International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 4 (2012) pp. 236 -243
© MIR Labs, www.mirlabs.net/ijcisim/index.html

Chaotic Time Series Prediction Using Combination of Hidden Markov Model and Neural Nets

Saurabh Bhardwaj[1], Smriti Srivastava[2], J.R.P Gupta[3] and Ashish Srivastava[4]

[1,2,3] Netaji Subhas Institute Of Technology, Delhi University,
New Delhi, India, 110078
[1] *bsaurabh2078@gmail.com* , [2]*ssmriti@yahoo.com* ,[3] *jrpg@gmail.com*

[4] XLRI School of Business and Human Resources
Jamshedpur, India, 831 035
s.ashish1@gmail.com

***Abstract*—This paper introduces a novel method for the prediction of chaotic time series using a combination of Hidden Markov Model (HMM) and Neural Network (NN). In this paper, an algorithm is proposed wherein an HMM, which is a doubly embedded stochastic process, is used for the shape based clustering of data. These data clusters are trained individually with Neural Network. The novel prediction approach used here exploits the Pattern Identification prowess of the HMM for cluster selection and uses the NN associated with each cluster to predict the output of the system. The effectiveness of the method is evaluated by using the benchmark chaotic time series: Mackey Glass Time Series (MGTS). Simulation results show that the given method provides a better prediction performance in comparison to previous methods.**

***Keywords*—Hidden Markov Models, Neural Networks, Time series prediction**

I. Introduction

Most of the frequently encountered data in nature form chaotic time series. A very well-known and frequently encountered chaotic time series is the Mackey-Glass time series. Examples of great interest include psychiatric study data, weather data and stock market indices. Hence, the need arises for a robust and dependable method which can accurately analyze and predict future values for any such time series. The Mackey-Glass time series is a chaotic time series that is frequently encountered in real-world applications, ranging from the production of red blood cells and phenomena for glucose metabolism, to stock market indices. Hence, being able to analyze the behavior and thereby predict future values for such a series would be highly beneficial. For modeling a chaotic system, the traditional method is to use a stochastic process, since it is a well-established fact that their behaviors have many similarities, and that there exists an equivalence measure between a chaotic system and a stochastic system [1]. For prediction, a Hidden Markov Model that has excellent pattern-matching ability has been chosen. Our paper uses a novel approach to combine the modeling capabilities of an ANNs (Artificial Neural Networks) with the pattern-matching abilities of HMM. Earlier papers have created hybrid models of HMMs and ANNs/RNNs -cases in point would be [2] and [3]. However, in those papers, the neural network has been used for refining the parameters of the HMM. In this paper we present a method whereby the final output comes from trained Artificial Neural Networks. Our Work: The earlier papers which have used an HMM-NN hybrid have typically used the neural network to iteratively optimize the α, β and π values for the HMM. We propose an algorithm whereby we create one large HMM using the training data. The training data set is then divided into vectors of 5 values each. The first 4 values of the vector are treated as the input, and the fifth value is treated as the output. Depending on the log-likelihood values of the input (as obtained from the HMM), the training data is divided into clusters. For each cluster, a feedforward ANN is created. Training of the ANN is done using the vectors that fall into that particular cluster, with the ANN being required to predict the fifth value of each vector using the first four as its inputs. Thus, while the modeling still remains stochastic, the prediction is made deterministic to quite an extent. For prediction using test data, the same process is followed. The 4 elements of the test data are taken, grouped into a vector, and the log-likelihood of that vector calculated. According to the log-likelihood, the appropriate cluster and hence the corresponding neural network is chosen. The output is then obtained from that trained network. The organization of the paper is as follows -section II presents an overview about the Mackey-Glass time series, and the use of HMMs in its analysis; sections III and IV discuss the modeling and prediction algorithms along with the respective flowcharts; section V presents our results and draws a comparison with the existing methods.

II. Overview

2.1The Mackey-Glass Time Series:

The Mackey-Glass time series is a non-linear time delay differential system as shown in (1)

$$\frac{dx}{dt} = \beta \frac{x_\tau}{1+x_\tau^n} - \gamma x \qquad (1)$$

Where β, γ, τ and n are real numbers and x_τ represents the value of the variable x at time (t-τ). Depending on the values of these parameters, the series displays a sequence of period-doubling bifurcations and chaotic dynamics [4]. It is seen that for $\tau > 17$, the chaoticity in the resulting system becomes highly noticeable.

This Mackey Glass Time Series is widely regarded as a benchmark for comparing the generalization abilities of various methods.

