ELECTRONICS AND ELECTRICAL ENGINEERING
ISSN 1392 – 1215 *2006. Nr. 4(68)*
ELEKTRONIKA IR ELEKTROTECHNIKA

TELECOMMUNICATION ENGINEERING
T180
TELEKOMUNIKACIJŲ INŽINERIJA

Neural Network Models for Internet Traffic Prediction

G. Rutka
Faculty of Electronics and Telecommunication, Riga Technical University,
Riga, Azenes str. 12, LV-1048, phone:+371 9627600, e-mail: gundegarutka@tvnet.lv

Introduction

Internet traffic prediction plays a fundamental role in network design, management, control, and optimization. The self-similar and non-linear nature of network traffic makes high accurate prediction difficult. It has been found in numerous studies that data traffic in high-speed networks exhibits self-similarity that cannot be captured by classical models, hence self-similar models have been developed [1]. The problem with self-similar models is that they are computationally complex. Their fitting procedure is time consuming while their parameters cannot be estimated based on the on-line measurements. The goal is to forecast future traffic variations as precisely as possible, based on the measured traffic history. Traffic prediction requires accurate traffic models that can capture the statistical characteristics of actual traffic. Our experiments inspect performance using multilayer perceptrons and radial basis function networks. There are no standard network topologies and algorithms around which all design efforts can be based.

Self- Similarity

The process is self-similar if its statistical behavior is independent of the time-scale. This means that averaging over equal periods of time does not change the statistical characteristics of the process.

Suppose $X=\{X_t: t=0,1,2\ldots\}$ is a covariance stationary stochastic process with mean μ, variance σ^2 and autocorrelation function r(k), where k=0,1,2… In particular, we assume that X has an autocorrelation function of the form:

$$r(k) \sim k^{(2-2H)} L(k) \text{ as } k \to \infty, \quad (1)$$

where H is called the Hurst parameter and L(k) is slowly varying at infinity, that is:

$$\frac{\lim_{t\to\infty} L(zt)}{L(t)} = 1 \quad \text{for all } z>0. \quad (2)$$

An example of such slowly varying functions which satisfies (2) is given by L(t)=log(t). The Hurst parameter H in (1) is in the range $0.5<H<1$ and it characterizes the process in terms of the degree of self-similarity and long time dependence. The degree of self-similarity and long-range dependence increases as $H \to 1$. For each m=1,2,3…, let $X^{(m)}=\{ X_k^{(m)}: k=1,2,3\ldots\}$ denote a new time series obtained by averaging the original series X over non-overlapping blocks of size m. That means, for each m=1,2,3… , $X_k^{(m)}$ is given by:

$$X_k^{(m)} = \frac{X_{km-m+1} + \ldots + X_{km}}{m}, \quad (3)$$

where k=1,2,3,… . Note that for each m, the aggregate time series $X^{(m)}$ defines a covariance stationary process [2,3].

In our experiments self-similarity will be estimated by the use of variance-time plot method. This is one of the easiest methods how to estimate Hurst's coefficient. In the process the variance of aggregate the self-similar process is defined:

$$VAR(X^{(m)}) = VAR(X)/m^{\beta}. \quad (4)$$

In the (4) β is calculated from the equation:

$$H=1-\beta/2. \quad (5)$$

The (4) can be rewritten is the following form:

$$\log\{ VAR(X^{(m)})\} \sim \log\{ VAR(X) \} - \beta \log\{m\}. \quad (6)$$

If VAR(X) and m are plotted on a log-log graph then by fitting a least square line through the resulting points we can obtain a straight line with the slope of $-\beta$.

Neural Networks

Neural networks are composed of simple elements operating in parallel. These elements are inspired by biological nervous systems. As in nature, the network function is determined largely by the connections between elements. We can train a neural network to perform a particular function by adjusting the values of the connections (weights) between elements.

Many authors have applied many different neural network architectures and algorithms to explore traffic modeling task. For example, for the Internet traffic prediction there is explored boosting feed forward neural

network, for self-similar traffic generation is used perceptron neural network with back propagation algorithm etc.

Multilayer perceptrons

The Multilayer Perceptron (or MLP) network is probably the most-often considered member of the neural network family. The main reason for this is its ability to model simple as well as very complex functional relationships. This has been proven through a large number of practical applications.

