Threshold Neural Network Model for VBR Video Trace

가변적 비디오 트랙을 위한 임계형 신경망 모델

장봉석
목포대학교 정보공학부 멀티미디어공학전공

Bong-Seog Jang(jang@mokpo.ac.kr)

요약

본 논문은 가변적 비디오 트랙을 위한 모델링 방법을 제시한다. 가변적인 비디오 트랙은 간헐적인 버스트 및 긴 구간 상관관계의 특성을 갖는다고 잘 알려져 있다. 이러한 데이터를 분석하기 위해서, 에러 임계값으로부터 구한 보조적인 선형 구조를 갖는 신경망 구조 모델 구축을 한다. 모델링 결과 테스트를 위해서, 혼돈 비선형 함수와 지수 랜덤 노이즈를 결합한 가변적 비디오 트랙을 발생하였다. 발생된 데이터를 모델링한 결과, 전통적인 신경망 모델에 비해서 제시된 모델이 보다 정확한 모델링 결과를 보여 주었다. 그러나 또한 제시된 모델에 ARMA을 결합한 결과가 제시된 모델 단독인 경우에 비해서 더욱 발생된 데이터의 통계적 특성에 근접함을 발견했다.

■ keywords: | 가변적 비디오 | 신경망 | 임계값 | ARMA | 모델링 | 학습 | 시계열 |

Abstract

This paper shows modeling methods for VBR video trace. It is well known that VBR video trace is characterized as longterm correlated and highly intermittent burst data. To analyze this, we attempt to model it using neural network with auxiliary linear structures derived from residual threshold. For testing purpose, we generate VBR video trace from chaotic nonlinear function combined with the geometric random noise. The modeling result of the generated data shows that the attempted method represents more accurately than the traditional neural network. However, we also found that combining ARMA to the attempted modeling method can yield a closer agreement to statistical features of the generated data than the attempted modeling method alone.

■ Keywords : | VBR Video | Neural Network | Threshold | ARMA | Modeling | Training | Time Series |

I. INTRODUCTION

Neural networks have been applied to time-series modeling and prediction problems[1]. In this paper a new neural network model is developed for highly variable and correlated time-series such as moving picture expert group (MPEG) coded variable bit rate (VBR) video data. The most popular neural networks use a multi-layered architecture trained using the error back-propagation (BP) algorithm. Neural networks

접수번호 : #060104-001
접수일자 : 2006년 01월 04일
심사완료일 : 2006년 01월 17일
교신저자 : 장봉석, e-mail : jang@mokpo.ac.kr

trained using the BP algorithm have been successfully applied to problems in optimization, control, pattern recognition, modeling and learning[2].

One of the limitations of feed-forward (FF) neural networks with BP training is that sudden changes in the output value are often not modeled accurately. Furthermore, the performance of BP trained FF neural networks will depend on its structure. The optimal structure for a given problem is generally not known [2][3].

The conventional neural model has been examined for use in solving the problems of multimedia communication such as call admission control and link capacity allocation problems in various networks[4]. Link capacity estimator using the neural model has been shown to outperform the Erlang based link capacity estimator. The conventional neural model has also been examined for use in the policing function in asynchronous transmission mode (ATM) networks[5]. The mean and peak bit-rates in ATM traffic were approximated using the model. These two parameters were used to predict possible violations of the negotiated bit-rates. The pi-sigma neural network has been used for the prediction of bandwidth allocation for real-timeVBR video over ATM networks[6]. The results were compared to those obtained using a recursive least square prediction method. In the simulation, the pi-sigma network was used to dynamically predict bandwidth requirements for full motion video traffic. The video traffic used in the simulation was filtered to smooth the bit-rate variations caused by scene changes. Two minutes of the filtered video data was used to train the pi-sigma neural network. It was shown that the on-line observation of the input traffic stream using the pi-sigma neural network allowed the prediction of near future bandwidth demands in the ATM network.

In this paper a new VBR video trace modeling method using the FF neural network will be developed. VBR video trace is modeled using the FF neural model with BP training. We willalso develop auxiliary neural networks to model the sudden amplitude changes in the trace. These auxiliary models have linear function instead of the sigmoid functions that will be switched by the prescribed amplitude ranges. This model is termed the threshold neural network (TNN) model. The TNN cannot easily process all the elements of stochastic video trace data. Therefore a procedure for modeling a residual data using the autoregressive moving average (ARMA) process will be examined.

