Learning a Nonlinear Model of a Manufacturing Process Using Multilayer Connectionist Networks

Charles W. Anderson Judy A. Franklin Richard S. Sutton

GTE Laboratories Incorporated
40 Sylvan Road
Waltham, MA 02254

Abstract

A connectionist (neural) network learns a nonlinear process model by observing a simulated manufacturing process in operation. The objective is to use the model to estimate the effects of different control strategies, removing the experimentation from the actual process. Previously we demonstrated that a linear, single-layer connectionist network can learn a model as accurately as a conventional linear regression technique, with the advantage that the network processes data as it is sampled. Here we present experiments with a multilayer extension of the network that learns a nonlinear model.

1 Introduction

The control of a manufacturing process can be very risky when the process is incompletely understood. The effects of minor adjustments in one stage of a process can be magnified in subsequent stages and result in a grossly inferior product. The risk of making adjustments can be decreased by building a model of the process and experimenting with changes to the controls of the model rather than of the actual process.

A model of a process is an abstraction that to some level of detail is an accurate simulation of the process. Below this level of detail and for other less-understood aspects of the process, the model can only be a mathematical approximation of the relationships between measurable variables. Ideally, the modeler knows enough about a process to simulate the known reactions and to implement mathematical approximations of the remaining relationships. Usually this much knowledge is not available and the modeler must resort to alternative methods.

A more general approach to modeling a process is to build a model incrementally by observing values of process sensors while the process is operating. The modeler first constructs an initial, parameterized model using all available knowledge about the process. Then an automatic parameter-adjustment procedure observes the process in operation and updates, or *learns on-line*, the model's parameters to drive the output of the model closer to the corresponding sensor values. Learning an accurate model is confounded by a large number of sensors, different ranges in values among sensor readings and ranges that vary over time, noise in the sensor readings, unknown and variable delays as material moves through various stages, and nonlinear interactions among sensed variables. This kind of model—a mathematical model that is learned on-line—is the subject of the research reported in this memo. We do not address how to use available knowledge to simulate the known dynamics of a process.

Connectionist networks show promise for handling many frequently-sampled, noisy sensors, and of tracking changes in a process, because of the parallel and incremental nature of a connectionist network's computations. In our work, we are combining these features of connectionist networks with new techniques for dealing with incompletely known sensor ranges, delays, and nonlinearity. Here we focus on our approach to tracking sensor ranges and learning nonlinear models. We will address delays in future work.

When the modeler knows all nonlinear relationships among the process variables and can implement them, modeling is straightforward. The modeler implements a preprocessing stage in which the known nonlinear functions transform the sensor values into a new set of inputs for the model. If the modeler includes all nonlinearities, a linear adaptive model would suffice that could be trained via well-known parameter adaptation algorithms, like the Least-Mean-Square (LMS) algorithm (Widrow and Stearns, 1985). If the modeler is not aware of every nonlinearity, he can guess which nonlinear functions might exist in the process. Existing process knowledge can constrain the number of nonlinear functions that could possibly be present. In general, the large number of possible functions prohibits including them all. Another difficulty is that the nonlinear relationships present in the process might change with time.

TH0333-5/90/0000/0404$01.00 © 1990 IEEE

There is no general solution to this problem, called the "representation problem" in the artificial intelligence field. When the modeler's knowledge is insufficient to specify a good preprocessor, the only alternative is to design a preprocessor using what knowledge is available and then adapt the preprocessor as experience is gained by observing the process. The set of functions implemented by the preprocessor must be changed to represent more accurately the information that is needed to form a good model.

The structure of a multilayer connectionist network is an adaptive preprocessor followed by an output stage. For example, the first layer of a two-layer network preprocesses the network's inputs and provides a new set of transformed values as input to the second layer. The second layer calculates the output of the network. Connectionist learning algorithms exist that treat learning in a uniform manner in all layers. Their ability to learn new representations in initial layers of a network partially accounts for the popularity and success of connectionist networks.

1.1 Previous Work

Members of the Connectionist Machine Learning Project at GTE Laboratories have tested and continue to develop a number of connectionist algorithms for learning process models on-line. The process that is the subject of most of our experiments is the coating process of a fluorescent-lamp manufacturing line.

