SYMBOLIC AND CONNECTIONIST MACHINE LEARNING TECHNIQUES FOR SHORT-TERM ELECTRIC LOAD FORECASTING

by

Jayendar Rajagopalan

Thesis Submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

ELECTRICAL ENGINEERING

APPROVED:

Saifur Rahman, Chairman

Yilu Liu

Hugh F. VanLandingham

December 1993
Blacksburg, Virginia

SYMBOLIC AND CONNECTIONIST MACHINE LEARNING TECHNIQUES FOR SHORT-TERM ELECTRIC LOAD FORECASTING

by

Jayendar Rajagopalan

Saifur Rahman, Chairman

ELECTRICAL ENGINEERING

(ABSTRACT)

This work applies connectionist neural network learning techniques and symbolic machine learning techniques to the problem of short-term electric load forecasting. The short-term electric load forecasting problem considered here is the prediction of bus loads one day ahead. The forecast quantities of interest are average integrated daily load and daily peak load.

The primary objectives of this work are two-fold: to determine the forces driving the load demand and produce a human intelligible model, and use of this model to forecast load for new, unseen scenarios.

In the first part of this work, connectionist techniques for modeling bus load is presented. Critical design issues for neural network modeling and implementation such as neural network architecture, training database creation, training dataset selection, training data normalization are presented in context of nonlinear modeling in general and electric load forecasting in particular. Local function approximation and nearest neighbor norms techniques are applied to this task. Simulations are performed for forecast of average bus loads of the town of Blacksburg, Virginia, U.S.A; the connectionist model is able to forecast integrated average daily load with an accuracy of about 2.5%. Connectionist

neural network knowledge acquisition algorithms are however, not mature enough, presently, to handle complex real world problems such as knowledge extraction from large databases. Presence of symbolic along with numeric data in input and output poses problems for data pre-processing for neural network training. Only at the time of completion of this thesis are researchers discussing the possibility of using special techniques to present symbolic data for neural networks. Also, multilayer feedforward networks trained by the backpropagation algorithm perform poorly in forecasting chaotic patterns such as those encountered in peak load demand.

Symbolic machine learning techniques are powerful concept acquisition techniques that extract underlying knowledge from large databases. They are sufficiently powerful to accept symbolic and numeric data. Inductive learning algorithms employing a statistical $\chi 2$ test as the splitting criterion are applied to extract load dependency information. The extracted patterns are expressed as graphic decision trees and equivalent human intelligible high level language if-then rules. Implementation details of the statistical decision algorithm are discussed and simulations are performed to construct decision trees. Using this model, new cases are forecast. This algorithm is capable of forecasting holiday and weekend loads too. The proposed algorithm is robust enough to handle raw, unprocessed databases which contain missing data. The peak load forecasting problem is solved using a simple methodology that combines the robustness of decision trees and the numerical accuracy of connectionist models.

The two paradigms, connectionist and symbolic learning techniques are compared from a knowledge acquisition and forecasting perspective and directions for further work suggested.

ACKNOWLEDGMENTS

I would like to take the opportunity to express my sincere appreciation and gratitude to my advisor Dr. Saifur Rahman for making the graduate study at Virginia Tech possible and a fruitful one. I sincerely believe that I could not have progressed without his unlimited caring support, patience and advice throughout my study here. I would like to thank Drs. Hugh F. VanLandingham and Yilu Liu for being on my committee.

I would like to thank past and present members of the Energy Systems Research Lab Dr. Saher Lahouar, Irislav Drezga, Yonael Teklu, Arnie deCastro, Mohammad Chowdary for making work at this lab a pleasure.

My gratitude to my parents for letting their only son go to far away lands for long periods of time. My salutations to Mother, Father, my teachers and Lord Nataraja for their constant guidance and support. They have helped me in taking every step in life and I pray for their well being and continued guidance. The author would also like to thank Ashok Patil, Dr. Nalini Ramesh, Sridhar Kowdley, Giridhar N. Rao, Dr. T.S Venkataraman and Sharon Anderson for their help and in making life a pleasant one.

TABLE OF CONTENTS

1.0 INTRODUCTION 1

1.1 Short Term Electric Load Forecasting 1

1.1.1 Short-Term Load Forecasting System Components 2

1.1.2 Scope of work 4

1.1.3 System Load Modeling 6

1.1.4 Review of Load Forecasting Literature 8

1.2 Connectionist Models for Short Term Electric Load Forecasting 10

1.3 Review of Connectionist Electric Load Forecasting Literature 12

1.4 Symbolic Machine Learning Techniques 14

1.5 Symbolic Machine Learning Techniques in Power Engineering 15

1.6 Objectives of Work 16

2.0 CONNECTIONIST MACHINE LEARNING TECHNIQUES FOR SHORT-TERM ELECTRIC LOAD FORECASTING 19

2. 1 Connectionist Models and Algorithms 19

2.1.1 The Artificial Neuron and Transfer Function 20

2.1.2 Activation rule 23

2.1.3 Architecture 24

2.1.4 Learning Algorithms 29

2.2 Mathematical Formulation of the Error Backpropagation algorithm 30

2.3 Physical Insight into Error Backpropagation Action 33

2.4 Implementation Issues of the EBPN algorithm..... 38

2.5 Case Studies: Neural Network Forecast of Electric Load 41

2.5.1 Average Load Forecasts 42

2.5.2 Daily Peak Load Forecasting 49

2.6 Knowledge Extraction Using Neural Networks 50

2.7 Conclusion..... 52

3.0 SYMBOLIC MACHINE LEARNING TECHNIQUES FOR SHORT-TERM ELECTRIC LOAD FORECASTING 61

3.1 Hierarchy and classification of machine learning systems 62

3.2 Inductive Learning Methods..... 63

3.3 Tree construction methods..... 67

3.3.1 Tests based on information entropy..... 67

3.3.2 Tests based on expected error..... 69

3.3.3 Tests based on statistical significance testing:..... 69

3.4 The Statistical Decision Tree algorithm 71

3.4.1 Statistical Significance Testing..... 71

3.4.2 Significance Level Adjustments..... 74

3.5 Case Study: Daily Load Forecast by Decision Tree Methodology..... 77

3.5.1 Average Load Forecasts 78

3.5.2 Maximum Load Forecasts 83

3.6 Two-step methodology to peak load forecast 84

3.7 Conclusions 85

4.0 RESULTS, CONCLUSIONS AND RECOMMENDATIONS 104

4.1 Results and Conclusions 104

4.2 Recommendations 106

BIBLIOGRAPHY 108

VITA 111

LIST OF ILLUSTRATIONS

Figure 1: Artificial Neuron and Transfer Function pp.22

Figure 2: Architecture of a Multilayer feedforward neural network pp.27

Figure 3: Architecture of a Recurrent neural network pp.28

Figure 4: Signal flow in multilayer feedforward networks pp.34

Figure 5: Multilayer feedforward neural network with linear feedthrough term pp.35

Figure 6: Flow chart of error backpropagation algorithm pp.36

Figure 7: Effect of momentum term on convergence pp.40

Figure 8: Variation of forecast error with number of hidden nodes pp.53

Figure 9: Neural network forecast of average integrated load, January 1992 pp.54

Figure 10: Neural network forecast of average integrated load, February 1992 pp.55

Figure 11: Neural network forecast of average integrated load, March 1992 pp.56

Figure 12: Forecast error of average integrated load: Jan., Feb., March 1992 pp.57

Figure 13: Neural network forecast of daily peak load, January 1992 pp.58

Figure 14: Neural network forecast of daily peak load, February 1992 pp.59

Figure 15: Forecast error of daily peak load: January, February, March 1992 pp.60

Figure 16: A General binary decision tree architecture pp.66

Figure 17: Algorithm of $\chi 2$ significance testing pp.72

Figure 18: Sample of rules for the average daily load of January 1992 pp.79

Figure 19: Decision tree obtained by statistical decision tree learning pp.87

Figure 20: Decision tree forecast of average integrated load, January 1992 pp.88

Figure 21: Decision tree forecast of average integrated load, February 1992 pp.89

Figure 22: Decision tree forecast of average integrated load, March 1992 pp.90

Figure 23: Average load forecast error, decision tree methodology, January 1992 pp.91

Figure 24: Average load forecast error, decision tree methodology, Feb. 1992 pp.92

Figure 25: Average load forecast error, decision tree methodology, March 1992 pp.93

Figure 26: Decision tree forecast of peak load, January 1992 pp.94

Figure 27: Decision tree forecast of peak load, February 1992 pp.95

Figure 28: Decision tree forecast of peak load, March 1992 pp.96

Figure 29: Peak load forecast error, decision tree methodology, January 1992 pp.97

Figure 30: Peak load forecast error, decision tree methodology, February 1992 pp.98

Figure 31: Peak load forecast error, decision tree methodology, March 1992 pp.99

Figure 32: Two-step method of peak load forecast, January 1992 pp.100

Figure 33: Two-step method of peak load forecast, February 1992 pp.101

Figure 34: Two-step method of peak load forecast, March 1992 pp.102

Figure 35: Two-step method forecast error: January, February, March 1992 pp.103

LIST OF TABLES

Table 1: Correlation between input variables and load demand pp. 44

Table 2: $\chi 2$ test of significance: Observed frequencies pp. 72

Table 3: $\chi 2$ test of significance: Expected frequencies pp. 73

Table 4: Forecasts based on rules generated by statistical decision trees pp. 82

Table 5: Comparison of 1-step and 2-step methods of peak load forecast pp. 86

CHAPTER I

INTRODUCTION

1.1 Short Term Electric Load Forecasting

Short term electric load forecasting (STLF) is prediction of electric load over an interval ranging from an hour to one week. STLF plays an important role in online scheduling and security functions of an Energy Management System (EMS). The close tracking of the system load by system generation at all times is a basic requirement in the operation of a power system. For economically efficient and reliable operation and effective control, this must be accomplished over a broad spectrum of time intervals. In this research work, the time scale of interest is over a period of hours and days which involves startup or shutdown of entire generating units, determination of interchange transactions with neighboring power systems, hydro scheduling, unit commitment, hydro-thermal co-ordination, fuel and maintenance scheduling. Thus STLF plays a key role in the formulation of an economic, reliable and secure operating strategy.

The basic quantity of interest in STLF is typically, the hourly integrated total system load. In addition, STLF is concerned with forecasting of daily system load and daily and weekly system energy. The principal objective of STLF is to provide the load predictions for the following three tasks:

- the basic generation scheduling function
- security assessment of power system at any point of time,
- timely information for the dispatcher

The primary application of STLF is to drive the *scheduling function* that determines the most economic commitment of generation sources consistent with reliability requirements, operational constraints and policies, and physical, environmental, and equipment limitations. The hydro-scheduling, unit commitment and hydro-thermal co-ordination function requires system load forecasts for the next day or next week to determine the least cost operating plans subject to the various constraints imposed on system operation.

The second application of STLF is for *predictive assessment* of power system security. The system load forecast is an essential data requirement of the off-line network analysis function for detection of future conditions under which the power system may be vulnerable. This information permits the dispatchers to prepare the necessary corrective actions such as bringing peaking units online, load shedding, power purchases, switching operations, etc., to operate the power system securely. This requires the knowledge of peak system load at any point of time during the forecast interval. Maximum load forecasting is performed to predict this quantity.

The third application of STLF is for providing the power system dispatchers with *timely information* of system load demand, with latest weather prediction and random behavior taken into account. The dispatchers need this information to operate the system economically and reliably.

1.1.1 Short-Term Load Forecasting System Components

The major components of a STLF system are the STLF model, data sources and the man-machine interface.

The STLF model implements the system load representation and the STLF algorithms. The data sources are the historical load and weather databases, the parameter databases, manually entered data by the dispatchers, and the real-time data obtained from the automatic generation control function of the EMS and the data link to the weather forecasting service. The manually entered data include weather updates, load forecasts, parameter data, or execution commands. In general STLF models use integrated load (MWh) data. The telemetered measurements in the real time databases are used by the AGC to determine the "measured" loads which are typically, integrated (and consequently smoothed) before they are used by the STLF model. The outputs of the STLF are provided to the dispatcher workstations and other EMS functions that require the load forecasts.

The timeliness and accuracy of STLF have significant effects on power system operations and production costs. System dispatchers must anticipate the system load patterns so as to have sufficient generation to satisfy the demand. At the same time, sufficient levels of spinning reserve and standby reserve are required to mitigate the impacts of *uncertainty* inherent in forecasts and in the availability of generating units. The cost of reserves is high since the units that make up the reserve are not fully loaded and consequently may be operating at less than their maximum efficiencies. Thus by reducing forecast error, reserve levels may be reduced without affecting the reliability and security of the system. In this way, operating costs are reduced. In addition, forecast error in load predictions results in increased operating costs. Underprediction of load results in failure to provide the necessary reserves which, in turn translates to higher costs due to use of expensive peaking units. Overprediction of load, on the other hand, involves the startup of too many units resulting in an unnecessary increase in reserves and hence operating costs.

1.1.2 Scope of work

Classifying electric load forecasting according to the size of forecast area, there are three types of forecasts performed:

- Region load forecasting
- Area load forecasting
- Bus load forecasting

These techniques are concerned with disaggregation of system load into smaller components. Region load forecasting, usually referred to as just load forecasting is the usual method of load forecasting performed by utilities. The entire service region is considered as a single load, without differentiating between any load consumption patterns in the system. Utilities with a wide range of geographical zones or structural subunits with climatic diversity, usually called areas, perform area load forecasting for obtaining the forecast of the total area load. Bus load forecasting (B.L.F) provides predictions of loads at key buses. These forecast values are used in system and area load forecasts. This research work is focused on bus load forecasting. B.L.F is a much more difficult task than area or region load forecasting due to the strong effects of few events and loads. Performing a thorough search of load forecasting literature, no special techniques or algorithms have been specifically developed for bus load forecasting [1]. Algorithms developed for region load forecasting have been applied for bus load forecasting, applying exactly the same parameter identification schemes. This is very surprising, especially considering the fact that bus load forecasting involves taking into account special non-numeric factors that dominate the load demand strongly. Conventional statistical techniques cannot be applied to B.L.F as the representation of few strong non-numeric variables are not amenable for representation in statistical models. Consequently, it is

expedient to develop new approaches to forecasting bus loads. A typical case of B.L.F is prediction of electric load of the town of Blacksburg, Virginia, U.S.A. This is an university town and the predominant consumption pattern is dictated by the work pattern of the university, Virginia Tech. Special events such as wind tunnel experiments, large machinery operation in various engineering departments, football games are potential sources of large stochastic loads. Geographically, Blacksburg is situated in New River Valley at an elevation of about 2200 feet above sea level and is susceptible to large and sudden weather variations. These factors compound the problem of bus load forecasting for this town. Virginia Tech Electric Service (VTES) which supplies most of the town of Blacksburg is responsible for operating a small power plant. It orders extra power to meet the load from the Appalachian Power Company a month in advance. If the amount of demand exceeds the energy ordered, the cost of power rises very steeply. It was desired by this organization to have an understanding of the forces driving the load demand so as to plan their purchases efficiently. Here it is not only sufficient to forecast the load accurately, but also to have a human intelligible form of the load forecasting process. Connectionist and symbolic machine learning techniques have rich representational capabilities, and are very powerful non-parametric models and function approximators. By the use of these techniques it is possible to not only come up with a forecast quantity but also determine the quality of forecast, sensitivity of load to deviations of predictor variables from their forecast or estimated values. Thus the scope of this work is to:

- determine the forces driving the load demand and produce a human intelligible model
- using this model, forecast load for new, unseen scenarios
- determine the quality of forecast, in terms of uncertainty associated with forecast.

1.1.3 System Load Modeling

The system load is the sum of all the individual demands at all nodes of the power system. In principle, one could determine the system load pattern if each individual consumption pattern were known. However, the demand or usage pattern of an individual load (device) or customer is quite random and highly unpredictable. Also, there is a very broad diversity of individual usage patterns in a typical utility. These factors make it impossible to predict the system demand levels by extrapolating the estimated individual usage patterns. Fortunately, however, the totality of the individual loads results in a distinct pattern which can be statistically predicted. The system load behavior is influenced by a number of factors. There are four major influencing factors:

- economic
- time
- weather
- random effects

The *economic environment* in which the utility operates has a clear effect on the electric demand consumption patterns. However, the economic climate of a region has a time scale of the order of years and does not enter into the short-term load forecasting model directly.

Three principal *time factors* - seasonal effects, weekly-daily cycle, and legal and religious holidays, play an important role in influencing load patterns. The seasonal changes determines whether an utility is summer or winter peaking. Sudden seasonal changes such as start of school year and vacation periods are important factors that need to be modeled. The existence of statutory holidays has a significant effect in lowering the

demand. This is accentuated by the modifications to consumption pattern by tendencies of extending the weekends. These are important factors and should be explicitly modeled.

Meteorological conditions are responsible for significant variations in the load pattern. This is because most loads have large components of weather-sensitive loads such as those due to space heating, air conditioning, and agricultural irrigation. In many systems, the weather variable is the most significant in terms of its effects on loads. The important weather variables that need to be considered in the load modeling process are maximum, minimum and average temperatures, deviations of temperature from normal, past temperatures, humidity, thunderstorms, wind speed, precipitation, cloud cover and light intensity.

Random disturbances are those that cause a variation in the load consumption pattern and cannot be explained in terms of any of the previously discussed factors. A power system is continuously subject to random disturbances reflecting the fact that the system load is a composite of a large number of diverse individual demands. In addition to a large number of very small disturbances, there are large loads - steel mills, synchrotrons, wind tunnels - whose operation cause large and sudden variations in electricity usage. These represent large unknown disturbances, as their operating schedules are unknown to the power system operator. There are also certain events such as special television programs whose occurrence is known *a priori*, but whose effect on load is unknown. This factor can be explicitly represented in the model as a random white noise process; Alternatively, they can be left without any explicit representation and a statistical estimate of uncertainty associated with effects of these events given. In this work, the latter method is used.

1.1.4 Review of Load Forecasting Literature

The technical literature displays a very wide range of methodologies and models for region load forecasting. Since no two utilities are identical, there is limited portability of a STLF model from one utility to another. On the other hand, the wide spectrum of techniques and standard algorithms can be tailored to the particularities of a specific system. As mentioned earlier there are allied functions of region load forecasting, area load forecasting and bus load forecasting. Utilities with a wide range of geographical zones or structural subunits with climatic diversity, usually called areas, perform area load forecasting for obtaining the forecast of the total area load. Bus load forecasting provides predictions of loads at key buses. Here are reviewed STLF algorithms and models that are relevant and applicable to bus load forecasting.

The system load is a random nonstationary process composed of thousands of individual components each of which behaves erratically without following any known physical law. As a result, all macroscopic models are empirical in nature and can be objectively evaluated only through extensive experimental evidence. The classification of STLF algorithms based on *type of load model used* is adopted. This criteria is useful in understanding both the load modeling and algorithm and is of direct relevance in formulating the neural network model for short term electric load forecasting.

Using the above classification criteria, there are two basic models:

- Peak Load Models
- Load Shape Models

a. Peak Load Models : In this class of models, the daily or weekly peak load is modeled, as a function of weather. Time does not play a role in such models which are typically of the form peak load = base load + weather-dependent component. i.e.

$$P = B + F(W) \tag{1.1}$$

where B is an average weather-insensitive load component to which the weather-dependent component F(W) is added. The weather variables W include the temperature at peak load time, a combination of predicted and historical temperatures, humidity, light intensity, wind speed, and precipitation. The function F(.) is empirically computed and is usually nonlinear. The advantages of peak load model are its structural simplicity and its relatively low data requirements to initialize and update. The parameters of the model are estimated through nonlinear regression. The disadvantages of such models are that they do not define the time at which peak load occurs, nor do they provide any information about the shape of the load curve. Models described in [2,3] belong to this category.

b. Load shape models : Such models describe the load as a discrete time series process over the forecast interval. The load sampling time interval is typically one hour or one day, while the quantity measured is generally the energy consumed over the sampling interval in MWh. Dynamic load models recognize the fact that the load is not only a function of the time of day, but also of its most recent behavior, as well as that of weather and random inputs. Dynamic models belong to two classes: *autoregressive moving average* (ARMA) models and state space models. However since every ARMA model has an equivalent state space model, the discussion is restricted to ARMA models. The ARMA model takes the general form

$$z(t) = y_p(t) + y(t) \tag{1.2}$$

where z(t) is the load at a discrete sampling time t of the forecast period, $y_p(t)$ is a component which depends primarily on the time of day and on the *normal* weather pattern for the particular day, y(t) is an additive load residual term describing influences due to weather pattern deviations from normal and random correlation effects. The additive nature of the residual load is justified by the fact that such effects are usually small compared to the time of day component. Nonlinear models describing the interaction of the periodic and residual components exist. The impact of the weather dependent variables is significant. The unknown parameters of the system are obtained by fitting the simulated models data to observed load and weather data. Models described in [4,5] belong to this category. In the load forecasting work, described in this dissertation, load is defined as the sum total of 24 hourly load readings recorded for one day. This is called the "Integrated Load". In essence, the integrated load combines the peak and troughs of the load curve. This integrated load quantity forms the basic building block for forecasting other values of load such as maximum load, minimum load and energy consumption patterns.

1.2 Connectionist Models for Short Term Electric Load Forecasting

Connectionist models, also known as artificial neural network models are composed of many nonlinear computational elements operating in parallel and arranged in patterns reminiscent of biological neural networks. They are physical cellular systems which acquire, store and utilize experiential knowledge. The computational elements, called neurons or nodes, are connected via weights that are typically adapted during use to improve performance. Connectionist models attempt to achieve good performance by adaptation of their structure to the problem being solved. Adaptation or learning is by adjustment of connection weights in time to improve performance based on current results, is a major focus of research in the neural network community.

Neural networks are non-parametric models. They prove to be more robust than statistical models when distributions are generated by non-linear processes that are strongly non-Gaussian. Once a network is trained, a network's response can accurately interpolate the region of the problem space it has learned. The artificial neural network can thus learn and approximate functions to any arbitrary degree of accuracy. In this work, use is made of the neural network's ability to perform *function approximation* for nonlinear modeling and forecasting. Conventional statistical techniques make underlying assumptions about the model to be determined. Use of neural networks with efficient training schemes mitigates the deleterious effects of reliance on large historical databases with possible obsolete and irrelevant data, assumptions of static load shapes and parameters. Important, but less relevant points in favor of neural networks for forecasting, are its capabilities of distributed knowledge storage and recall, fault tolerance, and adaptive pattern recognition capabilities.

In recent years, application of neural networks has gained momentum. The major advantage of neural networks is the self-learning capability. First, the network is presented with a set of correct input and output values. It adjusts the connection strength among the internal nodes during learning until the proper transformation is learned. Next, the network is presented with only the input data and it produces a set of output values. The development of the proper connection strength, or transformation function requires presentation of the input and output through several epochs. After a certain number of learning cycles the network will be able to produce accurate output data from input data similar to those used for learning. Neural networks trained by the above mechanism are powerful function approximators [6]. This suggests that the neural network may be suitable for load forecasting. The learning ability permits the continuous upgrading of the

network parameters which assures that the change of the underlying load shape will not affect the accuracy of the forecast.

1.3 Review of Connectionist Electric Load Forecasting Literature

In the power systems community, neural networks have been applied extensively to electric load forecasting and surpass all other areas in power engineering in the number of research publications. The enthusiasm of applying artificial neural networks and the hope of obtaining results superior to conventional statistical methods have been sustained primarily by two factors: the non-linear function approximation capabilities of neural networks and, the continuous improvements to neural network learning algorithms. In the following paragraphs, important developments in neural network applications to short-term electric load forecasting are reviewed, rather than providing an enumerative listing of all publications.

The first attempt to apply artificial neural networks to electric load forecasting was the work of Peng *et. al* [7]. In their work, the utilization of neural networks was investigated and justified by the problems of traditional approaches. Time series approaches utilizing autoregressive moving average models suffered the risk of numerical instability and high errors because they do not utilize weather information. Here the authors trained the neural network with all available data and modeled the entire 24 hour period with a single neural network. This is a global function approximation technique and notorious for low accuracy in modeling. As an alternative, El-Sharkawi *et. al* [19] presented an artificial neural network model for each hour of the day. The proposed artificial neural network used was a three layer network trained using the error backpropagation algorithm. Six working days were used to train the network and the forecasted temperature information was used as an input. Many heuristically chosen

architectures were tested over 30 weekdays on data of Washington state. The mean average percentage error for weekday hourly forecast was 2.06%. A similar approach was proposed by Peng *et. al* [9]. A neural network was used to generate the total daily load forecast. The approach was tested using 12 weeks data. The average errors ranged between -5.29% and +6.17%.

A nonfully connected network was proposed for short-term load forecasting by Cheng *et. al* [31]. The authors claimed that a fully connected network could not pick fluctuations in the weather and the proposed architecture would require a shorter training time. Three layers were used, and the input variables were determined using a autoregressive integrated moving average procedure. Two weeks of past data were used to train the network for forecasting the following week. This is an explicit time-series approach for load forecasting.