The organization of this paper is as follows. In section II general features of the conventional neural model are reviewed. Section III discusses the procedure of the TNN modeling approach. The analysis and synthesis steps are examined in this section. In section IV verification of the new method is discussed. Finally conclusions are presented in section V.

II. CONVENTIONAL NEURAL NETWORK MODELS

Many types of neural network and training algorithms have been developed [1-3]. The method used to train the neural network is based on error minimization using a finite set of input and output samples. To estimate the input-output relation in the multi-layered neural network, the input data is multiplied by weight vectors and the result is forwarded to the output layer through the hidden layers. In the training phase, the estimates are compared to actual data and the error signals are used to modify the weight vectors in each layer. This procedure is continued for each member of the training set until a desired error threshold is reached. Therefore the neural network yields the estimates of transfer functions between the given input and the output

time-series. We will use sigmoid functions for the nonlinear transfer functions in the hidden layer.

$$v_j = \frac{1}{1+\exp(-ah_j)} \quad (1)$$

where v_j represents the output of the j-th node, a is the slope parameter, and h_j is a weighted sum of the inputs at j-th node. The sigmoid function is differentiable and assumes a continuous range of values between 0 andl.

1. Multi-layered Neural Network

The multi-layered neural network consists of the hidden, input and output layers. A recurrent structure [2][3] is implemented by feeding the output back to the input nodes. We will only consider the FF structure here.

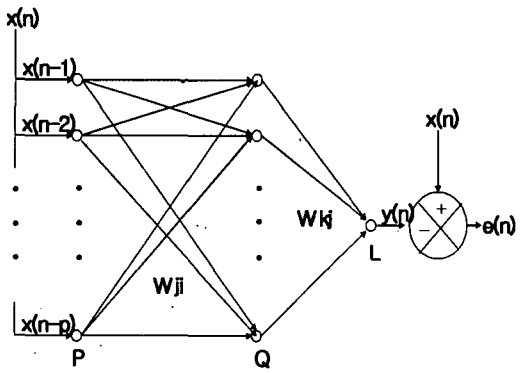

Figure 1. Feed-forward neural network structure

In this subsection, calculation of the estimates in the FF neural network is reviewed. A baseline neural model in this paper will consist of a hidden layer with nodes having the sigmoid functions, and an outputand input layer. In [Figure 1], the neural network has p input nodes in the input layer P, q hidden nodes in the hidden layer Q and one output node in the output layer L. The p-lagged elements of x(n) are used as the input data and y(n) is the estimate of x(n). The weight matrix elements are shown as w_{ji} for the input-to-hidden layer and w_{kj} forthe hidden-to-output layer. The error is obtained in the output layer by comparing x(n) and y(n).

Let us denote the vector $\underline{x}_P$ as the input data to the neural network. This vector can be comprised of time-delayed elements of the time-series $x(n-1), x(n-2), \cdots, x(n-p)$ and will be used to predict the outcome x(n). The FF calculations in the multi-layer neural network are as follows. The intermittent output at the hidden layer is written as

$$\underline{h}_Q = W_{QP}\underline{x}_p \quad (2)$$

where Q denotes the hidden layer and $\underline{h}_Q$ is a vector of the weighted sum of the input data. The subscript P is used to denote the input layer and $\underline{x}_P$ is the input vector having p elements. W_{QP} denotes the weight matrix for the input-to-hidden layer path. In the hidden layer each node uses a sigmoid transfer function to weight the outcome vector $\underline{h}_Q$. Hence the output of the hidden nodes are transformed by the nonlinear function given in Eq.(1).

The estimates are calculated in the output node as

$$\underline{y}_L = W_{LQ}\underline{v}_Q \quad (3)$$

where L denotes the output layer, $\underline{y}_L$ is a vector having the estimates obtained at each output node, W_{LQ} is the weight matrix between the output and hidden layers, and $\underline{v}_Q$ is a vector having elements of v_j obtained by Eq. (1).

