M.A. El-Sharkawi, R.J. Marks II and S. Weerasooriya,
"Neural networks and their application to power engineering",
in Advances in Control and Dynamic Systems, Volume 41,
edited by C.T. Leondes, (Academic Press, 1991).

Neural Networks and Their Application to Power Engineering

Mohamed A. El-Sharkawi
Robert J. Marks II
Siri Weerasooriya

Department of Electrical Engineering
University of Washington
Seattle, WA 98195

1 Introduction

Artificial neural networks have been studied for many years with the hope of achieving human-like performance in solving certain problems in speech and image processing. There has been a recent resurgence in the field of neural networks due to the introduction of new network topologies, training algorithms and VLSI implementation techniques. The potential benefits of neural networks such as parallel distributed processing, high computation rates, fault tolerance, and adaptive capability have lured researchers from other fields such as controls, robotics, energy systems to seek neural network solutions to some of their more difficult problems.

An artificial neural network can be defined as a highly connected array of elementary processors or *neurons.* Algorithms are then crafted about this architecture. Neurons are linked with interconnects analogous to the biological *synapse.* This highly connected array of elementary processors defines the system hardware. Specification of weights to perform a desired operation can be viewed as the net's software.

CONTROL AND DYNAMIC SYSTEMS, VOL. 41

Commonly used neural networks, such as the layered perceptron, are said to be *trained* rather than programmed in the conventional sense.

Computationally, neural networks have the advantage of massive parallelism and are not restricted in speed by the von Neumann bottle neck characteristic of more conventional computers. Neural networks are characterized by high parallelism and, in many cases, are significantly fault tolerant.

At this writing, the layered perceptron is receiving the most attention as a viable candidate for application to power systems. The layered perceptron is taught by example, as opposed, for example, to an expert system, which is taught by rules. The preponderance of data typically available from the power industry, coupled with the ability of the layered perceptron to learn significantly nonlinear relationships, reveals it as a viable candidate in the available plethora of solutions for solving significant power systems engineering problems. A layered neural network is illustrated in Figure 1.

Hopfield neural networks have also been proposed for application to combinatorial search problems in the power industry. In Hopfield nets, each neuron is connected to every other neuron, as is shown in Figure 2.

In this Chapter, we provide an overview of contemporary research aimed at application of the artificial neural network to electric power engineering.

2 A Brief History of Neural Networks

Serious mathematical treatment of neural networks is usually attributed first to McCulloch and Pitts [35] and, later, Hebb [21]. A flurry of activity in neural network research in engineering circles burned in the fifties and early sixties [45, 52]. The end of this phase was marked by the publication of the negative critique **Perceptrons** [38]. The spark

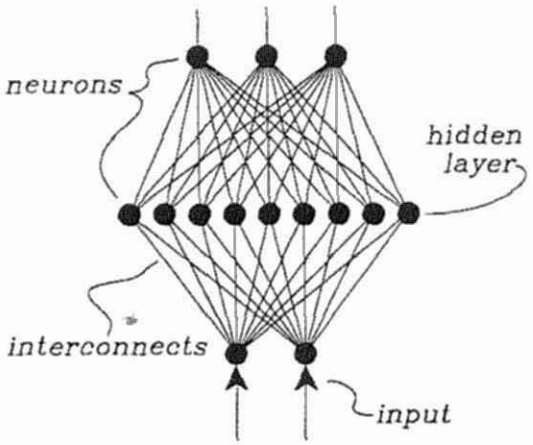


Figure 1: A layered neural network. As a layered perceptron, data is presented at the input and the output. The weights of the interconnects between the neurons are adjusted as a function of the data thereby 'training' the neural network the proper response.

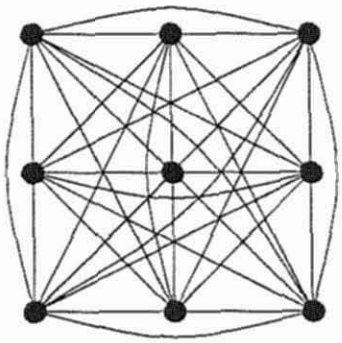


Figure 2: A homogeneously connected artificial neural network. Such architectures are used for Hopfield type artificial neural networks. The state (*i.e.* the number associated with) each neuron is determined by the state and interconnect weights of the other neurons. The state of one neuron may change, thereby changing another, etc., until the network reaches a steady state.

the exhuberant promotion of neural networks by Hopfield [22, 23] and some powerful nonlinear extensions of previous work [33].

We cannot, in this brief chapter, do justice to the recent rich history of artificial neural networks. Besides, it has already been done admirably elsewhere. The reader is referred specifically to the anthology of Anderson and Rosenfeld [6] where the development of artificial neural networks is presented as a delightful mix of commentary and classic paper reprints. Extensive bibliographies of the neural network literature are also available [50, 26].

3 Neural Network Paradigms

There is often a comparison made between artificial neural networks and their biological counterpart. Indeed, the reference to our circuitry as 'neural networks' is due to the pioneering of the field by scientists interested in the biological neuron [35, 21]. The undisputed success of biological neural networks remains highly motivating to those involved in artificial neural network research, not unlike the motivation of the flying bird was to the Wright brothers.

There is some shared terminology between the artificial and biological neural network. The links between neurons can be referred to as *synapses* or, more simply, interconnects. The neurons have also been referred to as *nodes* or, more recently, *neurodes*. Glossaries of terminology can be found in Eberhart and Dobbins [15] and Dayhoff [13].

3.1 Lateral Inhibition

Lateral inhibition describes the competition between a number of neurons for dominance. Roughly, as in capatalism, each neuron tries to turn off the other neurons while reinforcing itself. When the contest is over, the strongest neuron or neurons win with a numerically larger

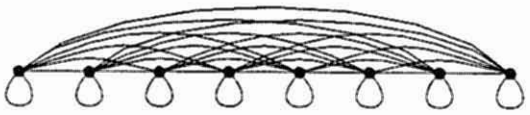


Figure 3: Illustration of a winner-take-all net. Each neuron is trying to turn off the other neurons while reinforcing itself.

state than the loosing neurons.

Specifically, consider the linear array of neurons illustrated in Figure 3. The interconnect weights between all of the neurons is $-w$ and the autoconnection of a neuron to itself will be denoted as a. We will assume both w and a are positive. Typically, a is much larger than w.

Let the state of the ith neuron at time n be $u_i[n]$. The input into the ith neuron at time $n+1$ is

$$s_i[n+1] = au_i[n] - \sum_{j \neq i} wu_j[n] \tag{1}$$

The new state of the neuron is then

$$u_i[n+1] = f\left(s_i[n+1]\right) \tag{2}$$

where

$$f(x) = \begin{cases} 0 & ; x \leq 0 \\ x & ; 0 \leq x \leq 1 \\ 1 & ; x \geq 1 \end{cases} \tag{3}$$

An inspection of the above equations reveals the dynamics of the competitive nature of this simple neural network as described in the first paragraph of this section. As an example, the reader is invited to try a simple 3 neuron example with $w = 0.1$ and $a = 1.1$. For initial states, $[0.9, 0.5, 0.1]$, convergence occurs in less than ten iterations of each neuron.

Neural networks of this type can either be implemented in discrete or continuous time. For continuous time implementation, shunt

capacitance in the weights results in a finite response time between two neurons.

For obvious reasons, the neural networks described in this section are referred to as *winner take all* nets. They have also been referred to as *maxnets* [30] and *king of the hill* [36] neural networks. Note that we can view the operation of finding a maximum a simple search problem.

3.2 Combinatorial Search

The principle of lateral inhibition can be used in artificial neural network architectures to solve certain combinatorial search problems [36, 23, 46, 47].

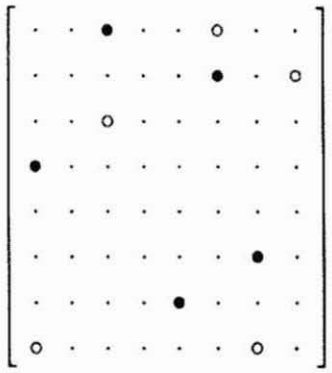
3.2.1 The Rooks Problem

A simple combinatorial search problem is the rooks problem. On an $N \times N$ chess board, we wish to place as many rooks as possible so that no rook can capture another. The maximum number of rooks that can be thus placed is N. One clear solution is to place N rooks on the diagonals. Although the rooks problem is simple, its discussion allows easy conceptualization to the more complicated Queens and Traveling Salesman problems [36].

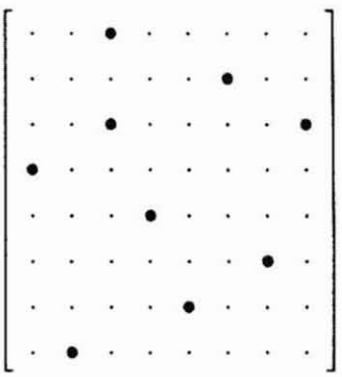
To solve the Rooks problem, we form an $N \times N$ array of neurons. Each row of N neurons will be connected in a winner-take-all configuration. Also, each column is connected in a winner-take-all configuration. Our aim is to require the $N \times N$ net to settle onto a solution that has, in steady state, only one neuron at a high state for each row and each column. The result is clearly a solution to the Rooks problem. The initial states of the N^2 neurons can be chosen randomly.

The Queens problem is analogous to the Rooks problem, except that queens, rather than rooks, are used. We must now provide, in addition, winner-take-all neural networks along each diagonal. If two neurons are connected by weights from two different winner-take-all nets, the composite weight is just the sum of the components.

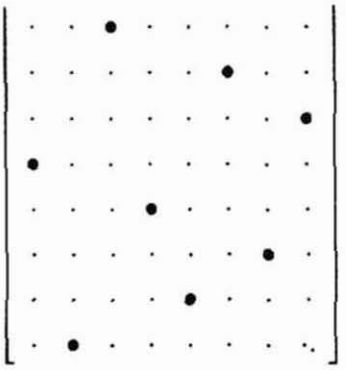
We illustrate the working of the Queens neural network by borrowing results from McDonnell *et.al.* [36]. After random initialization, the network responded with



where • denotes a neural state close to one, o denotes an intermediate value and · denotes a state close to zero. Additional iterations gave



Note that the third column has two neurons with states close to one. Interestingly, so does the third row. Since two neurons are trying to turn off the neuron in position (3,3), the final steady state result turns

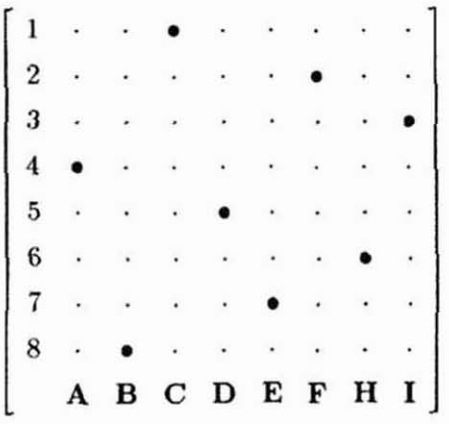


This is an acceptable solution. For $N > 3$, a chess board can support, at most, N queens.

