Neural Networks Applied to Traffic Management in Telephone Networks

BARRY E. AMBROSE AND RODNEY M. GOODMAN

In this paper, the application of neural networks to some of the network management tasks carried out in a regional Bell telephone company is described. Network managers monitor the telephone network for abnormal conditions and have the ability to place controls in the network to improve traffic flow. Conclusions are drawn regarding the utility and effectiveness of the neural networks in automating the network management tasks.

I. Introduction

Starting in 1990, Pacific Bell and CalTech began working on a real-time traffic management/expert system. This project was called Network Operations Analyzer and Assistant (NOAA). The task of NOAA was to take information from the Pacific Bell network management computer, use it to isolate and diagnose exceptional events in the network, and then recommend the same corrective advice as network management staff would in the same circumstances. To date, NOAA has been deployed in three major Regional Bell telephone companies.

The development of the expert system has been reported in previous papers [1], [2]. Applications of learning techniques to two datasets to facilitate the tasks of network management associated with NOAA are described in this paper. The studies reported on are: 1) time series analysis of occupancy and 2) learning trunk reservation limits.

II. Network Management

Network managers of telephone networks monitor the network for unusual events and can put controls in the network to reroute traffic or cut down on traffic entering the network when conditions warrant. Typically there are two network management centers for an entire company and each can take over from the other if a major disaster strikes.

Typical events that are of interest to the network manager are given in Table 1. In each of these cases, the network manager must diagnose what the problem is, based on observable symptoms and place controls in the network to reduce the impact of the network event.

A telephone network can be modeled as a set of nodes and links (see Fig. 1). Telephone traffic is carried on trunk groups. A trunk group is a set of trunks, each of which can support a conversation. Trunk groups are used to link the offices. If all the trunks are busy on a trunk group, then no new calls can be accepted on that trunk group. Typically a switch makes a number of attempts to connect the call, but if all routes available to it are busy, then the call is blocked.

The nodes are the telephone offices which act as switches for the telephone traffic. These offices route the traffic onto the appropriate trunk groups depending on the number dialed by the customer.

Network management staff may reroute traffic elsewhere (*expansive controls*) or cut the traffic off at its source (*restrictive controls*).

Expansive controls are appropriate for a single overload situation where due to statistical fluctuations in the offered traffic, a trunk group does not have enough capacity to handle its offered load. A typical expansive control might be an *overflow reroute* which reroutes calls to another trunk group after they have failed on the regular trunk group. Other controls may have to be put in at the same time as the main control to avoid routing loops.

Restrictive controls are appropriate for call-in conditions, where most of the traffic has a low probability of completion, but its presence is interfering with the normal network operations. A call-in condition exists when many people call the same telephone number at the same time; i.e., when concert tickets go on sale or a radio station runs a promotion. This traffic is characterized by a large number of call attempts per trunk and low holding time.

Data is available to the network management operators in the form of a display of the state of all trunk groups in the

Manuscript received October 1, 1995; revised February 1, 1996. This work was supported in part by Pacific Bell, the Center for Neuromorphic Systems Engineering (NSF Engineering Research Center Program) Grant EEC-9402726, the California Trade and Commerce Agency, Office of Strategic Technology Grant C94-0165, ARPA/ONR Grant N00014-93-1-0990, and in part by an AFOSR University Research Initiative Grant F49620-93-1-0332.

B. E. Ambrose was with the Electrical Engineering Department, California Institute of Technology; he is currently with AGL Systems, Pasadena, CA 91106 USA (e-mail: bambrose@agl.com).

R. M. Goodman is with the Electrical Engineering Department, California Institute of Technology, Pasadena, CA 91106 USA (e-mail: rogo@caltech.edu).

Publisher Item Identifier S 0018-9219(96)07171-X.

Table 1 Abnormal Events of Interest to Network Managers

Random overloads of trunk groups
Focused overloads caused by phone-ins
Unusual calling patterns, such as on Mother's Day
Earthquakes
Loss of a switching office
Loss of a transmission link

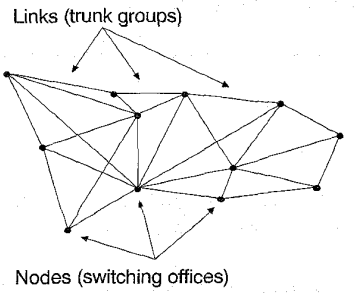


Fig. 1. Telephone network.

network. Overload conditions on trunk groups and alarm conditions in the telephone exchanges are highlighted.

This information is polled from the offices in network every 5 min for trunk group data, and every 30 s for switch alarm data and stored in a database. The neural network systems can then query this information using SQL database interrogation commands. SQL is a database query language.

Clearly it is important to the network manager to have an accurate estimate of where spare capacity exists in the network for the purpose of rerouting traffic. To aid the network manager, a system was developed to predict future values of trunk group capacity based on present data values, using neural networks. This allows the operators to implement reroute controls that need less adjustments over the course of time.