At the First International Forum on Neural Network applications to Power Systems [11], numerous results of feedforward neural network architectures for electric load forecasting were published. Two papers stand out from the rest; The first by Connor *et. al* [13] applied recurrent neural networks for electric load forecasting. Recurrent neural networks have feedback connections which enables previous states influence partially, future states of the network. This is very similar to time series and autoregressive approaches to neural network forecasting. Though recurrent networks displayed a performance inferior to that of statistical models and feedforward networks, proper design procedures of architecture, selection of training data and training procedures would help reap the full advantages of the superior theoretical properties of the recurrent model. The second paper was a comparison of forecast accuracy of neural network and statistical methodologies performed by M.C. Brace *et. al* [14]. The conclusion drawn was that artificial neural networks are *not a panacea to the forecaster*. Choice of input variables,

neural network architecture, training algorithm, training databases are extremely important and should be based on firm mathematical foundations. Part of the work detailed in this dissertation is devoted towards developing these methodologies for neural network modeling of nonlinear systems and forecasting.

In the Second International Forum on Neural Network applications to Power Systems [12], scheduled for April this year, 19 papers are scheduled in load forecasting sessions. They deal with data partitioning for neural network training, forecasting abnormal load conditions, heuristic parameter tweaking for weekends and seasonal changes, load curve shape forecasting, application of new topologies such as recurrent and Kohonen's self organizing unsupervised learning scheme and field implementation results. Soon, a lot of new techniques would be available for scrutiny, analysis and implementation.

1.4 Symbolic Machine Learning Techniques

An important reason neural network have not met widespread acceptability within the power industry for electric load forecasting is the "blackbox" concept associated with them. Neural networks produce a number $\in [0, 1]$ which is transformed to produce a forecast load in MWh. The uncertainty associated with forecast, sensitivity of load to various factors is not available. Most important of all, the operating personnel do not have an understanding of the forces driving the load demand and the reason for the neural network's output. Neural networks are accurate function approximators given good input data. Recently, it has been shown that feedforward neural networks are not robust with respect to input errors and weight errors [15]. The gain associated with the errors while propagating from input to output is greater than 1. It is thus imperative to develop a robust decision making system with which neural network forecasts can be verified.

Symbolic machine learning techniques and algorithms extract general concept descriptions of a process from a given set of examples and express it in human intelligible graphic or natural language form. The basic scenario for this branch of learning is the presence of an intelligent agent, who shown a collection of examples of a particular activity, employs an inference technique to derive useful information about such activity.

Inductive symbolic machine learning methods employ a top-down divide-and-conquer strategy for recursively partitioning a given data set into homogenous groups till a certain prespecified degree of intragroup homogeneity is reached. Inductive learning methods produce graphic "decision trees" and equivalent natural language "if-then" rules that compactly describe the key relationships between input variables and the dependent variable. The most important subpart of the decision tree construction process is the choice of splits at nodes. The knowledge extracted from large databases is used in forecasting new scenarios. It should be noted that symbolic machine learning techniques perform load forecasting in an order exactly reversed to the manner performed by neural networks. In symbolic machine learning techniques, knowledge is first extracted and forecasting performed next using knowledge obtained. In connectionist machine learning techniques, neural networks are first trained to provide an accurate forecast, and next the knowledge encoded as weights is extracted.

1.5 Symbolic Machine Learning Techniques in Power Engineering

There has been only one published instance of this methodology in power engineering and none in the load forecasting area. Decision trees have been applied to the real time transient stability assessment of electric power systems [16]. Transient stability is concerned with the ability to withstand severe contingencies. A measure of the stability margin of a system is the critical clearing time (C.C.T) i.e. the maximum time that a

contingency such as a fault may remain without causing the irrevocable loss of a machine's synchronism. In the tree methodology, the C.C.T provides a convenient means of classifying the state of a power system as stable or unstable. One decision tree is built per postulated contingency. For the operating parameters under consideration, the tree classifies the post contingency condition as either stable or unstable. During online operation, the power system EMS personnel can check the current system state for several possible contingencies and make sure that unstable conditions would not be caused by any contingencies, by taking suitable remedial control actions. Due to the highly structured format of the training set, the ID3 algorithm is easily applied to decision tree construction.

The load forecasting problem is a more difficult problem to solve due to presence of both numeric and non-numeric data as input variables and also due to the fact that there is no classification state into which the load demand can be classified into. The learning system has to perform autonomous grouping of loads to form homogenous clusters of data. Application of Classification and Regression Methodology for hourly load forecasting was performed by Les Atlas *et. al* [17]. In their investigation the objective was comparison of forecasting accuracy only and not knowledge extraction and machine learning of databases.

1.6 Objectives of Work

The objective of this work is to extract knowledge and provide a clear understanding to the power system dispatcher, the forces driving the bus load demands. The idea is to compress and organize large quantities of information present in load and weather databases in the form of compact decision trees with the twofold objective of uncovering salient parameters driving the load consumption patterns and to forecast new, unseen cases. The knowledge extracted can be verified with operator developed heuristics.

Thus, understanding how the system operates, the operating personnel would have a sense of confidence in their forecasting tool. An earlier work by Rahman *et. al* [10] on the expert system for electric load forecasting relied on the presence of an expert who could formulate accurate and consistent rules without bias. The performance of the expert system relied on the knowledge acquired from him or her through verbal analysis protocol. Also there was no guarantee that an expert would be available. Machine learning of databases as described in this work, avoids the reliance on the availability of an expert.

The quantities forecast are the integrated energy and maximum load for next day. The maximum load is the highest point on the load shape curve. This information would be used by power system personnel to plan the energy inputs required and for running contingency simulations of the network. In this work, strategies for neural network architecture selection, database creation, training data set selection, data normalization and training procedure applicable to short term electric load forecasting are presented. The neural network architecture strongly resembles conventional statistical models, incorporating the linear and non-linear functional relationships between input variables and load demand. This method of model building upon the strengths of statistical models, by non-parametric neural network function approximation schemes, enhances the accuracy of forecasts. The database creation techniques uses a "relevant time window concept" obtained by concatenating subparts of the entire load and weather databases. The training data set selection criteria improves the efficient use of a limited number of highly similar historical cases selected though a nearest neighbor norm criteria. The neural network is trained to perform local function approximation.

The connectionist models acquire and store knowledge as numerical connection weights. Translation of weights into human intelligible form is essential to label neural networks as knowledge acquisition systems. Research work has been in progress in recent

years to build special algorithms and tools that convert the numerical knowledge into "if-then" rules. These systems use the neural network as their "knowledge base" for making inferences. Applicability of these neural network knowledge extraction algorithms for electric load forecasting applications is examined. The above mentioned aspects of connectionist load forecasting are detailed in Chapter 2.

In Chapter 3, symbolic machine learning techniques for extracting knowledge from databases for forecast of average and maximum daily loads are developed. Issues related to decision tree growing, testing criteria, stop splitting criteria are developed and discussed. A new two-step methodology is proposed, where peak load forecasting is performed as a two step process combining the robustness of decision trees and accuracy of neural networks. Simulations are performed and results prove the viability of the proposed approach for short term bus load forecasting.

In Chapter 4, connectionist and symbolic machine learning techniques for short-term electric load forecasting are compared and directions for further work discussed.

CHAPTER II

CONNECTIONIST MACHINE LEARNING TECHNIQUES FOR SHORT-TERM ELECTRIC LOAD FORECASTING

The objective of applying connectionist machine learning techniques is two-fold: To extract knowledge about the system internal structure and dynamics from input-output relationships and use this knowledge to forecast load. Neural networks are powerful function approximators and can approximate complex nonlinear functions with a proper architecture, input training data and learning algorithm. In the first part of this chapter, these design issues for nonlinear function approximation and forecasting are developed and presented. Simulations are performed to forecast daily average "integrated" load and daily peak load. In the second part, the applicability of algorithms to extract knowledge from the connection weights in context of the load forecasting application is examined.

2. 1 Connectionist Models and Algorithms

Connectionist models, known to many as neural networks, are biologically inspired methods of architecture and algorithms that aim at improved performance in many tasks; Nonlinear modeling of processes and systems, forecasting, pattern recognition, speech processing, control of dynamic systems are but a few of the successful applications. These models attempt to achieve superior performance via dense interconnection of simple computational elements. Ideas drawn from neuropsyiology have led to their architecture being a dense interconnection of large number neurons and learning being the adaptation of weights of connections between neurons. Knowledge is encoded as connection weights. Neural networks since their renaissance in the late 1970s, have been the subject of great

research and recently have been applied to many real world problems and have in most cases equaled the performance of conventional techniques and in some cases surpassed them too. [20]. Neural network paradigms abound in architecture and learning algorithms since they were developed for specific classes of applications.

Connectionist models are specified by the following characteristics:

- transfer function of neuron
- activation rule
- architecture
- learning rule

2.1.1 The Artificial Neuron and Transfer Function

The basic building block of a neural network is a neuron. Its name is taken after the biological neuron and attempts in an elementary way to mimic its biological cousin. Incorporating what little has been understood from the mammalian brain, an artificial neuron interacts with other neurons through weighted links, equivalent of the axon. The output of the artificial neuron is a nonlinear function of the weighted sum of the inputs. This simple unit, connected to many other similar artificial neurons, modifies the connection weights incrementally to adapt the behavior of the massive interconnected artificial neural network. Neural networks trained with data sampled from an external system whose structure and dynamics are unknown, learn to behave like the external system. This characteristic of neural networks is used for building non-parametric models of complex systems.

Shown in Figure [1] is the neuron, the basic building block of a neural network. It receives inputs x_0, x_1, x_2, x_{n-1} from an external environment through weighted connections. Each connection is weighted by a factor w_i. The output of the first stage is sum of the weighted inputs. This is input to a nonlinear transfer function, which depends on the complexity of the model to be simulated and range of outputs. Commonly used transfer functions are the linear transfer function, sigmoidal transfer function, the hyperbolic tangent transfer function and the radial basis function. The linear transfer function is linear in the range [0,1] and produces an output proportional to input. It is used to model systems where the output is a linear function of the inputs. The most commonly used transfer function is the sigmoidal transfer function. It has a dynamic range of [0.2, 0.8] and a saturation regions in [0, 0.2] and [0.8, 1] approximately. In load forecasting applications, the output is the forecast load of the next day with finite *positive* values. Thus the neural networks used for the simulations are constructed using this type of transfer function. Hyperbolic tangent transfer function has a greater dynamic range of about [-0.8, 0.8]. Typically, they are used for applications where the output varies between a finite positive amount and a finite negative amount. This avoids the need for linear transformations of data for fitting within the [0,1] range of the sigmoidal transfer function. Recently Radial Basis Functions [6] have evoked a great deal of interest as the optimal transfer functions. Using the theory of function approximation, it has been shown that these transfer functions can map any nonlinear function exactly. However their mathematical intractability causes difficulty in weight updates. Though variations of these transfer functions exist, they are essentially variants of the above categories. It should be noted that the capabilities of the multilayer perceptrons stem from the nonlinearities used within artificial neural nodes.

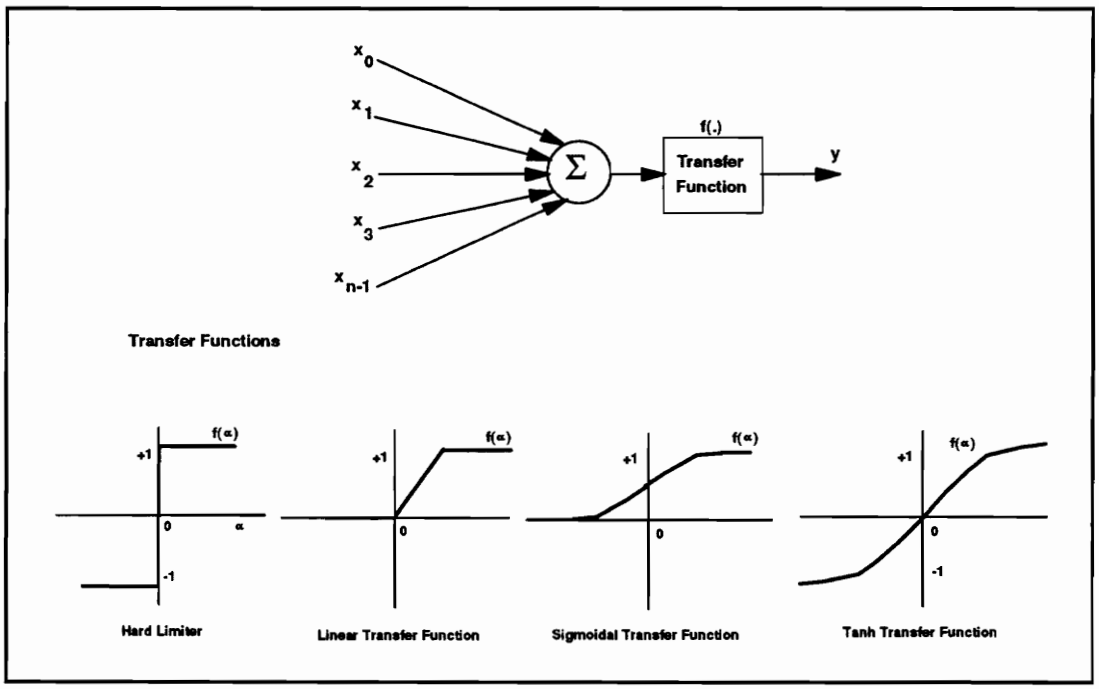

Figure 1: The Artificial Neuron and Transfer Functions

2.1.2 Activation rule

The activation rule determines the activation state of the neuron which in turn determines the overall behavior of the network. The strengths of connections, the weights, can be represented as a weight matrix W whose ij^{th} component w_{ij} is the weight of the connection from unit j to unit i; If the output of unit j at time t is $a_j(t)$ then the incoming signal along the connection from unit j to unit i is given by $w_{ij}a_j(t)$. The combined input to the unit, along with its current activation state, determines its new state according to an activation rule. The most common form for such a rule is the additive rule

$$x_i(t+1) = \sum_{j=1}^{N} w_{ij} a_j(t) + I_i, \qquad i = 1,\ldots,N \tag{2.1}$$

in which the state of unit at time t+1 is the linear sum of all the inputs at time t. Here I_i denotes an external bias input. Using a sigmoidal transfer function, and equation (2.1) an input along a positive connection weight tends to increase the activation, whereas along a negative weight it tends to decrease the neuronal activation. This activation rule is implemented in the research work conducted.

An extension to the above activation rule is to take

$$x_i(t+1) = k_i x_i(t) + \sum_{j=1}^{N} w_{ij} a_j(t) + I_i, \qquad i = 1,\ldots,N \tag{2.2}$$

where k_i is a scalar whose value is less than one. The significance of this term is the inclusion of memory trace of the previous activation state, which decays at a rate k_i. The presence of this term enables the network to track temporal signals effectively. In case of hourly load forecasting, with a time-series type of approach, this kind of neuron is of great significance. The model can be extended to higher order of recurrence, with the inclusion of previous memory states. In our approach to load forecasting, in contrast to time series

approach to modeling, a *local function approximation* approach is followed. The town of Blacksburg is susceptible to large and sudden changes in weather and load. Blacksburg being an university town, the largest consumer of power is the university and its related offices. The load consumption is dictated by the work patterns of the university; In addition to this, the weather is a factor that determines the load consumption. As the time series approach to modeling does not help in forecasting of chaotic load patterns, a local function approximation method is used.

2.1.3 Architecture

The architecture of a neural network specifies how the neurons are organized and connected. Fundamentally links between neurons are either unidirectional or bi-directional according to the permitted flow of signals. Examples of neural networks with bi-directional links are the Hopfield networks, Kohonen's self-organizing feature maps, Kosko's Bidirectional associative memories and Robert Hecht Nielsen's Counterpropagation networks. Typically these networks are applicable to optimization, partial recall and content addressability tasks. Neural networks with unidirectional links such as multilayer feedforward networks and recurrent networks, are applicable to nonlinear modeling, function approximation and pattern recognition tasks. Consequently the research described here and the discussion is directed towards the application of the family of multilayer feedforward networks.

Neurons in feedforward networks are arranged as units in groups or layers. There are three types of units: input, output and hidden. Input units receive external inputs either directly from a sensor or from other parts of an information processing system such as a database. Some inputs are clamped to a constant activation value. These inputs are called bias inputs. The output units send signals out of the network which correspond to the

result of the networks computation. The hidden units are those, whose inputs and outputs are within the network; they are invisible to the outside world. There are two standard architectures:

- Multilayer feedforward networks
- Recurrent networks

a. Multilayer feedforward networks: This architecture is a hierarchical design consisting of fully interconnected neighboring layers of processing units or neurons. Each layer consists of several individual processing elements. The macroscopic-scale detail of the feedforward architecture is shown in Figure [2]. The first "input" layer consists of processing elements that simply accept the individual components x_i of the input vector x and distribute them, without modification to all of the units of the second row. Each neuron on each row hereafter receives the output signal of the preceding layer and performs a nonlinear operation on the weighted sum of inputs. The nonlinear operation is defined by the exact transfer function used with each neuron. This continues through all of the rows of the network until the final row. The final row of the network consists of k units and produces the networks estimate y' of the correct output vector y. The activation dynamics of a feedforward network is completed after a few steps, equaling the number of layers. Activation patterns within the network proceed unidirectionally from the input layer through the intermediate layers to the output layer.

The operation of a feedforward network may be viewed in terms of performing a nonlinear mapping between a set of inputs and a set of outputs. The information processing operation that the feedforward network carries out is the approximation of a bounded mapping of function f: $A \subset R^n \rightarrow R^m$, from a compact subset A of a n-dimensional Euclidean space to a bounded subset f[A] of a m-dimensional Euclidean space, by means

of training on examples (x_1,y_1), (x_2,y_2),...,(x_k,y_k) of the mapping, where $y_k=\Phi(x_k)$. The samples of the mapping f are generated by selecting x_k vectors randomly from a pool of examples generated by a process; The operational use to which the network is put after "training", the weight adaptation procedure, is by providing the network, new inputs sampled from the system; The neural network outputs values, that would be expected of the physical sampled system.

Multilayer feedforward networks overcome many of the limitations of single layer perceptrons, but were not used in the past because effective credit assignment algorithms were not available. This scenario changed with the development of the error backpropagation algorithm [18].

b. **Recurrent networks**: consists of groups of units which feedback on itself. A simple form of a recurrent network is shown in Figure [3]. It consists of a single layer of neurons. Each neuron receives, in addition to the bias input, an input from an external environment along a variable weight connection, and a feedback signal from its output along a variable weight connection. Variations of this basic architecture exist. An example of a practical structured recurrent neural network is a multilayer feedforward network with additional feedback loops from output to the input. The recurrent network implements a *spatiotemporal* mapping of a point $x \in R^n \rightarrow y \in R^m$. The activation dynamics of a recurrent network is more complicated than that of a feedforward network due to the presence of feedback connections. Recurrent networks exhibit properties very similar to short term memory in humans, in that the state of the network output depends partly on their previous states. This architecture is very powerful in modeling time series processes and nonlinear modeling using data obtained by sequentially sampling a process.

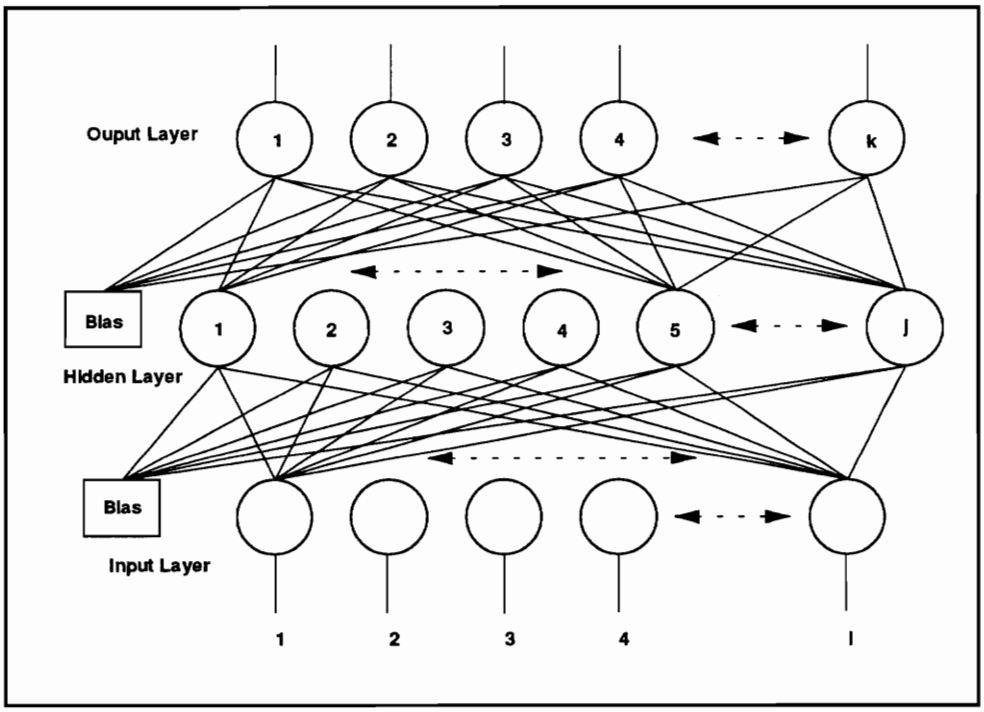

Figure 2: Architecture of a Multilayer Feedforward Neural Network

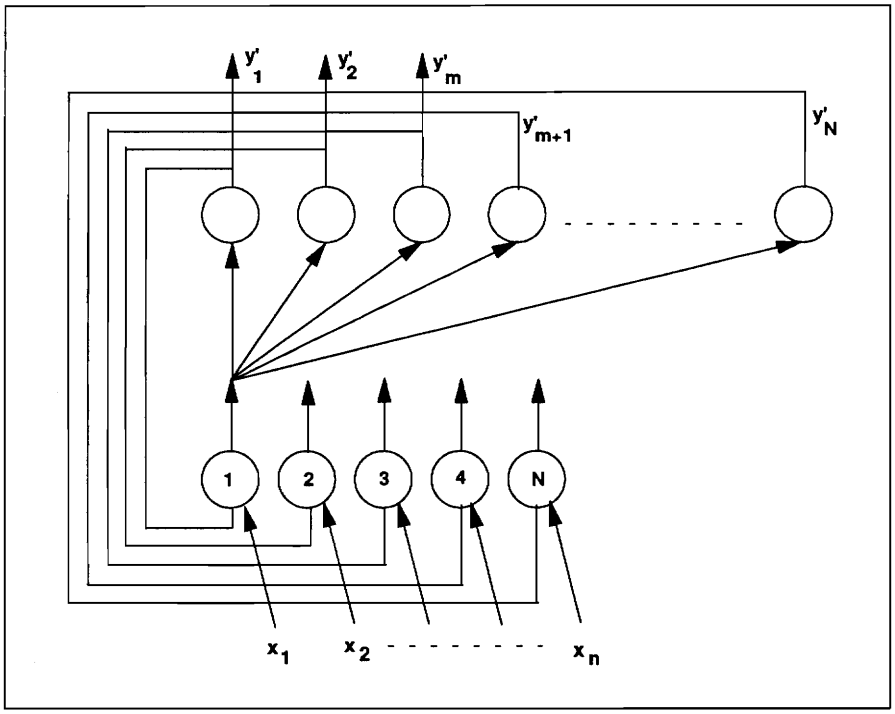

Figure 3: Architecture of a Recurrent Neural Network

2.1.4 Learning Algorithms

Learning algorithms in neural networks are procedures for adaptation of connection weights between neurons. The discussion is restricted to training algorithms for multilayer feedforward networks. The following sections explain *how* and *why* the algorithm works.

The error backpropagation algorithm (EBPN) can be said to the foremost rejuinivating factor in neural network history. The EBPN algorithm, has been successfully applied to a wide variety of problems. The training procedure is a repetitive application of a two step procedure: The first step is the *forward pass* where inputs to the neural network "bubble up" towards the output. The second step is the *backward pass* when errors "percolate down" from top (output side) to the bottom (input side). This process continues until the network reaches a satisfactory level of performance measured by the difference between true output and neural network estimated output.

Each layer in a neural network is composed several units. The units of the input and output layers are specified by the dimensions of input and output vectors respectively. Each unit in a layer can be visualized to be composed of a *sun* and several *planet* processing elements (Figure 4). The number of planets for each sun is determined by the number of suns in the preceding layer. Each planet receives 2 inputs: one from a sun of the preceding layer and a feedback from its own sun. Each planet produces two outputs: one to its own sun and a feedback to the sun from which it receives signals. It should be noted at the outset that the magnitude and time of activation of each input and output signals are distinct and different. The output row suns receive the "correct answer" y_i for their component of the output vector on each training trial. The scheduling of the network's operation during training consists of two "sweeps" through the network: the forward pass

starts by inserting the input vector x_i into the input layer. The signals are transmitted through the hidden layers to the output layer. While passing through the layers they combine with other signals and are subjected to nonlinear transformations as per the node transfer function. The output layer emits the network estimate which is compared to the true output, to produce a scalar value of error estimate. Now the network modifies its weights so as to minimize the output error. The backward pass commences when output suns compute the errors and transmit back to their planets, values that determine the magnitude of weight change. Further, the planets transmit these values to the suns, lower down in the layers for their planets to compute the weight changes. This cycle is repeated till satisfactory performance is obtained from the network.

2.2 Mathematical Formulation of the Error Backpropagation algorithm

Consider the general feedforward neural network with linear feedthrough term as shown in Figure [5]. The network has i nodes in the input layer, j nodes in the hidden layer and k nodes in the output layer. The weight matrices of the connection strengths are w_{ji} for input-hidden layer connections, w_{kj} for hidden-output layer connections and w_{ki} for the linear feedthrough part. Training assumes that each input vector is paired with a target vector representing the desired outputs; together these are called a training pair. Inputs applied to the network flow towards the output.

