input data: vybereme vstupy, které jsme si dříve načetli do paměti toolboxu

Target data: vybereme stejným způsobem i výstupy

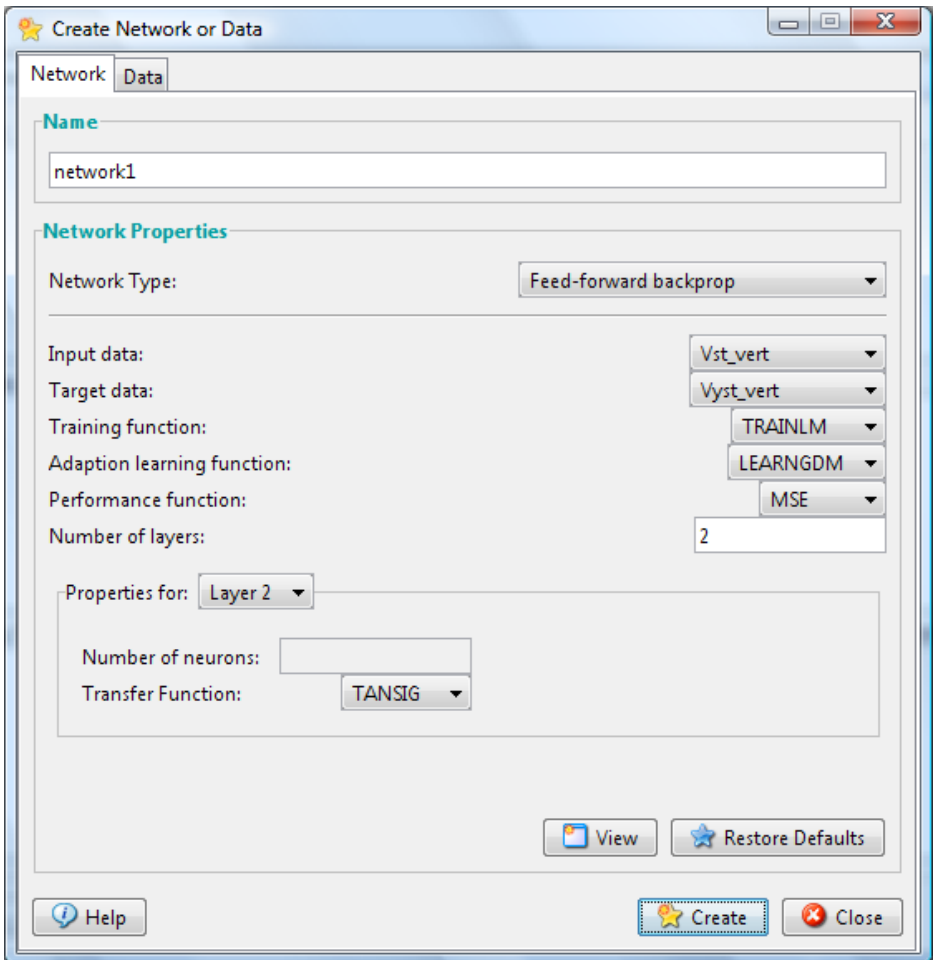
Number of layers: volíme počet vrstev sítě (nezbytné stisknout Enter bezprostředně po zadání číselné hodnoty, jinak se tato hodnota neuloží)

Properties for Layer: slouží pro konkrétnější nastavení jednotlivých vrstev.

Number of neurons: volba počtu neuronů v dané vrstvě (opět nutnost stisku Enteru)

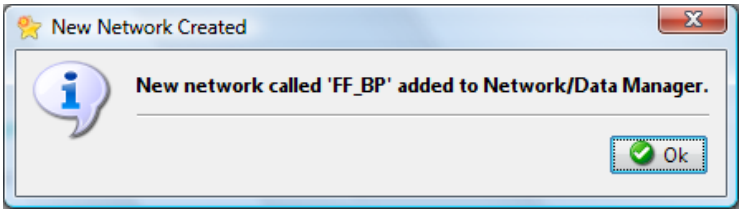
Transfer Function: Volba přenosové (aktivační) funkce

Ostatní parametry prozatím necháme nastaveny tak jak jsou, za zmínku stojí ještě MSE, což je střední chyba naší sítě. Může být jedním z kritérií pro ukončení trénování. Na záložce Data už není třeba nic nastavovat.

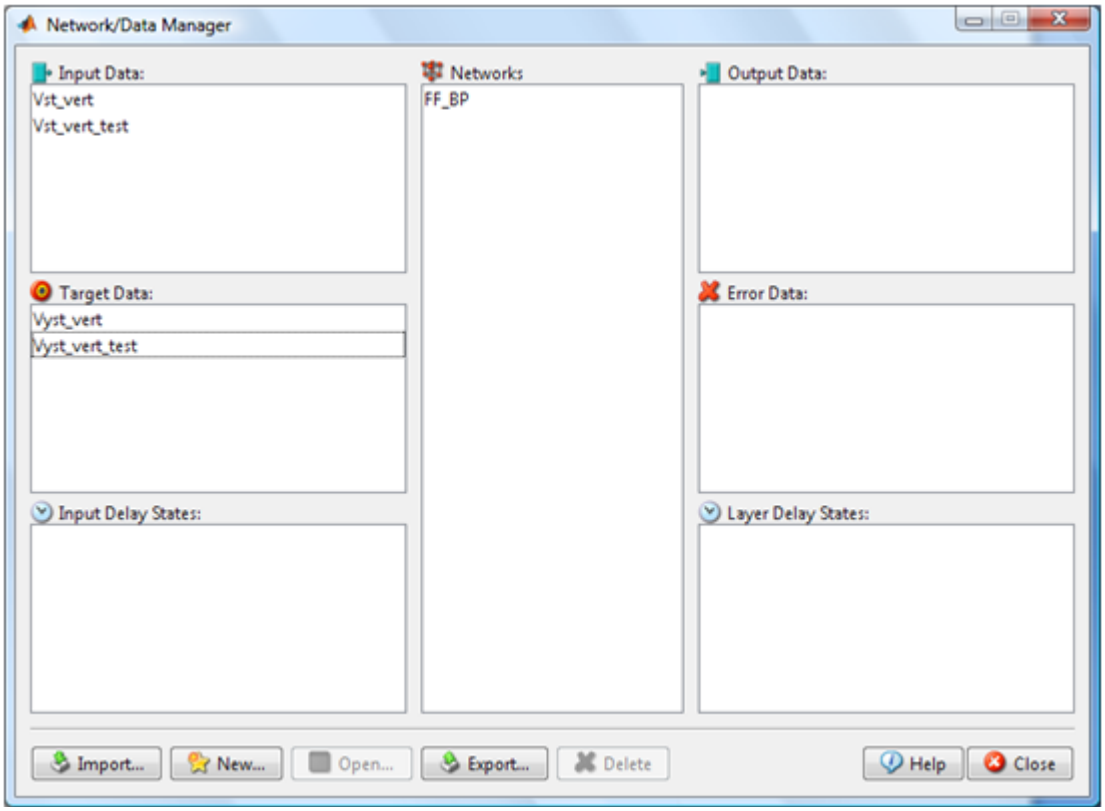


Obr. 10.3: Specifikace nové sítě

Pokud máme vyplněno podle svých představ, můžeme klepnout na tlačítko „Create“, čímž síť s danými parametry vytvoříme. O této akci nás informuje dialogové okno.



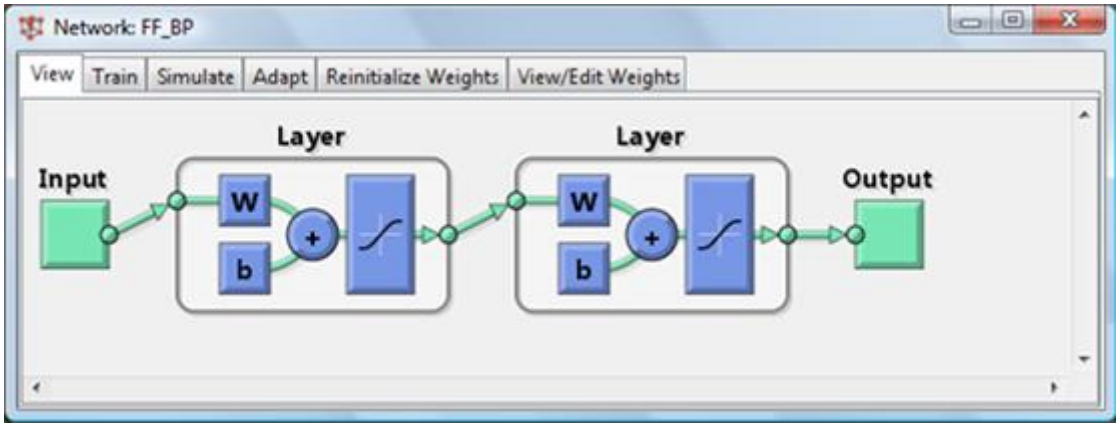
Obr. 10.4: Informace o vytvoření nové sítě



Obr. 10.5: Vhodně nastavený Network/data manažer

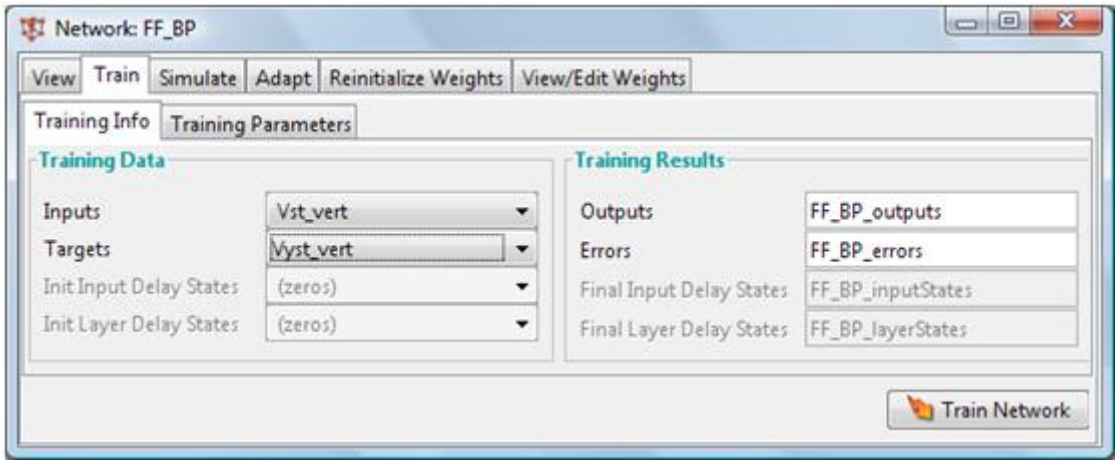
10.1 Trénování sítě

Nyní je vše připraveno započetí trénování sítě. Dvojklikem na název neuronové sítě (FF_BP) spustíme trénovací režim. Skládá se z několika záložek, ale pro nás budou důležité v podstatě jen první čtyři.



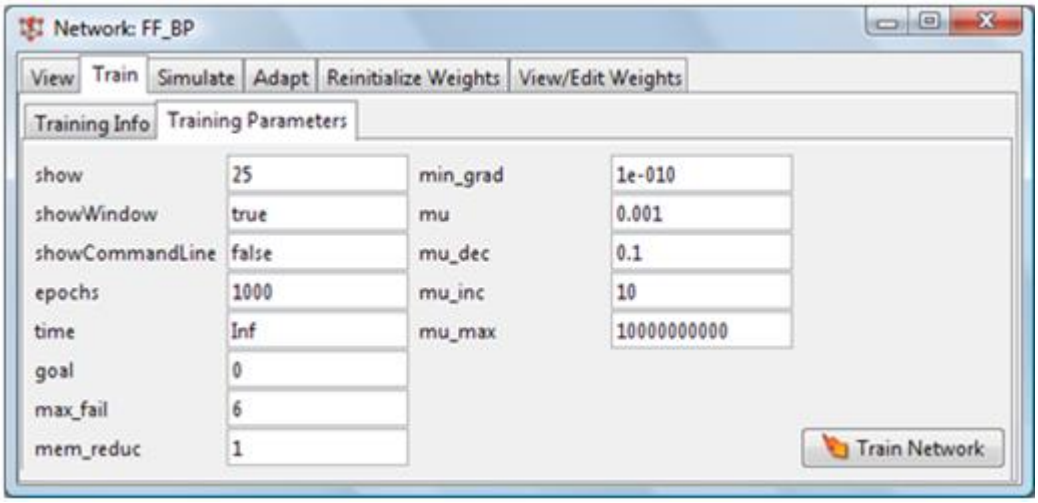
Obr. 10.6: Zobrazení topologie modelu trénované sítě

Při přepnutí na záložku Train (trénování) a podzáložku Training info opět musíme nastavit vstupní a výstupní data. Výsledky jsou uloženy do proměnných FF_BP_outputs (výstupy) a FF_BP_errors (chyby = rozdíl mezi skutečnými a vypočtenými hodnotami).



