1992

Neural networks and exponential smoothing : a comparison via application

Harriet L. Lyons
Lehigh University

AUTHOR:

Lyons, Harriet L.

TITLE:

Neural Networks and Exponential Smoothing: A Comparison Via Application

DATE: October 11, 1992

Neural Networks and Exponential Smoothing: A Comparison Via Application

by

Harriet L. Lyons

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Industrial Engineering

This thesis is accepted and approved in partial fulfillment of the requirements for the Master of Science.

8/5/92

Date

Professor Laura Burke
Thesis Advisor

Professor Bob Storer
Co-Advisor

Chairman of Department

Acknowledgments

I would like to express my sincere gratitude to my advisor, Professor Laura Burke, for her advice, encouragement and guidance. I would also like to thank my co-advisor, Professor Bob Storer, for his suggestions, criticism, and comments. It was her insight into neural networks and his knowledge of statistics together that gave direction to my work. Both were very generous with their time, and we worked well together as a team. I look forward to any future involvement with both of them.

Special thanks to my husband, parents and brother for all their support, both emotional and motivational.

Table of Contents

List of Figures

List of Tables

Abstract

Neural network technology is receiving increasingly more attention, and is being applied to a growing diversity of problems. This paper will focus on the application of backpropagation to forecasting customer demands for a component produced by Air Products and Chemicals, Inc. One network will be created to predict demand for each customer; then a second network will be created which will predict the total demand for the component. The networks will be compared to an existing system which uses exponential smoothing. Issues involving overtraining will be discussed.

Neural Network: "is a parallel, distributed information processing structure consisting of processing elements (which can possess a local memory and can carry out localized information processing operations) interconnected via unidirectional signal channels called connections. Each processing element has a single output connection that branches ("fans out") into as many collateral connections as desired: each carries the same signal - the processing element output signal. The processing element output signal can be of any mathematical type desired. The information processing that goes on within each processing element can be defined arbitrarily with the restriction that it must be completely local; that is, it must depend only on the current values of the input signals arriving at the processing element via impinging connections and on values stored in the processing element's local memory." [1 p.2]

"**Forecast:** to calculate or predict (some future event or condition) usually as a result of rational study and analysis of available pertinent data..."

Webster's New Collegiate Dictionary [2]

1. Introduction

Neural networks are an fairly new mathematical methodology which maps a set of inputs to a set of desired outputs. The motivational analogy of neural networks is to mimic the structure of the brain. They are being developed into useful tools, and new applications continue to be discovered. One area of study in which neural networks may play a significant role is that of forecasting.

Box and Jenkins summarize the uses of forecasting: "the use at time t of available observations from a time series to forecast its value at some future time $t + l$ can provide a basis for (a) economic and business planning, (b) production planning, (c) inventory and production control, (d) control and optimization of industrial processes." [3]

A wide variety of forecasting tools are being used at Air Products and Chemicals, Inc.[1] However, neural networks are just being explored.

To solve one of it's business problems, Air Products requires two month forecasts of customer demand for a certain

---

[1] Air Products and Chemicals, Inc. is a Fortune 200 Company which supplies industrial gases and chemicals, process equipment, specialty gases and chemicals, and a wide variety of services to customers world-wide.

component. Currently, Air Products is using a software package that utilizes Winters' Exponential Smoothing to generate the forecasts. The software picks the best parameters for each customer, and periodically reviews the model to make sure it is still appropriate. We are always looking for better forecasts, and we believed that neural network technology might be able to improve upon the current system.

2. Background on the Existing System

The existing system was purchased from American Software Institute, and is simply called ASI [4]. It takes 36 months of historical demand for each customer, and returns forecasts for 18 months into the future. Although ASI generates 18 months of forecasts, for the business problem necessitating the forecasts, we are only interested in the first two of these. When a new customer is entered into the system, ASI picks the best parameters out of a given set. These parameters include smoothing constants for seasonality and trend and permanent components, to name only a few. These parameters are kept for each customer. Every few months, these parameters are checked for continued suitability. There are roughly 20 parameters per customer used by the software to generate forecasts.

ASI works with a pyramid structure as shown in Figure 1. ASI first creates forecasts for each customer; then it totals the historical demand for each region and creates forecasts for each region. Then it totals the historical demand for all the customers and creates a forecast for the total demand. Since a better forecast can usually be generated for the total demand (which is smoother and more well behaved), ASI "forces" the high level forecasts "down" to the lower levels (i.e. region and then customer). This is done by multiplying the lower forecasts by the total upper level forecast and dividing by the sum of all the lower level forecasts. For example, the forced forecasts for a customer with 50 units forecasted at the customer level, where the customer level forecasts for each customer in the entire region total to 2000 units, and the regional forecast is 2500, becomes (50 * 2500) / 2000 = 62.5.

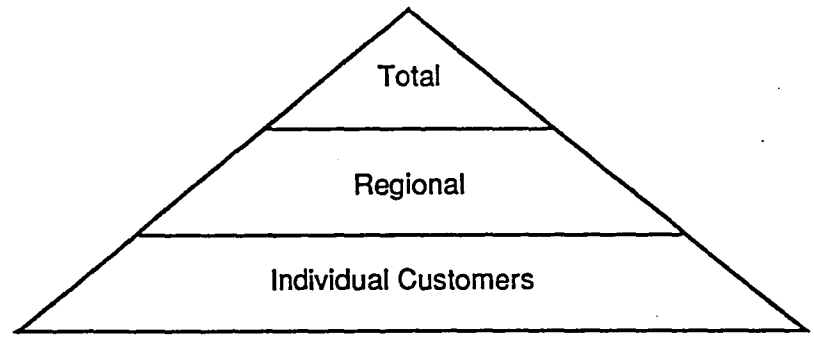


Figure 1. The ASI Pyramid

3. Introduction to Neural Networks

At this point, it is beneficial to briefly discuss neural networks. Edwin Miller, manager of the Knowledge Based Systems department at Air Products, describes a neural network concisely as "consisting of several layers of interconnected processing elements (crudely analogous to a brain's neurons), as shown below. Each element in a given layer is connected to every element in adjacent layers, and each connection has a weight representing the strength of the connection." [5] (Processing elements are also referred to as nodes or neurons.)

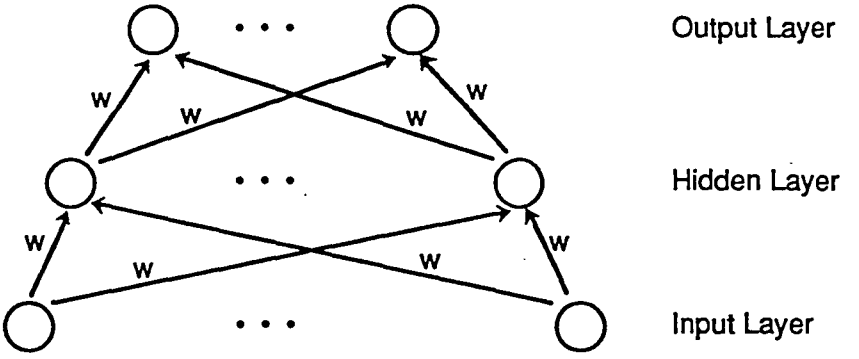


Figure 2. A 3-Layer Feedforward Neural Network. Nodes are circles; connections are lines; weights are "w".

"Training" refers to the process of repeatedly presenting the network with input/output examples to be modeled until the network can produce the desired output within a given tolerance. How to conduct training is one of the most important issues. Although there are many different neural network

paradigms (Kohonen, Hopfield, BAM, ART, to name only a few), we will only discuss backpropagation. In this paradigm, the weighted sum of the inputs to the input layer is calculated and passed on to the hidden layer(s). There a transfer function is applied to the sum, and that result is passed to the next layer. Among the transfer functions frequently used are sigmoid, hyperbolic tangent, and linear. From the output layer, we get the results of the network. These results are compared against our desired results and the error is "propagated" back through the network in the form of weight updates on the connections.

In order to facilitate convergence of the training process, a trainable "bias" may be added to each neuron. To use this feature, a connection is made between each neuron and the bias node. The weights on the new connections will change due to training, but the output of the bias node is always +1. [6]

Rumelhart, Hinton, and Williams [7] discuss *momentum*, a method to improve training time while enhancing the stability of the training process. A term that is proportional to the last weight update is added to the equation for the weight adjustments. This causes the network to follow the bottom of the error surface instead of crossing from side to side. [6]

Once training is completed, the network is asked to generalize to inputs not previously presented. On generalization, Wasserman

[6] states "Once trained, a network's response can be, to a degree, insensitive to minor variations in its input. This ability to see through noise and distortion to the pattern that lies within is vital to pattern recognition in a real-world environment. Overcoming the literalmindedness of the conventional computer, it produces a system that can deal with the imperfect world in which we live."

4. Networks to Forecast Demand for Each Customer

We began by attempting to create a neural network which would create forecasts for the individual customers. As we want to make a valid comparison with ASI, the network had only 36 input nodes (each of which represented one month of historical demand). Since we are only interested in the two step ahead forecasts, the output layer consisted of two nodes (each representing one month of forecasts). The software used was NeuralWorks Professional II/PLUS [8].

In order to decrease the size of the problem, 1000 customers were randomly selected from the entire customer base. For each customer, we had only 38 months of historical demand. No other information about the customers was kept.

4.1 Building the First Network

The input patterns consisted of all 38 demands, the first 36 months being the input nodes, and the last two, the desired output, as given in Figure 3.

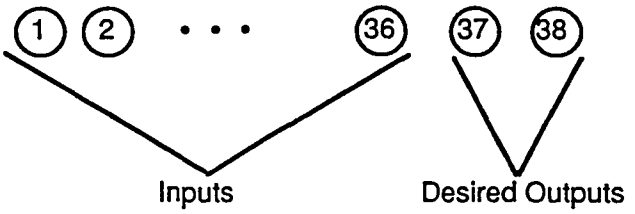


Figure 3. Format of the input patterns for customer level networks.

For all networks created, the method of creating input data files as outlined by Hecht-Nielsen was followed [1]. It involves splitting all available data into three files: the training set, test set, and validation set. The training set is used to train the network. The test set is used to prevent overtraining. And the validation set is held out until the very end; then, and only then, is it used to check the ability of the network to generalize. "The ideal criterion is that the test set be sufficiently comprehensive so that if the network performs well on it then the ultimate problem will be considered solved. In other words, the test set is constructed so that it contains essentially every possible case that will be encountered in the real world." [1 p. 115] The validation set may be kept by the customer to

facilitate final approval of the network (or an "acceptance set" may be created for this purpose). Use of these three sets increases the likelihood that the final network will be able to generalize well. All three sets must contain data that represent the distribution of the data. As we have a large quantity of data, it was broken down into the sets as follows: 50% for training, 25% for testing, and 25% for validation[2]. Therefore, we randomly selected 500 of the 1000 customers to be contained in the training set and 250 to be in each of the other two sets. Then it was verified by visual inspection of the histograms in Appendix A that all four sets (the original 1000 customers, training, test, and validation) come from roughly the same distribution.

Selection of the number of hidden units is critical because if there are too many hidden units there is a high potential for overtraining to occur. However, there are no set guidelines for the number of hidden nodes. According to several rules of thumb, we determined that 15 to 20 nodes might be appropriate.

---

[2] In a presentation at the Expert Systems and Neural Networks in Trading conference organized by International Business Communications (South Natick, MA) on January 24, 1991, Tom Schwartz, of Schwartz Inc., mentioned that another appropriate breakdown might be 85%, 10% and 5%. The appropriate breakdown depends on the amount of data available and the size of the network.

As we expected the relationships within the data at hand to be non-linear and fairly complex, we decided to deliberately exceed that range and oversize the network. We relied on the test set training method to avoid overtraining. We chose 22 hidden nodes. The architecture of this network is given in Figure 4.

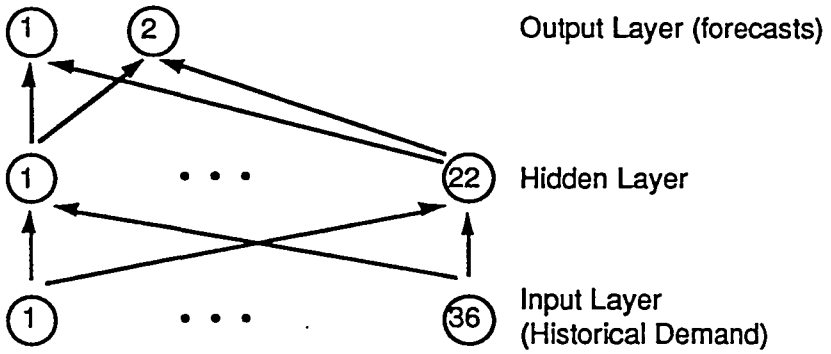


Figure 4. Architecture of the Neural Network.

This network follows the description given by Weigend, Huberman, and Rumelhart [9] of a d-n-2 network as "denoting:

- The d input units are given the values $x_{t-1}, x_{t-2}, \ldots, x_{t-d}$.
- The n non-linear hidden units are fully connected to the input units.
- The [non]-linear output unit[s are] fully connected to the hidden units, producing the prediction[s] as the weighted sum of the activations of the hidden units.
- Output and hidden units have adjustable biases.
- The weights can be positive, negative, or zero.
- There are no direct connections from input to output that skip the hidden layer."