3.2.3 The Traveling Salesman Problem

The Traveling Salesman problem [23] can also be viewed as an extension of the Rooks problem. We have, say, N cities denoted by **A,B,C,D,E....** The physical separation between cities **C** and **A** is $d_{\mathbf{AC}} = d_{\mathbf{CA}}$. We wish to arrange these cities in such a manner that a global round trip will be of minimum distance.

We will solve the Traveling Salesman problem with the use of an $N \times N$ neural network. If, in steady state, an $N = 8$ neural network reads



we would visit city **C** first, city **F** second, city **I** third, etc.

How do we set up such a net? Note, first of all, that the solution must satisfy the rooks problem. In other words, only one neuron can be on in each row and in each column. Thus, we start our net by using a Rooks problem neural network. In addition, we would like to discourage cities that are far apart from being listed together. This is accomplished by lateral inhibition of adjacent cities proportional to their separation. A large separation thus results in a large inhibition.

Consider, for example, neuron (C4) which means that city **C** is fourth to be visited. We will connect this neuron to all neurons corresponding to a visit in the number three and five positions. The connection to neuron (**F**5), for example, would be with a weight proportional to $-d_{\mathbf{DF}}$. The connection to neuron (**A**3) would be with a weight proportional to $-d_{\mathbf{AC}}$, etc. There is also a third set of weights to fine tune the number of neurons that are on in steady state. If two neurons are connected by more than one weight, the composite weight is simply the sum of the composite weights.

Randomly initialized, for the proper choice of weights, the neural network ideally approaches a solution of the traveling salesman problem.

3.2.4 Convergence proof

We will here offer a convergence proof for Hopfield type networks of the binary type when the neural nonlinearity is a unit step function. If the sum of the inputs to a neuron is positive, we set the state to one. We will also disallow autoconnects. If the sum is negative, the state is set to zero. Thus

$$u_j[n+1] = \mu\left(\sum_{j \neq i} T_{ij} u_j[n]\right) \tag{4}$$

where $T_{ij} = T_{ji}$ is the interconnect weight between neurons i and j and $\mu(\cdot)$ denotes the unit step function. We define the energy of the neural

network of N neurons at time n by

$$E[n] = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} T_{ij}u_j[n]u_i[n] \tag{5}$$

where, due to no autoconnects, we recognize that $T_{ii} = 0$. At time $n+1$, one neuron, say the kth, changes state. The overall energy of the net can also change. Denote this change by

$$\begin{aligned} \Delta E[n] = & \quad E[n+1] - E[n] \\ = & \quad -\tfrac{1}{2}\Delta u_k[n]\textstyle\sum_{i\neq k} T_{ik}u_i[n] \end{aligned} \tag{6}$$

where we have recognized that, for $i \neq k$, we have the relation $u_i[n] = u_i[n+1]$ and

$$\Delta u_k[n] = u_k[n+1] - u_k[n]$$

Given that neuron k has changed state, $\Delta u_k[n]$ can take on values of only -1 and 1. However, if $\Delta u_k = 1$, then this resulted from $\Delta u_k[n]\sum_{i\neq k} T_{ik}u_i[n] > 0$ and $\Delta E[n]$ in (6) is negative. Similarly, if $\Delta u_k = -1$, then this resulted from $\Delta u_k[n]\sum_{i\neq k} T_{ik}u_i[n] < 0$ and $\Delta E[n]$ in (6) is again negative. Thus, if the input sum to a neuron is other than zero, no matter what the state change, we have

$$\Delta E[n] < 0 \tag{7}$$

Thus, each change in state reduces the energy metric. The energy in (5) is only defined over the approximately N^2 possible combinations of N states of ± 1. It thus has a lower bound. Under our assumptions, the neural network must therefore cease decreasing energy at some point. Note that, without procedures such as *simulated annealing* [18] to assure otherwise, the iteration may stop at other than a global minimum.

Design of Hopfield type nets rests on one's ability to craft the interconnects so that the minimum of the resulting net's energy corresponds to the desired solution. Hopfield nets have been applied to a number of problems other than combinatorial search. For a more complete treatment, see Dayhoff [13] or Wasserman [49].

The fundamental Hopfield neural network can be used for applications other than combinatorial search [47], including associative memory [23] and converters [46]. There exist, however, numerous problems with Hopfield neural networks. Their capacity has shown to increase less than linearly with the number of neurons [2, 34]. The number of false stable states has been shown to increase greater than linearly with the number of neurons. This, despite the fact the required number of interconnects grows as the square of the number of neurons. Also, the time taken to *program* the neural network to generate the desired result can be quite significant [47]. In addition, for different asynchronous operations, Hopfield neural networks convergence to different solutions [11, 39].

The authors believe that the generic Hopfield neural network will survive primarily as a footnote in the development of neural networks. Nevertheless, there exists some other quite promising biologically motivated computational procedures for performing combinatorial search problems [44]. These proposed procedures must be tested against other cutting edge and more conventional methods of solving the combinatorial search problems. There are also some interesting variations on the Hopfield neural network that are worth noting. Here is a partial list.

The Boltzmann Machine. A variation of the Hopfield neural network which avoids false minima is the *Boltzmann machine* [1]. Here, with a given probability, the state of a neuron will be switched from that value dictated by the sum of its inputs. As time increases, this probability decreases[1]. Stochastic processes play a significant role in enhanced performance of many artificial neural networks [18, 31, 12].

[1]The name Boltzmann machine arises from the use of the Botzmann probability distribution.

projection neural network (APNN) [32] is a viable alternative to the Hopfield associative memory. If properly initialized, it has no false minima, will converge properly independent of asynchronous operation [39, 40], has a capacity that is proportional to the number of (excited) neurons and can operate with continuous instead of binary neural states.

The Bidirectional Associative Memory. The bidirectional associative memory (BAM) is a generalization of the Hopfield neural network [49] and suffers many of the same fundamental problems.

3.3 The Layered Perceptron

Currently, the artificial neural network most commonly used is the layered perceptron. A layered perceptron with one hidden layer is shown in Figure 1. Although convention varies, the interconnects from the input to the hidden neurons along with the hidden neurons constitute a layer. The hidden to output interconnects with the output neurons constitute a second layer. Thus, the perceptron in Figure 1 has two layers. In our treatment, we do not consider the input nodes to be neurons.

Layered perceptrons are trained by numerical data, in contrast, for example, to expert systems that are trained by rules. The layered perceptron operates in two modes: training and test. In the training mode, a set of representative *training data* is used to adjust the weights of the neural interconnects. Once these weights have been determined, the neural network is said to be trained. In the test mode, the trained neural network is activated by *test data*. The response of the layered perceptron should then be representative of the data by which it was trained. Typically, the test and training data are different sets. As we will discuss in the section on learning, training a machine to respond

properly to the same data on which it is trained is not learning, but is, rather, memorization.

A layered perceptron can be used as either a classifier or a regression machine. As a classifier, the layered perceptron categorizes the input into two or more categories. In power system security assessment, for example, the trained perceptron will categorize the power either as secure or insecure in accordance to the current system states. For regression applications, the output or outputs of the layered perceptron take on continuous values. Power load forecasting is an example of a regression application. Here, the output of the neural network corresponds to the forecasted load.

A layered perceptron with L layers if shown Figure 4. We assume there are I inputs and N_ℓ neurons in the ℓth hidden layer. The interconnects from the jth neuron in the $\ell-1$st layer to the ith neuron in the ℓth layer will be denoted by $w_{ij}(\ell)$. The state associated with the ith neuron in the ℓth layer is denoted by $s_i(\ell)$. The output of the layered perceptron is given by the states $\{s_i(L)|1 \leq i \leq N_L\}$ where the number of output neurons is N_L. For the layered perceptron in Figure 1, $L = 2$, $I = 2$, $N_L = 3$ and $N_1 = 9$.

The sum of the inputs to the ith neuron in the ℓth layer is

$$\sigma_i(\ell) = \sum_{j=1}^{N_{\ell-1}} w_{ij}(\ell) s_j(\ell - 1) \tag{8}$$

The state of a neuron is related to this value by the nonlinearity

$$s_k(\ell) = f(\sigma_i(\ell)) \tag{9}$$

where $f(\cdot)$ is referred to as a *sigmoid* or a *squashing function*. The most commonly used nonlinearity is

$$f(x) = \frac{1}{1 + e^{-x}} \tag{10}$$

This form has the useful property that

$$\frac{df(t)}{dt} = f(t)[1 - f(t)] \tag{11}$$

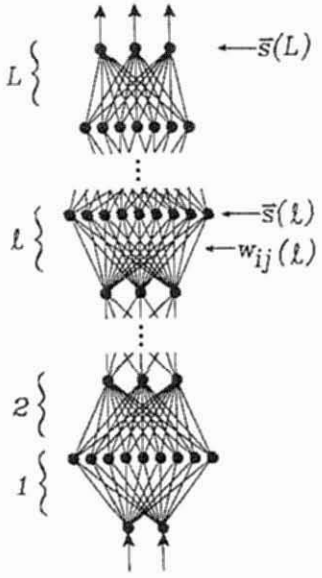


Figure 4: A layered perceptron with L layers.

The interconnects to the ℓth layer can be written in matrix form as $\mathbf{W}(\ell)$. Define the sigmoid vector operator, $\mathcal{S}_N$, such that, for any vector $\vec{v}$ of dimension N, the operation $\mathcal{S}_N\vec{v}$ results in a vector dimension N such that the ith element in the new vector is equal to the ith element in $\vec{v}$ subjected to the sigmoid nonlinearity in (10). In other words, if $\vec{v}$ is the vector of the sum of inputs into a layer of neurons, then $\mathcal{S}_N\vec{v}$ is the vector of the resulting vector states. We can then represent the input-output relationship of the layered perceptron with L hidden layers by the equation

$$\begin{aligned}\vec{o} &= \mathcal{S}_{N_L}\mathbf{W}(L)\mathcal{S}_{N_{L-1}}\mathbf{W}(L-1)\cdots\mathcal{S}_{N_\ell}\mathbf{W}(\ell)\cdots\mathcal{S}_{N_1}\mathbf{W}(1)\vec{i} \\ &= \mathcal{W}\vec{i} \end{aligned} \tag{12}$$

where $\vec{i}$ is the input vector and $\vec{o}$ is the corresponding response and the neural network operator is

$$\mathcal{W} = \mathcal{S}_{N_L}\mathbf{W}(L)\mathcal{S}_{N_{L-1}}\mathbf{W}(L-1)\cdots\mathcal{S}_{N_\ell}\mathbf{W}(\ell)\cdots\mathcal{S}_{N_1}\mathbf{W}(1) \tag{13}$$

There are commonly used variations on the layered perceptron architecture illustrated in Figure 4. The most common are

1. Interconnection between nonadjacent layers.

2. Feedback interconnects between layers (*recurrent* neural networks).

3.4 Training

The layered perceptron is trained with *training data*. For the load forecasting problem, for example, input training data might consist of a number of temperatures and the output is the forecasted load. Data from the previous year, for example, can be used. Once trained, the layered perceptron, presented with the temperatures of the current day will provide, as output, a forecast of the load for the next day.

