MIT Sloan School of Management

Working Paper 4347-02
February 2002

USE OF RECURRENT NEURAL NETWORKS FOR STRATEGIC DATA MINING OF SALES

J. Shanmugasundarum, M.V. Nagendra-Prasad, S. Vadhavkar, A. Gupta

Use of Recurrent Neural Networks for Strategic Data Mining of Sales Information

JAYAVEL SHANMUGASUNDARAM jai@cs.wisc.edu
Dept. of Computer Sciences, University of Wisconsin, Madison, WI 53706
M.V. NAGENDRA PRASAD nprasad@verticalnet.com
Advanced Products and Strategy Group, VerticalNet, Palo Alto, CA94304
SANJEEV VADHAVKAR vada@mit.edu
MIT School of Engineering, Room 1-270, Cambridge, MA 02139
AMAR GUPTA agupta@mit.edu
MIT Sloan School of Management, Room E53-311, Cambridge, MA 02139

Abstract. An increasing number of organizations are involved in the development of strategic information systems for effective linkages with their suppliers, customers, and other channel partners involved in transportation, distribution, warehousing and maintenance activities. An efficient inter-organizational inventory management system based on data mining techniques is a significant step in this direction. This paper discusses the use of neural network based data mining and knowledge discovery techniques to optimize inventory levels in a large medical distribution company. The paper defines the inventory patterns, describes the process of constructing and choosing an appropriate neural network, and highlights problems related to mining of very large quantities of data. The paper identifies the strategic data mining techniques used to address the problem of estimating the future sales of medical products using past sales data. We have used recurrent neural networks to predict future sales because of their power to generalize trends and their ability to store relevant information about past sales. The paper introduces the problem domain and describes the implementation of a distributed recurrent neural network using the real time recurrent learning algorithm. We then describe the validation of this implementation by providing results of tests with well-known examples from the literature. The description and analysis of the predictions made on real world data from a large medical distribution company are then presented.

Keywords: Inventory Management System, Neural Networks Applications, Recurrent Neural Networks, Statistical Reasoning, Signal-to-Noise Ratio.

Use of Recurrent Neural Networks for Strategic Data Mining of Sales Information

1 Introduction

With the advent of data warehousing, companies have started storing large amounts of historical data. One way of exploiting this information is by using such data to predict the future. In this paper, we address the specific problem of predicting future sales using information about past sales. For example, accurate predictions of future sales could lead to significant savings in inventory costs. We used recurrent neural networks to predict future sales using past sales data obtained from a large retail chain dispensing pharmaceutical drugs (called *Medicorp* [1,2,15] in this paper). The results of these experiments suggest that recurrent neural networks are a good way to predict trends in sales. However, it appears that the performance on prediction of actual sales figures is limited. We hypothesize that this limitation may be mostly attributed to (a) the insufficiency of data, which makes it difficult to detect long-term dependencies and (b) the uncertainty and pronounced effects of exogenous variables in the real world data. We provide a justification for the above statement by studying the predictive performance of recurrent neural networks on noisy mathematical functions, where the noise is intended to model the effects of exogenous factors. We show that recurrent neural networks are very effective at learning functions with a small amount of noise and that the performance degrades substantially as the noise level increases. The major phases of the project involved the following. First, a general recurrent neural network was implemented. This implementation, which could easily be adapted to run in distributed environments, was done using C++. The real-time recurrent learning algorithm, with appropriate modifications to make the implementation distributed, was used. This implementation

was then tested on some standard problems from the literature. Recurrent neural networks were then used to predict some real world data provided by *Medicorp*. The nature of the predicted results were then investigated and possible explanations for these results were put forth based on the ability of recurrent neural networks to learn noisy mathematical functions.

The main contributions of the paper are

- Investigation, analysis and explanation of the predictions and results based on real world data from a large medical distribution company.
- Exploring the use of data mining techniques for strategic information systems.
- Suggestions for alternative data mining architectures, which could prove useful.

The rest of the paper is organized as follows. Section 2 describes the problem to be solved in some detail and section 3 explains why the recurrent neural network architecture was chosen over other architectures for this problem. An implementation of the recurrent neural network is described in section 4. Section 5 presents and analyzes the prediction results on *Medicorp* data. Section 6 provides the conclusions and the lessons learned from this endeavor.

2 The Problem Domain

With hundreds of chain stores and with revenues of several billion dollars per annum, *Medicorp* is a large retail distribution company that dispenses pharmaceutical drugs to customers in a number of states in the United States. Just as any other retailer in its position, *Medicorp* is forced to carry a large standing inventory of products ready to deliver on customer demand. Unsatisfied customers frequently turn to competing stores, and *Medicorp* loses potential profits in such cases.

The problem is how much quantity of each drug should be kept in the inventory at each store and warehouse. *Medicorp* incurs significant financial costs if it carries excess quantities of drugs relative to the customer demand; especially because pharmaceutical drugs have limited shelf-lives Historically, *Medicorp* has maintained an inventory of approximately a billion dollars on a continuing basis and has used traditional regression models to determine inventory levels for each of the items. The corporate policy of *Medicorp* is governed by two competing principles: minimize total inventory and achieve highest level of customer satisfaction. The former principle is not quantified in numerical terms. On the latter issue, *Medicorp* strives to achieve a 95% fulfillment level. That is, if a random customer walks into a random store, the probability of availability of a requested item is 95%. The figure of 95% is based on the type of goods that *Medicorp* carries and the service levels offered by competitors of *Medicorp* for the same items. *Medicorp* has about 1000 stores, and maintains information on what is sold, at what price, and to whom. The last piece of data has not been utilized in any inventory-modeling endeavor at *Medicorp*. After reviewing various options, *Medicorp* adopted a "three-weeks of supply" approach [1]. This approach involves a regression study of historical data to compute a seasonally-adjusted estimate of the forecasted demand for the next three week period. This estimated demand is the inventory level that *Medicorp* keeps, or strives to keep, on a continuing basis. Each store within the *Medicorp* chain orders replenishments on a weekly basis and receives the ordered items 2-3 days later. Overall, this model yields the 95% target for customer satisfaction. The present work examines the use of strategic data mining techniques to achieve strategic corporate goals for customer satisfaction while simultaneously cutting down the current inventory.

Our technique relies on predicting customer demands and stocking inventories accordingly to keep up with demand. We started by looking at the transactional data warehouse at *Medicorp*. The *Medicorp* data warehouse is hundreds of gigabytes in size containing all sales information from 1986 to the present. From this vast data warehouse, we extracted a portion of the recent fields that we felt would provide adequate raw data for a preliminary statistical analysis:

- Date field – Indicates the date of the drug transaction
- Customer number – Uniquely identifies a customer (useful in tracking repeat customers)
- NDC number – Uniquely identifies a drug (equivalent to a drug name)
- Quantity number – Identifies the amount of the drug purchased
- Days of Supply – Identifies how long that particular drug purchased will last
- Sex fields – Identifies the sex of the customer
- Cost Unit Price – Establishes the per unit cost to *Medicorp* of the particular drug
- Sold Unit Price – Identifies per unit cost to the customer of the particular drug

The preliminary statistical analysis was utilized to help search for seasonal trends, correlation between field variables and significance of variables. Our preliminary statistical data provided evidence for the following conclusions:

- Women are more careful about consuming medication than men.
- Sales of most drugs showed no or little correlation to seasonal changes.
- Drug sales are heaviest on Thursdays and Fridays.
- Drug sales (in terms of quantity of drug sold) show differing degrees of variability:

Maintenance type drugs (for chronic ailments) show low degrees of sales variability.