2.2 Modeling Chaotic Systems

The apparently random behaviors of chaotic dynamical systems and stochastic systems are similar in the fact that both are highly dependent on initial conditions. This seems to suggest that the methodology used for analyzing stochastic systems could also be used to study chaotic dynamical systems. An equivalence relationship between chaotic and stochastic systems, as given in [1], is that the systems are defined as equivalent if they both share an ergodic invariant measure. Hence, stochastic models are well-suited for analyzing chaotic systems. Thus, it makes sense to use a stochastic process, and even more sense to use a doubly-embedded stochastic process like an HMM, for the analysis of a chaotic system like the Mackey-Glass time series.

2.3 Hidden Markov Models

The Hidden Markov Model (HMM) is an extension of Markov Processes or Markov Chains. An observable Markov Process has been defined as a stochastic model where the state of the system at a particular instant of time corresponds to an observable change or output. Hidden Markov Model is a statistical method that uses probability measures to model sequential data represented by sequence of observation vectors. To understand the working of a Hidden Markov Model (HMM), consider the following example. There are three bags full of candy, say M&Ms. Each of the candy can be blue, green, orange, red, yellow or brown in color. A monkey picks (uniformly at random) any one of the three bags, and from that bag, picks out one M&M (again uniformly at random). This M&M is then shown to a human observer. The monkey has been trained to replace the M&M taken out by him in the very same bag from which he picked it out. He dutifully puts the M&M back in the corresponding bag, and repeats the process again, by picking another bag and taking out another M&M. The human observer therefore has a sequence of M&M colors noted down (only till the monkey starts eating the M&Ms he’s taking out of the bags, but for argument’s sake let us assume that never happens).
The observation sequence formed by the colors noted down by the human may be modeled to be the output of an HMM, where the three bags may be thought of as states, and the six possible colors from each bag may be thought of as discrete outputs. The initial probability with which the monkey picks out the first bag in the sequence is given by π. The state transition matrix α denotes the probability with which he moves from one bag to another. For each bag, there is an observation matrix β which tells us the probability of each of the six colors appearing from that particular bag. Before addressing the major design problems of the HMM, the basic notations used are explained below.
There is a finite number of states ‘n’ in the model. At each time step, a new state is entered based upon a transition probability distribution which depends on the previous state .
After each transition is made, an observation output symbol is produced according to a probability distribution, which depends on the current state. There are two types of HMMs depending upon the data; Discrete HMMs and Continuous Density HMMs. These are defined by the type of data that they operate upon. Discrete HMMs (DHMMs) operate on quantized data or symbols. On the other hand, the Continuous Density HMMs (CDHMMs) operate on Continuous data and their emission matrices are not matrices but are distribution functions. Some of the common distribution functions used for the emission matrices for each state are Gaussian, Poisson or a mixture of Gaussians. Since the data used here is continuous, CDHMMs have been used with a single Gaussian function for each state. The basic notations for first order discrete HMM are as follows.

$\{O_1,O_2,O_3,.\ .\ .\ .\ .\ ,O_n\}$	: Observation Sequence
$\lambda(A, B, \pi)$	: Model of the System
A	: Transition Matrix
B	: Emission Matrix/Function
π	: Initialization Matrix
$Q\ \{q_1, q_2, ..\ .\ .\ .\ .\ ,q_n\}$	: State Sequence

If the Observations are continuous then discrete HMM cannot be directly applied on the data. For that quantization of data is required and this is usually done by Vector Quantization but vector quantization of these continuous signals can degrade the performance significantly if discrete probability density is used. Therefore, in the context of HMMs, continuous observation densities are used by placing some restrictions in the form of the probability density function (pdf), therefore, to ensure that the parameters of the pdf can be re-estimated in a consistent way. The most general representation of the pdf for which the re-estimation equations are applicable is the Gaussian density function.

There are three major design problems that are associated with an HMM. They are mentioned as follows:

1) Given the Observation Sequence {O1, O2, O3,. . . ,On} and the Model λ(A,B,π), computation of the probability of the observation sequence P (O|λ).

2) Given the Observation Sequence {O1,O2,O3,. . . . ,On} and the Model λ(A, B, π), computation of the State Sequence Q {q1, q2, ... ,qn}, that is optimal or most probable.

3) Tweaking of the Model parameters λ(A, B, π), such that the Probability of the Observation sequence, P(O|λ) is maximum.