Obr. 10.7: Nastavení trénovacích vzorů

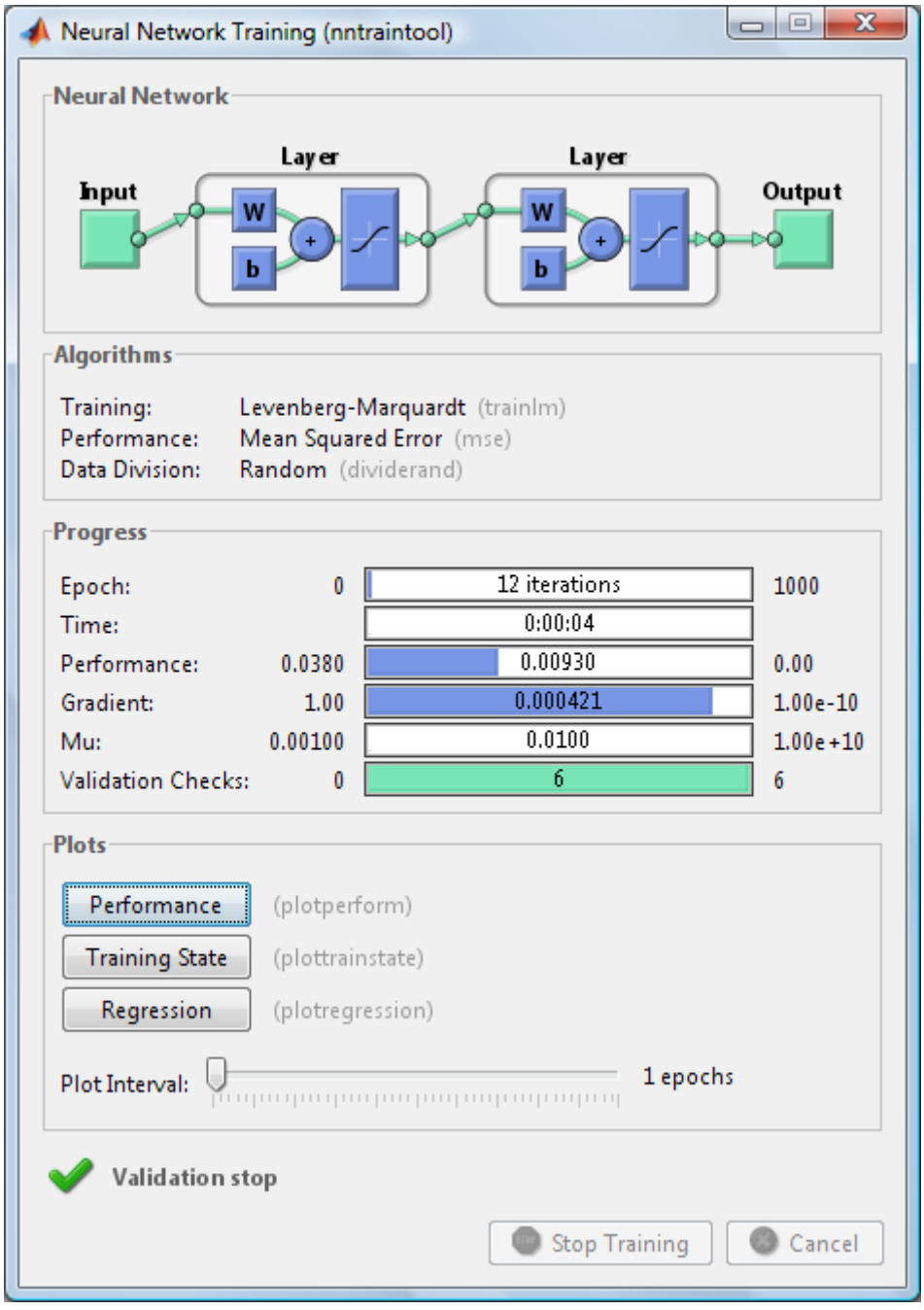
V dalším kroku jde o nastavení parametrů trénování. Nás bude nejvíce zajímat hodnota o počtu epoch, i když trénování je podmíněno nejmenší průměrnou chybou. To znamená, že je ukončeno, klesne-li chyba stanovenou hodnotu (mu). Takže se fakticky ani nemusíme dostat (a zpravidla ani nedostaneme) do stavu trénování 1000 epoch. Vše je tedy připraveno a můžeme klepnout na tlačítko „Train Network“, čímž započne proces trénování.



Obr. 10.8: Přehled nastavení trénovacích parametrů

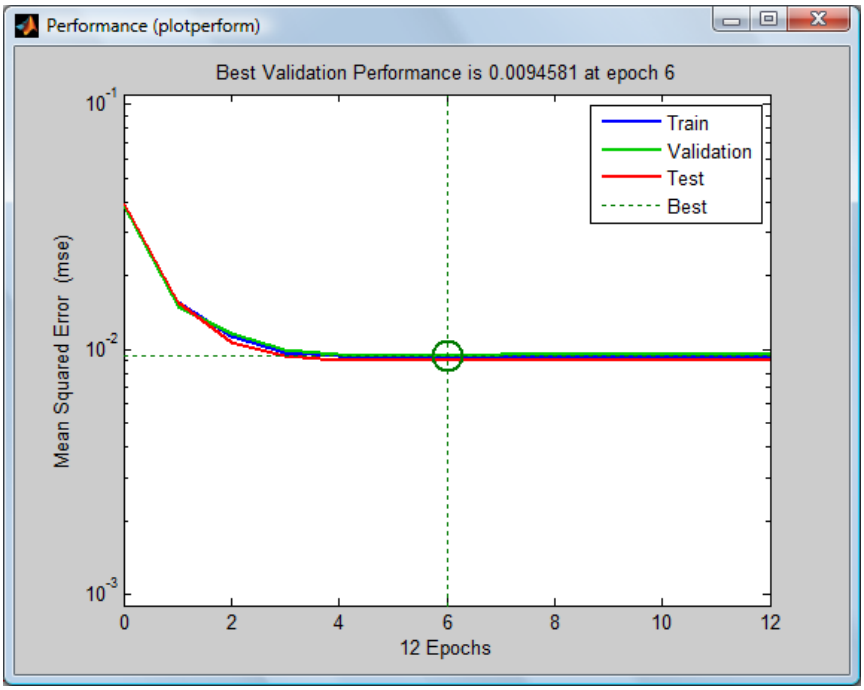
Celý proces trénování můžeme sledovat v přehledném grafickém rozhraní, jaké nntool v Matlabu 7.4 (release - září 2008) nabízí.

Pozn.: Grafická podoba trénování je luxusem poslední éry. Např. Matlab 7.1 vůbec nedisponuje takovýmto rozhranním. A uživatel nemá žádnou informaci o procesu trénování. Neví tedy vůbec, v jaké fázi se momentálně jeho síť nachází a zda nedošlo náhodou k zatuhnutí systému (asi 10% případů).



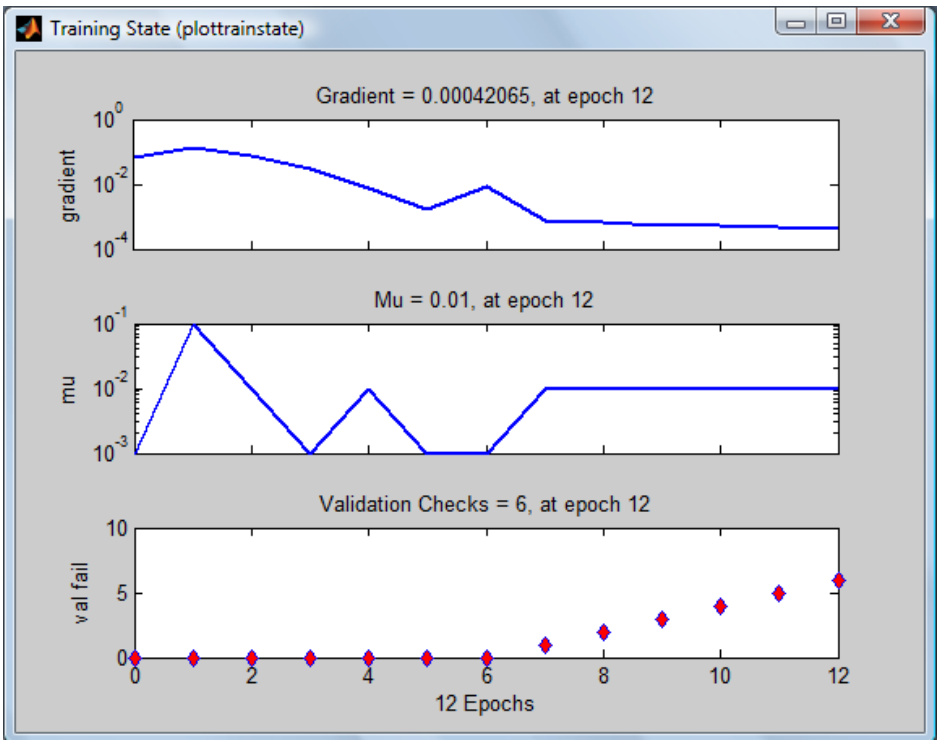
Obr. 10.9: Grafická podoba trénovacího procesu

Z výsledného okna jsme schopni vyčíst, že tato dopředná neuronová síť byla natrénována za čas 4 sekundy s průchodem 13 iterací (epoch). Performance nám udává chybu trénování Hodnota vlevo odpovídá počáteční chybě, a je samozřejmě vyšší než hodnota uprostřed, která je snížená, vlivem trénování. Tyto údaje jsou pro nás rozhodující. Chyba po trénování je 0,94%, což je slušný výsledek. Optimální by však byl řádově 10^{-4}. V praxi velmi těžko dosažitelný (v oboru neuronových sítí). Z grafické reprezentace výsledku se ještě dozvídáme, že zmíněnou chybu jsme dosáhli po 7 epochách, dále zůstávala stejná. Odchylku v odpovědích na validační řadu používáme jako kritérium pro ukončení trénování sítě. [16]



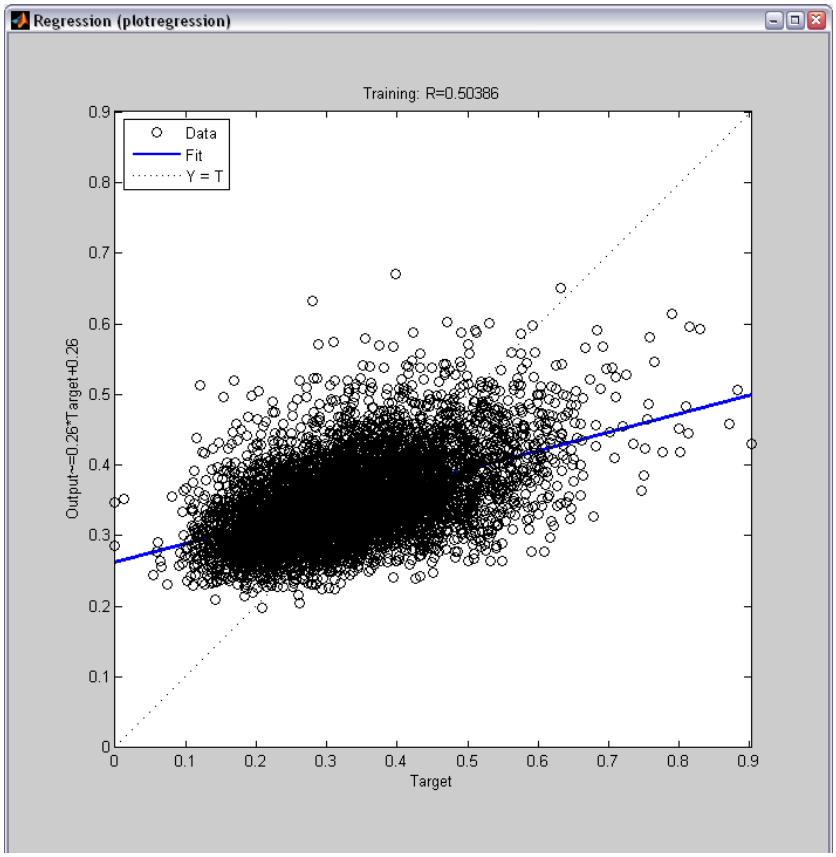
Obr. 10.10: Závislost trénovací chyby na počtu epoch

Navíc si můžeme prohlédnout jak se měnila závislost střední chyby (MSE – Mean Squared Error) na počtu epoch, spolu s doplňujícími informacemi o Gradientu (směrnice k výsledku) nebo Mu (střední chyba), či Validation Checks (validační řada).