Acute type drugs (for temporary ailments) show high degrees of sales variability.

3 Reasons for Choosing Recurrent Neural Networks

In this section, we explain the rationale behind choosing recurrent neural networks as the prediction technique for the problem at hand. After a survey of the time series prediction literature, we found that neural networks performed at least as well as other techniques in a majority of cases. For example, the results of the Santa Fe competition on time series prediction [11] suggest that the performance of neural networks is better than that of other techniques for predicting the future trends in stock prices. Mozer [9] explains the details of the neural network architectures used in that competition. The problem of predicting future sales, as with other time series prediction problems, requires the network to maintain state[1] information representing relevant information about the past sales. This information can be used to detect trends in the data to make future predictions possible. Of the neural network architectures that exploit state information, we decided to choose from either recurrent neural networks or time-delay neural networks because they seemed to be the most well studied, with a large body of work describing how to set the network parameters.

The results in [9] indicate that the predictive performance of recurrent neural networks and time delay networks do not differ greatly. We chose the recurrent neural network architecture because

[1] This is not necessary if the sales data for x days is given as input to the network, where x is large enough to subsume the period of any pattern in the data. However, in real world situations, one is unlikely to know the periods of such trends and hence, this workaround is not always feasible.

the length of the delay in time-delay networks has to be set in advance [7] and because recurrent neural networks are more general than time delay networks. The reason for the latter is that recurrent neural networks could learn to delay the inputs to any arbitrary length in time[2].

4 Implementation of Recurrent Neural Networks

In this section, we first describe the reasons for choosing a particular learning algorithm for recurrent neural networks. A distributed design for recurrent neural networks is also briefly described. Finally, we describe experiments used to validate the implementation of the distributed design.

4.1 Selecting the Learning Algorithm

The three main algorithms that exist for training recurrent neural networks and their advantages and disadvantages are listed below:

1. Back Propagation through Time: This is an extension of the regular back-propagation algorithm used in feed-forward neural networks. However, at each time instance, the back propagation algorithm takes time proportional to the number of time units after the start time. Also, the storage needs increase with time. Thus, this approach is impractical when the length of the sequence is unbounded and is computationally expensive when input sequences are long.
2. Time-Dependent Recurrent Back-Propagation: This algorithm is designed for training a continuous-time recurrent network. This algorithm has very modest space and time requirements. However, this algorithm does not allow for real-time learning.

[2] The length to which the recurrent neural network can delay the input signal increases with the number of units in it.

3. Real-Time Recurrent Learning: This algorithm can be used to train general recurrent neural networks and has space and time requirements which are independent of time (though still more than the requirement for Time-Dependent Recurrent Back-Propagation algorithm). An advantage of this algorithm is that it can be run on-line without waiting for the sequence to be completely seen.

Since we wanted to have a general implementation that could be used for many time series prediction tasks, we chose to use the Real-Time Recurrent Learning [14, 13] (RTRL) algorithm. Though this implementation is more expensive in terms of space and time when compared to the time-dependent recurrent Back-Propagation algorithm, the on-line nature of the RTRL algorithm allows for continuous learning and prediction over arbitrarily long sequences (such as what might be expected in our domain of interest).

4.2 Design of the Neural Network

The design of the neural network was made to facilitate implementation in a distributed environment. Nodes and edges in the neural network were modeled as objects and interactions between these components occurred only through well-defined interfaces. These interactions could be directly translated into messages in a distributed environment. Modifications were also made to the RTRL learning algorithm so that maximum parallelism could be exploited in a distributed implementation. These modifications ensure that the nodes in a network have to communicate only with other nodes connected to them (the output nodes, however, have to communicate with all the nodes in the network).

The implementation of this distributed design was done using C++ because the object-oriented concepts in the language matched the design well. The implementation was made general enough so that arbitrary topologies of networks could be instantiated and activation functions could be specified independently for every node. The learning rate can also be varied and a momentum term can be specified to filter out high frequency disturbances on the error surface that the network traverses during the learning phase.

4.3 Validation of the Implementation

We tested our implementation of the recurrent neural network by running it on simple domains like (a) Exclusive Or with Delay [14] and (b) Shift Register [10]. We present the learning results on the Delayed Exclusive Or problem in here.

4.3.1 The Delayed Exclusive Or Problem

This problem involves giving the network a continuous stream of two inputs. The Exclusive Or of the inputs at time t is the required output at time t + 2. Thus, the recurrent neural network should not only learn that the output is the Exclusive Or of the inputs but should also learn that the output is to be delayed by two time units. No information other than the input and the expected output is given to the network. Table 1 shows an example trace of inputs and outputs presented to the network.

Time	Input 1	Input 2	Output	Comments
1	1	0	0.5	Output undefined, so arbitrarily set to 0.5
2	1	1	0.5	Output undefined, so arbitrarily set to 0.5
3	0	0	1	Exclusive Or of inputs at time 1

4	1	1	0	Exclusive Or of inputs at time 2
...	…	…	…	…
N	0	1	0	Exclusive Or of inputs at time N-2
N+1	1	0	0	Exclusive Or of inputs at time N-1
N+2	0	0	1	Exclusive Or of inputs at time N

Table 1: Traces of Input and Output in Delayed Xor

4.3.2 Experimental Results

A recurrent neural network with 3 nodes was presented with the time delayed Exclusive Or patterns. A learning rate of 0.1 was used and training was done over 100000 test inputs. The momentum term was set to 0.5. The output of the network at some time *t* was said to be correct if it differed from the actual output by no more than 0.25 (thus, the squared error for a ``correct'' output should be less that 0.0625). As figure 1 shows, the network learned the relationship between the inputs and the output. We also tried training a five node recurrent neural network

with a 3-time delay Exclusive Or pattern and this function too was effectively learnt. The results that we obtained are consistent with those presented in [14]. A recurrent neural network with three nodes was also trained to be a shift register with two time delays between input and output. From the positive outcome of these experiments, and by carefully studying the execution traces of the program, we concluded that the implementation of recurrent neural networks was ``correct'' and could be used for real-world problems.

