NEURAL NETWORKS AND TIME SERIES FORECASTING

BY

NEJLA YIGIT

92537

SUBMITTED TO
DR. SAMEERA Y. KETOOLA

DEPARTMENT OF COMPUTER ENGINEERING
NEAR EAST UNIVERSITY
1988

CONTENTS

I sincerely [illegible]

preparing this [illegible]

survey. The [illegible]

committee and [illegible]

I [illegible]

extend [illegible]

[illegible]

to [illegible]

ACKNOWLDGEMENT

I sincerely wish to thank Dr. Sameera Y. Ketoola, under whose supervision I am presenting this graduation project. Her eager efforts of educating us cannot be put to words. But I dare say, that I have learned a lot from her profound lectures in the classroom and the guidance in this project.

I also wish to thank the faculty and staff of Near East University, who have assisted me in completing this work. The academic years at this institution have created a sense of maturity and professionalism in me as a future computer engineer, for which I will be ever in debt.

Nejla Yigit

PREFACE

In this paper for COM 400 course, I present the study for Neural networks and the time series forecasting. Artificial Neural networks, in technological terms, depicts the behavior of the human mind. Human brain - a complex organ of the human body, is being studied for centuries. However, most of its functions remains a mystery, though scientists and researchers have evaluated the main characteristics of the brain, which are memory, learning, adaptation, and forecasting. These functions are applied using networks of computers and various other electronic devices to simulate the "predicting behavior" of a given data model.

This paper includes references and scientific studies of various scientists and researchers. There are four chapters in this report which include Artificial Neural networks, times series and forecasting, learning in Neural networks and a program which emulates Artificial Neural networks.

In completing this project, I have used references from various authors and have tried to compile this information in orderly fashion. Since the stress is upon time series forecasting, I have used the information to deliver it in proper perspective. Not only I have learnt from the process of scrutinizing the information and selecting the right work, but I think this paper can be used for further references in Artificial Neural network field.

The process of writing paper has motivated me to read more about the subject even after the graduation. There is more to it then receiving just a grade. It has become my interest and may be in the future I can do further research in this topic.

CHAPTER 1

NEURAL NETWORKS

INTRODUCTION

Neural networks provide a unique computing architecture whose potential has only begun to be tapped. Used to address problems that are intractable or cumbersome with traditional methods, these new computing architectures - inspired by the structure of the brain - are radically different from the computers that are widely used today. Neural networks are massively parallel system that rely on dense arrangements of interconnections and surprisingly simple processor. [DoyHoff, P1]

Artificial Neural Networks take their name from the networks of nerve cells in the brain. Although a great deal of biological detail is eliminated in these computing models, the artificial neural networks retain enough of the structure observed in the brain to provide insight into how biological neural processing may work. Thus these models contribute to a paramount scientific challenge - the brain understanding itself. [DoyHoff, P1]

Neural networks provide an effective approach for a broad spectrum of applications. Neural networks excel at problems involving patterns - pattern mapping, pattern completion, and pattern classification. It may be applied to translate images into keywords, translate financial data into financial predictions, or map visual images to robotics commands. Noisy patterns - those with segment missing - may be completed with a neural network that has been trained to recall the completed patterns (for example, a neural network might input the outline of a vehicle that has been partially obscured, and produce an outline of the complete vehicle). [DoyHoff, P1]

Possible applications for pattern classification abound: Visual images need to be classified during industrial inspections; medical images, such as magnified blood cells, need to be classified for diagnostic test; sonar images may be input to a neural network for classification; speech recognition requires classification and identification of words and sequences of word. Even diagnostic problems, where results of test and answers to questions are classified into appropriate diagnoses, are promising areas for neural networks. The process of building a successful neural network application is complex, but the range of possible applications is impressively broad. [DoyHoff, P1-2]

Its utilize a parallel processing structure that has large numbers of processor and many interconnections between them. These processor are much simpler than typical central processing units (CPUs). In a neural network each processor is linked to many of its neighbors (typically hundreds or thousands) so that there are many more interconnections than processors. The power of the neural network lies in the tremendous number of interconnections. [DoyHoff, P2]

Neural networks are generating much interest among engineers and scientist. Artificial neural network models contribute to our understanding of biological models, provide a novel type of parallel processing that has powerful capabilities and potential hardware, and provide the potential for solving applications problems. [DoyHoff, P2]

It excite our imagination and relentless desire to understand the self, and in addition equip us with an assemblage of unique technological tools. But what has triggered the most interest in neural networks is that models similar to biological nervous system can actually be made to do useful computations, and, furthermore, the capabilities of the resulting systems provide an effective approach to previously unsolved problems. [DoyHoff, P2]

There are a variety of different neural network architectures, which illustrate their major components, and show the basic differences between neural networks and more traditional computers. Ours is a descriptive approach to neural network models and applications. Included are chapters on biological system that describe living nerve cells, synapses, and neural assemblies. The chapters on artificial neural networks cover a broad range of architectures and example problems, many of which can be developed further to provide possibilities for realistic applications.

RECOGNITION OF DIFFERENCE BETWEEN NEURAL NETWORKS AND TRADITIONAL TABLE

As discussed earlier, the traditional computer or a normal personal computer, does not have the ability to make decisions on it's own rather then, it relies on the instructions (in the form of add, subtract, multiply, division etc.) given by the programmer or a user. It can store these instruction in its memory, but evidently cannot modify or learn the patterns of computing algorithm on its own. This is where neural networks come into play. Rather then one microprocessor there are numerous parallel processors, which works simultaneously to provide exceptional computing power to the user.

A neural network derives its computing power through, first, its massively parallel distributed structure and, second, its ability to learn a therefore generalize; generalization refers to the neural network producing reasonable outputs for inputs not encountered during training (learning). These two information-processing capabilities make it possible for neural networks to solve complex (large-scale) problems that are currently intractable. In practice, however, neural networks cannot provide the solution working by themselves alone. Rather, they need to be integrated into a consistent system engineering approach. Specifically, a complex problem of interest is decomposed into a number of relatively simple tasks, and neural networks are assigned a subset of the tasks (e.g., pattern, recognition, associative memory, control) that match their inherent capabilities. It is important to recognize, however, that we have a long way to go (if ever) before we can build a computer architecture that mimics a human brain . [Haykin, P4]

Neural Networks offers the following useful properties and capabilities:

1. *Nonlinearity:* A neuron is basically a nonlinear device. Consequently, a neural network, made up of an interconnection of neurons, is itself nonlinear. Moreover, the nonlinearity is of a special kind in the sense that it is distributed throughout the network. Nonlinearity is a highly important property, particularly if the underlying physical mechanism responsible for the generation of an input signal (e.g., speech signal) is inherently nonlinear.)

2. *Input-Output Mapping:* A popular paradigm of learning called supervised learning involves the modification of the synaptic weights of a neural network by applying a set of labeled training samples or task examples. Each example consist of a unique input signal and the corresponding desired response. The network is presented an example signal picked at random from the set, and the synaptic weights (free parameters) of the network are modified so as to minimize the difference between the desired response and the actual response of the network produced by the input signal in accordance with an appropriate statistical criterion. The training of the network is repeated for many examples in the set until the network reaches a steady state, where there are no further significant changes in the synaptic weights ; The previously applied training examples may be reapplied during the training session but in a different order. Thus the network learns from the examples by constructing an input-output for the problem at hand. Such an approach brings to mind the study of nonparametric statistical inference which is a branch of statistics dealing with model-free estimation, or, from a biological viewpoint *tabula rasa* learning. Consider for example, a pattern classification task, where the requirement is to assign an input signal representing a physical object or even to one of several pre-specified categories (classes). In non parametric approach to this problem, the requirement is to estimate arbitrary decision boundaries in the input signal space for the pattern classification task using a set of examples, and to do so without invoking a probabilistic distribution model. A similar point if view is implicit in the supervised learning paradigm, which suggests a close analogy between the input-output mapping performed by a neural network and nonparametric statistical inference.
3. *Adaptivity.* Neural networks have a built-in capability to adopt their synaptic weights to changes in the surrounding environment. In particular, a neural network trained to operate in specific environment can be easily retrained to deal with minor changes in the operating environmental conditions. Moreover, when it is operating in a non-stationary environment (i.e., one whose statistics change with time), a neural network can be designed to change its synaptic weights in real time. The natural architecture of a neural network for pattern classification, signal processing, and control applications,

coupled with the adaptive capability of the network, makes it an ideal tool for use in adaptive pattern classification, adaptive signal processing , and adaptive control. As a general rule, it may be said that more adaptive we make a system in a properly designed fashion, assuming the adaptive system is stable, the more robust its performance will likely be when the system is required to operate in non-stationary environment. It is emphasized, however, that adaptivity does not always lead to robustness; indeed, it may do the very opposite. To realize the full benefits of adaptivity, the principle time constants of the system should be long enough to respond to meaningful changes in the environment.

4. *Evidential Response.* In the context of pattern classification, a neural network can be designed to provide information not only about which particular pattern to select, but also about the confidence in the decision made. The latter information may be used to reject ambiguous patterns, should they arise, and thereby improve the classification performance of the network.
5. *Contextual Information.* Knowledge is represented by the very structure an activation state of neural network. Every neuron in the network is potentially affected by the global activity of all other neurons in the network. Consequently, contextual information is dealt with naturally by a neural network.
6. *Fault Tolerance.* A neural network, implemented in hardware form, has the potential to be inherently fault tolerant in the sense that its performance is degraded gracefully under adverse operating conditions. For example, if a neuron or its connecting links are damaged, recall of a stored pattern is impaired in quality. However, owing to the distributed nature of information in the network, the damage has to be extensive before the overall response of the network is degraded seriously. Thus, in principle, a neural network exhibits a graceful degradation in performance rather than a catastrophic failure.
7. *VLSI Implementability.* The massive parallel nature of neural network makes it potentially fast for the computation of certain tasks. The same feature makes the neural network ideally suited for implementation using *very-large-scale-integrated* (VLSI) technology. The particular virtue of VLSI is that it provides a means of

capturing truly complex behavior in a highly hierarchical fashion., which makes it possible to use a neural network as a tool for real-time applications involving pattern recognition, signal processing and control.

8. *Uniformity of Analysis and Design.* Basically neural networks enjoy universality as information processors. We say this in the sense that the same notation is used in all the domains involving the application of neural networks. This feature manifests itself in different ways:

- Neurons in one form or another, represent an ingredient common to all neural networks.
- this commonality makes it possible to share theories and learning algorithms in different application of neural networks.
- Modular networks can be built through a seamless integration of modules.

BASIC STRUCTURE OF ARTIFICIAL NEURAL NETWORKS

Figure 1.1 depicts an example of a typical processing unit for an artificial neural network. On the left are the multiple inputs to the processing unit, each arriving from another unit shown at the center. Each inter connection has an associated connection strength, given as $w_1, w_2, \ldots w_n$. The processing unit performs a weighted sum on the inputs and uses a nonlinear threshold function, *f*, to compute its output. The calculated result is sent along the output connections to the target cells shown at the right. The same output value is sent along the output connections. [DoyHoff P7]

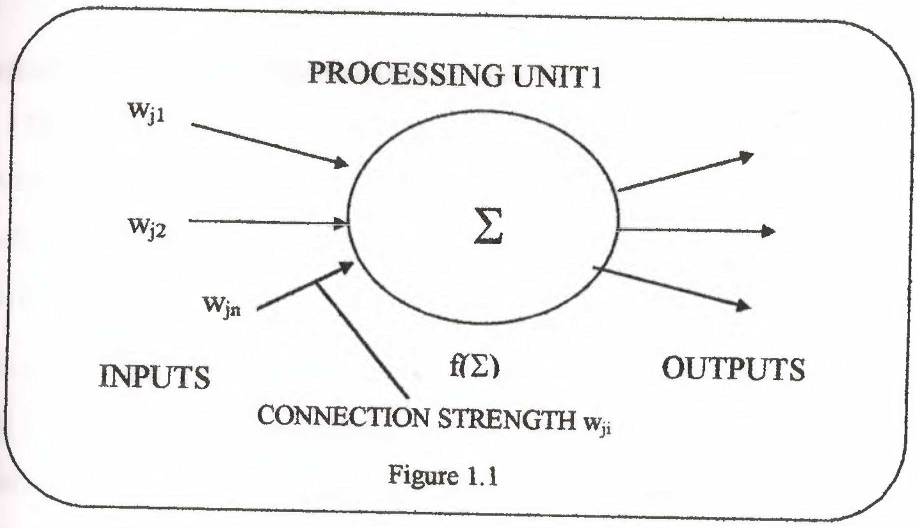


Figure 1.1

The neural network shown in figure 1.2 has three layers of processing units, a typical organization for the neural net paradigm known as back-error propagation. First is a layer of input units. These units assume the values of a pattern represented as a vector, that is input to the network. The middle *hidden* layer of this network consists of *feature detectors* - units that respond to particular features that may appear in the input pattern. Sometimes there is more than one hidden layer. The last layer is the output layer. The activities of these units are read as the output of the network. In some applications, output units stand for different classification patterns. However, Neural networks are not limited to three layers, and may utilize a huge number of interconnections. [DoyHoff P8]

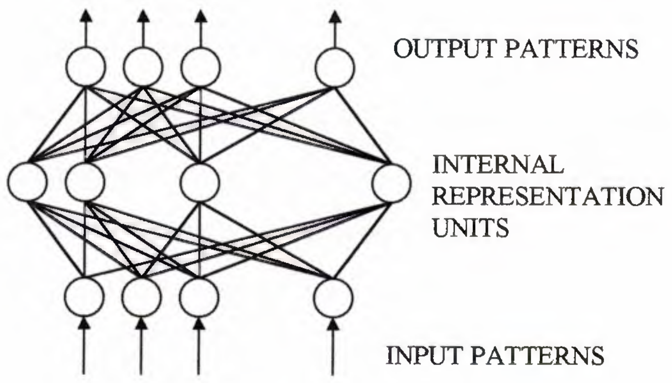


FIGURE 1.2

Each interconnection between processing units acts as a communication processing unit to another. These values are weighted by a connection strength when they are used computationally by the target processing unit. The connection strengths that are associated with each interconnection are adjusted during training to produce the final Neural network. [DoyHoff P8]

Some Neural networks applications have fixed interconnection weights; these networks operate by changing activity levels of neurons without changing the weights. Most networks, however, undergo a *training* procedure during which the network weights are adjusted. Training may be *supervised,* in which case the network is presented with target answers for each pattern that is input. In some architectures, training is unsupervised - the network adjusts its weights in response to input patterns without the benefit of target answers. In unsupervised learning, the network classifies the input patterns into similarity categories. [DoyHoff P10]

CHARACTERISTICS OF NEURAL NETWORKS

Neural networks are not programmed; they learn by example. Typically a Neural network is presented with a training set consisting of a group of examples from which the network can learn. These examples, known as training patterns, are represented as vectors, and can be taken from such sources as images, speech signals, sensor data, robotic arm movements, financial data and diagnosis information. [DoyHoff P10]

The most common training scenarios utilize supervised learning, during which the network is presented with the target output for that pattern. The target output usually constitutes the correct answer, or correct classification for the input pattern. In response to these paired examples, the Neural networks adjusts the values of its internal weights. If training is successful, the internal parameters are then adjusted to the point where the network can produce the correct answers in response to each input pattern. Usually the set of training examples is presented many times during training to allow the network to adjust its internal parameters gradually. [DoyHoff P10]

Because the learn by example, Neural networks have the potential for building computing systems that do not need to be programmed. This reflects a radically different approach to computing compared to traditional methods, which involve the development of computer programs. In a computer program, every step that the computer executes is specified in advance by the programmer, a process that takes time and human resources. The Neural network, in contrast, begins with sample inputs and outputs, and learns to provide the correct output for each input. [DoyHoff P10]

The Neural networks approach does not require human identification of features, or human development of algorithms or programs that are specified to the classification problem at hand, suggesting that the time and human effort can be saved. There are draw backs to the Neural network approach, however: That the time to train the network may not be known *a priori*, and the process of designing a network that successfully solves an application problem may be involved. The potential of the approach, however, appears significantly better than past approaches. [DoyHoff P10]

Neural network architectures encode information in a distributed fashion. Typically the information that is stored in a neural net is shared by many of its processing units. This type of coding is in stark contrast to traditional memory schemes, where particular pieces of information are stored in particular locations of memory. Traditional speech recognition systems, for example, contain a lookup table of template speech patterns (individual syllables or words) that are compared one by one to spoken inputs. Such templates are stored in a specific location of the computer memory. Neural networks, in contrast, identify spoken syllables by using a number of processing units simultaneously. The internal representation is thus distributed across all or part of network. Furthermore, more than one syllable or pattern may be stored at the same time by the same network. [DoyHoff P10]

