In *Proceedings of the International Joint Conference on Neural Networks* (IJCNN-92, Baltimore, MD), II:901–906.

Data Rectification using Recurrent (Elman) Neural Networks *

T. W. Karjala and **D. M. Himmelblau**
Department of Chemical Engineering
University of Texas at Austin

R. Miikkulainen
Department of Computer Sciences
University of Texas at Austin

Abstract

Nonlinear programming techniques are used to train Elman(1990) type simple recurrent neural networks to reconcile simulated measurements for a simple dynamic system, a draining tank. The networks are trained in a batch mode using the BFGS quasi-Newton nonlinear optimization algorithm. Noisy data are used for both training the networks and testing network performance. Recurrent Elman networks are able to significantly reduce the noise level in the process measurements without explicit knowledge of the nonlinear dynamics of the system.

1 Introduction

Rarely does the overall amount of raw material processed and energy furnished to a modern chemical plant or refinery exactly match the amount of product produced or energy consumed because of inaccuracies in the measurement of the process variables. It is the goal of data rectification to compensate for these random and nonrandom measurement errors by making suitable adjustments to the measurements. For steady state processes, methods have been developed that first detect and remove nonrandom or gross measurement errors, and then perform constrained weighted least squares minimization to make the adjustments[10, 1, 9, 6, 2, 5]. Others have applied nonlinear programming techniques directly to the dynamic process[7]. Both methods are limited to processes for which accurate models exist complex. The recurrent neural network approach outlined in this paper can be applied, provided sufficient historical data exist, to processes for which the detailed knowledge required to build accurate dynamic models is lacking.

2 Data Rectification and Neural Networks

Neural networks provide an alternative approach for data rectification. All current methods that have been proposed utilize redundancy in the plant measurements and explicit expressions for the plant model and/or constraints. More process variables are normally measured than are absolutely necessary to specify the state of a process given knowledge of the constraints, and the values of these variables can be adjusted so that the constraints are satisfied. It is proposed that neural networks be trained to used the redundancy in the plant measurements and knowledge of the process to rectify the raw data (Figure 1). Ideally, with a feed-forward network, inaccurate plant measurements at a given time would be used as the network inputs and the "true" values of the measurements would be used as the targets of the network during training. For real plant data however, the true values of the measured variables are not known and thus optimal target patterns for network training are not available. By reposing the problem as a modeling or time series prediction problem and asking the network to predict the current set of measurements based on past measurements, the lack of suitable targets can be avoided.

*This research was supported in part by Shell Development Corporation

3 Nonlinear Programming and Elman Networks

Many researchers using neural networks for time series prediction have chosen to "parallelize time" by incorporating a moving window of input values for standard feed-forward networks. The past N values of the M input variables might be used simultaneously as the network inputs resulting in networks with NxM input nodes. The length of the data window N must be long enough to capture the dynamics of each variable, but the number of nodes must be kept to a minimum in order to minimize the size of the network. In general, the size of the data window must be determined by trial and error, and each input variable in a multivariate time series should have a separate data window size for optimal performance. This simply adds to the number of parameters that must be tuned by trial and error when using a standard back-propagation training technique.

There has recently been considerable interest in using recurrent network architectures for time series prediction. Recurrent networks include links between nodes that feedback signals to other nodes on the same or prior layers. This provides networks with internal states and a form of memory. As a result, recurrent networks provide more than the simple 1-to-1 mapping of feed-forward networks. The outputs of such networks depend not only on the current inputs but also on previous inputs. Time is represented implicitly rather than explicitly through the use of a moving window.

The recurrent Elman architecture was chosen for this work because of its simplicity [3]. This architecture is similar to the standard feed-forward architecture with layers of input units, hidden units, and output units, but also includes a set of context units which save the prior activation of the hidden units and feedback the stored activation of the previous cycle to the hidden units in a fully connected manner (Figure 2). For the application under consideration, the input vectors correspond to the plant measurements at a given time step and the target vectors are the plant measurements at the next time step. The number of hidden nodes is equal to the number of context nodes and must be adjusted to fit the problem at hand. The input layer and the hidden layer had one bias node each, and Gaussian activation functions were used throughout the network. Our experience has shown that Gaussian activation functions give superior results when compared to sigmoidal functions.

