01 Jan 2004

Time Series Prediction with a Weighted Bidirectional Multi-Stream Extended Kalman Filter

Donald C. Wunsch
Missouri University of Science and Technology, dwunsch@mst.edu

Xiao Hu

Time Series Prediction with a Weighted Bidirectional Multi-stream Extended Kalman Filter

Xiao Hu

Applied Computational Intelligence Lab
Department of Electrical & Computer Engineering
University of Missouri-Rolla
Rolla, MO 65409 USA
Email: xhu@umr.edu

Donald C. Wunsch II

Applied Computational Intelligence Lab
Department of Electrical & Computer Engineering
University of Missouri-Rolla
Rolla, MO 65409 USA
Email: dwunsch@umr.edu

Abstract-**This paper describes the use of a multi-stream Extended Kalman Filter (EKF) to tackle the IJCNN 2004 challenge problem – Time Series Prediction on CATS benchmark. A weighted bidirectional approach was adopted in the experiments to incorporate the forward and backward predictions of the time series.**

I. INTRODUCTION

The goal of this competition is to provide a new benchmark for the problem of time series prediction. The proposed time series is the CATS benchmark (for Competition on Artificial Time Series). The artificial time series with 5000 data points is given. Within those, 100 values are missing. These missing values are divided in 5 blocks: 981-1000, 1981-2000, 2981-3000, 3981-4000 and 4981-5000, as indicated by the 5 circled regions in Figure 1.

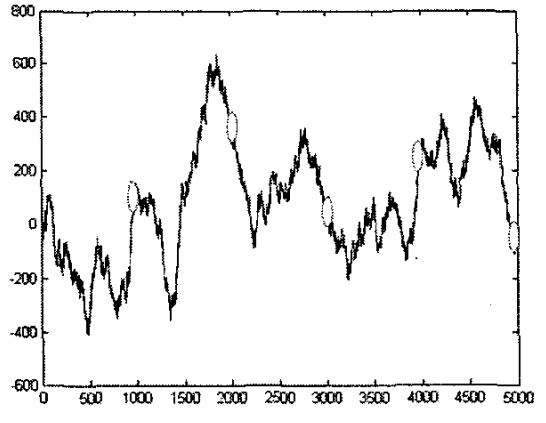

Fig 1. The CATS benchmark. The points in the five red ellipses are to be predicted.

This paper is organized as follows: Section II introduces the basic concepts of multi-stream Extended Kalman Filter training. Section III describes our weighted bidirectional approach and presents the prediction results. Conclusions are given in the Section IV.

II. MULTI-STREAM EXTENDED KALMAN FILTER

Multi-Stream Extended Kalman filter (EKF) is a practical, general approach to neural networks training. It consists of the following: 1) gradient calculation by Backpropagation Through Time (BPTT) [1][2]; 2) weight updates based on the extended Kalman filter; and 3) data presentation using multi-stream mechanics [3]. It has been widely used to deal with many different types of problems involving temporal systems. A typical problem might involve prediction, estimation and classification [4][5][6] or control [7][8]. The core of this training technique is the extended Kalman filter, which has become a standard technique used in a number of nonlinear estimation and machine learning applications. Singhal and Wu [9] first proposed weight updates based on EKF. (For background material on Kalman filter, see [10] and [11].) Weights are interpreted as states of a dynamic system [9], which allows for efficient Kalman training. Given a network with M weights and N_L output nodes, the weights update for a training instance at the time step n of the extended Kalman filter is given by:

$$A(n) = [R(n) + H'(n)P(n)H(n)]^{-1} \quad (1)$$

$$K(n) = P(n)H(n)A(n) \quad (2)$$

$$W(n+1) = W(n) + K(n)\xi(n) \quad (3)$$

$$P(n+1) = P(n) - K(n)H'(n)P(n) + Q(n) \quad (4)$$

$$P(0) = I/\eta_p, R(0) = \eta_r I, Q(0) = \eta_q I \quad (5)$$

In the above equations, R(n) is a diagonal N_L-by-N_L matrix, whose diagonal components are equal to or slightly less than 1. H(n) is an M-by-N_L matrix containing the partial derivatives of the output node signals with respect to the weights. P(n) is an M-by-M matrix defined as the approximate conditional error covariance matrix. A(n) is a

0-7803-8359-1/04/$20.00 ©2004 IEEE

N_L-by-N_L matrix referred as the global scaling matrix. K(n) is an M-by-N_L matrix containing the Kalman gains for the weights. $W(n)$ is a vector of length M containing all the weights values. $\xi(n)$ is the error vector of the network's output layer. While the motivation for the use of artificial process noise in Equation (5) was to avoid numerical difficulties, it was also found that the addition of the artificial process noise term significantly enhances the performance of the EKF algorithm in terms of rate of convergence, avoidance of local minima and quality of solution.