Distributed storage schemes provide many advantages, the most important being that the information representation can be redundant. Thus a Neural network system can undergo partial destruction of the network and may still be able to function correctly. Although redundancy can be built into other types of systems, the neural network has a natural way to organize and implement this redundancy; the result is naturally fault or error-tolerant system. [DoyHoff P12]

It is possible to develop a network that can generalize on the tasks for which it is trained, enabling the network to provide the correct answer when presented with new input pattern that is different from the inputs in the training set. To develop a Neural network which can generalize, the training set must include a variety of examples that are good preparation for the generalization task. In addition the training session must be limited in iterations, so that no "over learning" takes place (i.e., the learning of specific examples instead of classification criteria, which is effective and general). Thus, special considerations in constructing the training set and training presentations must be made to permit effective generalization behavior from a Neural network. [DoyHoff P13]

A Neural network can discover the distinguishing features needed to perform a classification task. This discovery is actually a part of the network's internal self-organization. The organization of features takes place in back-propagation. A network may be presented with a training set of pictures, along with the correct classification of these pictures into categories. The network can then find the distinguishing features between the different categories of pictures. These features can be read off from a "feature detection" layer of neurons after the network is trained. [DoyHoff P13]

A Neural network can be *tested* at any point during training. Thus it is possible to measure a learning curve (not unlike learning curves found in human learning sessions) for a Neural network. [DoyHoff P13]

All of these characteristics of Neural networks may be explained through the simple mathematical structure of the neural net models. Although we use broad behavioral terms such as learn, generalize, and adapt, the Neural network's behavior is simple and quantifiable at each node. The computations performed in the neural net may be specified mathematically, and typically are similar to other mathematical methods already in use. Although large Neural network systems may some times act in surprising ways, their internal mechanism are neither mysterious nor incomprehensible. [DoyHoff P13]

APPLICATIONS POTENTIAL OF NEURAL NETWORKS

Neural networks have far-reaching potential as building blocks in tomorrow's computational world. Already, useful applications have been designed, built and commercialized, and much research continues in hopes of extending this success. [DoyHoff P13]

Neural network applications emphasize areas where they appear to offer a more appropriate approach than traditional computing has. Neural networks offer possibilities for solving problems that require pattern recognition, pattern mapping, dealing with noisy data, pattern completion, associative lookups, or systems that learn or adapt during use. Examples of specific areas where these types of problems appear include speech synthesis and recognition, and image processing and analysis, sonar and seismic signal classification, and adaptive control. In addition, Neural networks can perform some knowledge processing tasks, and can be used to implement associative memory. Some optimization tasks can be addresses with Neural networks. The range of potential application is impressive. [DoyHoff P14]

The first highly developed application was handwritten character identification. A Neural network is trained on a set of handwritten characters, such as printed letters of the alphabet. Network training set then consists of the handwritten characters as inputs together with the correct identification for each character. At the completion of training, the network identifies handwritten characters in spite of the variation of the handwriting. [DoyHoff P14]

Another impressive applications study involved NETtalk, a Neural network that learns to produce phonetic strings, which in turns specify pronunciation for written text. The input to the network in this case was English text in form of successive letters that appear in the sentences. The output of the network was phonetic notation for the proper sound to produce given the text input. The output was linked to a speech generator so that an observer can hear the network learn to speak. This network trained by Sejnowski and Rosenberg (1987), learned to pronounce English text with a high level of accuracy. [DoyHoff P14]

Neural network studies have also been done for adaptive control applications. A classic implementation for Neural network control system was the broom-balanced experiment, originally done by Widrow (Widrow and Smith, 1963)using a single layer of adaptive network weights. The network learned to move a cart back and forth in such a way that a broom balanced upside down on its handle tip in the cart remained on end. More recently, application studies were done for teaching a robotic arm how to get to its target position, and to steadying a robotic arm. Research was also done on teaching a Neural network to control an autonomous vehicle using simulated, simplified vehicle control situations. [DoyHoff P14]

Many other applications, over a wide spectrum of fields, have been examined. Neural network were configured to implement the associative memory systems. They were applied to a variety of financial analysis problems, such as credit assessment and financial forecasting. Signal analysis has been attempted with Neural networks, as well as difficult pattern classification tasks that arise in biochemistry. In music, a string-fingering problem, that of assigning successive string and finger positions for a difficult violin passage, was studied with a Neural network approach. [DoyHoff P14]

Neural networks are expected to complement rather than replace other technologies. Tasks that are done well by traditional computer methods need not be addressed with Neural networks, but technologies that complement Neural networks are far-reaching. For example, expert systems and rule based knowledge processing techniques are adequate for some applications, although Neural networks have the ability to learn rules more flexibly. More sophisticated systems may be built in some cases from a combination of expert systems and Neural networks. Sensors for visual or acoustic data may be combined in a system that includes a Neural network for analysis and pattern recognition. Sound generators and speech-synthesizing electronic equipment may be combined with Neural networks to provide auditory inputs and outputs. Robotics and control systems may use Neural network components in the future. Simulation techniques, such as simulation languages, may be extended to include structures that allow us to simulate Neural networks. Neural networks may also play a new role in the optimization of engineering designs and industrial resources. [DoyHoff P15]

DESIGN CHOICES IN NEURAL NETWORKS

Many design choices are involved in developing a neural network application. (Figure 1.3). The first option is in choosing the general area of application. Usually this is an existing problem that appears amenable to solution with a Neural network. Next the problem must be defined specifically so that a selection of inputs and outputs to the network may be made. Choices for inputs and outputs involve identifying the types of patterns to go into and out of the network. [DoyHoff P15]

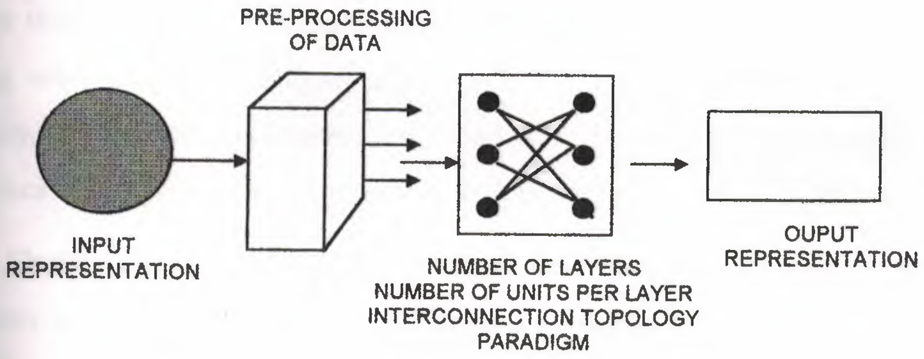


FIGURE 1.3
DESIGN CHOICES FOR A NEURAL NETWORK APPLICATION.

In addition, the researchers must design how those patterns are to represent the needed information (the representation scheme). For example, in an image classification problem, one could input the image pixel by pixel, or one could use a preprocessing technique such as a Fourier transform before the image is presented to the network. The output of the network then might have one processing unit assigned to represent each image classification, or, alternatively, a combination of several output units might represent each specific image classification. [DoyHoff P15]

Next, internal design choices must be made - including topology and the size of the network. The number of processing units is specified along with the specific interconnections that the network is to have. Processing units are usually organized into distinct layers, which are either fully or partially interconnected. [DoyHoff P16]

There are additional choices for the dynamic activity of the processing units. A variety of neural net paradigms are available; these differ in the specifics of the processing done at each unit and in how there internal parameters are adjusted. Each paradigm dictates how the readjustment of parameters takes place. This adjustment results in *learning* by the network. [DoyHoff P16]

Next there are internal parameters that must be "tuned" to optimize the neural net design. One such parameter is the learning rate from the back-error propagation paradigm. The value of this parameter influences the rate of learning by the network, and may possibly influence how successfully the network learns. There are experiments that indicate that learning occurs more successfully if this parameter is decreased during a learning session. Some paradigms utilize more than one parameter that must be tuned. Typically, the network parameters are tuned with the help of experimental results and experience on the specific applications problem under study. [DoyHoff P16]

Finally, the selection of the training data presented to the Neural network influences whether or not the network *learns* a particular task. Like a child, how well a network will learn depends on the examples presented. A good set of examples, which illustrate the task to be learned well, is necessary for the desired learning to take place; a poor set of examples will result in poor learning on the part of the network. The set of training examples must also reflect the variability in the patterns that the network will encounter after training. [DoyHoff P16]

Although a variety of Neural network paradigms have already been established, there are many variations which are currently being researched. Typically these variations add more complexity to gain more capabilities. Examples of additional structures under investigation include the incorporation of delay components, the use of sparse interconnections, and the inclusion of interaction between different interconnections. More than one neural net may be combined with outputs of some networks becoming the inputs of others. Such combined systems sometimes provide improved performance and faster processing times. [DoyHoff P16]

IMPLEMENTATION OF NEURAL NETWORKS

Implementation of Neural networks come in many forms. The most widely used implementations of Neural networks today are software simulators, computer programs that simulate the operation of Neural network. Such simulation might be done on a Von Neumann machine. The speed of the simulation depends upon the speed of the hardware upon which the simulation is executed. A variety of accelerator boards is available, vector processors, and other parallel processors may be used. [DoyHoff P17]

Simulation is key to the development and deployment of neural network technology. With a simulator, one can establish most of the design choices in a Neural network system. The choice of inputs and outputs can be tested as well as the capabilities of the particular paradigm used. Realistic training sets can be tested in simulation mode. [DoyHoff P17]

Implementation of Neural networks is not limited to computer simulation, however. An implementation can be an individual calculating the changing parameters of the network using paper and pencil. Another implementation would be a collection of people, each one acting as a processing unit, using a hand-held calculator. Although these implementations are not fast enough to be effective for applications, they are nevertheless methods for simulating a parallel computing structure based on Neural network architectures. [DoyHoff P17]

Because the precursors of today's Neural networks were built during the same period that the digital computer was being designed, digital computer simulation was not yet available. Neural networks were then made with electrical and electronic components, including resistors and motor driven clutches. Even though these designs appeared promising, the development of the digital computer soon dominated the field, and Neural networks were developed further using simulation. [DoyHoff P17]

One challenge to Neural network applications is that they require more computational power then readily available computers have, and the trade off in sizing up such a network are sometimes not apparent from a small-scale simulation. The performance of the Neural network must be tested using the network the same size as that to be used in the application. [DoyHoff P17]

The response of an artificial neural net simulation may be accelerated through the use of specialized hardware. Such hardware may be designed using analog computing technology or a combination of analog and digital. Macroscopic electronic components may be used, or the circuits may be fabricated using the semi-conductor devices. Development of such specialized hardware is underway, but there are many problems yet to be solved. Such technological advances as custom logic chips and logic-enhanced memory chips are being considered for Neural network implementations. [DoyHoff P18]

No discussion of implementations would be complete without mention of the original Neural networks-biological Neural nervous systems. These systems provided the first implementations of the Neural network architecture. Developed through billion of years of evolution, they use the substances available to living systems for learning and adaptation. Many details of their learning and information processing methods are still not known. However, there is some resemblance to the way that synthetic neural networks operate, although vast differences still remain. Much of what is known about biological neurons is not included in today's computational Neural networks. [DoyHoff P18]

CHAPTER 2

TIME SERIES AND FORECASTING IN ARTIFICIAL NEURAL NETWORKS

FORECASTING

INTRODUCTION

Forecasting is a key element of management decision making. This is not surprising, since the ultimate effectiveness of any decision depends upon a sequence of events following the decision. The ability to predict the uncontrollable aspects of these events prior to making the decision should permit an improved choice over that which would otherwise be made. For this reason, management systems for planning and controlling the operations of an organization typically contain a forecasting function. The following are examples are situations where forecasts are useful:

- ***Inventory Management.*** In controlling inventories of purchased spare parts at an aircraft maintenance facility, it is necessary to have an estimate of the usage rate for each part in order to determine procurement quantities. In addition, an estimate of the variability of forecast error over the procurement lead time is required to establish reorder points.
- ***Production Planning.***. To plan the manufacturing of a product line, it may be necessary to have a forecast of unit sales for each item by delivery period for a number of months in the future. These forecast for finished products can then be converted into requirements for semi finished products, components, materials, labor, and so on, so that the entire manufacturing system can be scheduled.
- ***Financial Planning.*** A financial manager has concern about the pattern of cash flow his or her company will experience over time. The manager may wish a prediction of cash flow broken down by type and time period for a number of future time periods as an aid in making current decisions.
- ***Staff Scheduling.*** The manager of a mail processing facility of the United States Postal Service needs a forecast of the hourly volume and mix of mail to be processed in order to schedule staff and equipment efficiently.
- ***Facilities Planning.*** Decisions about new facilities generally require a long-range forecast of the activities using the facilities. This is important in the design of the facility, as well as for justification of the investment required.

- ***Process Control.*** Forecasting also can be an important part of a process control system. By monitoring key process variables and using them to predict the future behavior of the process, it may be possible to determine the optimal time and extent of control action. For example, a chemical processing unit may become less efficient as hours of continuous operation increase. Forecasting the performance of the unit will be useful in planning the shutdown time and overhaul schedule.

From these examples and others that easily come to mind, we see that a forecast is a prediction of future events. The purpose of forecasting is to reduce the risk in decision making. Forecasts are usually wrong, but the magnitude of the forecasting errors experienced will depend upon the forecasting system used. By devoting more resources to forecasting, we should be able to improve our forecasting accuracy and thereby eliminate some of the losses resulting from uncertainty in the decision-making process. [Montgomery, Johnson, P1]

Because forecasting can never completely eliminate risk, it is necessary that the decision process explicitly consider the uncertainty remaining subsequent to the forecast. Often the decision is related conceptually to the forecast by

ACTUAL DECISION =DECISION ASSUMING FORECAST IS CORRECT + ALLOWANCE FOR FORECAST ERROR

This implies that the forecasting system should provide a description of forecast error as well as a forecast. Ideally the forecasting process should result in an estimate of the probability distribution of the variable being predicted. This permits risk to be objectively incorporated into the decision-making process. [Montgomery, Johnson, P3-4]

The forecast is not an end in itself; rather it is a means to an end. The forecasting system is a part of a larger management system and, as a subsystem, interacts with other components of the total system to determine overall performance. [Montgomery, Johnson, P4]

FORECASTING METHODS

Methods for generating forecasts can be broadly classified as qualitative or quantitative, depending upon the extent to which mathematical and statistical methods are used. Qualitative procedures involve subjective estimation through the opinions of experts. There are usually formal procedures for obtaining predictions in this manner, ranging from consolidation of the estimates of sales personnel to the use of Delphi-type methods to obtain a consensus of opinion from a panel of forecasters. These procedures may rely in part or marketing tests, customer surveys, sales force estimates, and historical data, but the process by which the information is used to obtain a forecast is subjective. [Montgomery, Johnson, P8]

On the other hand, a statistical forecasting procedures explicitly defines how the forecast is determined. The logic is clearly stated and the operations are mathematical. The method involve examination of historical data to determine the underlying process generating the variable and, assuming that the process is stable, use of this knowledge to extrapolate the process in the future. Two basic types of models are used:

Time Series: A time series is a time ordered sequence of observations (realizations) of a variable. Time series analysis uses only the time series history of the variable being forecasted in order to develop a model for predicting future values. Thus, if examination of past monthly sales of replacement tires for automobiles revealed a linear growth, a linear trend model might be chosen to represent the process and the appropriate slope and intercept estimated from historical data. Forecasts would be made by extrapolating the fitted model, as illustrated in Fig. 2.1

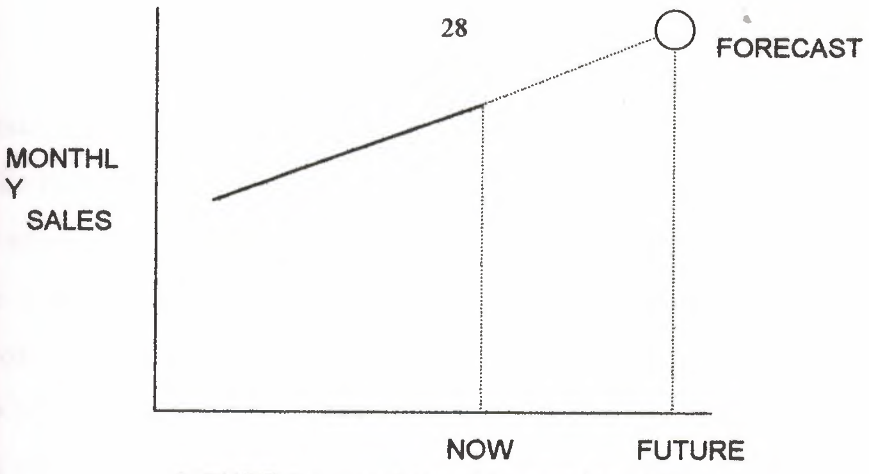


FIGURE 2.1: Linear-trend time series forecast

Casual Models: Casual Models exploit the relation ship between the time series of interest and one or more other time series. If these other variables are correlated with the variables of interest and if there appears to be some cause for this correlation, a statistical model describing this relationship can be constructed. Then, knowing the values of correlated variables, one can use the model to obtain the forecast. Of the dependent variable. For example, analysis might reveal a strong correlation between monthly sales of replacement tires and monthly sales of new automobiles 15 months before. Then information about new car sales 14 months ago would be useful in predicting replacement tire sales next month. The concept is illustrated in Fig. 2.2.