Assume there are M training data vector pairs. Let an input of $\vec{i} = \vec{u}^m$ correspond to a desired *target* response of $\vec{o} = \vec{t}^m$. For a given set of weights, let the actual response of the layered perceptron be

$$\vec{r}^m = W\vec{u}^m \tag{14}$$

Our goal in training is to choose the interconnect weights, and thus the neural net operator, so that the response vectors, $\{\vec{r}^m | 1 \leq m \leq M\}$ are, in some sense, close to the corresponding target vectors, $\{\vec{t}^m | 1 \leq m \leq M\}$. For the mth training data pair, the measure most commonly used is the mean square error

$$\begin{aligned} E^m = & \quad \tfrac{1}{2}\|\vec{t}^m - \vec{r}^m\|^2 \\ = & \quad \tfrac{1}{2}\textstyle\sum_{i=1}^{N_L}(t_i^m - r_i^m)^2 \end{aligned} \tag{15}$$

where the norm of a vector is defined by $\|\vec{v}\|^2 = \vec{v}^T\vec{v}$. The total error can be written as

$$E = \quad \textstyle\sum_{m=1}^{M} E^m$$

For a given set of training data, $\{\vec{u}^m, \vec{t}^m | 1 \leq m \leq M\}$, this error is totally specified by the weights in the set of matrices $\{\mathbf{W}(\ell) | 1 \leq \ell \leq L\}$. Our goal in training is to find the values for these weights that minimize the error in (16).

The task of finding the minimum of an error (or cost) function is a familiar topic in optimization theory. Envision a *weight space* with coordinates $w_{ij}(\ell)$. The error function E is a positive function in this space. We wish to find that point in space where E is minimum. There exists many approaches for finding such a minimum. The method most often used in the layered perceptron is a variation of the steepest descent method, called *error back propagation* [33]. Other methods, such as conjugate gradient descent and random training, have also been used to train the layered perceptron.

3.4.1 Steepest Descent

The training procedure for layered perceptrons called error back propagation is a steepest descent method for finding the minimum of a function. At the current point in the weight space, we compute the steepest slope and take a step in that direction thereby changing our location in weight space. The process is repeated until an acceptably low error is obtained. For a weight $w_{ij}(\ell)$, steepest descent can be written as

$$w_{ij}(\ell) \Leftarrow w_{ij}(\ell) - \eta \frac{\partial E}{\partial w_{ij}(\ell)} \tag{17}$$

where η is the step size.

3.4.2 Error Back Propagation.

Finding the response of a layered perceptron to a stimulus, as in (14) can, of course, be totally performed within the neural network architecture. Such an ability is a strong attribute of the neural network in terms

of parallel implementation. Error back propagation training of a layered perceptron has the same advantage. It can be performed totally within the neural network architecture. Many other search methods applied to the layered perceptron do not have this important property.

In its fundamental form, error back propagation is an implementation of steepest descent search defined in (17). A steepest descent adjustment to the weights is first made for the first training data pair. A second step is made in response to the second training data pair, etc. In each step, all of the weights in the network are adjusted. When all of the training data has been used, the cycle is again repeated starting from the first training data pair. The process is repeated until an acceptably low error results.

Error back propagation is mathematically based on the chain rule of partial derivatives from which we can write the derivative term in (17) as

$$\frac{\partial E^m}{\partial w_{ij}(\ell)} = \frac{\partial E^m}{\partial s_i(\ell)} \frac{\partial s_i(\ell)}{\partial \sigma_i(\ell)} \frac{\partial \sigma_i(\ell)}{\partial w_{ij}(\ell)} \tag{18}$$

We will now examine each of the three terms in this expansion. First, define

$$\delta_i(\ell) = \frac{\partial E^m}{\partial s_i(\ell)} \tag{19}$$

We will say more about this term later. Since

$$s_i(\ell) = f\left(\sigma_i(\ell)\right)$$

we can use (11) to write the second term in (18) as

$$\frac{\partial s_i(\ell)}{\partial \sigma_i(\ell)} = s_i(\ell)\left(1 - s_i(\ell)\right) \tag{20}$$

Thirdly, from (8), we conclude that

$$\frac{\partial \sigma_i(\ell)}{\partial w_{ij}(\ell)} = s_j(\ell - 1) \tag{21}$$

Thus, using Equations (19), (20) and (21), we can rewrite (18) as

$$\frac{\partial E^m}{\partial w_{ij}(\ell)} = \delta_i(\ell)\left[\left(s_i(\ell)\left(1 - s_i(\ell)\right)\right] s_j(\ell - 1)\right. \tag{22}$$

Our remaining job is to interpret $\delta_i(\ell)$ in (19) in order to interpret (22). The result will explain the use of the phrase *error back propagation*. First, for $\ell = L$, we recognize that $s_i(L) = r_i^m$ and use (15) to write

$$\begin{aligned} \delta_i(L) &= \frac{\partial E^m}{\partial s_i(L)} \\ &= \frac{\partial E^m}{\partial r_i^m} \\ &= r_i^m - t_i^m \end{aligned} \tag{23}$$

This value, of course, is simply the difference between the actual and target response observed at the output of the neural network. For $1 \leq \ell \leq L-1$, we have

$$\begin{aligned} \delta_i(\ell) &= \frac{\partial E^m}{\partial s_i(\ell)} \\ &= \sum_{j=1}^{N_{\ell+1}} \frac{\partial E^m}{\partial s_j(\ell+1)} \frac{\partial s_j(\ell+1)}{\partial s_i(\ell)} \\ &= \sum_{j=1}^{N_{\ell+1}} \frac{\partial E^m}{\partial s_j(\ell+1)} \frac{\partial s_j(\ell+1)}{\partial \sigma_j(\ell+1)} \frac{\partial \sigma_j(\ell+1)}{\partial s_i(\ell)} \end{aligned} \tag{24}$$

As before, each of these three terms is evaluated separately. From (19), we recognize that the first term in (24) is

$$\frac{\partial E^m}{\partial s_j(\ell+1)} = \delta_j(\ell+1) \tag{25}$$

The second term in (24) is

$$\frac{\partial s_j(\ell+1)}{\partial \sigma_j(\ell+1)} = s_j(\ell+1)\,(1 - s_j(\ell+1)) \tag{26}$$

The third term is

$$\frac{\partial \sigma_j(\ell+1)}{\partial s_i(\ell)} = w_{ij}(\ell+1) \tag{27}$$

Substituting (25), (26) and (27) into (24) gives the following desired result.

$$\delta_i(\ell) = \sum_{j=1}^{N_{\ell+1}} \delta_j(\ell+1)\,[s_j(\ell+1)\,(1 - s_j(\ell+1))]\,w_{ij}(\ell+1) \tag{28}$$

The δ_i values on on the ℓth level can thus be determined by the δ_i values on the $\ell+1$st level.

mth input, $\vec{u}^m$ gives a response of $\vec{r}^m$. This response is compared to the target response of $\vec{t}^m$ to determine how the weights in the neural network might be adjusted to give a better response. Each weight in the neural network is updated using the steepest descent equation in (17). The required error gradient for each weight is given in (22). The weight update, from this equation, is a function only of the states of the two neurons which the weight connects and $\delta_i(\ell)$. At the output layer, as is seen from (23), $\delta_i(L)$ is simply the error between the actual and desired output. At other layers, we see from (28), the values of $\delta_i(\ell)$ at other layers can be calculated from the states, interconnect values and the δ_i's from the previous layers. Thus, $\delta_i(L-1)$ can be evaluated from $\delta_i(L)$, the values of $\delta_i(L-2)$ can be determined by $\delta_i(L-1)$ and onward, all the way to the input. Thus, the error at the output is *back propagated* in order to adjust the weights using steepest descent[2]. The $m+1$st input data pair is applied to the network and the process repeated.

There are numerous variations to the basic error back propagation training algorithm. In order to improve convergence, for example, a momentum term can be and typically is included in the weight update procedure. Here, in addition to the change in weight specified by steepest descent, a fraction of the previous weight change is added. The use of momentum allows training to plow through some local minima.

3.4.3 Problems with back error propagation

Although back error propagation is the most widely used method to train multi-layer perceptrons, it is not the only nor necessarily the best approach. Indeed, most any algorithm that searches for a minimum can be used to train a layered perceptron. Back propagation is attractive because it can be performed within the neural network structure. The

[2]This training procedure is also referred to as the *generalized delta rule*.

algorithm.

1. **Training time.** Thousands of iterations can be required to train a layered perceptron on even a simple problem.

2. **Weight accuracy.** Back error propagation requires high computational precision. This is tied to the long training time in Item 1. Each iteration can result in a change in bits of only low significance. As such, training cannot be done on high speed, but low accuracy, analog electronic or optical devices. Once trained, however, a layered perceptron can be tested using low analog percision.

3. **Layering.** The required computational precision increases with the number of layers.

4. **Scaling.** The scaling problem can be illustrated through the *curse of dimensionality.* Specifically, for a problem of similar partition complexity, the required cardinality of the training data set grows exponentially with respect to the number of input nodes. Visualize, for example, a binary classifier with two inputs and a single output. In order to classify points within a unit square to a certain accuracy, assume that we require, say, 100 input-output data pairs. Increase the number of inputs to three now requires classification within a unit cube. For the same precision, we now have to train on 10 planes with 100 points for each plane. The required number of data pairs increases to about 1000. Roughly, if P pairs are required in one dimension, then P^N pairs are required in N dimensions. We note, however, that correlation relationships among the input data can affect this argument. Note that this problem is not specific to the layered perceptron, but is applicable to any classifier or regression machine trained by example.

4 Learning

The layered perceptron is an example of a classifier or, when the output is continuous, a regression machine, which is trained by data. Once trained, a good classifier or regression machine will properly respond to *test data*. For proper performance, the test data and the training data should be different, albeit from the same statistical source.

There is a difference between *training* and *memorization*. A trained classifier or regression machine can respond with confidence to a pattern which it has not seen before. The ability to properly classify data which has not been seen before is referred to as *generalization*. Memorization, on the other hand, guarantees that, when presented with a specific element in the training data set, the classifier will respond in exactly the same manner that it was trained. In the case of memorization, the response to data other than training data is not considered in the paradigm.

The ability to interpolate among the training data does not necessarily imply good generalization. We illustrate with an example from detection theory.

Consider the two solid points in Figure 5. The one on the left is a square and the one on the right is a circle. We assume the these are the centroids of two two-dimensional Gaussian random variables with the same variance. Given some observation point, the minimum probability of error solution results simply from determination of whether the point lies to the right or the left of the perpendicular bisector between the two centroids.