Most recurrent and ordinary feed-forward networks used today are trained using some form of back-propagation. Back-Propagation is actually a form of gradient descent and a good derivation can be found in [4]. The goal is to minimize the squared error between the actual network outputs and the target values by adjusting the weights in the network for all the output nodes i and patterns t:

$$\text{minimize } E[\mathbf{w}] = \frac{1}{2}\sum_t \sum_i (T_{ti} - O_{ti})^2 \tag{1}$$

Gradient descent is one of the simplest but also one of the slowest unconstrained optimization methods. Much better algorithms exist and are described in relation to neural networks in [11]. The single most popular method today for unconstrained optimization is the BFGS quasi-newton algorithm. For a detailed theoretical discussion see [8]. This method has consistently outperformed other methods on comparably sized problems. This algorithm is commonly available in most standard numerical libraries for workstations and mainframe computers.

The Elman networks used in this work were trained in batch mode, for a entire time series at a time. A subroutine was written to return the value of the above error, E, as a function of the network weights and training set. This subroutine was interfaced to a standard optimization package which iteratively adjusted the weights of the network until an optimum was found. By using the BFGS nonlinear programming algorithm, it was possible the avoid the time consuming tuning of learning rate and momentum required for optimal training using the back-propagation algorithm. We have found that adjustment of the weights via NLP takes as little as 10% of the time used by back-propagation.

4 An Example Dynamic System

To generate simulated measurements in time for network training and testing, one of the simplest dynamic systems found in engineering practice was used, namely the draining tank (Figure 3). In this system, the exit volumetric flow rate q is a function of the liquid level height h, and various physical parameters such as liquid density ρ, valve coefficient C_v, and gravitational constant g:

$$q = C_v\sqrt{\rho g h} \tag{2}$$

The height of the liquid level in the tank is governed by the first order, nonlinear differential equation

$$A\frac{dh}{dt} = q_i - C_v\sqrt{\rho g h} \tag{3}$$

where A is the cross-sectional area of the tank and q_i is the input flow rate to the tank. These equations were de-dimensionalized so that q_i, h, and q ranged from 0 to 1. This system was numerically integrated to generate time series data as a function of q_i and time for the three measured variables. Noise was added to each measured variable using

$$y = \eta + \epsilon \tag{4}$$

where y was the simulated measurement, η was the true value generated from the solution of (2) and (3), and ϵ was the random measurement error generated from

$$\epsilon = \mu(0.05) \tag{5}$$

where μ is a normally distributed variable with zero mean and unit variance. The resulting random measurement errors had zero mean and variance of 0.0025 and were independent of measurement magnitude.

5 Results for Example Problem

The liquid level h, from a typical training set is shown in Figure 4 without measurement errors. This same variable from a typical test set is shown in Figure 5. This data was generated by integrating the response of the system with respect to step changes implemented at different times in the input flow rate. In Figure 6, the output of a network with three hidden nodes is shown for a noisy test set. Note that the network is able to follow the dynamic changes that occurred when the input flow rate was stepped from one value to another. Table 1 shows the reduction in variance of the measurement noise accomplished by this network. These variances were calculated for the deviation δ_t between the true values of the measurements and the network's predictions for each time step:

$$\delta_t = (\eta_{net} - \eta)_t \tag{6}$$

Note that the output variance decreased by an order of magnitude for h and q but increased for the input flow rate q_i in both the training and test sets. Both h and q are smooth functions of time but q_i is discontinuous as shown in Figure 7.

For a network trained on data generated by repeatedly ramping the input flow rate as shown in Figure 8, the results in Table 2 were obtained. In this case the output variance of all the errors were reduced by an order of magnitude for the training set, but the test set output variance decreased by an order of magnitude for h and q but increased to 0.013 for the input flow rate. The reconciliation performance of the network for the liquid level is shown in Figure 9.

In an effort to determine the best performance of this network configuration on this problem, one network was trained with noisy input vectors and with the actual, noise free target vectors for the measurements from the ramped input flow data. This represents the ideal case in which the true values of the variables are available for training. The results are shown in Table 3. Figure 10 shows the network performance for the liquid level. Note that in this case the network was able to reduce the variance for all three variables.