Previous work resulted in two achievements. First, after considerable discussions on technical details of the line, we created a simulation, called CIMSIM (Computer Integrated Manufacturing SIMulation), of the coating process. Our second achievement was the development of a new connectionist learning algorithm, called the NADALINE (Sutton, 1988), or Normalized ADALINE. The NADALINE is an extension of Widrow's ADALINE learning algorithm (Widrow and Stearns, 1985) that normalizes the input values to have zero mean and unit variance.

Using CIMSIM, we demonstrated that the NADALINE algorithm learned a model as accurately as a conventional regression technique (Franklin, Sutton, and Anderson, 1988). The NADALINE algorithm has a practical advantage over regression in that it incrementally processes the sensor values as they are sampled. Standard regression procedures gather samples into a large matrix and processes the matrix at a later time. The NADALINE, however, is limited in its treatment of nonlinearity and delays. It forms only linear models and delays must be identified beforehand.

1.2 Overview

Here we describe an extension of the NADALINE, referred to as the nonlinear NADALINE, with which a connectionist network can learn nonlinear models. We used CIMSIM2, an elaboration of CIMSIM, to test the nonlinear NADALINE. CIMSIM2 and the modeling problem are described in Section 2. Section 3 briefly describes the two-layer connectionist network. Results of experiments with the nonlinear NADALINE network and CIMSIM2 are presented in Section 4, followed by conclusions in Section 5.

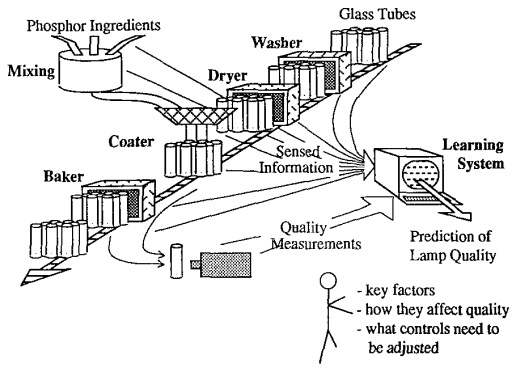

Figure 1: The Simulated Fluorescent-Lamp Coating Process

2 CIMSIM2: The Process Simulation

The nonlinear NADALINE was tested by applying it to the task of learning a model of the fluorescent-lamp coating process as simulated in CIMSIM2. As shown in Figure 1, trays of glass tubes pass through a sequence of stages. CIMSIM2 simulates individual trays and alters their attributes as they are processed at each stage. One of these attributes is the percentage of the tubes in the tray that have flaws. Flawed tubes are rejected at the end of the coating stage. The stages also have attributes whose values, along with those of the tray attributes, can be "sensed" by simulated sensors. CIMSIM2 currently simulates 16 sensors, including a sensor for percent rejects. The process monitoring task is to learn a model that predicts percent rejects from the sensor values.

The original CIMSIM simulated the interactions between process variables as linear functions with added noise and delays. To create CIMSIM2, we added the following nonlinear relationships. The effects of two of the sensed variables on the percent rejects were specified to be V-shaped, piecewise linear functions with each variable having a single optimum value for which the effect on the percent rejects was at a minimum. This kind of relationship is often found in real processes; performance drops as certain variables drift from their optimum values. The noise and delays present in CIMSIM are also present in CIMSIM2. These factors prevent a perfect model from ever being learned.

The purpose of CIMSIM and CIMSIM2 is to allow us to explore issues, such as noise, nonlinearity, delays, and changes over time, that arise in the online learning of models. The simulations do not accurately reflect the values of variables in the actual manufacturing process. To truly test the usefulness of our model-learning algorithms we must apply them to the actual process; the simulation studies are necessary for developing the algorithms in preparation for their installation at the manufacturing plant.

3 The Connectionist Network

This section summarizes the connectionist network and its computations. See Anderson, Franklin, and Sutton (1990) for details of the network structure and a complete specification of the nonlinear NADALINE learning algorithm.

The connectionist network is shown in Figure 2. The input to the network consists of the values sampled from the sensors at a given time. In our development of the nonlinear NADALINE and the experiments with CIMSIM2, we ignored the effects of delays. The network accepts the input values and calculates an output value. During training, CIMSIM2 provides the desired output value, which is the percent rejects.

Inputs to the network are not necessarily just the sampled sensor values. They can also be functions of combinations of sensor values. When some knowledge of the possible interdependencies among sensor values exists, these interdependencies should be represented by building in additional inputs as functions of sensor values. This preprocessing of the sensor data can greatly reduce the time required to learn a good model of the process. For our experiments, each input corresponded to a single sensor value.