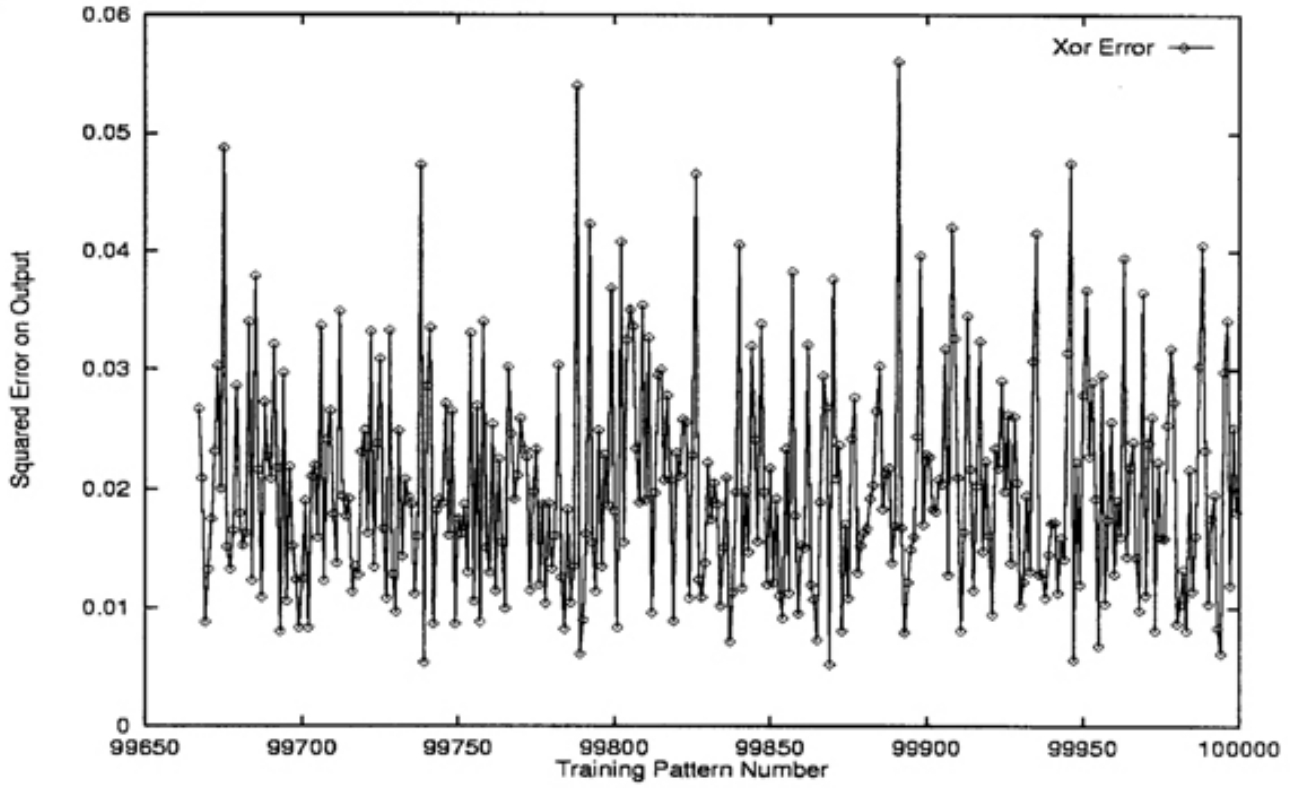

Figure 1: Squared Error on Output for two delay XOR

5 Sales Predictions

In this section, we present the prediction results on real-world data obtained using recurrent neural networks. We first discuss how the recurrent neural network was trained and how over-fitting was avoided. We then discuss the value of parameters chosen for the predictions and the reasons for the choice. Finally, we present an analysis of the prediction results and investigate the appropriateness of recurrent neural networks for this prediction task.

5.1 Architecture and input/output format

Since the requirement was to predict the sales one-day in advance, we chose to treat each day as one unit of time. Thus, we had 365 data points, which corresponded to each day in the year, as the training data set. At each time instance *t*, the sales for the day *t* were given to the network and the sales for day $t + 1$ were obtained as the output of the network. No other input was given because *Medicorp* provided no information about other variables. We used an implicit representation of time [6], i.e., time was not an explicit parameter, which was input to the recurrent neural network but was implicitly represented as the training period of the network.

The input sales were scaled between –1 and +1 using the formula:

$$scaled\_sales = 2 * (actual\_sales / max\_sales) - 1 \quad (1)$$

where *scaled_sales* is the scaled value of *actual_sales*, given that the maximum sales of the product in a day is *max_sales*. The output sales predicted by the neural network, which lies between 0 and 1 for logistic units, were scaled to give the predicted sales using the formula:

$$predicted\_sales = (predicted\text{-}value - x) * max\_sales / (1 - 2 * x) \quad (2)$$

where *predicted-value* is the value predicted by the network corresponding to the sales for the next day (the day after the current time step with respect to the network). The value of *x* served to scale the output to between *x* and $1 - x$ because logistic units have difficulty predicting extreme values (near 0 and 1).

The information about the actual value of the next day sales is provided to the network for updating its weights. The transformation performed to these sales before being given as the

required output to the network is the inverse transformation of the one presented above. Required prediction sales is determined by the formula:

$$required\ prediction = x + [actual\_future\_sales * (1 - 2 * x) / max\_sales] \quad (3)$$

where *actual_future_sales* is the actual value of sales on the day after the current day.

The 365 data points available for the year were used as the training data set. Since repeated training was sometimes required when we used small learning rates, the outputs of each unit was set to 0.5 (a neutral value between the extremes 0 and 1) at the beginning of each epoch.

5.2 Avoiding Over-fitting

One of the problems faced in training neural networks is over-fitting to the data used for training. After the network learns the general trends present, it tries to fit its predictions to the actual data that is presented for training. This leads to prediction results that are skewed to the idiosyncrasies of the training data set. We used the half-year data available for the first half of next year as the test data set to avoid over-fitting. The sum of the squared errors of the predicted sales for the first half of the next year was measured after each training epoch. When this error started rising, it meant that the neural network was starting to over-fit with respect to the training data [12]. This technique proved to be very effective for most of the predictions. This was because the minimum sum of the squared errors on the test data set far exceeded the sum of the squared errors on the test data set when the network ``converged''. Figure 2 shows a typical over-fitting curve.