III. Time Series Analysis of Occupancy

The NOAA system acts as a network manager monitoring traffic in the telephone network. Measurements come in to NOAA every 5 min. One of these measurements is trunk group occupancy, which is used to tell how much spare capacity is on a link in the network. By predicting the next measurement of trunk group occupancy, NOAA can make better recommendations about where to reroute traffic.

A. Occupancy

Occupancy is a moving average of 20 samples of the number of trunks occupied on a route. The samples are taken every 30 s, and the result is scaled to be between zero and 100%. Each trunk in a trunk group can carry a single conversation.

Neural networks, linear predictors, and local approximation techniques were used in this study for the time series prediction. See the Appendix for details of the algorithms used to train the neural networks, linear predictors, and local approximators.

The inputs to the neural network were the previous six values of occupancy over a 30 min period. The desired output was the next value of occupancy. A plot of the autocorrelation was used to decide the number of inputs.

A data set consisting of 18 000 observations of occupancy on a trunk group with 360 trunks between Los Angeles and Gardena was gathered from NOAA while it was running. This provided about ten weeks worth of data. The first 300 points of the data set are shown in Fig. 2. Here are some key features to note.

- The traffic level varies according to time of day. The plot shows the traffic level dropping on Friday evening, rising again on Saturday morning, and remaining level during Saturday afternoon.
- Spikes may be present in the time series, e.g., close to example 60.
- The variance of the occupancy varies with the traffic level. The plot becomes more jagged as the trunk group occupancy increases.

B. Cross Validation

The first step in testing a method is the division of the supplied data set into a test set and a training set. The training set is used to derive the coefficients of the model and the model is then tested on the test set. In testing two or more models, the one that does best on the test set is the better one.

In testing the relative merits of prediction techniques, a distinction must be made between learning ability and generalization ability. A good prediction method will generalize well on examples that have not been seen before, by learning the underlying function without learning the associated noise.

This can be tested using cross validation. With v-fold cross validation and a data set of size N, v tests are carried out. Each test employs $N - N/v$ samples as the training set and the remaining N/v samples as the test set. This makes maximum use of the data set and allows us to check the significance of the results.

Strict cross validation involves using a test set of one example and training on the remaining examples. This is done repeatedly leaving out one example at a time. In this way, maximum use is made of the data set in determining the generalization performance of the models. Strict leave-one-out cross validation would have been better but results in a lot more computation time. A single run for the neural network took about a week on a Sun Sparc 10, and strict leave-one-out cross validation using the same software would have taken almost a year. The back-propagation algorithm [3] is the culprit for the long run times, in addition to the need to duplicate the runs to assess the affect of the random weights used for initialization.

For this study, 50-fold cross validation was carried out. An advantage of using cross validation is that it provides some measure of the significance of the result. When comparing two models, one model may do better than another because 1) it more closely represents the underlying