An MLP is a network of simple neurons called perceptrons. The perceptron computes a single output from multiple real-valued inputs by forming a linear combination according to its input weights and then possibly putting the output through some nonlinear activation function. Mathematically this can be written as

$$y = \varphi(\sum_{i=1}^{n} w_i x_i + b) = \varphi(w^T x + b), \quad (7)$$

where w denotes the vector of weights, x is the vector of inputs, b is the bias and φ is the activation function.

A single perceptron is not very useful because of its limited mapping ability. No matter what activation function is used, the perceptron is only able to represent an oriented ridge-like function. The perceptrons can, however, be used as building blocks of a larger, much more practical structure. A typical MLP network (MLPN) consists of a set of source nodes forming the input layer, one or more hidden layers of computation nodes, and an output layer of nodes. There are three activation functions used in MLPN:

1. Log sigmoid transfer function also known as the logistic sigmoid (logsig):

$$\text{logsig}(x) = \frac{1}{1+e^{-x}}, \quad (8)$$

2. Hyperbolic tangent sigmoid transfer function also known as the hyperbolic tangent (tansig):

$$\text{tansig}(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}. \quad (9)$$

The logistic sigmoid and the hyperbolic tangent are related by:

$$\frac{\text{tansig}(x)+1}{2} = \frac{1}{1+e^{-2x}}. \quad (10)$$

Recalculating formula (10) we obtain:

$$\text{tansig}(x) = \frac{2}{1+e^{-2x}} - 1. \quad (11)$$

3. Linear transfer function also known as simple purelin function (purelin). The purelin transfer function is simply a linear function that produces the same output as its input:

$$\text{purelin}(x) = x. \quad (12)$$

These functions are used because they are mathematically convenient and are close to linear near origin while saturating rather quickly when getting away from the origin. This allows MLP networks to model well both strongly and mildly nonlinear mappings.

Mean square error (MSE) or mean squared error performance function is an accepted measure of prediction and is very often used in MLP networks. MSE is calculated:

$$\text{MSE} = \frac{1}{m}\sum_{i=1}^{m}(x_i - T)^2, \quad (13)$$

where x_i is i–value of a group of m values, T is a target or intended, i.e., desired, value for the product variable of interest.

Radial basis function networks

A general architecture of the radial basis function networks (RBFN) consists of three layers. The first layer consists of *n* inputs. They are fully connected to the neurons in the second layer. A hidden node has a radial basis function (RBF) as an activation function. The RBF is a radially symmetric function (e.g., Gaussian):

$$f(x) = e^{-\frac{(x-M)^2}{2\sigma^2}}, \quad (14)$$

where M and σ are two parameters meaning the mean and the standard deviation of the input variable x.

For a particular intermediate node *i*, its RBFi is centered at a cluster center c_i in the *n*-dimensional input space. The cluster center c_i is represented by the vector (w_{1i} . . . ,w_{ni}) of connection weights between the *n* input nodes and the hidden node *i*. The standard deviation for this cluster defines the range for the RBFi.

The RBF is no monotonic, in contrast to the sigmoid function. The second layer is connected to the output layer. The output nodes perform a simple summation function with a linear threshold activation function. The training of an RBFN consists of two phases: (1) adjusting the RBF of the hidden neurons by applying a statistical clustering method; this represents an unsupervised learning phase; (2) applying gradient descent (e.g., the back propagation algorithm) or a linear regression algorithm for adjusting the second layer of connections; this is a supervised learning phase [4].

Research Models

Our research is emphasized to self- similar traffic prediction using neural networks. Traffic data is taken from website http://freestats.com/. This data was collected for one year. For statistical analyses and neural network testing we use program package "MATLAB p6.5".

Firstly, we have deeply studied the character of the statistical material (traffic data). Accordingly to that we have calculated and proved that traffic data is self-similar.

Secondly, we modulate different neural network models to verify reliability of made prediction. In our research we work with two types of neural networks:

RBFN and MLPN, the same as feed forward back propagation network.

We have implemented several MLPN with two and three layers, each layer containing one-neuron and several RBFN. In our simulations with MPLN we use three kinds of transfer functions: 'tansig', 'logsig' and 'purelin' and for faster training we use Levenberg-Marquardt algorithm, variable learning rate and conjugate gradient algorithms. For RBFN we use performance goal =0 and spread constant=1.0 .