Obr. 10.11: Doplňující informace o trénování sítě

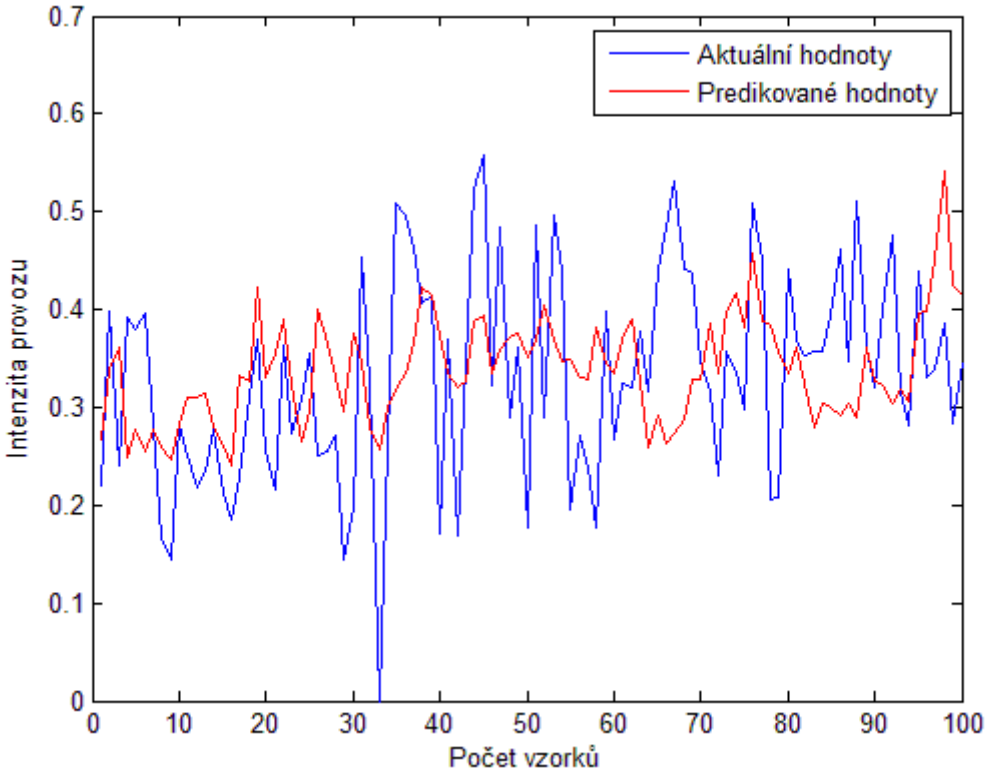
Zajímovou interpretací výsledku může být též zobrazení regresí (Obr. 10.12). V podstatě jde o jednoduchou metodu zakreslování pozic výstupních hodnot z neuronové sítě do grafu v němž je přímkou naznačena ideální stopa. Jedná se o přímkovou regresi, která vyjadřuje metodou minimalizace součtu čtverců rozdíl mezi vypočtenými hodnotami (predikovanými) a skutečnými [17]. Ideálně by tedy výpočty neuronové sítě měly ležet na této přímce, nebo se jí ale alespoň co nejvíce blížit. Tato metoda je však poměrně nepřesná a lze jí brát za orientační. Navíc u rekurentních neuronových sítí zobrazení regresí ani není možné.



Obr. 10.12: Rozložení regresní funkce vybavování dopředné sítě

10.2 Predikce známých hodnot (ověření tréninku)

Pokud bychom chtěli ověřit úroveň s jakou jsme natrénovali neuronovou síť, je možné predikovat hodnoty obsažené již v trénovacích vzorech. Nejedná se tedy o predikci v pravém slova smyslu, protože použité hodnoty by měla neuronová síť již znát (nejsou pro ni neznámé). Jedná se o subjektivní posudek. Avšak v tomto případě, lze pozorovat nepříliš dobrou predikci (Obr. 10.13), což může být z části dáno absencí zpětné vazby dopředné neuronové sítě.

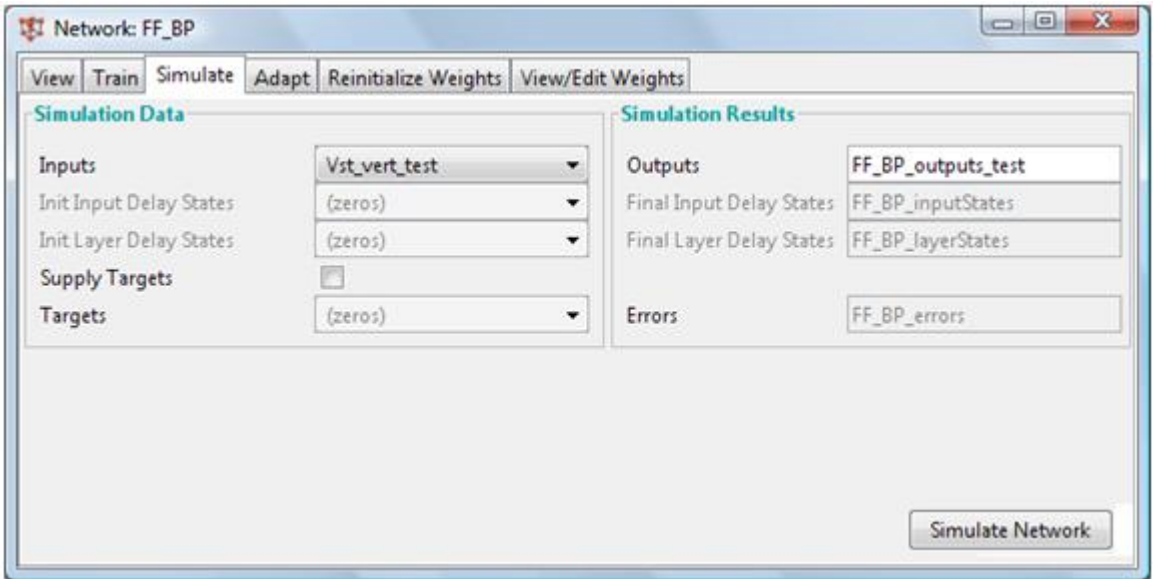


Obr. 10.13: Ověření trénování dopředné sítě

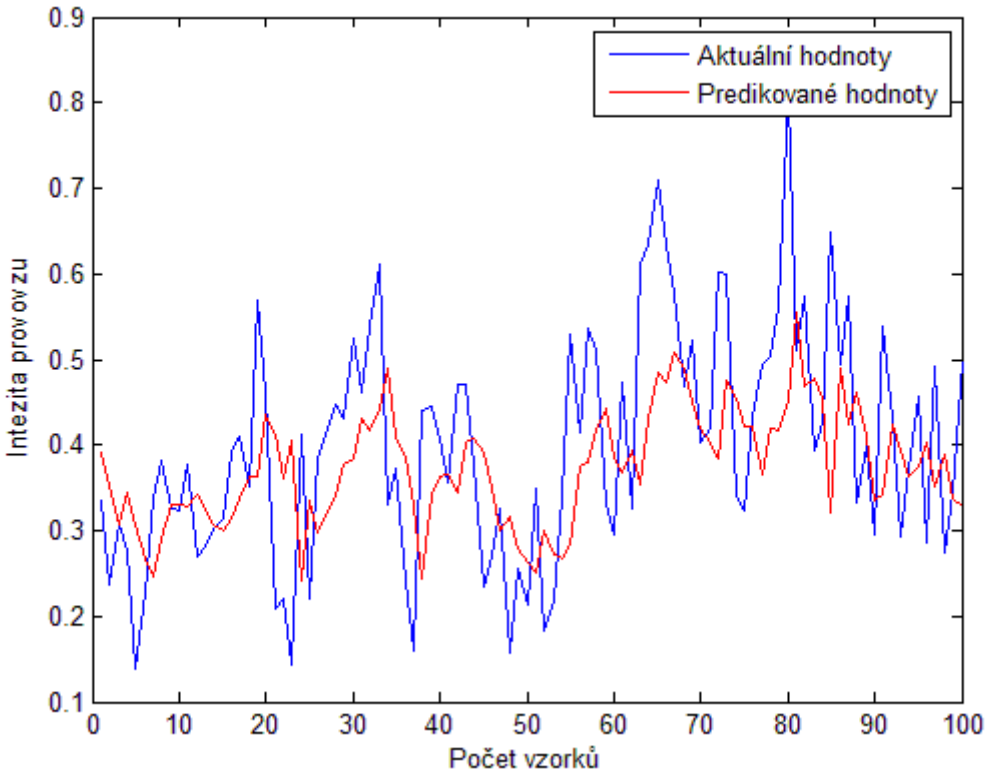
10.3 Predikce neznámých hodnot (princip generalizace)

Zatímco předchozí část (kap. 10.2) byla nepovinná a sloužila v podstatě jen pro naši ucelenější představu o způsobu fungovaní systému sítě, nyní se budeme zabývat skutečnou predikcí neznámých hodnot, nazývanou též princip generalizace. Tento princip je jednou ze základních vlastností neuronových sítí. Na základě natrénování jsou totiž schopny více, či méně predikovat nelineární časové řady.

Postupujeme tedy analogicky jako v přecházejícím případě, ale s tím rozdílem, že na vstup sítě přiložíme neznámá data (vst_vert_test) a simulujeme naučenou sítí hodnoty s časovou inkrementací (Obr. 10.14). Vidíme, že výsledky jsou uloženy do proměnné FF_outputs_test. Zpoždění vrstev (zpětná vazba) se u tohoto druhu sítě neuplatňuje.



Obr. 10.14: Předložení neznámých vstupních hodnot



Obr. 10.15: Predikce neznámých hodnot provozu

Pokud se nám výsledek predikce zamlouvá, můžeme síť adaptovat. Přepneme se tedy v nntoolu na záložku Adapt a určíme, podle jakých vstupů se má síť adaptovat. Pak je možné ji uložit se stávající konfigurací – nastavenými hodnotami synaptických vah.

11 Hledání vhodného modelu sítě

11.1 Síť NARX

Prozatím jsme si ukázali princip predikce pomocí dopředné neuronové sítě, u níž se veškerá dynamika projevuje pouze ve vstupní vrstvě. Nyní se podíváme na síť NARX, což nelineární autoregresní síť s exogenními (vnějšími) vstupy. Jedná se o dynamickou rekurentní síť, se zpětnou vazbou na jednu či několik vrstev. NARX model je založen na lineárním ARX modelu, který je běžně používán k modelování časových řad.

Charakteristická rovnice testovacího modelu:

$$y(t) = f(y(t-1), y(t-2), \dots, y(t-ny), u(t-1), u(t-2), \dots, u(t-nu)), \quad (11)$$

kde další hodnota výstupního signálu *y(t)* je závislá na předchozích hodnotách nezávislého vstupního signálu, ale také na předchozích hodnotách výstupního signálu u(t) (zpětná vazba).