Decoupled Extended Kalman Filter (DEKF) [7][12] was implemented in [13] as a natural simplification of EKF by ignoring the interdependence of mutually exclusive groups of weights, thereby allowing the computational complexity of EKF to be adjusted to the low requirements of the computational resources. Decoupling was crucial for early practical applications of the method, when speed and memory capabilities of workstations and personal computers were severely limited. Now, many problems are small enough to be handled without decoupling, that is, with global EKF (GEKF). In many cases, full coupling brings benefits in terms of quality of solution and overall training time [14]. GEKF is employed in this study. The advantage of the EKF approach over Backpropagation (BP) is that generally EKF can often produce results comparable to standard BP but with significantly fewer presentations of training data and less overall training epochs [12]. Although not limited to, the GEKF has been widely used in the training of the time-lagged recurrent networks, usually in the form of Recurrent Multilayer Perceptrons (RMLP), which are a natural synthesis of feedforward multilayer perceptron and single-layer fully recurrent networks [3]. Figure 2 depicts a simple RMLP with only internal recurrent connections. Figure 3 gives the flow chart of the network learning process.

The multi-stream procedure [15] was devised to cope with the sometimes conflicting requirements of training [14]. Consider the standard time-lagged recurrent neural network training problem: training on a sequence of input-output pairs. If the sequence is fairly homogeneous, then one or more sequential passes through data will probably produce good results. When the data sequence is heterogeneous, as most real time series are, (for example, this CATS benchmark), GEKF is not enough because it is basically just a one-stream learning algorithm, in which the tendency always exists for the network weights to be adapted to the currently presented training data at the expense of performance on previous data, called recency effect [16]. This recency effect is analogous to the difficulty that may arise in training feedforward networks if training data are presented always in the same order. Multi-stream training is based on the principle that each weight update should attempt to satisfy simultaneously the demands from multiple input-ouput pairs. In each cycle of training, a specified number N_S of starting points are randomly selected in a chosen set of files. Each such starting point is the beginning of a stream. The multi-stream procedure consists of progressing in sequence through each stream, carrying out weight updates according to current points. A consistent EKF update routine was also devised in the multi-stream procedure. The training problem is treated as a single shared-weight network, in which the number of original outputs is multiplied by the number of streams. In multi-stream training, the number of columns in H(n) is correspondingly increased to $N_S \times N_L$.

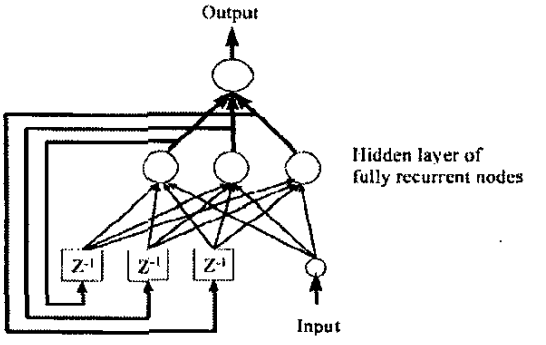

Fig 2. A simple architecture of a recurrent multilayer perceptron with fully internal recurrent connections, in which Z^{-1} represents the time lag.

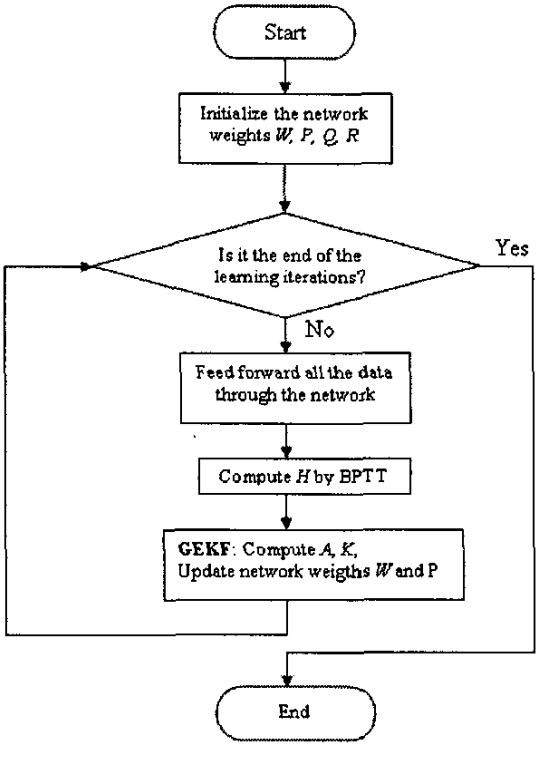

Fig 3. The flow chart of the RMLP learning process using GEKF, referred as in Equation (1)-(5).