2.3.1 The Forward Algorithm

The forward algorithm [5] is the solution to the first problem of the HMM. Here a forward variable $\alpha_t(i)$, is the probability of the partial observation sequence to time t, given at the state i and time t, given the model λ(A, B, π), is defined as in (2)

$$\alpha_t(i) = P(O_1, O_2, \ldots O_t,\ q_t= S_i|\lambda) \quad (2)$$

The forward variable maybe computed with the help of (3), (4) and (5).

Initialization: $$\alpha_1(i) = \pi_i B_i(O_1), \quad 1 \le i \le N \quad (3)$$

Induction: $$\alpha_{t+1}(i) = \left[\sum_{i=1}^{N} \alpha_t(i) A_{ij}\right] B_j(O_j) \quad (4)$$
$$1 \le t \le T-1, 1 \le j \le N$$

Termination: $$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \quad (5)$$

The solution to problem 2 is given by the Viterbi Algorithm [5], [6] and the solution to Problem 3 is given by the Baum Welch/ Expectation Maximization Algorithm [5],[6],[7],[8] Once we have the solutions to the above problems, the tools to shape based clustering of HMMs are available.

III. Our Algorithm For Modeling Of The Time Series

Fig. 1 shows the flowchart for our algorithm used to model and thereby analyze the time series.
The input test data is a one-dimensional continuous series:

$$X = \{x_t, t = 1,2 \ldots.., N\} \quad (6)$$

Where t is the time index and N is the total number of observations. For the shape based clustering process pre-processing of the time series is required. The basic steps for the preprocessing of time series data are as follows. As a first step the time series is converted into phase space to get the shape based clustering. The phase space of a time series is generated by using time delay embedding and embedding dimension. The conversion of time series into phase space and the time delay embedding is based upon “Takens Theorem”. If the value of time delay factor is τ and the embedding dimension is b, then the equation 6 in the phase space can be represented as shown in (7).

$$X_t = \left(x_{t-(b-1)\tau}, x_{t-(b-2)\tau}, \ldots \ldots \ldots, x_t\right) \quad (7)$$

Where X is a matrix whose row vector is a point in the phase space. This transformation preserves the nonlinear dynamics of the original time series. This series though one-dimensional, is dependent on past values and the series is modeled in terms of predictor and dependent variables.

If the value of τ = 4 and b = 4, then the phase space can be represented as

$$X_t = (x_{t-12}, x_{t-8}, x_{t-4}, x_t) \quad (8)$$

In SBBP approach the time series is examined in phase space and the method is based upon the time delay embedding τ and embedding dimension b. For constituting the required phase space, time delay embedding τ and embedding dimension b are the important parameters. In Table 1 it is important to note that each row of the table is a single point in a phase space.
The datasets hence created are used in the shape based clustering process. The performance criterion for similarities in shape for input data used here is known as Log – Likelihood. Log-Likelihood is defined as the Log of the value of the Probability of observation sequence given the model; Log (P (O|λ)).

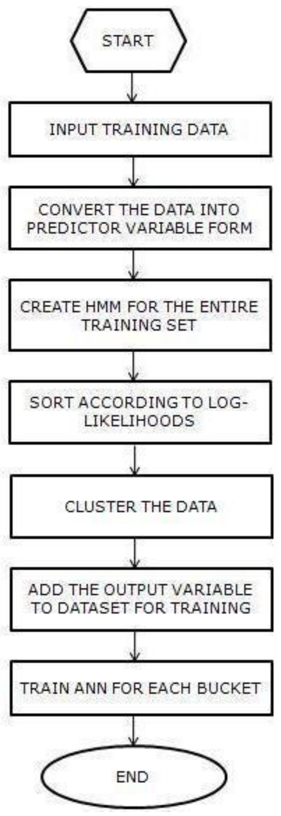
START
INPUT TRAINING DATA
CONVERT THE DATA INTO PREDICTOR VARIABLE FORM
CREATE HMM FOR THE ENTIRE TRAINING SET
SORT ACCORDING TO LOG-LIKELIHOODS
CLUSTER THE DATA
ADD THE OUTPUT VARIABLE TO DATASET FOR TRAINING
TRAIN ANN FOR EACH BUCKET
END

Fig.1. Flow Chart: Our Algorithm for modeling of time series

3.1 Training Data Input

The input test data is a one-dimensional continuous series which cannot be used as such unless it is converted into Usable/Classical dataset where shape based clustering can be done. This series though one-dimensional, is dependent on past values and the series is modeled in terms of predictor and dependent variables.
The choices of embedding dimension b and time delay τ were always based on user's experience with trial-and-error. Here, the time lag τ is determined using first minimum of mutual information technique or autocorrelation function and the embedding dimension b can be estimated using false nearest neighbor method.