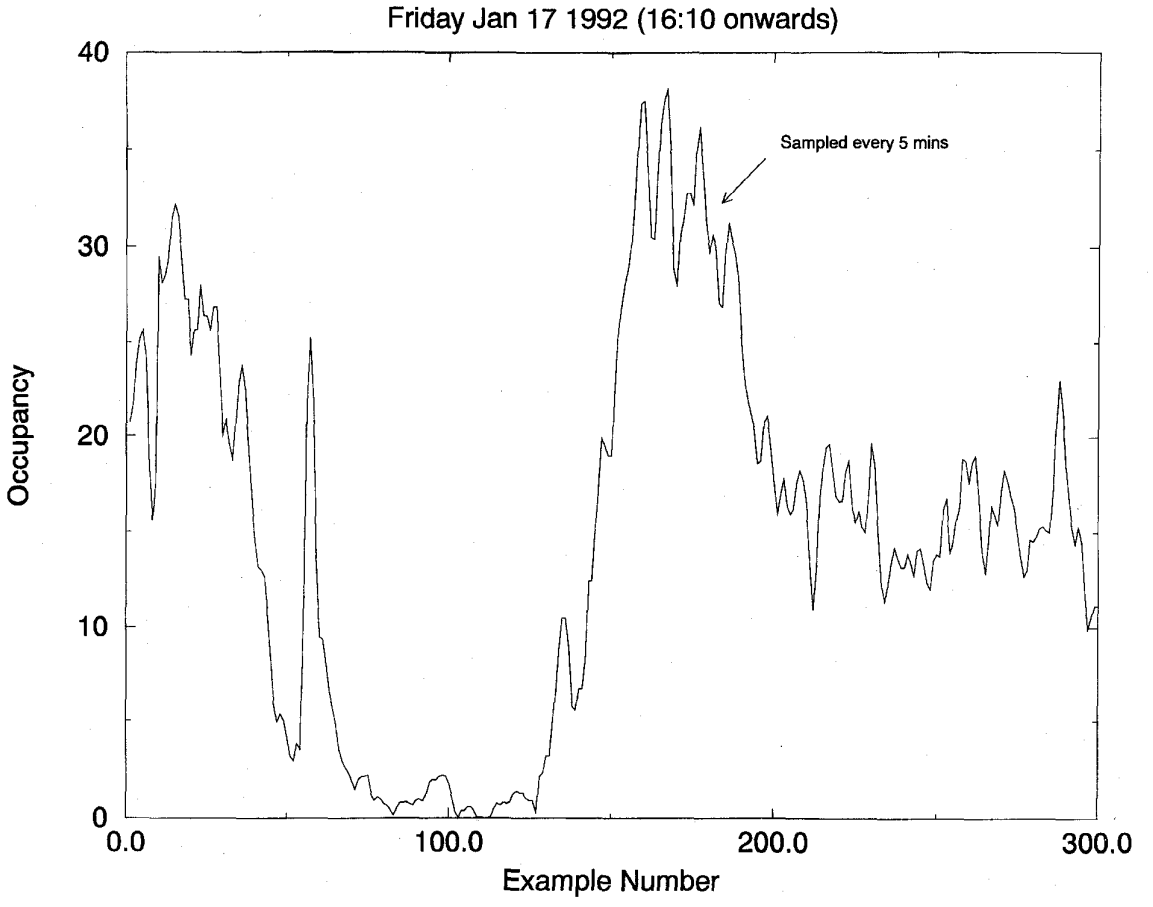


Fig. 2. Occupancy dataset.

Table 2 Neural Network Test Set Prediction Error for Varying Numbers of Hidden Units

Number of Hidden Units	Error
12	0.019 153 1
10	0.019 128 1
6	0.019 100 8
4	0.018 992 4
3	0.019 411 0

function or 2) by chance it more closely represents the test set noise terms. By repeating the experiment using a second test set, an indication of the significance of the second factor is obtained. The more repetitions of the generalization test are carried out, the more significant the results for generalization capability.

C. Neural Network

In these studies, a feedforward neural network with a single hidden layer was used. Quickprop [4] was used for training as it had faster convergence than standard backprop and was freely available on the Internet. For the neural network, trial and error shows that four hidden units and a linear output unit gives best results (see Table 2). The linear output unit is used to aid function fitting. This architecture is fixed prior to training (see Table 3).

Table 3 Scaled RMS Prediction Error for Data Set

Method	Error
Using last sample	13 487
Using linear predictor	13 130
Using neural network	13 054
Using local approximation	13 011

All methods predict the next observation based on the previous six observations. The linear predictor can be taken as the baseline performance to beat. Error is RMS prediction error multiplied by 10 000, with a difference of 10 being significant. In each case, 50-fold cross validation is used.

The neural network had six inputs, four hidden units, and one output. It was compared against a linear predictor which had six inputs, no hidden units, and one output. The neural network gave only about 0.6% improvement over the linear predictor and took much longer to train.

D. Hidden Unit Activations

A plot of hidden unit activations gives valuable insight into the features of the data set. There are four hidden units. One of the hidden units reacts strongly to the overall traffic level. One of the units reacts strongly to rate of change of traffic level while the other two reacts strongly to the rate

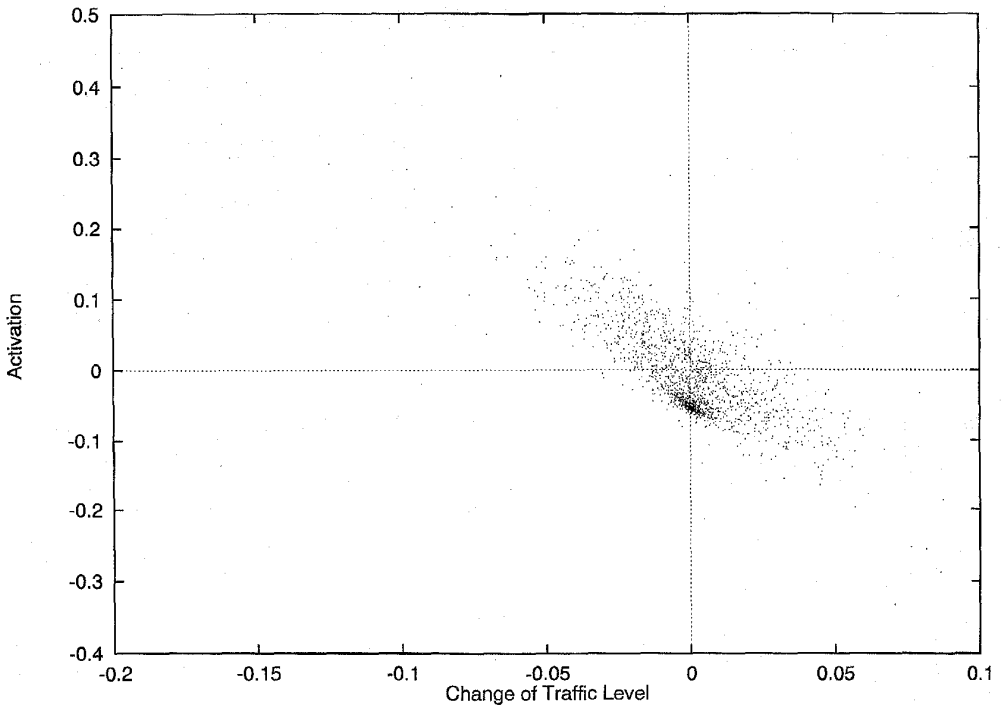


Fig. 3. Activation of hidden unit number three.

of rate of change of traffic level. Examples of two of the plots are given in Figs. 3 and 4. It is possible that the neural network is constructing a Taylor Series of the function of be approximated.