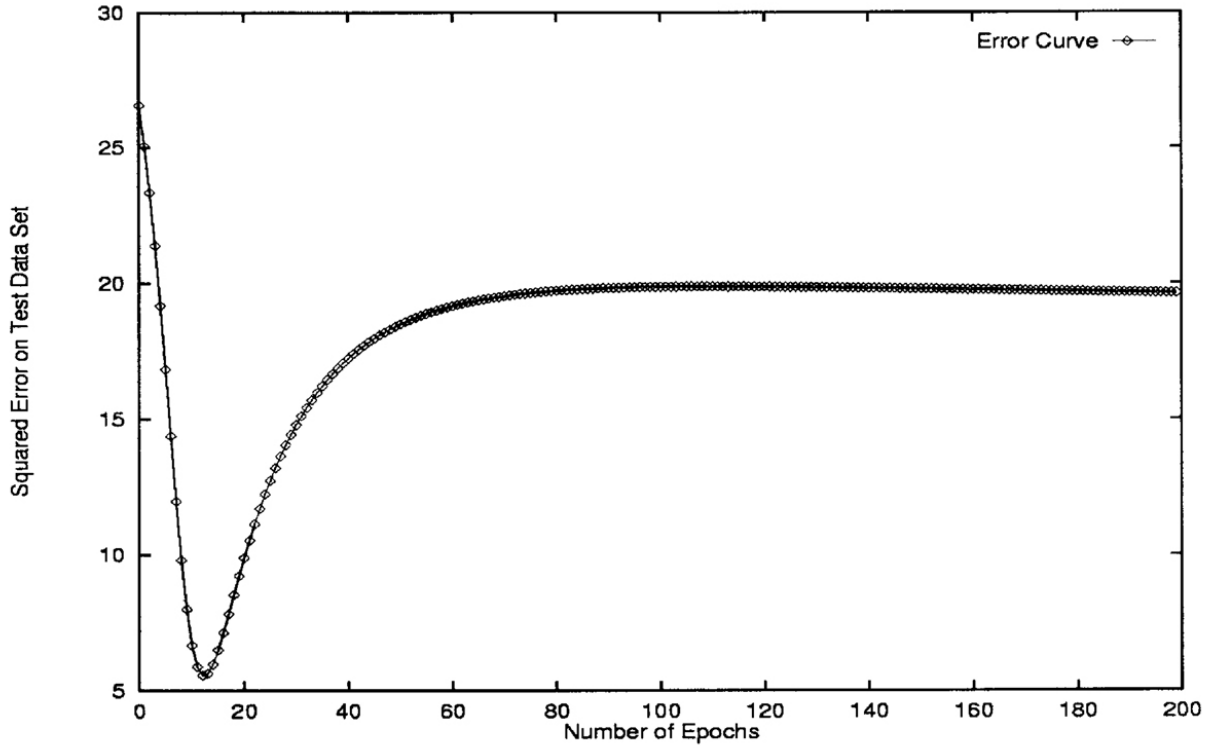

Figure 2: An Example Overfitting Curve

5.3 Determining Appropriate Parameters for the Architecture

There are four parameters, which have to be determined before using a recurrent neural network to predict values. They are

(a) The number of nodes in the recurrent neural network.

(b) The learning rate of the neural network

(c) The momentum rate used to remove high frequency variations in the error surface

(d) The variable x defined in section 5.1.

The momentum term was not experimented with as part of this project and was set to 0 in all the simulations. The details as to how the other parameters were determined are given below:

5.3.1 Learning Rate

The learning rate was chosen to be low enough so that the curve got by plotting the mean square error on the test data set (and the training data set) versus the number of training epochs was a smooth curve. What this meant was that the mean square error on the training data set monotonically decreased over time and the mean square error on the test data set decreased monotonically till the point of over-fitting, after which it increased monotonically. We experimented with various learning rates for the problem at hand and finally arrived at the learning rate of 0.001 for fast moving drugs and 0.01 for slow moving drugs. These learning rates give rise to smooth mean square error curves like the one given in figure 2. A higher learning rate of 0.01 for fast moving drugs gives rise to noisy error curves like the one shown in figure 3. Even higher learning rate of 0.1 does not lead to any learning at all.

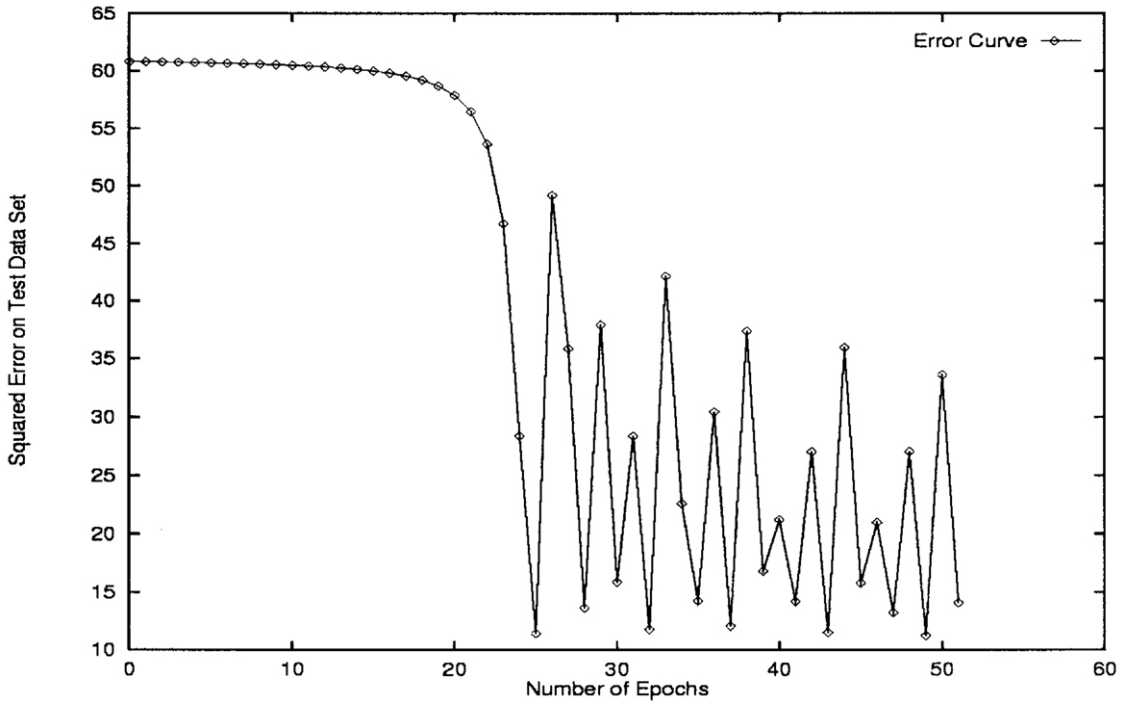

Figure 3: A Noisy Error Curve

5.3.2 Number of Nodes

Initially, we hypothesized that the larger the number of nodes in the neural network, the better the prediction is likely to be. However, on running a few experiments, we found that this was not the case and that there were networks with a certain number of nodes that were better suited for the task than other networks. We feel that the reason for this is *over-generalization* when there are too few nodes and *under-generalization* when there are too many nodes. That is, when there are too few nodes, the network is unable do enough processing and only has enough information to detect broad trends. However, when there are too many nodes, the network acts like a lookup table and stores specific variations in the output. Thus, the network is unable to generalize and pick up trends. In the case of fast moving drugs, surprisingly, a recurrent neural network with just 4 nodes performs better than most architectures with more number of nodes. This is probably because trends are more short-term in fast moving drugs and/or because a lot of generalization is required. Figure 4 shows how the error varies with the number of nodes in the network for a particular fast moving drug. This pattern was roughly followed for the other fast moving drugs too (only in one out of the 4 cases did the error for the four node network exceed the error of a network with a different number of nodes). We also notice that the error starts to decrease as the number of nodes is increased past 10. This decrease is however very gradual and the error of the 4 node network continues to be better than networks having any reasonable number of nodes. In our experiments, we used networks with 4, 10 and 20 nodes for predicting fast moving drugs. For slow moving drugs, the variation in error is roughly similar to the variation in figure 4 with the exception that the error for networks with 10 nodes is almost the same as the error for networks with 4 nodes.