A. Estimation of Time Delay Embedding

As has been explained before, for embedding dimension, we need to calculate the value of time delay τ for the required phase space. If the value of τ that we calculate is too small, the successive elements of the delay vector would end up being highly correlated. The two methods used for the estimation of τ are based on autocorrelation and mutual information respectively. In the first method of autocorrelation, for a time series given in (6), the time delay is estimated by calculating autocorrelation between X_t and $X_{t-\tau}$ as shown in (9).

$$C(X_t, X_{t-\tau}) = \frac{E(X_t, X_{t-\tau}) - E(X_t)^2}{E\{[X_t - E(X_t)]^2\}} \qquad (9)$$

If we plot values the values of X_t versus the corresponding values a fixed time lag τ, the autocorrelation C quantifies how these points are distributed. If they spread out evenly over the plane, then C is zero. If they tend to crowd along the diagonal, then C becomes larger than zero. The value of τ can be estimated at time instance when the autocorrelation function first crosses zero as the time delay.

B. Estimation of the Embedding Dimension

Now comes the part of the actual estimation of the embedding dimension. One way to estimate the embedding dimension "b" is using the nearest neighbor method. This implies that a b-dimensional phase space will have the same topological properties as the original phase space . A phase space is a b dimensional metric space into which a time series is unfolded. Taken's Theorem provides the theoretical justification for reconstructing state spaces using time-delay embedding. His Theorem proved that the state space of an unknown system can be reconstructed. If the embedding is performed correctly. the theorem guarantees that the reconstructed dynamics are topologically identical to the true dynamics of the system. The algorithm of finding false nearest neighbor is described as follows: For each data point we find the nearest neighbor in the b dimensional space which is given by the Euclidean distance between the two points. If the embedding dimension is changed from b to b+m, the Euclidean distance between the two points is calculated by (10)

$$R_i = \frac{|x_{i+1} - x_{j+1}|}{|x_i - x_j|} \qquad (10)$$

If R_i exceed a given threshold v, then b, X_i is marked as having a false nearest neighbor. Therefore, the embedding dimension is decided as high enough if the fraction of data points for which R_i >v is zero (or very close to zero).
Now after selecting the appropriate value of time delay embedding and embedding dimension the data is converted into the phase space form as shown in Fig. 2.

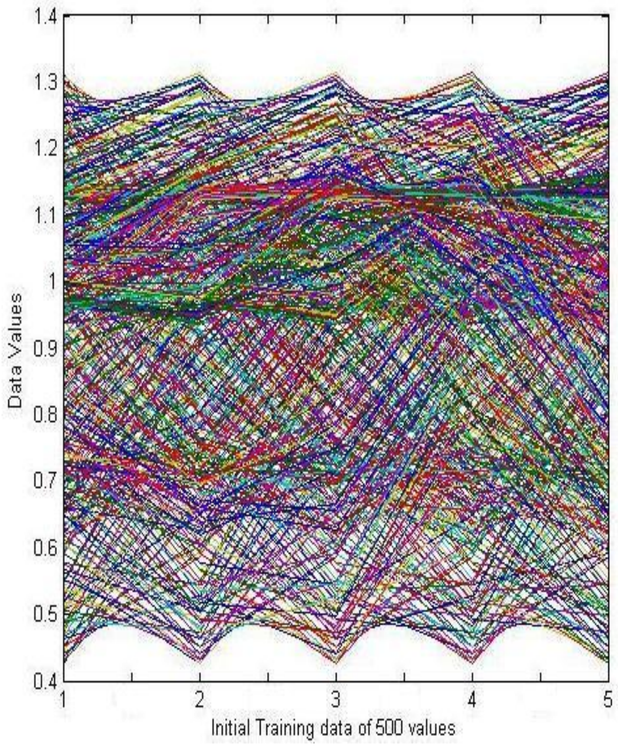

Fig. 2. Whole Training Data

It can be seen from the figures that the structure of this data is very complex and to train this data is a tedious task. Hence the data is divided into shape based batches/ clusters to make training easier.

The method for data representation is as follows.

Input: O (t-12) O (t-8) O (t-4) O (t)
Output: O (t+4)

Each dataset consists of five elements, four input elements which are known as predictor variables, and one output element known as the dependent variable. Table I shows the one dimensional vs. classical form representation of the whole data set.