Training of a multilayer feedforward network is performed by application of the following steps:

1. Select the next training pair from the training set; apply the input vectors to the network input.

2. Calculate the output of the network.

3. Calculate the error between the network output and the desired output.

4. Adjust the weights of the network in a manner so as to minimize output error.

5. Repeat steps 1 through 4 for each vector in the training set until the error for the entire training set is acceptably low.

The net input to any node in hidden layer j is

$$net_j = \sum_i w_{ji} o_i \tag{2.3}$$

where w_{ij} is the hidden-input layer weight matrix, o_i the inputs applied to input layer node i. The output of any node is layer j is

$$o_j = f(net_j) = f(\sum_i w_{ji} o_i) \tag{2.4}$$

For a sigmoidal transfer function

$$o_j = \frac{1}{1+e^{-(net_j+B_j)}} \tag{2.5}$$

where B_j is the bias term. The net input to any node in layer k is

$$net_k = \sum_k w_{kj} o_j + \sum_i w_{ki} o_i \tag{2.6}$$

The first term on the right hand side of the equation (2.6) is the output of the hidden layer nodes and the second term is the output of the linear feedthrough term. The output of any node in layer k is

$$o_k = f(net_k) = f(\sum_j w_{kj} o_j) + f_l(\sum_i w_{ki} o_i) \tag{2.7}$$

where $f_l(.)$ is the linear transfer function:

$$f_l(x) = \{x, 0 \le x \le 1\} \tag{2.8}$$

Let t_{pk} be the true output of any node k in the output layer for pattern p, and o_{pk} the actual network output; The error of the neural network is taken as the L_2 norm of the output error vector:

$$E_p = \frac{1}{2}\sum_k (t_{pk} - o_{pk})^2 \tag{2.9}$$

The objective function is to minimize the error, by adaptation of weights; Presentation of a dataset (x, y) produces some error at the output. After each presentation the connection weights are updated by small amounts:

$$\begin{aligned} w_{kj}^{(new)} &= w_{kj}^{(old)} + \Delta w_{kj} \\ w_{ji}^{(new)} &= w_{ji}^{(old)} + \Delta w_{ji} \\ w_{ki}^{(new)} &= w_{ki}^{(old)} + \Delta w_{ki} \end{aligned} \tag{2.10}$$

Implementation of the learning procedure requires an iterative gradient descent method to change the weights incrementally; Starting at the output layer, the weights are changed by an amount Δw_{kj} so as to minimize the output error:

$$\Delta w_{kj} = -\eta \frac{\partial E}{\partial w_{kj}} \tag{2.11}$$

where η is a parameter called the co-efficient of learning. Computation of the above equation requires evaluation of the right hand side. This is conveniently performed using the chain rule of derivatives;

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial net_k}\frac{\partial net_k}{\partial w_{kj}} \tag{2.12}$$

Using the above equations (2.6)-(2.9) and (2.12), equation 2.11 is:

$$\Delta w_{kj} = \eta(t_k - o_k) f_k^{'}(net_k) o_k \tag{2.13}$$

The weights of the input-to-hidden layer connections also have to be changed in proportion to the output error. The change in weights Δw_{ji} is computed by the chain rule:

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}} = -\eta \frac{\partial E}{\partial net_k} \frac{\partial net_k}{\partial o_j} \frac{\partial o_j}{\partial net_j} \frac{\partial net_j}{\partial w_{ji}} \tag{2.14}$$

The above equation is evaluated to be:

$$\Delta w_{ji} = \eta f_j^{'}(net_j) o_i \sum_k (t_k - o_k) f_k^{'}(net_k) w_{kj} \tag{2.15}$$

The weights of the output-to-input connections, forming the linear feedthrough term are updated by a value:

$$\Delta w_{ki} = -\eta \frac{\partial E}{\partial w_{ki}} = \eta (t_k - o_k) o_i \tag{2.16}$$

Thus the essence of the algorithm is the evaluation and correction of contribution of each particular weight to output error. This algorithm is shown graphically as a flowchart in figure [6].

2.3 Physical Insight into Error Backpropagation Action

The objective of training the network is to adjust the weights so that application of a set of inputs produces the desired set of outputs. The objective function is minimization of error at the output. The error backpropagation algorithm (EBPN) performs a gradient descent along the error surface of the network and reaches a minimal error point corresponding to the converged state of the network. The properties of the error surface are thus important in determining the convergence of the algorithm towards a stable and accurate solution.

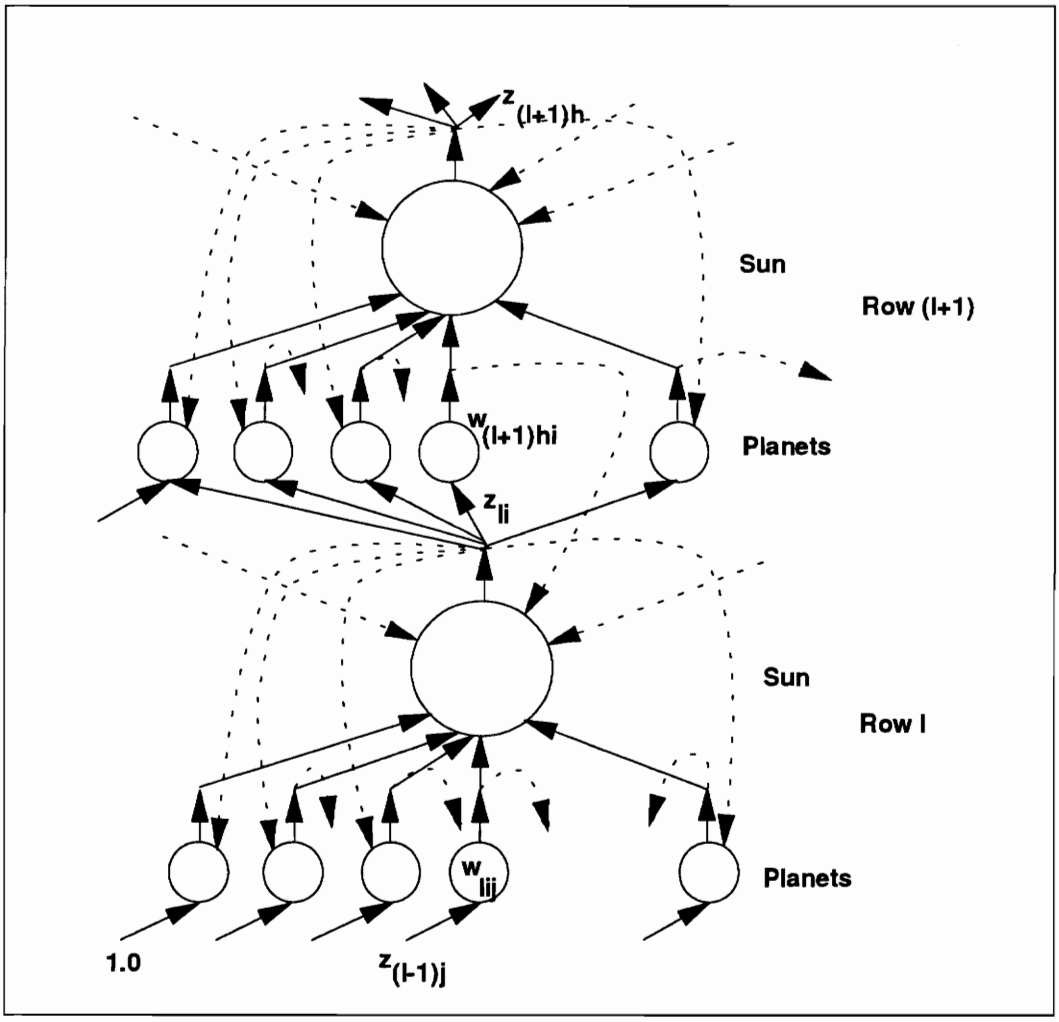

Figure 4: Signal Flow in multilayer feedforward networks

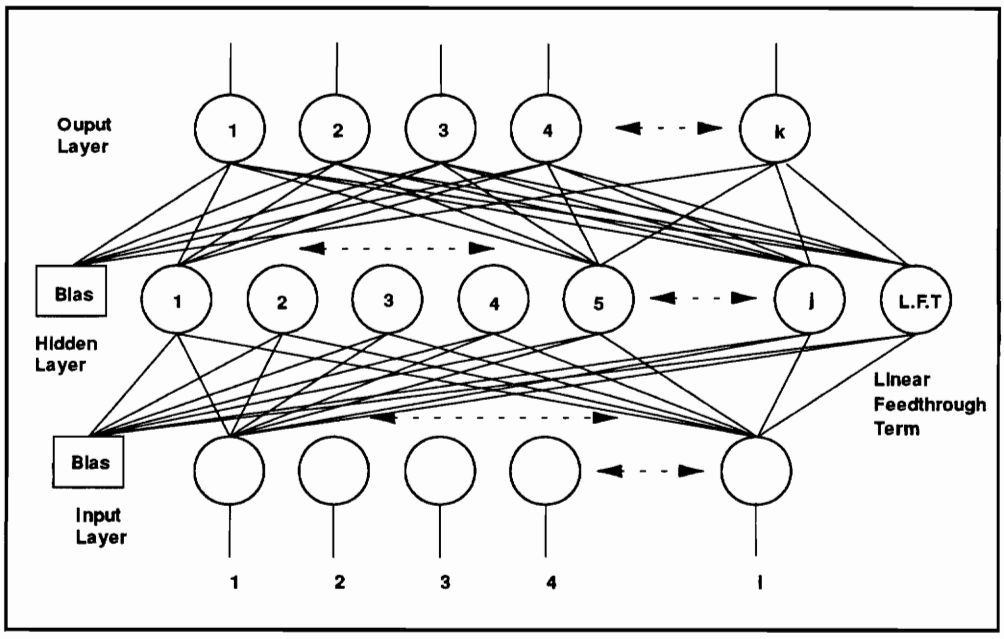

Figure 5: Multilayer Feedforward Neural Network with Linear Feedthrough Term

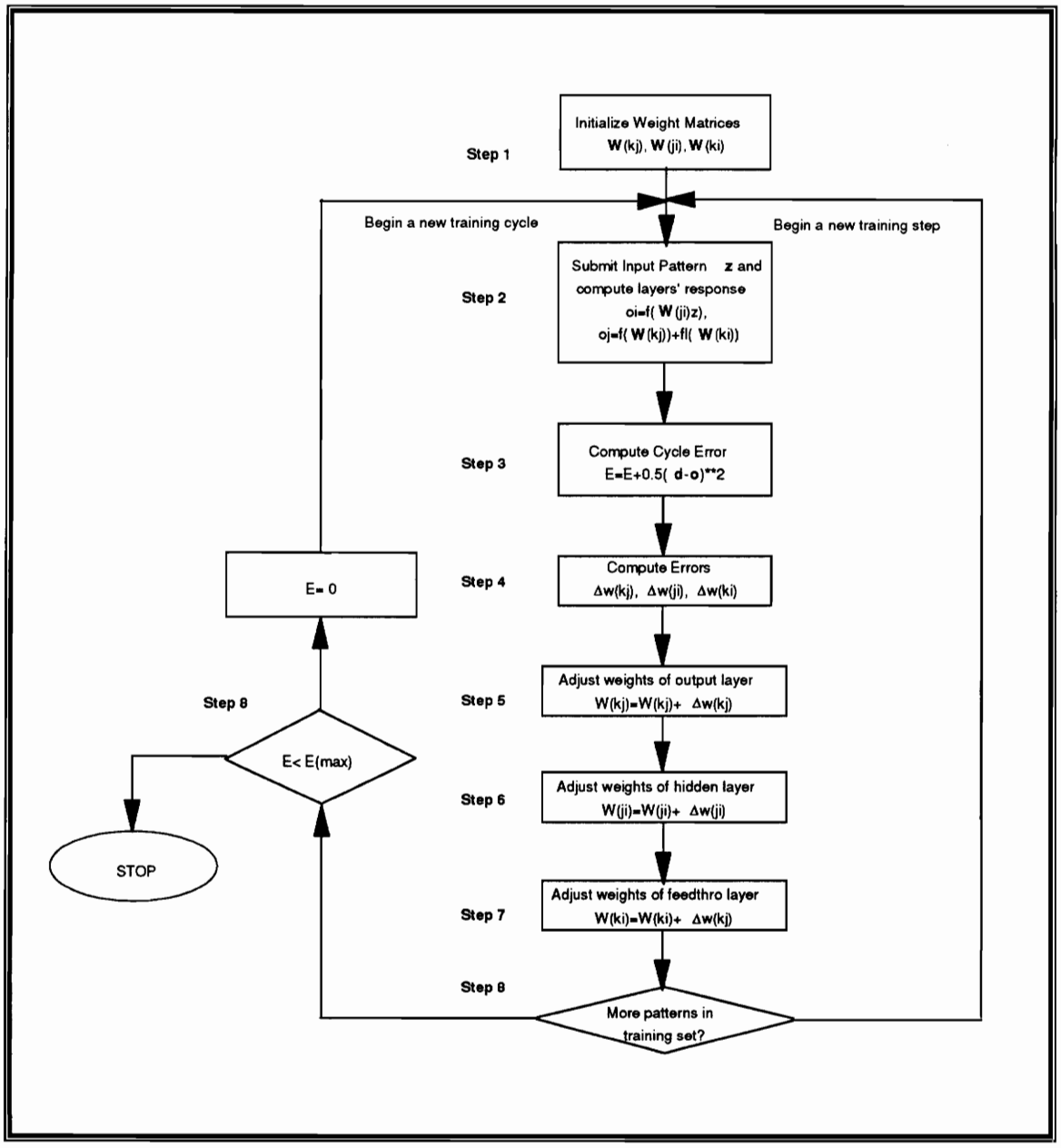

Figure 6: Flow chart of Error Backpropagation Algorithm

The error of the network for every pattern presented to it can be expressed as

$$F_k = |\Phi(x_k) - B(x_k, w)|^2 \tag{2.17}$$

where F_k is the error, x_k is the *k*th pattern presented to the network, $f(x_k)$ is the true output, $B(x_k,w)$ is the network estimate of the output for given weight matrices **w**. When the network is presented with infinitely many cases, the mean square error F(w) is

$$F(w) = \underset{N \to \infty}{Lt} \frac{1}{N} \sum_{k=1}^{N} F_k \tag{2.18}$$

The error surface of a EBPN network is the $(q+1)$ dimensional surface, q being the number of weights w. The variable w ranges over the q-dimensional space R^q and for each w a non-negative error surface height F is defined by F(w). The EBPN algorithm has the property that given any starting point $\mathbf{w}_0$, on the error surface that is not a minimum, the learning law will modify the weight vector w so that |F(w)| decreases. i.e. it goes downhill from the starting point. Three facts about error surfaces influence strongly the manner of use of the EBPN algorithm: a) The error surfaces, in general, have extensive flat areas and troughs with little slope. The implication is that it is necessary to move weights over a considerable distance before a significant drop in error occurs. b) The error surfaces have many *global* minima. Thus there exist many weight permutations that yield exactly the same overall input-output performance. This fact *helps* in ensuring convergence of *random* initial weights on the connections. c) The error surfaces have many *local* minima. This is a troublesome feature and modifications to the above basic algorithm are required to alleviate this problem.

2.4 Implementation Issues of the EBPN algorithm

Enhancements to the above basic algorithm are required to ensure convergence and, convergence within a reasonable time. In this section, various measures to ensure good function approximation are presented:

1. Adding a momentum term: This is a method for improving the training time of the EBPN algorithm while enhancing the stability of the process. It involves adding a term to the weight adjustment equation which is proportional to the previous weight change. Once an adjustment is made, it is "remembered" and serves to modify all subsequent weight changes. The weight adjustment equation incorporating the momentum term is

$$\Delta w_{kj}(t+1) = \eta(t_k - o_k) f_k^{'}(net_k) o_k + \alpha \Delta w_{kj}(t) \qquad (2.18)$$

where α is the momentum coefficient. The effect of addition of momentum term is illustrated in Figure [7]. Consider the 2-dimensional projection of the 3-dimensional error surface of simple 2 input 1 output network. Let the gradient descent commence at point A". The first change in weight from A' to A" was $\Delta w(t)$. The change in weight from A" to A"' is $\eta \nabla E(t+1) + \alpha \Delta w(t)$. As gradients at A' and A" are in the same direction, the resulting effect is summation of the two terms, hastening convergence towards M. Progression of learning causes the state of the network to B" through B'. Note that gradients at B' and B" are different. When network weights are updated at B", vector addition of gradient directions $\nabla E(t+n)$ and previous weight change $\Delta w(t+n-1)$, produces a net smaller weight change which leads to convergence towards M, the global minimum. Thus the momentum term helps to speed up convergence leading to an efficient and reliable learning profile. Incorporating the momentum term, the network tends to follow the bottom of narrow gullies in the error surface rather than crossing rapidly from side to side.

2. Adding a neuron bias: Addition of a neuron offsets the origin of the logistic function producing an effect that fixes the threshold level of the neuron. This permits rapid convergence of the training process.

3. Step size: The step size of change of weights is dependent on the components of the right hand side of equation (2.18). The only quantity under control of user is the learning coefficient α. The convergence to a stable set of weights is guaranteed only under a infinitesimal weight changes [18]. This implies a very long training time. Thus selection of the learning coefficient requires a tradeoff between the speed of convergence and actual convergence. There is very little to guide the decision other than experience. Also it is very problem dependent. In this work a learning coefficient of 0.35 is used.

4. Network paralysis: Large values of input data, large values of learning constant or inadequate network architecture cause a network to paralyze. This is the condition when connection weights do not change as training progresses. All neurons operate in a saturation mode as the derivative of transfer function f'(net) in this region is almost zero. This causes the right hand side of equation (2.18) to be very small leading to network paralysis. The most potent cures are proper choice of network hidden layer nodes and *preprocessing of input data.* Control of hidden layer activation dynamics is not possible from the input side and the only panacea is to chose adequate number of hidden layer neurons. Preprocessing of input data, so as to center it in the center of the dynamic range of the transfer function helps in partly alleviating the paralysis problem. Let us consider use of a sigmoidal transfer function. Each variable of the input data is first scaled between [0,1] and then transformed so that the mean of the datasets of each variable is centered to the middle of the transfer function. The output data are suitably descaled during testing to produce an estimate. This technique is applied to the training set of this work and satisfactory results are observed.

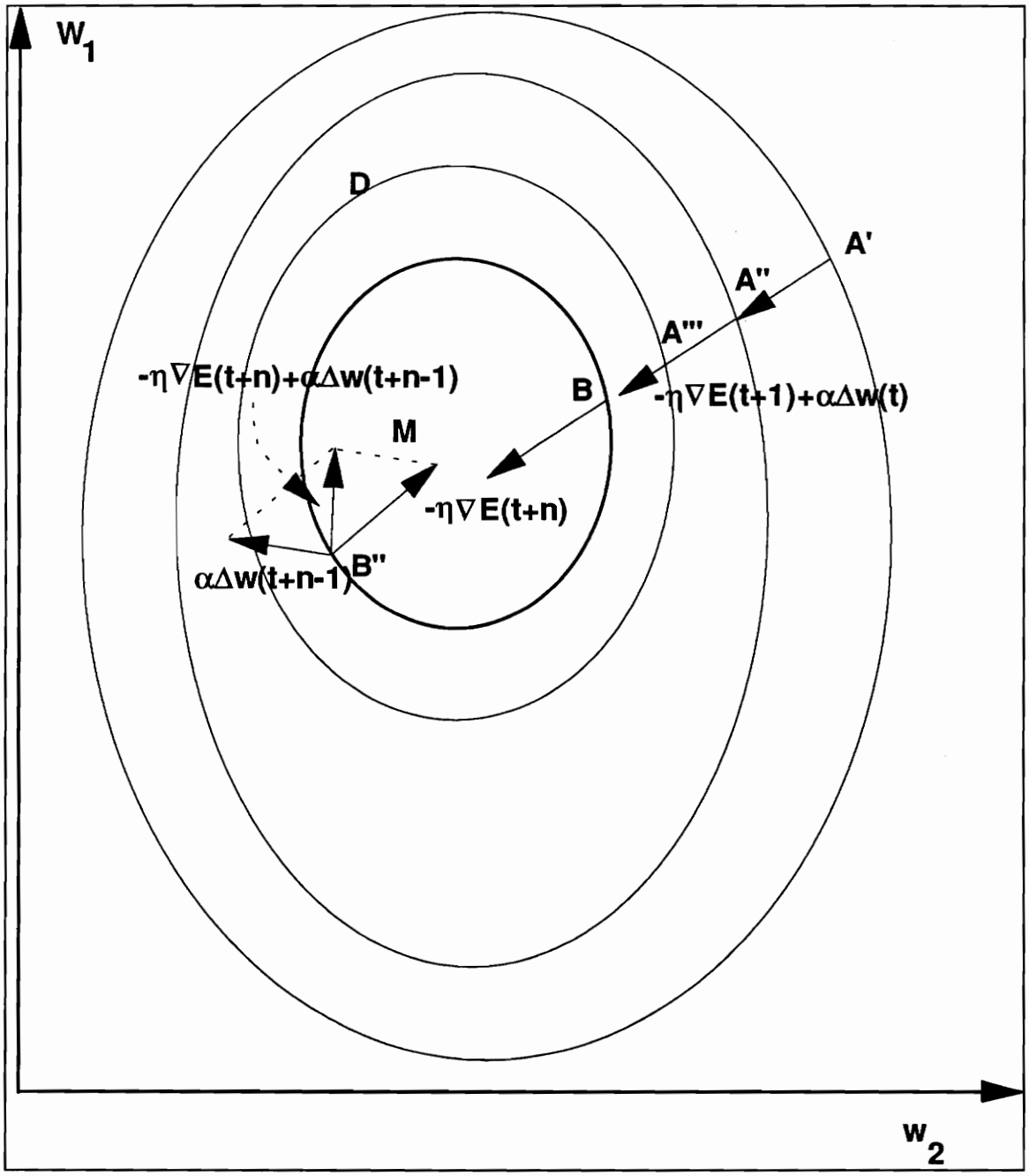

Figure 7: Effect of Momentum Term on Convergence

2.5 Case Studies: Neural Network Forecast of Electric Load

The first step is to prepare the data for neural network training. This step is actually quite detailed and crucial to the performance of the network and shall be best understood if subdivided into the following sub-parts:

- Input data identification
- Training vector selection
- Training data normalization

In the following paragraphs, the above issues are discussed sequentially;

Load data (in kW), temperature data (in degrees Fahrenheit), wind data (in m/sec), global horizontal insolation, vertical insolation data sampled at every 15 minutes was available from Virginia Tech Electric Service and the Solar Test Facility of the Department of Electrical Engineering. Using this data, forecasts had to be made for the average load demand and maximum load demand of next day for the town of Blacksburg. The average load demand or "Integrated Load" is the per hour equivalent of the total energy consumed during the 24 hour period assuming the energy consumed is constant throughout the day. This value is easy to use as the operators at the utility can compare this value with the values they are familiar with, the metered readings. Any other numerical value would require some manipulation before the operators can compare with the meter readings. The time of peak load is known easily from the work pattern or weather conditions and the forecast quantity of interest is the peak load. The daily peak load forecast is the highest point on the load shape curve and is of greater practical value to the electric utility. Using this information, the utilities plan their transactions, generation scheduling and generation mix.

The available data was transformed to a comprehensive hourly load and weather database. Fifteen minute data samples were averaged out to form hourly load data. The available data was screened to determine the relevant predictor variables. The output variables of interest are average daily load (kW) and maximum daily load (kW). The predictor variables were subjected to a correlation analysis and the results are tabulated in Table [1]. Predictor variables with the highest correlation magnitudes constitute the inputs in the training data sets. The only exception was the wind variable. It was included as a highly non-linear effect of wind on load demand was suspected. In the following sections, forecast methodologies of average and peak daily load are detailed.

2.5.1 Average Load Forecasts

The input variables of the training sets are *identified and selected* for average load forecasts through correlation analysis are:

- Average load of previous day
- Average daily temperature
- Maximum daily temperature
- Minimum daily temperature
- Wind
- Global Insolation

The training sets are chosen according to the following procedure:

Step 1: The month whose average daily load and maximum daily load is to be forecast is determined.

Step 2: The time window comprising of one month before and after the forecast month is chosen.

Step 3: Time window segments for the past 3 years are extracted from the load consumption database. The available databases at the utility determined this factor.

Step 4: Month wise "Load growth factors" (LGF) are computed by the following formula:

$$\text{LGF} = \frac{\text{Total energy consumption of (Month M: Year i)}}{\text{Total energy consumption of (Month M: Year (i-1))}} - 1$$

Step 5: Historical load data of the months comprising the time window segments are updated using load growth factors.

Step 6: The time window segments are concatenated into a single training database.

Artificial neural networks have a transfer function in [0,1]; To avoid network paralysis and ensure fast convergence the input data should be suitably normalized between [0,1]. The following *normalization* procedure is performed on each training set variable:

Step 1: Determine the maximum magnitude (**V-MAX**) of the variable type encountered.

Step 2: Divide every entry of the variable by **V-MAX**. Now all entries of the particular variable type are $\in$ [0,1].