In the learning phase, pairs of input-output vectors are presented to the neural network. For each presentation of the input-output pair the error signals

are calculated. The error signals are used to modify the weight matrix W using a total squared error minimization procedure. The process of modification of the Wwill be briefly discussed in the following section. After the weights have converged to their equilibrium values, estimates are computed using the W and new input data values.

Ideally, the residual series derived from the trained neural network is an uncorrelated sequence. In such a case the neural network can be regarded as a nonlinear whitening filter [2]. The nonlinear transfer functions in the neural network embed the temporal correlations in its structure [2]. However it is found that a Gaussian distributed residual series is often not easily achieved. For highly varying input data, it has been observed that the residual probability density function (pdf) is thatof a Gaussian Delta function [7]. The Gaussian Delta pdf has a large probability near zero and almost zero elsewhere.

2. Conventional Learning Algorithm

The term learning or training in this paper will be used to describe the optimization process of the weight vectors. In this section, we briefly review the BP algorithm. One can refer to[2][3] for more detailed descriptions of the algorithm.

In supervised learning, the errors between the actual and estimated time-series are used to optimize the weight vectors. The actualtime-series data is given by $\underline{x}_L$ denotes the output layer with nodes indexed 1 to l.

$$\underline{e}^T = [(x(n+\tau) - y_1, x(n+\tau-1) - y_2(n), \cdots, x(n+\tau-k) - y_k(n), \cdots, x(n+\tau-l) - y_l(n)] \quad (4)$$

where $\underline{e}^T$ is a vector of comprised of the errors, $\tau(\tau \geq 0)$ is the prediction interval, $x(n+\tau-k)$ is actual input data, and $y_k(n)$ is the estimate obtained from the FF neural network. The error $\underline{e}$ at each iteration is a random vector having the joint pdf $P(\underline{x}_P^T, \underline{y}_L)$ [2], where $\underline{x}_P$ is the input vector as

$$\underline{x}_P^T = [(x(n-1)x(n-2)\cdots x(n-p)] \quad (5)$$

The actual data $\underline{x}_L$ is τ steps ahead for the input data $\underline{x}_P$.

An approximation of the unknown transfer functionwhich describes the input-output relation is obtained using a gradient descent procedure in the error space. The objective is to minimize the total squared error ε, where it is defined as

$$\varepsilon = \frac{1}{2}\underline{e}^T\underline{e} \quad (6)$$

The $\underline{e}$ is the error vector having a total of l elements. Each element is obtained at the output node k and time n. Partial differentiation of the ε with respect to the weight matrix yields the gradients of the weight elements. Hence the weight elements are iteratively modified as

$$w_{kj}(t+1) = w_{kj} - \eta\Delta w_{kj}(t) \quad (7)$$

where t is the iteration index, w_{kj} is weight element in the hidden to output layer, η is the learning constant, and $\Delta w_{kj}(t)$ is the gradient. The t will be omitted in the equations that follow.

Since we consider linear sums in the output layer, $\Delta w_{kj}(t)$ withj=1,2,q is derived as

$$\Delta w_{kj} = \frac{\partial\varepsilon}{\partial w_{kj}} \quad (8)$$

Applying the chain rule to Eq.(8) we obtain the solution as

$$\Delta w_{kj} = \frac{\partial\varepsilon}{\partial y_k(n)}\frac{\partial y_k(n)}{\partial w_{kj}} = -2e_k(n)v_j \quad (9)$$

where $y_k(n)$ is the k-th lagged element of $\underline{y}_L$ thus

$$y_k(n) = \sum_{j=1}^{q} w_{kj} \upsilon_j \tag{10}$$

The function $e_k(n)$ is the k-th element of the error vector, and υ_j is the output of the j-th hidden node.

In the hidden layer the sigmoid transfer function is used at each node and the errors in the output layer are passed to the hidden layer. The gradients of weights in the hidden-to-input layer is obtained as

$$\Delta w_{ji} = \frac{\partial \varepsilon}{\partial w_{ji}} \tag{11}$$

Applying the chain rule to Eq.(11), we have

$$\Delta w_{ji} = \frac{\partial \varepsilon}{\partial y_k(n)} \frac{\partial y_k(n)}{\partial \upsilon_j} \frac{\partial \upsilon_j}{\partial P_j} \frac{\partial P_j}{\partial w_{ji}} \tag{12}$$

where P_j is a weighted sum of the input data. The P_j is written as

$$P_j = \sum_{i=1}^{p} w_{ji} x(n-i)$$

Simplifying (12) yields

$$\Delta w_{ji} = -2\delta_j \upsilon_j' x(n-i) \tag{13}$$

with

$$\delta_j = \sum_{k=1}^{l} e_k(n) w_{kj}$$

and

$$\upsilon_j' = \upsilon_j(1-\upsilon_j)$$

where x (n-i) (i=1,2, , p) represents the input data at i-th node in the input layer.