Review Of Studied Cases

The maximum data (website access statistics) was collected for one year - 365 days, 24 hours a day. In the total we have obtained 8760 observations. With the help of Matlab v.6.5 using method of variance-time plot we have calculated the Hurst parameter H, which describes whether the traffic is self-similar. From the sequence of obtained observations as a function of the aggregation level m, we estimate the Hurst parameter H. The obtained values of the slope are shown in Table 1.

Table 1. The variance-time plot values

m	log (m)	VAR(X)	log (VAR(X))
2	0.30103	3759	3.5751
3	0.47712	2881.1	3.4596
4	0.60206	2431.5	3.3859
6	0.77815	1912.1	3.2815
8	0.90309	1635.7	3.2137
12	1.0792	1037.7	3.0161
24	1.3802	629.26	2. 7988
40	1.6021	487.06	2.6876
73	1.8633	322.22	2.5082
146	2.1644	240.32	2.3808
730	2.8633	148.76	2.1725
1095	3.0394	147.01	2.1674
2190	3.3404	87.677	1.9429
2920	3.4654	109.73	2.0403
4380	3.6415	40.516	1.6076

The variance-time curve (Fig.1) shows an asymptotic slope that is easily estimated to be about –0.51, resulting in a practically identical estimate of the Hurst parameter H of about 0.745.

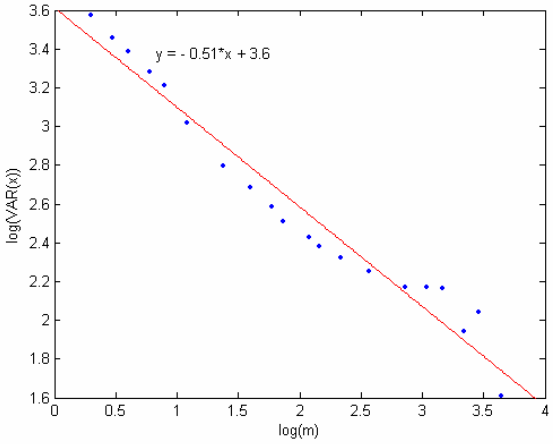

Fig. 1. The variance- time plot

Our experimental continuous-time stochastic process X(t) is considered to be statistical self-similar with parameter H =0.745 (0.5<H<1).

The second step of our experiment was to predict network traffic using different neural network models. For our experiments we used the obtained data for self-similarity calculations for m=2, 3, 4, 6, 8, 12, 24, 40, 73 and 146. We have created approximately 180 networks to test the prediction possibility and accuracy with neural networks. In Table 2–5 we have summarized the results of made experiments.

Table 2. Results for m=2, 3, 4 and 6 using MLPN.

Transfer function		Aggregation level m / Performance			
Layer1	**Layer2**	**2**	**3**	**4**	**6**
logsig	purelin	$1.7e^{-31}$	$1.5e^{-29}$	$5.9e^{-33}$	$1.4e^{-28}$
logsig	logsig	$1.2e^{-16}$	$1.8e^{-16}$	$1.7e^{-17}$	$7.1e^{-15}$
logsig	tansig	$1.5e^{-31}$	0	$4.3e^{-33}$	0
tansig	purelin	$3.6e^{-35}$	$8.3e^{-34}$	$7.29e^{-32}$	$6.9e^{-31}$
tansig	logsig	$1.4e^{-16}$	$3.4e^{-16}$	$3.6e^{-16}$	$3.6e^{-16}$
tansig	tansig	$1.4e^{-34}$	0	0	$1.1e^{-32}$
purelin	purelin	$3.2e^{-53}$	$7.9e^{-36}$	$1.6e^{-35}$	$1.8e^{-34}$
purelin	logsig	$1.8e^{-16}$	$7.5e^{-16}$	$2.7e^{-16}$	0
purelin	tansig	0	0	$1e^{-31}$	$3.2e^{-30}$

Table 3. Results for m=8, 12 and 24 using MLPN.