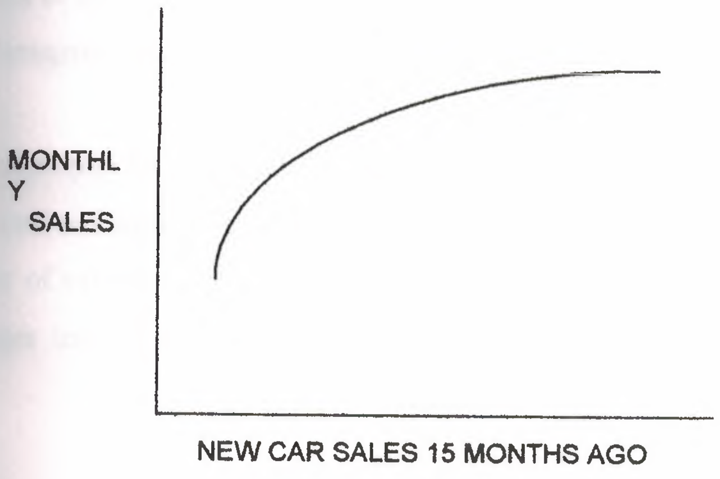


FIGURE 2.2: Causal Model

An obvious limitation to the use of causal models is the requirement that the independent variables be known at the time the forecast is made. The fact that tire sales are correlated with new car sales 15 months previous is not useful in forecasting tire sales 18 months in the future. Similarly, the knowledge that the tire sales are correlated with current gasoline prices is of little value, since we would know exactly the gasoline price in any future month for which we wished to forecast tire sales. Another limitation to the use of causal models is the large amount of computation and data handling compared with certain forms of time series models.

Actually, forecasting system often use a combination of quantitative and qualitative methods. The statistical methods are used to routinely analyze historical data and prepare a forecast. This lends objectivity to the system and results in effective organization of the information content of historical data. Then statistical forecast then becomes an input to a subjective evaluation by informed managers, who may their perception of future.

The selection of appropriate forecasting models is influenced by the following factors, most of which were discussed in the previous section:

1. Form of forecast required
2. Forecast horizon, period, and interval
3. Data availability
4. Accuracy required
5. Behavior of process being forecast (demand pattern)
6. Cost of development, installation, and operation
7. Ease of operation
8. Management comprehension and cooperation.

Computers play an important role in modern forecasting systems. They make it possible to store, retrieve, aggregate, desegregate, and otherwise manage time series data for a large number of variables. Complex statistical analysis is done easily. Many statistical software packages include forecasting modules. Also available are special purpose forecasting system software with powerful data management, analysis, and forecasting features. [Montgomery, Johnson, P8-11]

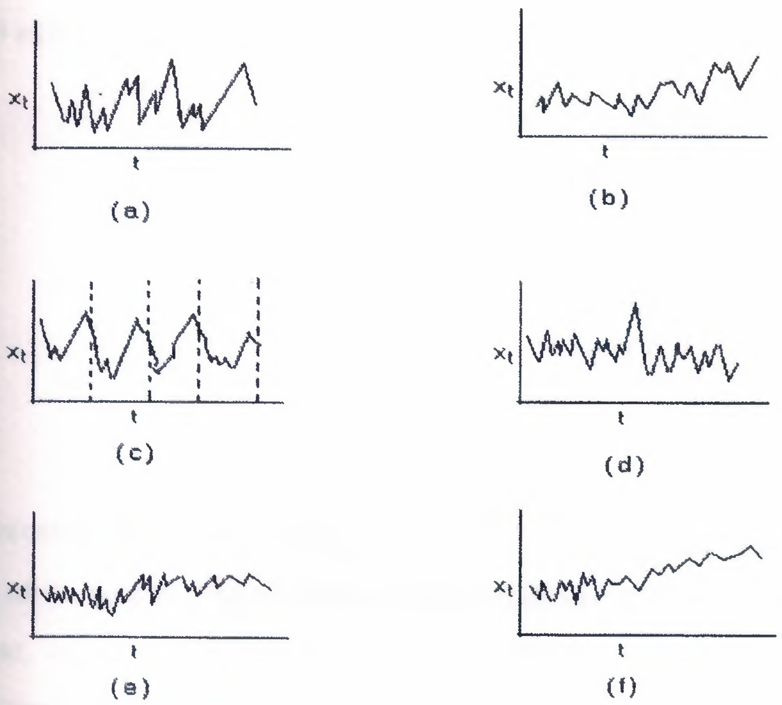
TIME SERIES MODELS

In this paper we will concentrate on forecasting using the time series analysis and models. Therefore it is necessary to explain the time series analysis in detail.

Characteristic of time series

For our purposes a time series is a sequence of observations on a variable of interest. The variable is observed at a discrete time points, usually equally spaced. Time series analysis involves describing the process or phenomena that the sequence. To forecast time series, it is necessary to represent the behavior of the process by a mathematical model that can be a good representation of the observations in any local segment of time close to the present. We usually do not require the model to represent very old observations, as they probably are not characteristic of the present, or observations far into the future, beyond the lead time over which the forecast is made. Once a valid model for the time series process has been established, an appropriate forecasting technique can be developed. [Montgomery, Johnson, P11]

Several characteristic patterns of time series are shown in fig. 2.3, where x_t is the observation for period *t*. In fig. 2.3a, the process remains at a constant level over time, with variation from period to period due to random causes. Pattern (*b*) illustrated a trend in then level of the process, so the variation from one period to the next is attributable to a trend in addition to random variation. In ©, the process level is assumed to vary cyclically over time, as in the case of a seasonal product. Seasonal variation can be attributed to some cause, such as weather, (e.g., the demand for soft drinks), institutions (e.g., Christmas cards), or policy (e.g., end-of quarter accounting) Most time series models for forecasting are developed to represent these patterns: constant, trend, periodic (cyclical), or a combination of the three. [Montgomery, Johnson, P11]



In addition, there are patterns resulting from a change in the underlying process. A transient, or impulse, pattern is illustrated by (d). For one period the process operated at a higher level before reverting to the original level. An example would be a temporary increase in sales caused by a strike at a competitor's plant. In (e), the change to a new level is permanent, and we refer to it as a step change. This could be caused by the acquisition of a new customer, for example. Finally, pattern (f) shows a process which has been operating at a constant level suddenly experiencing a trend. Since these three patterns of change are common in practice, we desire that our forecasting system identify permanent changes and adjust the forecasting model to track the new process. At the same time, we wish our forecasting system recognize random variations and transient changes and not react to these phenomena.

In forecasting demand for a product, we may need to use different forecasting models during various stages in the product's life cycle. For example, fig. 2.3 illustrated a life cycle having three distinct phases. During the growth phase, following introduction of the product, we might represent the process by a trend model, possibly with both linear and quadratic components. Once demand has leveled off, it would be desirable to switch to a constant-process model. During the final phase, when sales are declining, a trend model would again be appropriate. [Montgomery, Johnson, P11]

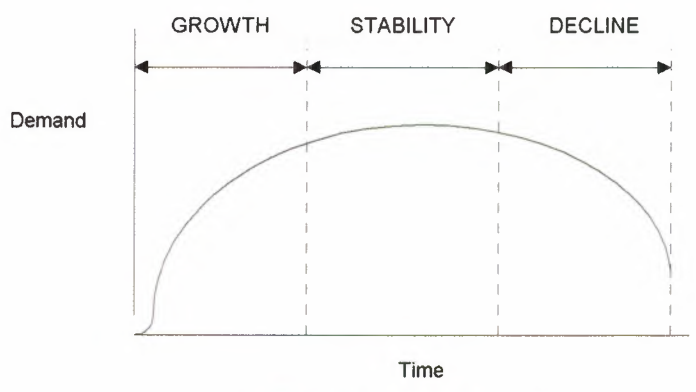


FIG. 2.4 Product life cycle

Representation of Time series

Many of the models used to represent time series are algebraic or transcendental functions of time, or some composite model that combines both algebraic and transcendental components. For example, if the observations are random samples from some probability distribution, and if the mean of that distribution does not change with time, then we may use the constant model. [Montgomery, Johnson, P13]

$$x_t = b + \in_t \qquad (2.1)$$

Where x_t is the demand in period t, b is the unknown process mean, and $\in_t$ is the random component, sometimes called the "noise" in the process. The random component has an excepted value of zero, and we usually assume that its variance is constant; that is, $E(\in_t)=0$ and $V(\in_t)=\sigma^{-2}_{\in}$. Note that this is equivalent to saying that x_t is a random variable with mean b and variance $\sigma^{-2}_{\in}$. Equation (2.1) is the appropriate model for the process illustrated in fig. 2.3a.

To represent the process of fig. 2.3*b*, we might assume that the mean of the process changes linearly with time and use the linear trend model

$$x_t = b_1 + b_2 t + \epsilon_t \qquad (2.2)$$

where $b_1$and b_2 are constants. Note that the slope b_2 represents the change in the average level of demand from one period to the next. Equation (2.3) gives a quadratic trend model:

$$x_t = b_1 + b_2 t + b_3 t^2 + \epsilon_t \qquad (2.3)$$

Cyclical variation may be accounted for by introducing transcendental terms into the model; for example,

$$x_t = b_1 + b_2 \sin(2\Pi t/12) + b_3 \cos(2\Pi t/12) + \epsilon_t \quad (2.4)$$

which would account for a cycle repeating every 12 periods. The models described above are of the following general form:

$$x_t = b_1 z_1(t) + b_2 z_2(t) + \ldots\ldots + b_k z_k(t) + \epsilon_t \qquad (2.5)$$

where the $\{b_i\}$ are parameters, the $\{Z_i(t)\}$ are mathematical functions of t, and ϵ_t is the random component. Thus for example, in equation (2.2), $Z_1(t) = 1$ and $Z_2(t) = t$. Note that this modeling approach represents the expected value of the process as a mathematical function of t.

Often it is desirable to define the origin of time as the end of the most recent period T. Then the model for the observation in period $T+\tau$ is

$$X_{T+\tau} = a_1(T)z_1(\tau) + a_2(T)z_2(\tau) + \ldots + a_k(T)z_k(\tau) + \epsilon_{T+\tau} \quad (2.6)$$

where the coefficients are now denoted by $\{a_i(T)\}$ to indicate that they are based on the current time origin T, and thereby distinguish them from the original-origin coefficients $\{b_i\}$.

Always keeping the origin of time on a current basis greatly facilitates the operation of a forecasting system. One simple technique in model selection is to plot historical data and look for patterns. As in any statistical data analysis procedure, graphic methods can be very useful in forecasting. Since the model should represent the near future for forecasting purposes, we usually judge its effectiveness by how well it describes the recent past. [Montgomery, Johnson, P14]

Forecasting with time series models

Time series forecasting consists of estimating the unknown parameters in the appropriate model and using these estimates, projecting the model into the future to obtain a forecast. For example, let $\boldsymbol{b_1}$ and $\boldsymbol{b_2}$ be estimates of the unknown parameters b_1 and b_2 in equation 2.2. If we currently are at the end of period T, the forecast of the expected value of the observation in some future period $T+\tau$ would be

$$\boldsymbol{x_{T+\tau}(T) = b_1 + b_2(T+\tau)} \qquad (2.7)$$

Thus the forecast simply projects the estimate of the trend component, $\boldsymbol{b_2}$, τ, periods into the future. This is illustrated in figure 2.5. [Montgomery, Johnson, P14]

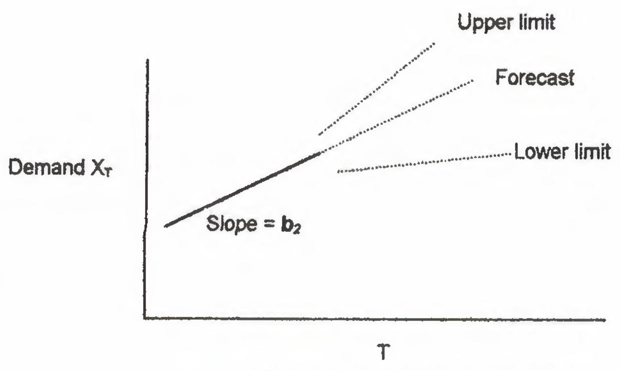


Figure 2.5: Forecasts and Prediction Intervals.

The forecast given by equation 2.7 is for a single period $T + \tau$. We may wish to forecast the sum of the observation in periods T+1, T+2,...,T+L. To obtain the *cumulative forecast,* we add the period forecasts as follows:

$$X_L(T) = \Sigma x_{T+\tau}(T) \qquad (2.8)$$

Cumulative forecasts are often required to predict total requirements over a lead time.

There are a variety of techniques for estimating the unknown parameters of time series models of equation 2.5 is linear in the unknown coefficients b_1, b_2, ...,b_k, and therefore conventional least squares methods can be used to obtain parameter estimates from historical data. In figure 2.5 , where the upper and lower prediction limits are determined so that there is a specified probability of their containing the actual observation.

Performance Criteria

There are a number of measures that can be used to evaluate the effectiveness of a forecasting system. Among the more important are forecasting accuracy, system cost, utility of output, and stability and responsiveness abilities. [Montgomery, Johnson, P16]

The accuracy of a forecasting method is determined by analyzing forecast errors experienced. If x_t is the actual observation in period t and $\mathbf{x}_t$ is the forecast for that period made at some prior time, the forecast error for period t is

$$e_t = x_t - \mathbf{x}_t \qquad (2.9)$$

For a given process and forecasting method, the forecast error is considered a random variable having mean $E(e)$ and variance σ_e^2. If the forecast is *unbiased,* $E(e) = 0$. While an unbiased forecast is desirable, it usually is more important that large forecast errors are rarely obtained. Hence a quantity such as the expected absolute error

$$E\,[|e_t|] = E\,[|x_t - \mathbf{x_t}|] \qquad (2.10)$$

or the expected squared error

$$E\,[e_t^2] = E\,[(x_t - \mathbf{x_t})^2] \qquad (2.11)$$

is commonly used as a measure of forecast accuracy. Note that the expected squared error, usually called the *mean squared error,* is equal to σ_e^2 if the forecast is unbiased.

In analyzing the accuracy of an installed forecasting method, is common to employ a *tracking signal test* each period. The purpose is to determine if the forecast is unbiased. The tracking signal is a statistic computed by dividing an estimate of excepted forecast error by a measure of the variability of forecast error, such as an estimate of the mean absolute deviation of forecast error. If the forecasting system yields unbiased estimates, the tracking signal should be near zero. Should the tracking signal deviate from zero by more than a prescribed amount, an investigation is made to determine if the forecasting model should be modified in order to better represent the time series process, which may have experienced a change such as that shown in fig.2.6., Note that this form of analysis can be applied to a statistical forecast, a judgmental forecast, or a combination of the two. [Montgomery, Johnson, P16]

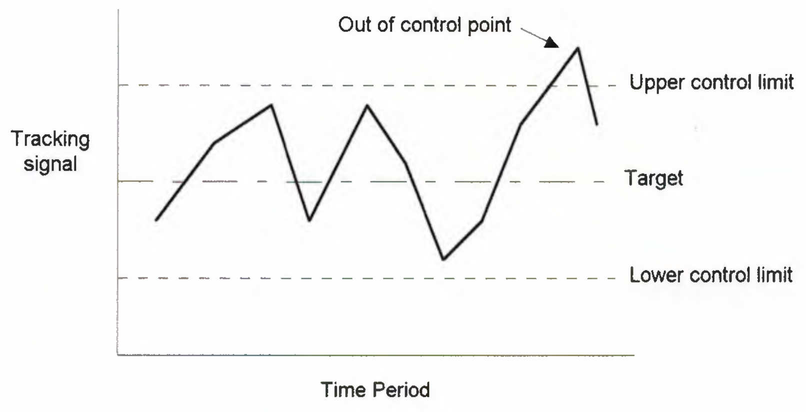


Fig 2.6: Forecast Control

Naturally cost is an important consideration in evaluating and comparing forecasting methods. There are one-time costs for developing and installing the system and periodic costs for operating it. With regard to operating costs, alternative forecasting procedures may differ, widely in the cost of data acquisition, the efficiency of computation, and the level of activity required to maintain the system.