Step 3: Compute the arithmetic mean (**V-AM**) of values of the variable.

Step 4: Perform a linear transformation to center **V-AM** at the center of the dynamic range of the neuron transfer function.

Step 5: Repeat Steps 1-4 for all input variables.

Variable #	Measurement Variable	Average Load	Maximum Load
1	Average Load	+ 1.00	+ 0.96
2	**Average Load (t-1)**	**+ 0.52**	**+ 0.45**
3	Δ(Avg. load(t)-Avg. load(t-1))	+ 0.27	+ 0.13
4	Δ(Avg. load(t)-Avg. load(t-2))	+ 0.37	+ 0.13
5	Maximum Load (t-1)	+ 0.38	+ 0.02
6	Δ(Max. load(t)-Max. load(t-1))	+ 0.23	+ 0.03
7	Δ(Max. load(t)-Max. load(t-2))	+ 0.30	+ 0.09
8	**Avg. Temp(t)**	**- 0.72**	**- 0.47**
9	Avg. Temp(t-1)	- 0.35	- 0.07
10	Avg. Temp.(t-2)	+ 0.01	- 0.06
11	**Max. Temp.(t)**	**- 0.61**	**- 0.14**
12	Max. Temp(t-1)	- 0.59	- 0.06
13	Max. Temp(t-2)	- 0.01	- 0.07
14	**Min. Temp.(t)**	**- 0. 69**	**- 0.24**
15	Min. Temp(t-1)	- 0.14	- 0.09
16	Min. Temp(t-2)	+ 0.03	- 0.03
17	**Wind**	**- 0.02**	**- 0.12**
18	**Global Insolation**	**- 0.64**	**- 0.11**

Table 1: Correlation between input variables and load demand

The above data normalization procedure is very effective for neural network training when combined with *local function approximation* (LFA) and *nearest neighbor norms(NNN)*, discussed below.

A function $\Phi(.)$ can be represented with an arbitrary degree of accuracy if there are enough free parameters and enough data to solve it stably. However if $\Phi(.)$ is complicated, there is no guarantee that any function F(.) will approximate $\Phi(.)$ accurately. The dependence of accuracy on representation can be reduced through LFA. The basic idea is to break up the domain $\Phi(.)$ into local neighborhoods and fit different parameters in different neighbors. LFA usually produces better fits for a given number of data points than Global Function Approximation (GFA). Conversely, the accuracy of representation can be increased through LFA as compared to GFA. This is especially true for large databases and data-rich cases such as electric load forecasting. Most GFA representation schemes reach point of diminishing returns when adding more parameters or more data only gives a marginal improvement of accuracy. Higher order terms may oscillate, and actually cause behavior to get worse. This problem is analogous to the Runge Interpolation phenomenon [33]. Adding more neighborhoods through LFA techniques is usually more significant than adding more parameters and going to higher order schemes. LFA makes it possible to use a function representation efficiently. The key is to choose the local neighborhood size properly, so that each neighborhood has just enough points to make local parameter fits stable, so that adding more points would not make significant improvements.

The load demand can be visualized as a hypersurface formed by the 6 input variables. As revealed by correlation analysis, the surface is highly nonlinear; From the cyclic nature of the load demand process, the surface is repetitive in time. The load forecasting problem is not a real time task and LFA can be applied. In LFA, the global

load demand hypersurface is approximated in a piecewise nonlinear manner by connectionist models. The segment modeled is the part of the load demand hypersurface that is estimated to resemble closely the next days load demand hypersurface. The choice of training vectors is through the NNN method.

We define a M point neighborhood of x as a collection of M points $\{y_i\}$ which are in some sense "close" to x. The "closeness" referred to here is measured by vectors norms of training vector sets with respect to forecast values. Computationally easy L_1 norms are used. To use LFA in the vicinity of point x, there are three basic steps to be followed:

Step 1: Pick a local representation. This is a particular neural network architecture. In the following sections, a neural network architecture using ideas from conventional statistical load forecasting is identified.

Step 2: Assign a neighborhood. This corresponds to the size of the hyperspace to be modeled and determines the number of training vectors to be chosen.

Step 3: Determine a local mapping chart. This corresponds to mapping of points in the neighborhood of the future values of input variables to the output variable.

The function representation scheme is a neural network. The above three steps imply that the following three issues have to tackled in context of connectionist modeling:

- Neural network architecture
- Neural network training
- Neural network testing

Feedforward multilayer neural networks are powerful function approximators [6]. The number of input variables determines the number of input layer nodes. From correlation analysis, six input variables were identified and these six input-layer-nodes shall

be used. The output layer nodes correspond to the forecast quantities of interest. In this case, the average integrated daily load and the maximum (peak) daily load are quantities that have to be forecast from known past load and forecast weather data. As the daily load forecasting is not a time constrained task, average and peak load forecasts are done separately. The neural networks approximate the respective functions independently. The number of hidden layer nodes is a very difficult decision for most problems. Algorithms such as Cascade Correlation [21] exist for autonomous growth of neural network hidden layer nodes to minimize output error along with adaptation of weights. In load forecasting, this is difficult to apply due to the following reason: In the cascade correlation algorithm, the number of training vectors are chosen *a priori* and the neural network grows and adapts weights simultaneously to minimize output error. However, there is no *a priori* method to determine the number of training vectors if the architecture of the network is unknown. Thus a trial and error procedure is adopted to determine the optimal number of hidden layer nodes. The number of hidden layer nodes is gradually increased from 2 to 20 and for each architecture a complete training and testing cycle is completed. The learning capabilities measured by forecast accuracy of the network is estimated for each hidden layer size. This procedure is performed on 20 randomly selected cases. The variation of forecast accuracy with hidden layer nodes is shown in Figure [8]. Learning capabilities are best if the number of hidden layer nodes is 12. Thus the architecture of the network is chosen as ((6 x 12 x 1) + (6 x 1 x 1)).

Neural networks are trained by the error backpropagation algorithm using training data sets processed by the above detailed method. During the training process, the training sets are corrupted dynamically by noise with zero mean and 10% peak value. This introduces subtle variations of the training data sets leading to slightly robust function approximating neural networks. The training data sets are presented at random to the

network. The network is trained till the maximum error on all training set vectors is less than 5%. For average load forecasting, learning takes about 50000 iterations and about 10 seconds of NeXT computer time (50MHz clock frequency). Once the network has converged, the test vector is applied and the forecast quantity appears at the output of the network. This procedure is applied to forecast average load for all weekdays of January, February and March 1992.

The performance of the neural network for daily average load forecasts is as shown in Figures [9]-[11]. As seen from the graphs, the connectionist model is able to track the changes in load due to weather and work patterns quite accurately. This proves the applicability, by actual simulation, of the proposed methods of training database creation, training set selection, data scaling and normalization schemes for local function approximation to bus load forecasting. The algorithm is sensitive to variations in weather and other variables. The forecasts are performed for weekdays only. Simulations for weekends is not possible with the few weekend datasets available in the database. Forecasts for holidays occurring on weekdays were performed, without supplying information about the holiday status to check the response of the network to the input variables. The forecasts are characterized by large positive errors, as expected. This phenomena is clearly seen for the forecasts of Spring Break in March 1992. (Figures 11 and 12). The average error, as defined as the sum of modulii of individual errors divided by the number of observations is computed for each month along with standard deviations of forecast error are computed. The average daily forecast errors for the working weekdays of the three months are 2.48%, 2.34% and 1.24%. The forecasts are steady and do not fluctuate drastically as the variances are computed to be 2.61, 3.85 and 1.75 respectively. Thus it can be inferred that neural networks trained by the above procedure are accurate in forecasting daily average "integrated" load.

2.5.2 Daily Peak Load Forecasting

The forecast quantity of interest in this case is the daily peak load. With reference to the Table [1] of correlation between possible predictor variables and maximum load, there is very little correlation. The only quantity correlated with the peak load is the forecast average load. Among the other variables, the same six predictor variables have the largest correlation magnitudes, very small though. As a first step, the same six predictor variables used for average load forecast were used for predicting the peak load. The results are tabulated as shown in Figures [13]-[14] for the months of January and February 1992. The errors are large as shown in Figure [15] and no meaningful forecasts can be obtained by this one-step procedure. Alternatively, if a robust average load forecast is available, using a two-step procedure, the daily peak load can be forecast accurately even with a *linear model* for this part.

As seen by the results of Piche [15], multilayer feedforward neural networks are not statistically robust to input and weight errors. Small noise in measurements and in the databases have the potential effect of producing bad forecasts. Though the selection procedure ensures, outliers or bad data in independent variables do not enter the training data set, outliers in the dependent variable (the so-called y-axis outliers) can certainly cause bad forecasts. Thus a robust and accurate average load forecast is required as a first step on which the peak load forecasts can be based upon. Statistical decision trees, presented in Chapter 3 can produce robust and accurate forecasts. Their utility in peak load forecasting is examined in great detail in the next chapter.

2.6 Knowledge Extraction Using Neural Networks

Neural networks are data intensive machine learning techniques that learn internal system representations through the above described error backpropagation algorithm. The system knowledge is encoded as connection weights between neurons. To derive a human intelligible model of the system being modeled, it is essential to translate the weights to high level syntactic sentences. Neural network learning has some significant advantages over the conventional decision tree methods of inductive learning [22]. Learning and recall performance of the EBPN algorithm was found to be statistically much more significant than ID3, a symbolic machine learning algorithm. This leads us to contemplate application of the EBPN for acquisition of underlying knowledge from large databases. This is based on the supposition that knowledge acquired by use of this paradigm may be superior to the knowledge acquired by conventional symbolic machine learning techniques.

Connectionist models create high level attributes automatically and base reasoning on these attributes rather than raw measurements. This leads to compact encoding of knowledge which is in contrast with the other methodologies that reason from raw attributes leading to large and complex representations of acquired knowledge.

There has been little activity in this area and consequently very little literature is available for application to complex problems such as the electric load forecasting problem. The first and perhaps only comprehensive work till date has been the work of S.I. Gallant [23]. The essential philosophy of the neural network based knowledge acquisition is as follows: Given a data-rich situation, with examples available to describe the input-output relationships of a system, it is possible to build a neural network that embodies the underlying knowledge and serves as the knowledge base to an expert system. The "connectionist expert system" has two main components: A feedforward

neural network is constructed from inputs, intermediate high-level features, and outputs. The weights are determined using the training set by "Pocket Algorithm" which is a variant of the Perceptron Learning Algorithm [23]. The weights of this network embody the domain knowledge. To translate this knowledge to "if-then" rules, the second component, "Matrix Controlled Inference Engine" (MACIE) runs input data through the network to obtain the most plausible answer. Application of this technique to electric load forecasting is fraught with numerous difficulties; The most important problem is the availability of "intermediate attributes" that explain the load demand using the input variables. In the electric load forecasting, intermediate attributes to explain the load demand by raw data such as temperature, humidity, wind conditions and global insolation are not available. The architecture of the neural network is dependent on the availability of these intermediate variables. Construction of neural network without intermediate variables leads to situations where the output cannot be explained directly in terms of the input which are not linearly separable. The algorithm can be applied only for symbolic input and output data. This is far from true in the electric load forecasting problem where inputs are a combination of numeric and symbolic data and the output is numeric.

Variants of the above technique [23] are valid for Boolean input and output classes. Thus, it can be inferred that neural networks for knowledge acquisition are at a fledgling stage and these techniques are not mature enough for real world applications. Symbolic machine learning techniques, which have been the subject of intense research and improvements over the past two decades are at a stage where they can be applied to complex real world problems. In the next chapter, techniques from symbolic machine learning area are examined for their ability to extract knowledge from databases and for forecasting on new cases based on knowledge extracted.

2.7 Conclusion

In this chapter, connectionist models have been applied to model complex input-output relationships of load dependency on independent measurable variables and has been used for forecasting. Design and implementation issues of neural networks such as training databases creation, training set selection, data scaling and normalization and architecture have been presented, drawing upon local function approximation and nearest neighbor norm schemes. These techniques can be applied to any nonlinear function approximation and forecasting problem. A complete algorithm for bus load forecast has been presented and applied to forecast accurately average integrated bus load demands. The mean errors are 2.48%, 2.34% and 1.24% for three months. Neural network models produce accurate and quite robust daily average load forecasts. They perform poorly in forecast of daily peak load. Powerful connectionist network algorithms that have capabilities of extracting knowledge from large databases with noisy, unstructured data that contain both symbolic and numeric variables characteristic of real world problems are not presently available in literature. Development of robust training schemes and knowledge acquisition algorithms would enhance the applicability of numerically accurate neural networks to real world nonlinear modeling and forecasting applications.

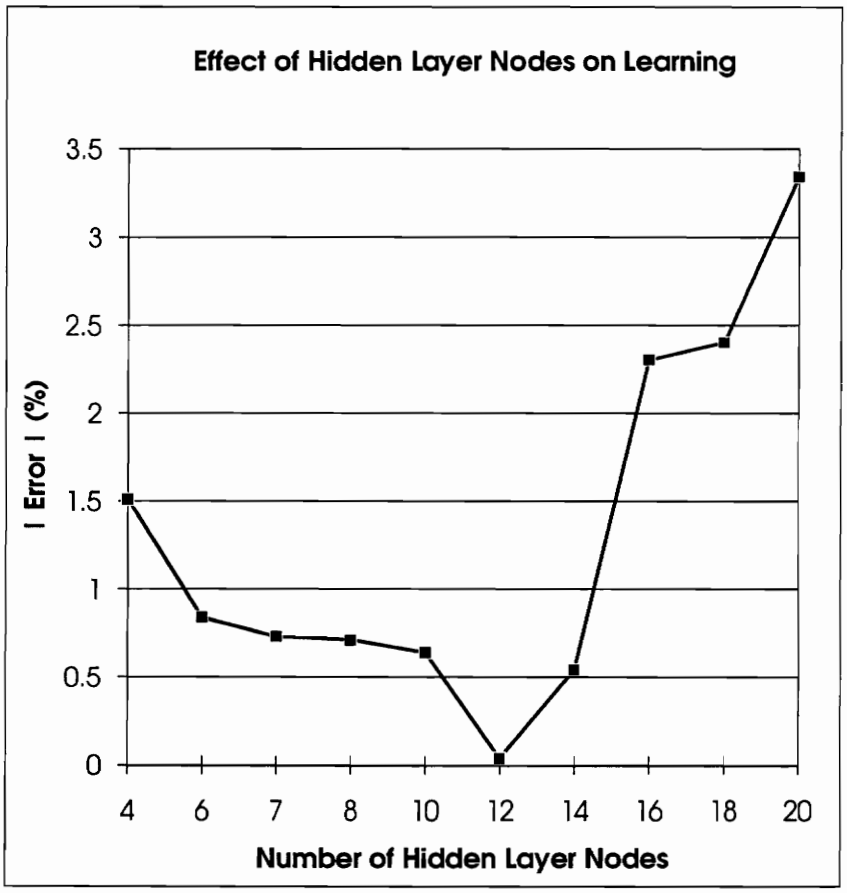

Figure 8: Variation of forecast error with number of hidden layer nodes

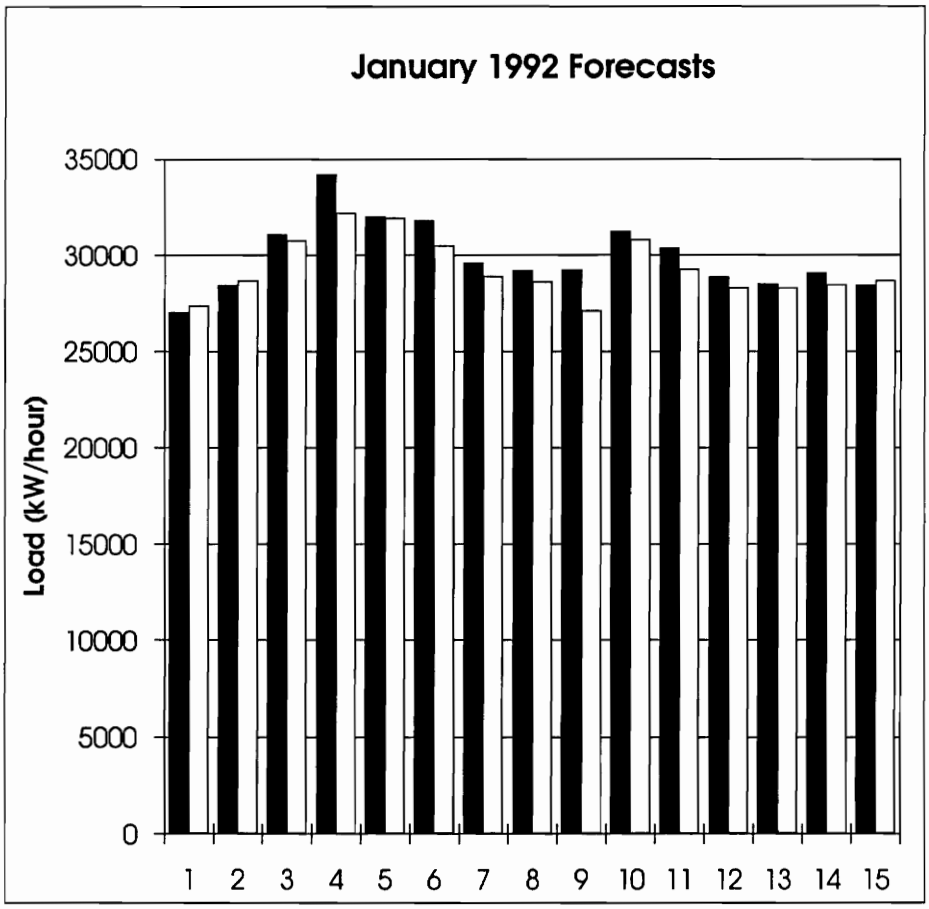

Figure 9: Neural network forecasts of daily average "integrated" load, Jan. 1992;

Weekdays only; Y-axis is load in kW

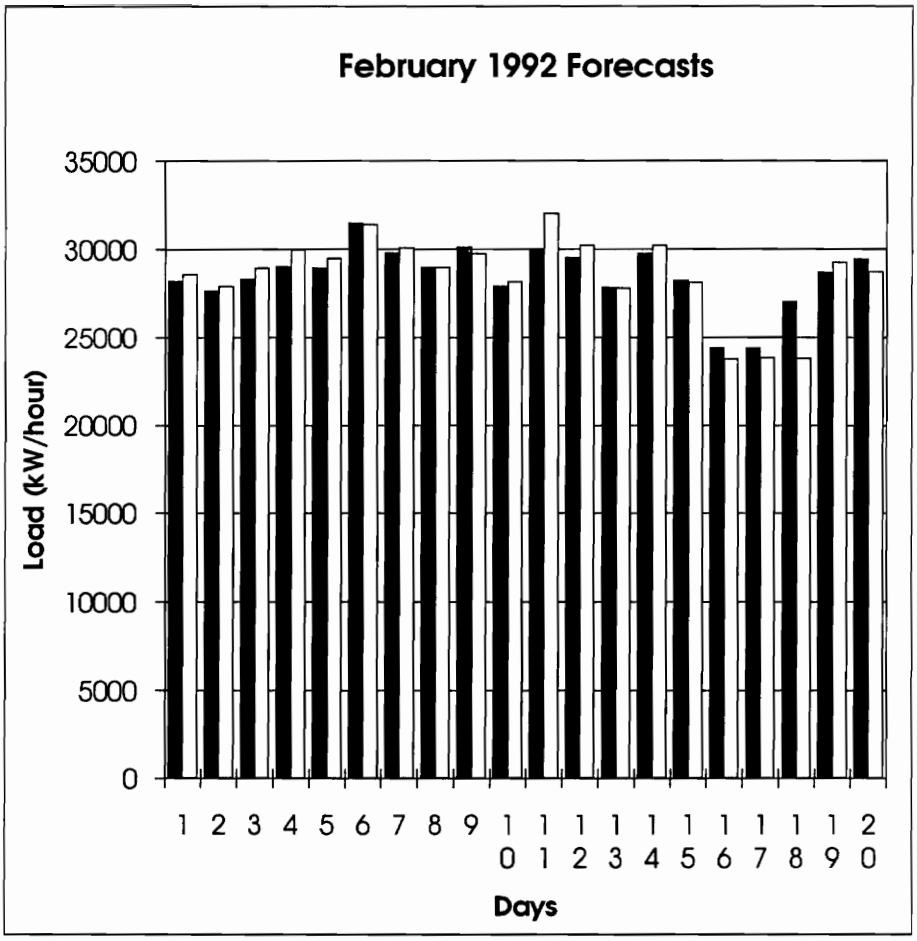

Figure 10: Neural network forecasts of daily average "integrated" load, Feb. 1992;

Weekdays only; Y-axis is load in kW

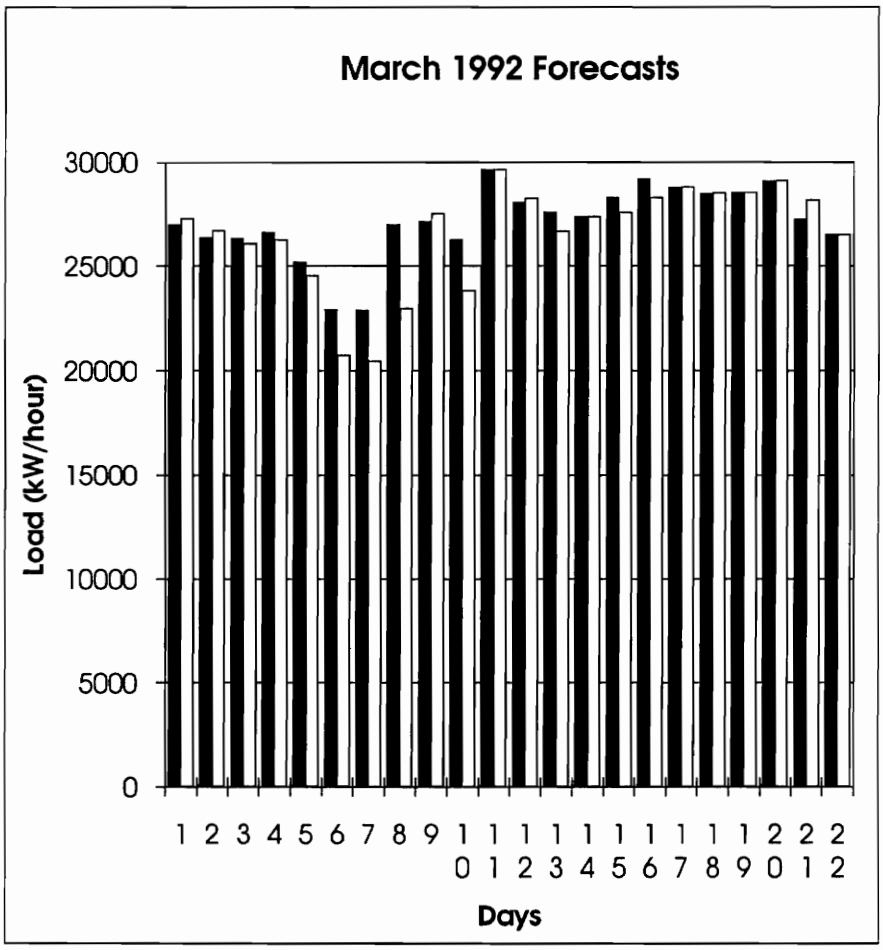

Figure 11: Neural network forecasts of daily average "integrated" load, Mar. 1992;

Weekdays only ; Y-axis is load in kW

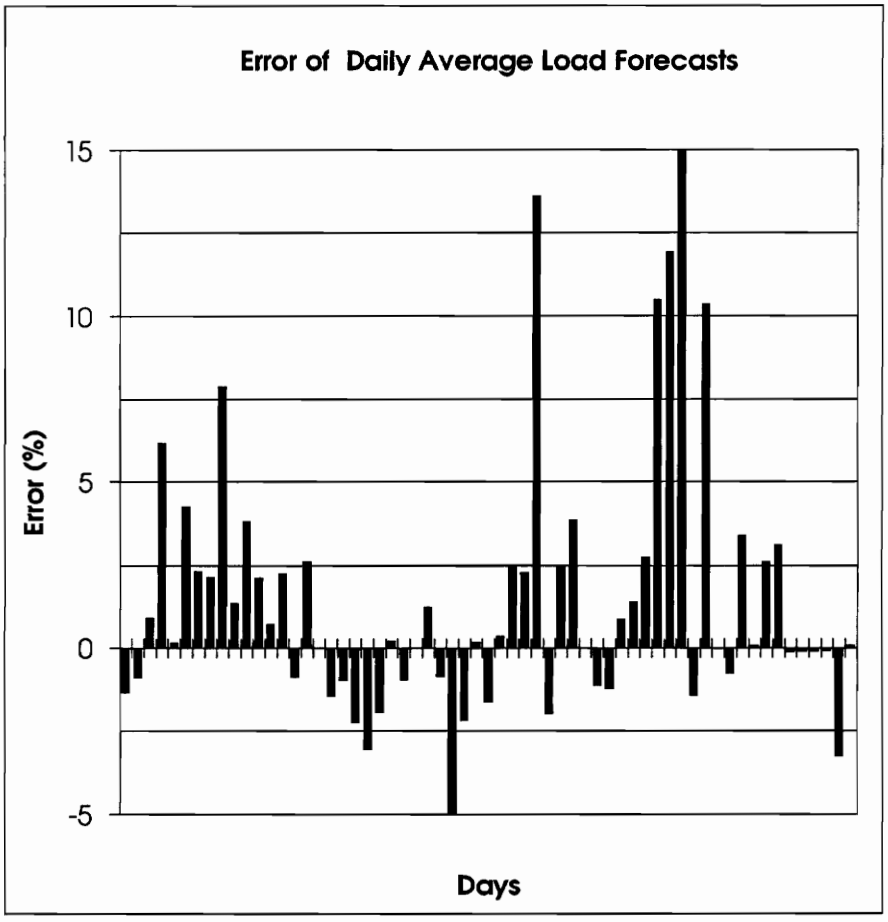

Figure 12: Forecast error of average integrated load: January, February and March 1992;

Holidays occurring on weekdays typically have a large positive error.