[3] A bias node was included in all networks built. The momentum coefficient used was .4. The learning rule was the normalized cumulative delta rule with learning coefficients of .3 and .15. The transfer function used was the hyperbolic tangent:

$$T = (e^{I'} - e^{-I'}) / (e^{I'} + e^{-I'})$$

where T = value transferred out of the node

I' = weighted sum into the node multiplied by a parameter from the learning and recall schedule.

Therefore, the data were scaled between -1 and 1. This was accomplished for each node (input and output) denoted by:

$$f_1, f_2, ..., f_I, f_{I+1}, ..., f_{I+D}$$

where I = number of nodes in the input layer and D = number of nodes in the output layer. The arrays:

$$m_1, m_2, ..., m_I, m_{I+1}, ..., m_{I+D}$$

$$M_1, M_2, ..., M_I, M_{I+1}, ..., M_{I+D}$$

are found such that m_k and M_k are the smallest and largest values, respectively, for node k. NeuralWorks creates a "MinMax table" to do the scaling. [8 RF-218] A MinMax Table specifies the ranges of the real world input and output data, denoted, respectively, by

$$(r_I, R_I) \text{ and } (r_D, R_D).$$

Then the linear mappings from the real world to the network are as follows:

Input:

---

[3] These parameters are described in detail in the Reference Guide for NeuralWorks (page RF-171).

$$i_j = \frac{(R_I - r_I) \times f_j + (M_j \times r_I - m_j \times R_I)}{(M_j - m_j)}$$

Desired Output:

$$d_k = \frac{(R_D - r_D) \times f_k + (M_k \times r_D - m_k \times R_D)}{(M_k - m_k)}$$

where f = a real world value (input for i, desired output for d)

i = a scaled value input to the network

d = a desired scaled output value

g = a real world output value

o = a scaled network output

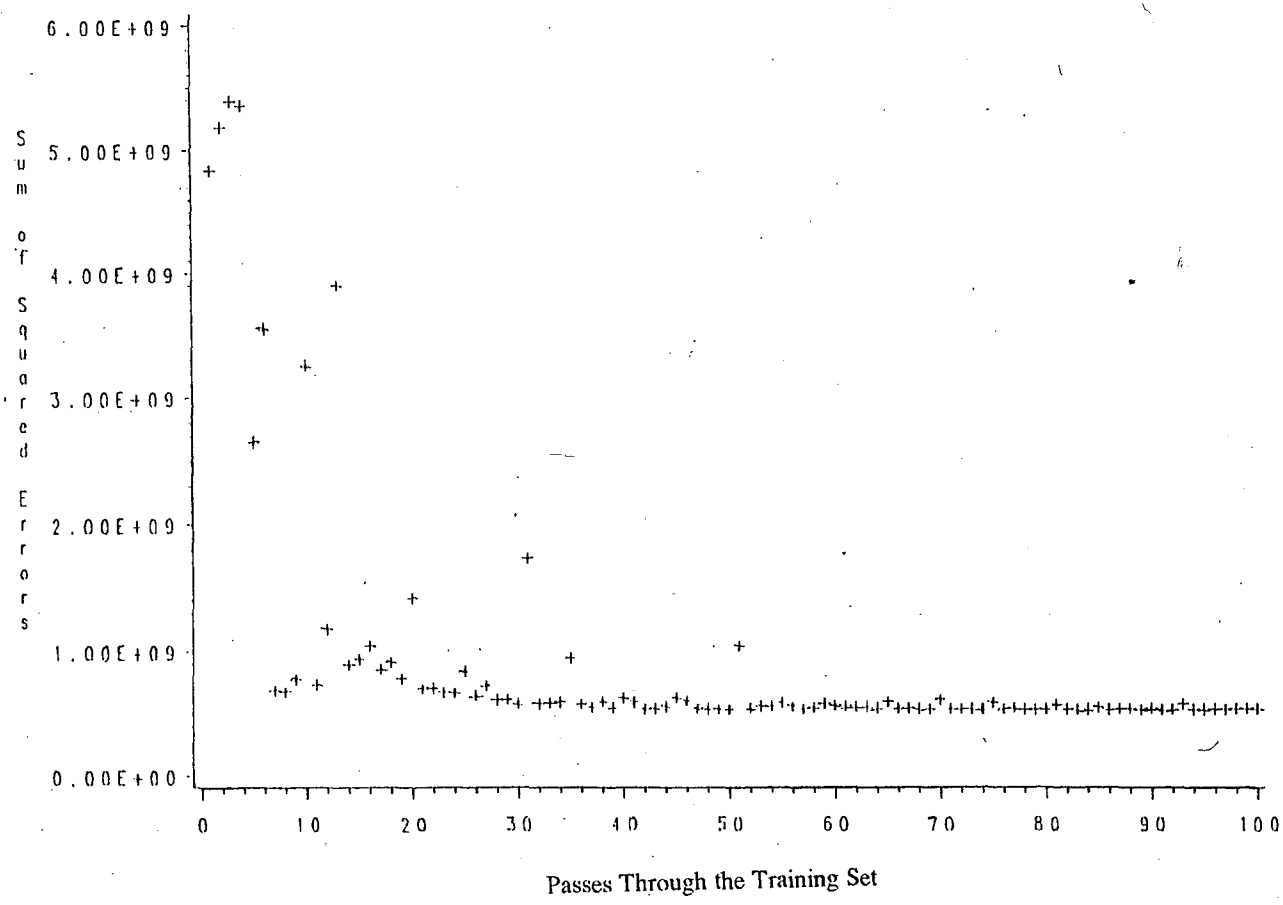
On output, the mapping from network output to real world is:

$$g_k = \frac{(M_k - m_k) \times o_k + (R_D \times m_k - r_D \times M_k)}{(R_D - r_D)}$$

Values that are outside of the MinMax Table ranges are linearly mapped using the same scale and offset values.

The training set was presented to the network once. Then the weights were frozen and the test set was presented. The training set was presented again, and so on. This cycle was completed 100 times. For each cycle, the total sum of squared errors on the test set was calculated (sse = sum ((demand1-prediction1)^2 + (demand2-prediction2)^2)). The 100 errors were plotted, yielding Figure 5.

Figure 5. Test Set Error vs. Training Cycles



As Hecht-Neilson [1] points out "the test set error typically decreases for a while, but then begins to increase again (often it eventually seems to level out, but not always).

"If we assume that the error curve of the network tested against the entire infinite set of possible examples would be approximately the same as that of the training test set curve (which is often only a crudely correct assumption) then, clearly, we want to stop training when this curve reaches its minimum. At this point, the network can be tested against the validation test set to verify adequate performance." [1 p. 116]

4.2 Results from the First Network

Although the error curve of Figure 5 does not appear to increase dramatically, it does flatten out after roughly 37 passes through the training set. Therefore, the error achieved after 37 passes was compared to the error from ASI. The total sum of squared error (sse) in ASI was 333,283,063; whereas the total sse from the network was 549,009,944, a 39% difference. As the network was not able to beat ASI, the validation set was not used. The reasoning was as follows: the point at which training was stopped was when the error curve for the test set flattened out. It is improbable that this point will be exactly the same for any other set of input patterns. Therefore, the weights produced after 37 passes through the training set might not produce the minimum error for the validation set. It is hoped

that by using the test set to attempt to safeguard against overtraining, the network will be able to generalize well, and will therefore do well on the validation set, as well. However, it cannot be expected to do as well on the validation set as it does on the test set. Since the goal is to provide a forecasting tool which provides better forecasts than ASI, there was no reason to present the validation set to a network which did not even perform better on the test set.

It is interesting to note that the raw error (actual demand - predicted demand) ((Figure 6) flattens out to zero. We would expect this because the learning process changes the weights to drive the error toward zero, as illustrated in the figure.

In order to determine where the network went awry, some further exploration of the results was conducted. Figure 7 (a and b) contains plots of the actual demand versus the demand predicted by the neural network for each of the two months being forecasted. Figure 8 (a and b) contains plots of the actual demand versus the demand predicted by ASI for each of the two months forecasted. It can be seen from these plots that ASI is better equipped to handle the few very large customers. This is so because ASI has the ability to create a distinct model for each customer, and can therefore recognize the large customers as such. As we attempted to create one network for all customers which only had historical demand as inputs, it was