When the amplitudes of input data change smoothly and the time-series is correlated, performance of the FF based neural network with BP training is quite reliable. For highly varying the traces, preprocessing the input data using a smoothing filter can usually improve the neural network performance [6].

III. THRESHOLD NEURAL NETWORKS

The variance of the VBR video data (input data) affects the performance of the neural network model.Large approximation errors arise when xundergoes rapid changes in amplitude. These events can be separately modeled by adding auxiliary models to the baseline model. To this end we will construct two additional output models.

The error signal is used to determine the threshold amplitudes for the VBR video. In training phase, the error signals greater than the prescribed threshold amplitudes can be used to train additional linear models. This scheme involves determining the transition amplitude and modeling the VBR video in the newly defined amplitude region. The structure of the TNN model is given in [Figure 2]. In the figure, two additional models are added to the input layer of the baseline model. Each is comprised of the weight vectors and output nodes. One drawback of the error threshold scheme is that the error values are only detected after one step behind at each observation.

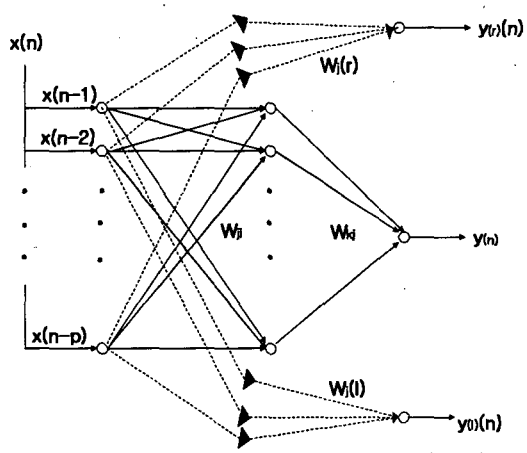

Figure 2. Threshold neural network (TNN) structure

The VBR video modeling procedure based on the TNN model has two steps. In the analysis step all

parameters in the TNN model and ARMA process are estimated from the sampled data. The ARMA process is used to model the residual time-series generated by the TNN model. In the synthesis step a time-series is generated using the TNN model and ARMA process. [Figure 3] illustrates these steps.Each procedure will be discussed in the following sections.

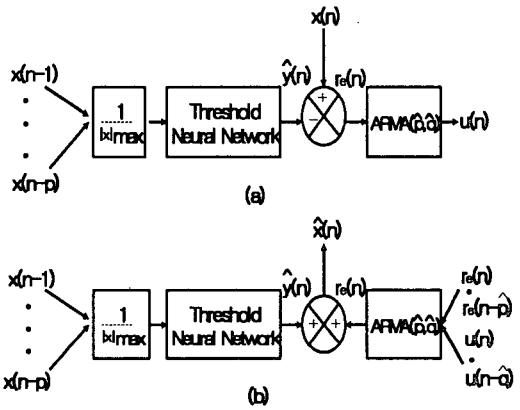

Figure 3. TNN model construction and synthesis steps with ARMA process
(a) input data analysis and
(b) input data synthesis step

1. Analysis Step

VBR video trace point x(n) is estimated using the baseline model in terms of the input vector $\underline{x}_P$. The delay τ will be set equal to 0. The video trace x is scaled such that its values are between 0 to 1 by applying $x = x/|x|_{\max}$. The $|x|_{\max}$ is the largest value in x. The sampled input data is used to train the neural network using the BP algorithm. When the total squared error reaches a stable point, auxiliary linear models are trained using prescribed threshold values. The threshold amplitude α is determined from the residual time-series of the baseline model. The selection criteria is based on the degree of correlation in the error series.