The utility of the forecast in improving management decisions will depend upon the timeliness and form of the forecast, as well as its accuracy. Benefits should be measured with regard to the management system as a whole. Forecasting is only one component of this total system. The objective is to reach good decisions, and usually this can be achieved with less than perfect forecasts. [Montgomery, Johnson, P17]

We may also wish to compare forecasting methods on the basis of their response to permanent changes in the time series process and their response to permanent changes in the time series process and their stability in the presence of random variation and transient changes, This can be done through simulation and, for certain statistical methods, by mathematical analysis. [Montgomery, Johnson, P17]

System design considerations

We do not intend to give a comprehensive description of how one goes about developing and installing a forecasting system. The process is similar to that used for design of many other types of management information systems. Instead, we describe some considerations, that are important for forecasting systems. [Montgomery, Johnson, P17]

In choosing the forecasting interval, there is a trade-off between the risk of not identifying a change in the time series process and the costs of forecast revision. If we forecast infrequently, we may operate for a long period under plans based on an obsolete forecast. On the other hand, if we use a shorter interval, we more frequently incur not only the cost of making the forecast but also the cost of changing plans to conform to the new forecast. The appropriate forecast frequency depends upon the stability of the process, the consequences of using an obsolete forecast, and the cost of forecasting and re-planning. [Montgomery, Johnson, P17-18]

The data required by the forecasting system are subject to recording and transmission errors and therefore should be edited to detect obvious or likely mistakes. Small errors in magnitude will not be identifiable, but they usually will have little effect on the forecast. Larger errors can be more easily detected and corrected. Also the forecasting system should not respond to extraordinary or unusual observations. If we are forecasting product demand, any sales transaction that is identified as non-typical or extreme should, of course, affect inventory records, but should not be included in data used for forecasting. For example, suppose a manufacturer who supplies a number of distributors acquires a new customer. The initial orders, since the customer is at first establishing inventory. [Montgomery, Johnson, P18]

Simulation is useful technique for evaluation alternative forecasting methods. This can be done retrospectively using historical data. For each method, one starts at some prior time point and simulates forecasting period by period up to the present. Measures of forecast error can then be compared among methods. If the future is excepted to differ from the past, a pseudo-history can be created based upon subjective expectations of the future nature of the time series and used in the simulation. Simulation is also useful in determining parameters of forecasting techniques, such as the best smoothing methods. [Montgomery, Johnson, P18]

It is convenient to think of the two primary functions of a forecasting system as *forecasting generation* and *forecast control.* Forecast generation involves acquiring data to revise the forecasting model, producing a statistical forecast, introducing management judgment, and presenting the results to the user of the forecast. Forecast control involves monitoring the forecasting process to detect out-of-control conditions and identify opportunties for improving forecasting performance. An essential component of the control function is the tracking signal test described in the previous section. Items that exhibit out-of-control tracking signals can be singled out for special attention by managers, and efforts can be directed toward modifying their forecasting models, if necessary. [Montgomery, Johnson, P18]

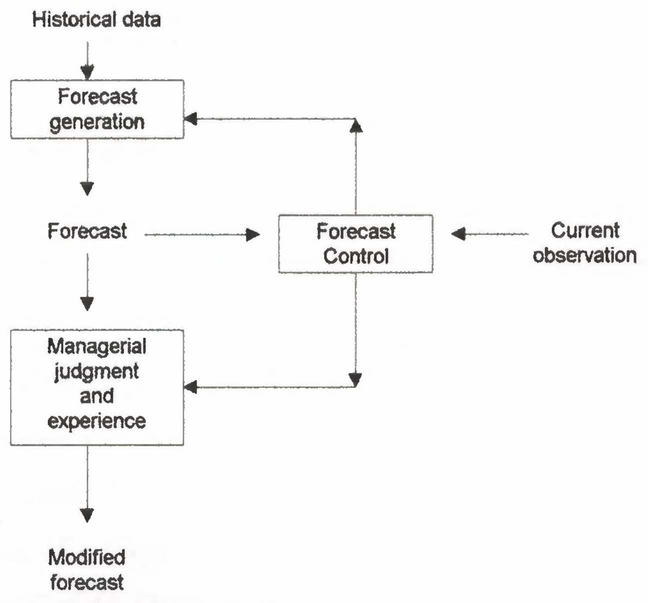


FIG 2.7 The forecasting system

Also the forecast control function should involve periodically summarizing forecasting performance and presenting the results to appropriate management. This feedback should encourage improvement in both the quantitative and qualitative aspects of the system. The relationship between forecast generation and forecast control is shown in fig.2.7.,

THE TIME SERIES PROBLEM

Predictability is fundamental to the modern scientific view of nature. When we write down Newton's law to calculate the motion of a projectile or a planet, we are implicitly assuming that the motion of such a system is predictable. It is the expectation that we can make meaningful predictions that drives us to seek underlying principles to explain the behavior of the systems we observe. In control engineering for example, the goal is often to combine measurements on a system, with some sets of fundamental rules to predict and control the systems behavior. In the time series problem we would like to use a series of measurements of a single observable as a function of time to predict what values future measurement will yield. [Vemuri, Rogers P1]

Many examples of time series are important in engineering and science. One of the best-studied time series is electric power demand. The ability to predict the demand placed on an electric power supply enables a system manager to make effective decisions about consumption of resources. Meteorologists have spent years studying various techniques for forecasting the weather, and although the full problem is inherently three-dimensional, some weather phenomena can be usefully studied as one carries with it major implications for how investing and securities trading are carried out. In this age of computerized trading, fast, effective analysis and forecasting strategies are highly sought after in the financial markets. Chemical engineers have studied the chaotic behavior of some chemical reactions as a time series problem in order to improve control over the rates at which these processes proceed. There are many important time series in medicine. For example, the white blood cell count of a cancer patient must be monitored and controlled. Decisions regarding drug dosages for such a patient can be greatly aided by predictions of the white blood cell count time series. Many other chemical relationships in the body, such as the blood glucose and insulin concentrations, can also be studied as time series. In addition, the EEG and ECG time series are of great interest. [Vemuri, Rogers P1]

Time series themselves exhibit reasonably well-understood behaviors. Often, as is the case for the price of a stock, a time series is composed of a long-term trend plus various periodic and random components. Some periodic components, such as a cyclic variation in the price of grain, are related to the seasons of the year, or they can be related to some other periodic phenomenon, such as a limit cycle. Linear and periodic components are usually easy to model and remove from the time series. One is then left time series forecasting problem. [Vemuri, Rogers P1]

The apparently random component of a time series usually falls into one of two categories. In the first case, the apparently random component it truly random; that is, the measurements are drawn from some underlying probability distribution. In this case, the random component can be characterized by a statistical distribution function or by the statistical moments of the data: mean, variance, skew, kurtosis, and so on. To a large extent, short-time-scale variations of stock prices are of this nature, as is the count rate in a Geiger counter placed near a radioactive isotope. In this category of time series, the simple statistical description of the system might be improved if the time series data are correlated on the time scale of interest. The level of water in a river can exhibit such behavior. The water level may fluctuate on short time scales, but measurements made within a single day will cluster around some mean that varies from day to day. Such correlations allow more precise predictions of future values and the excepted derivations from these predictions. [Vemuri, Rogers P1]

The second class of apparently random behavior in time series is not random at all, but rather, chaotic. A chaotic time series is characterized by values that appear to be randomly distributed and non periodic but are actually the result of a completely deterministic process. The deterministic behavior in a chaotic time series is usually due to underlying nonlinear dynamics. [Vemuri, Rogers P1]

The behavior of a dynamical system can be described in terms of its trajectory in phase space. For a system whose dynamics are a function of one variable (x), the phase space is the x' - x plane, often called the phase plane. The set of all values of x' and x taken by the system from a non-self intersecting trajectory in this plane. Figure 2.7 shows the phase space trajectory for the one dimensional non-linear oscillator governed by the equation [Vemuri, Rogers P2]

$$x'' = -x - \tfrac{1}{2}(x')^2 \qquad (2.12)$$

The trajectories of non-linear systems in phase space are generally constrained to move on surfaces that have significantly fewer dimensions then the full phase space of the system. A two dimensional dynamical system (for example, one that can move in two dimensions x and y) would have a four dimensional phase space, but might actually only move on the surface of a sphere inscribed in the four-dimensional phase space. Constraints such as this are the results of the conservation laws that severely limits the types of behavior the system can exhibit. For instance, the total energy of an isolated dynamical system can never increase, since energy is a conserved quantity. In a dynamical system without dissipation, the trajectories of the system in phase space are a set of nested closed curves. In a dissipative non-linear system, all initial conditions lead to trajectories that either lie on a single surface or converge to individual points in phase space. The set of these surfaces and points in phase space, to which all possible trajectories of the system converge, is called the attractor of the system. The attractor of a chaotic system has non-integral, or fractal, dimension and is called a strange attractor. [Vemuri, Rogers P2]

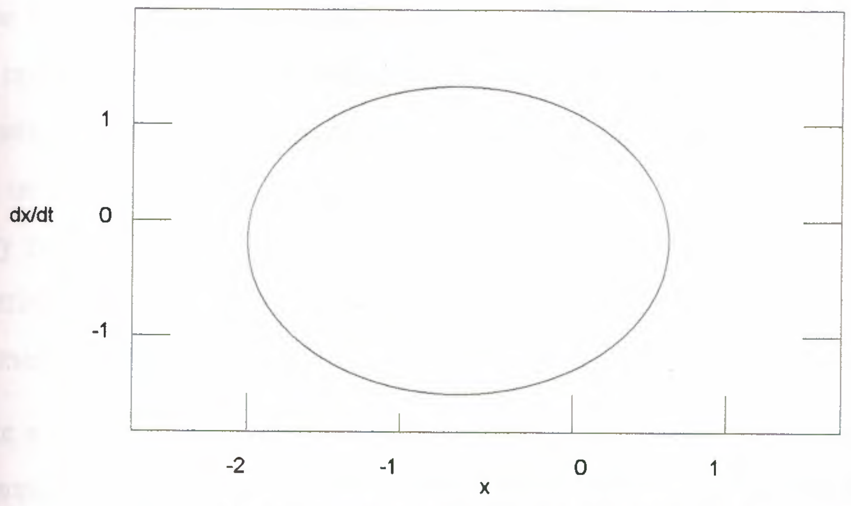


Figure 2.7 Trajectory in the phase plane of non-linear oscillator described by the above equation.

The importance of the strange attractor to the forecasting of a chaotic time series is twofold. First, its structure determines a theoretical limit to how far into the future the time series can be predicted. On the strange attractor surface, nearby trajectories diverge exponentially from one another, implying that any small error in a prediction of a chaotic time series will grow exponentially. The result is that long-term predictions are impossible. There is a natural time scale associated with this exponential growth of errors, which is specific to the type of chaotic system under consideration. This divergence is quantified by Liapunov exponent of the system. Since chaotic time series are deterministic, short term predictions of them can be made, as long as the length of the prediction is shorter then this error growth time. Second, the shape of the strange attractor determines how an Artificial Neural network predicts a chaotic time series. [Vemuri, Rogers P2]

Now that the desirability of modeling time series has been demonstrated, one might ask, "How do we go about making predictions of time series?" Ideally, we would like to use past data to construct a set of basic rules, like Newton's laws, that can be used to make predictions under very general circumstances. Unfortunately, this approach cannot always be carried out in practice. In some cases, the under lying principles are not known or are poorly understood because the system of interest is very complicated. This is the case with the stock market, in which relation ships various parameters are not known, and some of the relevant parameters, such as public opinion and world events, may not be accessible to us or quantifiable. Another problem with this approach is that often, even when the basic laws are known, direct solution of the equation is not possible without detailed information about initial values and boundary conditions. Fluid flow is an example of this situation because the hydrodynamic laws are known but their exact solutions requires us to specify the initial condition through out the volume of interest as well as boundary conditions along the entire surface, which may be quite complex. In practice, even barring the possibility of turbulence in the system, it is often impossible to make enough measurements to specify the system sufficiently. [Vemuri, Rogers P3]

In a second approach to time series analysis one avoids these problems by making the assumption that a well defined relationship exists between the past and future values of a single observable. In this phenomenological approach one seeks an approximate functional relationship between the past values and the future values one wishes to calculate. There are various ways to model this relationship. One can make recursive prescription for extrapolating the most recent data points based on the success of previous extrapolations. One can also parameterize the time dependencies of the various statistical moments and time derivatives of the time series of interest. Alternatively, one can try to find a single function that gives a future value of the observable as its output when some set of past observables is supplied as its input. This last model is implemented by Artificial Neural networks. [Vemuri, Rogers P3]

An Artificial Neural network is essentially a group of interconnected computing elements, or neurons. Typically, a neuron computes the sum of its inputs (which are either the outputs of other neurons or external inputs to the system) and passes this sum through a nonlinear function, such as sigmoid or hard threshold. Each neuron has only one output, but this output is multiplied by a weighting factor if it is to be used as a input to another neuron. The result is that there is a separate, adjustable weight parameter for each connection between neurons. [Vemuri, Rogers P3]

Neural networks typically exhibit two types of behavior. If no feedback loop connects neurons, the signal produced by an external input moves in only one direction and the output of the network is just the output of the last group of neurons in the network. In this case the network behaves mathematically like a non linear function of the inputs. This feed forward type of network is most often used in time series forecasting, with past time series values as the inputs and the desired future value as the output. The second type of network behavior is observed when there are feedback loops in the neuron connections. in this case the network behaves like a dynamical system, so the output of the neurons vary with the time. The neuron output can then oscillate, or settle down into steady state values, or, since the threshold function introduces nonlinearity into the system, they can become chaotic. [Vemuri, Rogers P3]

Since Neural networks are inherently non linear and often exhibit chaotic behavior, a great deal of research is being done on there dynamic behavior, especially that mimics real-world chaotic systems. such applications include modeling of chaotic processes in the brain and using chaotic dynamics to encode and decode speech signals for automated speech recognition. [Vemuri, Rogers P3]

In practice one usually subdivides the available time series data into two time segments: the training data set and the test data set. The data from the first segment is used to train the network. The network that results from the training process is then checked against the data from the test data set to determine whether the mapping performed by the network is a good representation of the time series and can therefore be expected to make reasonable predictions. This step can be accomplished in two ways. To test for short term prediction accuracy, the network is given the actual time series values from the test data set as its input and the resulting output is compared with the next time series point. This is done for every point in the test data set, and an error statistic is calculated. This error estimates the accuracy of short term predictions of the network. [Vemuri, Rogers P4]

The ability of the network to make longer term predictions can also be tested. To do this, the network is given an input vector from near the beginning of the test data set. The output of the network, which is the predicted future time series value, is then used as part of the next input vector. The output from the second prediction is likewise used as part of the third input vector. Continuing the process in this way, the network recursively propagates the time series forward in time to make a prediction many time steps ahead. The divergence of the prediction properties of Artificial Neural networks are very different so this is a useful test to perform before making predictions longer than one time step into the future. [Vemuri, Rogers P4]

Another practical issue in time series prediction is the size of the network. As in most Artificial Neural network research, no rule determines how many input and hidden-layer neurons to use. There are a few practical guidelines, however. In order for the network to be completely general mapping, there must be at least one hidden layer. This is a well known result of Artificial Neural networks research. The smallest network that can learn the training data is usually desirable, because it is more likely to be generalized to new time series data then a larger network. A network that is too large will tend to over fit the data points from the training data set without finding any underlying relationship between them. Such a network cannot predict the behavior of the time series for input vectors it hasn't seen before. A small network, with fewer parameters, is forced to find underlying relationship between inputs and outputs, and this mapping is more likely to be generalized to future time series behavior. On the other hand, a network that is too small may not have enough free parameters to learn the training data. In practice one often starts with a very small network and increases the number of hidden layer neurons until the desired prediction performance is obtained. [Vemuri, Rogers P4]

Similar considerations apply to the number of inputs to the network. The network must be given enough past data in each input vector to span the phase space of the time series, but too many inputs simply slows down the convergence of the training process and increase the number of possible undesirable mappings. Again, unless something is known about the dynamics generating the time series, experimentation is required to determine the optimum number of inputs. Performance on the test data set is usually the best guide available. [Vemuri, Rogers P4]

We have seen that the training process results in a Neural network that maps the past values of a time series into a future value of that time series. The formation of this mapping is the interpolation process, which Artificial Neural networks are particularly good at. In the training process the network effectively interpolates a surface between the input and output vectors. When the time series is dissipative dynamic system, the interpolation surface is generally an approximation to the attractor of the system or, in the case of chaotic system, the strange attractor. as an example consider the iterative map generated by the rule

$$x_{i+1} = ax_i(1 - x_i) \qquad (2.13)$$

which is chaotic when a $\geq$ 3.5699. When trained from data from this map, the network interpolates the parabolic shape of the locus of allowed points in the x_{i+1} -x_i and not the time series itself. This is an important point: Neural networks generally interpolates an underlying dynamic relationship, not the explicit time dependence of the time series. Many questions regarding exactly how Artificial Neural networks do this remain unanswered. For instance, no quantitative criteria exist for how a complicated dynamic system a particular Artificial Neural network can learn or how many past data are required to achieve some desired prediction accuracy. [Vemuri, Rogers P4]

PROCEDURES FOR FORECASTING

Technological forecasting is a subset of futures research. Futures research is an umbrella term which encompasses "any activity that improves understanding about the future consequences of present developments and choices" (Amara and Salanik, 1972, p. 415). Forecasting is:

- a statement about the future
- a probabilistic statement about the future
- a probabilistic, reasonably definite statement about the future
- a probabilistic, reasonably definite statement about the future, based upon an evaluation of alternative possibilities (p. 415)

Technological forecasting includes "all efforts to project technological capabilities and to predict the invention and spread of technological innovations" (Ascher, 1979, p. 165). Martino (1983) states that a technological forecast includes four elements: the time of the forecast or the future date when the forecast is to be realized, the technology being forecast, the characteristics of the technology or the functional capabilities of the technology, and a statement about probability.