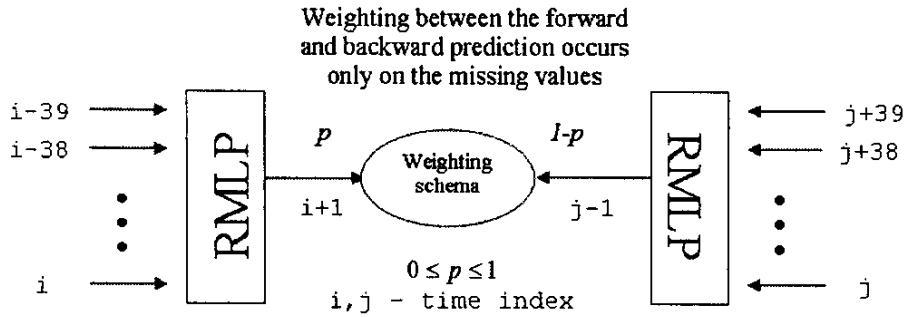

Fig 4. The weighting scheme for the incorporation of the results from the forward and the backward prediction. In the forward prediction, the next point is predicted by using the past 40 data points. In the backward prediction, the next point is predicted by using the future 40 data points, while the prediction moves backwards. The final prediction is computed by using Equation (6): $FinalPred = p * FwPred + (1-p) * BwPred$.

Similarly, the vector of errors ξ has $N_S \times N_L$ elements. The Kalman recursion produces weight updates which are not a simple average of the weight updates that would be computed separately for each output or stream.

III. METHOLOGY AND EXPERIMENT RESULTS

We describe a weighted bidirectional prediction combined with GEKF. After trying a few different network architectures, an RMLP architecture of 40-10R-10R-1L was chosen. The first and second hidden layers were both fully recurrent with a bipolar sigmoid transfer function. A linear transfer function was used in the output layer. The current data point, along with previous 39 data points, are presented as inputs to predict the value of the next point, which makes the input vector size 40-by-1 and output vector size 1-by-1. In other words, a sliding window with the size of 40 is moved along the time series while predicting the next point. One such input vector and its corresponding output vector form an input-output pair.

Initially, forward prediction was employed to validate the training technique. Every 20 points before each block of missing values were reserved as the test set. So, there were 100 test points with known values for the validation of the forward prediction approach. In the absence of prior information, we initialized the outputs of all neurons to zero at the start of each stream. The data were normalized within [-1 1] before being presented to the network for training. The network was executed for a number of steps N_t, the trajectory length, but updates were suspended for N_p time steps, called the priming length, at the beginning of each stream. Hence $N_t - N_p$ updates were performed in each training cycle. This priming scheme is based on the fact that the outputs of a stable network, after a suitable number of time steps, will be essentially independent of its initialization [3].

We employed multi-stream GEKF using ten (10) streams, a priming length of ten (10), and a trajectory length of 200. Because of the shortcomings of BPTT, which requires saving the entire history of network input and network state since the starting time, it was not used. Instead, one can use a bounded-history approximation, in which relevant information is saved for a fixed number h of time steps and any information older than that is forgotten. This is called truncated backpropagation through time [17][18], denoted as BPTT(h). Since BPTT(h) only utilizes the most recent information in a trajectory to compute derivatives, it may lead to a more stable computation of dynamic gradients than do the forward computations of dynamic derivatives. In this paper, we use truncated backpropagation through time with truncation depth h = 20 to compute the derivatives of the network output with respect to weights. The network was trained for 500 epochs, which account for $500 \times (N_t - N_p) = 500 \times 170$ updates, based on 500 × No.Stream × N_t = 500 × 10 × 200 input-output pairs. At the termination of training, the RMS error was approximately 0.0134. The prediction of the trained network on the test set of 100 points had an RMS error of about 0.0156. Please note that when performing the prediction on the test set, the current prediction has to be fed back to network input as part of the next input vector for the next prediction. At the prediction of the last point of each twenty points block in the test set, the network input vector has 21 known points and 19 predicted points. The same rationale applies to the final prediction of the missing values.