Once the baseline model is stabled, the local error e is tabulated. By finding e(n) which is greater then threshold value α, an auxiliary model yielding $y_{(r)}(n)$ is trained on the input data. Hence we may write it as

$$y_{(r)}(n) = (\underline{w}_P^{(r)})^T \underline{x}_P + \theta_r \qquad (14)$$

where $\underline{w}_P^{(r)}$ is the weight vector between input nodes and an additional output node yielding $y_{(r)}(n)$, and θ_r is a bias constant. The error of this sub-network is calculated as $e_r(n) = x(n) - y_{(r)}(n)$. Calculation of the gradients in terms of the error in the sub-network provides that the weight vector $\underline{w}_P^{(r)}$ is modified to minimize the total squared error as such case of the baseline model. A similar procedure is followed when the e(n) is less than $-\alpha$. In this case, the auxiliary model for yielding $y_{(l)}(n)$ is trained. The estimate is written as

$$y_{(l)}(n) = (\underline{w}_P^{(l)})^T \underline{x}_P + \theta_l \qquad (15)$$

where $\underline{w}_P^{(l)}$ is the weight vector between input nodes and another sub-output node yielding $y_{(l)}(n)$, and θ_l is a bias constant. The error in this case is obtained as $e_l(n) = x(n) - y_{(l)}(n)$.

After training the auxiliary models, the TNN uses the fixed values of the connection weights to calculate an estimate of $\hat{y}(n)$. Hence $y(n), y_{(r)}(n), y_{(l)}(n)$ are computed concurrently. The y(n) is output from the baseline model when k=1 in Eq.(3).

$$\hat{y} = \qquad (16)$$
$$MIN[(y(n), e(n-1)), (y_{(r)}(n), e_r(n-1)), (y_{(l)}(n), e_l(n-1))]$$

where the MIN[] operation takes the output value among y(n), $y_{(r)}(n)$, and $y_{(l)}(n)$ that yield the minimum error.

Since the error at time nis not known, the one-lag behind the error term is used in Eq.(16). This may degrade the overall performance of the TNN model. The residual series of the generalization phase is obtained as

$$r_e(n) = x(n) - \hat{y}(n) \tag{17}$$

where $r_e(n)$ is the residual time-series of the TNN process. This $r_e(n)$ is processed using the linear ARMA model discussed in time series analysis text elsewhere [8]. Hence the $r_e(n)$ yields

$$u(n) = r_e(n) - \sum_{i=1}^{\hat{p}} a_i r_e(n-i) + \sum_{j=1}^{\hat{q}} b_j u(n-j) \tag{18}$$

where u (n)is the time-series generated by the independent random noise process. If u(n) is Gaussian white noise, the sum of the TNN and the additional residual linear process will accurately represent the VBR video trace.

2. Synthesis Step

Using the estimated model, synthesis of the time-series $\hat{x}(n)$ can be carried out. It is a rather simple process because we only need $\hat{y}(n)$ and $r_e(n)$ values which were obtained in the previous section.

$$\hat{x}(n) = \hat{y}(n) + r_e(n) \tag{19}$$

where $r_e(n)$ can be obtained using $ARMA(\hat{p},\hat{q})$ process given in Eq.(18). ARMA solution is omitted in this paper. Interested readers refer [8][9]. The $\hat{y}(n)$ is the estimates from the TNNmodel. In reality the Eq.(19) includes many parameters and functions.

IV. TNN MODEL FOR VIDEO TRACE

To generate VBR video traces, we need to generate intermittent bursts [10]. The following nonlinear difference equation is used to generate intermittent bursts.

$$\chi(n) = \beta + \chi(n-1) + L\chi(n-1)^2 \qquad if \chi(n-1) < D$$

$$\chi(n) = (\chi(n-1) - D)/(1-D) \qquad if D \le \chi(n-1) < 1$$

where $L = (1-\beta-D)/D^2$ and $\beta = 10^{-6}, D = 0.4$, and we obtain the following Eq. as

$$c(n) = \Phi\chi(n) \tag{20}$$

where $\Phi = 15$. The trace c(n) of Eq.(20) is shown in [Figure 4].