TABLE I
ONE DIMENSIONAL VS. CLASSICAL FORM REPRESENTATION

$O_1, O_2, O_3, \ldots\ldots\ldots\ldots, O_{T-3}, O_{T-2}, O_{T-1}, O_T$

DATASET	INPUT	OUTPUT
1	O_1,O_5,O_9,O_{13}	O_{17}
2	O_2,O_6,O_{10},O_{14}	O_{18}
3	O_3,O_7,O_{11},O_{15}	O_{19}
4	O_4,O_8,O_{12},O_{16}	O_{20}

The datasets hence created are used in the shape based clustering process.

3.2 HMM Creation, Log-Likelihood & Clustering

An HMM is created for the training data. From this HMM, the log-likelihoods of each of the training vectors is calculated. The log-likelihoods are then sorted, from minimum to maximum. Depending on the difference between minimum and maximum log-likelihood, a fixed log-likelihood range per cluster is set, and the clusters assigned to each such interval, going from minimum to maximum, much like in [10],[11]. This gives us a certain number of clusters (although some of these clusters could be empty). Now, depending on the log-likelihood values of each of the training vectors, they are put into one of these clusters. Empty clusters are then deleted. By looking closely at the input datasets, it can be easily observed that data points which have similar values of Log-Likelihood have similar shape. As in [11], it can be easily observed that after a certain value of Log-Likelihood, the shape of the input changes and hence cannot be classified into the same cluster. For the Mackey Glass time series this value was observed at being somewhere around 30. Another important observation was that, if the number of observations for each cluster were above a fixed value, then the training of the NN for each cluster would not yield precise results. This value was set to 35. Using these basic criterions, an algorithm was developed which arranged the data elements into Shape Based clusters. The maximum size of the clusters was 35 and the maximum Log-Likelihood difference between two consecutive clusters was 30. Fig..3 shows the data before and after Shape Based Batching One important thing to observe here is that the number of batches is not fixed, and the SBB algorithm automatically decides the number of batches required.

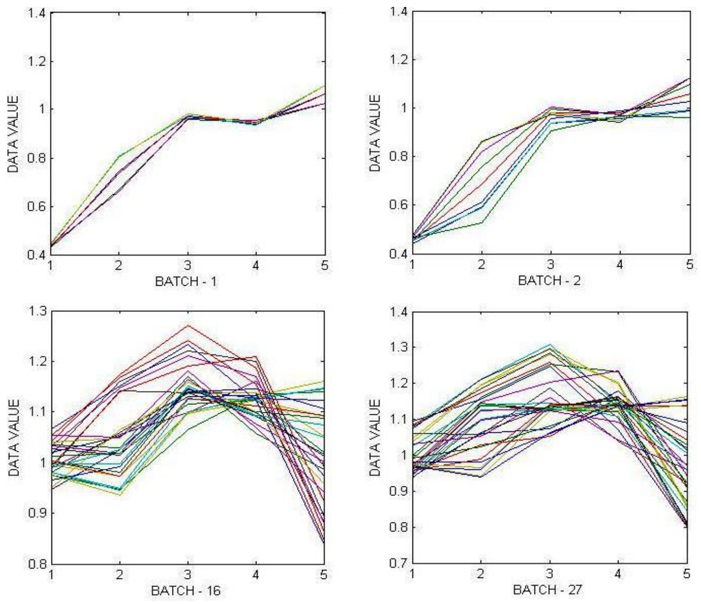

Fig. 3 Shape Based Batches / Clusters of data

Pseudo Code for shape based Batching / clusters is as shown in figure 4

```
Max LogLik Range = Φ
Max Data Range = Θ
StartLL = Minimum of Log Lik Values
StartData = 1
EndLL = Maximum of Log Lik Values
EndData = Last Data element
j=1
For (LogLik within (StartLL and StartLL+Φ) && Data within (StartData and StartData+Θ))
    Batch (j).Data=Accepted Data
    Batch (j).Range=Accepted Range
    StartLL=StartLL+Accepted Range
    StartData=StartData+Accepted Data
    j+=1
end
```

Fig.4. Pseudo-code of Shape Based Batching Process

3.3 Neural Network training

Each cluster is then assigned a neural network Hence the number of neural network used are equal to the number of clusters. The four values of each training vector are treated as inputs for the network, while the fifth ‘output’ sample is used to iteratively improve the weights of the network, such that the actual output of the network for the input vector matches the fifth sample The network used is a two layer feedforward network, with back propagation being used for updating the weights.

3.3 Neural Network training

Each cluster is then assigned a neural network Hence the number of neural network used are equal to the number of clusters. The four values of each training vector are treated as inputs for the network, while the fifth 'output' sample is used to iteratively improve the weights of the network, such that the actual output of the network for the input vector matches the fifth sample The network used is a two layer feedforward network, with back propagation being used for updating the weights.