E. Comparison with Recurrent Neural Network

For comparison, a recurrent neural network was trained using the methods of [5]. This recurrent neural network was given just one input and one output instead of the six inputs used by the other methods. This was because the recurrent neural network should be able to retain state information internally and not need the previous history as inputs. The best results that were achieved were not as good as those from the linear predictor. Many combinations of learning rates and number of hidden units were tried. These results suggest that the function space of this particular recurrent neural network is not rich enough to model the expected value function that needs to be approximated by the time series predictor.

IV. Learning Trunk Reservation Limits

In a typical telephone network, calls are usually set up using either one or two hops to get from source to destination. One hop traffic is called *direct routed* traffic. Two hop traffic is called *alternate routed* traffic.

In times of network congestion, network managers of telephone networks have the capability to impose trunk reservation on a trunk group. This means that when a small number of trunks are available on the trunk group, direct routed traffic is allowed to use these trunks but alternate routed traffic is blocked. The theory behind this is that direct routed traffic makes very efficient use of the network.

A. Trunk Reservation

Trunk reservation is a policy whereby in each trunk group, alternate routed traffic is blocked if there are fewer than R trunks free on the trunk group. R is known as the trunk reservation parameter. Recall from Section III that each trunk on a trunk group can carry a single conversation.

Akinpelu [6] shows that trunk reservation prevents networks having two stable states at high loads. Hunt and Laws show that a policy that chooses the least busy alternative for routing and implements trunk reservation is an asymptotically optimal policy in minimizing blocked traffic, as the number of network nodes increases [7].

Closed form solutions for the correct value of the trunk reservation parameter R to use as a function of network load, traffic mix on the trunk group, and number of trunks on the trunk group are not known, except in the asymptotic case for symmetric networks with many nodes and the same traffic offered to each node [8], [9]. In this section, a neural network is used to choose the value of the trunk reservation parameter as a function of input variables that will be described later.

B. Trunk Reservation as a Bet

Consider a trunk group of size N with a trunk reservation parameter R of one. When there are less than $N-1$ trunks occupied or exactly N trunks occupied, the trunk reservation parameter does not influence network behavior.

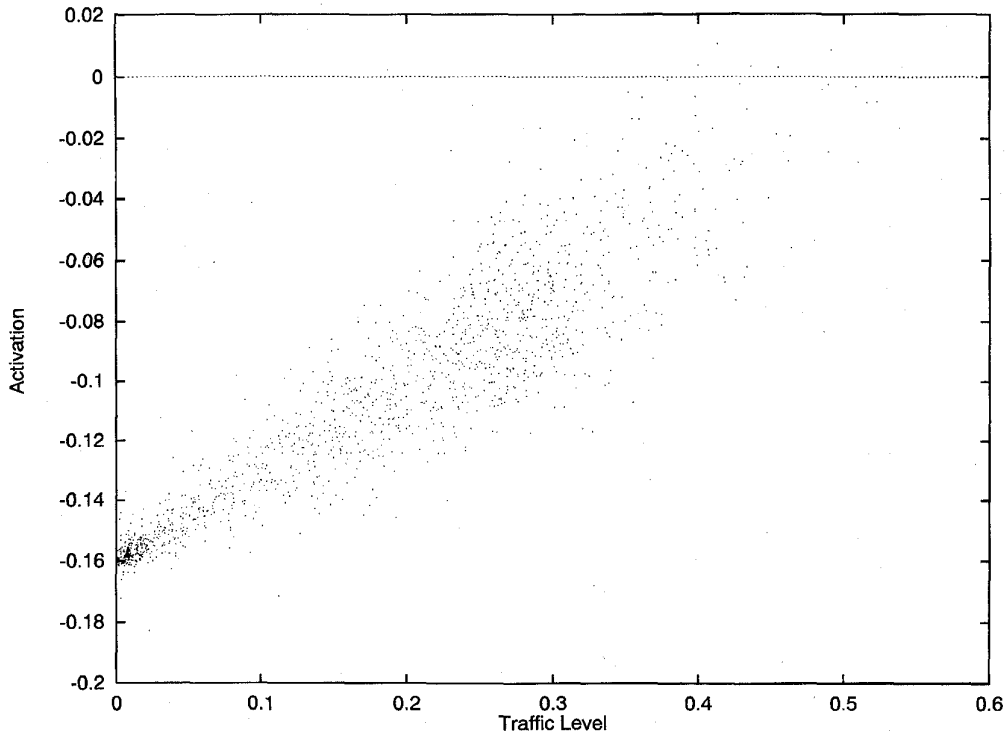


Fig. 4. Activation of hidden unit number four.