Forecasting a technology is a difficult task "beset with hazards" (Ayers, 1969, p. 18). Some of these hazards include: "the uncertainty and unreliability of data, the complexity of 'real world' feedback interactions, the temptation of wishful or emotional thinking, the fatal attraction of ideology, [and] the dangers of forcing soft and somewhat pliable 'facts' into a preconceived pattern" (p. 18). To offset the inherent ambiguity and uncertainty of forecasting, technological forecasters have developed a set of methodologies to assist them in their endeavor.

In general, as a technology moves from the early stages of laboratory development to widespread acceptance in the marketplace, the forecasting methodologies that are most appropriate move from qualitative to quantitative techniques. Since technological forecasting is employed to predict long-term technological developments, the methods ized are generally qualitative. Listed below is a brief description and discussion of some of the major qualitative methods and techniques which have been developed to forecast technological developments.

Delphi

The Delphi procedure is designed for the systematic solicitation of expert opinion. There are three characteristics which distinguish it from interpersonal group interaction: anonymity, iteration with controlled feedback, and statistical group response (Martino, 1983).

While many variations of the technique have been offered since it was originally developed at the Rand Corporation in, the conventional Delphi study proceeds as follows. A questionnaire designed by a monitor team is sent to a select group of experts. After the responses are summarized, the results are sent back to the respondents who have the opportunity to re-evaluate their original answers, based upon the responses of the group. By incorporating a second and sometimes third round of questionnaires, the respondents have the opportunity to defend their original answers or change their position to agree with the majority of respondents.

The Delphi technique, therefore, is a method of obtaining what could be considered an intuitive consensus of group expert opinions. The accuracy of the forecast produced is limited by the quality of opinions provided by the experts, and it should be noted that some authors (such as, Challis and Wills, 1970 and Wise,1976) have questioned the accuracy of the opinions of specialists.

Trend Extrapolation

A forecast can be generated by "observing a change through time in the character of something and projecting or extrapolating that change into the future" (Cornish, 1977, p. 108). In making such a forecast, the focus is on the long-term trend, so short-term fluctuations are disregarded. Trend extrapolations require that the forecaster have an understanding of the factors which contributed to change in the past, and possess confidence in the notion that these factors will continue to influence developments in a similar fashion in the future (Schwarz, Svedin, Wittrock, 1982, p. 20).

One commonly employed approach to trend extrapolation involves the use of growth curves (Cornish, 1977, pp. 110-111). Growth curves are loosely based upon the notion that the growth of a technology can be charted in the same way organic growth can be charted. For example, the growth in height and weight of an individual can be charted, and will commonly display a pattern which indicates a leveling off around early adulthood. It is believed that the growth pattern of a technology can also be plotted and charted in a similar fashion. As an illustration, Martino describes how this particular technique can be utilized in charting and forecasting the growth in, and leveling off, of the number of cable television subscribers.

Regarding the accuracy of trend extrapolation as a forecasting technique, Ascher (1978) questions its "objectivity and reliability" (p. 183). Schnaars (1989) goes even further and admonishes forecasters to discount trend extrapolations. He notes that trends and patterns have no life of their own and are susceptible to sudden changes, and that focusing on trends alone "is often a search for the will-o'-the wisp" (p. 152). As an example of a misuse of trend extrapolation, he notes the actions taken by American electronics firms with regard to television manufacturing. Through the 1950s and the 1960s, television sets steadily grew larger. As American firms continued to make large, cabinet-based systems, Japanese firms began to concentrate on making portable sets. While the American firms acted on the belief that the existing trend toward larger sets would continue, the actual trend within the marketplace shifted toward a greater variability in size.

Historical Analogy

The use of analogy in forecasting involves a "systematic comparison of the technology to be forecast with some earlier technology that is believed to have been similar in all or most important respects" (Martino, 1983, p. 39). Forecasting by analogy is one of the simpler and more common ways to forecast the growth of a new technology, though as a method its accuracy has been questioned on several accounts. Schnaars

(1989) notes that the method has limited predictive value as "what happened before in an industry often blinds those already in the industry to developments that come from outside" (p. 153). In his study of home video forecasts, Klopfenstein (1985) found that many erroneous forecasts of videodisk players sales were based on comparison with the previous introduction of color television. He asserts that historical analogy can serve as a useful guide to forecasting a new technology, but great care must be taken in making the comparison. Martino (1983) asserts that when drawing an analogy, consideration must be given to the numerous dimensions which are known to have an effect on technological change (see Martino, 1983, pp. 40-49 for a discussion of these dimensions). The real challenge facing a forecaster, therefore, is the task of identifying a technological innovation which will truly serve as an accurate historical precedent upon which to base a forecast by analogy.

Scenarios

Each of the above forecasting methods has its own advantages and disadvantages. Therefore, in many cases, it is helpful to combine several methods and forecasts into one. Martino (1983) notes that scenario construction is an effective method for combining forecasts and forecasting methodologies into a holistic composite.

Cornish (1977) describes a scenario in simple terms: "it is simply a series of events that we imagine happening in the future." In other words, scenario writing is "making up stories about the future" (p. 11). Schwarz, Svedin, and Wittrock (1982) note that the term "scenario" has numerous meanings. It can be used as a description for "a hypothetical, likely or unlikely, development or situation; a development which is described as caused to some extent by the actions and reactions of various actors: a desirable or undesirable development or situation" (p. 28). Kahn and Wiener (1967) assert that it is a method which can be employed to focus attention on causal processes and crucial decision points.

Martino (1983) states that scenarios serve three basic purposes:

1) to display the interactions among several trends and events in order to provide a holistic picture of the future;

2) to help check the internal consistency of the set of forecasts on which they are based
3) to depict a future situation in a way readily understandable by the non-specialist in the subject area (p. 148).

While noting the unreliability of forecasting methods, Schnaars (1989) is a strong advocate of the use of scenarios. He notes that they do not pretend to predict the future but rather present a set of possible futures. Godet (1983) notes that since the value of a forecast is dependent upon the underlying assumptions, and that quite often several sets of assumptions can be offered upon which several scenarios can be constructed, no forecast should be published "without giving an indication of the estimated probability of the corresponding scenario" (p. 190).

By accounting for a range of possibilities, scenarios can be distinguished from the other methods listed above. They do not generate or present the same degree of specificity, and have even been described as an "alternative to forecasting" (Schnaars, 1989).

ADAPTIVE NETWORKS AND TIME SERIES PREDICTION

A dynamical System Perspective

One of the central roles of science in the study of naturally occurring phenomena is in forecasting, given knowledge about a system and its past behavior. Two basic methods by which such predictions are made may be classified into a model-based approach and a statistical approach. The first approach assumes that there is sufficient a priori information (for instance in the form of physical conservation laws) that a first-principles derivation may be made to construct an accurate model of the mechanism which is generating the observed processes. There are two difficulties with this approach. One is that it is not often possible to generate an accurate model since the underlying 'laws' may not be fully understood, as in stock-market forecasting. Secondly, even if an accurate model can be constructed, the specification of the current state from which the model can predict, may require much more information than is practically obtainable. In weather forecasting the model takes the form of a set of partial differential equations whose initial state requires functions to be continuously specified in three dimensions. Whereas in practice one may have a spatially non-uniform sparse sample of observations to specify the initial state. [Vemuri, Rogers P12]

The second approach attempts to analyze the sequence of observation the sequence of observations produced by the underlying mechanism directly. From the statistics or dynamics obtained from the observation sequence one hopes to be able to infer some knowledge about the future evolution of the observation sequence. The problem with this latter approach is that nature tends to produce very complicated, often irregular, chaotic behavior which is apparently the result of a self interaction of the system with a large (possibly infinite) number of degrees of freedom. Consistent with this viewpoint, contemporary forecasting theory has developed to assume that the observation sequence may be considered to be one specific realisation of a random process, where the randomness arises from the many independent degrees of freedom interacting linearly. However, the emerging view in dynamical systems theory is that apparently random behaviour may be generated by deterministic system with only a small number of degrees of freedom, but which interact *nonlinearly* to produce deterministic chaos. This reflects one of the underlying *generator* of the observation sequence, since most of contemporary physics and engineering relies on the superposition principle-an inherent assumption concerning a system's linearity. [Vemuri, Rogers P12]

Of course, certain classes of artificial 'neural' networks (particularly multilayer perceptron-like networks) may be considered as flexible nonlinear parameterised models where the parameters may be adapted according to the available data. Indeed, adaptive network techniques have been used recently with some success to predict the behaviour of chaotic time series -deterministic sequences whose second-order persistent statistics seem to indicate that they are random. This success stems from the ability of adaptive networks to produce an interpolation surface which approximates the actual nonlinear map which generated the data. Thus, adaptive networks may be applied to time series prediction, as long as the observed time series is generated by an underlying iterative mapping, if the mapping itself is 'smooth' enough to allow an interpolation surface to be constructed. [Vemuri, Rogers P12]

It is known that current network models perform well when operating as static pattern classifiers, and the reason for their effectiveness in this domain may be explained by exploiting relationship with traditional discriminate analysis. However, such network models do not manipulate dynamic information appropriately. The best methods for automatic continuous speech recognition are the established hidden Markov models, and not 'neural' networks. The encoding of time by adaptive networks, and how such networks should deal with temporal sequences is, perhaps, the major difference between the artificial connectionist models and real neural networks. This paper accepts the limited temporal repertoire of connectionist models, but emphasizes a class of problems where , by viewing them as models of dynamical systems, a broad response range is obtainable by a suitable choice of parameter values. Thus use of adaptive networks as forecasting models for time series prediction by exploiting the links between adaptive networks, dynamical systems theory and functional interpolation. [Vemuri, Rogers P12]

Dynamical systems preliminaries

It is sufficient to consider a dynamical system as specifying the evolution, or 'flow' of the state of a system in its phase space. We are only interested in finite dimensional phase spaces of dimension *d*, which determines the number of possible degrees of freedom available to the system. The state of a system at time t_i is given by a vector $x(t_i) \in \mathbf{R}^d$ and the evolution of this vector is determined, generically, by a nonlinear equation of motion, $dx/dt = f(x)$. For deterministic dynamical systems, the evolution trajectories, the solutions of the equation of motion, do not cross in the phase space. Although the system could potentially evolve in the full *d* dimensional phase space, it is usually the case, due to conservation laws, imposed restrictions or dissipative dynamics, that the flow of a system is asymptotically constrained to evolve on a much lower dimensional non-Euclidean sub-manifold M of dimension *m*. This asymptotic hyper-surface is called an *attractor*. If an attractor may be characterised by a non-integer dimension (a 'fractal' Hausdorff dimension), then the attractor is a *strange attractor* and evolution on this attractor is chaotic. That is, if two initial conditions are specified which are close together. Their subsequent evolution trajectories will diverge exponentially. Thus, unless the initial condition is known to infinite precision, it is impossible to be able to predict the state of the system at a future time. Since evolution on the hypersurface is a reduced-dimension representation the dynamics compared to the potential dimensionality of the original equations of motion, the asymptotic behavior should be specified by a model system with fewer degrees of freedom. It is usually, the case in physical systems that this is an enormous reduction in the degrees of freedom of the system, corresponding to self organization. [Vemuri, Rogers P13]

Unfortunately, this sub-manifold is non-Euclidean (only in a local region may the manifold be characterised by a Euclidean space of dimension *m*), and adaptive networks are only really useful when performing transformations between Euclidean spaces. However, theorems have been developed which allow arbitrary, smooth hypersurfaces to be 'embedded' into spaces which are diffeomorphic (the embedding transformation and its inverse are differentiable) to an equivalent-dimensional Euclidean space. In particular a smooth *m* dimensional manifold may be embedded in a Euclidean space of at most dimension $2m + 1$. This is the basis of phase space reconstruction techniques developed since 1980 when Takens and Packard discussed methods where the phase space information equivalent to the underlying dynamical system could be reconstructed from a single time series and derivatives obtained from the time series. Taken's approach was to effect a reconstruction using time-lagged vectors created from the time series, a technique known as the 'method-of-delays-, which was extended into a regularized analysis tool to deal whit real data. It is analogous to the window length employed in linear predictive analysis in digital signal processing. According to the method-of-delays, an n dimensional phase portrait can be obtained by constructing the sequence of vectors

$$x_i^T = (x_i, x_{i-1}, \ldots, x_{i-n+1}) \qquad (2.14)$$

from the discrete time series $x_1, x_2, x_3, \ldots$ The work of Takens showed that this construction is sufficient (but not necessary) to give an embedding of the underlying m dimensional manifold if $n \geq 2m+1$.From the point of view of networks, the number of input units is given by n, the size of the enclosing Euclidean space in which the dynamical system is embedded. [Vemuri, Rogers P13]