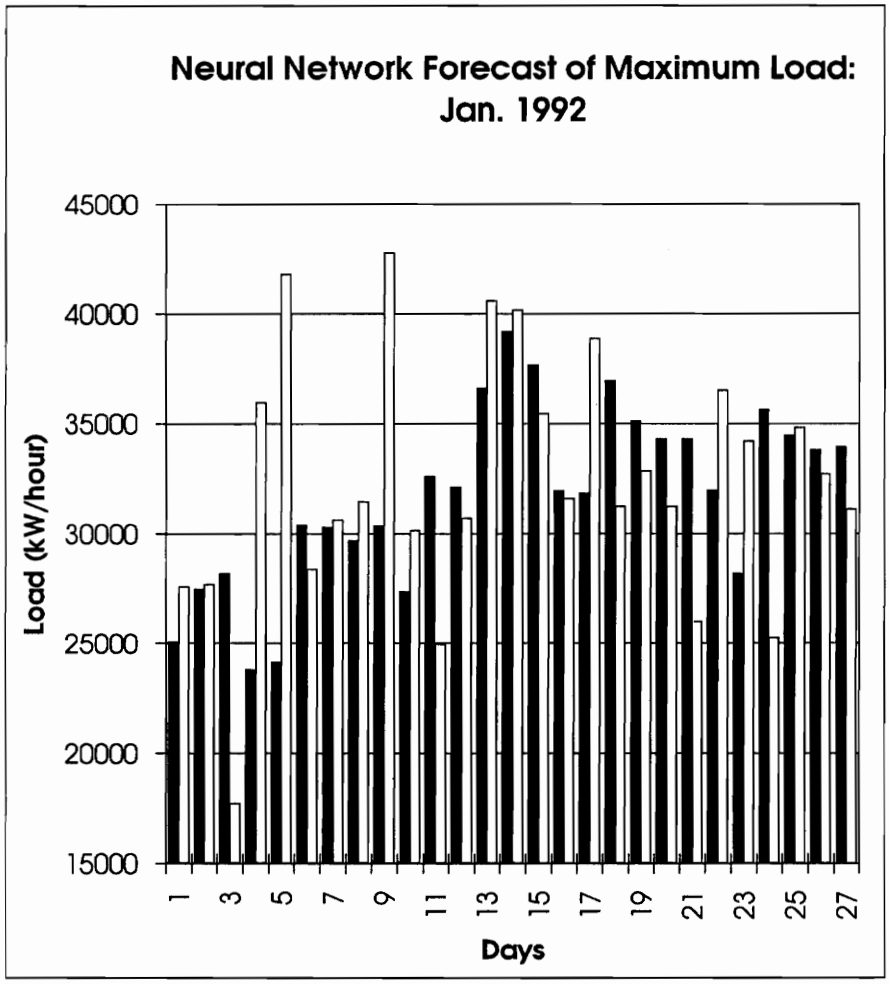

Figure 13: Neural network forecasts of peak load, January 1992;

Weekdays only; Y-axis is load in kW

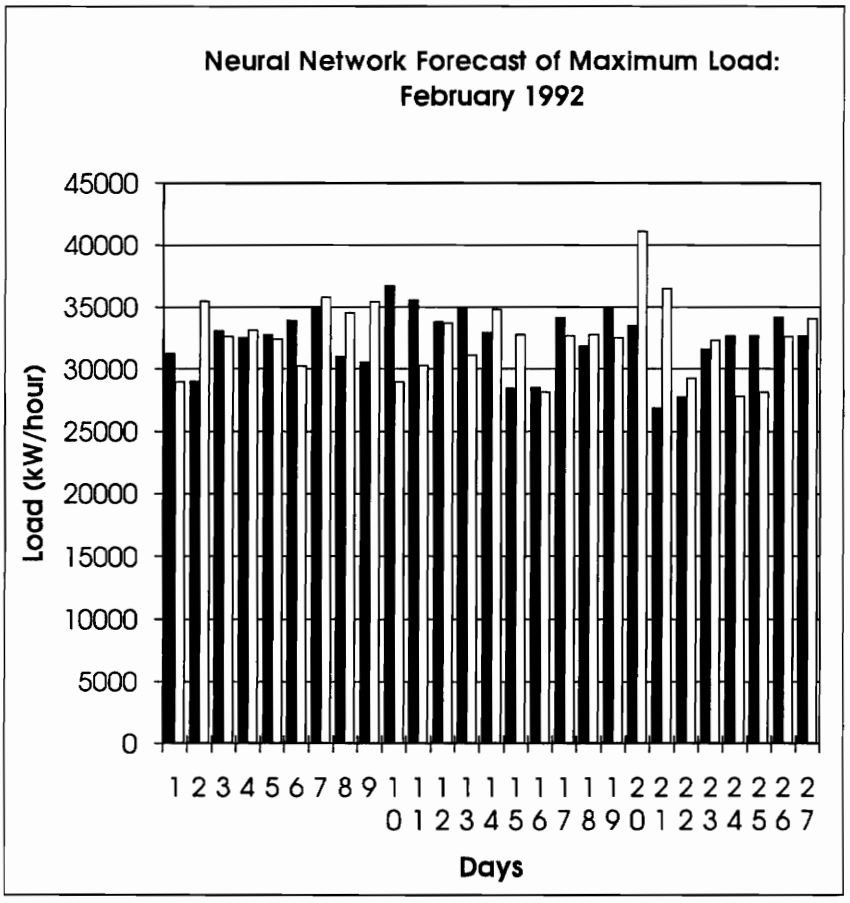

Figure 14: Neural network forecasts of daily peak load, February 1992;

Weekdays only; Y-axis is load in kW

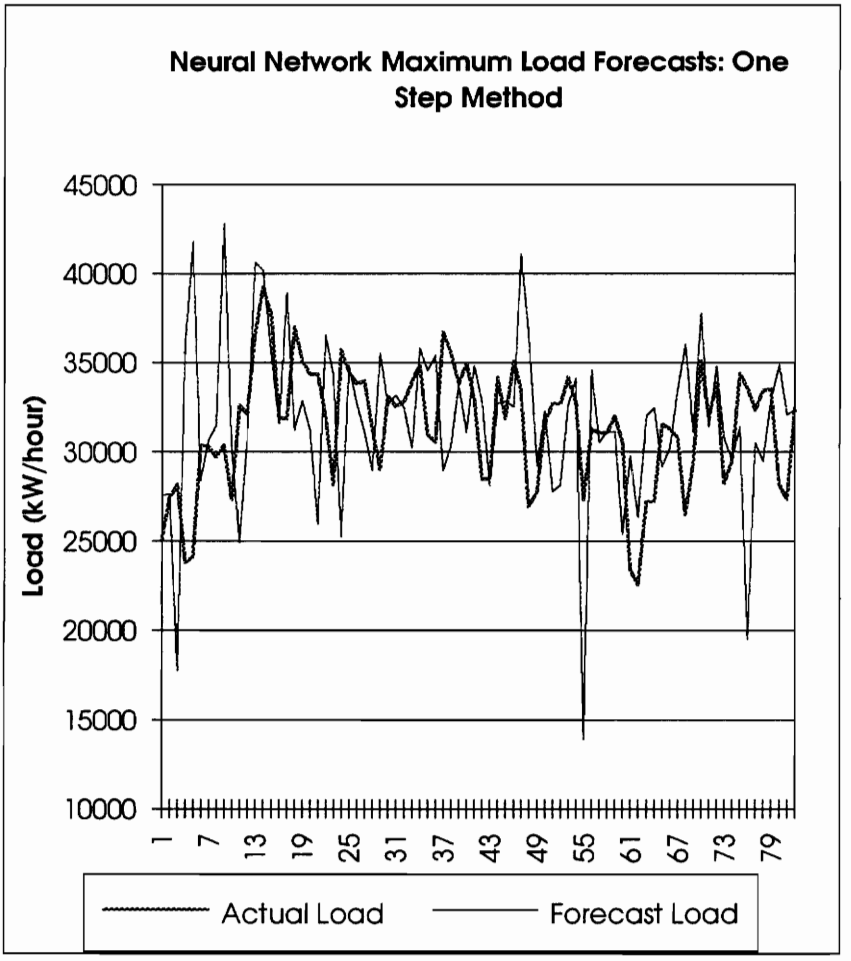

Figure 15: Forecast error of daily peak load: January, February and March 1992

CHAPTER III

SYMBOLIC MACHINE LEARNING TECHNIQUES FOR SHORT-TERM ELECTRIC LOAD FORECASTING

Symbolic machine learning techniques and algorithms extract general concept descriptions of a process or system from a given set of examples and express it in human intelligible graphic or natural language form. In this chapter, symbolic machine learning mechanisms are applied to extract knowledge from electric load consumption and weather databases. Inductive learning techniques are apt for this task. Inductive learning algorithms applicable to electric load forecasting are analyzed and the statistical decision tree algorithm is adapted to extract load consumption patterns. The extracted patterns are expressed in human intelligible high level language as graphic decision trees and equivalent "if-then" rules. Implementation details of the statistical decision algorithm are discussed and simulations are performed to extract the knowledge. Using this model, unseen cases are forecast. This algorithm is capable of forecasting holiday and weekend loads too. The proposed algorithm is robust enough to handle raw, unprocessed databases which contain missing data. An approach integrating the two methodologies for peak load forecasts is presented.

3.1 Hierarchy and classification of machine learning systems

Research to understand the process of learning and theories to formalize the process have been the subject of intense work since the turn of this century in diverse disciplines such as cognitive science, human psychology and computer science. The amount of inference performed by the learning agent is an indication of the autonomous

learning capability of the system. Based on the degree of inference to be performed, machine learning systems are classified into the following categories:

a) **Rote learning systems**: This corresponds to the lowest level of human and machine learning. It corresponds to direct implantation of knowledge directly in the learning system by a teacher. No inference or other transformation of knowledge is required by the learner for utilization. The performance of rote-learning systems is very limited as circumstances not memorized by the system cannot be solved.

b) **Systems learning by instruction**: Knowledge acquired from a teaching agent is transformed by the learning system into an internal representation applicable to the problem. The primary responsibility of structuring and representing the knowledge remains with the teaching agent. The inference capabilities of the system are limited to the transformation of knowledge into a readily usable form. The learner's role in this mode of learning may be considered as syntactic reformulation of the knowledge provided by the primary source.

c) **Systems learning by analogy**: This mode of learning combines inductive and deductive learning. Inductive learning discovers a common substructure between the problem domain and an analogous structure of knowledge in the knowledge base. The next step, deductive inference, maps the knowledge from the chosen domain in the knowledge base to the problem domain.

d) **Systems learning from examples**: This mode requires significant degree of inference. Learning from examples by *induction* is a powerful technique in this category. This method involves concept acquisition by inductive learning from a set of examples provided by the teacher. The learning system formulates rules that best fits the examples. Rules, the acquired concepts should be general enough to explain all of the examples provided.

Systems with inductive learning capability are useful for solving practical real world problems. Practical inductive learning systems are useful for extracting knowledge from examples in databases which are unstructured, contain errors and omissions.

e) **Learning form observation and discovery**: This mode of learning is commonly called unsupervised learning and requires the learner to discover concepts autonomously and formulate theories explaining the observations. The external teacher provides no aid in this process.

From the above hierarchical classification, it can be clearly be seen that inductive learning techniques are the apt for data-rich, example intensive problems such as electric load forecasting. Inductive learning techniques, as shall be seen below, have a sound mathematical foundation and can be applied to many real world problems.

3.2 Inductive Learning Methods

Given a chaotic, unstructured agglomeration of examples drawn from a process, inductive learning methods detect the underlying structure and relationship using principles of induction. The learning agent is an inductive learning algorithm (ILA). A set of examples or instances called training set is the input to the ILA. Each instance defines a vector of values of the independent or predictor variable and the corresponding value of the response variable. The predictor and response variables may be numeric valued continuous variables or symbolic unordered categorical variables. The output of the ILA is a parsimonious set of rules or a graphic decision tree which explains the response variable in terms of the predictor variables. As compared to connectionist models, this is a more general framework of learning, that permits inclusion of symbolic variables in the machine learning and knowledge representation scheme.

The main components required for machine learning of underlying structure of a process or system are: a) the training set, and b) learning algorithm.

a) **The training set**: A *training set* is a collection of *objects* whose classes are known. *Objects* are descriptions of a *response variable*, through a set of *independent variables*. It contains examples or vectors of measurement made on the system along with the value of the output variable. For a set of values of the independent variables, the response variable would assume a certain value or belongs to a certain *category* or *class*.

Essentially two kinds of variables are encountered in the measurement vector. Numerical or ordered variables are measured values of the attribute that take on real numbers. Categorical variables take on values from a finite set of symbols and do not have any natural ordering. In electric load forecasting applications the training sets used, contains objects with the following attributes:

- the *global insolations*, with *continuous numeric* values
- the *temperature*, with *continuous numeric* values. Variants include *maximum*, *minimum* and *average* temperature values that contain key information.
- the *humidity*, with *continuous numeric* values
- the *type-of-day*, whose values are *holiday*, *weekend and working* day
- the *wind* conditions, with *continuous numeric* values.
- the *load*, which is the response variable, with *continuous numeric* values.
- *previous load* with *continuous numeric* values as *predictor* variables.

b) **Learning algorithm**: A learning task accomplished through a learning algorithm is the problem of finding a general set of rules that work well on the objects in a given training set. Techniques for inductive learning of key decision making information from data have

originated in diverse theoretical backgrounds depending on the nature of available data and complexity of underlying theory. Real world problems contain examples in databases. These databases contain examples which are noisy, contain errors and omissions. Usually no single theory or rule is sufficient to cover all the cases; reversal of trends, region wise context sensitive relationships may be encountered. The learning algorithm should have sufficient dexterity and capability to handle these features. All inductive machine learning algorithms have the following components and structure:

Part 1: An input training data set S;

Part 2: A test "T", with possible outcomes O_1, O_2, O_3,...,O_n. Each sample is subjected to T, resulting in a partition of the input data sets into S_1, S_2, S_3,..,S_m. T now becomes the new root of the decision tree and for each outcome O_i, a subsidiary decision tree is built by invoking the same procedure recursively on the set S_i. The decision tree consists of a *test* in each sub-node, with a set of mutually exclusive possible outcomes. Each outcome is associated with a subsidiary decision tree or *leaf*, which is a terminal node.

Part 3: The above splitting procedure is performed recursively until all possible splits are insignificant. This condition has to be detected through appropriate tests.

Part 4: The completed learning procedure is a decision tree. A *decision tree* is a recursive structure for expressing learning and classification rules. Shown in Figure 18 is a binary decision tree. The word binary stems from the fact the branches are binary. Each node, represented as a circle is a test. Data are partitioned into subsets that are further tested or are terminal classes, represented by square boxes. The decision tree can be readily converted into a set of rules. Finally, the decision tree is tested for its ability to classify new unseen cases sampled from the population.

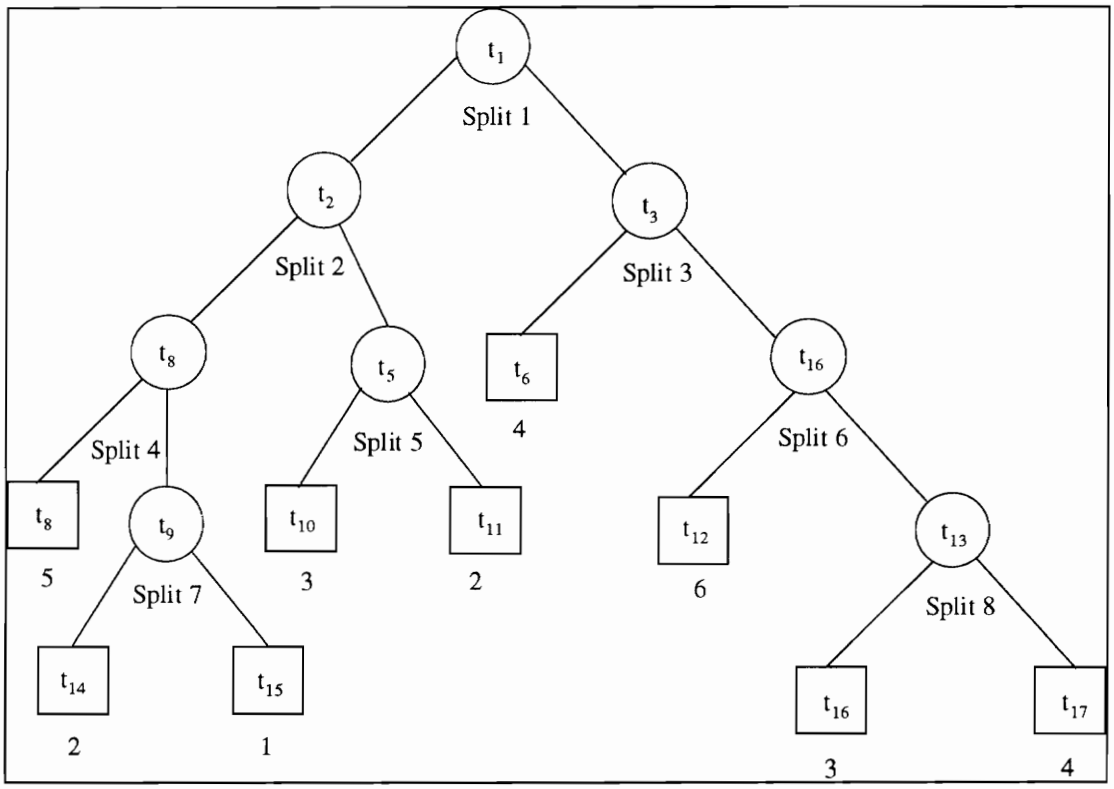

Figure 18: A general binary decision tree architecture

The underlying assumption of all inductive learning methods is that, if patterns that account for class membership of known objects is found, these patterns will predict the class of new, unseen objects.

3.3 Tree construction methods

The entire construction of a tree, revolves around the following three elements;

a. The selection of the splits.

b. The decisions when to declare a node terminal or to continue splitting it.

c. The assignment of each terminal node to a class.

These three elements, fundamental to construction of a decision tree are examined in greater detail. These procedures are applied repetitively and judiciously to produce a compact and accurate tree. Selection of splits is an important and crucial task. The other two elements are usually part of the selection of split process.

Selection of splits: Partitioning of given data sets into smaller subsets involves application of a test. Tests, as mentioned earlier, have evolved from diverse backgrounds and are classified into three major categories:

- Tests based on information entropy
- Tests based on expected error
- Tests based on statistical significance

3.3.1 Tests based on information entropy

Artificial Intelligence (AI) techniques employ a top-down induction of decision tree technique with the objective of maximizing the information gain by production of

splits at every node. ID3 algorithm and its descendants such as C4 belong to this category. At each step, a new node is added to the decision tree by partitioning the training examples based on their value along a single, most informative attribute. The attribute chosen is one which minimizes the following entropy function:

$$E(A) = -\sum_{i=1}^{V} \frac{S_i}{S} \sum_{j=1}^{N} \frac{k_{ji}}{S_i} \log_2 \frac{k_{ji}}{S_i}$$

where E(A) is the entropy associated with each node

V is the number of values of attribute A

K_{ji} is the number of examples in the jth category with the ith value for the dependent variable for the attribute A

S is the total number of examples

S_i is the number of examples with i^{th} value for the dependent variable for the attribute A

N is the number of categories of the dependent variable

The stop splitting criterion is a χ^2 test. The number of splits produced at each node by ID3 equals the number of categories of the predictor variable. This causes production of redundant rules necessitating rule simplification as a post processing feature. In this algorithm there is no systematic method of grouping continuous valued variables into grouped categories. Also, this algorithm favors predictor variables with many categories and categories of predictor variables that combine in many ways. Thus, this algorithm, is not readily applicable to short-term load forecasting.

3.3.2 Tests based on expected error

The basic principle of this methodology, is to minimize the expected error at each node when the tests are replaced by probabilistic classifiers. The testing criteria at each node is limited to *value of a single variable* and consequently only binary splits are produced at each node. The Classification and Regression Trees (CART) algorithm belongs to this category. The predictor variable used in splitting is determined by the following maximization criterion: Maximize the Gini function (GF) given by:

$$GF = 1 - \sum_{k=1}^{j} p_k{}^2$$

where p_k is the probability of class k.

Maximization of this criterion is equivalent to choice of variables that reduce uncertainty the greatest or minimize the expected error. In this method, assignment of terminal classes is a two-step process. First, pruning of the fully grown tree is performed. In the pruning step, weak links or branches, are eliminated and as a next step, remaining terminal nodes are assigned classes with minimum resubstitution error. Though work done as part of this methodology help placed the decision tree methodology in a firm footing, there arises the basic question of why splits should be only binary. Again this methodology, favors predictor variables those categories combine in many ways. Investigation shows that the accuracy of results produced by this methodology is slightly inferior to those obtained by the backpropagation algorithm for hourly electric load forecasting [17].

3.3.3 Tests based on statistical significance testing:

Tests of this category generate splits based on predictor variables that have the highest level of significance with respect to the response variable. The significance tests is a $\chi 2$ test. Termination of split and assignment of terminal classes occur when all potential

splits are not significant with respect to the predictor variables. This condition is detected by application of a boundary condition that the expected error is within a certain threshold. Tests based on statistical significance will be the subject of further analysis and application due to the following reasons:

- Splits based on this test are not restricted to binary partitions. In fact the best k-way partitions are determined.
- Stopping rules are an integral part of the growing procedure and no post-processing such as pruning is required
- Both numeric and categorical variables are handled with ease
- A built-in test of significance is present
- An automatic significance level threshold adjustment procedure to compensate for reduced population samples is present.
- Predictor variables with many categories or categories that combine in many ways are not favored, as a dynamic grouping procedure is applied to merge and split categories during the growth process.

The statistical significance testing framework has been subject to far fewer applications compared to its cousins, the information entropy and CART methodologies. ID3 has the above mentioned disadvantages and is not pursued any further. Studies [17] have shown that CART performs poorly on the short-term electric load forecasting problem. Availability of PC software, KnowledgeSeeker [33] for implementing the decision tree framework was also an important factor in the use of this paradigm.

3.4 The Statistical Decision Tree algorithm

We wish to determine whether two variables, the response and predictor variables are related (dependent) and if so, we wish to predict one variable based on knowledge of the other. This is equivalent to construction of a parsimonious set of rules describing the input-output relationship from a training set. The statistical decision tree methodology uses a statistical test of significance, the $\chi 2$ test, to determine the significance and the most significant predictor at every step. Tests of dependence for bivariate count data are performed on observed data arranged in contingency tables. The name *contingency table* is derived due to the fact that we are interested in ratification of a hypothesis between the two variables and identification of contingency between the predictor and response variables.

Let the dependent variable have $d \geq 2$ categories and a particular predictor under analysis $c \geq 2$ categories. A subproblem in the construction of the decision tree is to reduce the given $c \times d$ *contingency table* to the most significant $j \times d$ table by combining categories of the predictor variable. Computation of $T_j^{(i)}$, the χ^2 statistic for the ith method of forming a $j \times d$ table ($j = 2,3,..,c$), and choice of $T_j^{(*)}$ (the χ^2 statistic for the best $j \times d$ table) determines the best possible grouping of the particular predictor variable. This procedure is repeated for all predictor variables and finally the best partition is chosen based on the best $T_j^{(*)}$. Before dwelving any further into the statistical significance testing framework some basic elements in the algorithm are discussed.

3.4.1 Statistical Significance Testing

Given groups of a predictor variable cross-tabulated with groups of the response variable, is the level of significance in predictor-response association substantial to suggest

any meaningful relationships? This question, fundamental in determination of best splits, is answered using the χ^2 test, a test of association. The algorithm of χ^2 testing for significance is shown in Figure 16.

Null hypothesis: $H_0 \Rightarrow$ The two variables are independent. (no association exists).

Alternative hypothesis: $H_1 \Rightarrow$ The two variables are dependent (a significant relationship exists).

Test Statistic: $\chi 2 = \sum_{i,j} \frac{(n_{ij} - E_{ij})^2}{E_{ij}}$, where n_{ij} is the observed frequency and E_{ij} the expected frequency.

Rejection Region: Reject H_0 if $\chi 2$ exceeds the tabulated value of chi-square for a given accepted type 1 error rate α and degrees of freedom df = (r-1)(c-1), where r is the number of rows in the contingency table and c is the number of columns in the contingency table.

Figure 16: Algorithm of $\chi 2$ significance testing

Consider the following example that illustrates the principle of chi-square test of association applied to a 2×2 contingency table.