When there are exactly $N-1$ trunks occupied and an alternate routed call arrives, trunk reservation causes this alternate routed call to be blocked. The expectation is that a direct routed call will arrive shortly that can make better use of the free trunk.

From this viewpoint, trunk reservation can be looked upon as a bet. When an alternate routed call is rejected, the network loses revenue, but the bet is that a direct routed call will arrive within a short length of time that will produce more revenue, because it makes more efficient use of network resources. The longer it takes for the direct routed call to arrive, the bigger the revenue loss from rejecting the alternate routed call (see Fig. 5).

If it is assumed that the alternate routed call has a holding time (average length) of T_h time units, and the direct routed call has twice the revenue generating potential of an alternate routed call, then the point when the bet has been lost is about $2/3T_h$ after the alternate routed call has been rejected.

C. Simulation

For the traffic simulation, a mixture of light (4.4 Erlangs), medium (6.7 Erlangs), and heavy (8.9 Erlangs) traffic was used. Memoryless arrivals and holding times were assumed. An average holding time of 180 s was assumed. See [10], [11] for the reasons why these assumptions are standard. The simulated network had ten nodes and ten unidirectional trunks between each node. Because light traffic occurs far more often in real life than medium or heavy, a weight of

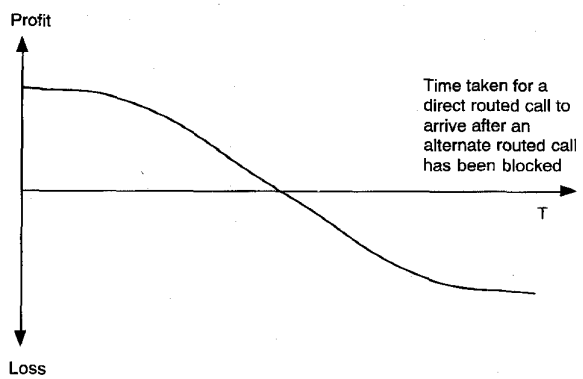


Fig. 5. Profit following trunk reservation decision.

0.8 was given to the light, 0.15 to the medium, and 0.05 to the heavy traffic for the purpose of computing blocking.

The algorithm generated test cases and training cases as follows. For each alternate routed call that encounters trunk reservation, a test was carried out. The neural network inputs were noted at the time the call arrived. Then a count was carried out of the number of direct routed calls that arrived subsequent to the alternate routed call and got blocked. The counting period was two thirds of a holding time. The number of blocked direct routed calls was written to a training file as the desired value of trunk reservation on the route.

The study simulated 500 h of traffic data for each training run. This was used to train the neural network. There is no

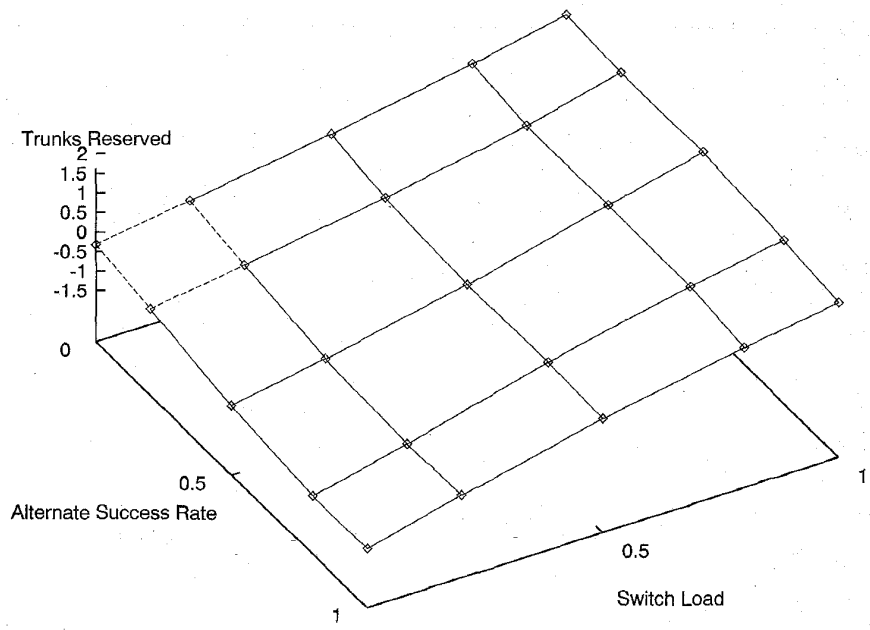


Fig. 6. Neural network output.

reason why the neural network cannot be trained online in a real network.

D. Inputs

Neural networks and linear predictors were used in this study. See the Appendix for details of the algorithms used to train the neural networks and linear predictors.

The inputs were the switch loading and the alternate success rate. The *switch loading* was defined to be the number of occupied trunks attached to the switch divided by the total trunks attached to that switch. The *alternate success rate* was defined to be an exponential moving average of the rate at which calls are successfully carried after failing to get a trunk on the direct route. The rationale is that if there is spare capacity on the alternate routes, then there is less of a need to provide trunk reservation. The results show it impacts the level of suggested trunk reservation in the expected way (See Fig. 6).