Figure 6. Raw Error vs. Training Cycles

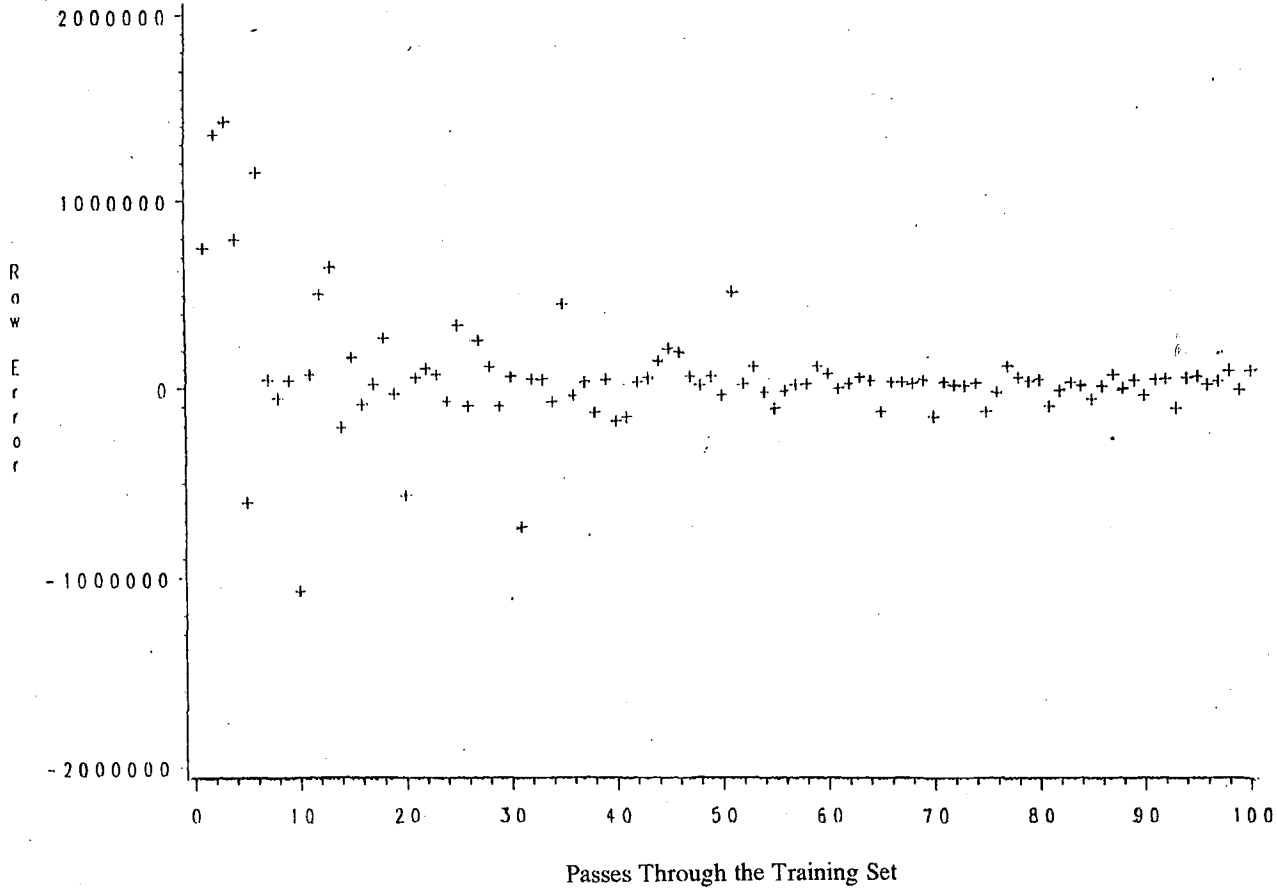


Figure 7a Actual Demand versus Neural Network Predictions

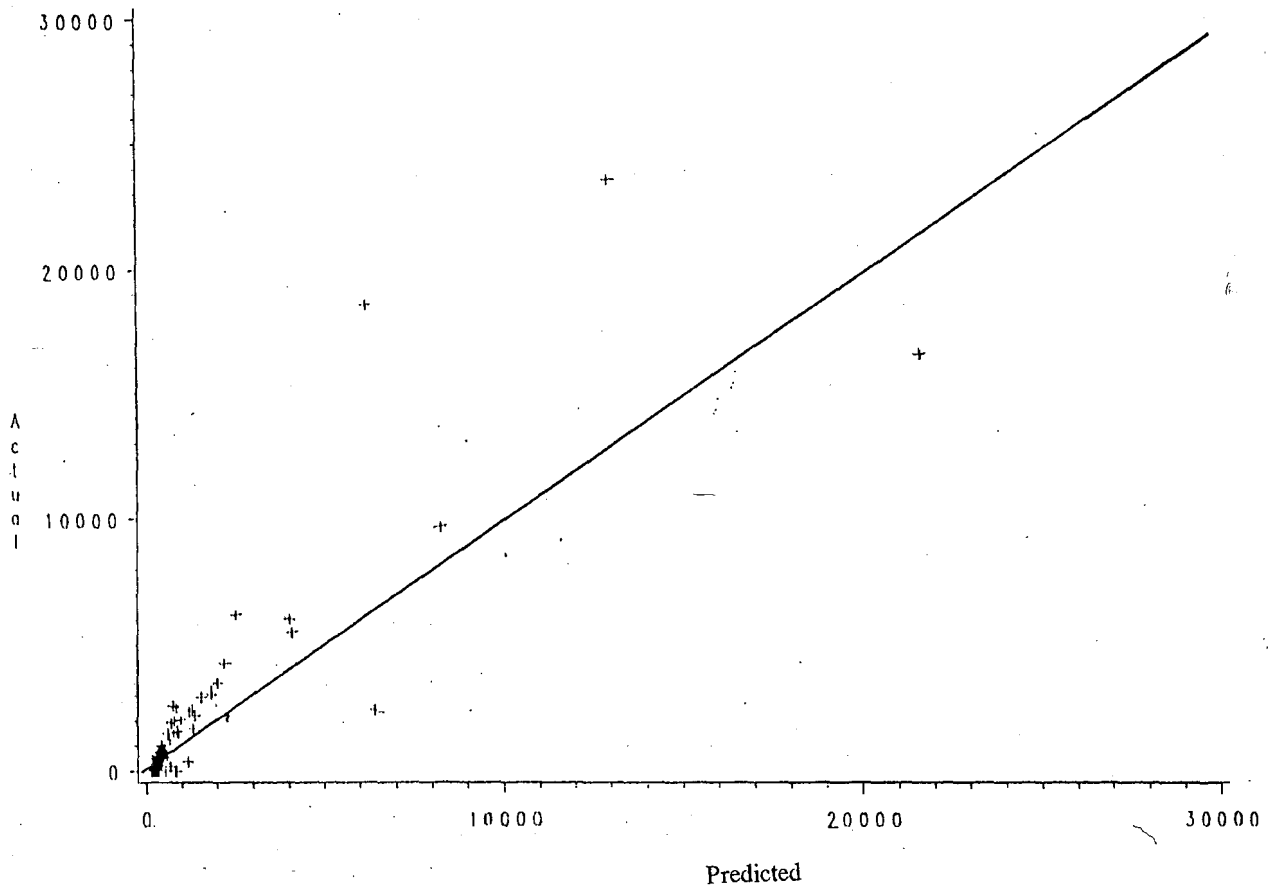


Figure 7b

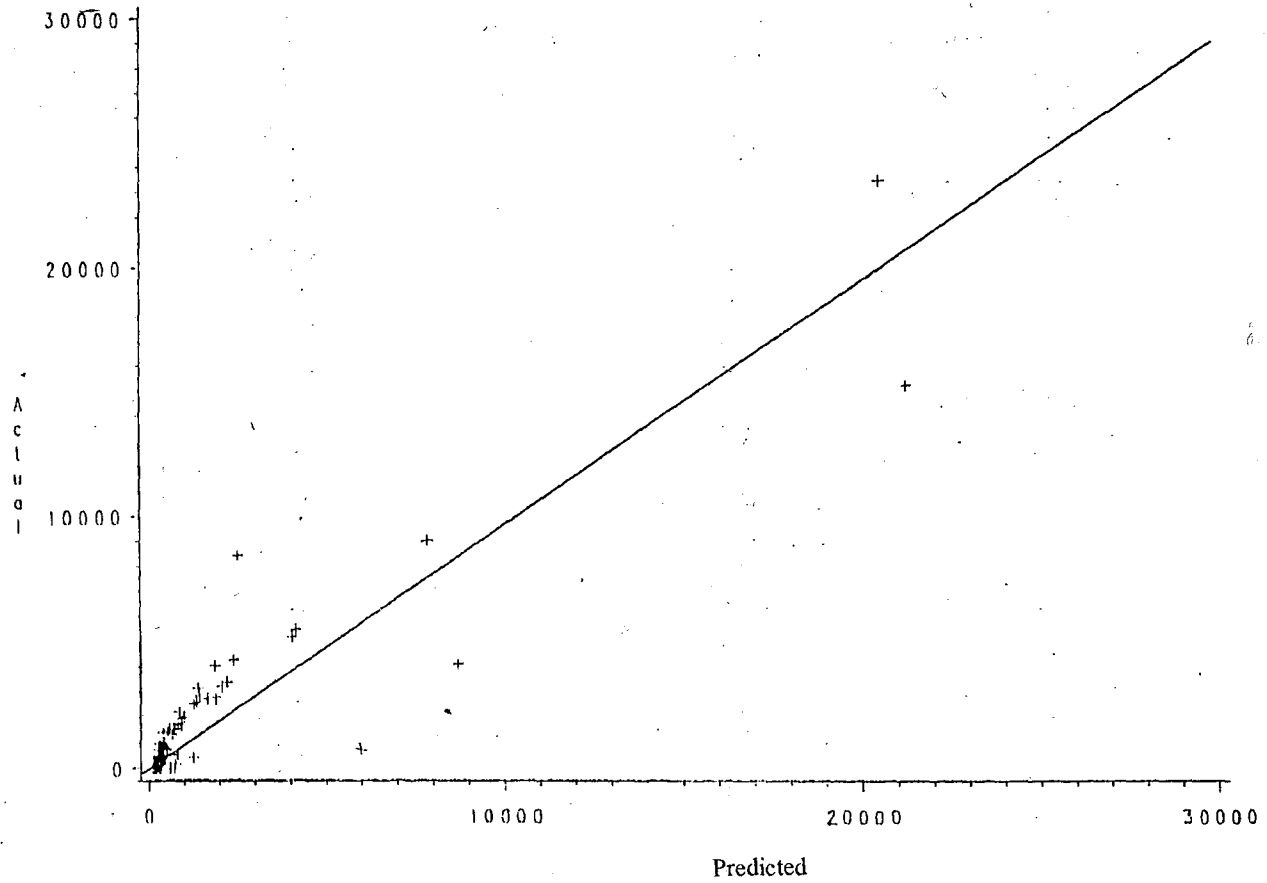


Figure 8a Actual Demand versus ASI Predictions

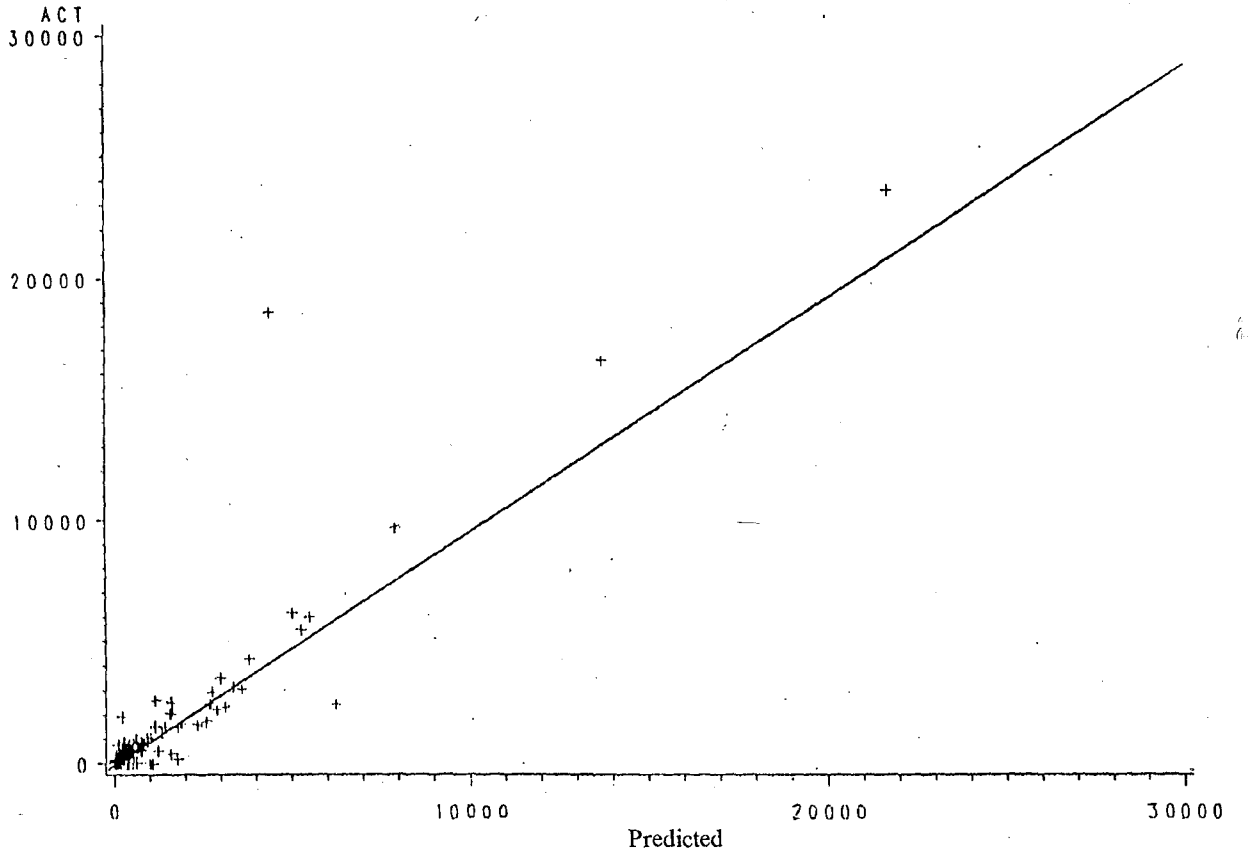
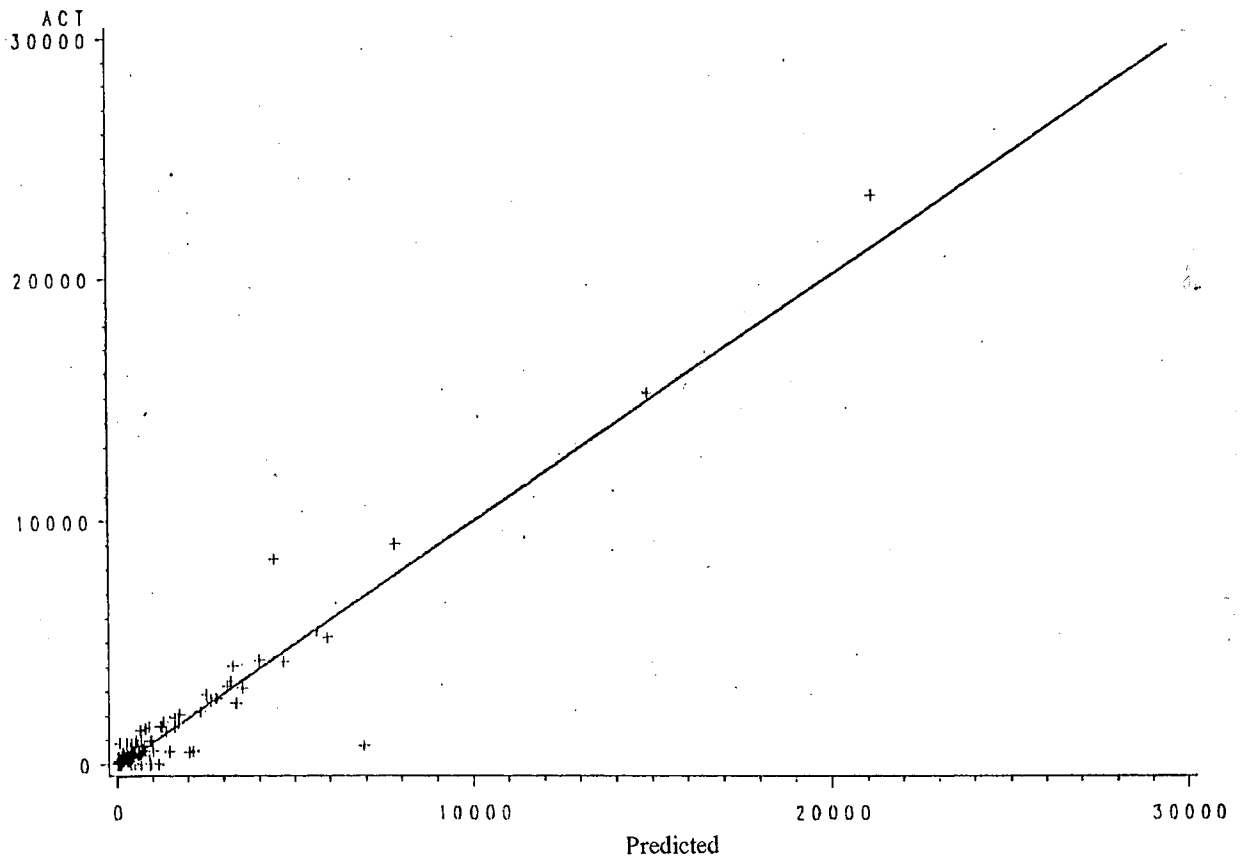


Figure 8b



not able to make that distinction. The sum of squared errors for the large customers is very big relative to the that for the small customers; therefore, this distinction is vital.