III. Our Algorithm For The Prediction Of Time Series

Fig. 5 shows the flowchart for our algorithm used to predict future values of the time series.

For the process of prediction the test data is also grouped into usable/ classical form. The row vectors of input test datasets are assumed to be observation sequences for the Hidden Markov Model that was previously trained during the shape based batching process Using problem 1 we find out the value of Log-Likelihood for the set of predictor variables and hence determine what its shape is. Using the value of the Log-Likelihood, we can assign a cluster to the dataset and use it as an input to the Neural Network of that cluster. The output of the Neural Network of that cluster is the predicted output/dependent variable of the system. This predicted output is compared with other values and a error is drawn.

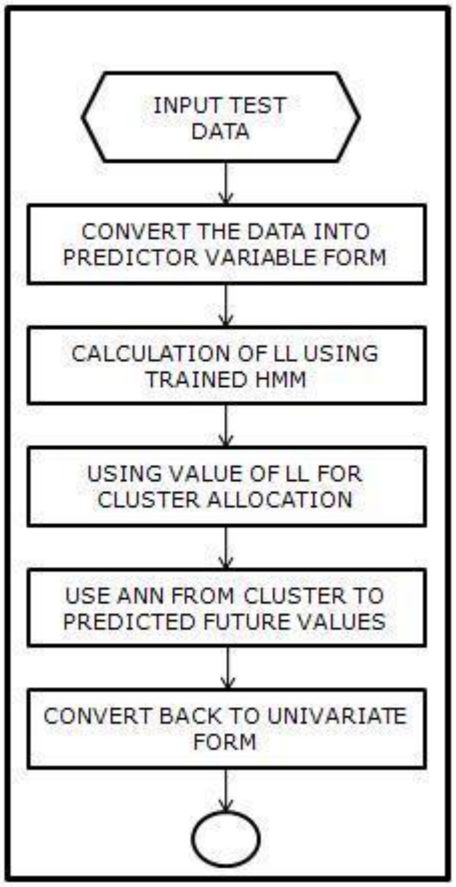
INPUT TEST DATA
CONVERT THE DATA INTO PREDICTOR VARIABLE FORM
CALCULATION OF LL USING TRAINED HMM
USING VALUE OF LL FOR CLUSTER ALLOCATION
USE ANN FROM CLUSTER TO PREDICTED FUTURE VALUES
CONVERT BACK TO UNIVARIATE FORM

Fig.5. Flowchart: Our algorithm for prediction of time series

IV. Experimental Results & Conclusion

The approach was applied to the Benchmark Data of the Mackey Glass Time Series. Here we have taken first 500 data points as training samples and next 500 data points as test samples to show the efficiency of the method.

The test-data results of the predicted values are compared with the actual values and error plot for the learning and test data points are as shown in the Fig.6 and Fig.7 respectively. We have compared our results with other methods. For Time series prediction that have been performed in the past. In case of ANN the number of batches formed was 16 while keeping the log-likelihood difference in each batch 30 and the size of each batch was 35.

Table II shows the root mean square error comparison method of various approaches and the results reported here are very encouraging