Consider, then, memorization from the training data shown by the hollow squares and circles. Since we require the classifier to properly categorize all points, the resulting partition boundary would follow the winding dashed line shown. Clearly, this line would become more winding with the increase of the data set cardinality. This observation leads

us to the conclusion that some trained classifiers should not generate a zero probability of error corresponding to the training data. This, rather, is memorization.

Are there cases where the error corresponding to training data should be zero? Yes. This is generally true when their is no noise or ambiguity in the data. How then, might we determine whether the classifier or regression machine has learned or memorized? The answer is that a properly trained classifier or regression machine should respond with the same error to training data as to test data. Note that this is a necessary though not sufficient condition. If the error from the test data is much higher than that from the training data, then, chances are, the neural system is over determined. In other words, the degrees of freedom in the classifier or regression machine is too high. For the layered perceptron, this is the number of interconnects which, of course, is related to the number of neurons in the hidden layer [10]. If the error from the test and training data are similar, we are not guaranteed of proper training. Note, for example, that any partition line passing through the midpoint between the two centroids in Figure 5 would result in a classifier with the same error for training and testing. Only the perpendicular bisector gives the unique minimum error solution.

4.0.4 Classifier performance assessment

A measure of the goodness of learning for a classifier is the resulting probability of error for test data. As explained in the previous section, the optimal measure may not be zero.

Consider, as an illustration, the two dimensional closed curve in Figure 6. The solid line represents the unknown *concept*. Within the curve we wish to classify the ordered pair as one. Outside, the classification is zero. Based on available training data, the classifier tries to learn the classification boundary. The estimate of the classification boundary is the *representation* shown by the dashed curve. If the training data

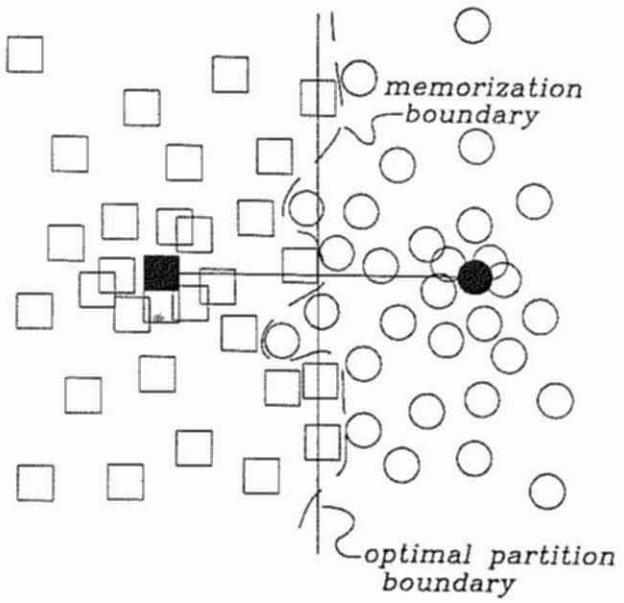


Figure 5: An illustration of the difference between learning and memorization. The solid square and circle denote centroids of two gaussian random variables with the same variance. The hollow circles and squares are corresponding realizations. If the training data is memorized, the winding broken line will be the classification boundary. Optimal detection theory, though, teaches that the vertical perpendicular bisector between the two centroids is the optimal partition. It is this boundary we wish to 'learn'.

noise is uncorrupted by uncertainty, we would expect the representation boundary to approach the concept boundary as the cardinality of the training data set increases. For a finite size training set, the resulting probability of error is equal to the probability of false classification. This is equal to the shaded area in Figure 6 [48].

For a layered perceptron, the classification problem in Figure 6 can be evaluated using two inputs corresponding to the (x, y) coordinates of the input, and a single output corresponding which offers its estimate

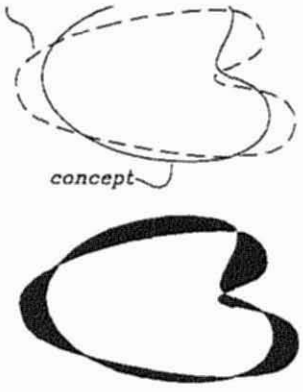


Figure 6: The *concept*, shown by the solid line, is to be learned. The broken line denotes the learned *representation*. The probability of error is equal to the probability a point is chosen is the shaded area. If the training data is chosen randomly, then a decrease in the probability of error also requires a decrease in the probability of learning something new.

of the proper classification. The output neuron will typically take on a continuous value between, say, zero and one. Typically, this value would be thresholded at $1/2$. That is, if the value of the output were above $1/2$, we would announce a 'one'. A zero would result from an output value below one half. The possible errors are the false alarm with probability

$$\alpha = \text{Prob}[1 \text{ is announced given that the proper class is zero}]$$

and the probability of a false negative

$$1 - \beta = \text{Prob}[0 \text{ is announced given that the proper class is one}]$$

The quantity β is also sometimes referred to as the *detection probability*. Generally, as the detection probability increases, so does the false alarm probability. In a layered perceptron with a single output, this trade off can be realized simply by choosing different values of the output neuron's threshold. As the threshold decreases, the false alarm

probability and detection probabilities increase. The relationship between the two errors is referred to in communications as the *receiving operating characteristic* or ROC curves. Using ROC curves to assess classifier performance was suggested by Eberhart and Dobbins [15]. The concept can be generalized to layered perceptrons with multiple outputs.

There exists a relatively large literature on detection theory. To the authors' knowledge, a comparative study between neural networks and more conventional nonparametric detectors has yet to be performed.

In certain cases, such as thermo nuclear meltdown and power system security assessment, a relatively high false alarm rate can be tolerated in order to achieve a high detection probability. In other cases, such as choosing the most efficient of two power sources, we are more interested in the total probability of error given by

$$\text{Prob[error]} = \alpha \times \text{Prob[0]} + (1 - \beta) \times Prob[1]$$

where, for example, Prob[1] = the probability that the proper classification is one. As we vary the threshold of the single output neuron from zero to one, there exists an intermediate minimum value for Prob[error]. In the multi output case, there exists a setting of thresholds which will minimize the error probability.

4.1 Determining the best net size

The degrees of freedom of the neural network, equal to the number of interconnects and therefore related to the number of hidden neurons, must be matched, in some sense, to the complexity of the classification boundary. Visualize, for example, the problem of classifying the integer parts of continuous numbers from one to ten as odd or even. This problem clearly requires more degrees of freedom (and therefore more neurons) than classifying whether the same numbers are greater or less than five.

Currently, in the absence of parametric guidance, the only proposed method of determining the best number of hidden neurons is through the use of comparative cross validation among two or more neural networks. (We consider the augmentation of one neural network to another by an increase or decrease in the number of hidden neurons as a different net). Moving from a small number of hidden neurons to a large number must decrease the overall probability of error while maintaining an equivalent error performance for the test and training data. When the perceptron's performance on training data begins to lag, we have started the process of memorization.

4.2 Query Based Learning

When a classifier or regression machine with a static architecture is trained by random example, *the more that is learned, the harder it is to learn*[3]. This is true of the multilayered perceptron. Indeed, in the absence of data noise, additional learning takes place in a multi layered perceptron only if new data is introduced that the neural network improperly classifies. The closer the representation comes to the concept, the smaller the chance that this happens.

To illustrate, consider the classification problem of learning the location of a point a on the interval $0 \leq a \leq 1$. We choose a point at random on the unit interval. If it is to the right of a, we assign it a value of one. If it is to the left of a, the result is 0. It is clear that, after a number of data points have been generated at random on the unit interval, that a lies somewhere between the rightmost 0 and the left most 1. Call this subinterval C. If we generate a new data point that does not lie in the subinterval C, we have learned nothing new. If the new point lies in the subinterval C, then we revise the subinterval and make it's duration shorter. Doing so, however, decreases the chance that the next data point contains new information. That is, the probability decreases that

[3] *i.e.* you can't teach an old dog new tricks

the more we learn about the location of the point a, the harder it is to learn. One approach to counteract this phenomenon is with the use of oracles in query based learning [3, 24, 25].

4.2.1 Oracles

In supervised learning, each feature vector is assigned a classification (or regression) value or values. There is usually a cost associated with this assignment, such as the cost of performing an experiment, computational overhead or simply time. We can envision this process as a presentation to an *oracle* the feature vector. For a cost, the oracle will reveal to us the proper classification or regression value associated with that vector. Note that, if we have deep pockets to pay the oracle, there is no need to for a classifier or regression machine such as the layered perceptron. Any feature vector we desire can be taken to the oracle for proper categorization.

In many cases of interest, we have the freedom to choose the feature vectors that we present to the oracle. Ideally, we would like to present those vectors to the oracle that, in some sense, will result in training data of high information content. The motive is to effectively train the classifier or regression machine with a low training data cost. Query based training is concerned with the manner in which the training vectors that will result in high information data are chosen.

Note that, as is illustrated in Figure 6, the binary classification problem is totally determined by the classification boundary. Indeed, here is an obvious case where the importance of data to the classification can be noted. Roughly, the closer a feature vector is to the concept classification boundary, the more information it contains. One way to exploit this observation is through interval halving. Between each feature vector classified 0 and each classified 1, there exists a classification boundary. In many cases, taking the geometric midpoint of these two

the boundary. This is assured, for example, if the underlying concept is convex.

To illustrate interval halving, let's return to the problem of finding the point a on the interval $(0, 1)$. After N randomly generated points on this interval, we would expect (in the sense of statistics), that the distance between the right most zero and the left most one is about $1/N$. Using interval halving, on the other hand, this is reduced to about 2^N. The acceleration in learning is indeed remarkable.

4.2.2 Inversion of the Layered Perceptron

Another approach to query based learning is, in effect, to ask a partially trained classifier or regression machine "What is it you don't understand?". The response of the classifier or regression machine is taken to the oracle for proper categorization and the result is added to the training data set. The classifier is then further trained and the process repeated.

How might we apply this query approach to, say, a trained layered perceptron classifier with a single output? Assuming that the output neuron is thresholded at one half to make the classification decision, the representation boundary in feature vector space is the locus of all inputs that produce an output of one half. This locus of points corresponds to feature vectors of maximum confusion. In other words, when presented with such a vector, the neural network is uncertain to the corresponding classification. If there were a technique to find a number of these points, they could be taken to the oracle to clear the confusion. The data from the oracle could then be used for training data. The perceptron can then be retrained to yield a higher accuracy. The question is, how can the locus of confusion be generated? The answer is through inversion of the neural network [24, 25, 29].

One technique for inversion of the layered perceptron has been proposed by Hwang *et.al.* [24, 25]. The approach is basically the dual of back propagation. Instead of holding the training data constant and adjusting the weights by using steepest descent, the weights are held constant and the input is adjusted using steepest descent to give an output of one half. Clearly, a number of inputs will give the response of one half. Variations are imposed by changing the initial starting point of the input in the iteration procedure. Use of inversion in query based learning has resulted in a significant improvement in accuracy of a trained layered perceptron in comparison with a second neural network trained with a randomly selected data set of the same cardinality. In practice, data near (rather than on) the representation boundary was used to accelerate training.