Functional interpolation preliminaries

One view on the operation of adaptive, feed forward layered networks such as the multilayer perceptron is that they perform well for certain tasks by exploiting their modeling flexibility to create an implicit interpolation surface in a high-dimensional space. Specifically, in mapping a finite set of P, n dimensional 'training' patterns to the corresponding n' dimensional 'target' patterns, s: $R^n \rightarrow R^n$ one may think of this map as being generated by a 'graph' $\Gamma \subset R^n \otimes R^{n'}$ (in the same way that an Ordnance Survey contour plot, s:$R^2 \rightarrow R$ mapping the surface to a high may be viewed as a landscape in three dimension). The input and target pattern pairs are points on this graph. The learning phase of adaptive network training corresponds to the optimization of a fitting procedure for Γ based on knowledge of the data-points. This is curve fitting in the generally high ıensional space $R^n \otimes R^{n'}$. Thus *generalization* becomes synonymous with *interpolation* ng the constrained surface which is the 'best' fit to Γ. The radial basis function network s introduced simply to make this point more explicit, but it also applies to networks :h as the multi-layer perceptron. It is clear that, by analog by curve fitting on one or two ıensions, one can create an interpolation surface which is guaranteed to pass through ery point in a finite training set provided that the model is of a sufficiently high order g. sufficient numbers of hidden units equivalent to a sufficient number of Fourier efficients). However, this is incorrect strategy for real data which is confused by rinsic and intrinsic noise effects and corresponds to over training a network. Often a ge amount of prior knowledge is required to allow a fitting surface to be produced ich is just smooth enough to fit the structure in the data, thus allowing good ıeralisation performance, without being over-complex to permit the fitting of noise on ɔ of the data. [Vemuri, Rogers P13]

nthetic problems

Specially we consider the problem of reconstructing the *generator* of data, where ɛ 'data' is a single time series obtained either from an iterated chaotic map or from a nlinear differential equation. The aim in both cases is to use deterministic networks to empt a reconstruction of the generator of the data, which may then be used to nthesize data with the characteristics of the true data. Since the time series may be aotic using the network to generate data will produce a flow in the phase which will pidly diverge from the actual flow from a given starting value. This emphasizes the point at networks do not interpolate the time series themselves. [Vemuri, Rogers P13]

eal problems

The illustrative examples considered so far have been 'ideal' in the sense that we ve had control over the time series. In particular, it was not necessary to consider effects ıe to real background extrinsic noise, intrinsic noise effects in the data (quantization rors-the data was generated with 32-bit floating-point arithmetic)or sampling effects. dditionally we had extra knowledge regarding the 'order' of the problem. In real physical situations we do not have this control. In speech wave form prediction where the speech was quantised to $\approx$ 14 bits, had a high signal-to-noise ratio and was over sampled with respect to the vowel sounds. Nevertheless, it was still difficult to obtain adequate network solutions to reproduce the wave forms (although relevant 'features' could be reproduced). The noise has the effect of forcing the actual low dimensional behavior back into a high dimensional space. This implies that the apparent 'order' of the data is higher than it needs to be, which complicates the task of finding an appropriate window size in which the evolution of the system may be captured. The significant advances in dynamical systems practice have been in devising regularized approaches to deal with noise problems to determine how to project the data into subspaces in which the variance of the noise is minimized. Currently there are no equivalent regularized schemes to deal with these problems using adaptive networks and this is an obvious area for future development. [Vemuri, Rogers P17]

In addition to problems due to limited control over data (such as reducing, but not eliminating noise), one is also often confronted with time series over which there can be no control at all. Typical examples in this category are economic and social forecasting where available prior knowledge for model-building is minimal. Even if a long history of samples is obtainable, it is likely that the generator of the data (if one exists) is not static but evolves on a slower time scale to the local variation of the data. Thus the significance of generated data depends upon its history. In such circumstances one wishes to be able to predict the likely evolution of a system based on a sparse, noisy, non-uniformly sampled set of data points where the reliability of each data point can be different. [Vemuri, Rogers P17]

FORECASTING TIME SERIES

Typically, the decision problem requires a forecast over a number of future periods. That is one must forecast over some *lead time* or *Planning horizon.* The length of the planning horizon depends upon the nature of the decision problem. These considerations also determine whether a forecast is required for each period in the planning horizon or whether a forecast of the total demand over all periods in the horizon will be adequate. The former is called a *period-by-period* forecast, and the latter, a *cumulative* forecast.[Montgomery, Johnson p165]

Forecasting yields a prediction about future events. If a forecast is made by extrapolating a time series model, assuming that, over the forecast horizon, the process will behave as it did in the recent past. Specifically, it is assumed that the model form is correct and that the true parameter values do not change. In that case, the estimates of model parameters computed from historical data would yield accurate forecasts. In most situations, these assumptions are not exactly correct; the underlying process is changing with time. Therefore, one could expect forecast accuracy to decrease as the planning lead time increases. A forecast of March sales, made at the end of February, will likely be more accurate then a forecast of October sales, also made at the end of February. [Montgomery, Johnson p.165-166]

The accuracy of a forecasting procedure can be quantitatively described by the variance of the forecast error. Even if the process does not change over time, the forecast will differ from the actual demand because of random variation in the demand process and errors in estimating the parameters in the model. even if the true parameter values were known, there would be forecast errors because of the random variation. Thus, the variance of the forecast error is the function of the variance of the demand process and the sampling variances of the statistics used to estimate model parameters. The forecast error also depends upon the forecast lead time τ. As a practical matter, it is always desirable to have an estimate of the forecast error variance in order to quantify the uncertainty, or risk, associated with the forecast. [Montgomery, Johnson, p.166]

Normally a time series model is used to provide an estimate of the expected value of the process at some future period. By adding or subtracting a multiple of the standard deviation of the forecast error, this point estimate can be converted into a *prediction interval*. In some cases, the probability that the actual time series realization will fall in the interval can be stated. Prediction intervals can be very useful in describing the uncertainty associated with a forecast. In some cases, it is desirable to forecast the probability distribution of demand directly. [Montgomery, Johnson, p.166]

Period and Cumulative Forecasts:

Assume the following time series model,

$$x_t = b_1 z_1(t) + b_1 z_1(t) + \ldots + b_k z_k(t) + \epsilon_t \qquad (2.15)$$

where the $\{b_i\}$ are constants, the $\{z_i(t)\}$ are mathematical functions of time t, and ϵ_t is the random component having mean 0 and variance σ_ϵ^2. At time T, we have estimated the coefficients $\{b_i\}$ from historical using, say, discounted least squares and have the following forecasting equation in terms of the original time origin

$$x_{T+\tau}'(T) = \Sigma^k_{i=1} b_i'(T) z_i(T+\tau) \qquad (2.16)$$

or, if a current time origin is used,

$$x_{T+\tau}'(T) = \Sigma^k_{i=1} a_i'(T) z_i(T+\tau) \qquad (2.17)$$

The above forecasting equation provides a point estimate of the expected demand in period $T+\tau$. If it is desired a period-by-period forecast over a forecast horizon L periods, then the forecasting equation is evaluated successively for $\tau = 1,2,\ldots,L$, to obtain

$$x_{T+1}',\ x_{T+2}',\ \ldots,\ x_{T+L}' \qquad (2.18)$$

The Cumulative demand over a horizon L periods, starting with period $T + 1$, will be denoted by $X_L(T)$ and is defined by

$$X_L(T) = \Sigma^{k}_{\tau=1} x_{T+\tau}' \qquad (2.19)$$

The cumulative forecast computed at time T is

$$X_L(T)' = \Sigma^{k}_{\tau=1} x_{T+\tau}'(T) \qquad (2.20)$$

Depending upon the nature of the $\{z(t)\}$, it may be possible to calculate the cumulative forecast directly , rather than first calculating each of the L individual forecasts and then adding. [Montgomery, Johnson, p.166]

FORECAST ERRORS

There are a number of possible definitions of the forecast error experienced in a period, depending upon the prior point in time at which the forecast was made. For example, the forecast error for October could be computed based on a forecast made at the end of September, or it might be calculated based upon a forecast made at the end of July. In the former case, one would be calculating the *one-period-ahead* forecast error, and in the latter, the *three-period-ahead* forecast error. Naturally, other possibilities exist for different forecast lead times. In general, the *τ-period-ahead* forecast computed for period T is the actual demand in period T less the forecast for period T made at the end of period T-τ, or

$$e_\tau(T) = x_T - x_T'(T-\tau) \qquad (2.21)$$

The circumstances of the particular decision problem for which the forecast is required will dictate the value, or values of the forecast lead time τ that are of interest in monitoring forecast performance. For statistical purposes, the analysis of forecast error will generally require computation of only $e_1(T)$, and for convenience, it is sometimes referred to as "the forecast error", without stating τ=1.

Mathematically, we can define the *L-period-cumulative forecast error,* computed at time T, as the total actual demand in periods T-L+1, T-L+2, ..., T, less the cumulative forecast for these periods made at time T-L,

$$\begin{aligned} E_L(T) &= X_L(T-L) - X_L'(T-L) \\ &= \Sigma_{\tau=1}{}^{L} e_\tau(T-L+\tau) \qquad (2.22) \end{aligned}$$

Variance of Period forecast errors:

From equation 2.21 we can observe that the τ-period forecast error is the difference of two random variables, the demand in period T and the forecast made τ periods before. These random variables are independent since the forecast is based upon the demand values prior to period T and we assume the demands are independent. Therefore the forecast error variance is the sum of the variance of the demand process, σ_{ϵ}^{2}, and the forecast error; that is,

$$Var[e_{\tau}(T)] = Var[x_T] + Var[x_T'(T-\tau)] \qquad (2.23)$$

The variance of the forecast is a function of the variances and covariances describing the uncertainty in using estimates of the model parameters in the forecasting equation. The sampling variances of the estimators are defined as

$$\mathrm{Var}[b_i'] \equiv E\{[b_i'-E(b_i')]\} \equiv V_{ii} \qquad (2.24)$$

and the covariance between b_i and b_j is defined as

$$Cov(b_i',b_j') = E\{[b_i'-E(b_i')][\,b_j'-E(b_j')]\} \equiv V_{ij} \qquad (2.25)$$

where E is the expected value operator.

If the forecasting equation is 2.16, then the variance of the forecast is

$$\mathrm{Var}[x_{T+\tau}'(T)] = \mathrm{Var}[\Sigma^{k}_{i=1} b_i'(T) z_i(T+\tau)] \qquad (2.26)$$

In matrix notation the above equation becomes

$$\mathrm{Var}[x_{T+\tau}'(T)] = \mathbf{z}'(T+\tau)\mathbf{V}\mathbf{z}(T+\tau) \qquad (2.27)$$

where $\mathbf{z}(T+\tau)$ is a k-component column vector of the independent variables evaluated at $T+\tau$, and $\mathbf{V}$ is the variance-covariance matrix having elements V_{ij}. If a current-origin forecasting equation is used then

$$\mathrm{Var}[x_{T+\tau}'(T)] = \mathbf{z}'(\tau)\mathbf{V}\mathbf{z}(\tau) \qquad (2.28)$$

where $\mathbf{V}$ now must be the variance-covariance matrix of the $\{a_i'(T)\}$.

Variance of cumulative forecast errors:

The cumulative forecast over an L-period horizon and the associated cumulative forecast error were defined by equation 2.21 and 2.22, respectively. From the latter equation, we see that the cumulative forecast error is the difference between the cumulative demand actually experienced and the cumulative forecast. Assuming that the period demands are mutually independent random variables, these two quantities are uncorrelated, so the variance of the forecast error is the sum of the variance of the cumulative demand and the variance of the forecast.

The results can be stated symbolically by assuming that we are at the end of period T and wish a cumulative forecast for the next L periods. The forecast is, for an original-original model,

$$X_L'(T) = \Sigma_{\tau=1}^{L} x_{T+\tau}'(T)$$

$$=\mathbf{b}'(T)\,\Sigma_{\tau=1}^{L}\,\mathbf{z}(T+\tau) \qquad (2.29)$$

where $\mathbf{z}'(T+\tau) = [z_1(T+\tau), z_2(T+\tau), \ldots, z_k(T+\tau)]$ is the vector of independent variables evaluated at $T+\tau$. The variance of the forecast is the following function of the variances of the $\{b_i'(T)\}$:

$$\mathrm{Var}[X_L'(T)] = [\Sigma_{\tau=1}^{L}\,\mathbf{z}(T+\tau)]'\,\mathbf{V}\,[\Sigma_{\tau=1}^{L}\,\mathbf{z}(T+\tau)] \qquad (2.30)$$

where **V** is the variance-covariance matrix having elements $V_{ij} = \mathrm{Cov}(b_i, b_j)$.

It should be observed that even though the forecast is the sum of L period forecasts, the forecast variance is not the sum of the period forecast variances. This is because all the period forecasts in the cumulative forecast are based upon the same **b'**(T) [or **a'**(T)] and therefore are correlated. The variance of the cumulative error in forecasting for periods $T+\tau, T+\tau+1, \ldots, T+L$ is

$$\mathrm{Var}[E_L(T+L)] = \mathrm{Var}[X_L(T)] + \mathrm{Var}[X_L'(T)]$$
$$=L\sigma_{\epsilon}^{2} + \mathrm{Var}\,[X_L'(T)] \qquad (2.31)$$

Estimation of Expected Forecast Error:

The forecasting procedures described before have involved:

1. choosing a time series model to represent the demand process,
2. estimating the parameters of this model, and
3. extrapolating the estimated model into the future.

We have assumed that the time series model is of the general form

$$x_t = \mu(t) + \epsilon_t \qquad (2.32)$$

where the expected value of x_t is $\mu(t)$, some mathematical function of t, and ϵ_t is a random component having mean 0 and variance σ_ϵ^2. Thus, in choosing a time series model, we are assuming the expected value of demand changes over time according to the mathematical model $\mu(t)$. If we have chosen the correct model, that is, if the process mean is changing with t in the assumed manner, and if the statistical procedure used to estimate parameters in the model yields unbiased estimates, then the expected forecast errors will be zero. We can examine forecast errors to evaluate the adequacy of the model. If it does satisfactorily represents the process, we would expect the average value of the forecast errors to be zero.

Given that we have a history of, say, one-period-ahead forecast errors, $e_1(1)$, $e_1(2)$, ..., $e_1(T)$, we have a choice of several statistics that could be used to estimate the expected forecast error. We could average all past errors to obtain

$$Y_T' = \Sigma_{t=1} e_1(t).\ 1/T = YI(T)/T \qquad (2.33)$$

where $Y(T)$ is defined implicitly as the sum of the forecast errors, or *cumulative error*. Often $Y(T)$ is used for forecast evaluation, rather then Y_T' because one does not have to keep up with the value T to compute the cumulative error each period by

$$Y(T) = Y(T+1) + e_1(T) \qquad (2.34)$$

Since we are primarily interested in the representation of the forecasting model in the near future, we logically might wish to give more weight to recent forecast errors than to older data. We could adopt the moving average concept and use only the last N errors in average:

$$1/N \,.\, \Sigma_{t=T-N+1}^{T} e_1(t) \qquad (2.35)$$

All these statistics are linear combinations of past errors with weights adding to one. Therefore, each has an expected value of zero, if the expected forecast error is zero. Because of noise (σ_{ϵ}^{2}) in the time series, these statistics will be random variables, distributed about their means.

CHAPTER 3

LEARNING

LEARNING IN ARTIFICIAL NEURAL NETWORKS

Among the many interesting properties of a Neural network, the property that is of primary significance is the ability of the network to *learn* from its environment, and to *improve* its performance through learning: the improvement in performance takes place over time in accordance with some prescribed measure. A neural network learns about its environment through an iterative process of adjustments applied to its synaptic weights and thresholds. Ideally, the network becomes more knowledgeable about its environment after each iteration of the learning process. [Haykin, p.45]

There are too many nations associated with "learning" to justify defining the term in a precise manner (Minsky, 1961). Moreover, the process of learning it is a matter of viewpoint, which makes its all the more difficult to agree on a precise definition of the term (Natarajan,1991). For example, learning viewed by a psychologist is quite different from learning in a classroom sense. Recognizing that our particular interest is in neural networks, we use a definition of learning that is adapted from Mendel and McClaren (1970).

We define learning in the context of neural networks as follows:

Learning is a process by which the free parameters of a neural networks are adapted through a continuing process of stimulation by the environment in which the network is embedded. The type of learning is determined by the manner in which the parameter changes take place.

This definition of the learning process implies the following sequence of events:

1. The neural network is *stimulated* by an environment.
2. The neural network *undergoes changes* as a result of this stimulation.
3. The neural network *responds in a new way* to the environment, because of the changes that have occurred in its internal structure.

To be specific, consider a pair of node signals x_j and v_k connected by a synaptic weight w_{kj}, as depicted in Fig.3.1. Signal x_j represents the output of neuron j, and signal v_k

represents the internal activity of neuron k. In the context of synaptic weight w_{kj} the signals x_j and v_k are commonly referred to as *presynaptic* and *postsynaptic activities,* respectively. Let $w_{kj}(n)$ denote the value of the synaptic weight w_{kj} at time n. At time n an *adjustment* $\Delta w_{kj}(n)$ is applied to the synaptic weight w_{kj} yielding the updated value $w_{kj}(n+1)$ We may thus write

$$w_{kj}(n+1)= w_{kj}(n)+ \Delta w_{kj}(n) \qquad (3.1)$$

where $w_{kj}(n)$ and $w_{kj}(n+1)$ may be viewed as the *old* and *new* values of the synaptic weight w_{kj} respectively. Equation (3.1) sums up the overall effect of events 1 and 2 implicit in the definition of the learning process presented above. In particular, the adjustment $\Delta w_{kj}(n)$ is computed as a result of stimulation by the environment (event 1), and the updated value $w_{kj}(n+1)$ defines the change made in the network as a result of this stimulation (event 2). Event 3 takes place when the response of the new network, operating with the updated set of parameters { $w_{kj}(n+1)$ } is reevaluated. [Haykin, p.46]

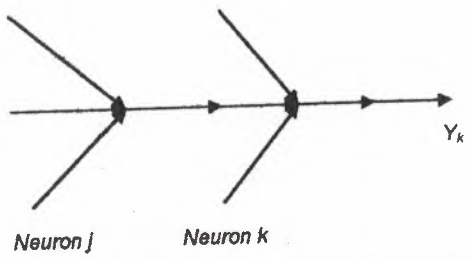


Fig 3.1 Signal Flow Graph depicting a pair of neurons *j* and *k* embedded in a neural network. Both neurons are assumed to have the same activation function.