4.3 Addition of a "Demand Set Node"

To facilitate the recognition of large customers, but without any further data than the 36 months of input history, a "demand set node" was added to the input patterns. We believed that this node would provide relevant information that was contained in the given 36 months of demand history. The value for this node was calculated from the 36 months as follows:

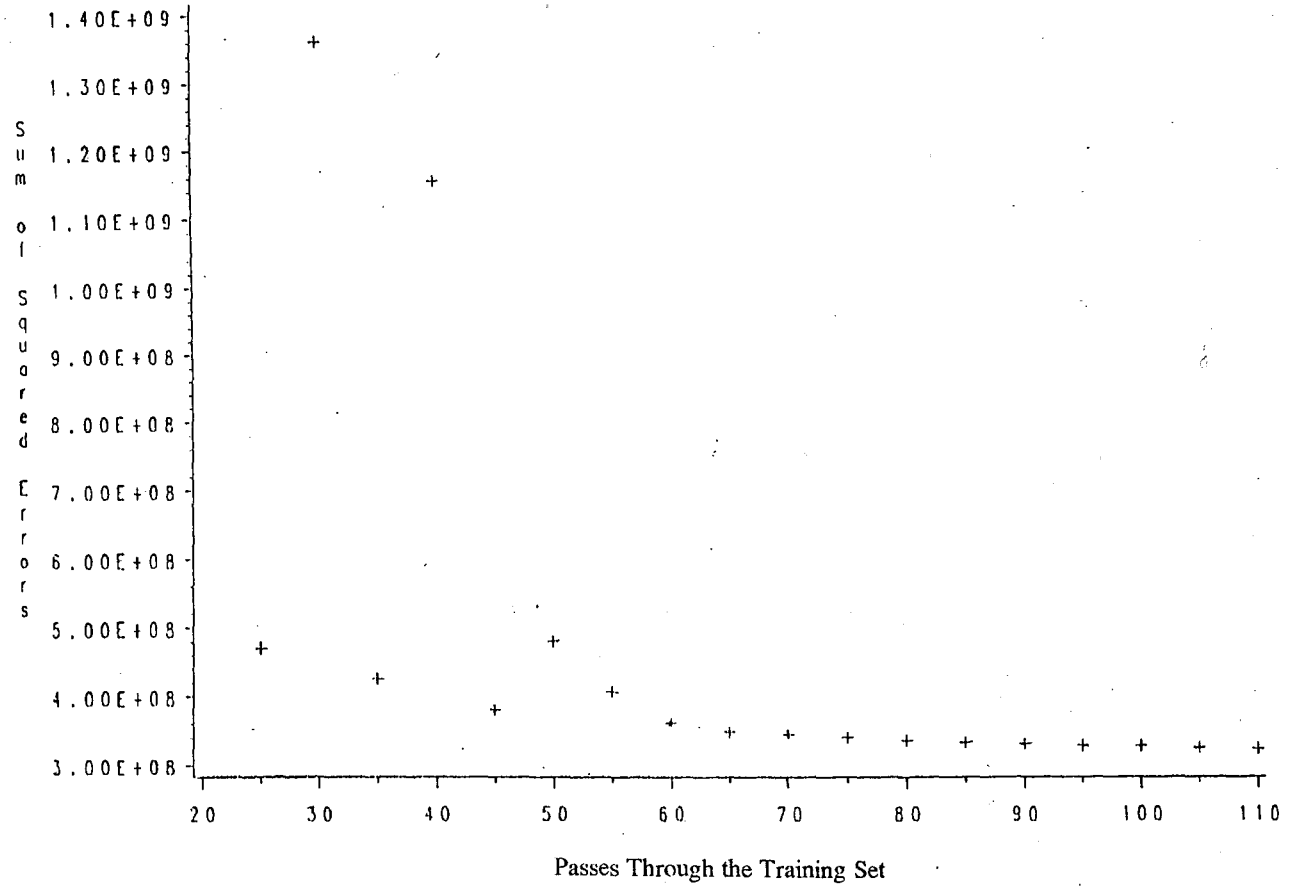
$$\lfloor (\Sigma(\text{last 6 months demand}))/ 6000 \rfloor$$

For example, the demand set node for a customer whose total demand for the last six months is 12,421 is 2, which is the average number of thousands of units demanded per month.

A new network was built using the same training, test, and validation sets used before, but with the demand set node added to the input layer. The errors on predicting the test set were monitored during training after every five cycles as previously outlined.

The number of training cycles to after which the error from the test set flattened out can be seen from Figure 9 to be 100. After 100 training cycles the total sum of squared errors was

Figure 9. Error versus Training Cycles with Demand Set Node



327,923,352. This is smaller than the error from ASI for the test set. Since we had an encouraging result on the test set, the validation set had to be presented and the error from that set compared with the error from ASI from that set. The error from the network on the validation set was 964,764,659; whereas the error from ASI on the validation set was 788,609,362 (an 18% difference). Further examination revealed that although the histograms for the validation and test sets looked similar, the validation set contained more customers with high demand one month and little the next month. This means that the network was not trained on data containing the same patterns as the test set.

4.4 A 12 Month Moving Window as Input to the Neural Network

Using a moving window of time may provide more accurate forecasts than simply using all the historical demand at once. Therefore, we divided the input patterns into moving windows of 12 months input and two desired months of output demand as illustrated in Figure 10.

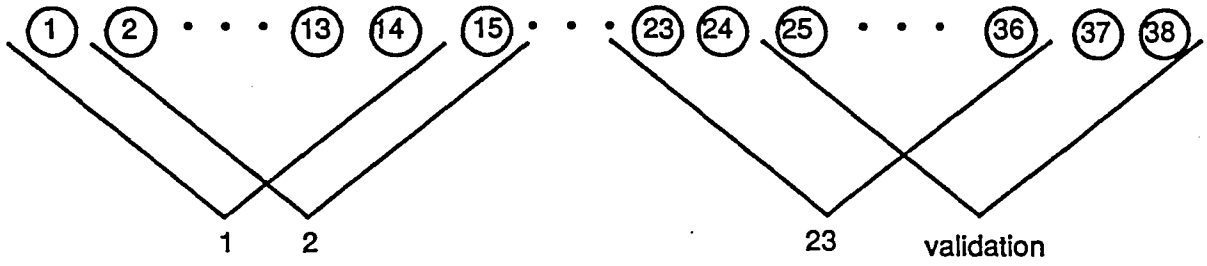


Figure 10. Moving 12 Month Windows of Historical Demand.

Since these patterns represented different periods of time we could attempt to capture potential seasonality in the data. (Examples of seasonality include when May is always a high demand month or when demand is different in the first quarter than in the second ever year, and these behavior patterns repeat over time.) In order to enable the neural network to determine seasonality, 12 (0,1)-nodes were added to the input layer. Each node represented one month of the year. The value of node j is 1 if the last month of input data represented actual demand from month j, and 0 otherwise. For instance, if the value for the 12^{th} historical demand node in Figure 11 represented the demand for May, then all the "month" nodes would be 0 except for the 5^{th} one (which would be 1) because May is the 5^{th} month of the year.

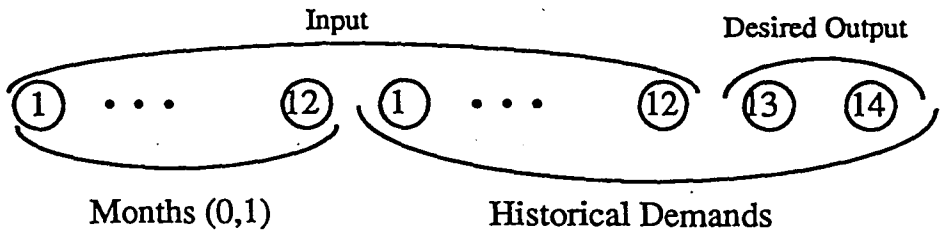
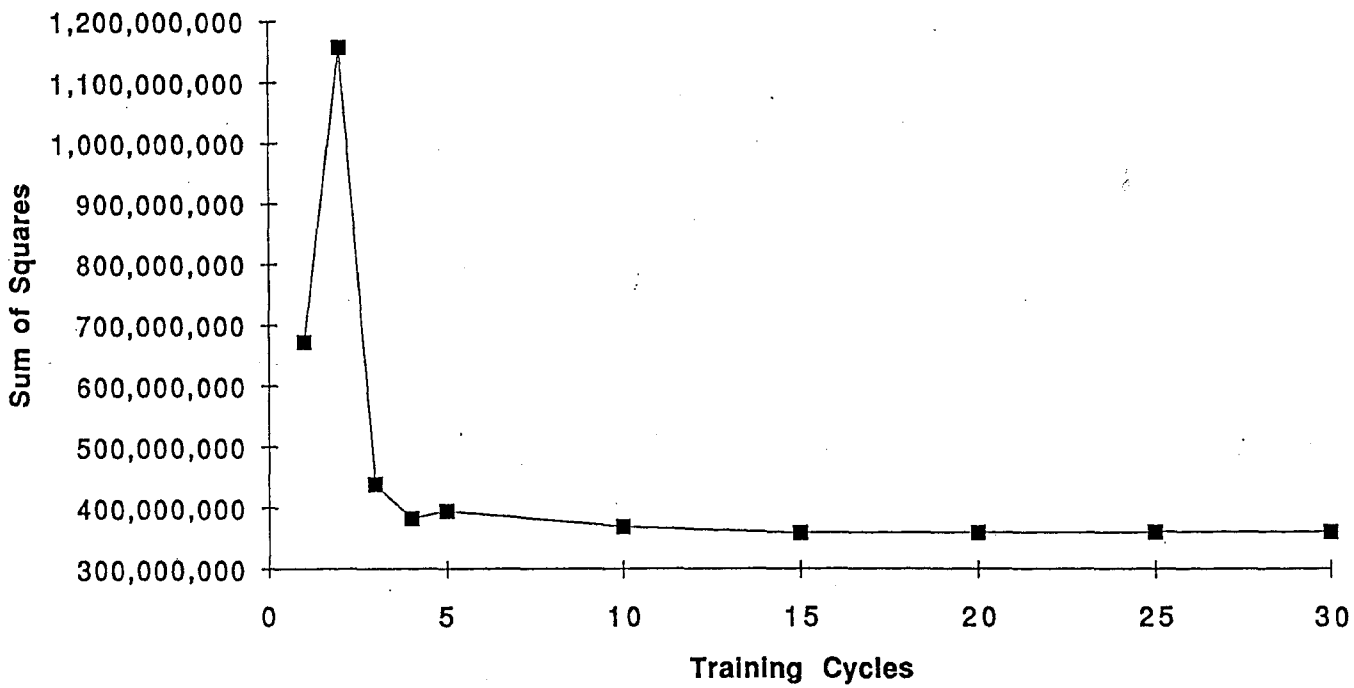


Figure 11. Input Patterns Consisting of a Moving Window of Demand and Seasonality.

The resulting training set consisted of 11,500 patterns of 24 nodes each (500 customers * 23 patterns per customer). (The validation and test sets still had 250 patterns each.)

This network was trained as outlined previously. From the error versus training cycles graph given in Figure 12 we tried to choose that point at which the curve flattens out. It is very important to recognize that there are no rules for picking such a point; however, the selection has drastic impact on the performance of the network on the test set. In this case we might pick either five or ten training cycles. If we chose to train for ten cycles through the training set, the error on the test set would be much larger than if we chose to train for five cycles. In fact, the results for five cycles are better than for ASI; whereas for ten cycles, they are worse! These results are summarized below.

Figure 12. Error versus Training Cycles with Moving Window and Seasonality



	total sse	% different from ASI
10 training cycles:	1,117,316,685	41 % worse
5 training cycles:	632,827,766	19% better
ASI:	788,609,362	

Table 1. Comparison of Sum of Squared Errors on the Test Set for Three Models

These results highlight the necessity of monitoring the network to avoid overtraining. Due to the extreme sensitivity of this problem to the amount of training, the analyst in charge of such a system would have to pick the earliest point at which the errors start to flatten out in order to obtain reasonable forecasts.

5. Forecasting Total Demand

Another very important piece of information that ASI utilizes but the customer level neural network does not is the forecast for the total demand. (Recall that ASI forecasts the total demand and then "forces" that forecast to the customers via ratios.) Therefore, it was believed that the next step should be to forecast the total demand for the next two months. The input patterns for the training set were created similarly to the previous network by taking a moving window of 14 months of

known historical demands; the first 12 of each 14 corresponded to the input layer, and the last 2 were the desired outputs. (Therefore, there were 36 - 14 +1 = 23 patterns in the training set.) The test set consisted only of the last 12 months of history and the two months of actual demand to be forecasted. Since there were so few patterns available (while still utilizing only 36 months of historical demand), a validation set was not used.

5.1 Direct Two-Step Forecasting

As in the customer level networks, the first networks built to forecast total demand predicted the two months directly. Networks were built and trained with 2, 3, 4, 5, and 6 hidden nodes. The results were the best when there were 4 hidden nodes. The resulting architecture is as in Figure 13.