4.2.3 Adaptive Learning

In the training of a layered perceptron, an assumption of stationarity of the training data is typically made. In a number of cases of interest, however, the training data is a slowly varying nonstationary process. Consider, as an example, training data for the load forecasting problem generated in a developing urban area. Training data from five years prior will be different in character to data more recently generated. In order for the layered perceptron's weights to adapt to a slowly varying nonstationarity, such a procedure should

1. still respond appropriately to previous training data if those data are not in conflict with the new training data and

2. adapt to the new training data even when it is conflict with portions of the old data.

The adaptively trained neural network (ATNN) of Park *et.al.* [43] assures proper response to previous training data by seeking to minimize a weight sensitivity cost function while, at the same time, minimizing

the mean square error normally ascribed to the layered perceptron. Although space does not permit a detailed explanation, we will illustrate the performance of the ATNN through an exemplar problem [43]. Later in this chapter, the procedure will be applied to the load forecasting problem.

In Figure 7, 100 training data pairs were generated using the solid curve. When a layered perceptron is trained with these points using error back propagation, the response to test data is indistinguishable from the solid curve. The 101st data point is introduced ot 0.5. It is 10% larger than the other datum there. When the layered perceptron is retrained using error back propagation, the generalization is shown by the dots. When trained using the ATNN, the dashed line results as the generalization. Clearly, the dashed line has adapted to the new data point without a resulting drift of the other data. Such was not the case for error back propagation. A detailed explanation of the ATNN is given in Park *et.al.* [43].

4.2.4 Unsupervised Learning

The layered perceptron is trained using *supervised learning.* The perceptron is told the desired output for each input pattern. *Unsupervised learning*, on the other hand, does not require knowledge of the output. The classifier, rather, looks for similarity of structure in input patterns and groups them accordingly. The most visible of neural networks paradigms using unsupervised learning are *adaptive resonance training* (ART) [20] and the Kohonen feature map [27] both of which exist in various forms [6].

As a rule, if supervised training can be applied to a problem, it is preferable to unsupervised learning. One learns better with a teacher than without one.

Unsupervised learning typically compares an input pattern to a number

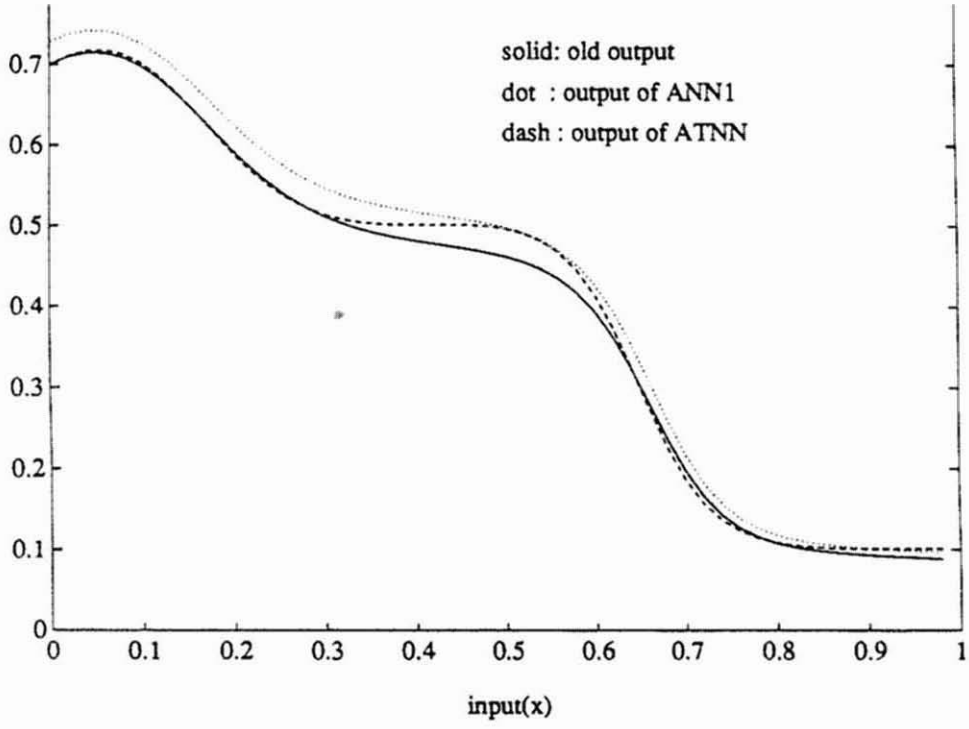


Figure 7: Illustration of the use of the adaptively trained neural network.

of representative stored templates. If the new pattern is sufficiently close to an existing template, then information from the pattern is used to modify and reinforce the template. If the pattern is not close to an existing template, then a new template can be formed.

As with other neural network paradigms, there exist other non neural network approaches to unsupervised learning (*e.g.* k-means clustering).

4.3 Comparative Performance

Other artificial neural networks have fallen from favor in an application sense because, quite simply, they are not competitive with other

more conventional approaches. The same question must be posed in regard to the layered perceptron. Does the layered perceptron preform better than other classifiers or regression machines programmed from examples using supervised learning? Although abstract analysis of this question may be possible in some cases, it must ultimately be answered in regard to actual data. Comparisons of the layered perceptron have been performed with *classification and regression trees* (CART) and *nearest neighbor lookup* for such problems as power security assessment and load forecasting and, in each case, have shown the layered perceptron to perform better in terms of classification or regression accuracy [7]. Both of these competing algorithms can be implemented using parallel processing.

In comparison with nearest neighbor lookup, the layered perceptron was shown to interpolate much more smoothly and with greater accuracy for the problem of power security assessment [3, 16].

5 Neural Network Implementation

Implementation of artificial neural networks is still quite immature. Implementation can currently be broken into the following categories.

1. **Emulators and simulators.** Currently, the most commonly used computational method for neural networks is simulation using standard software and/or emulator boards. Ironically, serial computation is here used to evaluate the performance of these highly parallel algorithms. Emulation packages and electronics are available from a number of vendors. Software is also available in association with some books on neural networks [15].

2. **Analog Electronics.** The speed of analog electronics is attractive for implementing neural network algorithms [37]. The percision of analog electronics, as we have noted, is

not high enough for back error propagation - the most commonly used training procedure. Analog electronics can be used, however, in other neural networks and in the testing of trained multilayer perceptrons. A superb overview is given by Graf and Jackal [19].

3. **Digital Electronics.** Digital electronics will be the implementation technology of choice for neural networks in the near future. The technology as applied to neural networks is expanding rapidly and will be the first viable option to emulation. Atlas and Suzuki [9] give a thorough review.

4. **Optronic Implementation.** Optics offers a quite promising medium for the implementation of neural networks [17]. Consider, for example, the high connectivity required for neural networks. Multilevel VLSI must be used in electronics to avoid shorting since electrons cannot go through electrons. Photons, on the other hand, can go through photons. For this reason, optics is capable of extremely high interconnect capabilities. On the negative side, optical implementation is quite far behind electronics in maturity of implementation. As in electronics, fast optics is analog optics. The same comments in regard to required accuracy are also applicable here.

6. Selected Applications to Power Systems**

Artificial Neural Networks have been recently proposed as an alternative method for solving certain traditional problems in power systems where conventional techniques have not achieved the desired speed, accuracy or efficiency.

Neural Network (NN) applications that have been proposed in the literature up to date can be broadly categorized under three main areas: Regression, Classification and Combinatorial Optimization. The applications involving regression includes Transient Stability, Load forecasting, Synchronous machine modelling, Contingency screening and Harmonic evaluation. Applications involving classification include Harmonic load identification, Alarm processing, Static security assessment and Dynamic security assessment. In the area of combinatorial optimization, there is topological observability and capacitor control.

In the following sections, we provide an overview of the reported Neural Networks (NN's) applications to power systems. A more in depth treatment of the material can be found in the respective references.

6.1 TRANSIENT STABILITY

Stability of a power system deals with the electro-mechanical oscillations of synchronous generators, created by a disturbance in the power system. Whether or not the post disturbance process leads to loss of synchronous operation, is the subject of primary concern. When the disturbance is small and when the system oscillations

---

**Portions of this section are reprints with permission from IEEE [16,51,54,55,56,57,58,59,62,63,65], 1989-1991.

equilibrium point, concepts of linearized systems analysis can be applied to determine the stability of the power system. This is known as *steady state* or *small signal stability* assessment. However, when the disturbance is large and when the oscillatory transients are significant in magnitude, nonlinear system theory or explicit time domain simulations have to be used to analyze the system stability. The ensuing analysis is known as *transient stability* assessment.

6.1.1 Problem Description

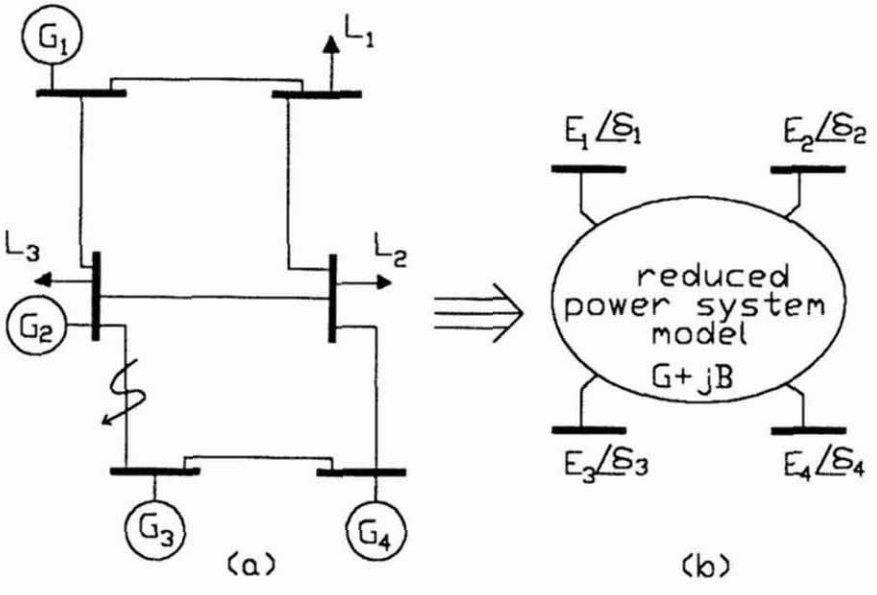


Figure 8. Network reduction for stability calculation

Figure 8(a) shows a small test power system. It has 6 buses with 4 generators and three loads. Since transient stability analysis is focused on the generator dynamics through a few cycles following the fault, certain simplifying assumptions can be made. All generators are replaced by the corresponding internal emfs (E) behind a transient reactance (X_d') Each load is replaced by a fixed admittance based on

the pre-fault power flow. These assumptions are combined with generic circuit reduction techniques, to reduce the topology of the original power system to one that is shown in figure 8(b). This reduced power system forms the basis for transient stability calculations.