Stručný výpis výsledků testování:

Topologie: Dvouvrstvá rekurentní síť 10-1

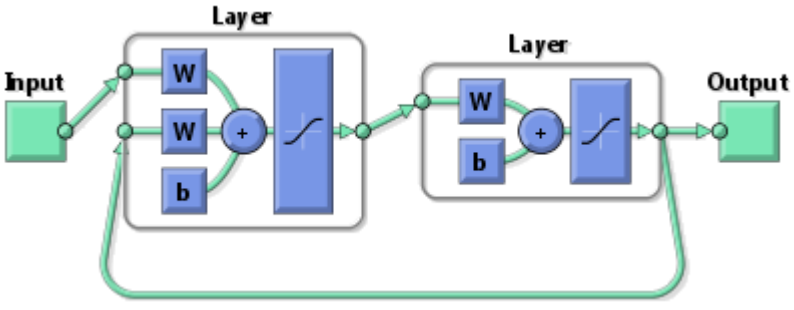
Aktivační funkce: Tansig

Doba trénování: 5 minut

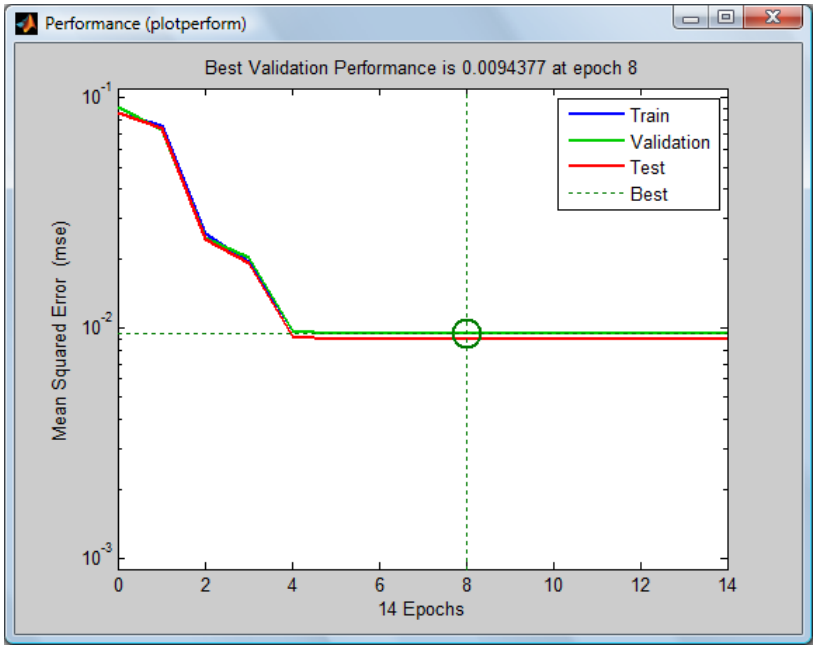
Počet iterací: 14

Nejlepší dosažená střední chyba: 0,95%

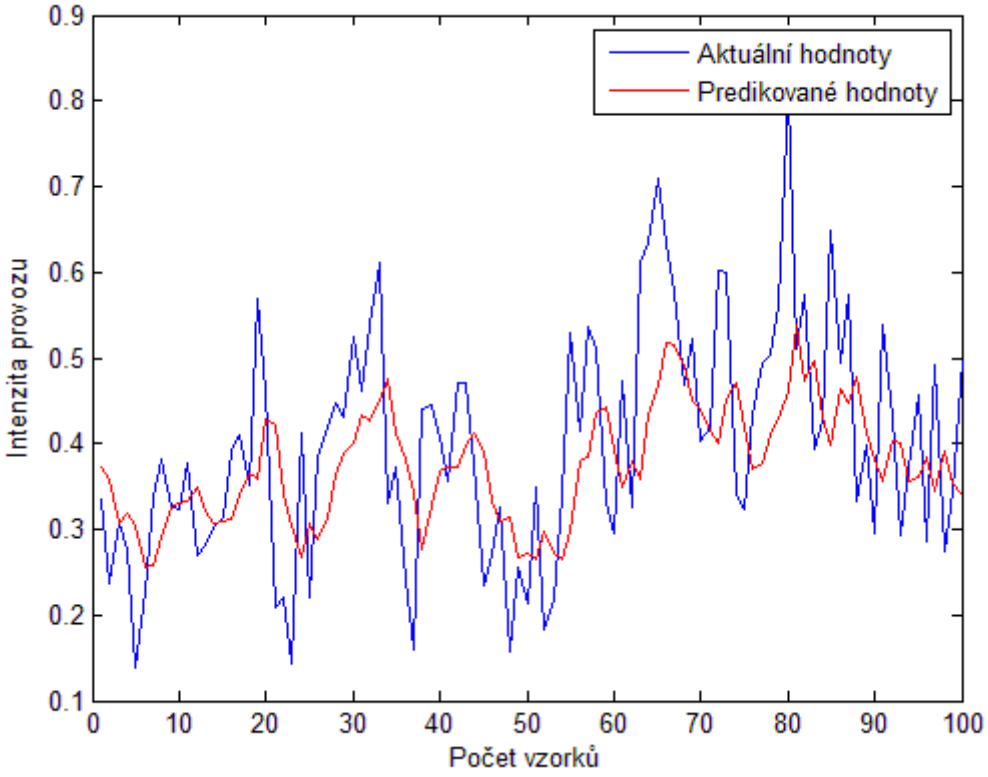
Schéma sítě:



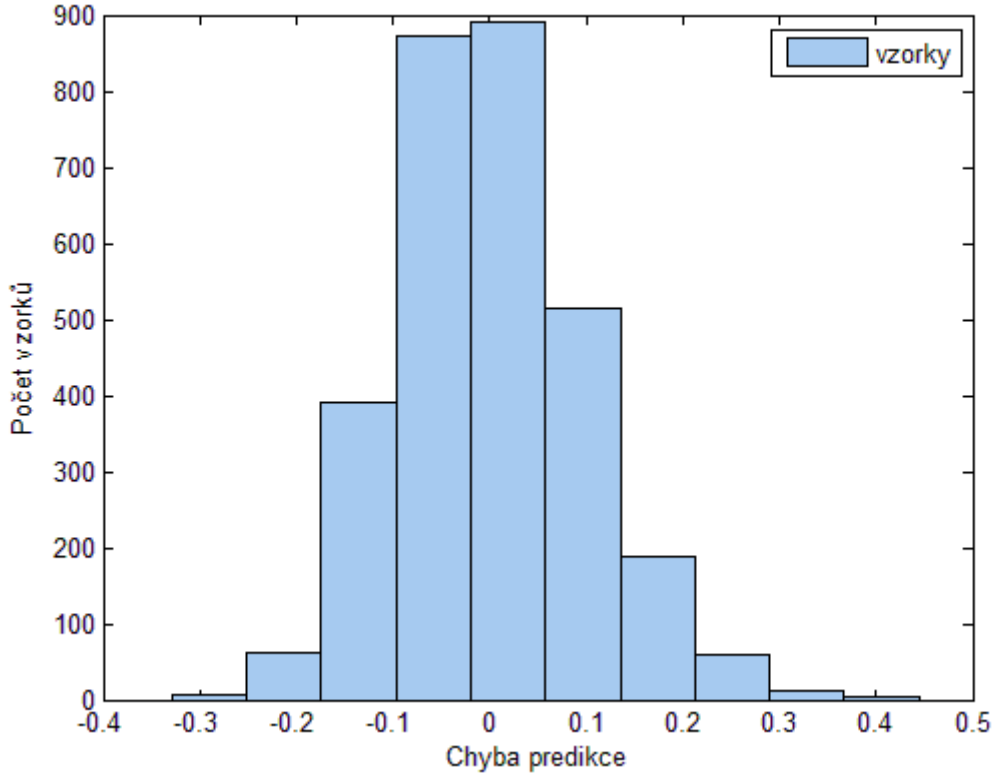
Obr. 11.1: Schéma sítě NARX



Obr. 11.2: Závislost střední chyby na počtu epoch sítě NARX



Obr. 11.3: Predikce síťového provozu sítě NARX



Obr. 11.4: Histogram výskytu chyb predikce pomocí sítě NARX

Z výsledků je patrné, že síť NARX se dokáže velmi rychle naučit trénovacím vzorům a předkládat relativně uspokojivé výsledky. Tuto síť bychom pravděpodobně použili v situaci, kde máme veliké množství vzorků a potřebujeme je přikládat v reálném čase na neuronovou síť. Rovněž průběh predikce v porovnání se skutečnými hodnotami můžeme považovat za úspěšný (vzhledem k tomu, že se jedná o základní model NARX sítě).

Výhody:

- Rychlost učení a vybavování
- Efektivita
- Jednoduchost

Nevýhody:

- Velmi těžká gradace řešení (modifikace sítě nezlepšuje výsledky)

11.2 Síť NARX-SP (Series-parallel)

Stejně jako u předchozího modelu sítě, i tady se na tvorbě výstupní hodnoty podílí vstupní nezávislé hodnoty, ale i výstupní dřívější hodnoty.

Jak je patrné ze schématu tohoto modelu sítě, zpětná vazba se neuplatňuje tak jako tomu bylo u modelu NARX. Namísto toho jsou předchozí hodnoty (vstupní i výstpuní) do výpočtu zahrnuty jiným způsobem. Obě jsou totiž přivedeny na vstup sítě, jako by se jednalo o dva vstupy. V tom je tato varianta modelu unikátní.

Stručný výpis výsledků testování:

Topologie: Dvouvrstvá rekurentní síť 10-1

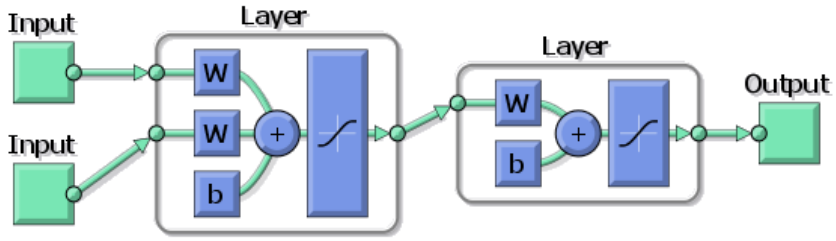
Aktivační funkce: Tansig

Doba trénování: 10 minut

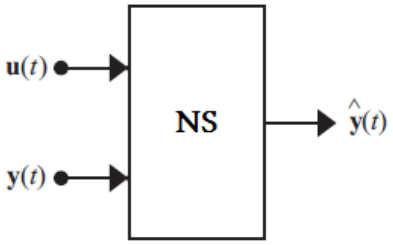
Počet iterací: 7

Nejlepší dosažená střední chyba: 0,92%

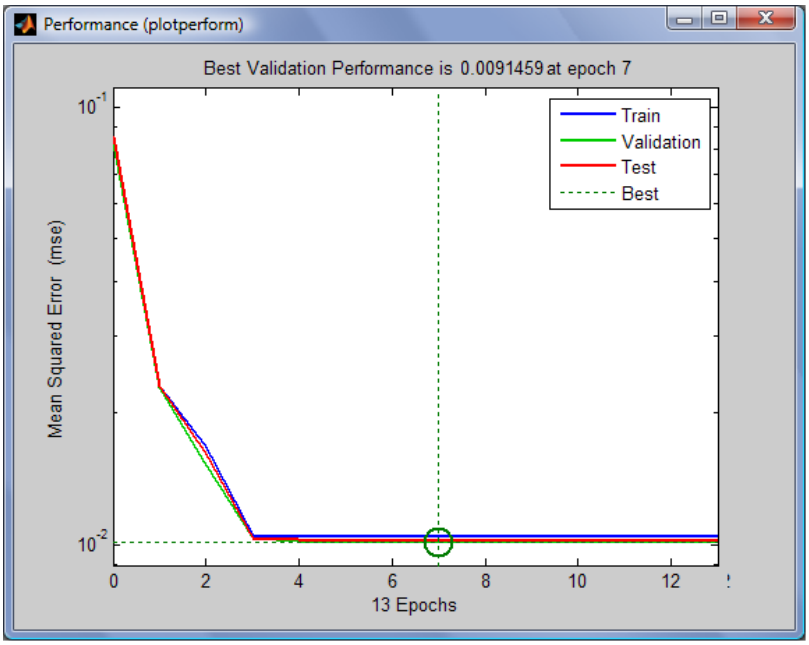
Schéma sítě:



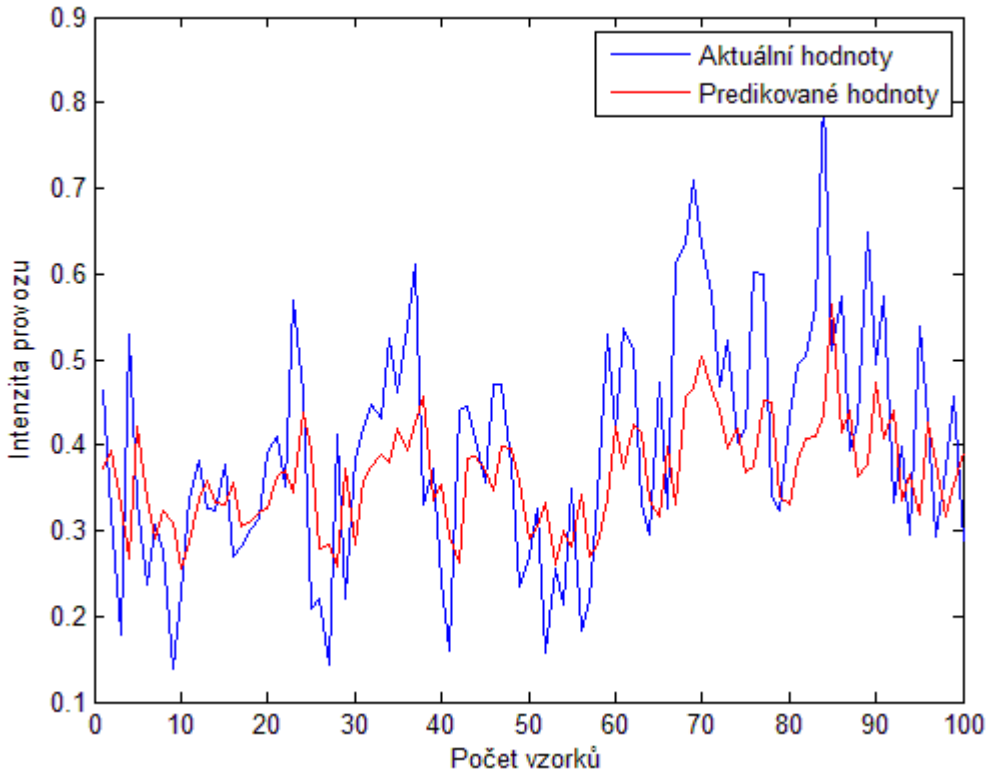
Obr. 11.5: Schéma sítě NARX-SP



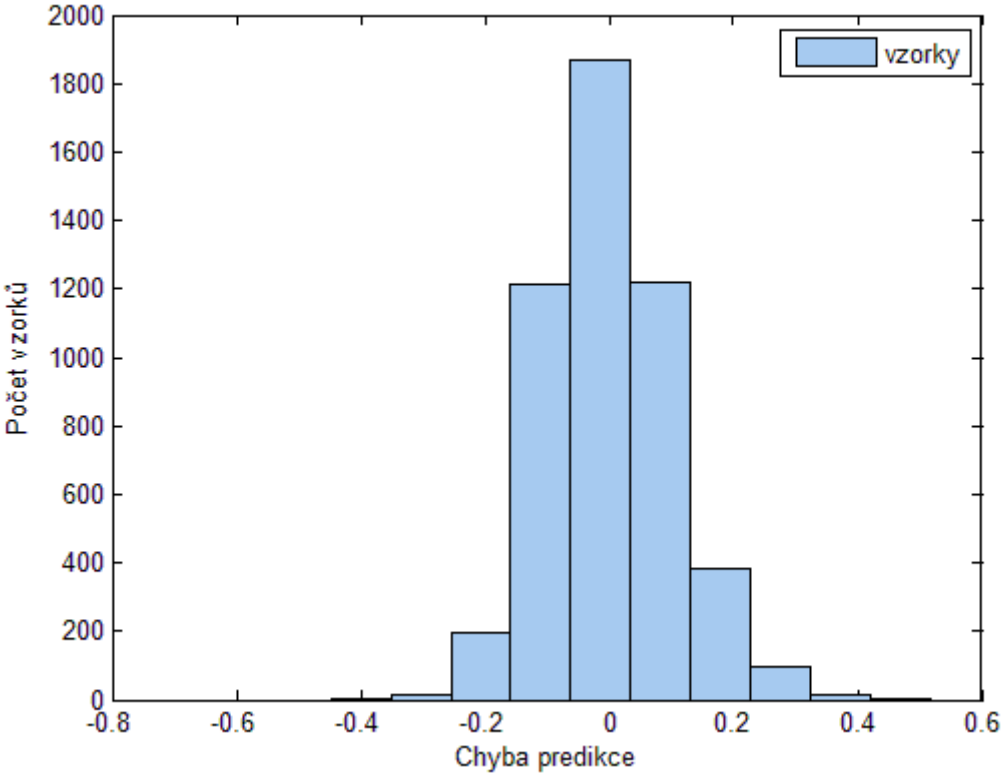
Obr. 11.6: Zjednodušené schéma sítě NARX-SP



Obr. 11.7: Závislost střední chyby na počtu epoch sítě NARX-SP



Obr. 11.8: Predikce síťového provozu sítě NARX



Obr. 11.9: Histogram výskytu chyb predikce pomocí sítě NARX-SP

Tento model sítě sice není tak rychlý jako předchozí, za to je však přesnější. Vykazuje menší střední chybu, což lze pozorovat také z průběhů predikovaných hodnot (Obr. 11.8). Predikce je věrnější, síť lépe reaguje. Také v histogramu chyb lze rozlišit určitý menší rozptyl, cože je jenom dobře.

Výhody:

- Věrnější predikce
- Menší chyba

Nevýhody:

- Pomalejší
- Velmi těžká gradace řešení

11.3 Elmanova síť

Tato síť patří do skupiny Layer –Recurrent network (LRN), obecné rekurentní neuronové sítě. LRN využívá opět zpětné vazby s jednotným zpožděním, připadajícím na každou vrstvu, kromě poslední. Může jít tedy o např. třívrstvou síť. Naproti tomu, Elmanova síť je pouze dvouvrstvá s rozdílnými aktivačními funkcemi v jednotlivých vrstvách. Využívá trénovací algoritmus Backpropagation.

Stručný výpis výsledků testování:

Topologie: Dvouvrstvá rekurentní síť 10-1

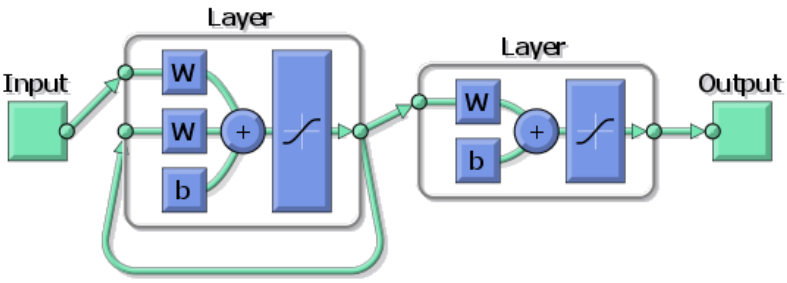
Aktivační funkce: Tansig ve skryté vrsvě, Purelin ve výstupní vrstvě

Doba trénování: 14 minut

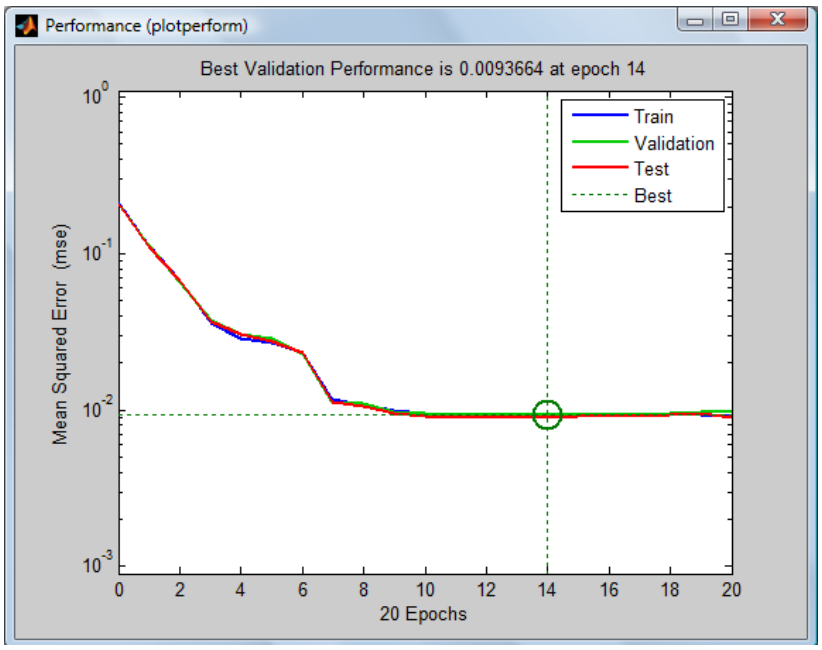
Počet iterací: 14

Nejlepší dosažená střední chyba: 0,94%

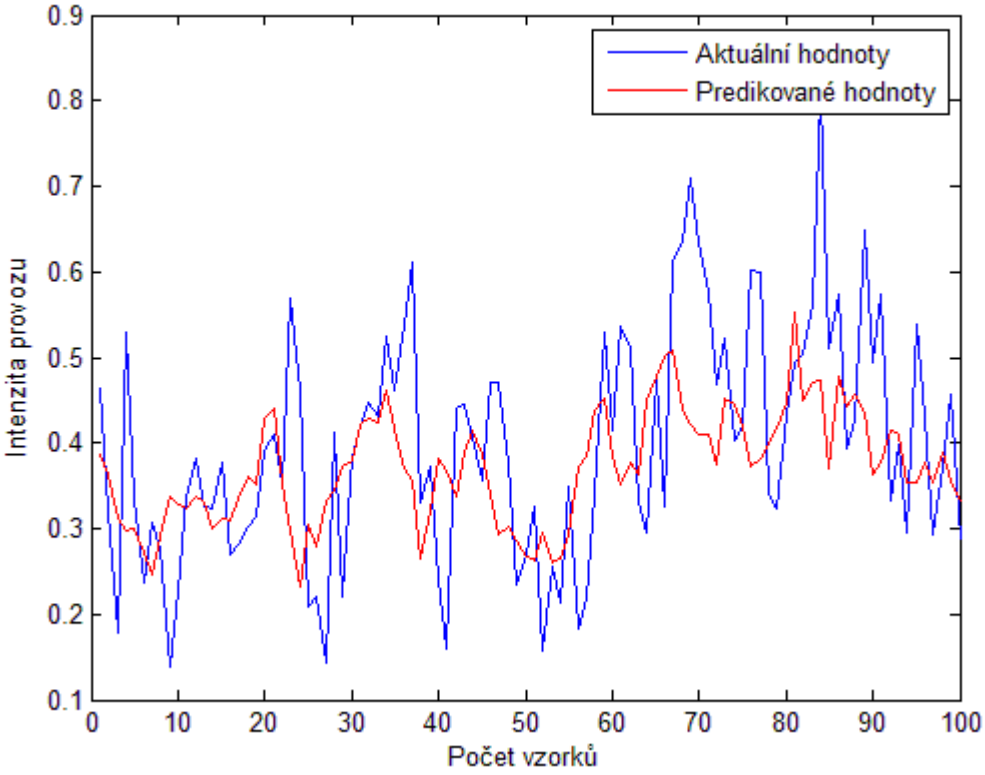
Schéma sítě:



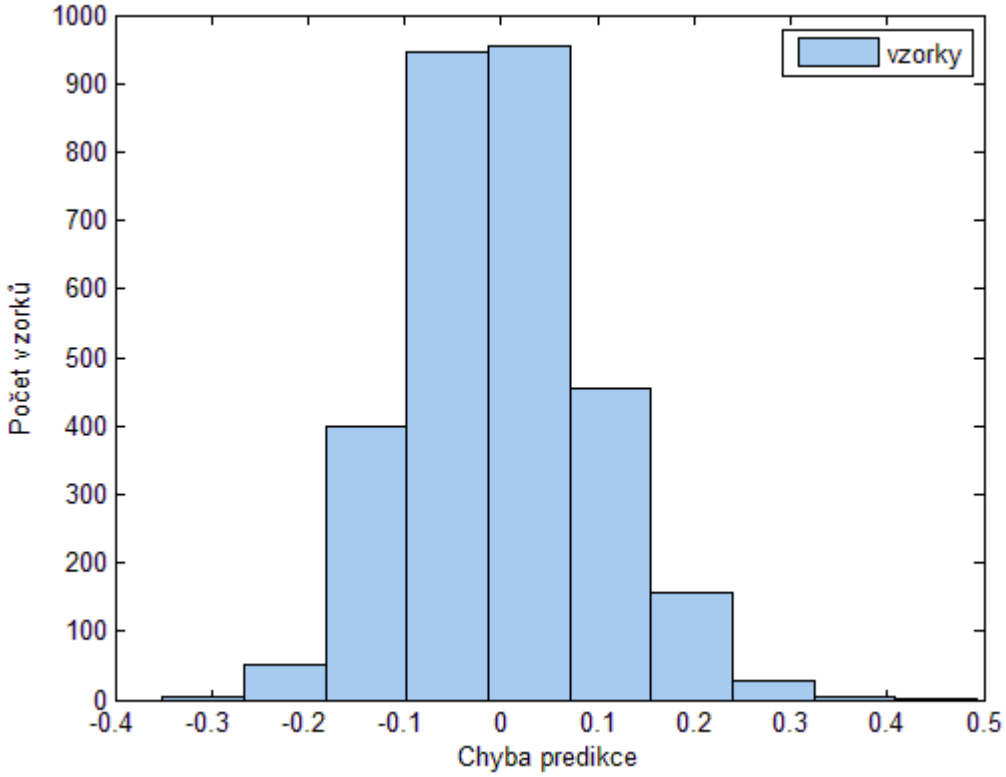
Obr. 11.10: Schéma Elmanovy sítě



Obr. 11.11: Závislost střední chyby na počtu epoch Elmanovy sítě



Obr. 11.12: Predikce síťového provozu pomocí Elmanovy sítě



Obr. 11.13: Histogram výskytu chyb predikce pomocí Elmanovy sítě

Elmanova síť nevykazuje žádné zázračné výsledky, ani v rychlosti ani ve velikosti chyby. Má podobné výsledky jako síť NARX, ale s jedním podstatným rozdílem. Tato síť lépe kopíruje nástupní hrany laloků průběhu intenzity provozu, kdežto model NARX lépe reaguje na sestupné. Je to patrné z grafického vyjádření.

11.4 Obecná rekurentní síť Layer – Recurrent network (LRN)

Jak již bylo řečeno v rozboru předchozího modelu, jedná se v podstatě o obecný zápis struktury populární Elmanovy sítě, která tvoří podmnožinu LRN. Zde máme poměrně velkou volnost při návrhu topologie sítě. Prozatím se však omezíme na dvouvrstvou variantu s odlišnými aktivačními funkcemi než u Elmanova modelu.

Stručný výpis výsledků testování:

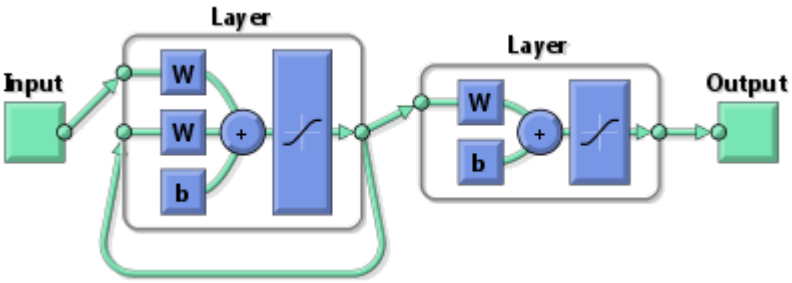
Topologie: Dvouvrstvá rekurentní síť 10-1

Aktivační funkce: Tansig

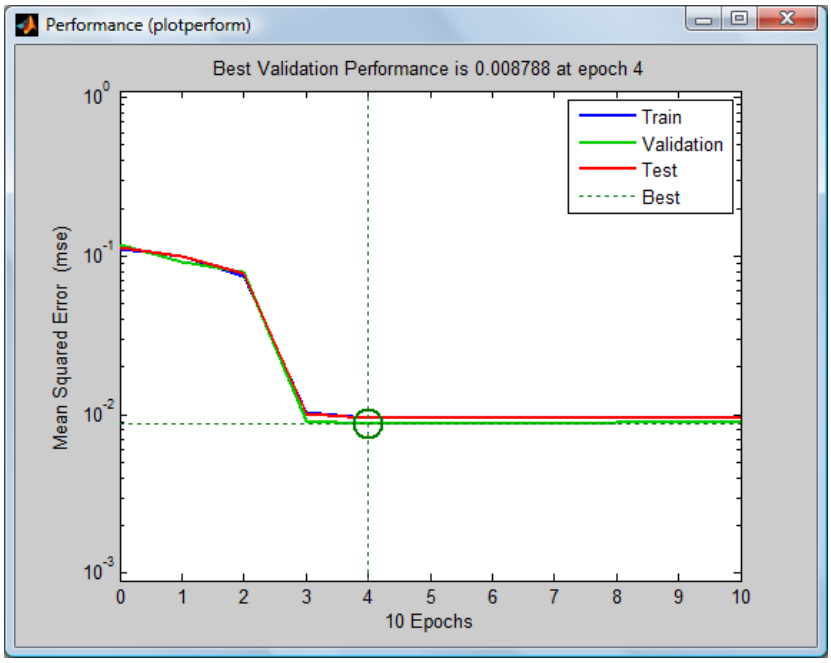
Doba trénování: 8 minut

Počet iterací: 4

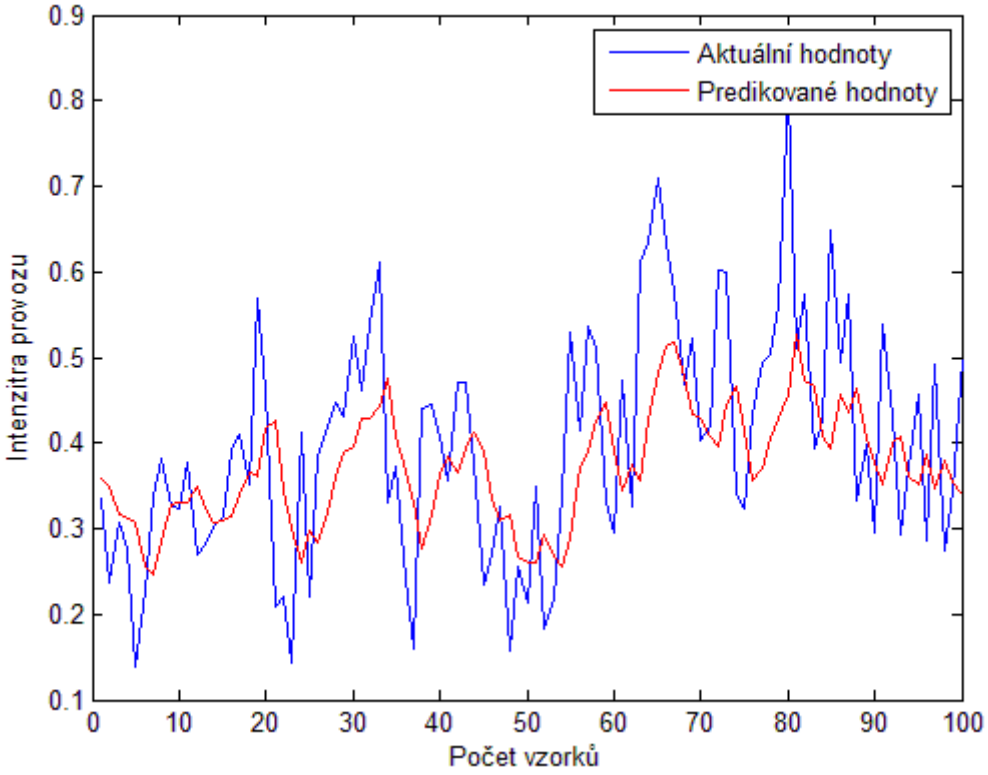
Nejlepší dosažená střední chyba: 0,88%



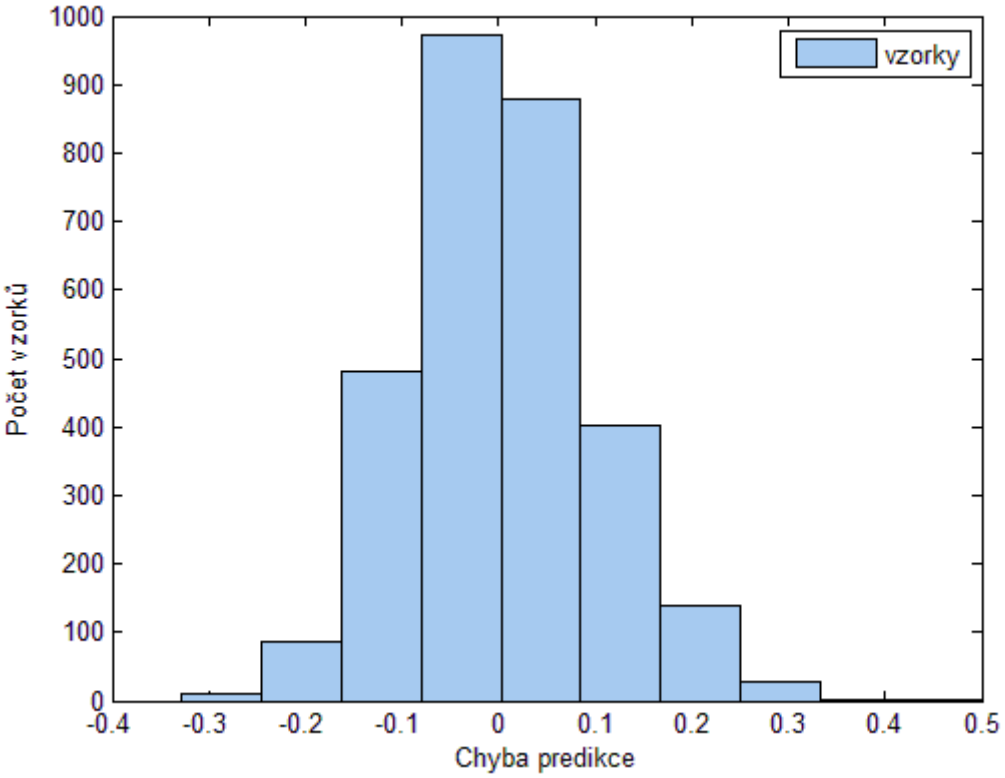
Obr. 11.14: Schéma obecné rekurentní sítě



Obr. 11.15: Závislost střední chyby na počtu epoch sítě LRN



Obr. 11.16: Predikce síťového provozu pomocí LRN sítě



Obr. 11.17: Histogram výskytu chyb predikce pomocí LRN sítě

S touto sítí jsme dosáhli nejlepšího výsledku. Chyba 0,88% je daleko pod hranicí všech ostatních modelů. Ze zkušenosti z testování lze obecně říci, že změna aktivačních funkcí předepsaného modelu (tady jde vlastně o modivikovaný Elmanův model) se nevyplácí. Dochází k nárůstu výpočetní náročnosti a ke zvětšení střední chyby. V tomto případě je však toto nepsané pravidlo popřeno. Změna aktivační funkcí dané situaci jednoznačně prospěla. V další kapitole se tedy budeme zabývat výhradně modifikací této sítě, za účelem gradace řešení naší vstupní úlohy.