The choice of Elman networks with three input nodes, three hidden nodes, three context nodes and three output nodes was based on qualitative examination of network performance on rectification of the liquid level measurement. It is probable that other configurations could be found that would do a better overall job if suitable quantitative criteria was used to select the optimal network size.

6 Conclusions

The Elman recurrent architecture was able to significantly reduce the noise level in simulated process measurements for the simple dynamic system studied here. The network improved the precision of the smooth variables h and q but increased it for the discontinuous or piecewise continuous variable q_i. These results were obtained by training the networks only on the noisy measurements, without an explicit equation for the process model. It should be possible to apply these techniques to actual process data. Maximum performance was obtained by training the network with noisy inputs and with the true, noiseless outputs but such target data is not normally available in practice.

The networks were trained using straightforward, unconstrained nonlinear optimization techniques. This made it possible to avoid the trial and error associated with tuning the learning rate and momentum terms required in the various back-propagation algorithms.

The random measurement errors used in this work were uncorrelated and purely Gaussian in nature. Future work will investigate the effect of correlated errors, gross errors, and biases on network performance.

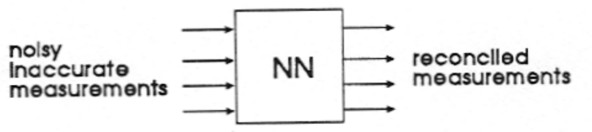

Figure 1: **Data Rectification using a Neural Network.**

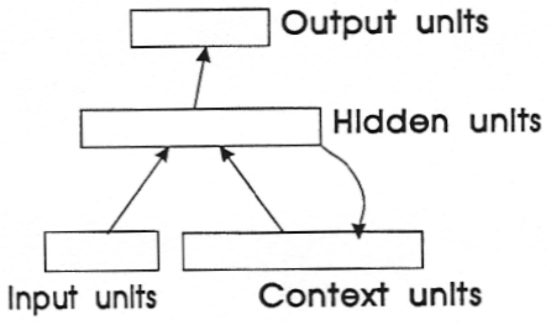

Figure 2: **The Elman Architecture.**

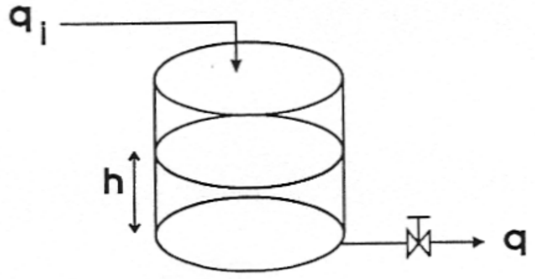

Figure 3: **A Simple Dynamic System, The Draining Tank.**

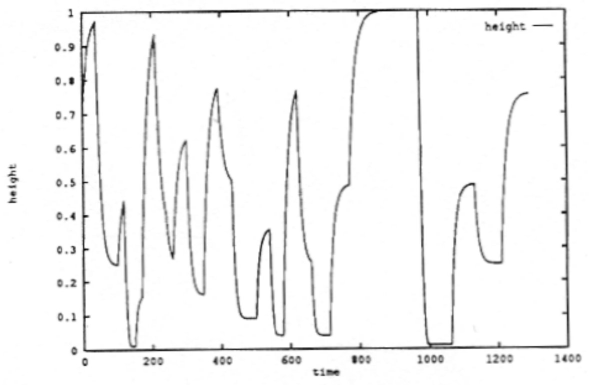

Figure 4: **Liquid Level h from the Step Training Set.**(Before the addition of measurement noise)

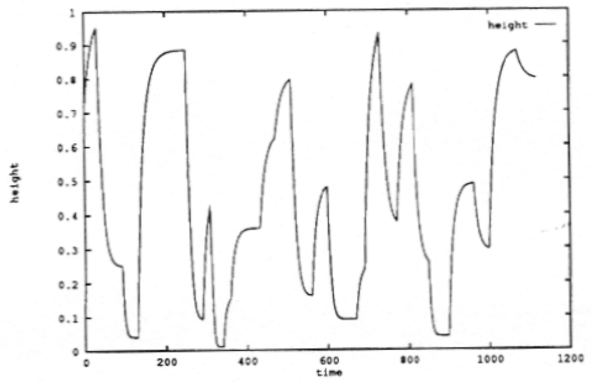

Figure 5: **Liquid Level h from the Step Testing Set.**(Before the addition of measurement noise)