The interpretation of fractional values of the neural network output was as follows. If the output of the network was 0.1, this indicated that the "bet" would be lost nine times out of ten. Given this interpretation, a rounding approach was taken, i.e., 0.1 was rounded to a trunk reservation of zero. Similarly a neural network output of 0.9 was rounded to a trunk reservation value of one. In other words, rounding to the nearest integral value was used or the bet was taken if the chance of winning was over 50%.

Quickprop [4] was then run for 200 iterations to tune the neural network weights. About ten such simulation cycles were necessary to get good results.

The neural network had two inputs and two hidden units. A linear output unit is used to aid in function fitting. Since good results were obtained with two hidden units, the number of hidden units was not varied.

Table 4 Blocking Probabilities for Neural Network and Fixed Reservation Parameters on Test Traffic

	Blocking Probability	Std. Dev.
Linear Predictor	0.012 775 6	0.000 110 6
Neural Network	0.012 751 7	0.000 037 4
Fixed Res of 0	0.014 862 2	0.000 047 7
Fixed Res of 1	0.012 927 6	0.000 033 6
Fixed Res of 3	0.015 770 3	0.000 041 7

For comparison, Table 4 also shows the performance of the neural network compared to a linear predictor with the same inputs and output. The neural network performs slightly better. Over ten runs with different random seeds for the traffic simulator, the difference between the two methods is found to be nonzero at the significance level of 98%. It can be concluded that the neural network would be slightly better than the linear predictor in learning this problem.

The neural network had two inputs, two hidden units, and one output. It was compared against a linear predictor which had two inputs, no hidden units, and one output. The neural network gave only about 0.2% improvement over the linear predictor and took much longer to train.

V. Conclusions

In Section III the application of different learning techniques to time series prediction of telephone traffic occupancy is considered. Local approximation and neural networks are two techniques that generalize well. Using a data set with about 18 000 elements, it is possible to be confident in the comparative performance of the different prediction techniques.

It should be noted that in network management, function approximation is important. There are many functions that

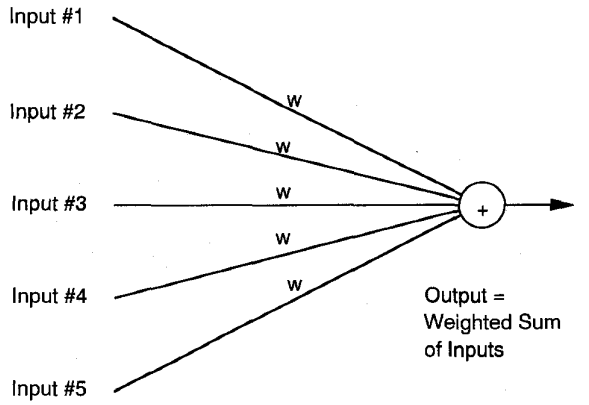


Fig. 7. Linear predictor.

would be difficult to solve for analytically, that can be obtained by fitting a function to observed network data. For example, the optimum trunk reservation parameters are considered in Section IV.

The linear predictor and the neural network are good function approximators and difficult to beat. The linear predictor has advantages over the neural network in that it is quick to train and there is little or no danger of overfitting. On the other hand, the neural network can provide a better fit for functions that are a nonlinear function of their inputs.

Cross validation is the key technique for testing learning ability. The cross validation technique allows maximum use of the dataset, by dividing it into many small test sets. In this way, the significance of the difference in performance between the various prediction methods can be judged. For this reason, cross validation is the perfect method for testing generalization capability.

APPENDIX

The learning techniques used in the following sections include linear predictors, neural networks, and local approximation. A brief description of each technique is given here.

A. Linear Predictors

A linear predictor is a simple type of neural network whose output is a linear combination of its inputs (see Fig. 7).

Suppose there are N inputs x_i to the linear predictor, which has weights w_i and an output y. Then the output y for an input vector X is given by

$$y(X) = \sum_{i=1}^{N} w_i x_i. \tag{1}$$

Suppose now that there are P input vectors X_p, each of which is of the form $(x_1^p, \vdots, x_N^p)$, and a desired output $\hat{y}_p$ for each of these input vectors. Then a sum of squared error (SSE) function E can be defined as follows:

$$E = \sum_{p=1}^{P} (\hat{y}_p - y(X_p))^2. \tag{2}$$

A possible learning rule would be to carry out an adjustment of each weight to minimize the error E over all the training patterns:

$$w_i(\text{new}) = w_i - \eta \frac{\delta E}{\delta w_i} \tag{3}$$

where η is a small constant, termed the learning rate. If the learning rate is too small, convergence is slow. If the learning rate is too big, there may be no convergence. Trial and error is used to find the appropriate learning rate.