To enhance the performance of prediction, a scheme (shown in Figure 4) of the combination of the forward prediction and the backward prediction was employed. After the validation of the training technique by the forward prediction aforementioned, all the available data points were used in the training with no test points reserved. In the backward prediction, forty (40) "future" data points were used as the input to predict the value of the current data point. In the training of the forward prediction network and the backward prediction network, multi-stream GEKF using ten (10) streams, a priming length of ten and a trajectory length of 200 was employed. Each network was trained for $500 \times (N_t - N_p) = 500 \times 170$ updates, based on 500 ×

No.Stream × N_t = 500 × 10 × 200 input-output pairs. At the termination of training, the RMS error from the forward prediction was approximately 0.0134 and the RMS error from the backward prediction was approximately 0.0132 from thirty (30) simulations with different initial conditions.

At the prediction of the missing values, a linear symmetric weighting scheme was employed to incorporate the results from the forward and backward prediction to enhance the performance. Since we knew nothing about the signal, we assumed that the symmetric weighting approach was the best. So, we incorporated these heuristics to compute the final predictions by assigning a belief degree p to the results of the forward and backward prediction. At the point 981, the forward prediction had a belief degree p = 100% and the backward prediction had a belief degree $(1-p)$ = 0%, vice versa at the point 1000. For the other points between the point 981 and the point 1000, the belief rate p of the forward prediction was evenly decreased from 100% to 0% while the belief rate of backward prediction was $1-p$, being increased from 0% to 100%. Figure 4 depicts the weighting scheme between the forward and the backward prediction. The final predictions on the missing values were computed as follows:

$$FinalPred = p * FwPred + (1-p) * BwPred \quad (6)$$

except that the prediction on the points 4981- 5000 can only rely on the forward prediction.

Figure 5 shows the results of forward, backward and final prediction on the points 981-1000, 1981-2000, 2981-3000 and 3981-4000, averaged on the thirty (30) simulations with the different random initial conditions. Figure 6 gives the means and the standard derivations of the final predictions on the missing points.

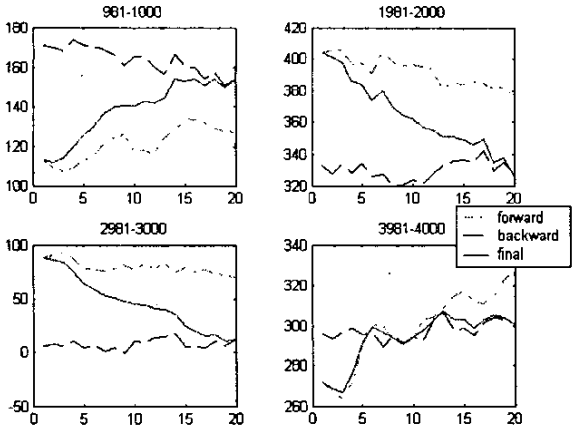

Fig 5. The results of the forward, backward and final predictions on the points 981-1000, 1981-2000, 2981-3000 and 3981-4000.

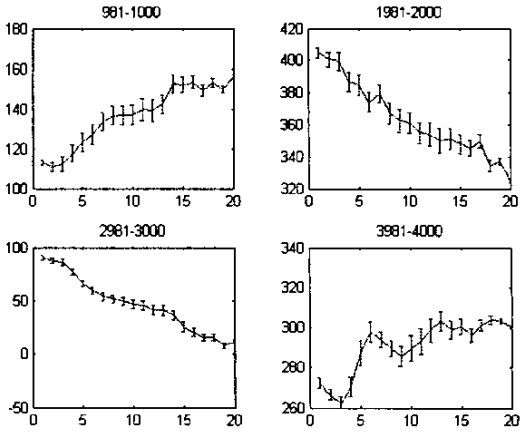

Fig 6. The mean and standard deviation of the final predictions on the points 981-1000, 1981-2000, 2981-3000 and 3981-4000.

IV. CONCLUSIONS

This paper presents our approaches in details on solving the IJCNN 2004 challenge problem. Multi-stream GEKF with a weighted bidirectional prediction approach was employed in the experiments to predict the missing values in the CATS benchmark. The whole time series with the filled predicted values is shown in Figure 7. The results illustrate our approach is effective and is able to achieve good predictions on the missing values of CATS.