A prescribed set of well-defined rules for the solution of a learning problem is called a *learning algorithm.* As one would expect, there is no unique learning algorithm for the design of neural network. Rather, we have a 'kit of tools' represented by a diverse variety of learning algorithm, each of which offers advantages of its own. Basically, learning algorithm differ from each other in the way in which the adjustment Δw_{kj} to the synaptic weight w_{kj} is formulated. Another factor to be considered is the manner in which a neural network (learning machine) relates to its environment. In this latter context, we speak of a *learning paradigm* referring to a *model* of the environment in which the neural network operates. We may thus offer the *taxonomy of learning* described in fig.3.2. the elements of this taxonomy are explained in the sequel. [Haykin, p.46]

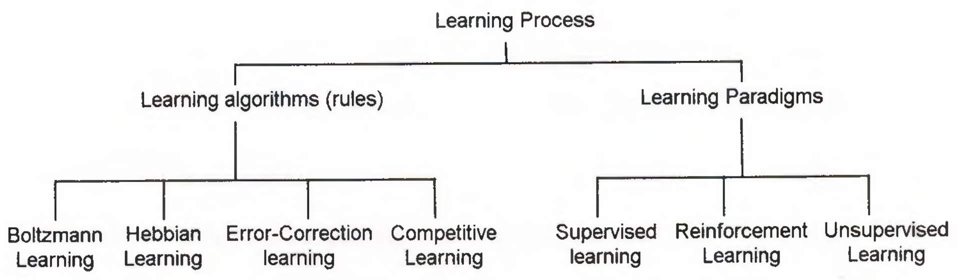


FIG 3.2 A taxonomy of the learning process

There are four basic rules: *error-correction learning, Hebbian learning, competitive learning,* and *Boltzmann learning.* Error-correction learning is rooted in optimum filtering. In contrast, both Hebbian learning and competitive learning are inspired by neurobiological considerations. Boltzmann learning is different altogether in that it is based on ideas borrowed from thermodynamics and information theory.

Error-Correction Learning

Let $dk(n)$ denote some *desired response or target response* for neuron k at time n. Let the corresponding value of the *actual response* of this neuron be denoted by $y_k(n)$. The response $y_k(n)$ is produced by a *stimulus* (vector) $\mathbf{x}$ (n) applied to the input of the network in which neuron k is embedded. The input vector $\mathbf{x}(n)$ and desired response $dk(n)$ for neuron k constitute a particular *example* presented to the network at time n. It is assumed that this example and all other examples presented to the network are generated by an environment that is probabilistic in nature, but the underlying probability distribution is unknown. [Haykin, p.47]

Typically, the actual response $y_k(n)$ of neuron k is different from the desired response $d_k(n)$. Hence, we may define an *error signal* as the difference between the target response $d_k(n)$ and the actual response $y_k(n)$, as shown by

$$e_k(n) = d_k(\mathrm{n}) - y_k(n) \qquad (3.2)$$

The ultimate purpose of error-correction learning is to minimize a *cost function* based on the error signal $e_k(n)$, such that the actual response of each output neuron in the network approaches the target response for that neuron in some statistical sense. Indeed, once a cost function is selected, error-correction learning is strictly an optimization problem to which the usual tools may be brought to bear. A criterion commonly used for the cost function is the *mean-square-error criterion,* defined as the mean-square value of the *sum of squared errors:*

$$J=E[\ \tfrac{1}{2}\textstyle\sum_{\mathrm{k}}\ e^2_{\ k}\,(n)] \qquad (3.3)$$

Where E is the *statistical expectation operator*, and the summation *is over all the neurons in the output layer* of the network. The factor ½ is used in Eq.(3.3) so as to simplify subsequent derivations resulting from the minimization of J with respect to free parameters of the network. Equation (3.3) assumes that the underlying processes are *wide-sense stationary.* Minimization of the cost function J with respect to the network parameters leads to the so-called *method of gradient descent* (Haykin, 1991; Widrow and Stearns, 1985). However, the difficulty with this optimization procedure is that it requires knowledge of the statistical characteristic of the underlying processes. We overcome this practical difficulty by settling for an *approximate* solution to the optimization problem. Specifically, we use the *instantaneous value* of the sum of squared errors as the criterion of interest:

$$E(n) = \tfrac{1}{2} \Sigma_{\mathrm{k}} e_k^2(n) \qquad (3.4)$$

The network is then optimized by minimizing $E(n)$ with respect to the synaptic weights of the network. Thus, according to the *error-correction learning rule* (or *delta rule,* as it is sometimes called), the adjustment Δw_{kj} made to the synaptic weight w_{kj} at time n is given by (Widrow and Hoff, 1960)

$$\Delta w_{kj}(n) = \eta e_k(n) x_j(n) \qquad (3.5)$$

where η is a positive constant that determines the *rate of learning.* In other words, the adjustment made to a synaptic weight is proportional to the product of the error signal (measured with respect to some desired response at the output of that neuron) and the input signal of the synapse in question. Note that this input signal is the same as the output signal of the presynaptic neuron that feeds the neuron in question. [Haykin, p.48]

Error correction learning behaves like a closed feedback system. Hence care has to be exercised in the choice of the value assigned to the learning rate parameter η, so as to ensure *stability* of the error-correction learning process. Indeed, the learning rate parameter η has a profound impact on the performance of error-correction learning in that it affects not only the rate of convergence of learning but also the convergence itself. If η is small, the learning process proceeds smoothly, but it may take a long time for the system to converge to a stable solution. If, on the other hand, η is large, the rate of learning is accelerated, but now there is a danger that the learning process may diverge and the system therefore becomes unstable. [Haykin, p.49]

A plot of the cost function J versus the synaptic weights characterizing the neural network consists of a multidimensional surface referred to as an *error-performance surface* or simply *error surface*. Depending on the type of processing units used to construct the Neural network, we may identify two distinct situations:

1. The Neural network consists entirely of processing units, in which case the error surface is bowl-shaped with a *unique* minimum point (barring the existence of a degenerate solution).
2. The Neural network consists of nonlinear processing units in which case the error surface has a *global minimum* (perhaps multiple global minima) as well as local minima.

In both cases, the objective of the error-correction learning algorithm is to start from an arbitary point on the error surface (determined by the initial values assigned to synaptic weights) and then moves towards global minimum, in a step-by-step fashion. In first case this objective is indeed attainable. In the second case, on the other hand, it is not always attainable, because it is possible for the algorithm to get trapped at a local minimum of the error surface and therefore never be able to reach a global minimum. [Haykin, p.49]

Hebbian Learning

Hebb's postulate of learning is the oldest and most famous of all learning rules; it is named in honor of the neuropsychlogist Hebb (1949). Quoting from Hebb's book, *The Organization of Behavior* (1949, p.62):

> When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic changes take place in one or both cells such that A's efficiency as one of the cells firing B, is increased.

Hebb proposed this change as a basis of associative learning (at the cellular level), which would result in enduring modification in the activity pattern of a spatially distributed *assembly of nerve cells*.

The above statement is made in a neurobiological context. We may expand and rephrase it as a two-part rule as follows (Stent, 1973; Changeux and Danchin, 1976):

1. *If two neurons on either side of a synapse (connection) are activated simultaneously (i.e., synchronously), then the strength of that synapse is selectively increased.*
2. *If two neurons on either side of synapse are activated asynchronously, then that synapse is selectively weakened or eliminated.*

Such a synapse is called *Hebbian synapse*. More precisely, we define a Hebbian synapse that uses a *time-dependent, highly local, and strongly interactive mechanism to increase synaptic efficiency as a function of the correlation between the presynaptic and postsynaptic activities.* From this definition we may deduce the following four key mechanisms (properties) that characterize a Hebbian synapse:

1. *Time-dependent mechanism*. This mechanism refers to the fact that the modification in a Hebbian synapse depend on the exact time of occurrence of the presynaptic and postsynaptic activities.

Local Mechanism. By its very nature, a synapse is the transmission site where information-bearing signals (representing ongoing activity in the presynaptic and postsynaptic units) are in *spatiotemporal* contiguity. This locally available information is used by a Hebbian synapse to produce a local synaptic modification that is input-specific. It is this local mechanism that enables a Neural network made up of Hebbian synapses to perform unsupervised learning.

Interactive mechanism. Here we note that the occurrence of a change in Hebbian synapse depends on activity levels on both sides of the synapse. That is, a Hebbian form of learning depends on a true interaction between presynaptic and postsynaptic activities in the sense that we cannot make a prediction from either one of these two activities by itself. Note also that this dependence or interaction may be deterministic or statistical in nature.

Conjunctional or correlational mechanism. One interpretation of Hebb's postulate of learning is that the condition for a change in synaptic efficiency is the conjunction of presynaptic and postsynaptic activities. Thus, according to this interpretation, the co-occurrence of presynaptic and postsynaptic activities (within a short interval of time) is sufficient to produce the synaptic modification. It is for this reason that a Hebbian synapse is sometimes referred to as *Conjunctional synapse*.

Competitive Learning

In competitive learning, as the name implies, the output neurons of a Neural network compete among themselves for being the one to be active (fired). Thus, whereas in Neural network based on Hebbian learning several output neurons may be active simultaneously, in the case of competitive learning only a single output neuron is active at any one time. It is this feature that makes competitive learning highly suited to discover those statistically salient features that may be used to classify a set of input patterns.

The idea of competitive learning may be traced back to the early works Von der Malsburg (1973) on the self organization of orientation sensitive nerve cells in the striate cortex, Fukushima (1975) on a self organizing multilayer Neural network known as *cognitron,* Willshaw and Von der Malsburg (1976) on the formation of patterned neural connections by self organization, and Grossberg (1972,1976a,b) on adaptive pattern classification. Also, there is substantial evidence for competitive learning playing an important role in the formation of topographic maps in the brain (Durbin et al., 1989), and recent experimental work by Ambros-Ingerson et al. (1990) provides further neurobiological justification for competitive learning.

There are three basic elements to competitive learning rule:

- A set of neurons that are all the same except for some randomly distributed synaptic weights, and which therefore *respond differently* to a given set of input patterns.
- A *limit* imposed on the "strength" of each neuron.
- A mechanism that permits the neurons to *compete* for the right to respond to a given subsets of inputs, such that only one output neuron, or only one neuron per group, is active (i.e., "on") at a time. The neuron that wins the competition is called a *winner-takes-all* neuron.

Accordingly, the individual neurons of the network learn to specialize on sets of similar patterns, and thereby become *feature detectors*.

In the simplest form of competitive learning, the Neural network has a single layer output neurons, each of which is fully connected to the input nodes. The network may clude *lateral connections* among the neurons, as indicated in figure 3.3. In the network chitecture described herein, the lateral connections perform lateral inhibition, with each uron tending to inhibit the neuron to which it is laterally connected. The rest of the naptic connection in the network of figure 3.3 are excitatory.

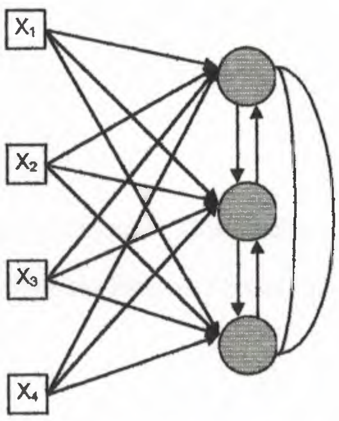


FIG 3.3 Architectural graph of a simple competitive learning network with feed forward connections from the source nodes to the neurons, lateral connections among the neurons.

For neuron j, say, to be the winning neuron, its net internal activity level v_j for a ecified input pattern **x** must be the largest among all the neurons in network. The output gnal y_j of winning neuron j is set equal to one; the output signal of all the neurons that se the competition are set equal to zero.

Let w_{ji} denote the synaptic weight connecting input node I to neuron j. Each euron is allotted a fixed amount of synaptic weight (all weights are positive), which is stributed among its input nodes; we have

$$\Sigma_i w_{ji} = 1 \text{ for all } j \qquad (3.6)$$

neuron learns by shifting synaptic weights from its inactive to active input node. If a euron does not respond to a particular input pattern, no learning takes place in that neuron. If a particular neuron wins the competition, then each input node of that neuron relinquishes some proportion of its synaptic weight, and the weight relinquished is then distributed equally among the active input nodes. According to the ***standard competitive learning rule***, the change Δw_{ji} applied to synaptic weight w_{ji} is defined by

$$\Delta w_{ji} = \eta(x_i - w_{ji}) \quad \text{If neuron } j \text{ wins the competition} \qquad (3.6a)$$

$$\Delta w_{ji} = 0 \quad \text{If neuron } j \text{ loses the competition} \qquad (3.6b)$$

where η is the learning-rate parameter. This rule has the overall effect of moving the synaptic weight vector $\mathbf{w}_j$ of winning neuron j toward the input pattern $\mathbf{x}$.

Boltzmann Learning

The *Boltzmann learning rule,* named in honor of L. Boltzmann, is a stochastic learning algorithm derived from information-theoretic and thermodynamic considerations. In a Boltzmann machine, the neurons constitute a recurrent structure, and they operate in a binary manner in that they are either in "on" state denoted by +1 or in an "off" state denoted by -1. The machine is characterized by an *energy function E,* the value of which is determined by the particular states occupied by the individual neurons of the machine, as shown by

$$E = - \tfrac{1}{2} \Sigma_i \Sigma_j \, w_{ji} s_j s_i \qquad (3.7)$$

where s_i is the state of neuron i, and w_{ji} is the synaptic weight connecting neuron i to neuron j. The fact that $i \neq j$ means simply that none of the neurons in the machine has self-feedback. The machine operates by choosing a neuron at *random*- say, neuron j - at some step of the learning process, and flipping the state of neuron j from state s_j to state $-s_j$ at some temperature T with probability

$$W_{(sj \rightarrow -sj)} = 1/1 + \exp(-\Delta E_j/T) \qquad (3.8)$$

where ΔE_j is the *energy change*(i.e., the change in the energy function of the machine) resulting from such a flip. Note that T is not a physical temperature, but rather a pseudo-temperature. If this rule is applied repeatedly, the machine will reach *thermal equilibrium.*

The neurons of a Boltzmann machine partition into two functional groups: *visible* and *hidden.* The visible neurons provide an interface between the network and the environment in which it operates, whereas the hidden neurons always operate freely. There are two modes of the operation to be considered:

- *Clamped condition,* in which the visible neurons are all clamped onto specific states determined by the environment.
- *Free-running condition,* in which all the neurons (visible and hidden) are allowed to operate freely.

This is a distinctive feature of Boltzmann machine that it uses only locally available observations under two operating conditions as mentioned above.

Supervised Learning

An essential ingredient of supervised or *active* learning is the availability of an external *teacher*, as indicated in the arrangement of figure 3.4. In conceptual terms, we may think of the teacher as having knowledge of the environment that is represented by a set of *input-output examples*. The environment is, however, unknown to the Neural network of interest. Suppose now that the teacher and the neural network are both exposed to a training vector (i.e., example) drawn from the environment. By virtue of built-in knowledge, the teacher is able to provide the Neural network with a desired or target response for that training vector. Indeed, the desired response represents the optimum action to be performed by the Neural network. The network parameter are adjusted under the combined influence of the training vector and the error signal; the *error signal* is defined as the difference between the actual response of the network and the desired response. The adjustment is carried out iteratively in a step by step fashion with the aim of eventually making the Neural network emulate the teacher; the emulation is presumed to be optimum in some statistical sense. In other words, knowledge of the environment available to the teacher is transferred to the Neural network as fully as possible. When the condition is reached, we may then dispense with the teacher and let the Neural network deal with the environment thereafter completely by itself.