The first operation performed by the connectionist network is a normalization of the input values. The normalization is identical to the procedure defined as part of the NADALINE algorithm (Sutton, 1988). The result is a set of input values with zero mean and unit variance. This step can speed learning considerably, especially for cases in which the sensor values have widely varying means and variances. NADALINE performs the normalization by maintaining exponentially-weighted averages of each sensor's mean and variance. The value of the parameter λ controls the exponential weighting.

The normalized inputs become inputs to two layers of adaptive units. The units of the first layer are called hidden units, because their outputs are not visible outside the network. Each hidden unit has a numerical weight associated with each input value, plus an additional "bias" weight. The hidden units compute nonlinear functions that are parameterized by their weights. Hidden units learn new functions by adjusting the weight values. A general method is not known for determining the number of hidden units that a particular problem requires. The effect of the number of hidden units on performance was explored experimentally by testing a number of different sized networks, as described in the following section.

The second layer of adaptive units contains a single unit—the output unit. It receives as input the normalized sensor values plus the outputs of the hidden units. This unit has its own weights associated with each input, plus a bias weight.

The calculations performed by each unit is a function of its inputs and weight values. The network is trained by adjusting the weights in every unit in such a way that the network's output more closely matches the desired output. In this case, the output of the network is a prediction of the percent rejects and the weight update algorithm is based on a gradient-descent

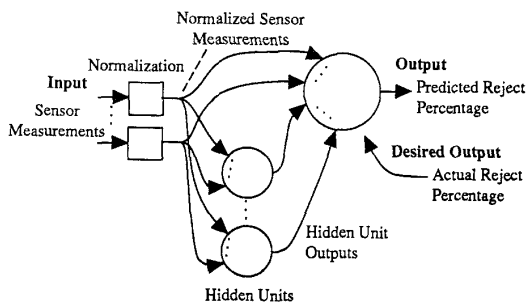

Figure 2: The Connectionist Network

procedure in the mean square error between the predicted and the actual percent rejects (from CIMSIM2). The update algorithm is a synthesis of the NADALINE algorithm and the error back-propagation algorithm of Rumelhart, Hinton, and Williams (1985).

For some connectionist network applications, the weight update algorithm can terminate after it achieves a desired accuracy (small difference between the network output and the desired output). However, the problem of modeling a changing process is never complete. The dynamics of a manufacturing line involve many time scales. Some aspects of the process will change very slowly and necessitate constant adjustment of the model to match the drift in the process.

4 Results

We ran CIMSIM2 for 10 simulated weeks, sampling all 16 sensors every 15 minutes. We trained the network on the first 9 weeks of data for a total of 6,048 samples. Then we tested the trained network by freezing the weights and calculating the root-mean-square (RMS) error over the 10^{th} week. This error is a measure of how well the network generalizes its 9 weeks of learning experience to the additional week of novel data.

We trained networks having different numbers of hidden units and values of λ. We repeated each experiment five times. Figure 3 shows the resulting average errors. The value of 0.999 for λ is clearly the best value of those tested. Which value is best depends on the speed with which process variables change and the delays in the process between sensed variables and their effects on the percent rejects.

For all but the smallest value of λ, the error decreases as the number of hidden units increases. However, the error does not change significantly as the number of hidden units is increased past 10. There is currently no way of determining how many units in a network will result in the best solution for a given problem other than simply varying the number and observing the results. One must perform similar experiments when applying the connectionist learning algorithms to the real process in order to explore the effects of various parameter values.