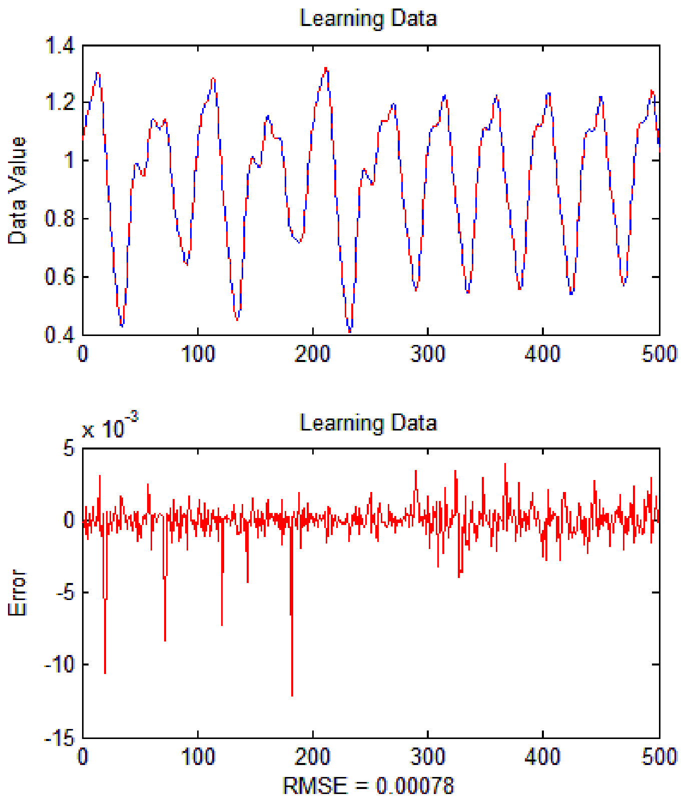

Fig.6 learning data prediction

TABLE II
COMPARISON OF PREVIOUS METHODS

Learning algorithm	**Results**
Liner Predictive Model , [12]	0.55
Auto Regressive Model , [12]	0.19
Wang and Mendel , [12]	0.091
Cascade Correlation N N , [13]	0.06
6th Order Polynomial , [13]	0.04
Kim and Kim , [13]	0.026
EPNet , [14]	0.02
DENFIS (Offline) , [15]	0.016
Data-Driven Linguistic Modeling Using Relational Fuzzy Rules , [17]	0.009
ANFIS , [17]	0.007
GEFREX , [18]	0.0061
Md. Rafiul Hassan[10]	0.0055
HMMSBB+FIS[11]	0.0018
Hidden Markov Model + Neural Nets	**0.0017**

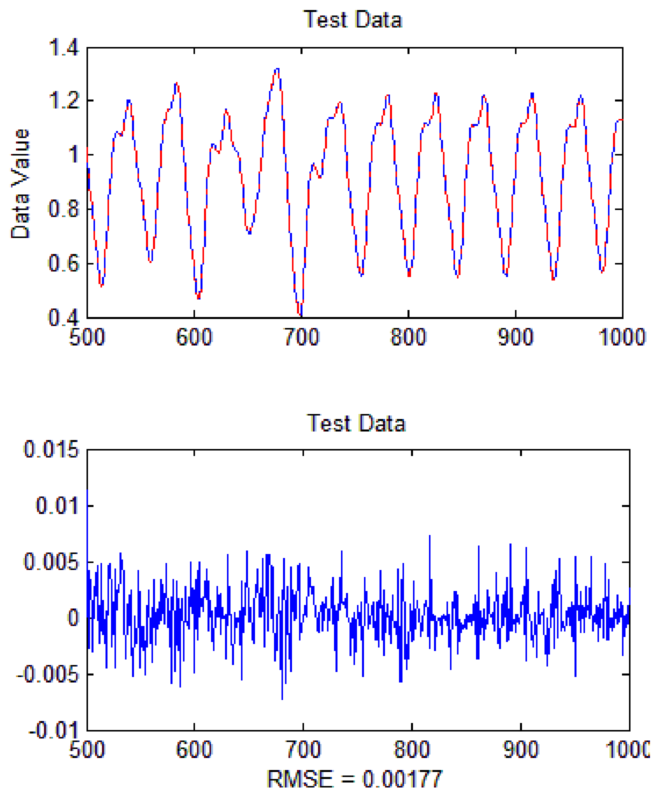

Fig.7. Test Data Prediction

REFERENCES

[1] Q. H. Wu, Y. J. Cao, "An equivalent stochastic system model for control of chaotic dynamics," The *34th IEEE Conference on Decision and Control*, 3, 2898-2903, 1995.

[2] Yoshua Bengio, Renato De Mori, Giovanni Flammia, Ralf Kompe, "Global Optimization of a Neural Network -Hidden Markov Model Hybrid," *IEEE Transactions on Neural Networks*, vol. 3, no. 2, pp. 252–259, 1992.

[3] Rohitash Chandra, Christian W. Omlin, "Evolutionary Training of Hybrid Systems of Recurrent Neural Networks and Hidden Markov Models," World Academy of Science, Engineering and Technology, 15, 58-63, 2006.

[4] Mackey, M., and Glass, "Oscillation and Chaos in Physiological Control Systems", *Science*, Vol. 197, pp. 287-289, 1977.

[5] A.B. Poritz, "Hidden Markov models: a guided tour," *in Proc. of ICASSP*, pp. 7-13, 1988.

[6] I.L. MacDonald and W. Zucchini, "Hidden Markov and Other Models for Discrete-Valued Time Series," Chapman and Hall, 1997.

[7] A.P.Dempster, N.M. Laird, and D.B. Rubin, "Maximum-likelihood from incomplete data via the em algorithm," *J. Royal Statist. Soc*. Ser. B, 39, 1977.