Table 2: Observed Frequencies

	Low Demand	High Demand	Totals
< 50 F	75	103	178
≥ 50 F	71	196	267
Totals	146	299	445

Table 3: Expected Frequencies

	Low Demand	High Demand	Totals
< 50F	58.4	119.6	178
≥ 50F	87.6	179.4	267
Totals	146	299	445

Given the observed frequencies, the next step is to compute the expected frequencies [Table 3]. Since the exact proportion of categories in the whole population is never known, the observed frequencies are assumed to be representative of the population. The best estimate of proportion of low temperature demand days is (146/445)*(178/445). The expected frequency of low temperature low demand days is 146*(178/445) = 58.4. Similarly other expected frequencies are computed.

The $\chi 2$ (uncorrected) = $(75-58.4)^2/58.4+(103-119.6)^2/119.6+(71-87.6)^2/87.6+(197-179.4)^2/179.4= 11.71$

The $\chi 2$ (corrected) = $(16.6-0.5)^2/58.4 + (16.6-0.5)^2/119.6 + (16.6-0.5)^2/87.6 + (16.6-0.5)^2/179.4 = 11.01$

The number of degrees of freedom = υ = # of classes - # on independent relationships = 1

Looking up the χ^2 table with $\upsilon = 1$, the critical value for $\alpha = 0.001$ is 10.83. Thus the null hypothesis can be rejected with a very high degree of confidence establishing the validity of a relationship between temperature and load. This concept extended to an arbitrary j × d contingency table, is used in the calculation of splits in the growth of statistical decision trees.

3.4.2 Significance Level Adjustments

Significance test level α is the accepted Type 1 error rate. The Type 1 error rate is the probability of rejecting the null hypothesis when it is true. In decision tree methodology, significance tests are applied to a fraction of the samples of the population and not the initial samples. Corrections are necessary while applying prespecified significance levels to the decision tree methodology.

a. An adjustment factor for the best variable

b. An adjustment factor for the best groupings of the best variable

a. Adjustment factor for the best variable: The type 1 error rate α has to corrected for the fact that the most significant variable is considered for splitting a node, instead of any variable.

The Bonferroni adjustment factor is a constant used to compensate for the fact that a pure subset of the population, rather than the whole population is used in the analysis. Since all the predictor variables are considered at each split, independently of each other, the Bonferroni adjustment factor is just the number of predictor variables. However, the chance of resplitting on a previously split variable is very small due to the fact that at each step, best *k*-way, not binary splits are done. Thus it would be sufficient to use a correction factor of total number of predictor variables N_V, less the number of splits $N_n(t)$ used to reach current node, as the Bonferroni correction factor is

$$N_{BV}(t) = N_v - N_n(t)$$

b. Adjustment factor for the best groupings of the best predictor variable: Tree growing by splitting and stop splitting of growth are important decisions based on significance test levels. A significance level of say, $\alpha = 0.05$ is specified as the maximum

permissible type 1 error. In decision tree methodology, the best possible grouping, with highest significance levels are subjected to significant tests and not the initial grouping. Thus the chance of finding a split significant, when infact no relationship exists is greatly increased. Since α specified is for all samples of the population that are sampled, it is necessary to correct the specified α when applying the test on best groupings. Theoretically, if the final grouping of categories is selected to be the best possible of all possible combinations of the c initial categories into any number of groups, the second Bonferroni adjustment factor is the total number of ways of distributing the c categories into k groups where $2 \leq k \leq c$. Thus, by the Bonferroni inequality, the significance α_O at which the best grouping should be test at is

$$\alpha_o = \frac{\alpha}{\sum_{k=2}^{c} N(c,k)}$$

where N(c, k) is the number of ways of grouping c categories into k groups.

Thus, the corrected significance level used is:

$$\alpha_o = \frac{\alpha}{N_{BV}(t) N_{BC}(t)}$$

The complete statistical decision tree algorithm is as follows: Let k be the total number of predictor variables, d the number of categories of the response variable, i the predictor variable counter, j the final number of merged categories. Set i = 0.

Step 1: Given the response variable, select a predictor variable.

Step 1.1: For continuous valued predictor variables, sort and arrange the predictor variables in an ascending order. Divide the predictor variable into N initial categories. N is usually set to within 10 to prevent overconservative results. For categorical variables, prespecified categories are readily available.

Step 1.2: Construct a $N \times d$ contingency table. For continuous valued response variables, this is a $N \times N$ contingency table.

Step 1.3: Find pairs of categories of the predictor whose $2 \times d$ subtable is least significantly different as determined by a $\chi 2$ test of association.

Step 1.4: If this significance does not reach a critical value α_0, merge the two categories.

Step 1.5: If the significance is $< \alpha_0$, set $T_j^{(i)}$ to the highest significance level of any category obtained by use of this predictor variable.

Step 2: If $i < k$, choose next predictor variable ($i = i + 1$) and go to step 1.

Step 3: Choose as "Most Significant Predictor" (M.S.P), the predictor variable with largest $T_j^{(i)}$. Set $T_j^{(*)} = T_j^{(i)}$.

Step 4: If $T_j^{(*)} \geq \alpha_0$, construct branches of tree based on M.S.P.

Step 5: If $T_j^{(*)} < \alpha_0$, terminate growth for this node. Assign terminal classes based on available M.S.P. For continuous valued response variables, set mean expected value as average of predictor values in the category and standard deviation as variance of predictor values in category.

Step 6: Go to next node and start from step 1.

Step 7: If no nodes are remaining, terminate growth process.

3.5 Case Study: Daily Load Forecast by Decision Tree Methodology

The use of decision tree methodology for short-term electric load forecasting is based on the following objectives:

a. To systematically acquire knowledge from data about the dynamics of the process. Here, it is desired to understand in a quantitative manner the forces driving the load demand.

b. Based on the knowledge extracted, construct a model to predict future, unknown values of load using values of measurable variables such as temperature, wind, insolation etc.

The available measurement variables were initially screened using correlation analysis so that a concise, relevant set of predictor variables could be chosen. The predictor variables of interest were the average daily load (kW) and maximum daily load (kW). The measurement variables were subject to a correlation analysis and the results are tabulated in Table [1].

For a fair performance comparison between connectionist models and symbolic machine learning techniques, the same predictor variables were used. To obtain compact and relevant decision trees, the training sets are chosen according to the following procedure:

Step 1: The month whose average daily load and maximum daily load is to be forecast is determined.

Step 2: The time window comprising of one month before and after the forecast month is chosen.

Step 3: Time window segments for the past 3 years are extracted from the load consumption database.

Step 4: Month wise "Load growth factors" (LGF) are computed by the following formula:

$$LGF = \frac{\text{Total energy consumption of (Month M: Year i)}}{\text{Total energy consumption of (Month M: Year (i-1))}} - 1$$

Step 5: Historical load data of the months comprising the time window segments are updated using load growth factors.

Step 6: The time window segments are concatenated into a single training database.

Step 7: Parameter α, the accepted error rate of Type 1 error for growing of the decision tree is chosen. To obtain conservative results, we specify $\alpha = 0.01$, i.e. an acceptable type 1 error rate of 1%.

Applying the statistical decision tree algorithm of Section 3.4, the decision tree of Figure 35 is produced. The decision tree is readily converted to an equivalent set of "IF-THEN" rules which is shown in Figure 36.

3.5.1 Average Load Forecasts

The average load demand is of practical use to the power system planning personnel as it helps in planning and allocation of the generation mix. The base load is usually met by nuclear and coal fired plants. The maximum load demand is of greater economic value to the power system operators as it indicates the amount of power that has to be imported or met by costly peak power capping units. Thus, forecasts of both average and maximum load are of importance to the power system operations personnel. Three rules explain all possible weekend scenarios. The maximum standard deviation associated with a holiday forecast is 5.27%.

RULE #1 IF
Type of Day = hol & Max. Temp. = [25.9,36.1)
THEN
expected Avg. Load = 24998; std = 1672

RULE #2 IF
Type of Day = hol & Max. Temp. = [36.1,42.5)
THEN
expected Avg. Load = 28552; std = 2248.2

RULE #3 IF
Type of Day = hol & Max. Temp. = [42.5,67.6] & Avg. Temp. = [29.01,51.836) & Max. Temp. = [42.5,46.22)
THEN
expected Avg. Load = 25568; std = 1680.9

RULE #4 IF
Type of Day = hol & Max. Temp. = [42.5,67.6] & Avg. Temp. = [29.01,51.836) & Max. Temp. = [46.22,50.27)
THEN
expected Avg. Load = 23407; std = 2124.1

RULE #5 IF
Type of Day = hol & Max. Temp. = [42.5,67.6] & Avg. Temp. = [29.01,51.836) & Max. Temp. = [50.27,56.696)
THEN
expected Avg. Load = 25508; std = 2331.8

RULE #6 IF
Type of Day = hol & Max. Temp. = [42.5,67.6] & Avg. Temp. = [29.01,51.836) Max. Temp. = [56.696,62.402)
THEN
expected Avg. Load = 22926; std = 1216.8

RULE #7 IF
Type of Day = hol & Max. Temp. = [42.5,67.6] & Avg. Temp. = [29.01,51.836) & Max. Temp. = [62.402,67.6]
THEN
expected Avg. Load = 28253; std = 0

RULE #8 IF
Type of Day = hol & Max. Temp. = [42.5,67.6] & Avg. Temp. = [51.836,59.756]
THEN

expected Avg. Load = 21770; std = 2977.7

RULE #9 IF
Type of Day = wnd & Min. Temp. = [4.2,19.7)
THEN
expected Avg. Load = 29715; std = 1568

RULE #10 IF
Type of Day = wnd & Min. Temp. = [19.7,26.492)
THEN
expected Avg. Load = 26617; std = 925.43

RULE #11 IF
Type of Day = wnd & Min. Temp. = [26.492,35.06)
THEN
expected Avg. Load = 25535; std = 581.39

RULE #12 IF
Type of Day = wnd & Min. Temp. = [35.06,56.894]
THEN
expected Avg. Load = 24788; std = 672.57

RULE #13 IF
Type of Day = wor & Avg. Temp. = [13.53,29.01)
THEN
expected Avg. Load = 31546; std = 878.03

RULE #14 IF
Type of Day = wor & Avg. Temp. = [29.01,32)
THEN
expected Avg. Load = 30561; std = 679.79

RULE #15 IF
Type of Day = wor & Avg. Temp. = [32,38.94) & Min. Temp. = [4.2,30.632)
THEN
expected Avg. Load = 29505; std = 569.1

RULE #16 IF
Type of Day = wor & Avg. Temp. = [32,38.94) & Min. Temp. = [30.632,35.06)
THEN
expected Avg. Load = 28791; std = 844.83

RULE #17 IF

Type of Day = wor & Avg. Temp. = [38.94, 43.898) & Global Ins. = [0,5798.34)
THEN
expected Avg. Load = 28790; std = 654.64

RULE #18 IF
Type of Day = wor & Avg. Temp. = [38.94, 43.898) & Global Ins. = 5798.34,27509.48]
THEN
expected Avg. Load = 27995; std = 434.26

RULE #19 IF
Type of Day = wor & Avg. Temp. = [43.898, 51.836) & Global Ins. = [0,5798.34) &
Min. Temp. = [28.202,43.5) & Wind = ???
THEN
expected Avg. Load = 27737; std = 424.69

RULE #20 IF
Type of Day = wor & Avg. Temp. = [43.898,51.836) & Global Ins. = [0,5798.34) &
Min. Temp. = [28.202, 43.5) & Wind = [0.3,7.25)
THEN expected Avg. Load = 28249; std = 124.15

RULE #21 IF
Type of Day = wor & Avg. Temp. = [43.898, 51.836) & Global Ins. = [0,5798.34)
Min. Temp. = [28.202,43.5) & Wind = [8.1,11)
THEN
expected Avg. Load = 28665; std = 85.193

RULE #22 IF
Type of Day = wor & Avg. Temp. = [43.898, 51.836) & Global Ins. = [0,5798.34) &
Min. Temp. = [43.5,56.894]
THEN expected Avg. Load = 26962; std = 0

RULE #23 IF
Type of Day = wor &Avg. Temp. = [43.898,51.836) & Global Ins. = [5798.34,19515.6)
THEN
expected Avg. Load = 27167; std = 364.16

RULE #24 IF
Type of Day = wor & Avg. Temp. = [51.836,59.756]
THEN
expected Avg. Load = 26786; std = 438.09

Figure 17 : Sample of rules for the average daily load of January 1992

Table 4: Forecasts based on rules generated by Statistical Decision Trees

Case #	Month	Forecast Variable	Average Error %	Standard Deviation	# of Rules
1	January	Average Load	3.80	3.39	24
2	February	Average Load	2.99	3.48	16
3	March	Average Load	4.69	6.82	12
4	January	Maximum Load	4.92	6.37	15
5	February	Maximum Load	3.28	2.99	21
6	March	Maximum Load	5.91	4.45	15

For holiday consumption patterns, the maximum temperature determines the load demand. The general observed trend of the rules is, higher the maximum temperature, lower is the demand. The working day patterns contain a richer variety of complex relationships as seen in rules 13 through 24. While the primary variable affecting the load consumption is average temperature, the minimum temperature and global insolation are also significant variables. *Wind*, which has a very small correlation coefficient of -0.02 and -0.12 with average load demand and maximum load demand respectively is a significant factor during some working days (implying a nonlinear relationship). The algorithm is able to extract the working day load consumption patterns very well. The maximum standard deviation of any working day rule is 2.93% and the mean standard deviation of all rules is 1.71%. Thus it can be inferred that the statistical decision tree algorithm is a powerful learning procedure. The true error rates can be estimated by performing simulations on

data not seen by the decision tree during the growing phase. Figures [19] - [21] are forecasts based on rules obtained from the decision tree. A band of expected load ± standard deviation specifies the confidence interval. The confidence interval is typically broader for holidays due to the fewer instances of holiday vectors in the training set. From the graphs it can be seen that the algorithm is able to pick the trend and pattern of load consumption from measurable variables such as temperature, wind and global insolation. The mean errors in forecast are tabulated in Table [4]. The errors are within 5% and from Figures [22]-[24] it is seen that errors are large only for holidays. From table [4] it is seen that this algorithm is able to forecast quite accurately the maximum load demand. It should be noted that these results were obtained by analysis of trees grown on raw metered data. No preprocessing was performed on the data prior to formation of the training sets. The robustness properties are quite evident from the results.

3.5.2 Maximum Load Forecasts

The maximum load demand pattern, as seen by the correlation analysis presented in Table 1, has very little correlation to any of the independent predictor variables. As correlation analysis is a test of linear relationships, it might be that there is a nonlinear or time delayed relationship between the input variables and peak load. The statistical decision tree algorithm is applied to forecast peak load demand. Simulations were conducted for all days of January, February and March 1992. The results are as shown in Figure [26]-[28]. The forecast if specified by a confidence interval of forecast value ± standard deviation of forecast. As seen from the graphs, highly improved forecasts, as compared to connectionist neural network models are obtained. The average errors are computed to be 4.92%, 3.28%, 5.91% and standard deviations of 6.37%, 2.99%, 4.45%.

for the three months respectively. These forecasts are not good enough for real world applications and need the methodology needs to be improved upon; Compact decision trees are produced as seen by the number of equivalent rules; 15 for January, 21 for February and 15 for March.

3.6 Two-step methodology to peak load forecast

A hybrid approach for forecast of peak load is presented. First, decision trees are built using a database. They identify the driving parameters for the forecast problem and express in a clear hierarchical fashion, their influence on the load. Next, the forecast is performed using the decision tree to obtain a robust average integrated load on the forecast day. As the peak load value is highly correlated to average load, the average load forecast value, obtained by the decision tree methodology is input to a neural network to obtain the peak load forecast. The neural network here, is the equivalent of a non-linear regression model. Application of the neural network, can be thought of as a refinement process. The neural network is used with an aim of improvement of precision of information. The training database creation, training dataset selection, data normalization and network architecture design methods are exactly the same as for the individual methods. Simulations are conducted are results are as shown in Figure [32]-[35] and Table [5] are the performance characteristics of the proposed "two-step" methodology. Working day forecasts are more important than weekend or holiday forecasts, as only during these days are the chances of load demand increasing to high levels exists. The forecast accuracy of the method for working days is very accurate with forecast errors of 2.45%, 1.76%, and 1.82% respectively.

3.7 Conclusion

In this chapter the statistical decision tree algorithm has been applied to extract concept description of the load forecasting process. The concepts acquired, knowledge, is expressed in human intelligible natural language low-arity "if-then" rules and equivalent graphic decision trees. Decision trees are constructed using a $\chi 2$ statistical test of significance. Stop splitting and terminal class assignment procedures are an inherent part of the tree growth process and no post-processing such as pruning are required. The algorithm is applicable to databases containing both numeric and symbolic data. Using the knowledge extracted from the load and weather databases, new unseen cases corresponding to forecast cases are applied and accurate results obtained. For peak load forecasts, highly robust forecasts are obtained by use of this paradigm. To improve the numerical accuracy of decision tree forecasts, neural networks are applied to refine the forecast load in a two step procedure. The working day, peak load forecast errors are 2.45%, 1.76% and 1.82% for the three months for which simulations was performed.

Table 5: Comparison of 1-step and 2-step methods of peak load forecast

Case #	Month	Method	Error %	S.D
1	January 1992	1-step	12.62	20.28
2	February 1992	1-step	9.45	14.31
3	March 1992	1-step	10.21	12.35
4	January 1992	2-step	4.12	5.07
5	February 1992	2-step	3.09	4.47
6	March 1992	2-step	5.98	8.10
7*	January 1992	2-step	2.45	3.24
8*	February 1992	2-step	1.76	1.50
9*	March 1992	2-step	1.82	1.48