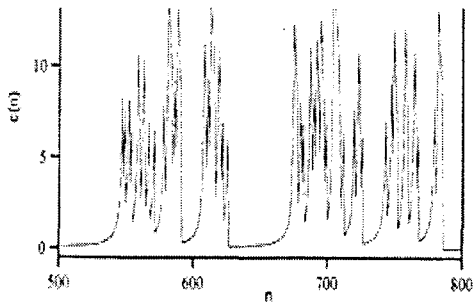

Figure 4: VBR video trace from Eq.(20)

The nonlinear difference equation Eq.(20) will be used in this section to test the modeling procedures discussed in the last section. Now, we need to generate VBR video trace. Superposed on the generated samples from Eq.(20) is random noise drawn from a geometric pdf. The amplitude at time nsatisfies the pdf as

$$P(s) = \gamma^s(1-\gamma)$$

for integral values of $s \ge 1$ and $\gamma < 1$. The parameter $\gamma = 0.8$ is selected to generate a random VBR video trace having a mean equal to 5 and varianceto 20. The video trace is the sum of the output of the nonlinear difference equation and a stochastic time-series as

$$x(n) = c(n) + s(n) \tag{21}$$

where c(n) is given in Eq.(20) and s(n) is a geometrically distributed random variable.

Using the VBR videotrace from Eq.(21), we test TNN modeling procedure. First, we need to determine the baseline architecture. The number of input nodes p and hidden nodes q are free-parameters. To estimate these parameters we proceed as follows. Let Mbe equal to the number of disjoint regions spanned by x(n). The nonlinear intermittent function Eq.(20) is comprised of 22 quantized levels. Following [85], the relation between M, p and q is given by

$$M(p,q)=\sum_{i=0}^{q}\binom{p}{i}$$

where if p=5 and q=3, the value of M is equal to 26. Therefore, p=5 and q=3are chosen for the baseline architecture.

The VBR video trace x(n) is scaled by a scale factor equal to 70. In analysis step the TNN model is trained using the sampled 2000 points. Using the stabled model, $\hat{y}(n)$ is estimated from 6000 points of trace data x(n) which are not include in the time-series points used in the training. The residual series $r_e(n)$ is obtained using Eq.(17).

The best model for the residual series $r_e(n)$ was found as the ARMA(11,0) model which is equal to AR(11) model. The residual series u(n) was obtained and its mean and variance were found to equal 0.001 and 31respectively. The residual of the combined system was revealed as Gaussian white noise. Therefore we can conclude that the TNN with ARMA(11,0) process approximately represents VBR video trace x(n).

The $\hat{x}(n)$ is generated using $\hat{y}(n)$ and $r_e(n)$ for the TNN and ARMA(11,0) process in the synthesis step. For comparison purpose, the synthesized series was also constructed using the FF neural network and the TNN model. The estimated output of these three models are denoted as $\hat{x}_{BP}(n)$, $\hat{x}_{TNN}(n)$ and $\hat{x}(n)$. [Table 1] shows statistics for the un-scaled $x(n)$, $\hat{x}(n)$, $\hat{x}_{TNN}(n)$ and $\hat{x}_{BP}(n)$. The statistics are matched reasonably well in the case of $\hat{x}(n)$. As clearly shown in table, $\hat{x}_{BP}(n)$ estimates the mean level. However the variance is underestimated. This indicates that the time variation in $x(n)$ is not captured by the FF neural network model with BP training. The $\hat{x}_{TNN}(n)$ shows a better agreement than the $\hat{x}_{BP}(n)$ but the variance is still low. The maximum (max.) and minimum (min.) values of the time-series are also shown in the table.

Table 1. Statistical results for each model

Models	Mean	Variance	Max.	Min.
$x(n)$	8.32	37.63	47.54	1.00
$\hat{x}_{BP}(n)$	8.06	4.65	15.37	3.96
$\hat{x}_{TNN}(n)$	8.32	12.11	33.31	1.20
$\hat{x}(n)$	9.15	30.72	39.42	0.01