The admittance matrix of the base power system can be written as,

$$\mathbf{Y} = \begin{bmatrix} \mathbf{Y}_{GG} & \mathbf{Y}_{GL} \\ \mathbf{Y}_{LG} & \mathbf{Y}_{LL} \end{bmatrix}$$

where subscripts G and L stand for generator and load buses respectively. The modified admittance matrix corresponding to the reduced power system where all load buses are eliminated as shown in figure 8(b) is given by,

$$\mathbf{G} + j\,\mathbf{B} = [\mathbf{Y_i}^{-1} + (\text{diag}\,\frac{1}{X_{di}})^{-1}]^{-1}$$

where

$$\mathbf{Y_i} = [\,\mathbf{Y}_{GG} - \mathbf{Y}_{GL}\,\mathbf{Y}_{LL}^{-1}\,\mathbf{Y}_{LG}\,]$$

For the reduced power system, the equations for generator and rotor dynamics can be written as follows.

$$M_i\,(d^2\delta_i/dt^2) + D_i\,(d\delta_i/dt) + P_{e_i} = P_{m_i} \qquad (i = 1,...N_G) \quad (29)$$

$$d\delta_i/dt = \omega_i \quad (30)$$

$$P_{e_i} = E_i \sum_j E_j\,[\,G_{ij}\cos(\delta_i - \delta_j) + B_{ij}\sin(\delta_i - \delta_j)\,] \quad (31)$$

where,

M_i, D_i - inertia and damping constants of the i^{th} generator

P_{c_i} - electrical power output of i^{th} generator

P_{m_i} - mechanical power input to the i^{th} generator

E_i - equivalent field voltage behind the transient reactance X_d'

G_{ij}, B_{ij} - real & imaginary parts of the reduced admittance matrix

δ_i - rotor angle of the i^{th} generator relative to a synchronous reference

ω_i - angular velocity of i^{th} generator relative to the same synchronous reference

N_G - number of generators in the system

Equations (29) and (30) are the differential equations governing the rotor dynamics of the i^{th} generator. Equation (31) gives the electrical power output of the i^{th} generator calculated by applying Kirchoffs Laws.

Transient stability is determined by observing the variation of δ_is' as a function of time in the post-fault period. Power system is said to be transiently stable for a given disturbance if the oscillations of all rotor angles damped out and eventually settled down to values within the safe operating constraints of the system. For any disturbance, the transient stability of a power system depends on three basic components: the magnitude of the disturbance, the duration of the disturbance and the speed of the protective devices. For example, in the case of a transmission line fault, assume that the line section is first isolated and then successfully reclosed. There exists a threshold parameter known as the *Critical Clearing Time* (CCT) where if the fault is cleared before this time, the power system remains stable. However, if the fault is cleared after the CCT, the power system is

likely to become unstable. Hence, stability analysis may involve the calculation of the CCT for a given contingency.

CCT is a complex function of pre-fault system conditions, disturbance structure and the post-fault conditions. There are two commonly used methods for calculating CCT, namely 1) Numerical integration and 2) Liapunov-type stability criteria [53]. The first method involves extensive time domain simulation of the power system while the scope of the second method is limited by its restrictive assumptions. Due to the many possible pre-fault operating conditions and types of faults, computationl effort needed to assess the CCT for each of these scenarios is prohibitive.

6.1.2 Neural Network Approach

The estimation of CCT can be looked at as a regression problem where pre-fault system parameters are used to predict the CCT for the corresponding fault. A multi-layer perceptron was proposed to be trained using back-propagation to learn a set of input attributes and the corresponding CCTs for a specified fault under varying operating conditions [53].

The inputs to the NN (α_i) for a specified contingency are selected as,

$$\alpha_i = \delta_i(t_0) - \delta_0(t_0) \tag{32}$$

where $$\delta_0 = \frac{1}{M_0} \sum_i M_i \delta_i$$

$$M_0 = \sum_i M_i$$

$$\alpha_{NG+i} = \frac{P_{m_i} - P_{f_i}}{M_i} \tag{33}$$

δ_0 is known as the center of mass while ω_0 is known as the center of angle. P_{f_i} corresponds to the reduced electrical power output of the i_{th} generator during fault initiation. This change from the steady state electrical power P_{c_i} is brought about due to the change in network impedance caused by the fault and also due to the effect of the transient reactance of the generators.

$$\alpha_{2NG+i} = (P_{m_i} - P_{f_i})^2/M_i \quad (i = 1,.....N_G) \tag{34}$$

The NN input quantity given by equation (33) gives a measure of the rotor angle deviation at the instant of fault clearing. The input quantity described by equation (34) is a measure of the individual acceleration energy of the generators of the system accumulated during the fault [53].

The output of the NN is the CCT corresponding to the given contingency under the described inputs. During generation of training data, CCT for the corresponding input quantities is obtained by repetitive numerical integration of the post-disturbance system equations using different reclosing times. The CCT would correspond to the maximum time for reclosure after the initial isolation of the line in order to maintain synchronous operation.

For a specific test of the algorithm, a 3-phase fault was simulated at location shown in figure 3.1(a). The CCT was calculated for the case where the fault was initially isolated by tripping the line and the system subsequently restored by reclosing the line. 30 training patterns were generated for a combination of different loading levels and two different base power system topologies. The trained NN was used to estimate the CCT for the same type of fault under varying load levels and varying topologies. The estimated CCT was compared to the analytical value calculated through numerical integration. Close comparison of results was reported.

The ability of a NN to generalize between different network topologies was observed. This adaptability was facilitated by providing training data corresponding to couple of different base topologies. This is a key idea that could be applied to training NN's for problems with time varying power system topologies.

So far, the merit of the NN in calculating the CCT is limited to the above mentioned fault scenario and the restrictive second order model of the generator. Simulations are also restricted to simple 3-phase line faults. The ability of the NN to predict CCT under more complicated fault scenarios is not clear. The training data should be produced by using a higher order generator model to include other transients caused by the presence of damper windings and excitation systems.

6.2 LOAD FORECASTING

Forecasting electrical load in a power system with lead-times varying from hours to days, has obvious economic as well as other advantages. The forecasted information can be used to aid optimal energy interchange between utilities thereby saving valuable fuel costs. Forecasts also significantly influence important operations decisions such as dispatch, unit commitment and maintenance scheduling. For these reasons, considerable efforts are being invested in the development of accurate load forecasting techniques.

6.2.1 Problem description

Most of the conventional techniques used for load forecasting can be categorized under two approaches. One treats the load demand as a time series signal and predicts the load using different time series

analysis techniques. The second method recognizes the fact that the load demand is heavily dependent on weather variables. The general problem with time series approach include the inaccuracy of prediction and numerical instability [42]. The main reason for instability is not considering the weather information which is known to have a profound effect of load demand. Numerical instability is caused by computationally cumbersome matrix manipulations.

The conventional regression type approaches use linear or piecewise-linear representations for the forecasting function. The accuracy of this approach is dependent on the functional relationship between the weather variables and electric load which must be known a priori. This approach cannot handle the non stationary temporal relationship between the weather variables and load demand.

6.2.2 Neural Network Approach

NN can combine both time series and regression approaches to predict the load demand. A functional relationship between weather variables and electric load is not needed. This is because NN can technically generate this functional relationship by learning the training data. In other words, the nonlinear mapping between the inputs and outputs is implicitly imbedded in the NN.

The NN approach proposed in [42,54] uses previous load data combined with actual and forecasted weather variables as inputs, and the load demand as the output. As an example, to predict the load at the k^{th} hour on a 24 hour period, the NN uses the following input/output configuration.

NN inputs : k, L(24,k), T(24,k), L(m,k), T(m,k) and $T_p(k)$

NN output : L(k)

where,

k	- hour of predicted load
m	- lead time
L(x,k)	- load at x hours before hour k
T(x,k)	- temperature at x hours before hour k
$T_p(k)$	- predicted temperature at hour k

During training, the actual temperature T(k) is used instead of $T_p(k)$. Different NNs are trained to predict the load demand at varying lead times. The results are reported too be better than those obtained through some of the existing extensive regression techniques.

One of the test results presented in [42] is given for brevity. Five sets of actual load and temperature data were used in the study. Each set contained data corresponding to 8 consecutive days as shown in table 1. Out of each set, data corresponding to the six weekdays were selected. No weekends or holidays were included.

Table 1. Test data sets

sets	Test data from
Set #1	01/23/89 - 01/30/89
Set #2	11/09/88 - 11/17/88
Set #3	11/18/88 - 11/29/88
Set #4	12/08/88 - 12/15/88
Set #5	12/27/88 - 01/04/89

From [42] courtesy of IEEE, (C) IEEE,1990

The NN was trained to forecast the hourly load with one hour lead time. Table 2 shows the forecasting error(%) of each day in the test sets. Each day's result is averaged over a 24 hour period. The average error for the 5 test sets was found to be 1.40%.

Table 2: Error(%) of hourly load forecasting with one hour lead time

days	set #1	set #2	set #3	set #4	set #5
day 1	(*)	1.20	1.41	1.17	(*)
day 2	1.67	1.48	(*)	1.58	2.18
day 3	1.08	(*)	1.04	(*)	1.68
day 4	1.40	1.34	1.42	1.20	1.73
day 5	1.30	1.41	(*)	1.20	(*)
day 6	(*)	1.51	1.29	1.68	0.98
average	1.35	1.39	1.29	1.36	1.64

(*: predicted temperature, T_p is not available)

From [42] courtesy of IEEE, (C) IEEE,1990

.2.3 Comments

he results show that NN can be trained to predict the load demand y among its training patterns. However, one network cannot handle ll cases where enough and sparce representation exist in the training st. For example, a NN trained to predict electric loads of normal eather conditions, may not do accurate prediction during extreme eather conditions such as cold snaps and heat waves. To predict lectric loads under these conditions, a separate NN may be needed. lso the holidays cannot be accurately predicted. It is also worth ientioning that the above restrictions are also applied to all existing chniques.

.3 SYNCHRONOUS MACHINE MODELLING

nchronous generators are the only available choice for bulk electric ower generation. Hence, the synchronous machine dynamics are tal to power system stability in both steady state and transient state erating modes. Accurate modelling of the synchronous machine namics is imperative for the operation and control of any power stem.

6.3.1 Problem Description

As mentioned in section 6.1, when stability analysis with a high degree of accuracy is desired, a 2nd order model for the synchronous machine is often inadequate. Other operating modes of the synchronous generator are needed in order to achieve the required degree of accuracy. For example, the dynamics caused by the damper windings, armature reaction, excitation system, saliency and other inherent control loops are important in determining the accurate behavior of the synchronous machine.