Straightforward differentiation gives

$$\frac{\delta E}{\delta w_i} = -2 \sum_{p=1}^{P} x_i^p (\hat{y}_p - y(X_p)). \tag{4}$$

From (3) and (4), provided η is sufficiently small, the optimum weights to minimize the error, E, with respect to the weights can be found. As a practical matter, an extra input is always added to the linear predictor which is set to a constant, either one or -1. The constant input gets multiplied by a weight to give a constant term in (1). This allows the linear predictor to estimate a wider range of functions. Equation (1) now becomes

$$y(X) = \sum_{i=1}^{N} w_i x_i + C. \tag{5}$$

B. Neural Network Architectures

Many neural network architectures can be found in the neural network literature, for example, feedforward, recurrent, and Hopfield neural networks. Feedforward neural networks will be discussed shortly. Recurrent neural networks are similar to feedforward networks, except that they contain feedback connections. Hopfield networks are usually used for optimization problems. In this paper, only feedforward neural networks are considered. See [3] for more on architectures and associated training techniques.

C. Feedforward Neural Networks

The feedforward neural network is one of the most widely used neural networks. An example of a simple feedforward neural network is shown in Fig. 8. In this case, the output y as a function of the inputs x_1^p and x_2^p for input pattern p is given by

$$y = \sum_{i=1}^{3} W_i \tanh \left(\sum_{j=1}^{2} w_{ij} x_j^p - \theta_i \right) - \Theta. \tag{6}$$

Here θ_i, the threshold term, can be treated as a weight applied to an extra input set to a constant -1. In this case, there are three so-called hidden units, each with output:

$$h_i = \tanh \left(\sum_{j=1}^{2} w_{ij} x_j^p - \theta_i \right). \tag{7}$$

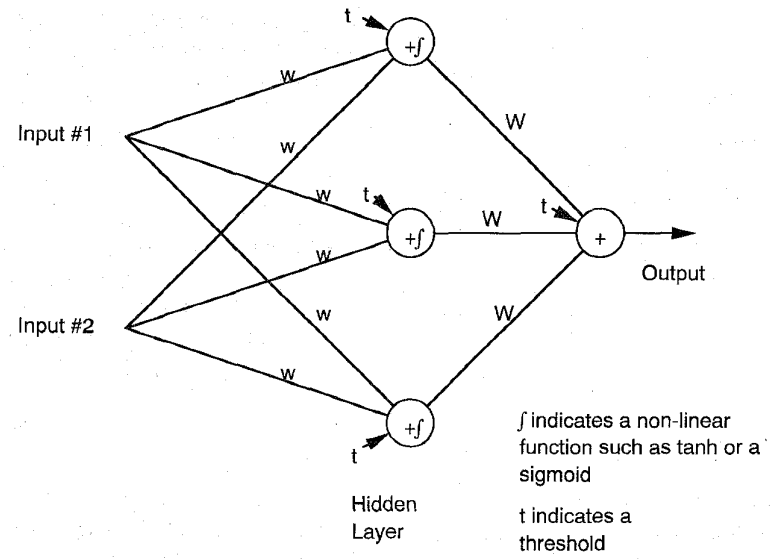


Fig. 8. Feedforward neural network.

If it is assumed that there are P input vectors X_p, each with a desired output $\hat{y}_p$, an SSE term E can be defined:

$$E = \sum_{p=1}^{P} (\hat{y}_p - y(X_p))^2. \tag{8}$$

The first layer weights are updated according to

$$w_{ij}(\text{new}) = w_{ij} - \eta \frac{\delta E}{\delta w_{ij}}. \tag{9}$$

Similarly, second layer weights are updated according to

$$W_i(\text{new}) = W_i - \eta \frac{\delta E}{\delta W_i} \tag{10}$$

using the backpropagation algorithm [3].

D. Backpropagation

The backpropagation algorithm can be derived by differentiating (8) with respect to the appropriate weight. It derives its name from that fact that errors are propagated from the network outputs toward the network inputs.

Define e_p as the error when input pattern p is presented. Then

$$e_p = \hat{y}_p - y(X_p) \tag{11}$$

and the total error E is given by

$$E = \sum_{p=1}^{P} e_p^2. \tag{12}$$

The desired derivative for any weight w is given by

$$\frac{\delta E}{\delta w} = \sum_{p=1}^{P} \frac{\delta e_p}{\delta w}. \tag{13}$$

The derivative of e_p with respect to W_i is given by

$$\frac{\delta e_p}{\delta W_i} = -2 e_p h_i \tag{14}$$

where h_i is the output of the hidden unit, given by (7). The derivative of e_p with respect to w_{ij} is given by

$$\frac{\delta e_p}{\delta w_{ij}} = -2 e_p (1 - h_i^2) x_j^p \tag{15}$$

using the chain rule for differentiation and the fact that

$$\frac{\delta \tanh(x)}{\delta x} = (1 - \tanh(x)^2). \tag{16}$$

By substituting (14) and (15) in (13) and then using (9) and (10), the desired weight update can be calculated.