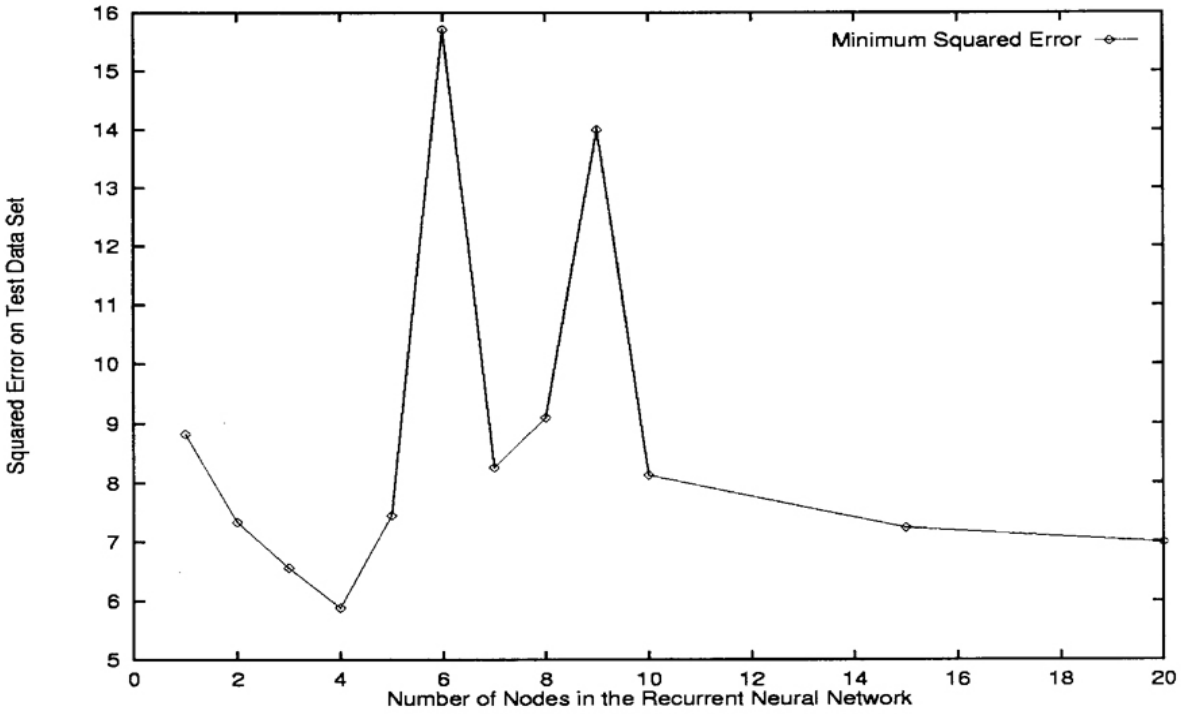

Figure 4: Squared Errors versus Number of Nodes in the Network

5.3.3 Output Scaling

Since each node in the recurrent neural network has a logistic activation function, it is very difficult for the network to output extreme values (i.e., near 0 and 1). In order to overcome this difficulty, we studied the effect of scaling output in the range x to 1-x linearly so that the effective output range was between 0 and 1. Figure 5 gives the minimum sum of the squared error on the test set of a fast moving drug for different values of x. This figure is similar for slow moving drugs. We see that the error increases with the value of x. The reason for this is twofold:

(a) As the value of x increases, even slight differences in the output are magnified when the scale is adjusted to the [0, 1] range.

(b) The networks do not make high amplitude predictions for this application.

Since the error when x = 0 is about the same as the error when x = 0.1 and because slow moving drugs require the network to predict 0 sales, we chose to use the latter setting for our experiments.

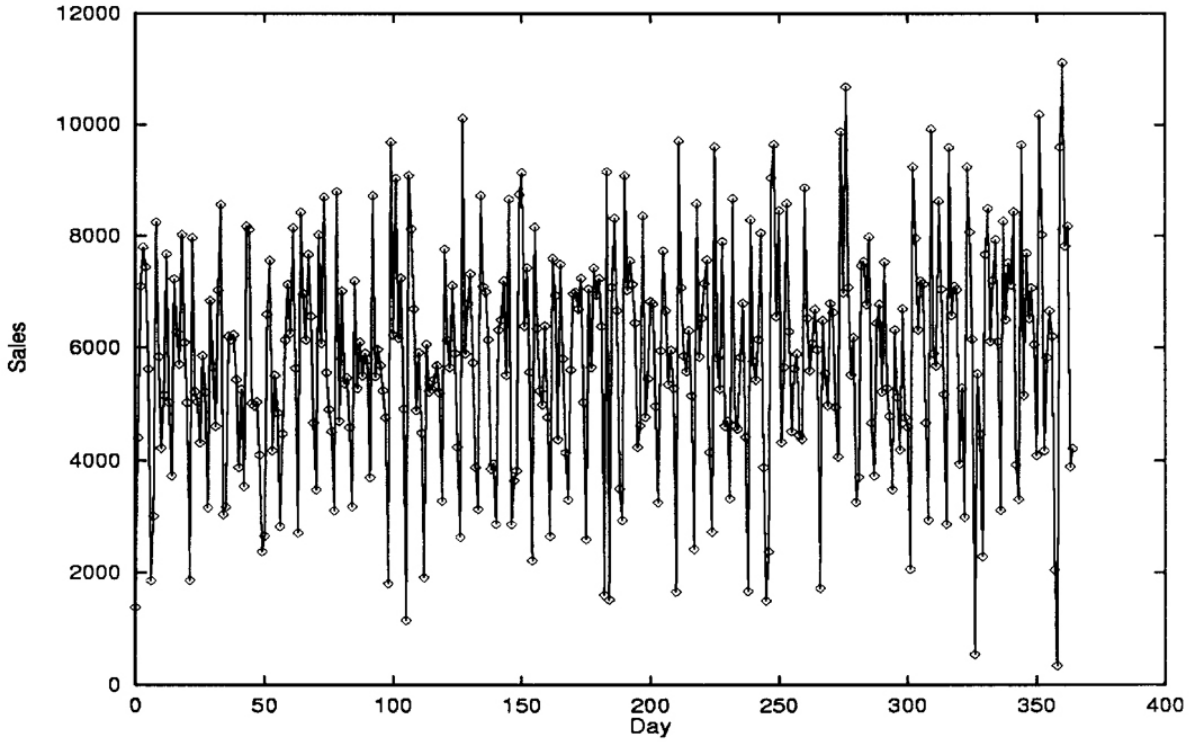

Figure 5: Effect of Output Scaling on Error

5.4 Prediction Results

In this section, we present some graphs that show the prediction results of the recurrent neural network. For the sake of brevity, sales prediction results are shown for only two of *Medicorp*'s products. These two were selected as representative samples from the ten products that we studied in this work. Figure 6 shows the actual sales of drug 1 (a fast moving drug) on each day of the past year and figure 7 shows the actual sales of drug 1 for the first half of the next year. Figure 8 shows the predicted results of a recurrent neural network having 4 nodes. Figures 9 and 10 show the predictions of architectures with 10 and 20 nodes. The predictions of these two architectures do not minimize the squared error on the test set offer comparison on the various architectures. Note

that the scales of the graphs are different because each graph was scaled to show the maximum amount of detail. Thus, Figures 7 and 8 look different although they show the same trends.