Transfer function		Aggregation level m / Performance		
Layer1	**Layer2**	**8**	**12**	**24**
logsig	purelin	$1.6e^{-26}$	$4.1e^{-28}$	$9.7e^{-28}$
logsig	logsig	$9e^{-16}$	$5.2e^{-14}$	$3e^{-15}$
logsig	tansig	$2.1e^{-31}$	0	0
tansig	purelin	$8.1e^{-29}$	$1.1e^{-29}$	$3.5e^{-31}$
tansig	logsig	$2.1e^{-14}$	$2.5e^{-16}$	$8.1e^{-15}$
tansig	tansig	$3.1e^{-29}$	0	0
purelin	purelin	$4.6e^{-36}$	0	$6e^{-41}$
purelin	logsig	0	0	0
purelin	tansig	$5.3e^{-29}$	$4.6e^{-30}$	$1.9e^{-30}$

Table 4. Results for m=40, 73 and 146 using MLPN.

Transfer function		Aggregation level m / Performance		
Layer1	**Layer2**	**40**	**73**	**146**
logsig	purelin	$3.3e^{-28}$	$5.4e^{-28}$	$2.9e^{-27}$
logsig	logsig	$2e^{-15}$	$5.5e^{-15}$	$4.8e^{-13}$
logsig	tansig	$2e^{-31}$	$1.8e^{-26}$	$4.9e^{-33}$
tansig	purelin	$1.3e^{-29}$	$2.9e^{-33}$	$3.1e^{-30}$
tansig	logsig	$6.7e^{-15}$	$9.9e^{-16}$	$2.7e^{-15}$
tansig	tansig	0	0	$3.9e^{-31}$
purelin	purelin	$6.6e^{-29}$	$2e^{-28}$	0
purelin	logsig	$1.2e^{-15}$	$1.4e^{-16}$	1
purelin	tansig	$7.2e^{-28}$	$7e^{-30}$	1

As we see in Table 2 – Table 4, the best performance (performance=0) was attempted with 2 layers MLPN using such transfer functions: "logsig- tansig", "tansig-tansig," "purelin-purelin", "purelin-logsig" and "purelin-tansig". The worst results of performance using 2 layers MLPN were cases with transfer functions "logsig-logsig" and

"tansig-logsig". We tried to improve the performance in these cases, multiplying the number of layers. In the Table 5 we can see the results.

Table 5. Results of 3 layer MLPN.

m	Transfer functions for 3 layers					
	logsig-logsig-			tansig-logsig-		
	purelin	logsig	tansig	purelin	logsig	tansig
2	$4.9e^{-31}$	$5.2e^{-15}$	0	$4e^{-32}$	$1e^{-14}$	$5.6e^{-29}$
3	$2.4e^{-30}$	$4.9e^{-15}$	0	$9e^{-29}$	$4.1e^{-15}$	0
4	$1.1e^{-29}$	$1e^{-14}$	0	$8.3e^{-29}$	$2.5e^{-14}$	$7.6e^{-34}$
6	$7.4e^{-27}$	$7.5e^{-15}$	0	$1.3e^{-26}$	$2e^{-14}$	$3.3e^{-30}$
8	$1.6e^{-27}$	$1.7e^{-15}$	0	$4.6e^{-31}$	$3.1e^{-14}$	$5.4e^{-32}$
12	$3.3e^{-28}$	$7.3e^{-14}$	0	$4.8e^{-28}$	$6.3e^{-15}$	0
24	$2e^{-27}$	$1.7e^{-13}$	0	$1.3e^{-26}$	$5.2e^{-15}$	$4.9e^{-32}$
40	$8.3e^{-28}$	$1.3e^{-14}$	0	$7.9e^{-27}$	$6.7e^{-15}$	$4.9e^{-32}$
73	$1.8e^{-31}$	$1.1e^{-13}$	0	$2.8e^{-26}$	$1.4e^{-14}$	$1.2e^{-32}$
146	$5.4e^{-30}$	$3.3e^{-13}$	0	$8.2e^{-29}$	$5.2e^{-13}$	$1.4e^{-30}$

Table 5 shows that the performance has been improved. Ideally, in the case when MLPN has transfer function "logsig-logsig-tansig", the performance is 0. The worst results of performance using 3 layers MLPN were cases with transfer functions "logsig-logsig-logsig" and "tansig-logsig-logsig". It means that it is not recommended to use transfer function "logsig" in the last layer of MLPN.