Výhody:

- Velká variabilita a volnost při návrhu architektury
- Výborná hodnota střední chyby

Nevýhody:

- Výskyt větších chyb (viz histogram)

12 Hledání optimální varianty zvoleného modelu sítě

Velmi dobré výsledky předchozího modelu trochu kazí složení chyb predikce, což lze pozorovat v histogramu chyb (Obr. 11.17). Pokud bychom v hypotetické rovině uvažovali dvě sítě s totožnými hodnotami středních chyb predikce, bylo by nutné zaměřit se na to, která z nich má lepší rozložení chyb. Je totiž výhodnější, aby síť vykazovala více menších chyb, než méně velkých. V prvním případě se průběh predikce velmi významně blíží ke skutečnému provozu, ve druhém by bylo možno sledovat množství zákmitů a nepřesností. Právě v tomto ohledu se rozvíjí naše snažení. Výchozí sítí je LRN model popsaný výše.

12.1 Varianta LRN sítě 10-10-1

Stručný výpis výsledků testování:

Topologie: Třívrstvá rekurentní síť 10-10-1

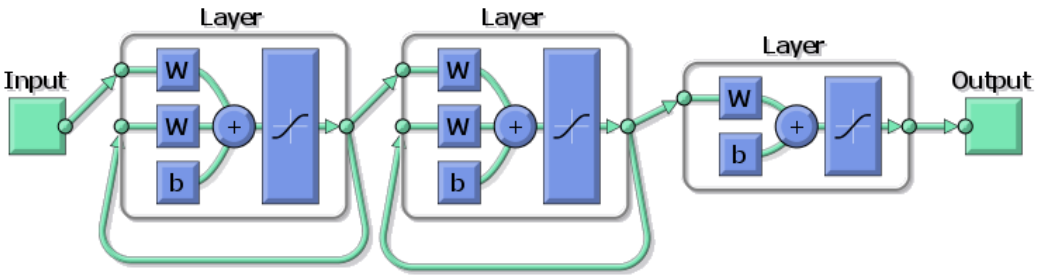
Aktivační funkce: Tansig

Doba trénování: 47 minut

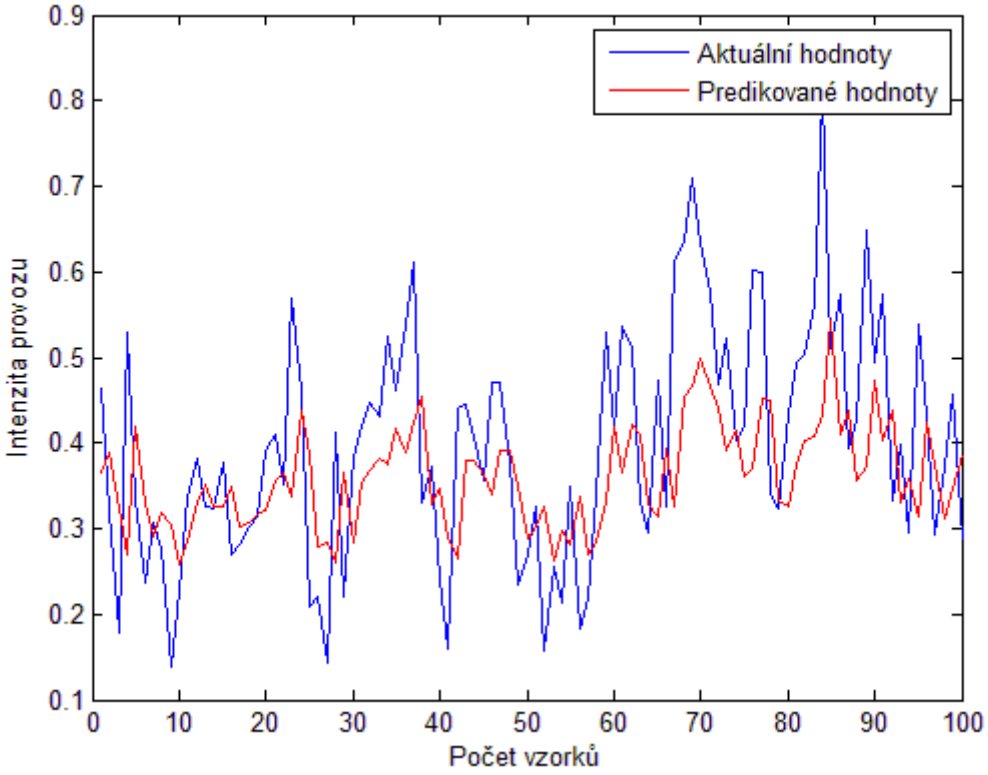
Počet iterací: 10

Nejlepší dosažená střední chyba: 0,92%

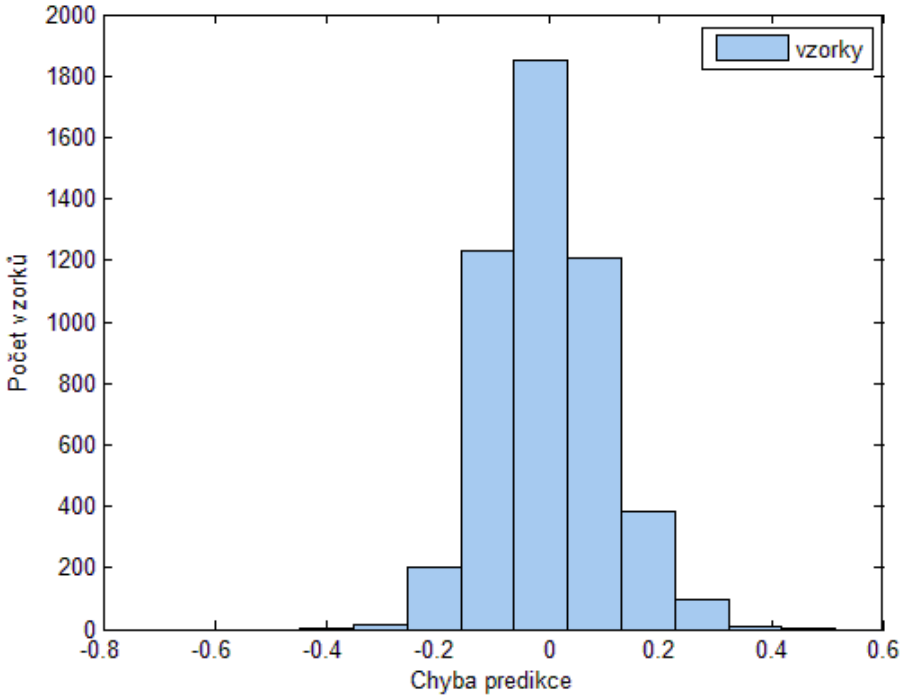
Schéma sítě:



Obr. 12.1: Schéma obecné rekurentní sítě LRN 10-10-1



Obr. 12.2: Predikce síťového provozu pomocí modifikované LRN sítě



Obr. 12.3: Histogram chyb predikce pomocí modifikované LRN sítě

Doba trénování se významně prodloužila, což je dáno složitější topologií sítě a zvýšeným počtem průchodů trénovací množiny síťí. Mírně se zvýšila i střední hodnota chyby. Co nás však zajímá nejvíce je, to proč jsme celou úpravu prováděli, a sice změna rozložení chyb predikce. Skutečně lze pozorovat znatelně lepší výsledky, projevující se mnohem věrnější predikcí (Obr. 12.2) a chudším rozložením chyb (Obr. 12.3). Jak bylo uvedeno již při předzpracování dat a tvorbě trénovacíh vzorů (kapitola 8.), jedná se o normované hodnoty. Tudíž nás skutečně zajímá nejvíce průběh predikce a od toho odvozené rozložení chyb. Můžeme prohlásit, že s obecnou rekurentní neuronovou sítí LRN typu 10-10-1, jsme schopni dosáhnout nejlepších výsledků při predikaci síťového provozu.

13 Shrnutí poznatků

Výhody:

- Učení se (není třeba znát přesný matematický popis vstupů – NS si zjistí algoritmus sama)
- Princip Generalizace (na základě předchozí vlastnosti je NS schopná správně reagovat i na neznámé vstupy)
- Cenová dostupnost a vývoj (potřebujeme „pouze“ Matlab – čím novější, tím lepší. Verze Mablabu 7.6 obsahuje NN toolbox 6, který má spoustu výhod a vylepšení)

Nevýhody:

- Při trénování veliké nároky na hostující počítač
- Doba trénování narůstá se složitostí sítě (výsledek predikce se nemusí zlepšovat)
- Dopředu není jasný výsledek (úspěch/neúspěch)
- Variabilita řešení různých problémů různými sítěmi

Nároky na PC:

Architektura:

Systém s jednojádrovým procesorem – trénování je pomalejší, vytížení počítače je maximální, zásah do jeho běhu má vliv na trénování (ostatní aplikace – icq, skype, stahování…)

Systém s dvoujádrovým procesorem – většina předchozích problémů odpadá, doba trénování se mírně zkracuje.

Operační systém:

Windows XP Professional – trénování stabilní

Windows Vista Ultimate – problematická práce s nntoolem, během práci na s proměnnými i se sítí často docházelo ke kolizím (ztrácení tlačítek, zatuhnutí) Nejlepší doporučená konfigurace: Dvoujádrový procesor + Windows XP Professional + pokud možno co nejnovější Matlab.

14 Závěr

Predikce je velmi důležitou a užitečnou metodou v různých oborech. Při tvoření této práce jsem se setkal s jejími mnoha různorodými využitími (ať již šlo snahu predikovat vývoj kurzu měny nebo o předpovědi vlhkosti vzduchu). Pro tento projekt se stala středem pozornosti možnost předpovědi síťového provozu pomocí neuronových sítí. Nutno podotknout, že se jedná o perspektivní problematiku, která má dobré vyhlídky do budoucnosti. Avšak cesta k úspěchu nebyla vždy jednoduchá. Největším problémem bylo pozdržení vývoje, kvůli milným závěrům vědců, na dlouhých 13 let. I přes tento veliký handicap se neuronové sítě stále jeví jako dobrá varianta pro predikci sítového provozu. Abychom však byli schopni učinit objektivní závěry o tom, jak si neuronové sítě skutečně stojí v oboru predikce hodnot nelineárních časových řad, museli bychom provést tentýž rozbor pomocí ostatních možných metod jenž jsou v práci zmíněny. Jedná se především o ARIMA, FARIMA, ARMA modely a o Teorii chaosu. Mezi nejdůležitější vlastnost sítového provozu patří samo-podobnost (která má fraktálovitý charakter).

V důsledku variability řešení pomocí neuronových sítí, bylo nutné testovat každý model zvlášť. Nebylo možné uplatnit žádné zevšeobecnělé pravidlo. Výsledky testování jednotlivých modelů potvrdily teoretický předpoklad uvedený v kapitole 6.1.2., a sice to, že nejpřijatelnější výsledky v mnoha případech lze dosáhnout již pomocí dvouvrstvé neuronové sítě. Jinými slovy, modifikace je možná, avšak nevede ke zlepšení výsledků. V první fázi řešení byl hledán nejvhodnější model sítě. Tím se ukázal být model Obecné rekurentní sítě – LRN model. V další fázi bylo hledáno optimální varianty tohoto modelu. Především v ohledu rozložení chyb a věrnosti predikce. Zde nebylo potvrzeno pravidlo uvedené v kapitole 6.1.2 o počtu neuronů při návrhu sítě. Výsledkem druhé fáze bylo nalezení Obecné rekurentní sítě typu 10-10-1. Ta vykazovala nejlepší výsledky (i za cenu drobného zvýšení střední chyby) ze všech sítí. Pro tento účel bylo využíváno vývojového prostředí Matlab 7.6 s nádstavbou Neural network toolbox 6. Pro agregaci intenzit vstupních dat byl vytvořen program v prostředí Visual studio 2005 (viz příloha C a C.1).