Table 1: **Data Rectification Results.** Network with three hidden nodes trained on noisy step data.

	measurement	input error variance	output error variance
Training Set	q_i	0.0025	0.0051
	h	0.0025	0.00032
	q	0.0025	0.00040
Testing Set	q_i	0.0025	0.0049
	h	0.0025	0.00026
	q	0.0025	0.00023

7 Nomenclature

A	cross-sectional area of tank
C_v	valve coefficient
E	training set error
g	gravitational constant
h	height of liquid in tank
i	output node index
O_{ti}	activation of output node i at time t
q_i	input flow rate
q	output flow rate
T_{ti}	target value for output node i at time t
t	time, time index
$\mathbf{w}$	vector of network weights
y	simulated measurement
δ	deviation between measured value and network prediction
ϵ	random measurement error
η_{net}	network prediction of measured variable
η	true value of measured variable
μ	normally distributed variable with zero mean and unit variance
ρ	liquid density

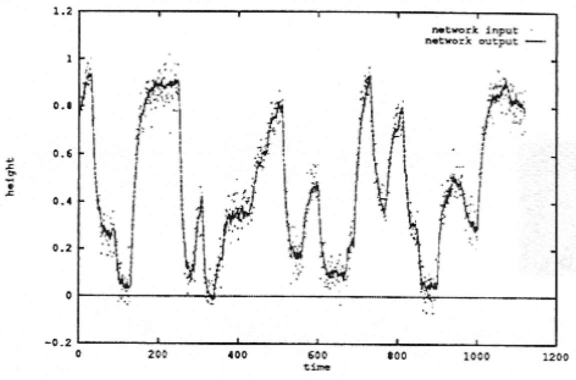

Figure 6: **Rectification of the Liquid Level for Step Changes in Input Flow Rate.** The Elman network was trained using data containing measurement noise in both the input patterns and target values.

References

[1] C. M. Crowe. The maximum-power test for gross errors in the original constraints in data reconciliation. In *Proceedings of the PSE '91*, pages II.7.1–14, Montebello, Canada, August 5–9, 1991 1991.

[2] C. M. Crowe, C. Y. Garcia, and A. Hrymak. Reconciliation of process flow rates by matrix projection, Part I: Linear case. *AIChE Journal*, 29:881, 1983.

[3] J. L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[4] J. A. Hertz, A. S. Krogh, and R. G. Palmer. *Introduction to the Theory of Neural Computation.* Addison Wesley, 1991.

[5] C. D. Iordache, R. S. H. Mah, and A. C. Tamhane. Performance studies of the measurement test for detection of gross errors in process data. *AIChE Journal*, 31:1187, 1985.

[6] J. Y. Keller, M. Darinach, and G. Krzakala. Gross error estimation in linear steady state systems. In *Proceed. IFAC/IMACS Symp. "SAFEPROCESS 91"*, pages 211–214, Baden-Baden, September 10–13 1991.

[7] I.-W. Kim, M. J. Liebman, and T. F. Edgar. Robust error-in-variables estimation using nonlinear programming techniques. *AIChE Journal*, 36(7):985–993, 1990.

[8] D. G. Luenberger. *Linear and Nonlinear Programming.* Addison Wesley, second edition, 1989.

[9] H. Takiyama, Y. Naka, E. O'Shima, and A. Adriani. Sensor-based data reconciliation method and application to the pilot plant. *Journal Chemical Engineering Japan*, 24(3):339–346, 1991.

[10] A. C. Tamhane and R. S. H. Mah. Data reconciliation and gross error detection in chemical process networks. *Technometrics*, 27(4):409–422, 1985.

[11] R. L. Watrous. Learning algorithms for connectionist networks: Applied gradient methods of nonlinear optimization. Technical Report MS-CIS-88-62, Department of Computer and Information Science, University of Pennsylvania, 1988.