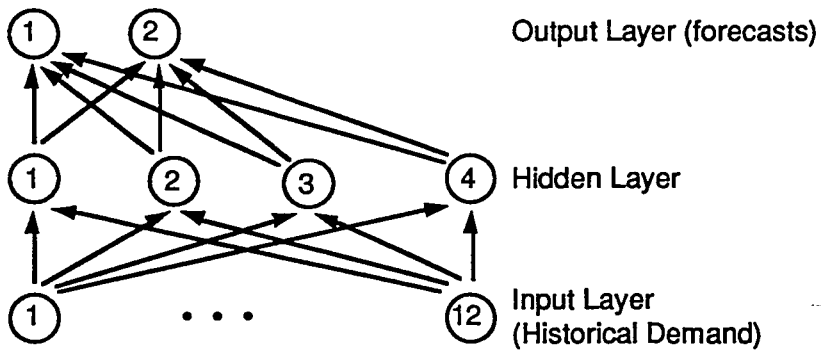
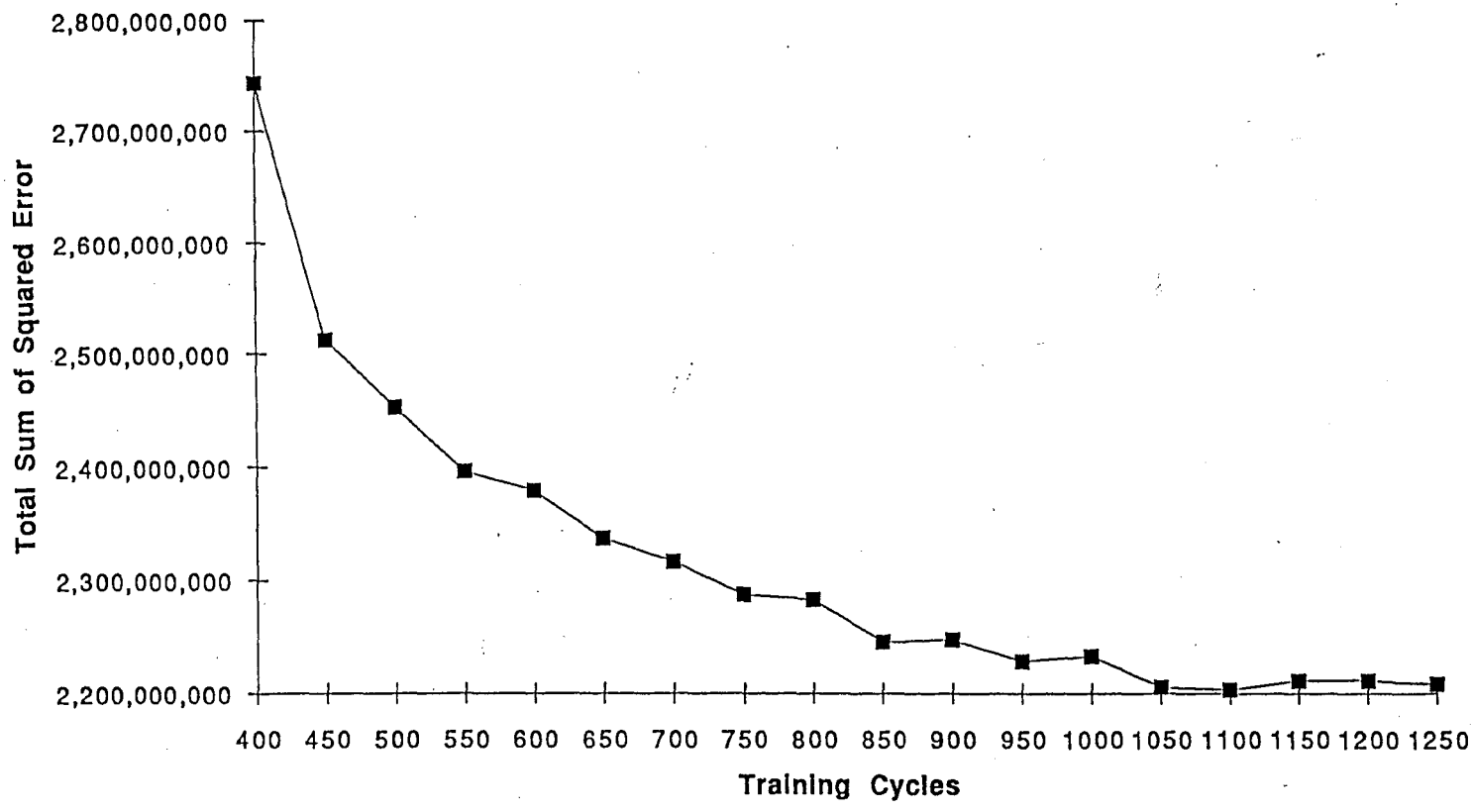


Figure 13. Architecture of the Network to Forecast Total Demand.

Figure 14. Test Set Error versus Training Cycles for Total Demand Network



We presented the training patterns to the network 50 times, and then we presented the test set. This process was iterated until the training set had been presented 1,250 times. From the graph of the error on the validation set (Figure 14), it can be seen that the error was minimized after 1,100 training cycles.

The error curve increases after 1,100 training cycles and then decreases again after 1,200 cycles. After investigation, we determined that 1,100 training cycles does not represent a local minimum; the error does not decrease below the level at 1,100 cycles.

One can see that if total sum of squared errors is the desired criteria for judging the forecasts, then ASI provides better forecasts. If, however, the criteria is absolute error, then the neural network provides better forecasts. It should be noted that the network outperforms ASI in the first month forecasted, and ASI outperforms the network in the second.

The performance of this network after 1100 training cycles as compared with ASI is summarized in Table 2.

		1st month	2nd month
Network	predicted	705,769	773,308
	actual	695,219	727,565
	absolute error	10,550	45,743
	total sse		2,203,724,549
ASI	predicted	654,588	711,368
	actual	695,219	727,565
	absolute error	40,631	16,197
	total sse		1,913,220,970
Net - ASI	absolute error	-30,081	29,546
	total sse		290,503,579
			ASI 15% better

Table 2. Comparison of Models Forecasting 2-Step Ahead Total Demand

Because the performance was so close for the two methods, we could not draw any conclusions as to which is the better model based on error statistics only.[4] Some understanding of the purpose for the forecasts would be required to decide which would be best suited. For example, if the forecasts are needed to

[4] It is important to mention that conclusions made in this chapter are based on only two data points - the two months forecasted. One logical next step will be to corroborate these findings with more data.

plan for next month, the network would be the choice; if they are needed to plan for two months out, ASI might be the choice.

5.2 Iterated Single-Step Forecasting

An alternative to direct multi-step prediction is iterated single-step prediction. The latter method generally produces more accurate forecasts [10]. Therefore, we created a neural network which forecasted one month only. The forecast was then fed back into the input layer of the network to generate the second step ahead forecast. The architecture for this network, as shown in Figure 15, was very similar to the two-step ahead network created previously. The only difference was that there is one less output node. (Four hidden units were used, as before.)

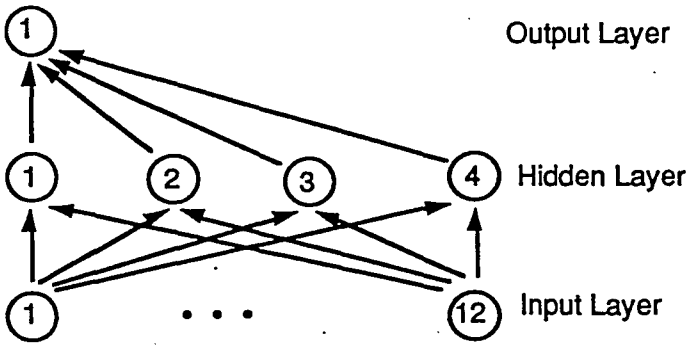


Figure 15. Architecture of One-Step Ahead Forecaster.

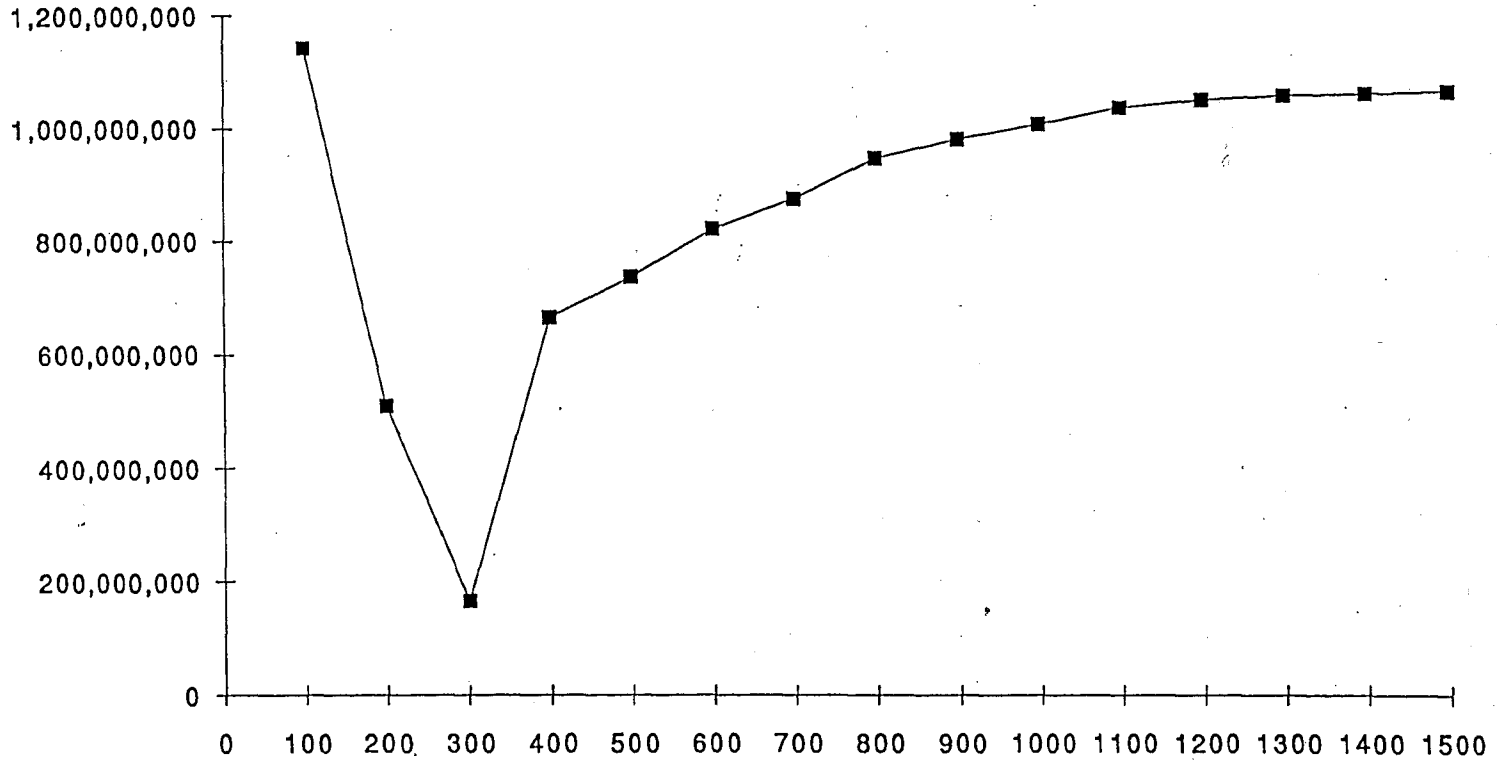
We presented the test set to the network after every 100 passes through the training set. The resulting error chart (Figure 16) has an unusual shape. As can be seen in the chart, training on this network should be terminated after roughly 300 training cycles.

This network completely outperforms ASI, as can be seen from table 3.

		1st month	2nd month
Network	predicted	682,308	731,802
	actual	695,219	727,565
	absolute error	12,911	4,237
	total sse		184,646,090
ASI	predicted	654,588	711,368
	actual	695,219	727,565
	absolute error	40,631	16,197
	total sse		1,913,220,970
Net - ASI	absolute error	-27,720	-11,960
	total sse		1,728,574,880
			Net 90% better

Table 3. Comparison of Models Forecasting 1-Step Ahead Total Demand

Figure 16. Error vs. Training Cycles for the Single Step Prediction of Total Demand

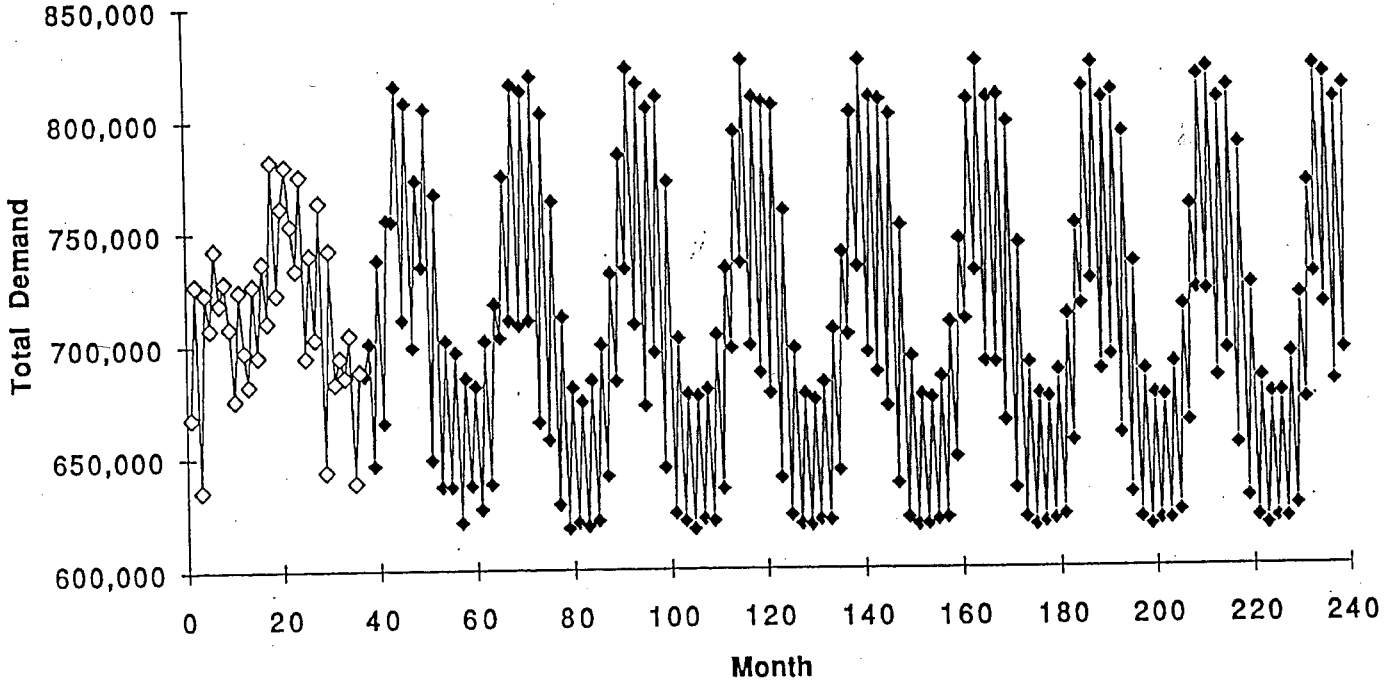


We generated the eventual forecast function in Figure 17 by iteratively using the forecast from month i as input to the network to forecast for month i + 1. (The neural network and the code to generate the eventual forecast function is given is given in Appendix B.)