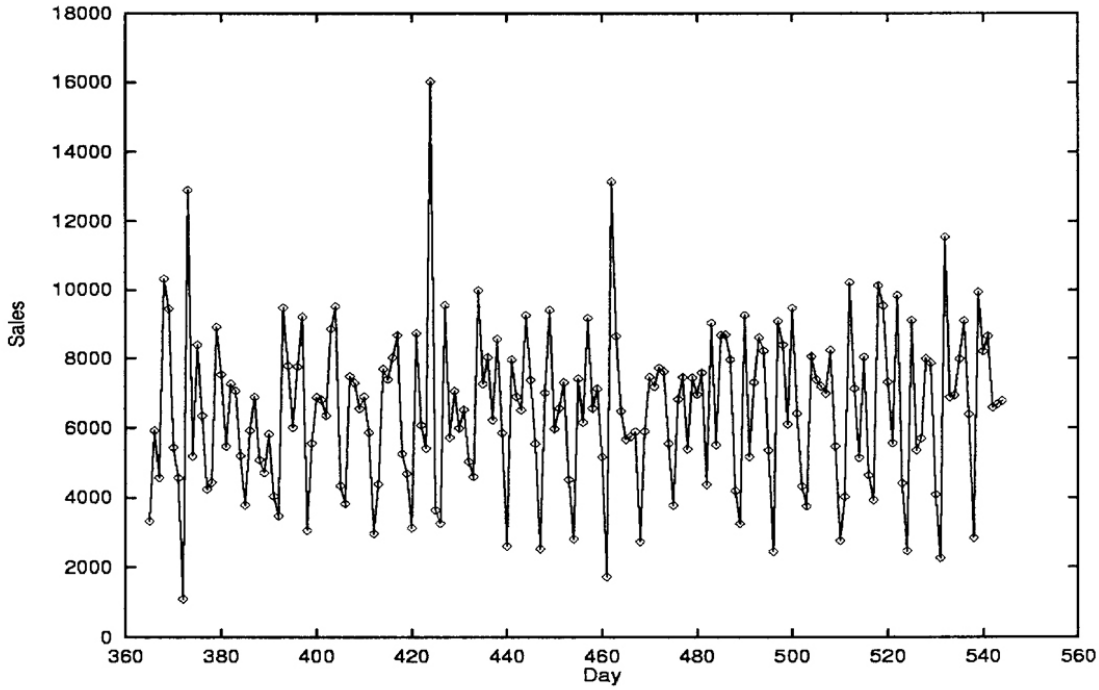

Figure 6: Sales of Drug 1 in over a year

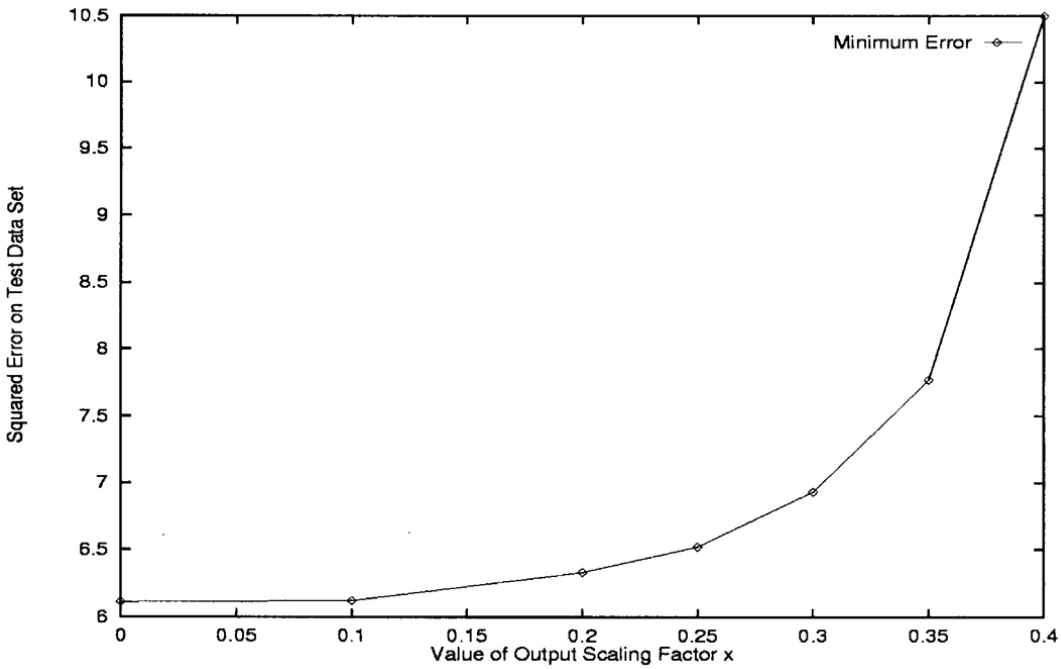

Figure 7: Sales of Drug 1 in first half of next year

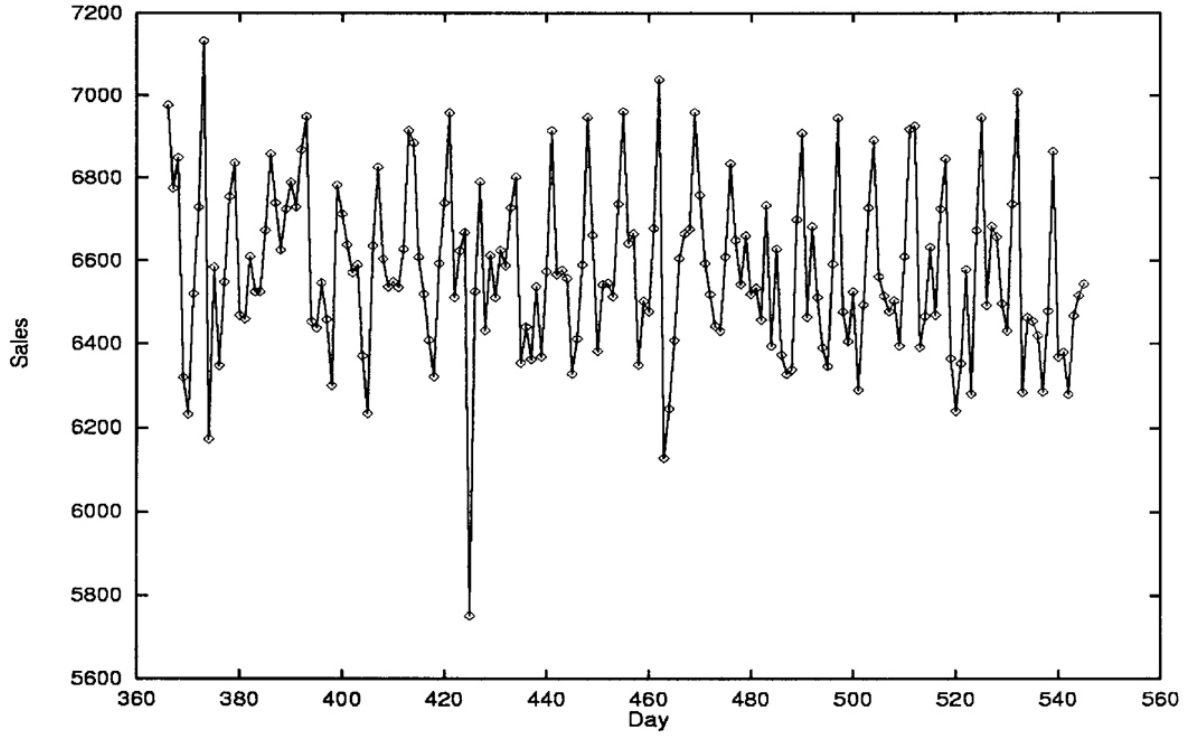

Figure 8: Predicted Sales of Drug 1 in first half of next year (4 node network)