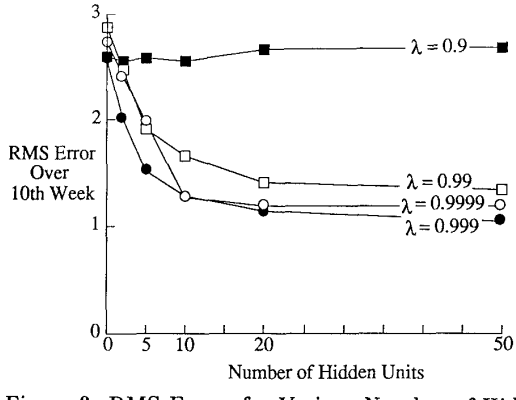

Figure 3: RMS Errors for Various Number of Hidden Units

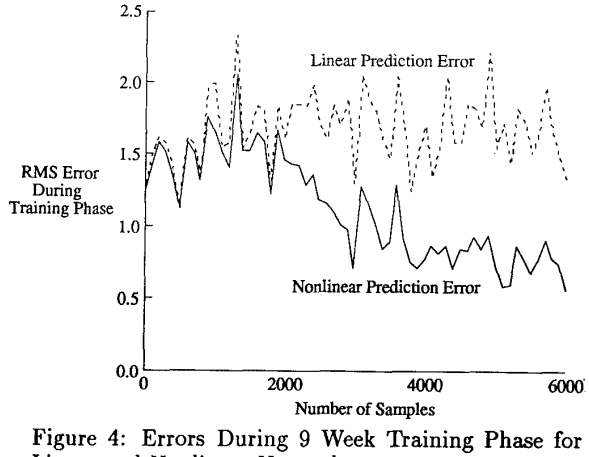

Figure 4: Errors During 9 Week Training Phase for Linear and Nonlinear Networks

A network with no hidden units is simply a single-unit, linear network, exactly as used in the original experiments with CIMSIM reported in Franklin, et al., (1988). The presence of hidden units obviously improves the accuracy of the model learned from data generated by the nonlinear simulation CIMSIM2.

The difference that hidden units make in the predictions made by the learned model is illustrated by the following results. We trained a network with 10 hidden units and with $\lambda = 0.999$. The error during training is plotted in Figure 4 along with the error of a linear network, with no hidden units. The graph shows that the linear network failed to improve throughout the training phase, while the nonlinear network decreased the RMS error from approximately 1.2 to 0.5.

The models learned by the two networks are more readily compared by superimposing the outputs of the networks and the actual percent rejects. We did this for the first day of the novel testing data, resulting in Figures 5 and 6. Figure 5 shows the actual percent rejects generated by CIMSIM2 (dashed line) and the percent rejects predicted by the linear network (solid line). The corresponding graph for the nonlinear network with 10 hidden units is in Figure 6. The percent rejects predicted by the nonlinear network follows much

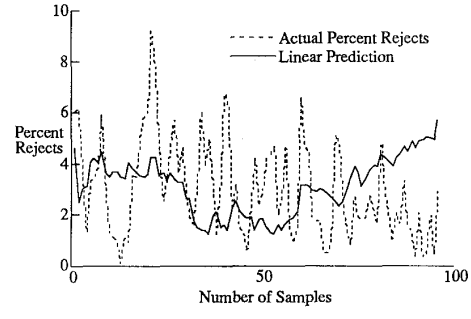

Figure 5: Actual Percent Rejects Compared with Linear Predictions During First Day of Testing Period

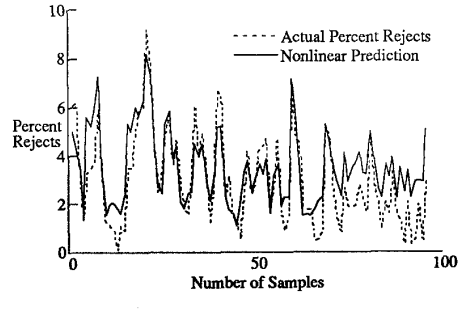

Figure 6: Actual Percent Rejects Compared with Nonlinear Predictions During First Day of Testing Period

more closely the actual percent rejects from CIMSIM2. The model learned by the network can never exactly match the actual percent rejects, because of the noise and delays in the process simulation that we assumed to be unknown.

Note that the percent rejects obtained from CIMSIM2 do not accurately reflect the values actually observed moment to moment in the real manufacturing line. CIMSIM and CIMSIM2 incorporate the kinds of relationships between process variables that probably exist in the actual process, but do not accurately represent all variables, magnitudes, time scales, and disturbances.

Anderson, et al. (1990) reports the results of additional experiments in which we varied other parameters of the nonlinear NADALINE algorithm, such as constant factors that control the magnitude of every weight update.

5 Discussion

The two-layer connectionist network with the nonlinear NADALINE learning algorithm successfully learned an accurate model of the simulated fluorescent-lamp coating process by observing the process on-line. The network steadily decreased the error in its output during the 9 weeks of training. During the 10 week of new data, the model accurately predicted the percent rejects, demonstrating that the model generalizes well to novel situations.