As can be seen in Figure 17, the neural network discovered a periodic function which follows a sinusoidal function with a period of two years. This function looks reasonable given the input data. One possible explaination for the two year period is that the middle part of the input data cover a period of growth, but the last part of the data covers the begining of the recent economic recession. These components are used in many industries such as the automotive industry which were hit by the recession; therefore, there was a decrease in demand. The network was able to interpret the initial low demand, increase, and then decrease in demand as a two year cycle.

The neural network also determined that the demand alternates and therefore produced forecasts that alternate each month, high one month then low the next. This can be explained by noting that there are a few customers that make up a large portion of the total demand. These customers tend to place orders for two months supply at a time; therefore the demand does, in fact, alternate by month.

Figure 17. Eventual Forecast Function for Single Step Network With 4 Hidden Nodes after 300 Training Cycles.



6. Conclusion

Neural networks can generate forecasts of customer demand for the particular component of interest which exhibit lower sum of squared errors than those currently being generated at Air Products. For the customer level, one such network takes a moving window of 12 historical demands and 12 seasonality nodes as input. This network would need careful monitoring to avoid overtraining. Forecasts of total demand would be generated using iterated single-step forecasting.

Further research on this application may improve upon these results. There are several other models for the customer level that can be constructed. Instead of using a "demand set node" (which proved interesting at best), assigning a customer number in order of total demand from each customer might allow the network to differentiate the large customers even more. Experimentation on the number of hidden nodes should also yield improved forecasts. Finally, the iterated single-step method should be attempted.

Although these results are encouraging, a significant amount of work remains to implement this technology. The one critical hurdle to be overcome is the large amount of human interaction necessary to create and maintain a neural network system.

Someone must watch for overtraining, homogeneity of the sets of input patterns, appropriate parameters, convergence criteria, etc. Once the network is built, it must be monitored to determine when re-training is required. These take a considerable amount of time. These issues must be suitably addressed before a neural network forecasting system can be implemented successfully at Air Products.

References

[1] Hecht-Nielsen, Robert. Neurocomputing. HNC, Inc. and University of California. San Diego: Addison-Wesley Publishing Company, 1990. p116.

[2] Webster' New Collegiate Dictionary, Springfield, MA: G. & C. Merriam Company, 1975.

[3] Box, George E. P. and Jenkins, Gwilym M. Time Series Analysis: Forecasting and Control. Oakland. California: Holden-Day, 1976.

[4] American Software, Inc. Product Group & Item Forecasting. 470 East Paces Ferry Road, Atlanta, Georgia, 30305, (404) 261-4381.

[5] Miller, Edwin. Memorandum on 24 September 1991.

[6] Wasserman, Philip . Neural Computing: Theory and Practice. New York: Van Nostrand Reinhold, 1989.

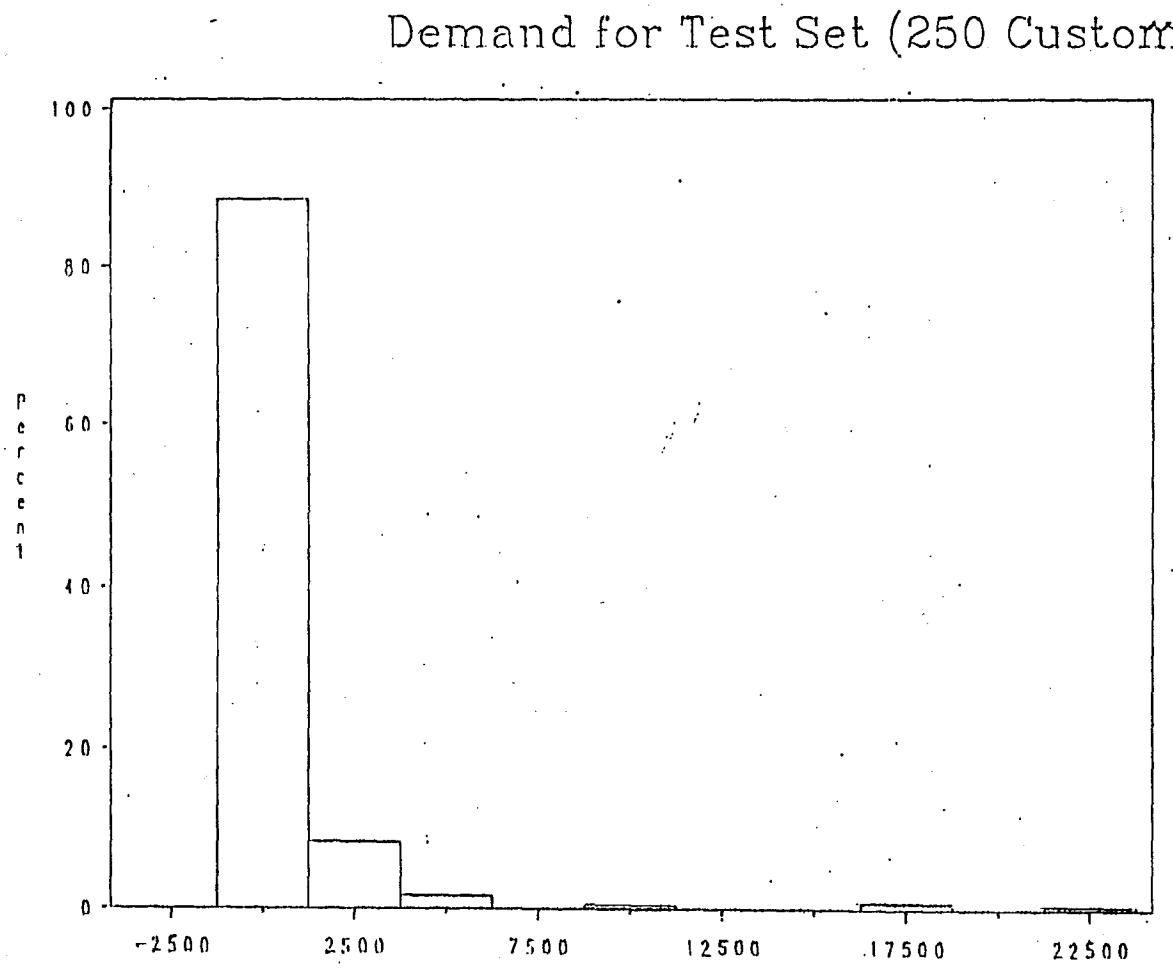
[7] Rumelhart, D. E., Hinton, G. E., and Williams, R. J., "Learning Internal Representation by Error Propagation." Parallel Distributed Processing. Vol 1. Cambridge, MA: MIT Press, 1986.

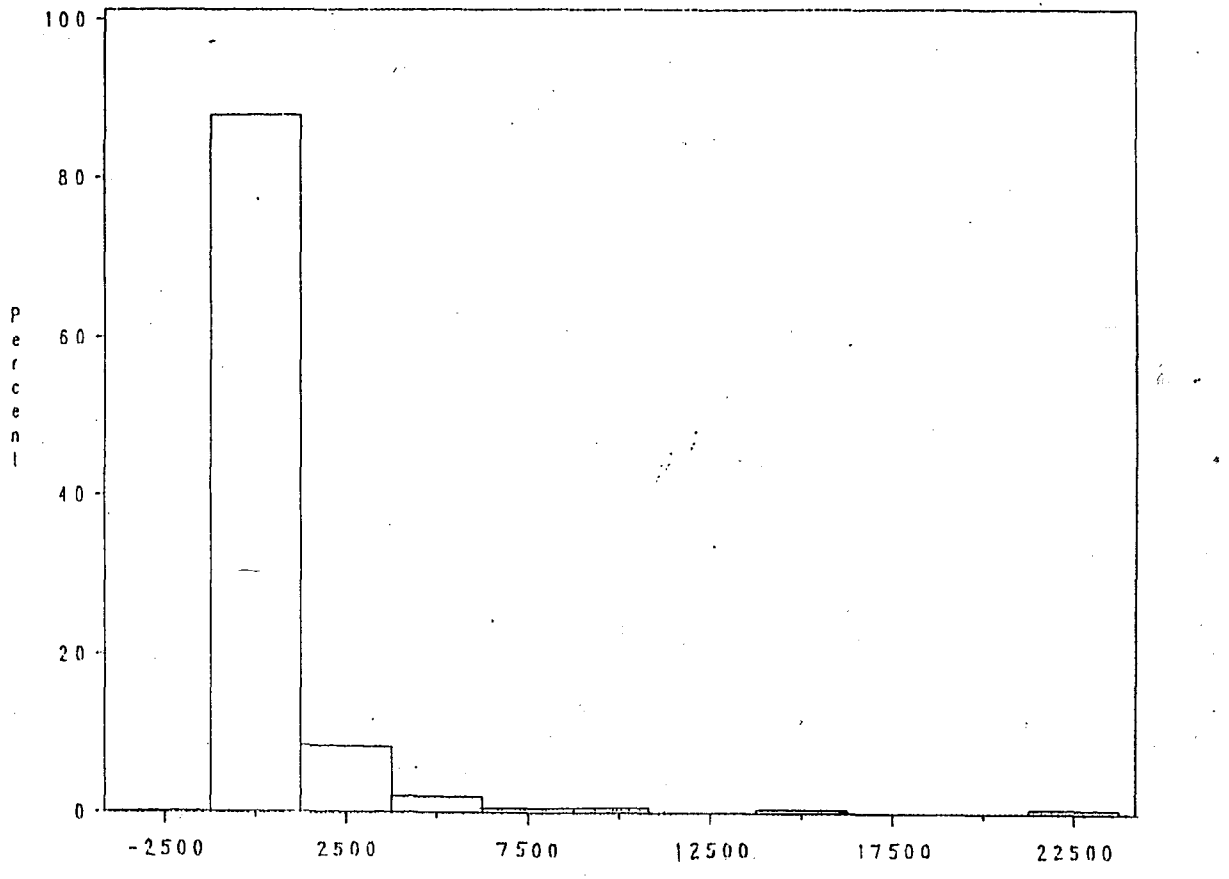
[8] NeuralWorks Professional II/PLUS, NeuralWare, Inc., Penn Center West, Building IV, Suite 227, Pittsburgh, PA 15276, (412) 787-822.

[9] Weigend, Andreas S., Huberman, Bernardo A., and Rumelhart, David E. "Predicting the Future: A Connectionist Approach", International Journal of Neural Systems. Vol 1, number 3, 1990, p193.

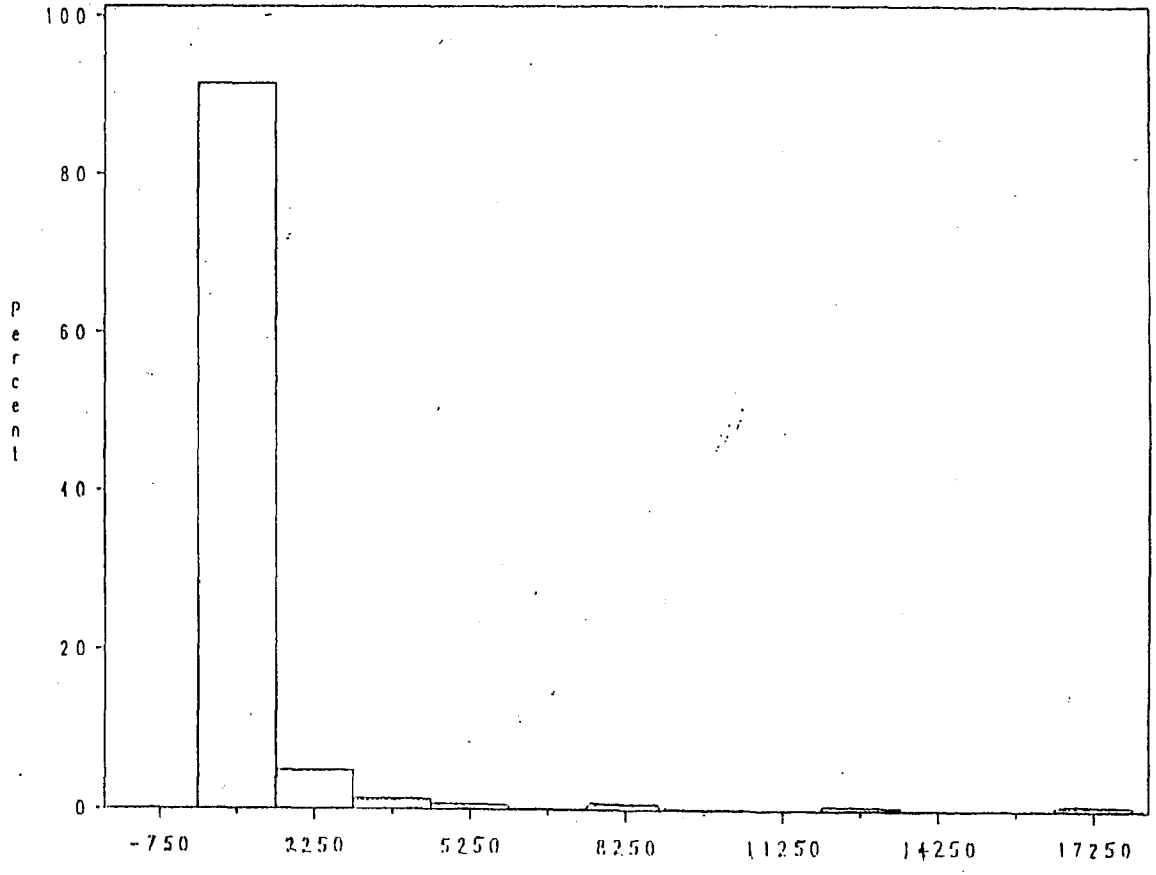
[10] Farmer, D. and Sidorowich, J., "Exploiting Chaos to Predict the Future and Reduce Noise", Evolution, Learning, and Cognition, Edited by W. C. Lee, Singapore: World Scientific Publisher, 1988, p 277.