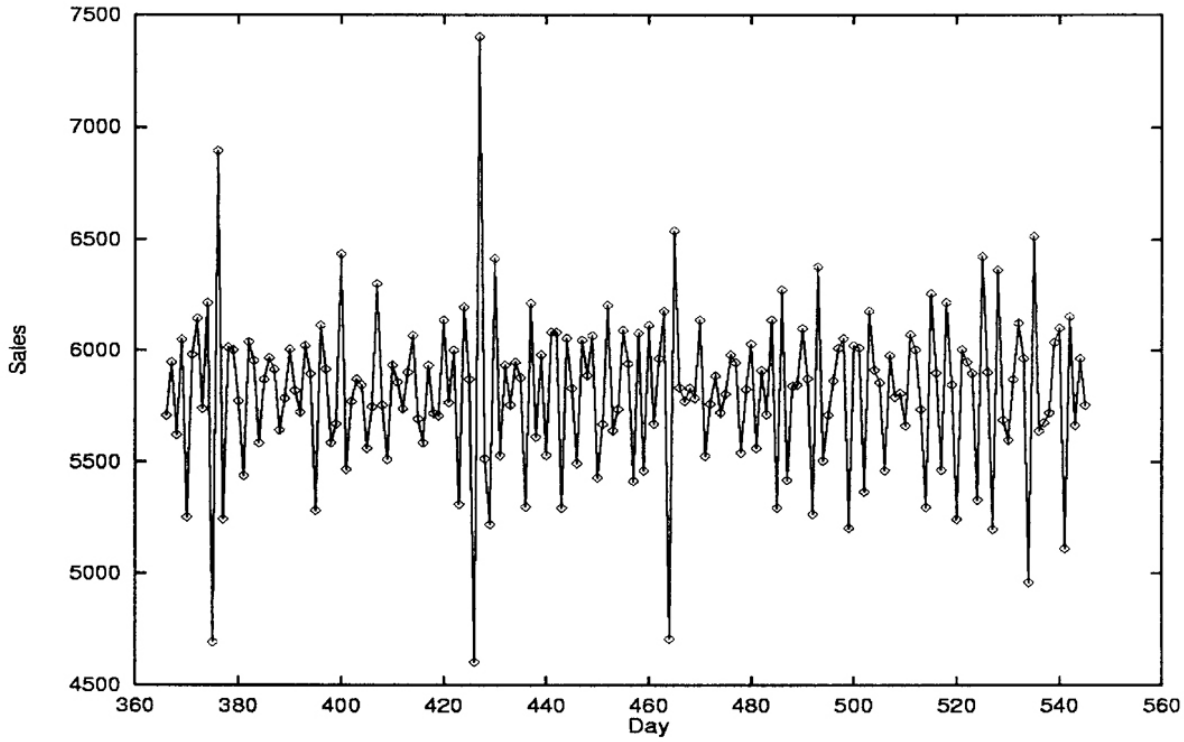

Figure 9: Predicted Sales of Drug 1 in first half of next year (10 node network)

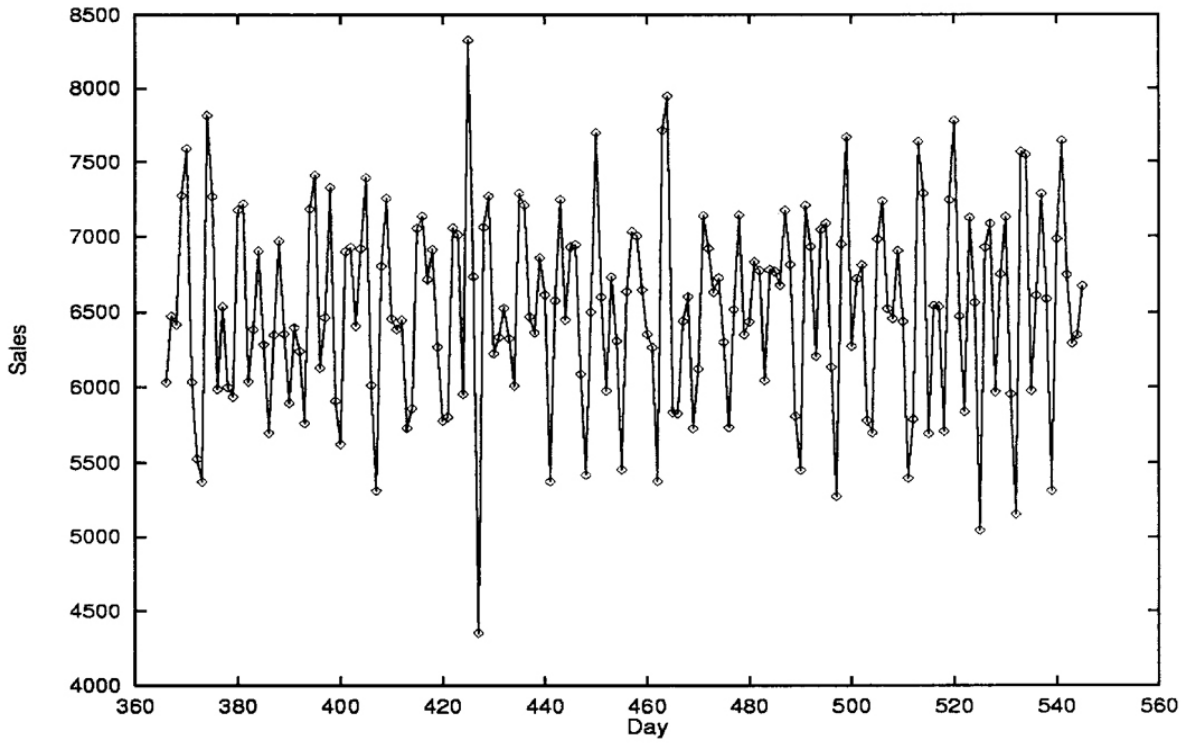

Figure 10: Predicted Sales of Drug 1 in first half of next year (20 node network)

5.5 Analysis of Prediction Results

On interacting with *Medicorp management*, we found out that they were interested primarily in two prediction results:

(a) The trend of the sale for the next day, i.e., whether sales were going to increase, decrease or remain the same and

(b) The actual amount of sales for the next day. An accurate determination of sales would automatically provide the information in (a).

The *Medicorp* management suggested that trend analysis would be very useful even if accurate prediction for the next day were not possible. This is because any indicator of trends is likely to

lead to savings. In this section, we determine how the predictions of the recurrent neural networks measure against these two metrics.

5.5.1 Trend Analysis

In general, the predicted results seem to follow the trends well for fast moving drugs. The predictions in figure 8 follow the actual trends 70% of the time. If the prediction was always up or always down (fast moving drugs rarely have the same sales for two consecutive days), the expected percentage of correct trend predictions is 50%. Thus, we can see that the neural network uses regularities in the data to predict future trends. In general, the prediction of trends was good for all the fast moving drugs we investigated. Table 2 shows the percentage of correct trend predictions for four fast moving drugs.

Fast Moving Drug Number	% of Right Trend Predictions
1	60.55%
2	62.77%
3	70.00%
4	65.00%

Table 2: Trend Predictions for Fast Moving Drugs

The trend prediction in the case of slow moving drugs is also good. For instance, the predictions in drug 4 match the actual trends 52.78% of the time. This is better than always predicting that the sales would remain the same (the most frequent category), which gives only 46.11% accuracy. Thus, we see that the neural networks learn some sort of patterns in the sales of slow moving drugs too. The percentage of correct trend predictions for four slow moving drugs is given in table 3.