The form of supervised learning is indeed the error-correction learning discussed previously. It is a closed loop feed back system, but the unknown environment is not known in the loop. As a performance measure for the system, we may think in terms of the mean-squared error (i.e., the expected value of the sum of squared vectors) defined as a function of the free parameters of the system. This function may be visualized as a multidimensional error-performance surface or simply error surface, with the free parameters as the coordinates. The true error surface is averaged over all possible input-output examples. Any given operation of the system under the teacher's supervision is represented as a point on the error surface. For the system to improve performance over time and therefore learn from the teacher, the operating point has to move down successively toward a minimum point of the error surface; the minimum point may be local minimum or a global minimum. A supervised learning system is able to do this by virtue of some useful information it has about the gradient of the error surface corresponding to the current behavior of the system. The gradient of an error surface at any point is a vector that points in the direction of *steepest descent.* In fact, in the case of supervised learning from examples, the system uses an instantaneous estimate of the gradient vector, with the example indices presumed to be those of time. The use of such an estimate results in the form of "random walk." Nevertheless, given an algorithm designed to minimize the cost function interest, and given an adequate set of input-output examples and enough time permitted to do the training, a supervised learning system is usually able to perform such tasks as pattern classification and function approximation satisfactorily.

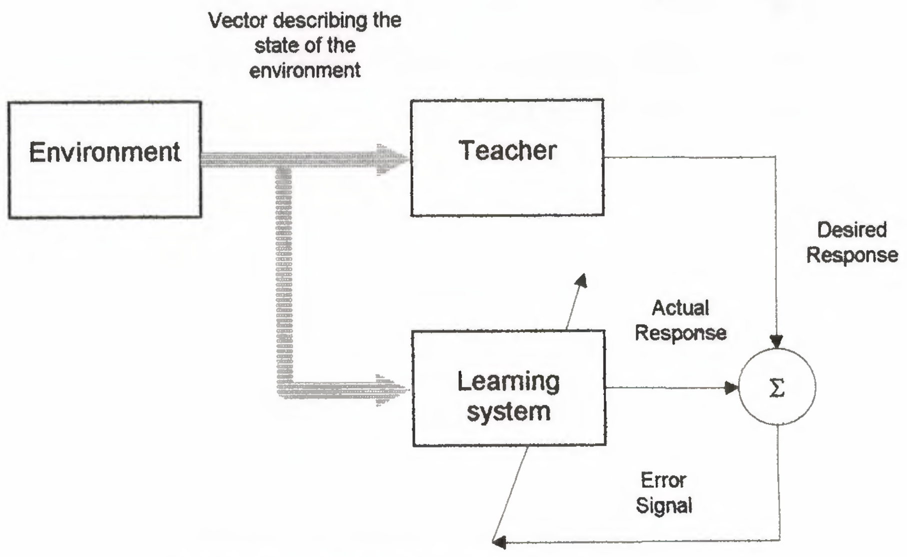


FIG 3.4 Block diagram of supervised learning

Example of supervised learning algorithm include the ubiquitous *least-mean-square (LMS) algorithm* and its generalization known as the *back-propagation (BP) algorithm*. The LMS algorithm involves a single neuron, where as the back-propagation algorithm involves a multilayered interconnection of neurons. The back-propagation algorithm derives its name from the fact that error terms in the algorithm are back-propagated through the network, on a layer-by-layer basis. Naturally, the back-propagation algorithm is more powerful in application then the LMS algorithm. Indeed, the back-propagation algorithm includes the LMS algorithm as a special case.

Supervised learning can be performed in an off-line or on-line manner. In the *off-line* case, a separate computational facility is used to design the supervised learning system. Once the desired performance is accomplished, the design is "frozen," which means that the neural network operates in a *static* manner. On the other hand, in *on-line* learning the learning procedure is implemented solely within the system itself, not requiring a separate computational facility. In order words, learning is accomplished in *real time*, with the result that the neural network is *dynamic*. Naturally, the requirement of on-line learning places a more severe requirement on a supervised learning procedure than off-line learning.

A disadvantage of supervised learning, regardless of whether it is performed off-line or on-line, is the fact that without a teacher, a neural network, can not learn new strategies for particular situations that are not covered by the set of examples used to train the network.

Unsupervised Learning

In *unsupervised* or *self-organized* learning there is no external teacher or critic to oversee the learning process, as indicated in fig 3.5. In order words, there are no specific examples of the function to be learned by the network. Rather, provision is made for a *task-independent measure* of the quality of representation that the network is required to learn, and the free parameters of the network has become tuned to the statistical regularities of the input data, it develops the ability to from internal representations for encoding features of the input and there by create new classes automatically(Becker, 1991).

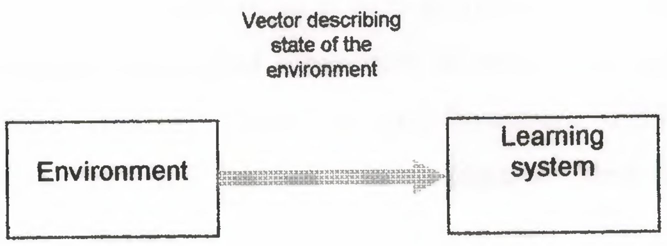


FIG 3.5 Block diagram of unsupervised learning

To perform unsupervised learning, we may use a competitive learning rule. For example, we may use a neural network consists of two layers, namely, an input layer and a competitive layer. The input layer receives the available data. The competitive layer consists of neurons that compete with each other (in a prescribed fashion) for the "opportunity " to respond to features contained in the input data. In its simplest form, the network operates in accordance with a "winner-takes-all" strategy the neuron with the greatest total input "wins" the competition and turns on; all the other neurons then switch off.

Supervised Versus Unsupervised Learning

Among the algorithm used to perform supervised learning, the *back-propagation algorithm* has emerged as the most widely used an successful algorithm for the design of multilayer feed forward networks. There are two distinct phases to the operation of back propagation learning: the forward phase and the backward phase. In the forward phase the input signals propagate through the network layer by the layer, eventually producing some response at the output of the network. The actual response so produced is compared with a desired (target) response, generating errors signals that are then propagated in a backward direction through the network. In this backward phase of operation, the free parameters of the network are adjusted so as to minimize the sum of squared errors. Back-propagation learning has been applied successfully to solve some difficult problems such as speech recognition from text (Sejnowski and Rosenberg, 1987), handwritten-digit recognition (LeCun et al., 1990a), and adaptive control (Narendra and Parthasarathy,1990). Unfortunately, back-propagation and other supervised learning algorithm may be limited by their poor scaling behavior. One possible solution to the scaling problem is to use an unsupervised learning procedure. In particular, if we are able to apply a self-organization process in a sequential manner, one layer at a time, it is feasible to train deep networks in time that is linear in the number of layers.

CHAPTER 4

TIME SERIES AND FORECASTING PROBLEM

ELECTRIC LOAD FORECASTING

USING AN ARTIFICIAL NEURAL NETWORK

ELECTRIC LOAD FORECASTING USING AN ARTIFICIAL NEURAL NETWORK

Various techniques for power system load forecasting have been proposed in the last few decades. Load forecasting with lead-times, from a few minutes to several days, helps the system operator to efficiently schedule spinning reverse allocation. In addition, load forecasting can provide information which is able to be used for possible energy interchange with other utilities. In addition to these economical reasons, load forecasting is also useful for system security. If applied to the system security assessment problem, it can provide valuable information to detect many vulnerable situations in advance.

Traditional computationaly economic approaches, such as regression and interpolation, may not give sufficiently accurate results. Conversely, complex algorithmic methods with heavy computational burden can converge slowly and may diverge in certain cases.

A number of algorithms have been suggested for the load forecasting problem. Previous approaches can be generally classified into two categories in accordance with techniques they employ. One approach treats the load patterns as a time series signal and predicts the future load by using various time series analysis techniques. The second approach recognizes that the load is then predicted by inserting the predicted weather information into the predetermined functional relationship.

General problems with the time series approach include the inaccuracy of prediction and numerical instability. One of the reasons that it gives inaccurate results is that it does not utilize weather information. There is a strong correlation between the behavior of power consumption and weather variables such as temperature, humidity, wind speed and cloud cover. This is specially true in residential areas. The time series approach mostly utilizes computationally cumbersome matrix oriented adaptive algorithms, which in certain cases may be unstable.

Following is a program which combines both time series and regression analysis. As is the case with time series approach the Artificial Neural network traces previous load patterns and predicts. The weather information is used as a mode. It performs non-linear modeling and adaptation.

Program for Electric Load Forecasting:

In this Artificial Neural network program, I have used data for the previous week of a residential area. This data includes the hourly temperature data for one week. The program calculates the following:

- Average temperature of one day
- Maximum temperature of one day
- Minimum temperature of one day
- Average temperature of the previous week
- Peak Power load for each day
- Total Power load for each day
- Average power load for previous week
- Forecasting for next week.

The topology for the Artificial Neural network for the electric load forecasting is as follows:

Input Neurons: Average Temperature, Maximum temperature, minimum temperature of day (d); Total 3 neurons

Output neuron: Peak load forecasting for next week

The data used is fictional, but logical as it does not negate the laws of nature. It can be assumed that the input temperature data is in degrees centigrade. This data is included in the *TEMPDATA.TXT* file. There are 7 rows in the input file, each having 24 columns, The first row contains the hourly temperature data for day 1 and so on. The results are obtained in *LOADDATA.TXT* file. The program is written in Pascal. The input, output and the program is included in appendices along with input and output data files.

APPENDICES

TEMPDATA.TXT

```
8 7 6 7 7 8 8.5 9 9.5 10 11 12 12 12 12.3 12 11 11 10 9.5 9 9.3 9.2 8.4

8.3 8 8 7 7 6 7 7 8.4 8.8 9 10 11 11 12 11 10 9 8 7.6 7 7 6 6

5 6 6 6.5 7 7.5 8 8 9 9.5 10 11 12 13 14 13 12 11 10 9 8 7 7 6.5

5 5.5 6 6.5 7.5 8 8.6 9 9.4 10 10.4 11 11.2 12 13 12 12 11 10 9 8 7.5 7 6

6 6.5 7 7.5 8 8.5 8.7 9 9.1 10 10.3 11 11 12 13 12 11 10 10 9 8.5 8 7.5 7

7 7.5 8 8.5 9 9.3 9.7 10 11 11 11.4 12 12 13 14 13 12 11 10 9 8.5 8 7.8 7.6

7 7.7 8 8.4 9 9.6 9.8 10 11 11 11.5 12 12 13 12 12 11 10 9.5 9 8 8 7 7
```

THE PROGRAM

```
program ANN_generate (input,output);

uses

   crt,dos;

var

  pload,tload,tave,tpeak,tlow: real;

  sum,max,min,ploadwk,wk:real;

  t: array[1..7,1..24] of real;

  d: array[1..7] of integer;

  I, J : integer;

  in1, in2 :text;


begin

    clrscr;

    Assign(in1, 'TEMPDATA.TXT');

    Assign(in2, 'loaddata.TXT');

    reset(in1);

    rewrite(in2);

    writeln(in2,'|Day|Max Temp|Min Temp|Ave Temp|Peak Load|Total Load|');

    writeln(in2,'================================================');


   wk:=0;

        for I:=1 to 7

        do begin

                tpeak:=0;

                 tlow:=0;

                 sum:=0;
```

```
                    for J:=1 to 24 do begin
                         read(in1, t[I,J]);
                         sum:=sum + t[I,J];
                         tave:=sum/24;
                         if t[I,J] > tpeak then tpeak:=t[I,J];
                         if t[I,J+1] < t[I,1] then tlow:=t[I,J];
                         end;
                    writeln('The max temperature on day ',I, ' is ',tpeak:4:2);
                   writeln('The min temperature on day ',I, ' is ',tlow:4:2);
                   writeln('The average temperature on day ',I,' is ',tave:4:2);
                   pload:=tpeak * 0.15;
                   tload:=sum * 0.15;
                   writeln('The Peak load on day ',I,' is ',pload:4:2);
                   writeln('The total load on day ',I,' is ',tload:4:2);
                   wk:=wk+pload;
                   writeln;
   writeln(in2,I,'   ',tpeak:4:2,'   ',tlow:4:2,'     ',tave:4:2,'     ',pload:4:2,'     ',tload:4:2);
  end;
     ploadwk:=wk/7;
     writeln('The forecast for the next week load is ',ploadwk:4:2);
     writeln('See loaddata.txt file for results');
     Writeln('Press enter to quit');
     writeln(in2,' ');
     writeln(in2,'The load forecast for next week is ',ploadwk:4:2);
     readln;
 Close(in1);
Close(in2);  { Close file, save changes }
```

end.

LOADDATA.TXT (RESULT)

```
|Day|Max Temp|Min Temp|Ave Temp|Peak Load|Total Load|

==========================================================

1     12.30        8.40    9.53      1.85        34.30

2     12.00        6.00    8.34      1.80        30.02

3     14.00        6.50    9.00      2.10        32.40

4     13.00        6.00    8.98      1.95        32.34

5     13.00        7.00    9.19      1.95        33.09

6     14.00        7.60    10.01     2.10        36.05

7     13.00        7.00    9.73      1.95        35.03


The load forecast for next week is 1.96
```

CONCLUSION

In this research paper, I have concentrated on Artificial Neural networks and time series forecasting. The phrase "Artificial Neural network" actually comes from the biological term neurology, which is the study of human brain and its functions.

Human brain is a complex part of the body. Over the years, scientist and researchers are studying the functions of the brain and how the neurons work. However, there is lot to be learned about, but it is known that the human brain has the potential to memorize, learn, adapt and predict so to speak. Scientist have evolved procedures which simulate this behavior of the brain and is called Artificial Neural network, using computers or network of electronic devices. Nevertheless they have not managed to match the capabilities of the human brain.

In this paper, Artificial Neural networks and their applications and characteristics are discussed. Furthermore, time series and forecasting is discussed in detail. Forecasting - a characteristic of Artificial Neural networks, though, not fully accurate but are helpful in predicting the future to some extent. There is a percentage of error involved in forecasting, but if scrutinized enough it can produce results which are beneficial to both industry and business in terms of financial control and economic monitoring.

In chapter 4, a program is included which gives a fairly good example for training an Artificial Neural network. Although, the data is fictional but the prediction is sensible.

To sum up, it is safe to say that Artificial Neural networks have a big future in the industry. The evolution of this technology is on the right path, however, there is more to be researched. The techniques known so far, have produced logical results - but there is more to be learned.

REFERENCES

&

BIBLIOGRAPHY

REFERENCES

- Simon Haykin (1994). Neural Networks: A comprehensive foundation.
- Judith E. Doyhoff (1992). Neural networks Architecture: An introduction.
- Douglas C. Montgomery, Lynwood A. Johnson (1990). Forecasting and time series analysis, 2nd edition.
- V. Rao Vemuri, Robert Rogers (1994). Artificial Neural networks; forecasting time series
- Amara, R. and Salanik, G. (1972). Forecasting: From conjectural art toward science. Technological Forecasting and Social Change , 3 (3), 415-426.
- Ascher, W. (1978). Forecasting: An appraisal for policymakers and planners . Baltimore: Johns Hopkins University Press.
- Ascher, W. (1979). Forecasting: An appraisal for policymakers and planners (rev. ed.). Baltimore: Johns Hopkins University Press.
- Ayers, R. (1969). Technological forecasting and long-range planning. New York: McGraw-Hill Book Company.
- Challis, A. & Wills, G. (1970). Technological forecasting. In D. Ashton & L. Simister (Eds.) The role of forecasting in corporate planning, pp. 100- 124. London: Staples Press.
- Cornish, E. (1977). The study of the future. Washington, D.C.: World Future Society.
- Klopfenstein, B. (1985). Forecasting the market for home video players: A retrospective analysis. Unpublished doctoral dissertation. The Ohio State University.
- Klopfenstein, B. (1986). Forecasting the market for new communication technology: The home video player experience. Paper presented at the annual meeting of the Broadcast Education Association.
- Klopfenstein, B. (1989). Problems and potential of forecasting the adoption of new media. In J. Salvaggio & J. Bryant (Eds.) Media use in the Information Age: Emerging patterns of adoption and consumer use (pp. 21-41). Hillsdale, NJ: Lawrence Erlbaum Associates.

- Martino, J. (1983). Technological forecasting for decision making. New York: Elsevier Science Publishing Company.
- Schnaars, S. (1989). Megamistakes: Forecasting and the myth of rapid technological change. New York: The Free Press.
- Schwarz, B., Svedin, U. & Wittrock, B. (1982). Methods in future studies. Boulder, Colorado: Westview Press.
- Wise, G. (1976). The accuracy of technological forecasts: 1890- 1940. Futures, 8 (5), 411-419.