* Working days only

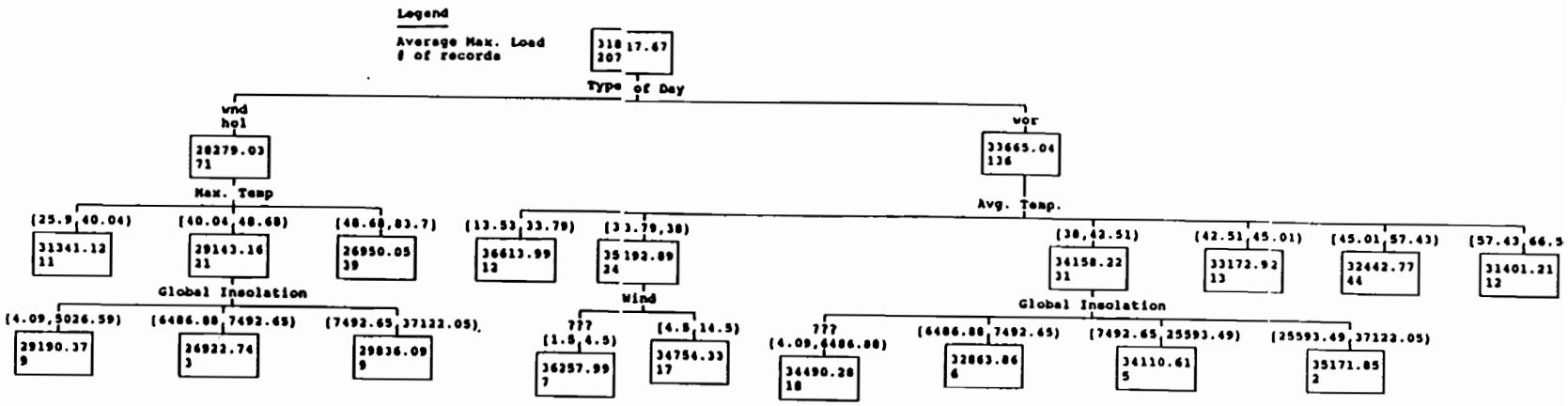

Fig. 19: Decision Tree obtained by Statistical Decision Tree Learning

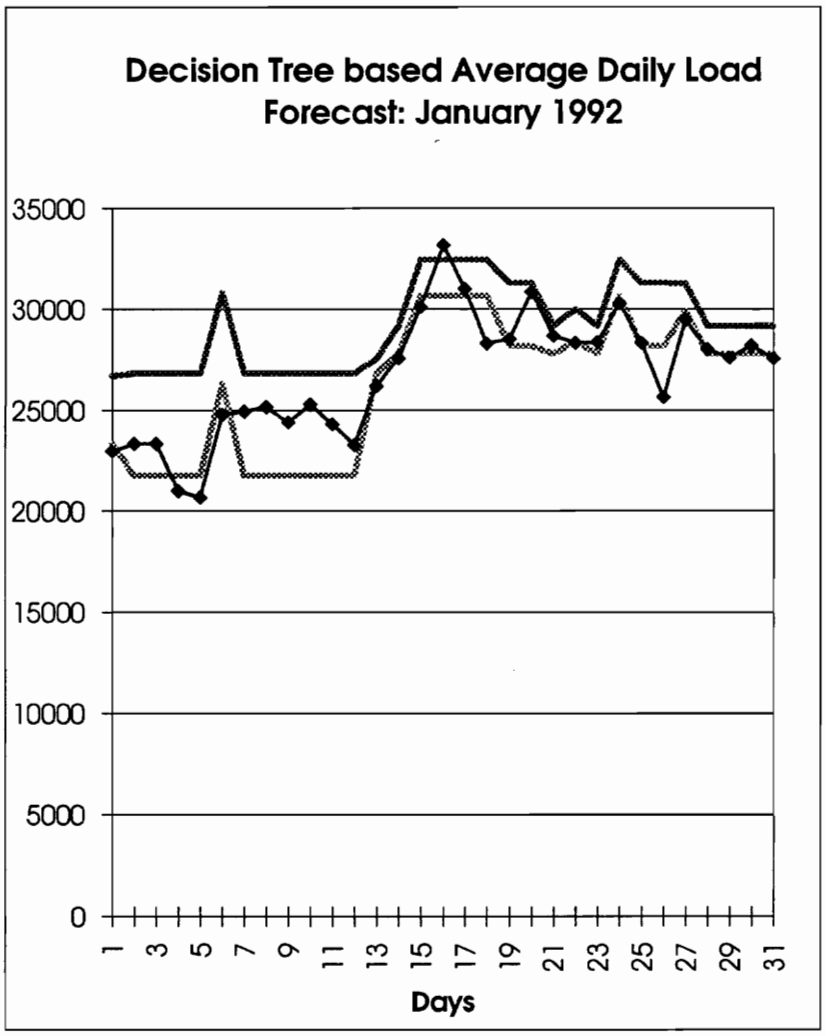

Figure 20: Decision Tree forecast of average load, January 1992

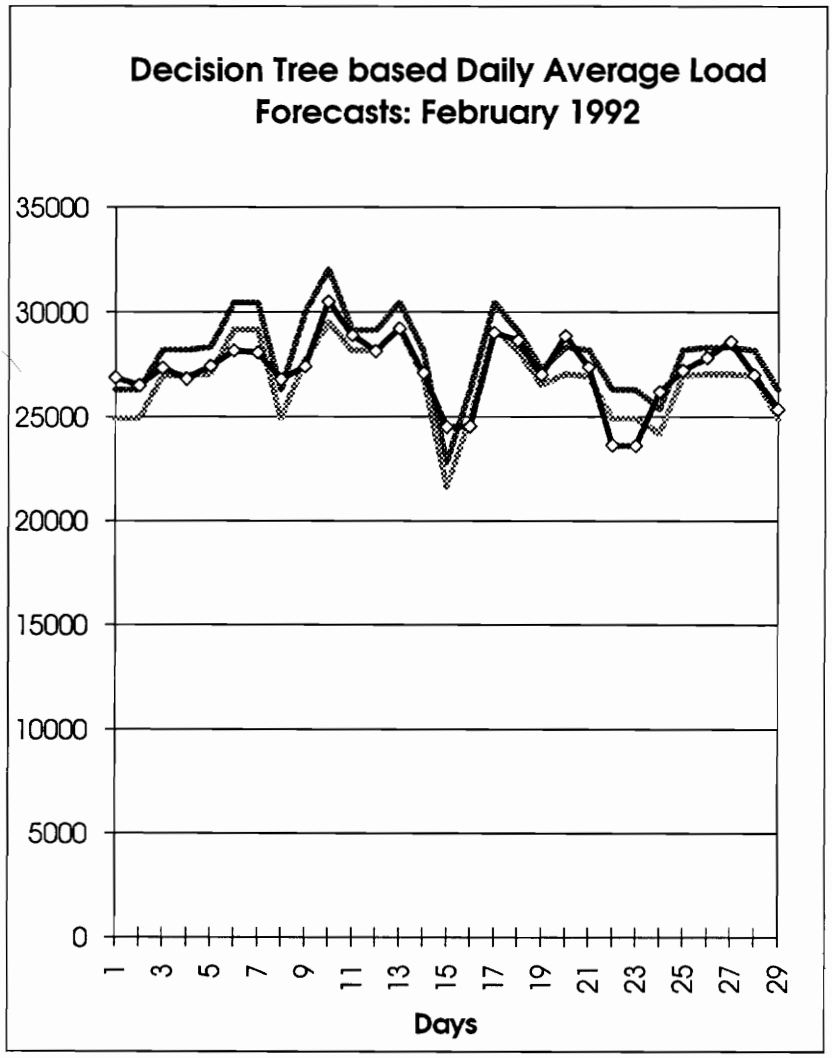

Figure 21: Decision Tree forecast of average load, February 1992

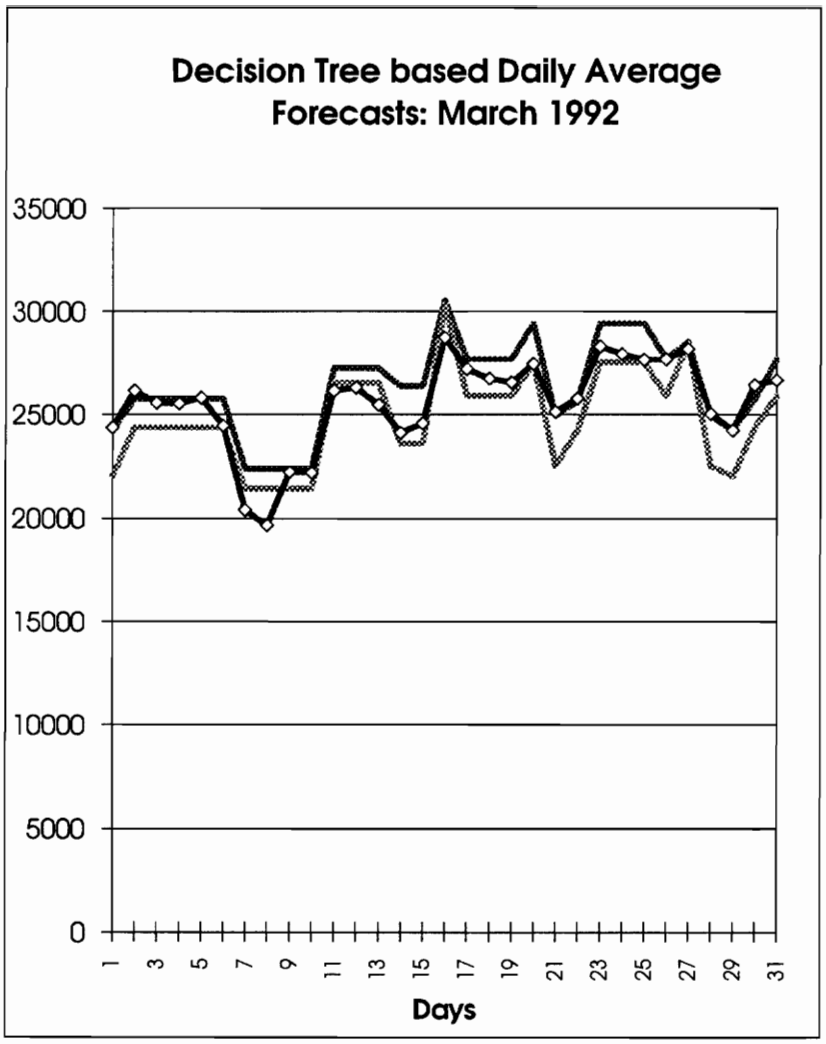

Figure 22: Decision Tree forecast of average load, March 1992

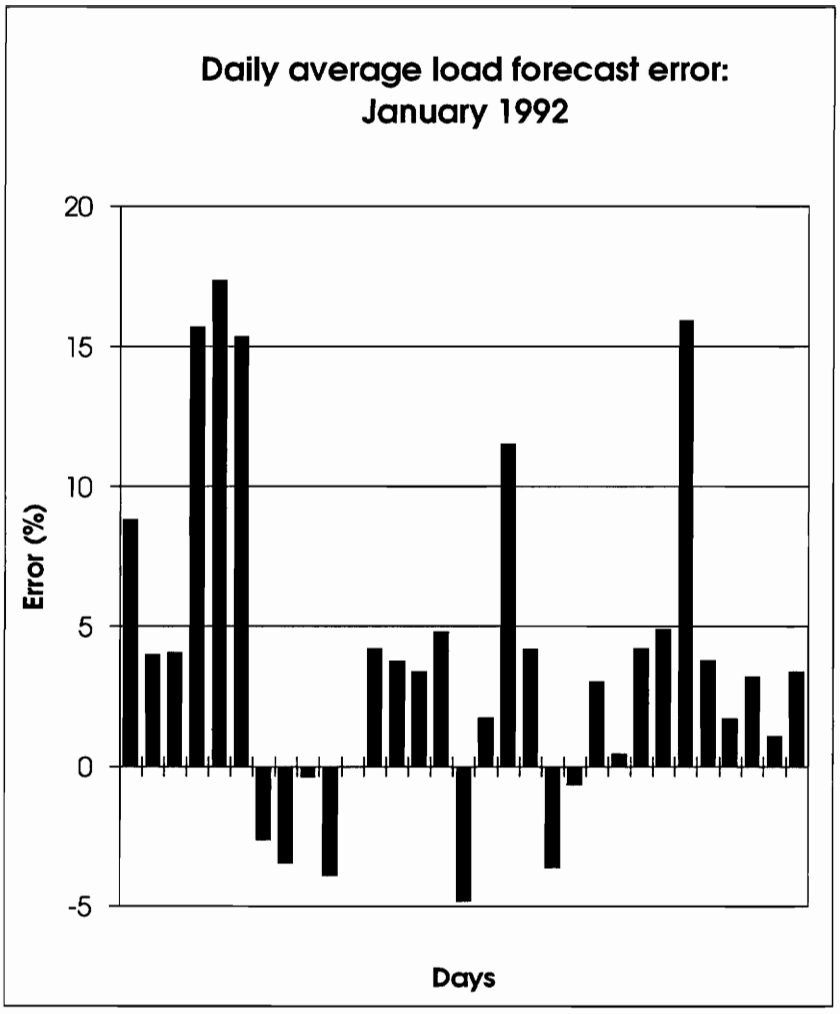

Figure 23: Average load forecast error, January 1992

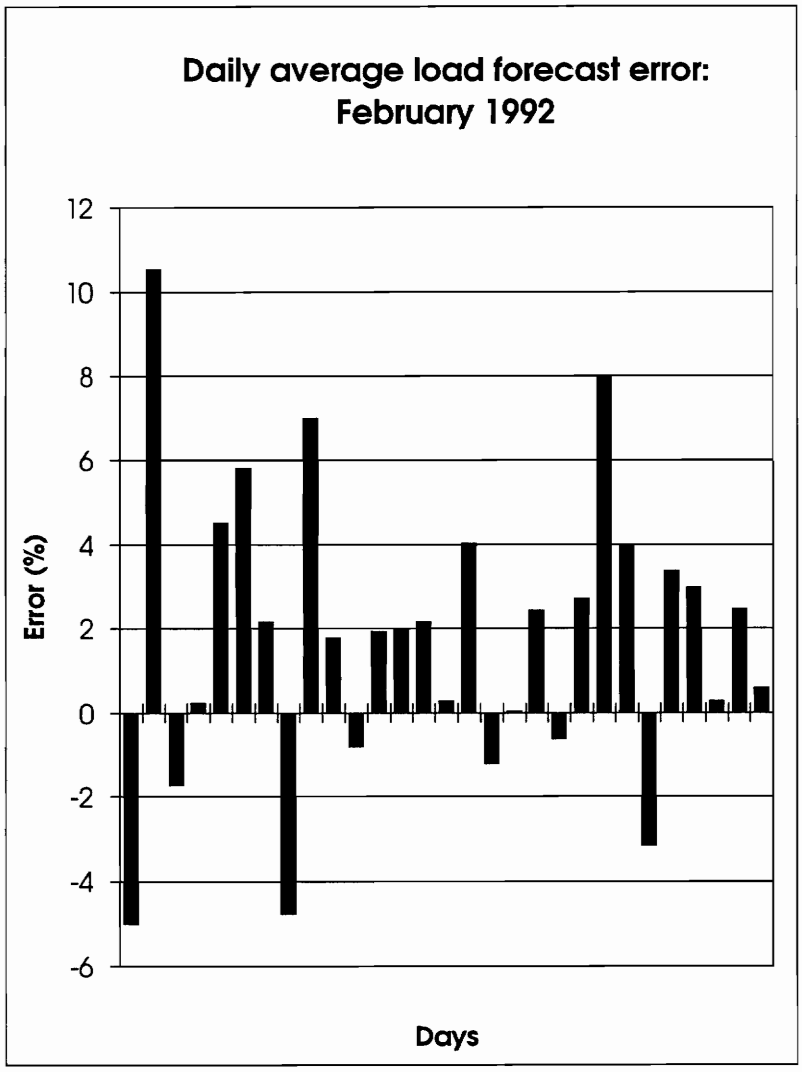

Figure 24: Average load forecast error, February 1992

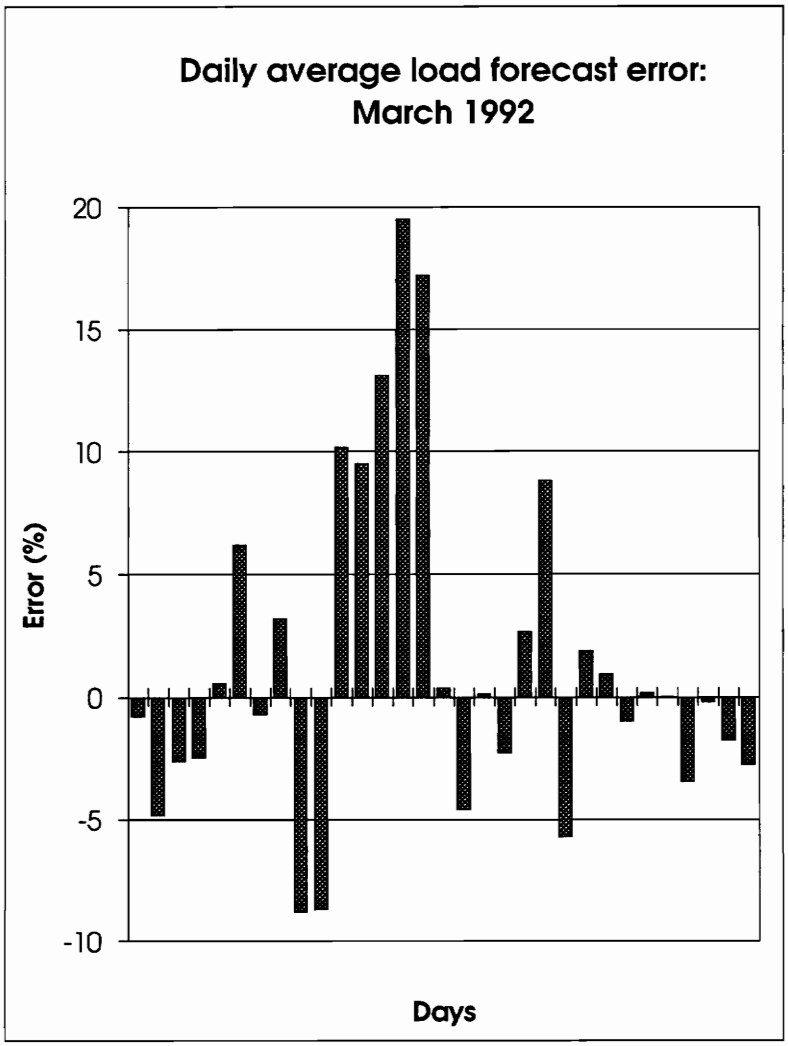

Figure 25: Average load forecast error, March 1992

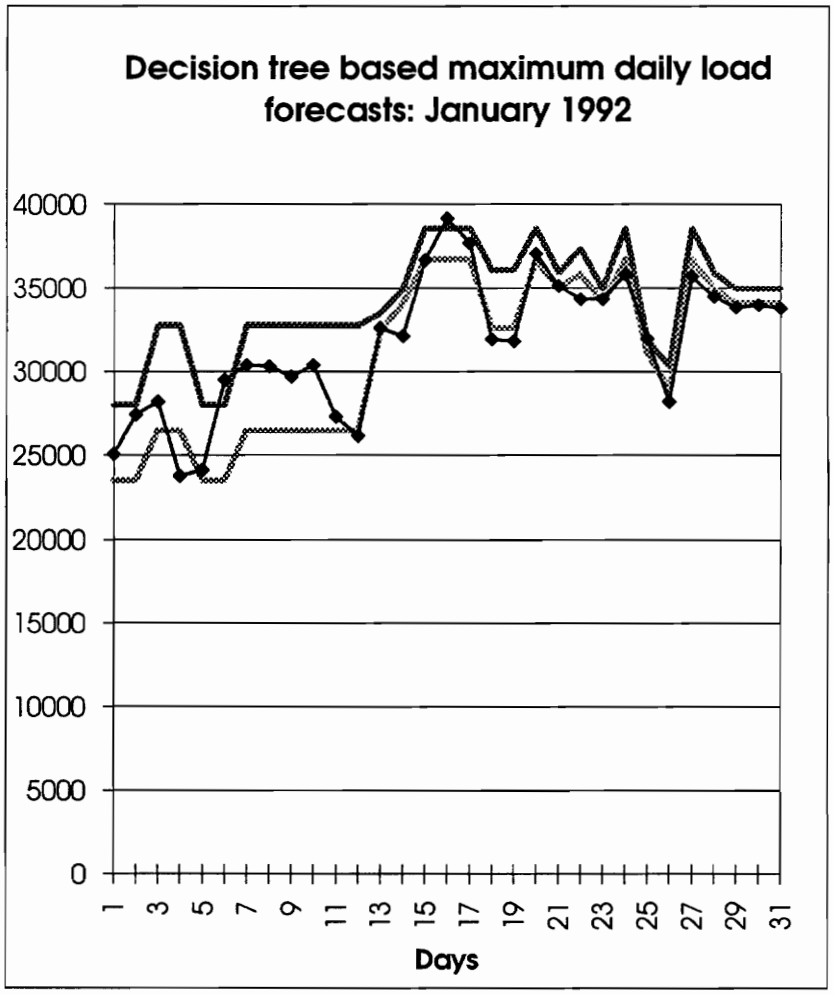

Figure 26: Decision Tree forecast of peak load, January 1992

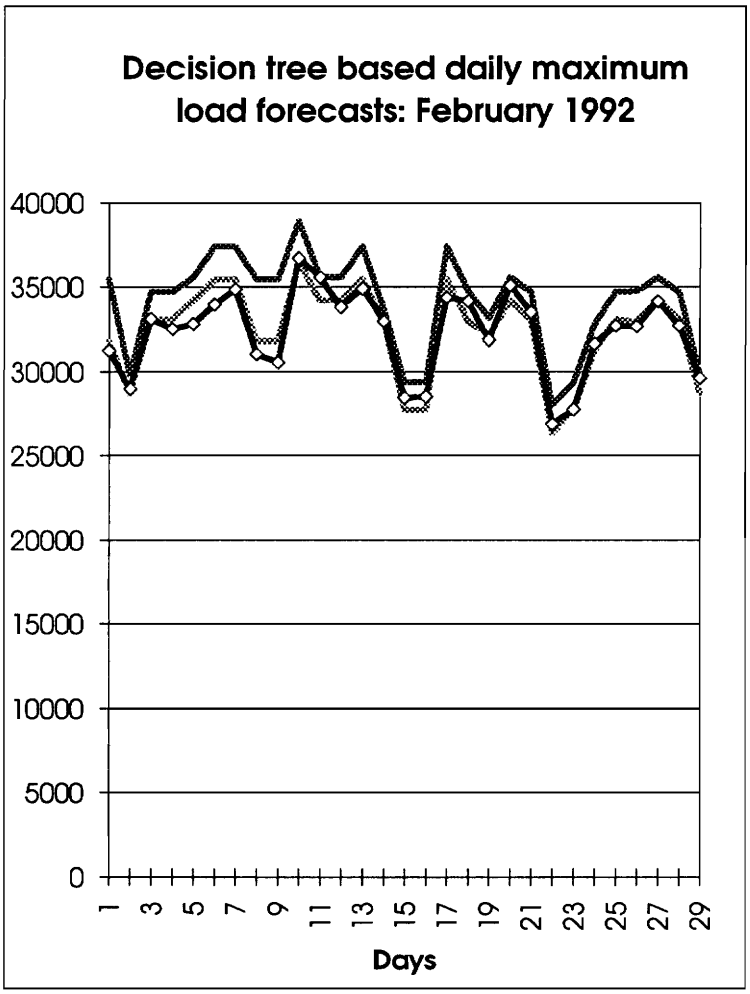

Figure 27: Decision Tree forecast of peak load, February 1992

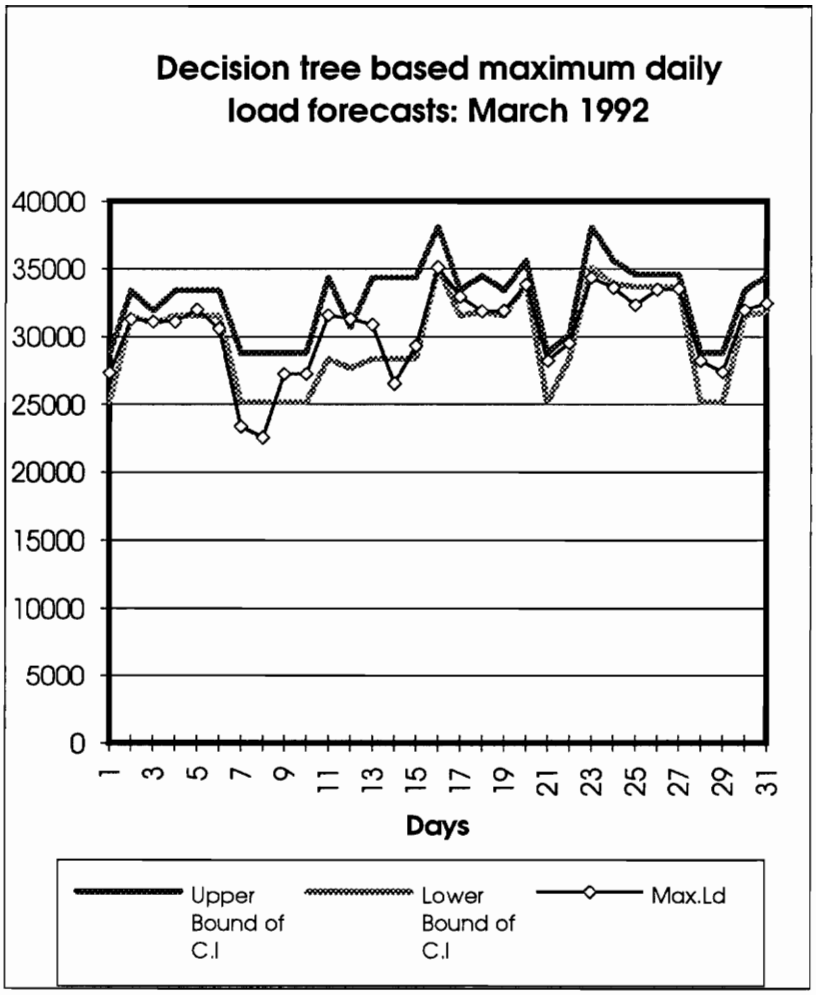

Figure 28: Decision Tree forecast of peak load, March 1992

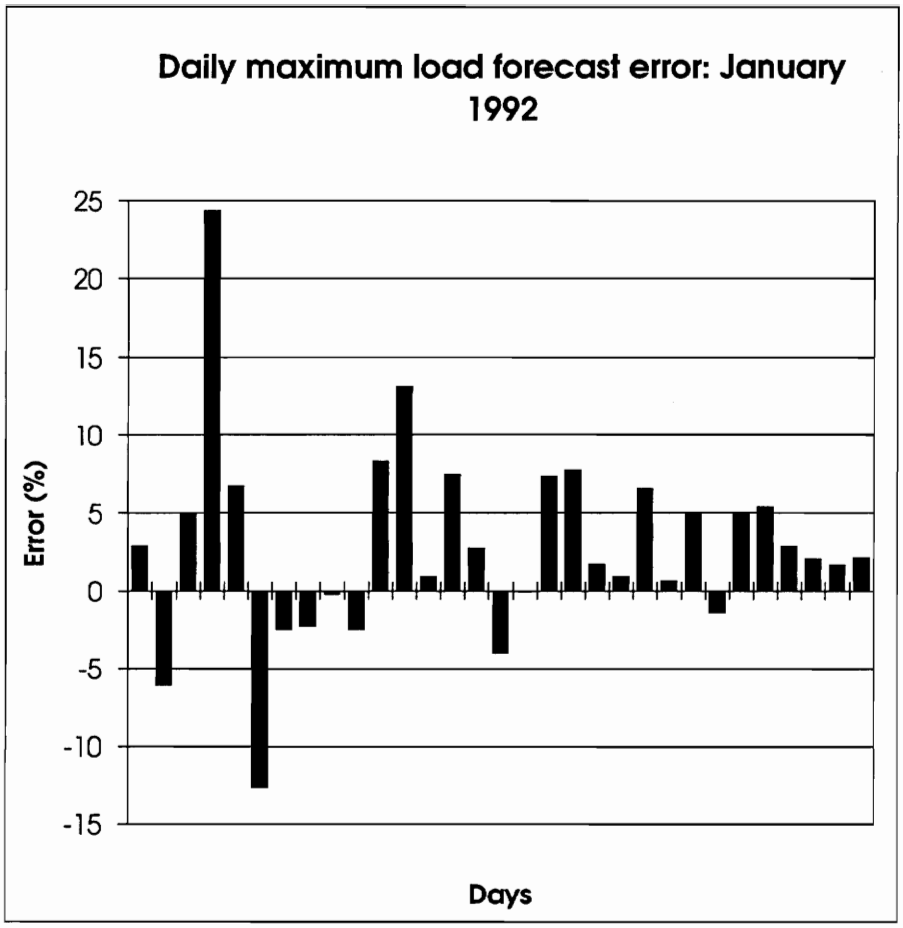

Figure 29: Forecast error of peak load, January 1992

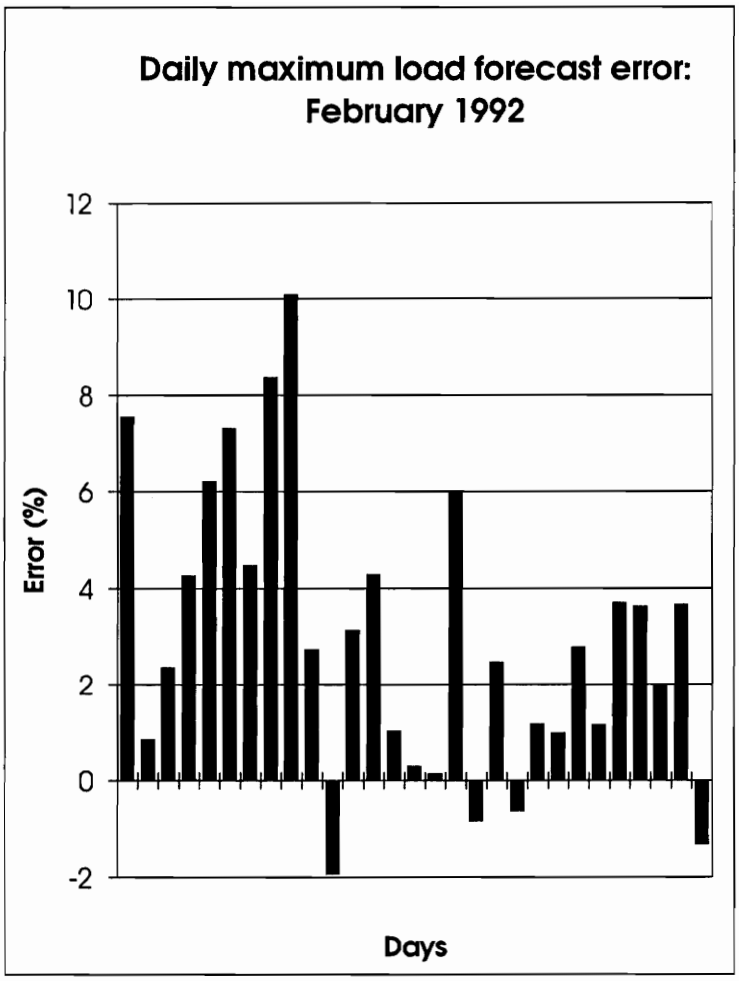

Figure 30: Forecast error of peak load, February 1992

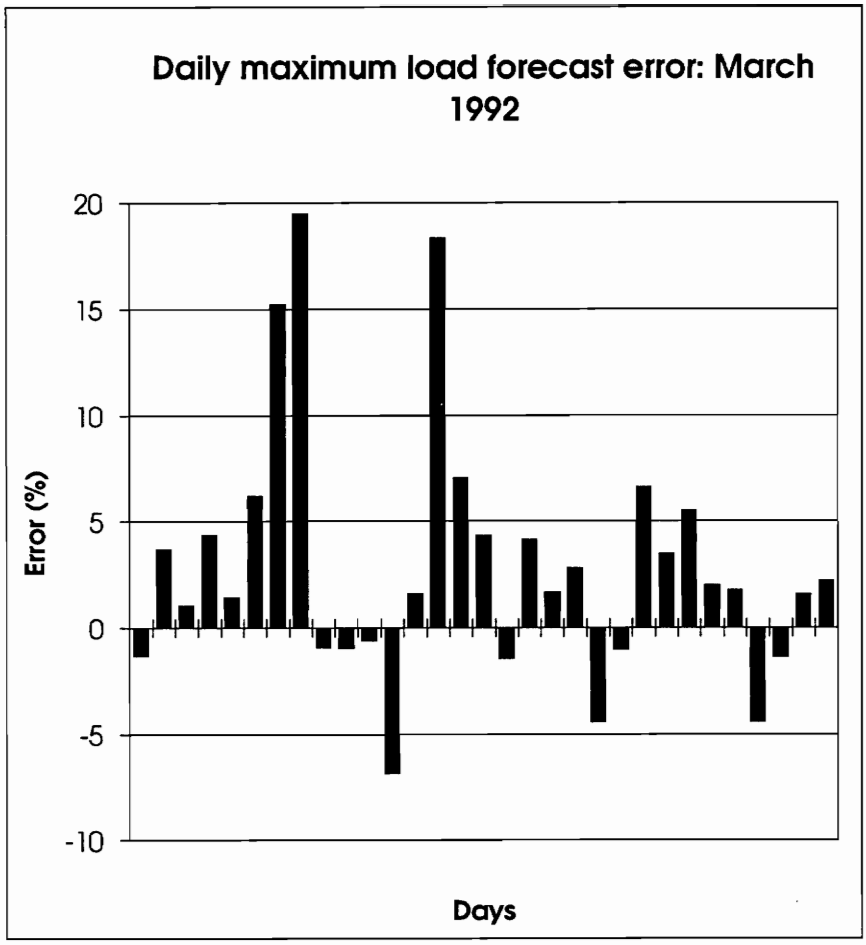

Figure 31: Forecast error of peak load, March 1992

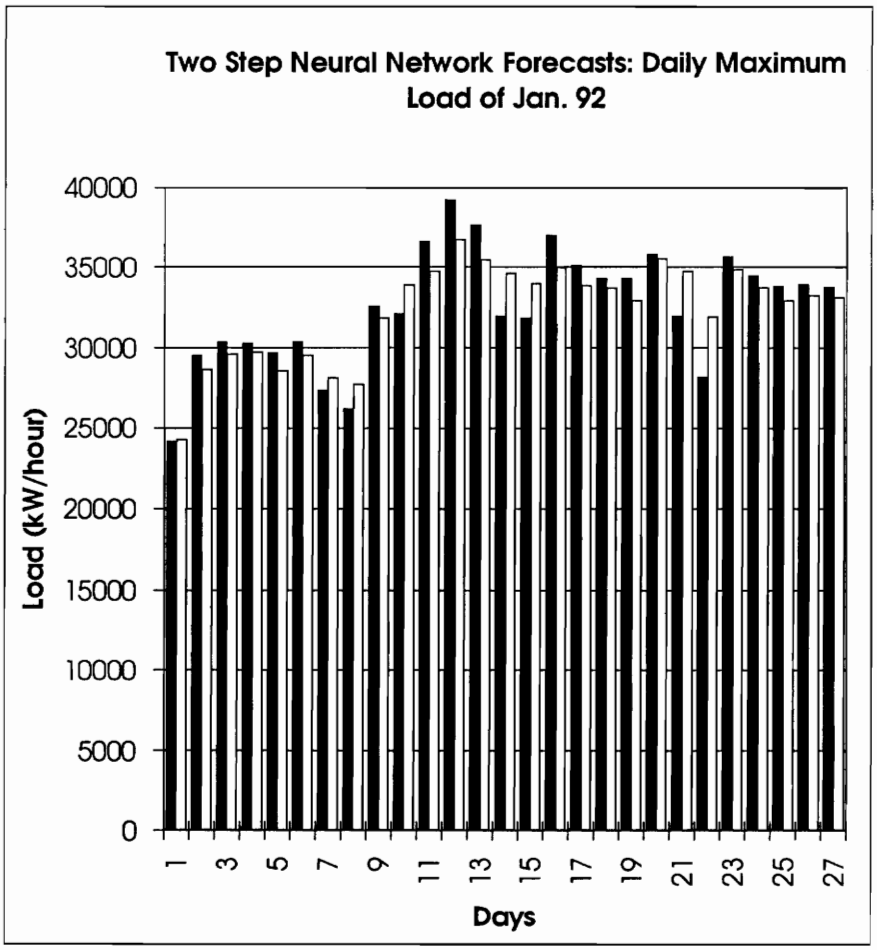

Figure 32: Two-step method of peak load forecast, January 1992

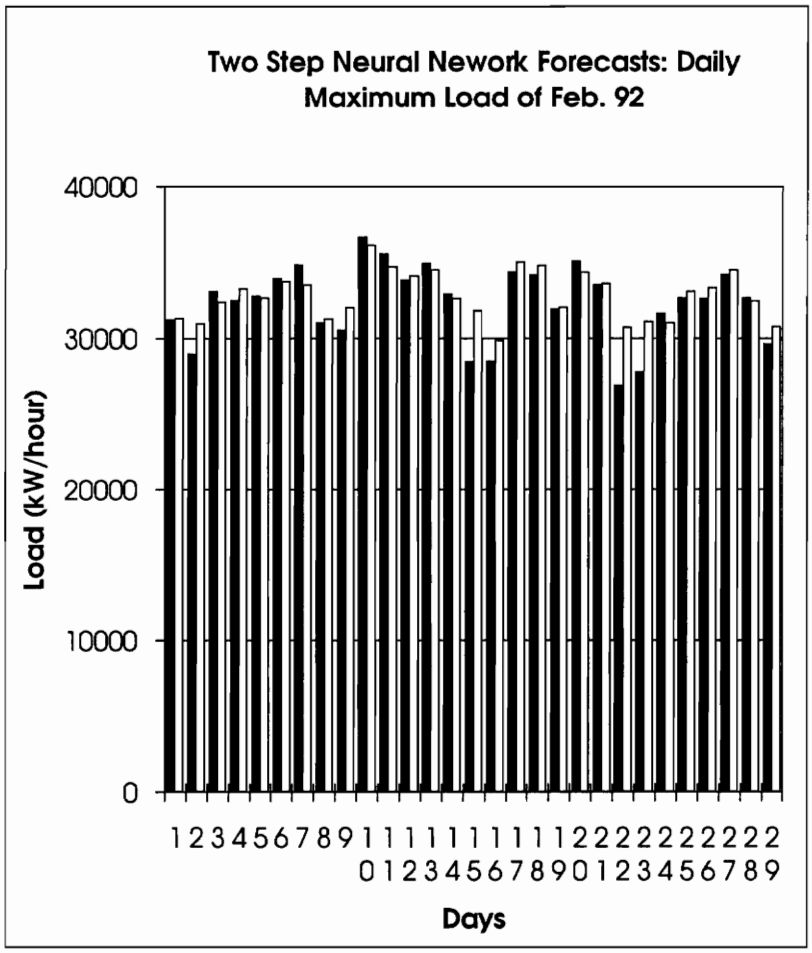

Figure 33: Two-step method of peak load forecast, February 1992

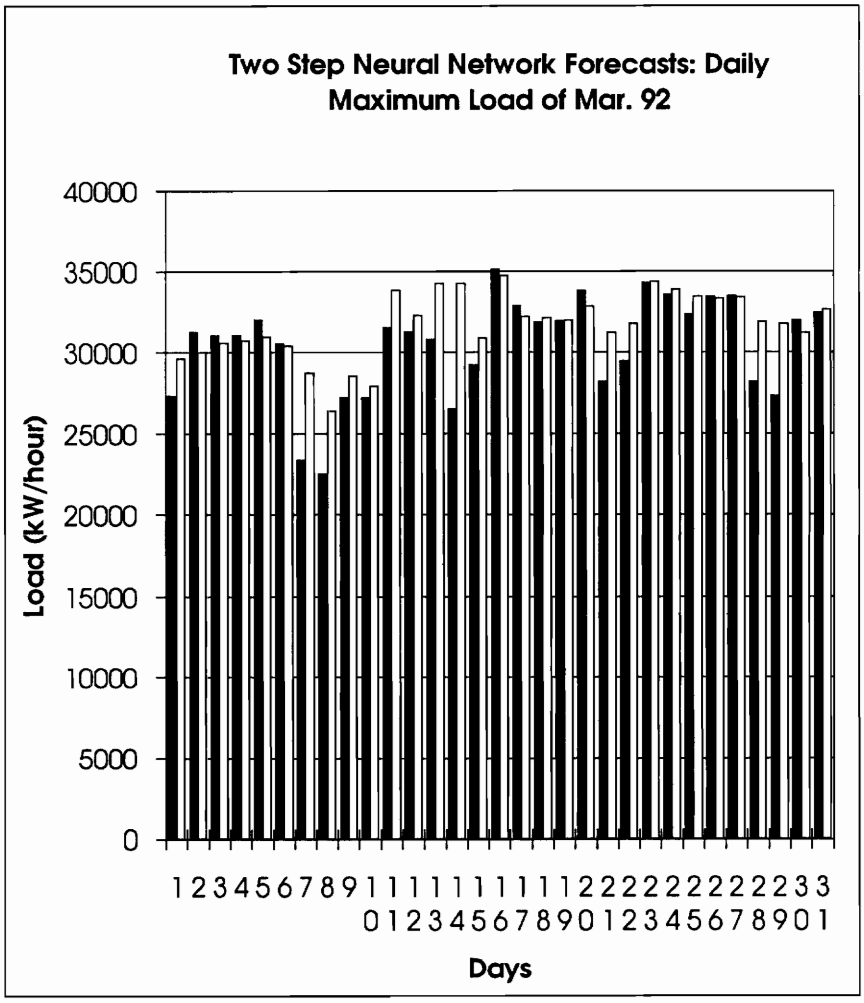

Figure 34: Two-step method of peak load forecast, March 1992

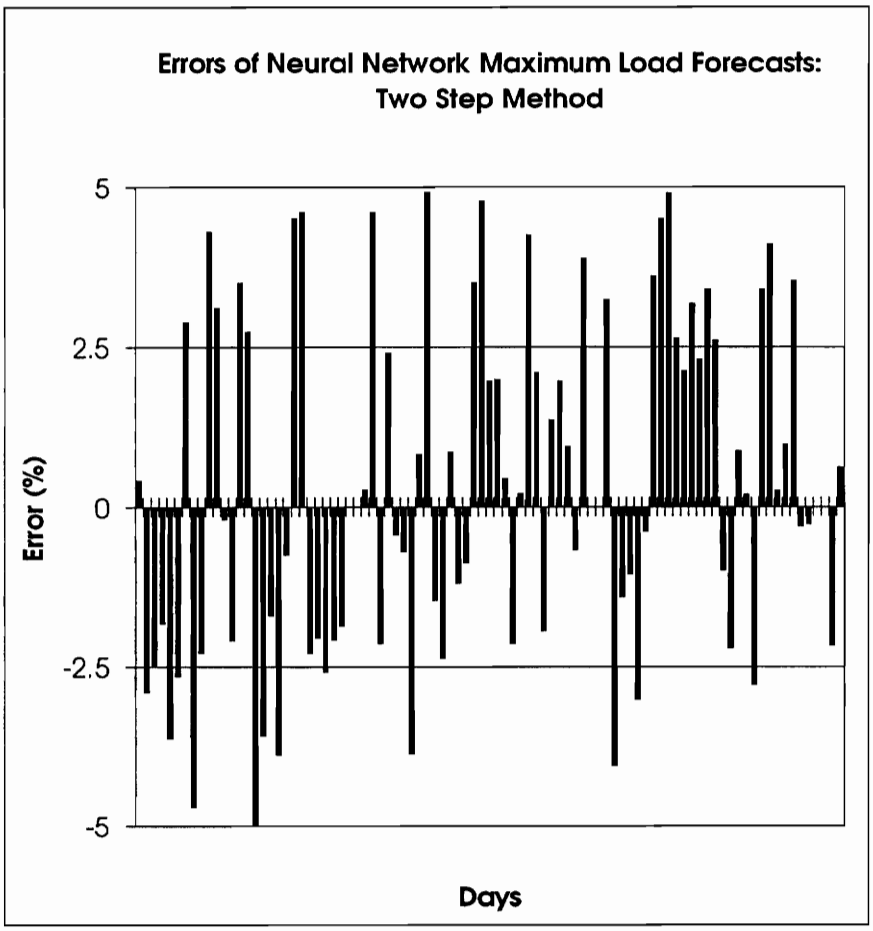

Figure 35: Two-step method forecast error (peak load),

of January, February and March 1992

CHAPTER 4

RESULTS, CONCLUSIONS AND RECOMMENDATIONS

4.1 Results and Conclusions

In this work, connectionist and symbolic machine learning techniques have been applied to forecast bus load. The problem of bus load forecasting has been considered first, from a modeling perspective. Connectionist neural network architectures have been applied to model the input-output relationships. Instead of modeling the entire load demand hypersurface formed by the input variables, a local function approximation technique is presented that uses a clustering algorithm called nearest neighbor norm to choose relevant datasets. The techniques, presented here can be applied to any non-linear function approximation, modeling and forecasting problem. Neural network training database creation is performed by utilizing a "relevant time window" concept. Training dataset selection is by application of the nearest neighbor norm method. This selection criteria improves the efficient use of a limited number of highly similar historical cases. These design issues have been presented and applied to forecast weekday average integrated load. The average error of forecast over a three month period, for working days, is 2.02% and standard deviation of forecast error is 2.74. As seen earlier, there are insufficient datasets to train neural networks on holiday patterns. Daily peak load demands have very little correlation to any of the input variables and the behaviour of the the peak load demand is chaotic. Neural networks forecast error is about 10%. Connectionist neural network models trained through the backpropagation algorithm, perform poorly in forecasting the peak load demand. Present day connectionist model based knowledge

acquisition algorithms are not mature enough to translate the connection weights into human intelligible natural language descriptions.

To solve the problems of poor performance of neural networks, holiday and peak load forecasting problem, and knowledge acquisition from databases, an alternate paradigm, symbolic machine learning technique is applied. Inductive learning techniques are particularly applicable to data-rich situations where examples that characterize the input-output relationships abound. Inductive learning algorithms extract general concept descriptions from examples and are apt for applications where the dependent (output) variable is unclassified, but takes on real vales continuous values.

The statistical decision tree algorithm is determined to be the one that is most applicable to bus load forecasting given the presence of raw, unstructured, noisy databases with complex context-sensitive relationships. The statistical decision tree utilizes a $\chi 2$ test of significance as the splitting criterion to partition data into homogenous compact clusters. The important issue of adjustment of significance test levels for pure subsets of data is discussed and a solution methodology presented. Average load forecast by this methodology has a forecast error of 3.83% and standard deviation of forecast error of 4.56. It is to be noted that for forecast of daily average load, neural networks are more accurate. However they cannot be applied to holidays due to insufficient data. Simulations are performed to forecast the daily peak load for all days and the average forecast error is 4.71% with standard deviation of forecast error of 4.61. It is to be noted that the type of day cases considered for simulations included holidays, weekends and weekdays. Knowledge acquired is expressed as human intelligible natural language "if-then" rules based on attributes discovered autonomously through significance tests. The robustness of decision tree methodology to noisy data is evident from the results. To combine the robustness of decision trees and numerical accuracy of connectionist modeling, a hybrid

two step method of peak load forecast is presented; Use is made of the fact that peak load is highly correlated on average load of forecast day. First, a robust estimate of average load is forecast using the decision tree methodology. Using the forecast value as an input to a connectionist model along with other predictor variables, a numerically accurate peak load forecast is obtained. The forecast accuracy using the two-step method is 4.39% and standard deviation of forecast error is 5.88 for all days. For working days only, the average forecast error is 2.01% and the standard deviation of forecast error is 2.07.

Thus, this work satisfies all the objectives set forth and has additional advantages, summarized below:

- A technique to extract knowledge from large unstructured databases, about forces driving the bus load demand, is developed and applied.
- The decision tree methodology is able to compress and organize large quantities of information present in load and weather databases in the form of high level natural language "if-then" rules and equivalent graphic decision trees. This provides the power system operations personnel confidence in the tool as they can understand the forecast procedure.
- A robust decision support system with which neural network forecast can be verified is presented.
- A procedure to improve the numerical accuracy of decision tree forecasts of peak load is presented.

4.2 Recommendations

The work presented here applies knowledge bases techniques, symbolic and connectionist, to bus load forecasting. The general purpose algorithm presented here is

portable and can be applied to diverse geographical areas. The technique presented here can be improved further and directions are suggested:

- Correlation analysis is a procedure for identification of linear relationships between output variable and a number of independent, input variables. Neural networks are nonlinear function approximation procedures. Their power to model complex nonlinear relationships is not fully exploited as input variables that have a highly nonlinear effect on load are not part of the input training set. Developments in the statistical sciences are necessary to identify significant nonlinear relationships. The $\chi 2$ test of significance is a possible solution. However, preprocessing of data to form contiguous, homogenous groups is necessary before the $\chi 2$ can be applied. Identification of significant relationships using statistical decision tree and use of the variables of the testing criteria as neural network variables is a path worth pursuing.