Slow Moving Drug Number	% of No Changes	% of Right Trend Predictions
1	52.77%	55.56%
2	51.11%	61.11%
3	50.00%	52.33%
4	46.11%	52.78%

Table 3: Trend Predictions for Slow Moving Drugs

5.5.2 Error in Sale Prediction

Unfortunately, recurrent neural networks do not predict the actual sales figures too well. This can be seen by comparing figures 7 and 8, representing sales and predictions of fast moving drugs. Although the predicted results follow the trends rather well, the amplitude of the predictions is much less than that of the actual results. The results are worse in the case of slow moving drugs. Even the constant prediction does not fall to 0 because of uncertainty about the next peak.

We use the root mean square error (scaled between 0 and 1 using the formula used to scale the actual output sales) of the predicted sales with respect to the test set as the measure of the deviation between the predicted output and the actual output. Tables 4 and 5 tabulate these values for four fast moving and four slow moving drugs respectively.

Fast Moving Drug Number	Root Mean Square Error
1	0.17
2	0.23
3	0.20
4	0.22

Table 4: Prediction Error for Fast Moving Drugs

Slow Moving Drug Number	Root Mean Square Error
1	0.20
2	0.23
3	0.25
4	0.19

Table 5: Prediction Error for Slow Moving Drugs

This root mean square error for fast moving drugs is very high (approximately 20% of the total variation). This coupled with the fact that the output varies within a range of only 0.6 about 90% of the time suggests that the amplitudes predicted are not likely to be too useful. In the case of slow moving drugs, although the output is usually 0, the network cannot predict this because of uncertainty about the height of the next peak.

We feel that the prediction of the amount of future sales is not very accurate mainly due to the uncertainty introduced due to the effects of exogenous variables. This hypothesis is consistent with that presented in [9]. Recurrent neural networks can effectively learn some fundamental mathematical functions and this learning rapidly deteriorates in the presence of noise (which is intended to model the effects of exogenous variables).

The prediction results suggested in the previous sections suggest that, although recurrent neural networks perform well in detecting trends, they do not always predict the correct magnitude of sales. The effect of exogenous variables on *Medicorp*'s sales could be a major cause for this difference in prediction and actual results. Predictions become weaker as the amount of noise (which could be construed as the effects of exogenous variables) becomes more pronounced.

6 Conclusions and Future Work

The rapid growth of business databases has overwhelmed the traditional, interactive approaches to data analysis and created a need for a new generation of tools for intelligent and automated discovery in data. Knowledge discovery in databases presents many interesting challenges within the context of providing computer tools for exploring large data archives. Inventory control is a nascent application for neural network based strategic data mining.

The paper presents preliminary data from research effort currently underway in the Sloan School of Management in strategic data mining [1, 2, 15]. Earlier efforts from this research group concentrated on the use of multi layer perceptron (MLP) and time delay neural networks (TDNN) for inventory control. Prototype based on these networks was successful in reducing the total level of inventory by 50% in *Medicorp*, while maintaining the same level of probability that a particular customer's demand will be satisfied.

In this paper, we designed and implemented a recurrent neural network and used it to predict sales. The predicted results were a good indicator of the trends but did not have strong (high amplitude) predictions. We have put forth one possible explanation for this phenomenon in terms of exogenous variables. More specifically, noise weakens the predictions of recurrent neural networks, which otherwise are able to accurately predict data. One possible reason for the weak prediction of *Medicorp*'s sales could be the pronounced effects of exogenous variables on sales. Since the effects of these variables would appear as noise to the neural network (in the absence of any information about the variables), the predictions are conservative and not too high in

amplitude. However, the trends in sales are predicted accurately, in spite of the influence of these exogenous variables.

During the course of this project, we also investigated alternatives to the recurrent neural network architecture. Barto et. al. [3] use Associative Reward Penalty (ARP) units instead of standard logistic units in a neural network. A modified architecture could use these units in place of logistic units in a neural network and use eligibility traces so that the units remember the past and use it to predict the future. Since the ARP units are stochastic, they have fewer tendencies to get stuck in local minima. Though using the architecture will not avoid the problems due to exogenous variables, this architecture may be better suited for prediction tasks. Including exogenous variables in the predictions and exploring this new architecture are plans for future work.

7 Acknowledgments

We thank Kanti Bansal and Aparna Agrawal for their help in building training data for the neural network. We also thank Gerardo J. Lemus Rodriguez, Auroop Ganguly and Neil Bhandar for their helpful ideas and discussions. Proactive support from the top management of *Medicorp* throughout the endeavor is greatly appreciated.

References

[1] Bansal, K. Vadhavkar, S. and Gupta, A., "Neural Networks based Forecasting Techniques for Inventory Control Applications" in International Journal of Agile Manufacturing, Vol. 2. No. 1, 1998.

[2] Bansal, K. Vadhavkar, S. and Gupta, A., "Neural Networks Based Data Mining Applications For Medical Inventory Problems" in Data Mining and Knowledge Discovery, Vol2. Issue 1, 1998.

[3] Barto, A.G., “Adaptive Neural Networks for Learning Control: Some Computational Experiments,” Proceedings from the IEEE Workshop on Intelligent Control, 1985.

[4] Barto, A.G., Jordan, M.L., “Gradient Following Without Back-Propagation in Layered Networks,” Proceedings of the IEEE First Annual International Conference on Neural Networks, June 1987.

[5] Barto, A.G., Sutton, R.S., Anderson, C.W., ``Neuron like Adaptive Elements That Can Solve Difficult Learning Control Problems," IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-13, No. 5, Oct. 1983.

[6] Elman, J.L., ``Finding Structure in Time," Cognitive Science, vol. 14, pp. 179-211, 1990.

[7] Hertz, J.A., Krogh, A.S., Palmer, R.G., ``Introduction to the Theory of Neural Computation," Santa Fe Institute Studies in the Sciences of Complexity, Lecture Notes vol. I, Addison-Wesley, 1991.

[8] McClelland, J.L., Rumelhart, D.E., ``Explorations in Parallel and Distributed Processing," MIT Press, Cambridge, 1988.

[9] Mozer, M.C., ``Neural Net Architectures for Temporal Sequence Processing," Predicting the future and understanding the past (Eds. A. Weigend and N. Gershenfeld), Addison-Wesley, 1993.

[10] Rumelhart, D.E., McClelland, J.L., ``Parallel Distributed Processing: Explorations in the Microstructure of Cognition," MIT Press, Cambridge, 1986.

[11] Weigend, A.S., Gershenfeld, N.A., ``Time Series Prediction: Forecasting the Future and Understanding the Past," A Proceedings Volume in the Santa Fe Institute Studies in the Sciences of Complexity, Addison-Wesley, 1994.

[12] Weigend, A.S., Huberman, B.A., Rumelhart, D.E., ``Predicting the Future: A Connectionist Approach," International Journal of Neural Systems, vol. 1, pp. 193-209, 1990.

[13] Williams, R.J., Peng, J., ``An Efficient Gradient-Based Algorithm for On Line Training of Recurrent Network Trajectories," Neural Computation, vol. 2, pp. 490-501, 1990.

[14] Williams, R.J., Zipser, D., ``Experimental Analysis of the Real-time Recurrent Learning Algorithm," Connection Science, vol. 1, no. 1, pp. 87-111, 1989.

[15] Dhond, A., Gupta, A. and Vadhavkar, S. “Data Mining Techniques for Optimizing Inventories for Electronic Commerce,” in ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2000) Industrial Track, 2000.