By varying the network's structure, we found that the accuracy of the model increased as the number of hidden units increased, but the increase in accuracy with more than 10 units was relatively small. We also varied the value of λ and found that $\lambda = 0.999$ resulted in the best accuracy. Performance was worse for higher and lower values of λ. This shows that a decay rate of $\lambda = 0.999$ best matches the delays present in CIMSIM2.

For our experiments, CIMSIM2 contained only simple nonlinearities that are functions of single variables; we did not investigate interactions between multiple variables. In the real manufacturing line, the dependence between sensor values and the rate of rejection will certainly involve such interactions. For example, the effect on percent rejects of a viscosity variable probably depends on air humidity and temperature. Though not shown here, the nonlinear NADALINE algorithm is capable of learning nonlinear interactions like these. In fact, the hidden units initially learn functions of many sensor values and only through further experience do they learn that some sensor values are not useful as inputs to the model.

The extension of the connectionist network to additional layers of hidden units is straightforward, but was not investigated. It is possible to implement certain functions more efficiently—using fewer connections—with multiple hidden layers. However, it is not clear that additional layers provide any advantage in learning speed. Theoretical analysis of the error back-propagation algorithm suggests that, in general, the convergence rate of the algorithm changes little as layers of hidden units are added, though convergence can be faster in certain circumstances (Tesauro, He, and Ahmad, 1989).

6 Conclusions

The nonlinear NADALINE algorithm successfully learned an accurate model of the simulated nonlinear process. However, general conclusions are limited by the fact that CIMSIM2 only simulated simple nonlinearities and a small number of variables. Another limitation is that we designed CIMSIM2 to exhibit approximately the kinds of delayed, nonlinear, and noisy relationships that exist in the real manufacturing process. The most significant test of the algorithm's ability is its application to the real process. We plan to install the algorithm at the manufacturing plant and perform extensive tests. The simplicity of the connectionist network and nonlinear NADALINE algorithm makes them conducive to implementation on a relatively small computer, making it easy to perform experiments in a real manufacturing environment.

7 Future Work

In future work, we will address the problem of delays. Our approach will be to combine Sutton's (1988) temporal-difference methods for handling uncertain delays with the nonlinear, NADALINE algorithm. We will test this in the actual manufacturing plant. The incorporation of techniques for dealing with unknown delays will make this approach more generally applicable.

An interactive environment is being constructed to facilitate the interpretation of what is learned by the connectionist network. Plant operators and managers will be able to query the model for predictions of percent rejects for a variety of sensor values. For example, they will be able to estimate the effects of an increase in a temperature reading, perhaps in combination with a decrease in humidity. Our eventual goal is closed-loop control by combining model learning algorithms with methods for directly controlling the process.

Acknowledgments

John Doleac, Bob DaSilva, Paul Feltri, Dominic Checca, and Niru Patel were instrumental in building the original CIMSIM. Wendy Chow identified an error in the nonlinear NADALINE algorithm. John Vittal and Oliver Selfridge helped improve the clarity of this report.

References

Anderson, C., Franklin, J., and Sutton, R. (1990) The nonlinear NADALINE algorithm for connectionist networks and its application to modeling a manufacturing process. GTE TM-0242-01-90-509, GTE Laboratories Incorporated, Waltham, MA.

Franklin, J., Sutton, R., and Anderson, C. (1988) Application of connectionist learning methods to manufacturing process monitoring. *Proceedings of the Third IEEE International Symposium on Intelligent Control*, Arlington, VA.

Rumelhart, D.E., Hinton, G.E., and Williams, R.J. (1985). Learning internal representations by error propagation. Institute for Cognitive Science Technical Report 8506. La Jolla, CA: University of California, San Diego. Also in Rumelhart, D.E. and McClelland, J.L. (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition, Volume 1: Foundations.* Cambridge, MA: MIT Press.

Sutton, R.S., (1988) NADALINE: A normalized adaptive linear element that learns efficiently, GTE TR-88-509.4, GTE Laboratories Incorporated, Waltham, MA.

Sutton, R.S. (1988) Learning to predict by the methods of temporal differences. *Machine Learning*, **3**, 9–44.

Tesauro, G., He, Yu, and Ahmad, S. (1989) Asymptotic convergence of backpropagation. *Neural Computation*, **1**, 382–391.

Widrow, B. and Stearns, S.D. (1985). *Adaptive signal processing.* Englewood Cliffs, NJ: Prentice-Hall.