Výsledný model lze v praxi použít například pro dynamickou alokaci šířky pásma jednotlivých front u frontového systému. Neuronová síť totiž může predikovat zvýšený provoz sítě (např. při real-time přenosech), v jehož důsledku by mohlo dojít k zahazování paketů v jedné frontě, zatímco druhá by byla poloprázdná. Obecně tedy lze říci, že neuronové sítě slouží svou činností k vylepšení systémů.

15 Seznam použitých zdrojů

[1] MAŘÍK, V., ŠTĚPÁNKOVÁ, O., LAŽANSKÝ, J. a kolektiv: *Umělá inteligence (4),* Praha: Academia 2003, 475 s. ISBN 80-200-1044-0

[2] KRÁKORA, M., SOJKA, M.: *Vícevrstvý perceptron a RBF sítě* [online]. 5. června 2001. Dostupné na Internetu:

< http://cmp.felk.cvut.cz/cmp/courses/recognition/zapis/rpz8_01l.ps>

[3] ŠIMEČEK. P.: *Úvod do fraktálů a chaosu* [online]. Dostupné na internetu:

<http://mks.mff.cuni.cz/library/UvodDoFraktaluAChaosuPS/UvodDoFraktaluAChaosuPS.ps >

[4] LELAND, W., TAQQU., M., WILSON, D.: *IEEW/ACM transactions on networking, vol. 2,no.1,february 1994*

[5] KACÁLEK, J.: *Metody modelování a predikce sítového provozu,pojednání o disertační práci,* Brno, duben 2008

[6] VOLNÁ, E.: *Neuronové sítě 1,* Ostravská univerzita, Přírodovědecká fakulta, Ostrava 2002.

[7] ZÁPADOČESKÁ UNIVERZITA V PLZNI, Fakulta aplikovaných věd, katedra informatiky a výpočetní techniky [online]. *UIR.* Dostupné na internetu:

<http://fav.q-e-e.net/UIR/P%f8edn%e1%9aky/07.Neuronov%e9%20s%edt%ec/>

[8] ZÁVODNÝ, L., HANUS, S.: *Využití umělé neuronové sítě pro empirický model šíření signálu* [online]

Dostupné z intenetu: <http://dsp.vscht.cz/konference_matlab/matlab04/zavodny.pdf>

[9] TAUFER, I., DRÁBEK, O., SEIDL, P.: *Umělé neuronové sítě – základy teorie a aplikace*

[10] JIRSÍK, V.: *Umělé neuronové sítě* [online]. Dostupné na internetu: <http://jaja.kn.vutbr.cz/~belovic/MUIN/03_UMI_umele_neuronov%E9_s%EDt%EC.pdf>

[11] WIKIPEDIA, *Neuronová síť* [online]. Dostupné na internetu:

<http://wikipedia.infostar.cz/n/ne/neural_network.html#Structure>

12] WIKIPEDIA, *Neuronové sítě.* [online] Dostupné na internetu:

<http://cs.wikipedia.org/wiki/Neuronov%C3%A1_s%C3%AD%C5%A5>

[13] CROVELLA, E., BESTAVROS, A.: Computer Science Depatrment, Boston University, Self-Similarity in World Wide Web Traffic – Evidence and Possible Causes

[14] LETLANT, W. E., MEMBER, IEEE.: On the Self-similiar Nature of Ethernet traffic (Extended version) vol. 2. February 1994

[15] SAH, K., BOHACEK, S., JONCKHEERE, E.: On the Predictability of Data Network Traffic, Denver, Colorado 2003, 0-7803-7896-2/03/$17.00_2003 IEEE

[16] DEMUTH, H., BEALE, M., HOGAN, M.: *Neural Network Toolbox™ 6 User's Guide* [online]. 2009. Dostupné na internetu:

< http://www.mathworks.com/access/helpdesk/help/pdf_doc/nnet/nnet.pdf>

[17] DOHNAL, L.: Desatero pro porovnání výsledků dvou metod [online]. Dostupné na internetu: < http://www1.lf1.cuni.cz/~ldohna/dohnal/desatero.htm>

16 Seznam použitých symbolů a zkratek

BP	Back Propagation – algoritmus zpětné šíření chyby
FARIMA	The Fractional Autoregressive Intergrated Moving Average -
INNS	International Neural Network Society – mezinárodní společnost pro výzkum neuronových sítí
LAN	Local Area Network – lokální síť
MLP	Multilayer perceptron - vícevrstvá perceptronová síť
NS	Neuronová síť
PDP	Paralel Distributed Processing – paralelní procesy
UNS	Umělá neuronová síť
TRM	Trénovací množina
WWW	World Wide Web – celosvětový web

17 SEZNAM PŘÍLOH

A. OBSAH CD

1. Diplomová práce:

   Elektronická verze diplomové práce – soubor pdf

2. Zachycená data:

   Jednotlivá data se kterými bylo pracováno jsou v adresáři Zachycena_data. Jde o zachycený síťový provoz, který je vstupem celé práce.

3. Agregovaná data:

   Pomocí programu pro agregaci intenzit provozu jsou vstupní data upravena a uložena v adresáři Agregovana_data, v souboru date.txt.

4. Program na agregaci intenzit provozu:

   Je umístěn v adresáři TrafficTrace. Je možné jej spustit buď jako exe soubor z cesty cesty\TrafficTrace\TrafficTrace\bin\Debug\TrafficTrace.exe jako spustitelný soubor (viz podmínky v příloze C1. Spuštění programu), nebo otevřením projektu v adresáři TrafficTrace.

5. Konfigurační soubory neuronových sítí:

   Jednotlivé testované neuronové sítě jsou vyexportovány a uloženy adresáři nntool. Všechny proměnné jsou uloženy v souboru promenne.mat.

B. PODROBNOSTI ZPRACOVÁVANÝCH DAT

Traffic Trace Info

DumpFile: 200107050900.dump
FileSize: 196.28MB
Id: 200107050900
StartTime: Thu Jul 5 09:00:01 2001
EndTime: Thu Jul 5 11:49:56 2001
TotalTime: 10194.47 seconds
TotalCapSize: 150.12MB CapLen: 96 bytes
# of packets: 2000000 (1789.41MB)
AvgRate: 1.47Mbps stddev:3.40M

IP flow (unique src/dst pair) Information

# of flows: 1224 (avg. 1633.99 pkts/flow)
Top 10 big flow size (bytes/total in %):
71.9% 24.6% 1.4% 1.0% 0.1% 0.1% 0.1% 0.0% 0.0% 0.0%

IP address Information

# of IPv6 addresses: 562
Top 10 bandwidth usage (bytes/total in %):
73.3% 73.3% 24.7% 24.6% 1.0% 1.0% 0.1% 0.1% 0.1% 0.1%

protocol	packets	bytes	bytes/pkt
total	2000000 (100.00%)	1876334980 (100.00%)	938.17
ip6	2000000 (100.00%)	1876334980 (100.00%)	938.17
tcp6	1471991 (73.60%)	1402437196 (74.74%)	952.75
http(s)	1302 (0.07%)	1046551 (0.06%)	803.80
http(c)	1148 (0.06%)	140544 (0.01%)	122.43
smtp	2586 (0.13%)	553935 (0.03%)	214.21
nntp	62 (0.00%)	4836 (0.00%)	78.00
ftp	8008 (0.40%)	899091 (0.05%)	112.27
pop3	3917 (0.20%)	2472391 (0.13%)	631.20
ssh	8349 (0.42%)	1606318 (0.09%)	192.40
dns	24 (0.00%)	5088 (0.00%)	212.00
bgp	21620 (1.08%)	2111442 (0.11%)	97.66
other	1424975 (71.25%)	1393597000 (74.27%)	977.98
udp6	4246 (0.21%)	556075 (0.03%)	130.96
dns	2498 (0.12%)	310791 (0.02%)	124.42
other	1748 (0.09%)	245284 (0.01%)	140.32
icmp6	94304 (4.72%)	11751022 (0.63%)	124.61
rtopt6	429120 (21.46%)	461549329 (24.60%)	1075.57
pim6	339 (0.02%)	41358 (0.00%)	122.00

Obr 16.1: Zobrazení zachycených dat pro vstup do NS

C. ZDROJOVÝ KÓD PROGRAMU PRO AGREGACI DAT

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO;

namespace TrafficTrace
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void label1_Click(object sender, EventArgs e)
        {

        }

        private void otevřiToolStripMenuItem_Click(object sender,
      EventArgse)
        {
            OpenFileDialog dialog = new OpenFileDialog();
            dialog.Filter =
               "txt files (*.txt)|*.txt|All files (*.*)|*.*";
            //dialog.InitialDirectory = initialDirectory;
            dialog.Title = "Select a text file";
            if (dialog.ShowDialog() == DialogResult.OK)
            {
                this.fileName = dialog.FileName;
            }
        }

        private double convertToDouble(string cislo)
        {
            string [] spl = cislo.Split(new char [] { Convert.ToChar(".")
         });
            double Ceil;
            double.TryParse(spl[0].ToString(), out Ceil);
            int delka =  spl[1].Length;
            string digits = spl[1];
            double cil = 0;
            for (int i = 0; i < delka; i++)
            {
                if (digits[i] != Convert.ToChar("0"))
                {

                    double.TryParse(digits, out cil);
                    i = delka + 1;
                }
```

```
    }
    double dev = 1000000;
    return Math.Round(Ceil + cil/dev, 1);
}

private void button1_Click(object sender, EventArgs e)
{
    StreamReader SR = null;
    string krok = textBox1.Text;
    double Krok = 1;
    if (krok != null)
    {
        try
        {
            Krok = Convert.ToDouble(krok)/1000;
        }
        catch { }
    }

    using (TextWriter tw = new StreamWriter("c://date.txt"))
    {
        try
        {
            int AggrBytes = 0;
            SR = File.OpenText(this.fileName);
            string S;
            int i = 0;
            double time = 1 + Krok;
            S = SR.ReadLine();
            while (!SR.EndOfStream)
            {
                S = SR.ReadLine();
                string[] split = S.Split(new char[] {
                   Convert.ToChar(" ") });
                double d = convertToDouble(split[0]);
                if (d > 1)
                {
                    /*if (d > 501)
                    {
                        return;
                    } */
                    if (d > time)
                        {

                            //write to file

                        tw.WriteLine(time.ToString().Replace(",",
                 ".") + " " + AggrBytes.ToString());
                            AggrBytes = 0;
                            time += Krok;
                        }
                        else
                        {
                            AggrBytes += Convert.ToInt32(split[5]);
                        }

                }
                i++;
```

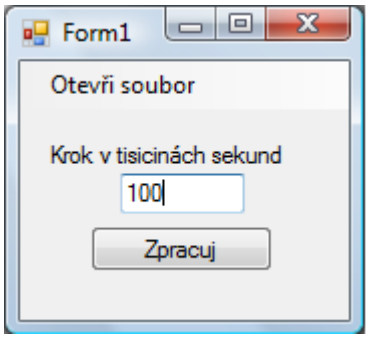
```
                }

            }
            finally
            {
                if (SR != null)
                    SR.Close();
            }
        }

    }
}

}
```

C.1 Spuštění programu

Program byl vytvořen ve vývojovém prostředí Visual studio 2005. Pro jeho spuštění pomocí .exe souboru je nutné mít v počítači nainstalovaný .Net Framework minimálně verze 2. Lépe ovšem přímo Visual studio (je možná i verze z roku 2008).

Po spuštění programu je třeba načíst soubor se zachycenými daty klepnutím na položku „Otevři soubor“ (dec-pkt-4.tcp) a zadat časový interval pro agregaci (v našem případě 100 ms). Poté klepneme na tlačítko „zpracuj“. Výsledek bude uložen za několik sekund do výstupního souboru date.txt.

Pozn. V programu je nastavena standardní cesta k výstupnímu souboru C://date.txt. Avšak na počítači se systémem Windows Vista může jeho spuštění znamenat potíže, neboť systém nedovoluje druhým aplikacím vytvářet soubory v kořenovém adresáři. V tomto případě je tedy vhodné cestu změnit, nejlépe na jiný logický oddíl.



Obr 17.1 Grafická podoba programu pro agregaci intenzit provozu