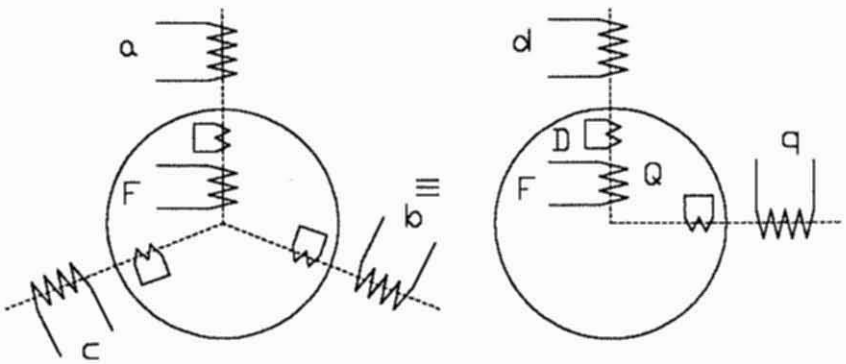


Figure 9. (a) 3Φ windings of the synchronous m/c (b) d-q axis equivalent model

Figure 9(a) shows a three phase representation of a synchronous machine. The figure shows the stator, field and damper windings. Figure 9(b) shows the equivalent d-q axis model obtained through Blondale's transformation. In addition to the two mechanical mode equations, flux linkages of the d,q axis and field windings can be used to derive the following 7th order model for the synchronous generator.

$$
\begin{aligned}
\frac{d\omega}{dt} &= \frac{1}{M}\,(P_m - P_e - P_d) \\
\frac{d\delta}{dt} &= \omega_b(\omega - 1) \\
\frac{d\psi_d/\omega_b}{dt} &= v_d + r_a i_d + \omega \psi_q \\
\frac{d\psi_q/\omega_b}{dt} &= v_q + r_a i_q - \omega \psi_d \\
\frac{d\psi_F/\omega_b}{dt} &= v_F - r_F i_F \\
\frac{d\psi_D/\omega_b}{dt} &= - r_D i_D \\
\frac{d\psi_Q/\omega_b}{dt} &= - r_Q i_Q
\end{aligned}
\qquad (35)
$$

where,

P_m, P_e, P_d	- mechanical, electrical and damping powers
ψ_d, ψ_q	- d and q components of armature flux linkages
i_d, i_q	- d and q components of armature currents
v_d, v_q	- d and q components of armature voltages
r_a, r_F	- armature and field resistance
ψ_D, ψ_Q	- d and q components of damper winding flux linkages
i_D, i_Q	- d and q components of damper winding currents
r_D, r_Q	- d and q components of damper winding resistances
ψ_F, i_F	- flux linkages and current of the field circuit

The d,q and F axis fluxes and currents are related by the following two matrix expressions.

$$\begin{bmatrix} \psi_d \\ \psi_F \\ \psi_D \end{bmatrix} = \frac{1}{\omega_0} \begin{bmatrix} x_d & x_{md} & x_{md} \\ x_{md} & x_F & x_{md} \\ x_{md} & x_{md} & x_D \end{bmatrix} \begin{bmatrix} -i_d \\ i_F \\ i_D \end{bmatrix} \tag{36}$$

$$\begin{bmatrix} \psi_q \\ \psi_Q \end{bmatrix} = \frac{1}{\omega_0} \begin{bmatrix} x_q & x_{mq} \\ x_{mq} & x_Q \end{bmatrix} \begin{bmatrix} -i_q \\ i_Q \end{bmatrix}$$

where,

x_d, x_q - d and q components of armature self inductance

x_{md}, x_{mq} - d and q components of armature mutual inductances

Equations (35) and (36) are linked to the external power system as follows.

$$\begin{aligned} P_e &= i_q \psi_d - i_d \psi_q \\ v_d &= v_t \sin \delta \\ v_q &= v_t \sin \delta \end{aligned} \tag{37}$$

Equations (35) through (37) can be written as a nonlinear state space model

$$\frac{d}{dt}\mathbf{X} = \mathbf{f}(\mathbf{X}, \mathbf{U}) \tag{38}$$

where $\mathbf{X} = \begin{bmatrix} i_D \\ i_Q \\ i_d \\ i_F \\ i_q \\ \omega \\ \delta \end{bmatrix}$ and $\mathbf{U} = \begin{bmatrix} v_t \sin \delta \\ v_F \\ v_t \cos \delta \\ \frac{P_m}{M} \end{bmatrix}$

$$\mathbf{X}(k+1) = \mathbf{F}(\mathbf{X}(k), \mathbf{U}(k))$$
$$\mathbf{Y}(k) = \rho(\mathbf{X}(k)) \tag{39}$$

The set of matrix equations described by (39) have to be solved at each time step in order to generate an evolving trajectory of the states based on a given input sequence. This type of trajectory generation is common in time-domain transient stability analysis where the generator responses are repeatedly simulated as function of time for many operating and contingency scenarios. This type of calculations is both repetitive and time consuming.

6.3.2 Neural network approach

In order to avoid the time consuming calculations associated with solving a non-linear state space model, a NN approach is proposed [55]. A multi-layer perceptron is trained to emulate the state space equations of the synchronous motor. The proposed learning and retrieving phases of the neural network are shown in figure 10.

Lets assume that there is a full state output, i.e $\mathbf{Y}(k) = \mathbf{X}(k)$. During training, patterns of $\mathbf{Y}(k)$ and $\mathbf{U}(k)$ are given to the NN with the corresponding target $\mathbf{Y}(k+1)$. These patterns are either randomly generated within the specified operating region or corresponds to points on a set of pre-selected training trajectories. In the retrieving phase, NN estimated state trajectories for different arbitrary input sequences.

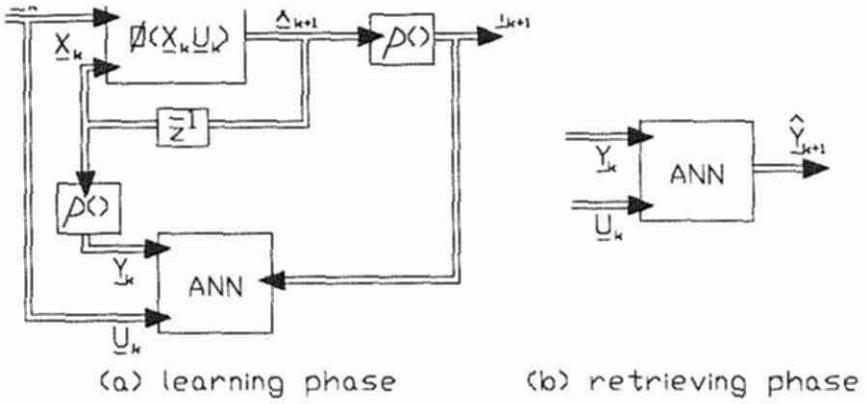


Figure 10. (a) Learning (b) Retrieving phases of the Synchronous m/c NN simulator

From [55] courtesy of IEEE, (C) IEEE,1989

The NN has 11 inputs consisting of the elements of vectors **Y**(k) and **U**(k) while the 7 outputs consist of the elements of vector **X**(k+1). The specific example given in reference [55] compares the NN model output against the actual motor states for a step change in the field voltage v_F. Close model following is observed for the given test.

6.3.3 Comments

Using an NN to simulate the synchronous machine dynamics can significantly speed up the transient stability calculations. However, accurately training a NN with 11 inputs and 7 outputs, to model the synchronous generator within a bounded operating space, is non-trivial. The training patterns should be sufficiently representative of the operating space so that the NN can accurately generalize its learning for an arbitrary input sequence. A recurrent neural network topology with its inherent temporal properties, is probably more suited for this type of application.

6.4 CONTINGENCY SCREENING

6.4.1 Introduction

A contingency in a power system, is an abnormal event (such as faults) which could be potentially damaging to power system components. Contingency screening is a relatively fast and approximate method of identifying whether a contingency may result in a violation of any of the operating constraints of the power system. The evaluation of the operating constraints due to a contingency is called *security assessment*, and is discussed in Section 6.7. Contingency screening helps select a critical set of potentially damaging events for more accurate analysis.

6.4.2 Problem description

Contingency selection, in its simplest form, is dealing with forming a list of contingencies which may result in steady state voltage or thermal limits violations in the post contingency power flow condition.

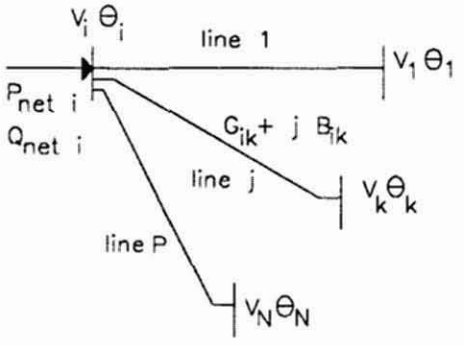


Figure 11. A simple power system

For a simple power system, such as that in figure II.4, the real and reactive power injections at the i^{th} bus can be expressed as,

$$P_{net\,i} = V_i \sum_k V_k [G_{ik} \cos \theta_{ik} + B_{ik} \sin \theta_{ik}] \quad (40)$$

$$Q_{net\,i} = V_i \sum_k V_k [G_{ik} \sin \theta_{ik} - B_{ik} \cos \theta_{ik}] \quad (41)$$

where $P_{net\,i}$, $Q_{net\,i}$ are the net real and reactive injections at i^{th} bus, and θ_{ik} is defined as

$$\theta_{ik} = \theta_i - \theta_k \text{ and } \mathbf{Y} = \mathbf{G} + j\mathbf{B}$$

Equations (40) and (41) can be solved for V_i and θ_i at all nodes. The power flow on line j between nodes i and k is then given by the equations,

$$P_{line\,j} = G_{ik} (V_i^2 - V_i V_k \cos \theta_{ik}) - B_{ik} V_i V_k \sin \theta_{ik} \quad (42)$$

$$Q_{line\,j} = - B_{ik} (V_i^2 - V_i V_k \cos \theta_{ik}) - G_{ik} V_i V_k \sin \theta_{ik} \quad (43)$$

$$S_{line\,j} = \sqrt{P_{line\,j}^{\,2} + P_{line\,j}^{\,2}} \quad (44)$$

The voltage magnitudes (V_i) obtained by solving equations (40) and (41) and line flows ($S_{line\,j}$) obtained from equation (44) constitute the so called *security variables*, which are the variables that decide the status of the system security. Any magnitude violation of these variables will result in an insecure system. Post-contingency security limits for bus voltages and line powers can be defined as,

$$\left.\begin{array}{l} V_U \geq V(\lambda) \geq V_L \\ S_{MAX} \geq |S(\lambda)| \end{array}\right\} \; Z_U \geq Z(\lambda) \geq Z_L \quad (45)$$

$z_i(\lambda)$ denotes the post contingency value of the i^{th} security variable corresponding to λ^{th} contingency. If all the above inequalities are satisfied the system is labelled as secure under the λ^{th} contingency.

time consuming and often computer intensive. To obtain a fast and approximate method for selecting key contingencies is known as *Contingency screening.* Contingency screening can be performed by several methods, among them are the *Distribution Factor* and the *Performance Index.*

With the Distribution Factor based method, the post-contingency Security variables are calculated by

$$\mathbf{S}(\lambda) = \mathbf{S}(0) + \mathbf{H}(\lambda)\ \Delta\mathbf{Y}(\lambda) \tag{46}$$

where $\Delta\mathbf{Y}(\lambda)$ corresponds to the change in a network due to the λ^{th} contingency. This could be either a change in network admittance due to a transmission line outage or the change in real power due to a generator outage. $\mathbf{H}(\lambda)$ is known as the transfer matrix whose elements are a set of factors which represent the sensitivity of the line flows to the above variations. Therefore, these partial derivatives can either be line outage distribution factors or generation shift factors corresponding to the type of the λ^{th} contingency.