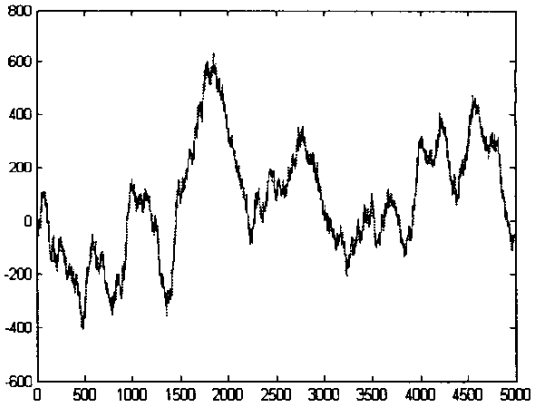

Fig 7. The whole time series with the predicted missing values

ACKNOWLEDGEMENT

Partial support for this research from the National Science Foundation, and from the M.K. Finley Missouri endowment, is gratefully acknowledged.

REFERENCES

[1] P.J. Werbos, "Backpropagation through time: what it does and how to do it". Proceedings of IEEE, 78(10), 1550-1560, 1990.

[2] P.J. Werbos. The Root of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting. John Wiley & Sons, Inc., New York, NY, 1994.

[3] L. Feldkamp, D. Prokhorov, C. Eagen, and F. Yuan, Enhanced Multi-Stream Kalman Filter Training for Recurrent Networks. In J. Suykens and J. Vandewalle (Eds.) Nonlinear Modeling: Advanced Black-Box Techniques, pp. 29-53, Kluwer Academic Publishers, 1998.

[4] G.V. Puskorius and L.A. Feldkamp, "Signal processing by dynamic neural networks with application to automotive misfire detection", in Proceedings of the World Congress on Neural Networks, pp.585-590, San Diego, 1996.

[5] K.A. Marko, J.V. James, T.M. Feldkamp, G.V. Puskorius, L.A. Feldkamp, and D. Prokhorov, "Training recurrent networks for classification", in Proceedings of the World Congress on Neural Networks, pp.845-850, San Diego, 1996.

[6] Emad W. Saad, Danil V. Prokhorov, Donald C. Wunch II, "Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks", Neural Networks, IEEE Transactions on, vol. 9, No.6, pp. 456 –470, Nov. 1998.

[7] G.V. Puskorius, and L.A. Feldkamp, "Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks", IEEE Trans. on Neural Networks, vol.5, no.2, pp.279-297, 1994.

[8] G.V. Puskorius, L.A. Feldkamp, and L.I. Davis, Jr., "Dynamical neural network methods applied to on-vehicle idle speed control", Proceedings of the IEEE, vol.84, no.10, pp.1047-1420, 1996.

[9] Singhal, S.; Wu, L., "Training feed-forward networks with the extended Kalman algorithm", Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP-89), vol.2, pp. 1187 –1190, 1989.

[10] B.D.O. Anderson and J.B. Moore, Optimal Filtering, Englewood Cliffs, NJ: Prentice-Hall, 1979.

[11] S. Haykin, Adaptive Filter Theory, Englewood Cliffs, NJ: Prentice-Hall, 1996.

[12] Puskorius, G.V.; Feldkamp, L.A., "Decoupled extended Kalman filter training of feedforward layered networks", Neural Networks, Proceedings of IJCNN-91-Seattle International Joint Conference on , vol. 1, pp. 771 –777, 1991.

[13] Xiao Hu; John Vian; Jai Choi; David Carlson; Donald C. Wunsch II, "Propulsion vibration analysis using neural network inverse modeling", Neural Networks, 2002, Proceedings of the International Joint Conference on, vol.3, pp. 2866 –2871, 2002.

[14] J.F. Kolen, S.C. Kremer, A Field Guide to Dynamic Recurrent Networks, Wiley-IEEE Press, New York, March 2001.

[15] L.A. Feldkamp, G.V. Puskorius, "Training controllers for robustness: Multi-stream DEKF", In Proceedings of the IEEE International Conference on Neural Networks, pp.2377-2382, Orlando, FL, 1994.

[16] G.V. Puskorius, L.A. Feldkamp, "Multi-stream extended Kalman filter training for static and dynamic neural networks", in Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, vol.3, pp.2006-2011, Orlando, FL, 1997.

[17] R. J. Williams and J. Peng, " An efficient gradient-based algorithm for on-line training of recurrent network trajectories". Neural Computation 2, pp. 490-501, 1990.

[18] L.A. Feldkamp, G.V. Puskorius, "Truncated backpropagation through time and Kalman filter training for neurocontrol" Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on , Volume: 4 , 27 June-2 July 1994 Pages:2488 - 2493 vol.4.