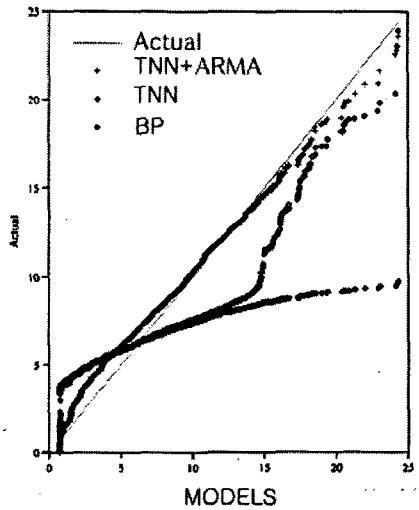

Figure 5 Q-Q Plot comparison for each model

The Q-Q plot for each model is shown in [Figure 5]. In [Figure 5] the strict line shows actual model, $x(n)$. The TNN+ARMA, TNN, and BP represent $\hat{x}(n)$,

$\hat{x}_{TNN}(n)$, and $\hat{x}_{BP}(n)$ respectively. The $\hat{x}_{BP}(n)$ deviates from the actual mode, $x(n)$. The $\hat{x}(n)$ follows the actual $x(n)$ reasonably compared to the other models. In conclusion, the FF neural network with BP training is not a reliable modeling paradigm for the stochastic trace modeled in this chapter. The TNN with ARMA process outperforms the FF neural network with BP training in modeling the VBR video x.

V. CONCULSION

The conventional neural network model is not adequate to model VBR video trace possessing sudden changes in amplitude. If changes occur at random intervals in time theestimates of the FF neural network with BP training wander about the near mean level of the trace.

We have developed additional sub-network structures based on the residuals obtained in the training phase of the FF neural network. This was required to increase the accuracy of the model. Once the values of the weights have been stabled, no further minimization in the total squared error of the conventional neural model was observed. By developing the auxiliary models for the values of x(n) which are outside of the prescribed amplitude range $\pm\alpha$, the error was reduced.

The residual series of the TNN model was analyzed. It was found that using the ARMA process to model the residual series one could further remove the linear correlations in the residualseries. Synthesis of the series using the TNN and ARMA process yielded a close agreement to statistics of the generated VBR video trace.

With these results, we can conclude that VBR video trace cannot be easily modeled using traditional stationary probabilistic methods. This is due to the sudden changes and burst characteristics inherent to VBR video traces.

참 고 문 헌

[1] A. S. Weigend and N. A. Gershenfeld, Eds., *Time Series Prediction Forecasting The Future and Understanding The Past*, Addison Wesley, 1993.

[2] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice Hall, 1998.

[3] Z. M. Zurada, *Introduction to Artificial Neural Systems*, West, 1992.

[4] E. Liu, L. Cuthbert, J. Schormans, and G. Stoneley. "Neural Network in Fast Simulation Modeling,"IEEE-INNS-ENNS International Joint Conf. on Neural Networks (IJCNN'00) -Vol.6, pp.6109-6113, 2000.

[5] A. A. Tarraf, I. W. Habib, and T. N. Saadawi, "A novel neural network traffic enforcement mechanism for ATM networks", IEEE J. on Selected Areas in Comm., Vol.12, No.6, pp. 1088-1096, Aug, 1994.

[6] S. Chong, S. Li, and J. Ghosh, "Predictive dynamic bandwidth allocation for efficient transport of real-time VBR video over ATM", IEEE J. on Selected Areas in Comm., Vol.13, No.1, pp.12-23, Jan, 1995.

[7] J. W. Woods and H. Stark, *Probability and Random Process with Application to Signal Processing*, Prentice Hall, 2001.

[8] W. S. Wei, *Time Series Analysis*, Addison Wesley, 2005.

[9] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, *Time Series Analysis Forecasting and Control*, 3rd Ed., Prentice Hall, 1994.

[10] K. C. Chandra, C. Thompson, K. S. Meier-Hellstern, and P. E. Wirth, "Source model

for ISDN packet data traffic", Proc. of IEEE IT Workshop on Information Theory, Multiple Access, and Queueing, pp.30-34, 1995.

저 자 소 개

장 봉 석(Bong-Seog Jang) 종신회원

- 1989년 : 조선대학교 전산통계학과(학사)
- 1993년 : 메사츄세스 주립대, 전산학과(석사)
- 1997년 : 메사츄세스 주립대, 전산학과(박사)
- 1997년~1999년 : 삼성전자 통신연구소 선임연구원
- 1999년~현재 : 목포대학교 멀티미디어공학전공 조교수

<관심분야> : 멀티미디어통신, 3D Digital Contents