APPENDIX A. Histograms of Sets of Data
(Two Histograms are given for each set - one for each month forecasted)

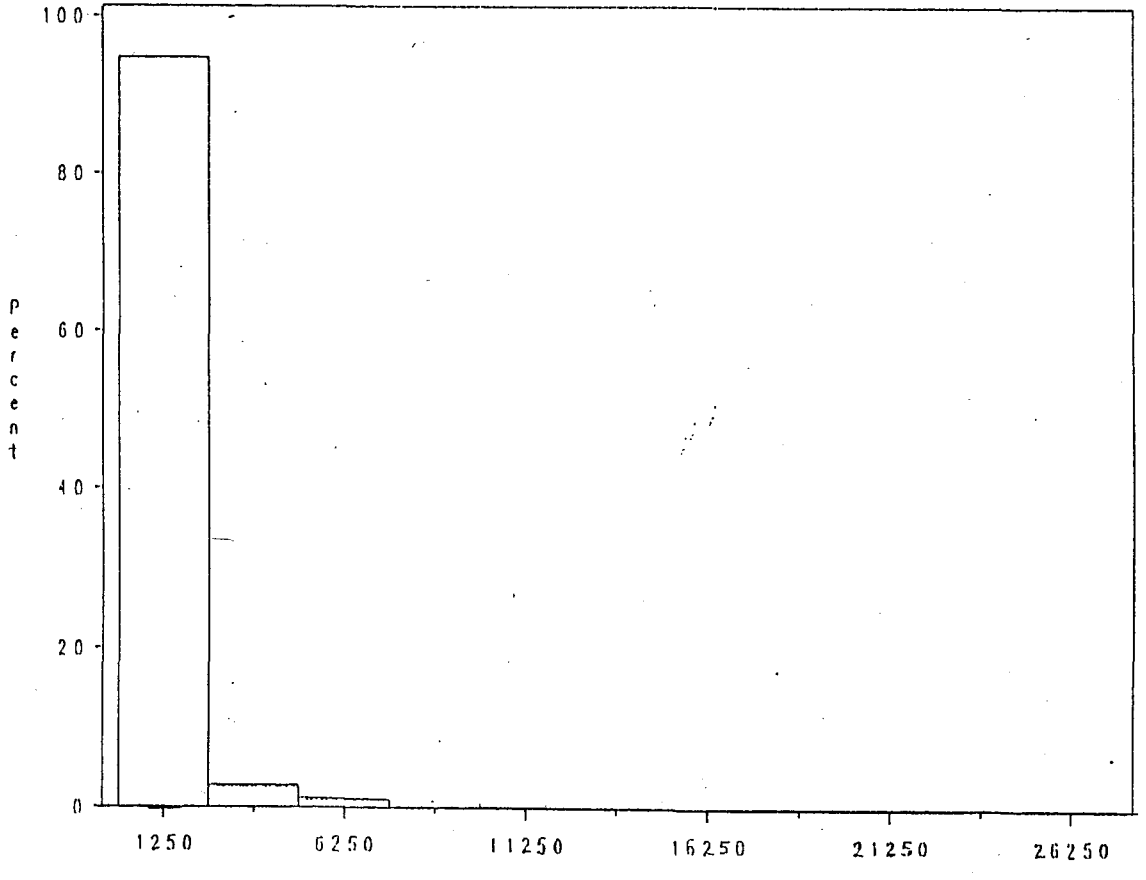


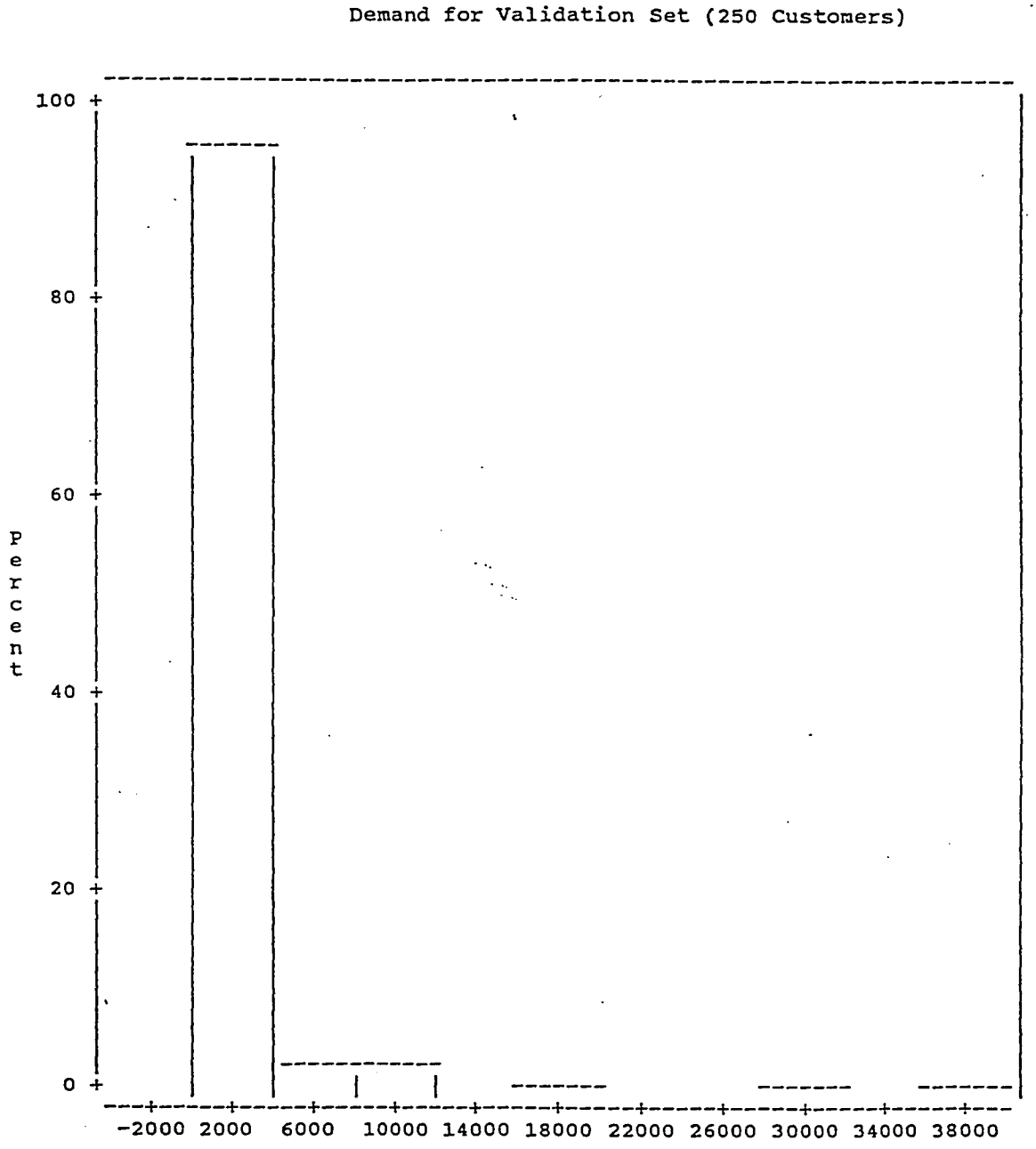

Demand for Test Set (250 Customers)
Percent
100
80
60
40
20
0
-2500
2500
7500
12500
17500
22500

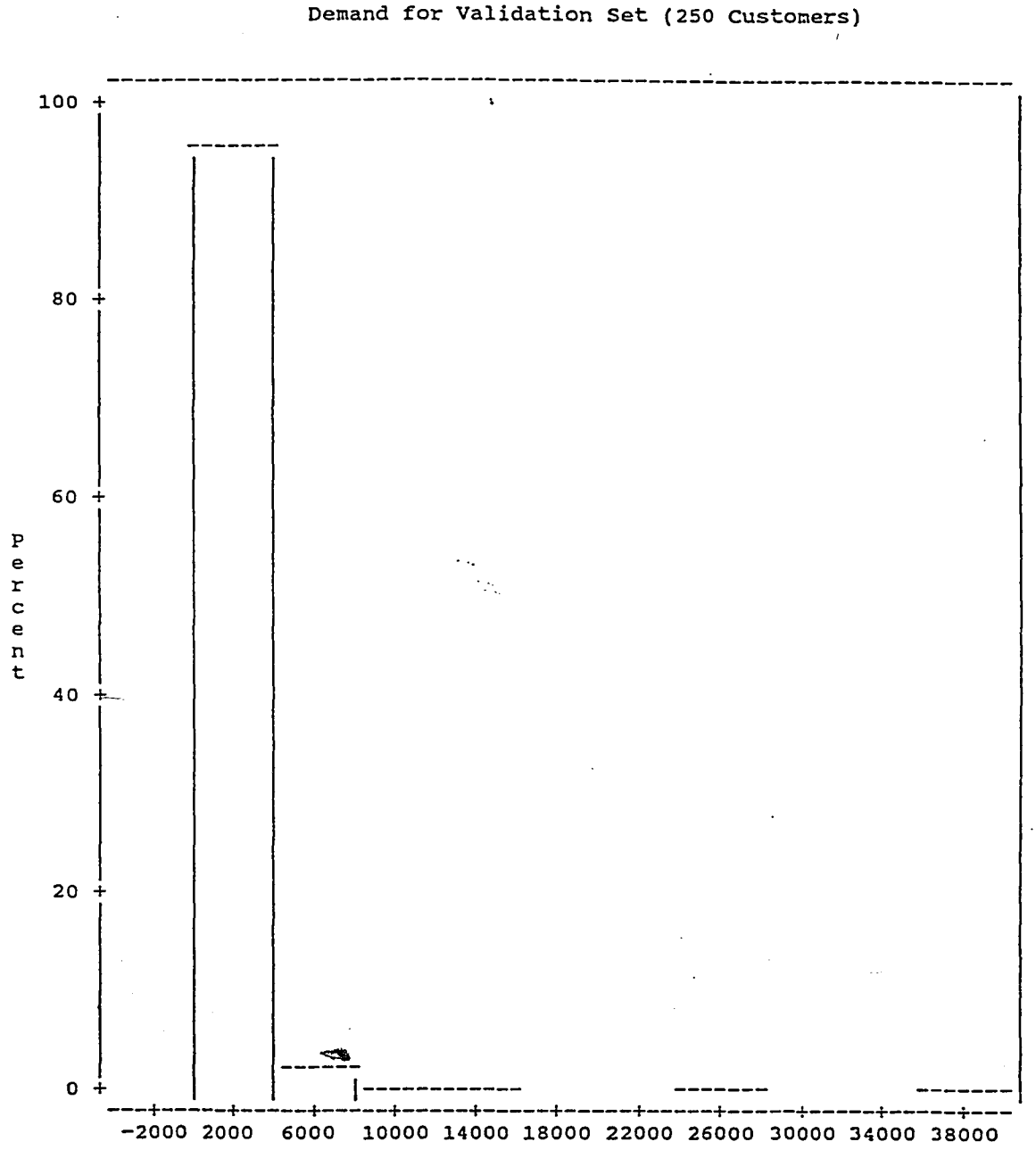
Actuals for Training Set (500 Customers)