- This work has proposed a method that combines the robustness and reliability of decision trees and numerical accuracy of neural networks to forecast daily loads. New algorithms truly integrating the conceptual transparency of decision trees and the numerical accuracy of neural networks need to be developed. First, a decision tree can be grown, and next, the fully grown tree should be *transformed* to a deterministic neural network architecture. These methods avoid the trial and error procedure associated with determination of optimal number of hidden layer nodes, which was done in this work. The neural network model can then be trained using the error backpropagation algorithm to produce numerically accurate forecasts. This method would then truly integrate the symbolic and connectionist approaches. A load forecasting model and algorithm developed by this methodology can be used both as a human intelligible explanatory model and as a numerically accurate forecast tool.

BIBLIOGRAPHY

1. G. Gross and F. D. Galiana, "Short-Term Load Forecasting," *Proc. IEEE*, vol 75, pp 1558-1573, December 1987.

2. T.N. Goh, H.L. Ong and Y.O. Lee, "A new approach to statistical forecasting of daily peak power demand", *Elec. Power Syst. Res.*, vol. 10, No. 2, pp. 145-148, March 1986.

3. P.C. Gupta, "A stochastic approach to peak power demand forecasting in electric utility systems", *IEEE Trans. Power App. Syst.*, vol. PAS-90, no.4, pp. 1757-1767, July/Aug. 1971.

4. K.P. Rajurkar and J.L. Nissen, "Data-dependent system approach to short-term load forecasting", *IEEE Trans. Syst. Man. Cybern.*, vol. SMC-15, No. 4, pp. 532-536, July/Aug. 1985.

5. A.M. Schneider, T. Tanekawa and D. Schiffman, "24-hour electric utility load forecasting", in *Comparative Models for Electric Load Forecasting*", D.W. Bunn and E.D. Farmer, Eds., New York, NY, Wiley, 1985.

6. T. Poggio and F. Girosi, "Networks for Approximation and Learning", *Proc. of the IEEE*, vol.78, No. 9, Sept. 1990, pp. 1481-1497.

7. T. M. Peng, N. F. Hubele and G. G. Karady, "Conceptual Approach to the Application of Neural Network for Short-Term Load Forecasting," *Proc. IEEE 1990 International Symposium on Circuits and Systems*, pp 2942-2945, New Orleans, LA, May 1990.

8. M. A. El-Sharkawi, S. Oh, R. J. Marks, M. J. Damborg and C. Brace, "Short-Term Electric Load Forecasting Using an Adaptively Trained Layered Perceptron," *Proc. First International Forum on Application of Neural Networks to Power Systems*, pp 8-11, Seattle, WA, July 1991.

9. T. M. Peng, N. F. Hubele and G. G. Karady, "Advancement in the Application of Neural Networks for Short-Term Load Forecasting," *IEEE Trans. PWRS*, vol 7, pp 250-257, February 1992.

10. S. Rahman and R. Bhatnagar, "An Expert System Based Algorithm for Short Term Load Forecast," *IEEE trans. PWRS*, vol 3, pp 392-399, May 1988.

11. *Proc. of the First International Forum on Applications of Neural Networks to Power Systems*, M.A. El-Sharkawi, R. J. Marks, Eds., Seattle, Washington, July 23-26, 1991.

12. *Proc. of the Second International Forum on Applications of Neural Networks to Power Systems,* Yokohama, Japan, April 19-23, 1993.

13. J. T. Connor, L.E. Atlas and D. Martin, "Recurrent Neural Networks and Load Forecasting", *Proc. of the First International Forum on Applications of Neural Networks to Power Systems*, pp. 22-25, Seattle, Washington, July 1992

14. M. C. Brace, J. Schmidt and M. Hadlin, "Comparison of the Forecasting Accuracy of Neural Networks with Other Established Techniques," *Proc. First International Forum on Application of Neural Networks to Power Systems*, pp 31-35, Seattle, WA, July 1991.

15. Steven Piche, "Robustness of Feedforward Neural Networks", International Joint Conference on Neural Networks, pp. 346-351, vol. 2, Baltimore, Maryland, June 7-11, 1992.

16. L. Wehenkel and M. Pavella, "Decision Trees and Transient Stability of Electric Power Systems", *Automatica*, vol. 27, No. 1, pp. 115-134, 1991.

17. L. Atlas, R. Cole, J. Connor, M.A. El-Sharkawi, R.J. Marks II, Y. Muthuswamy, E. Barnard, "Performance comparisons between backpropagation networks and classification trees on three real-world applications", *Advances in neural information processing systems*, vol. 2, Touretzky, D., Morgan Kaufmann, San Mateo, California, 1990, pp. 622-629.

18. T. L. McClelland, D.E. Rumelhart, and the PDP Research Group, 1986, **Parallel Distributed Processing: Explorations in the microstructure of Cognition**, vol. 1, chapter 8, Cambridge, The MIT Press.

19. D. C. Park, O. Mohammad, M. A. El-Sharkawi and R. J. Marks II, "An Adaptively Trainable Neural Network Algorithm and Its Application to Electric Load Forecasting," *Proc. First International Forum on Application of Neural Networks to Power Systems*, pp 8-11, Seattle, WA, July 1991.

20. David A. White, Donald A. Sofge, "**Handbook of Intelligent Control**", van Nostrand Reinhold, New York, NY, 1992.

21. Fahlman and the Cascade Correlation Algorithm.

22. Jude W. Shavlik, Raymond J. Mooney, and Geoffrey G. Towell, "Symbolic and Neural Learning Algorithms: An Experimental Comparison", *Machine Learning*, vol. 6, pp. 11-143, 1991.

23. Stephen I. Gallant, "Connectionist Expert Systems", *Communications of the ACM*, vol. 31, No. 2, February 1988, pp. 152-169.

24. Ishwar K. Sethi and Jae H. Yoo, "SIR: Simultaneous Induction of Rules using Neural Networks", *International Joint Conference on Neural Networks*, vol. 1, pp. 359-363, Baltimore, Maryland, June 7-11, 1992.

25. Ryszard S. Michalski, Jaime G. Carbonell, Tom M. Mitchell, "**Machine Learning: An Artificial Intelligence Approach**", Morgan Kaufmann, Los Altos, California.

26. D. W. Bunn and E. D. Farmer, Comparative Models for Electrical Load Forecasting, John Wiley & Sons, New York, NY, 1985.

27. J. Quinlan, "Induction of Decision Trees", *Machine Learning*, vol. 1, pp. 86-106, 1986.

28. Leo Breiman, Jerome H. Freidman, Richard A. Olsen, Charles J. Stone, "**Classification and Regression Trees**", Wadsworth and Brooks/ Cole Advanced books and software, Pacific Grove, California, 1984

29. G.V. Kass, "An Exploratory Technique for Investigating large Quantities of Categorical Data", *Applied Statistics*, No. 2, vol. 29, pp. 119-127, 1980.

30. David Biggs, Barry de Ville, Ed Suen, "A method of choosing multiway partitions in classification and decision trees", *Journal of Applied Statistics*, vol. 18, No. 1, pp. 49-62, 1991.

31. S. T. Chen, D. C. Yu and A. R. Moghaddamjo, "Weather Sensitive Short-Term Load Forecasting Using Nonfully Connected Artificial Neural Network," *IEEE Trans. PWRS*, vol 7, pp 1098-1105, August 1992.

32. K. L. Ho, Y. Y. Hsu and C. C. Yang, "Short-Term Load Forecasting Using a Multilayer Neural Network with an Adaptive Learning Algorithm," *IEEE Trans. PWRS*, vol 7, pp 141-149, February 1992.

33. Lee Johnson, Class Notes, Math 4445, "Numerical Analysis Part I", Department of Mathematics, Virginia Tech.

34. Barry de Ville and Ed Suen, "Knowledge Seeker, 1st Mark Technologies, Suite 608, 16 Concourse Gate, Ottawa, Ontario, K2E-7S8, Canada

35. Ishwar K. Sethi, "Entropy Nets: From decision trees to neural networks", Proc. of the IEEE, vol. 78, No. 2, 1991, pp. 1605-1613.

Vita

Jayendar Rajagopalan was born on February 4, 1966 in Madras, India. He obtained his Bachelors degree in Electrical Engineering from the University of Madras in May 1988, and his Masters degree from the Indian Institute of Science, Bangalore, India in May 1990. In the Bachelors program he was awarded the academic excellence award in the junior, sophomore and senior years. He enrolled in the Masters Program at Virginia Tech in Fall 1990 and has been working on neural network and knowledge based system applications to power system engineering. His industrial work experience includes working with the HVDC analog simulator at the Central Power Research Institute, Bangalore during winter 1988, graduate scholarship with Mitsubishi Electric Corporation, Kobe Works, Japan from May 1989 to November 1990 and summer work with the Technical University of Delft and KEMA, The Netherlands in summer 1992 on neural network applications to power engineering. Mr. Rajagopalan enjoys reading, long drives on quiet roads and traveling.

R. Jayendar