[8] Jeff A Blimes, "A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models," International Computer Science Institute, Berkeley, California. April 1998.

[9] Cory Myers, Steven Kay, and Michael Richard, "Signal separation for nonlinear dynamical systems," *Proc. of ICASSP*, 4:129-132, 1992.

[10] Md. Rafiul Hassan, Baikunth Nath and Michael Kirley, "HMM based Fuzzy Model for Time Series Prediction", in *Proc. of the IEEE International Conference on Fuzzy Systems*, Canada 2006 , pp.2120-2126, 2006.

[11] Smriti Srivastava, Saurabh Bhardwaj, Advait Madhvan and J.R.P Gupta, "A Novel Shape Based Batching and prediction approach for Time Series using HMMs and FISs," communicated in 10th *International conf. on Intelligent systems design and applications* will be held on Dec. 2010, at Cairo, Egypt.

[12] D. Kim and C. Kim, "Forecasting time series with genetic fuzzy predictor ensemble," *IEEE Trans. Fuzzy Syst.*, vol. 5, pp. 523–535, Nov.1991.

[13] L.-X. Wang and J. M. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, pp. 1414–1427, Nov. 1992.

[14] X.Yao and Y.Lin, "A new evolutionary system for evolving artificial neural networks," *IEEE transactions on neural networks*, vol. 8, pp. 694-713, 1997.

[15] N. K. Kasbov and Q.Song, "DENFIS: Dynamic Evolving Neural-Fuzzy Inference System and its Application for Time Series prediction," *IEEE transactions on Fuzzy Systems*, vol. 10, pp. 144-154, 2002.

[16] A.E.Gaweda and J.M.Zurada, "Data-driven linguistic modeling using relational fuzzy rules," *IEEE transactions on fuzzy systems*, vol. 11 (1), pp. 121-134, 2003.

[17] J.S.R Jang, "ANFIS: Adaptive-Network-Based Fuzzy Inference System," *IEEE Trans. Syst., Man, Cybern*, Vol. 23, pp. 51-63, 1993.

[18] M. Russo, "Genetic Fuzzy Learning," *IEEE transactions on Evolutionary Computation*, vol. 4, pp. 259-273, 2003.

Author Biographies

Saurabh Bhardwaj was born on 20 January 1978 at Meerut, India. He received his B.E (Electronics & Instrumentation) degree in 2001 from Meerut Institute of Engineering & technology, Meerut and M.tech (Instrumentation) in 2008 from Panjab University , Chandigarh . He is currently pursuing the PhD degree in Instrumentation and Control from University of Delhi. His research interest includes Hidden Markov Model, Fuzzy Logic , Gaussian Mixture Model.

Smriti Srivastava received the B.E. degree in electrical engineering and the M.Tech. degree in heavy electrical equipment from Maulana Azad College of Technology [now Maulana Azad National Institute of Technology (MANIT)], Bhopal, India, in 1987 and1991, respectively, and the Ph.D. degree in intelligent control from the Indian Institute of Technology, Delhi, India, in 2005. From 1988 to 1991, she was on the faculty of MANIT, and since August 1991, she has been with the Department of Instrumentation and Control Engineering, Netaji Subhas Institute of Technology, Delhi University, New Delhi, India, where she is currently the Associate Head of the department and holds the rank of Professor. She has a number of publications in journals and conferences in the area of neural networks, fuzzy logic, and control systems. She has given a number of invited talks and tutorials mostly in the area of fuzzy logic, process control, and neural networks. Her current research interests include neural networks, fuzzy logic, and hybrid methods in modeling, identification, and control of nonlinear systems.

J.R.P Gupta was born at Lahladpur , patna in India on 22nd july 1948. He received his B.Sc(Engg) in electrical engineering from MIT Muzaffarpur in 1972 and PhD from University of Bihar in 1983. He was Assistant Professor at MIT Muzaffarpur for more than a decade. He than switch over to RIT Jamshedpur where he served for about 7 years . He joined Netaji Subhas Institute of Technology in 1994 as professor of Instrumentation and control engineering. He is presently professor and head of Instrumentation and control engineering department, University of Delhi. He has guided 8 PhD and many Master thesis. He has more than 100 publications in reputed journals and conferences. He is senior member of IEEE and recipient of IETE India best paper award. His area of research includes power electronics , control systems and biomedical instrumentation.

Ashish Srivastava : Ashish Srivastava is currently a post graduate student at XLRI, Jamshedpur. He has a Bachelor in Engineering from Netaji Subhas Institute of Technology, University of Delhi.He is interested in market mix modeling, hidden markov models and neural network.