E. Local Approximation

Another name for the Local Approximation technique might be "history will repeat." The idea is straightforward: a training set and a test set are given. To carry out a prediction for any particular test set example, find examples in the training set which most closely resemble it. Then train a linear predictor on that subset of the training set and use it for prediction. This procedure is repeated for each example in the test set.

The algorithm is more precisely specified as follows. Consider the case of a time series where the task is to estimate the next term in the time series, given the previous six. Assume two sets of data are available: 1) S_0, a training set consisting of n_0 examples, and 2) S_1, a test set consisting of n_1 examples.

Each training set example is of the form $\{y_1', y_2', y_3', y_4', y_5', y_6', y_7'\}$. Each test set example is of the form $\{y_1, y_2, y_3, y_4, y_5, y_6, y_7\}$ where y_7 is to be predicted from $y_1 \cdots y_6$.

A forecast is carried out for each of the n_1 examples based on a model derived from the training data S_0. Instead of using a single model for all the test examples, as is usually the case, a different model is derived for each test example. The model is a linear predictor which is derived as follows. Choose a neighborhood $N_a < n_0$. Pick from the

training set the N_a examples which minimize the Euclidean distance d, where

$$d = \sqrt{\sum_{i=1}^{6} (y_i - y_i')^2}. \tag{17}$$

Here y' indicates a number from the training set and y indicates the current example from the test set. Finding these examples can be done in one pass through the training set and a linear predictor trained using only those examples, as detailed earlier.

Excellent results are reported for this method on chaotic series using a small neighborhood and noise free measurements to a high precision [12].

The disadvantage of this method is that the most accurate predictions require keeping online a large number of training examples. In this case, if traffic spikes occur infrequently, a large volume of data is needed for each trunk route to be sure that the spike is captured. In contrast, the neural network model is more attractive, requiring a much smaller amount of information to be stored (i.e., the weights) to characterize the function to be estimated.

REFERENCES

[1] R. M. Goodman, B. E. Ambrose, H. Latin, and S. Finnell, "Network operations analyzer and assistant (NOAA): A real-time traffic rerouting expert system," in *Globecom*, Florida, Dec. 1992.

[2] ——, "Network operations analyzer and assistant (NOAA): A hybrid neural network/expert system for traffic management," in *IFIP*, San Francisco, Apr. 1993.

[3] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*, vol. 1, *Santa Fe Institute Studies in the Sciences of Complexity*. Reading, MA: Addison-Wesley, 1991.

[4] S. E. Fahlman, "Faster-learning variations on back-propagation: An empirical study," in *1988 Connectionist Models Summer School*. San Francisco: Morgan Kaufmann, 1988.

[5] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neur. Computat.*, vol. 1, pp. 270–280, 1989.

[6] J. M. Akinpelu, "The overload performance of engineered networks with nonhierarchical and hierarchical routing," *Bell Syst. Techn. J.*, pp. 1261–1280, Sept. 1984.

[7] F. P. Kelly, "Bounds on the performance of dynamic routing schemes for highly connected networks," *Mathemat. Operat. Res.*, vol. 19, no. 1, pp. 1–21, Feb. 1994.

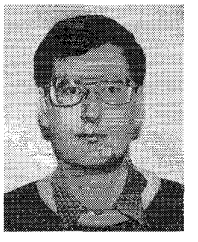
[8] D. Mitra and J. B. Seery, "Comparative evaluations of randomized and dynamic routing strategies for circuit-switched networks," *IEEE Trans. Commun.*, vol. 39, pp. 102–116, Jan. 1991.

[9] D. Mitra and R. J. Gibbens, "State-dependent routing on symmetric loss networks with trunk reservations-ii: Asymptotics, optimal design," *Ann. Operat. Res.*, vol. 35, pp. 3–30, 1992.

[10] M. Schwartz, *Telecommunications Networks*. Reading, MA: Addison-Wesley, 1987.

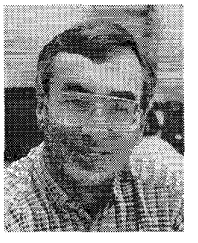
[11] R. B. Cooper, *Introduction to Queueing Theory*, 2nd ed. New York: Elsevier, 1981.

[12] J. D. Farmer and J. J. Sidorowich, "Predicting chaotic time series," *Phys. Rev. Lett.*, vol. 59, no. 8, pp. 845–848, Aug. 1987.



Barry E. Ambrose received the B.E. degree from University College, Cork, Ireland, the M.Sc. degree from Trinity College, Dublin, Ireland, and the Ph.D. degree in electrical engineering from the California Institute of Technology, Pasadena, CA, in 1986, 1990, and 1995, respectively.

Prior to his present position with AGL Systems, Pasadena, CA, he worked for the Network Planning Department of the Irish Telephone Company (Telecom Eireann).



Rodney M. Goodman received the B.Sc. degree from Leeds University, U.K., and the Ph.D. degree from the University of Kent, U.K., in 1968 and 1975, respectively.

In 1975, he joined the the California Institute of Technology, Pasadena, where he is a Professor of Electrical Engineering. He is currently the Director of the NSF Center for Neuromorphic Systems Engineering at Caltech.