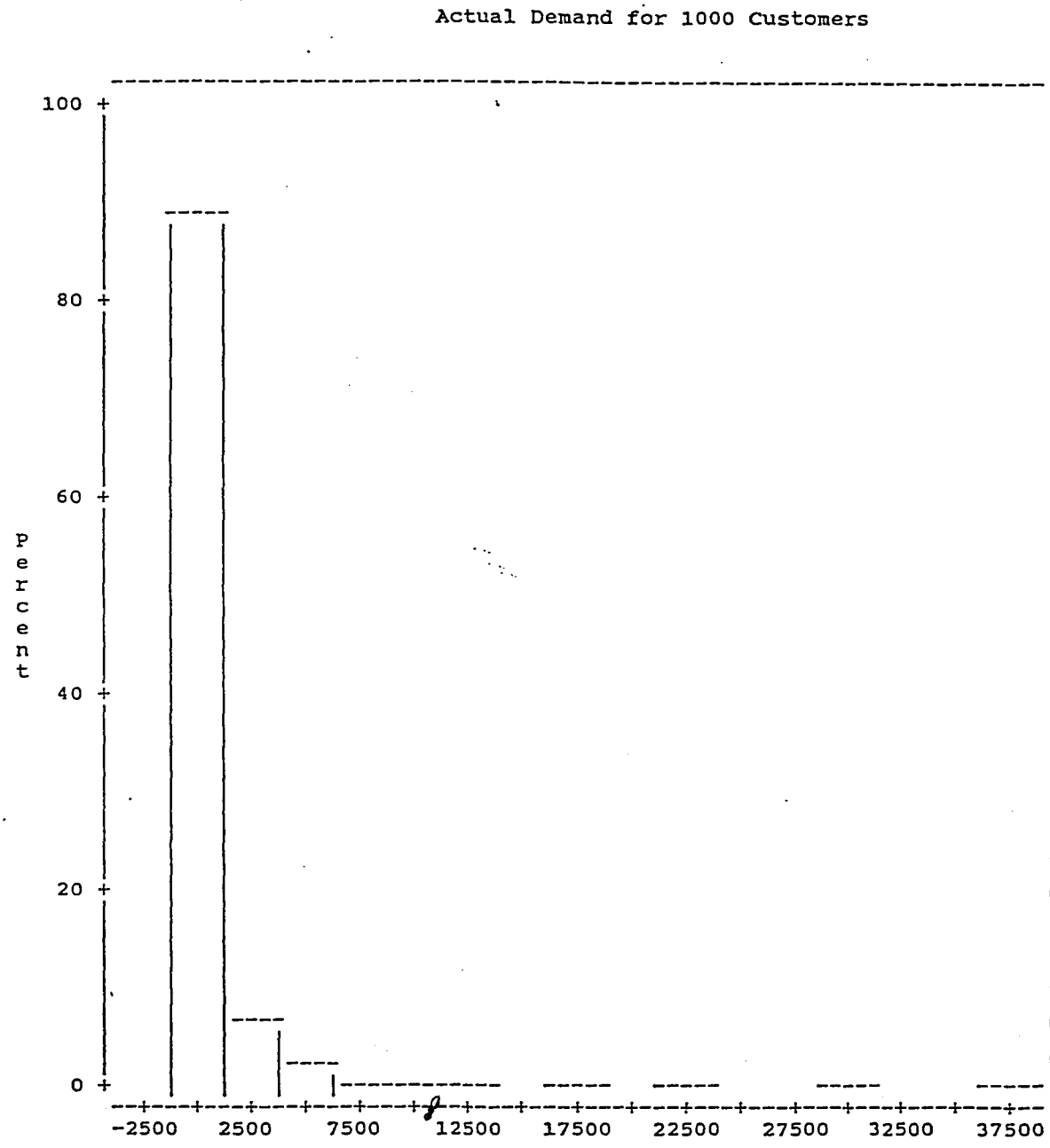


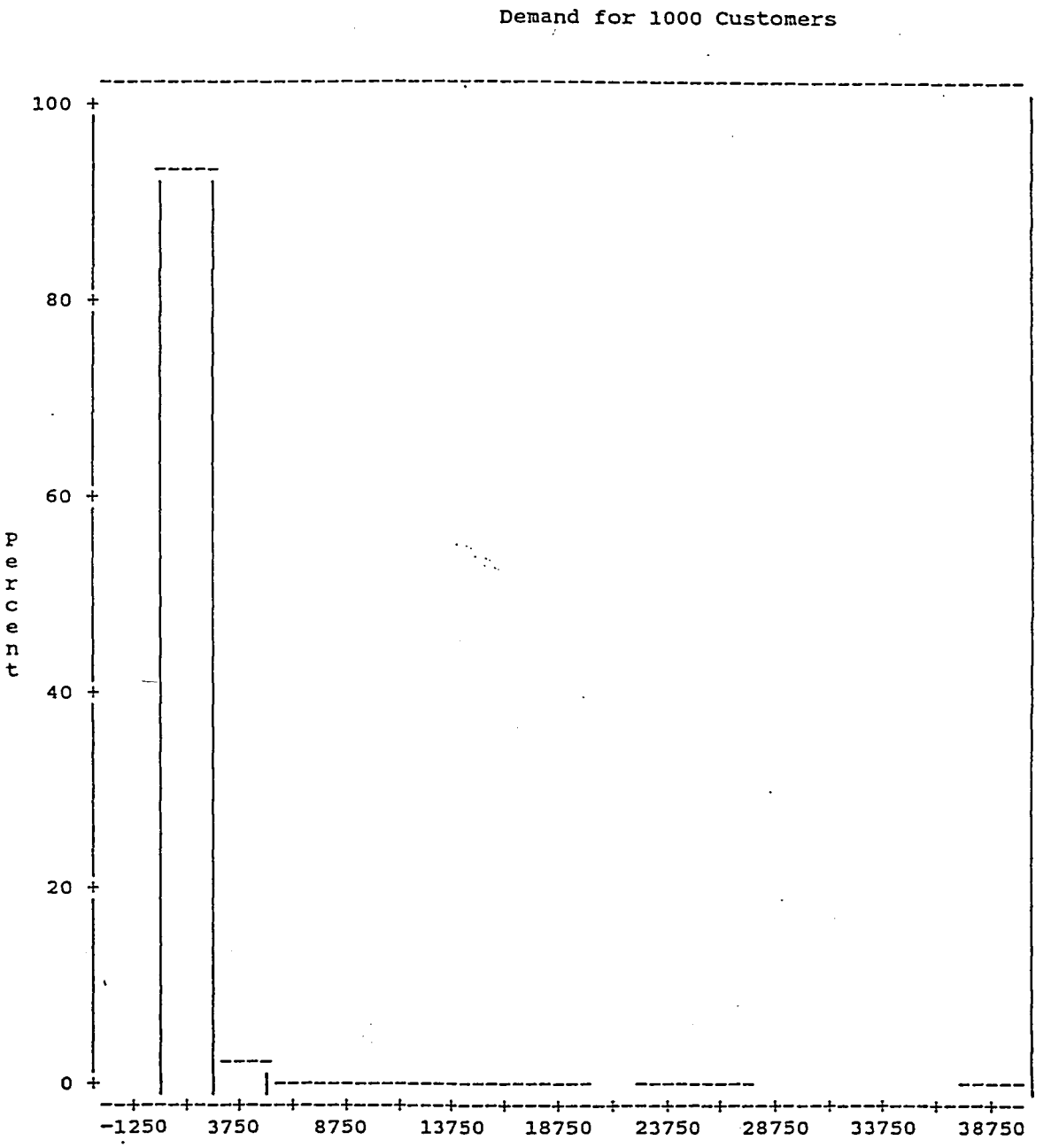
Actuals for Training Set (500 Customers)




Demand for Validation Set (250 Customers)
Percent
100
80
60
40
20
0
-2000 2000 6000 10000 14000 18000 22000 26000 30000 34000 38000


Demand for Validation Set (250 Customers)
100
80
60
40
20
0
percent
-2000
2000
6000
10000
14000
18000
22000
26000
30000
34000
38000


Actual Demand for 1000 Customers
percent
100
80
60
40
20
0
-2500
2500
7500
12500
17500
22500
27500
32500
37500


Demand for 1000 Customers
Percent
100
80
60
40
20
0
-1250
3750
8750
13750
18750
23750
28750
33750
38750

Appendix B: The iterated One-Step Neural Network to Forecast Total Demand and the Eventual Forecast Function

```
*** SAS code version of the one-step ahead neural ***;
*** network forecaster.  This code also generates ***;
*** the 20 year eventual forecast function.  (The ***;
*** plot is not generated here.)                  ***;
*** Created July, 1992  by Harriet L. Lyons.      ***;

** 1 is bias, 2-13 are input nodes  **;
Data history;
  array z(280) zz1-zz280;  *** z's = time series;
    z(1)=   667689;
    z(2)=   726930;
    z(3)=   635339;
    z(4)=  723151;
    z(5)=  707525;
    z(6)=  742597;
    z(7)=  718580;
    z(8)=  727984;
    z(9)=  707960;
    z(10)=  675755;
    z(11)= 724108;
    z(12)= 697460;

data history;  set history;
  array z(280) zz1-zz280;
  array Yin(13) YY1-YY13; ** unscaled inputs ;
  array Xout(20) xxout1-xxout20; ** all scaled values;
    Yin(1)= z(1) ;
    Yin(2)= z(2) ;
    Yin(3)= z(3) ;
    Yin(4)= z(4) ;
    Yin(5)= z(5) ;
    Yin(6)= z(6) ;
    Yin(7)= z(7) ;
    Yin(8)= z(8) ;
    Yin(9)= z(9) ;
    Yin(10)= z(10) ;
    Yin(11)= z(11);
    Yin(12)= z(12);

do I = 1 to 240;   *** forecast for 3 years ;
  do j=1 to 12;
    Yin(j) = z(j+i-1);
  end;
```

```
   ** Read and scale input into network *;
   ** x2 is input 1, x13 is input 12    *;

Xout(2) = Yin(1) * .00000969+-6.9593569;
Xout(3) = Yin(2) * (1.0537741e-005)+(-7.6550049);
Xout(4) = Yin(3) * (1.0537741e-005)+(-7.6550049);
Xout(5) = Yin(4) * (1.0537741e-005)+(-7.6550049);
Xout(6) = Yin(5) * (1.0537741e-005)+(-7.6550049);
Xout(7) = Yin(6) * (1.0537741e-005)+(-7.6550049);
Xout(8) = Yin(7) * (1.0537741e-005)+(-7.6550049);
Xout(9) = Yin(8) * (1.0537741e-005)+(-7.6550049);
Xout(10) = Yin(9) * (1.0752977e-005)+(-7.8317858);
Xout(11) = Yin(10) * (1.0752977e-005)+(-7.8317858);
Xout(12) = Yin(11) * (1.0752977e-005)+(-7.8317858);
Xout(13) = Yin(12) * (1.0752977e-005)+(-7.8317858);


** Generating code for PE 0 in layer 3 *;
Xout(14)  = (-0.065512054)+(-0.65304154) * Xout(2)+
    (-0.28332129) * Xout(3)+(-0.22807129) * Xout(4)+
    (0.43668315) * Xout(5)+(-0.43300751) * Xout(6)+
    (0.034910686) * Xout(7)+(0.0474988) * Xout(8)+
    (0.0059591993) * Xout(9)+(0.35130057) * Xout(10)+
    (0.12700793) * Xout(11)+(0.87453079) * Xout(12)+
    (0.33095482) * Xout(13);

Xout(14) = tanh( Xout(14) );

** Generating code for PE 1 in layer 3 *;
Xout(15)  = (0.072443113)+(0.062924318) * Xout(2)+
    (-0.023038119) * Xout(3)+(-0.20059879) * Xout(4)+
    (-0.38352284) * Xout(5)+(0.22233321) * Xout(6)+
    (0.031286795) * Xout(7)+(-0.2147585) * Xout(8)+
    (-0.079028048) * Xout(9)+(0.18169956) * Xout(10)+
    (-0.32272851) * Xout(11)+(-0.040567964) * Xout(12)+
    (-0.13757579) * Xout(13);

Xout(15) = tanh( Xout(15) );

** Generating code for PE 2 in layer 3 *;
Xout(16)  = (0.51086879)+(-0.14268588) * Xout(2)+
    (0.24350415) * Xout(3)+(-0.1280665) * Xout(4)+
    (-0.28067172) * Xout(5)+(0.48116192) * Xout(6)+
    (-0.45402175) * Xout(7)+(0.50139827) * Xout(8)+
    (0.27197465) * Xout(9)+(0.073758774) * Xout(10)+
    (0.60602796) * Xout(11)+(-1.0222355) * Xout(12)+
    (-0.34886909) * Xout(13);

Xout(16) = tanh( Xout(16) );
```

```
** Generating code for PE 3 in layer 3 *;
Xout(17)  = (-0.56164324)+(-0.76799136) * Xout(2)+
    (0.91546154) * Xout(3)+(-0.3789292) * Xout(4)+
    (1.0022947) * Xout(5)+(-0.14396407) * Xout(6)+
    (0.21785158) * Xout(7)+(-0.82619166) * Xout(8)+
    (0.017511081) * Xout(9)+(0.0852044) * Xout(10)+
    (0.35464713) * Xout(11)+(1.1420908) * Xout(12)+
    (-0.048709556) * Xout(13);

Xout(17) = tanh( Xout(17) );

** Generating code for PE 0 in layer 4 *;
Xout(18)  = (-0.15891412)+(0.53716618) * Xout(14)+
    (-0.38866025) * Xout(15)+(-0.6971423) * Xout(16)+
    (-1.0497816) * Xout(17);

Xout(18) = tanh( Xout(18) );

** De-scale and write output from network *;

Yout = ((821334-635339)*Xout(18)+
          (.8*635339+.8*821334))/1.6;

 z(i+12) = Yout;
 output;
end;

 ** now the z's contain the entire time  ;
 ** series (actual and forecast).        ;

run;
```

Vita

Harriet Lyons was born on September 17, 1967 in New Haven, Connecticut, to Anne and Allan Kupferman. She graduated from Carnegie-Mellon University in Pittsburgh, Pennsylvania with a Bachelor of Science in Applied Mathematics/ Operations Research.

She is currently employed by Air Products and Chemicals, Inc. in Allentown, Pennsylvania. She spent two and one half years in the Operations Research group and is currently with the Statistical Sciences department.