The prediction experiments we also made with RBFN. The obtained results are in Table 6.

Table 6. The results of experiments made with RBFN.

Aggregation level m	Result
2	not available
3	not available
4	650 epochs, performance =$2.9e^{-6}$
6	775 epochs, performance =$6.6e^{-7}$
8	700 epochs, performance =$1.1e^{-7}$
12	725 epochs, performance= $1.4e^{-5}$
24	350 epochs, performance =$8.3e^{-8}$
40	200 epochs, performance =$1.4e^{-11}$
73	100 epochs, performance =$2.4e^{-9}$
146	50 epochs, performance =$4.9e^{-5}$

Table 6 shows that prediction models using RBFN is not very efficient. Depending on your personal PC technical potential, the obtained results are calculated within 20-60 minutes. In the cases of aggregation level m=2 and m=3, the PC wasn't able to produce result. The RBFN features can explain it. RBFN needs the same number of epochs as the number of neurons.

Conclusions

The performance of traffic prediction model of MLPN can be improved by multiplying the number of layers. There is no need to increase the number of neurons in each layer.

Calculations using RBFN models take a long time and the performance goal in all cases was not met. It is not recommended to use RBFN to predict traffic in the presence of self-similarity, because of the complex calculation and long calculation period.

MLPN models are the best models for traffic prediction in the presence of self-similarity.

This work has been partly supported by the European Social Fund within the National Program "Support for the carrying out doctoral study program's and post- doctoral researches" project "Support for the development of doctoral studies at Riga Technical University".

References

1. **Bestavros A**., **Crovella M. E.** Self-similarity in world wide web traffic: Evidence and possible causes// IEEE/ACM Trans. Networking, 1997. – Vol. 5. – P. 835–846.
2. **Fower B., Thomas Dr.** A short tutorial on fractals and internet traffic// The telecommunications review, 1999.
3. **Ghaderi M.** On the Relevance of Self-Similarity in Network Traffic Prediction// Tech. Rep., CS-2003-28, University of Waterloo 2003.
4. **Nørgaard M.** Neural Network Based System Identification toolbox, vers.2 for Matlab.

Presented for publication 2006 02 28

G. Rutka. Neural Network Models for Internet Traffic Prediction // Electronics and Electrical Engineering. - Kaunas: Technologija, 2006. – No. 4(68). – P. 55–58.

This paper presents a view of models used for Internet data (traffic) prediction using neural network applications. We look at the problem of traffic prediction in the presence of self-similarity. Self-similarity is an important characteristic of traffic in high-speed networks that cannot be captured by traditional traffic models. Our experiments inspect performance using multilayer perceptrons and radial basis function networks. Ill. 1, bibl. 4 (in English; summaries in English, Russian and Lithuanian).

Г. Рутка. Модели нейронных сетях для предсказания трафика Интернета //Электроника и электротехника. – Каунас: Технология, 2006. – № 4(68). – С. 55–58.

В работе рассказано о моделях, которые используются для предсказания нагрузки в нейронных сетях. Мы рассматриваем проблему предсказания трафика в случае самопохожести. Самопохожесть – это очень важная особенность трафика в быстроскоростных сетях, которую невозможно встретить при обычной модели трафика. Наши эксперименты показывают использование сетей, функционально основанных на использовании многослойных персептронов и радиальной базисной функции. Ил 1, библ. 4 (на английском языке; рефераты на английском, русском и литовском яз.).

G. Rutka. Interneto duomenų srautų prognozavimo neuroninių tinklų modeliai // Elektronika ir elektrotechnika. – Kaunas: Technologija, 2006. – Nr. 4(68). P. 55–58.

Straipsnyje tiriami modeliai, taikomi interneto duomenų srautams prognozuoti panaudojant neuroninius tinklus. Duomenų srauto prognozavimo problema analizuojama savaiminio panašumo atvejais. Savaiminis panašumas yra svarbi duomenų srauto didelės spartos tinkluose charakteristika, kuri neįvertinama tradiciniais duomenų srautų modeliais. Eksperimentuose tiriamas našumas naudojant daugiasluoksnius perceptronus ir radialines bazines funkcijas. Il. 1, bibl. 4 (anglų kalba; santraukos anglų, rusų ir lietuvių k.).