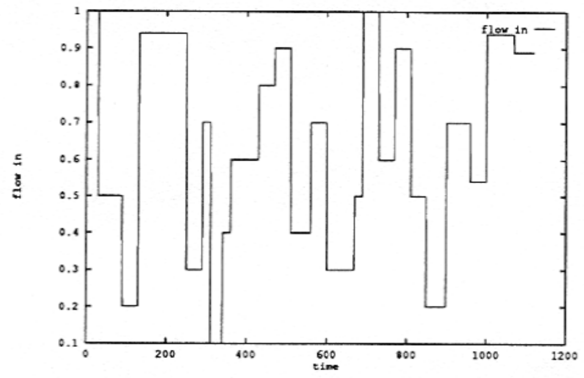

Figure 7: **Input Flow Rate q_i for the Step Training Set.**(Before the addition of measurement noise)

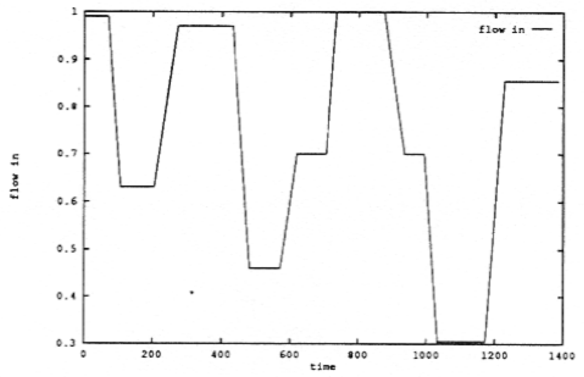

Figure 8: **Input Flow Rate q_i for the Ramp Training Set.**(Before the addition of measurement noise)

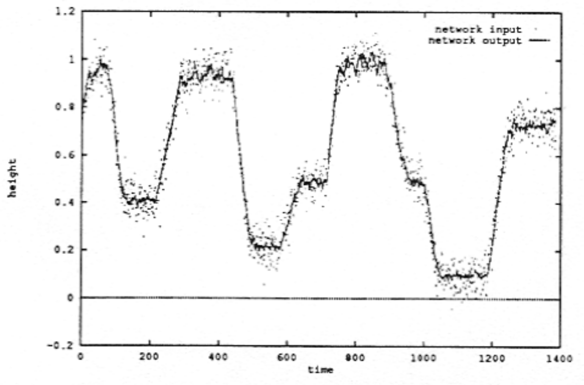

Figure 9: **Rectification of the Liquid Level for Ramp Changes in Input Flow Rate.** The Elman network was trained using data containing measurement noise.

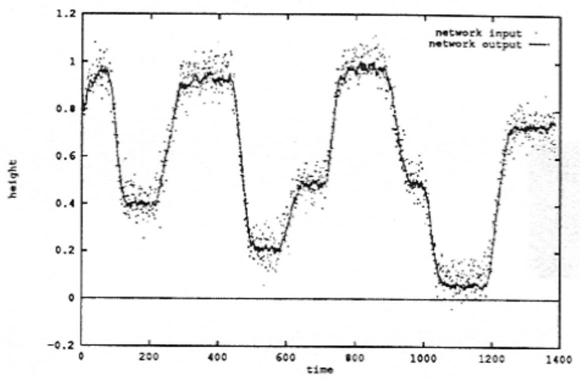

Figure 10: **Rectification of the Liquid Level for Ramp Changes in Input Flow Rate.** The Elman network was trained using data containing measurement noise in the inputs but with the true values as targets.

Table 2: **Data Rectification Results.** Network with three hidden nodes trained on noisy ramp data.

	measure-ment	input error variance	output error variance
Training Set	q_i	0.0025	0.00066
	h	0.0025	0.00028
	q	0.0025	0.00041
Testing Set	q_i	0.0025	0.013
	h	0.0025	0.00027
	q	0.0025	0.00030

Table 3: **Data Rectification Results.** Network with three hidden nodes trained on clean ramp data.

	measure-ment	input error variance	output error variance
Training Set	q_i	0.0025	0.00062
	h	0.0025	0.00026
	q	0.0025	0.00022
Testing Set	q_i	0.0025	0.00074
	h	0.0025	0.00039
	q	0.0025	0.0018