In the Performance index (PI) based methods, an index associated with each contingency is calculated as follows:

$$PI(\lambda) = \frac{1}{2}\sum_{i} w_i (V_i^{(\lambda)} - V_{i\,ref})^2 + \frac{1}{2}\sum_{i} w_k (S_k^{(\lambda)}/S_{k\,MAX})^2 \tag{47}$$

where

w_i, w_k - weighting factors
$V_{i\,ref}$ - the desired value of V_i
$S_{k\,MAX}$ - the maximum rating of the k^{th} line current

Based on the value of $PI(\lambda)$ being less/greater than a certain threshold "TH", the contingency λ is classified as secure/insecure.

NN approach is proposed for contingency screening [56]. It is based on identifying the contingent branch overloads. The question of contingent voltages is not addressed in this study. This is known as active power contingency screening which is based on the DC load flow concept. By assuming that all voltage magnitudes V_i are equal to unity and that all angles θ_i are small ($\sin \theta_i = \theta_i$), equations (40) and (42) can be reduced and put in matrix notation as,

$$\mathbf{P}_{net} = \mathbf{B}\,\boldsymbol{\theta}$$
$$\mathbf{P}_{line} = \mathbf{T}\,\boldsymbol{\theta} \qquad (48)$$

For secure operation, $|P_{line\ k}| \leq S_{k\,MAX} \quad \forall \quad k \in \{lines\}$

A collection of NNs are trained where each NN is dedicated to a specific line outage. The inputs to the NN are:

$B_{ij} \quad \forall \quad i, j \in \{buses\}$ (post-contingency system)

$P_{net\ i} \quad \forall \quad i \in \{buses\}$,

and the outputs are:

$P_{line\ k} \quad \forall\ k \in \{lines\}$

binary flag $\in \{0, 1\}$ indicating secure/insecure status.

The concept was tested on a small power system with 6 buses and 9 lines. Training data was generated for 9 contingencies and 9 different discrete loading levels giving 81 different patterns. Only line contingencies were considered. A line contingency was simulated by halving the admittance between the corresponding buses. Each contingency was handled by a separate NN.

6.4.4 Comments

The proposed NN based contingency screening method is effective for a small power system. The minimum input dimension is equal to twice the number of buses plus the number of lines. Therefore, for a larger power system, the input variables can be excessively large. Under such cases, training a single NN for contingency screening will be difficult.

6.5. HARMONIC IDENTIFICATION AND EVALUATION

6.5.1 Introduction

Nonlinear loads and other harmonic producing loads have existed in power systems for many years. Today, the number of harmonic producing devices is rapidly rising due to the development of high power semiconductor switches and converters.

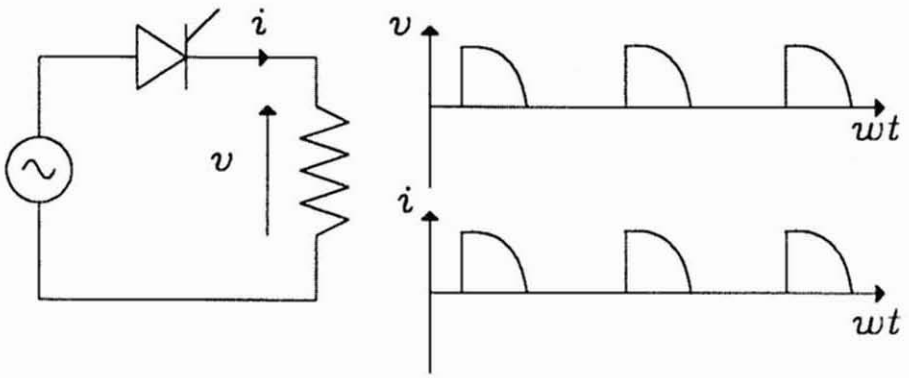


Figure 12. Phase controlled rectifier

Figure 12 indicates a simple phased controlled rectifier connected to a resistive load. The figure shows the load voltage and current. This nonsinusoidal load current, unless filtered, will be drawn from the power system. If a large number of such solid state devices and circuits are used, the nonsinusoidal current will give rise to harmonic

voltage drops across system components, thereby distorting the voltage wave form of the system. This can cause potentially damaging problems to the power system such as misoperation of protective relays, overheating of capacitor banks, increased losses in transmission systems, insulation failure in cables, increased losses in transformers and noise in communication circuits.

6.5.2 Problem Description

It is necessary to analyze and predict the behavior of current and voltage harmonics so that appropriate action could be taken to reduce their adverse effects. So far, model based analysis has been inaccurate and time consuming due to the nonlinearity of the harmonic components, the random behavior of harmonic signals and the wide variety of harmonic profiles of all solid state circuits.

6.5.3 Neural Network Approach

As a first step to identifying harmonic loads, a multi-layer perceptron was used to identify the type of harmonic load from among a set of pre-specified choices [57]. The training data for the NNs are generated by monitoring the current wave forms corresponding to each specific type of harmonic load. The fast fourier transform (FFT) of the digitized current wave form is used to produce the harmonic frequency spectrum. Different combinations of harmonic magnitudes and phases are then fed to the NN as inputs with the corresponding load type as the output.

Figure 13(a) shows the structure of the NN used to learn the harmonic/load relationship in the example given in reference [57]. The NN input are chosen among 31 harmonic magnitudes and phases. The output is one of 5 load groups, namely Personal Computer (PC), Television Set (TV), Video Tape Recorder (VTR), Fans(FNS) and

a NN is trained under each case with different combination of inputs.

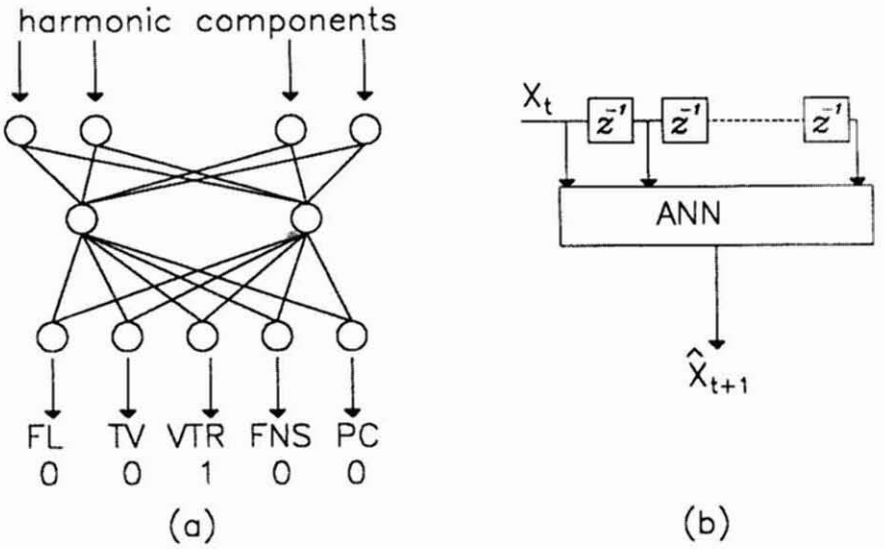


Figure 13. Identification of harmonic loads using NNs

From [57] courtesy of IEEE, (C) IEEE,1989

Case I: Magnitude of harmonic currents of order h = 1,2,.....31;

Case II: Magnitude of odd harmonic currents of order h = 1,3,5,....,31;

Case III: Magnitudes of harmonic currents of order h = 2, 3, 4, 5, 7, 9, 11 and phase angles of order k = 3, 5, 7, 9, 11;

Table 3 Correct classification as a percentage

Learning Set	Testing Set								
	Case I			Case II			Case III		
	A	B	C	A	B	C	A	B	C
A	90	92	86	96	73	68	100	100	100
B	94	99	78	84	98	95	100	100	100
C	61	99	97	92	99	97	90	96	100

From [57] courtesy of IEEE, (C) IEEE,1989

The ability to correctly classify the load based on the harmonic currents is investigated for the three cases. NNs are trained and tested using 3 separate data sets. Several NN architectures with different numbers of hidden layers are used to find the optimal NN design. Table 3 gives the performance under the 3 cases for the NN design with six hidden neurons.

It is clearly seen that NN trained under case III configuration has the best classification performance.

In subsequent development, a multi-layer perceptron was used to predict the magnitude of a selected harmonic in a time series form [58].

$$X^{(i)}(t+1) = f(X^{(i)}(t), X^{(i)}(t-1), \ldots\ldots, X^{(i)}(t-k))$$

where,

$X^{(i)}(t)$ - magnitude of the i^{th} harmonic at time t

A series of multi-layer perceptrons were trained to predict the magnitude $X^{(i)}(t+1)$ based on a time series of the past magnitudes. The structure of the NN is given in figure 13(b). The performance of the NN was compared with another nonlinear system identification algorithm known as the Revised Group Method of Data Handing (RGMDH). The NN identifier was observed to give an error distribution of lower variance compared with the RGMDH algorithm.

6.6 ALARM PROCESSING AND FAULT DIAGNOSIS

The control centers of a power system are continuously interpreting large number of alarms signals to determine the status of the system components and to evaluate the power system operation. This process is very complex because of two key reasons:

1. Alarm pattens are not unique to a given power system problem. Same fault may manifest in different alarm patterns based on the current topology and operating status of the power system.

2. Alarm pattern are likely to be contaminated with noise due to equipment problems, incorrect relay settings, interference, or miscalibrated meters.

Expert system techniques have been widely tested for analyzing alarm signals. The formulation of rules, however, requires precise definitions of the power system and its operational strategies which may widely vary depending on the utility. Therefore, expert system technique are known to suffer from a high customization effort.

6.6.1 Neural network approach

The ability of a power system operator to diagnose a system problem by analyzing a set of multiple alarms is a form of pattern recognition. Accurate classification of noisy alarm patterns is also a key shortcoming in most of the conventional techniques. Therefore, NN's with their ability to classify noisy patterns seems a logical choice for alarm processing. The NN is also capable of associating different alarm patterns to the same system fault by training the NN with a set of *information rich* data that represents different operating scenarios [59]. Figure 14 shows a block diagram showing the concept of intelligent alarm processing (IAP) using NNs.

Learning and retrieving phases of the IAP NN is presented in figure 14. The NN training set is generated by first creating a credible set of contingencies and then deriving the possible alarm patterns under

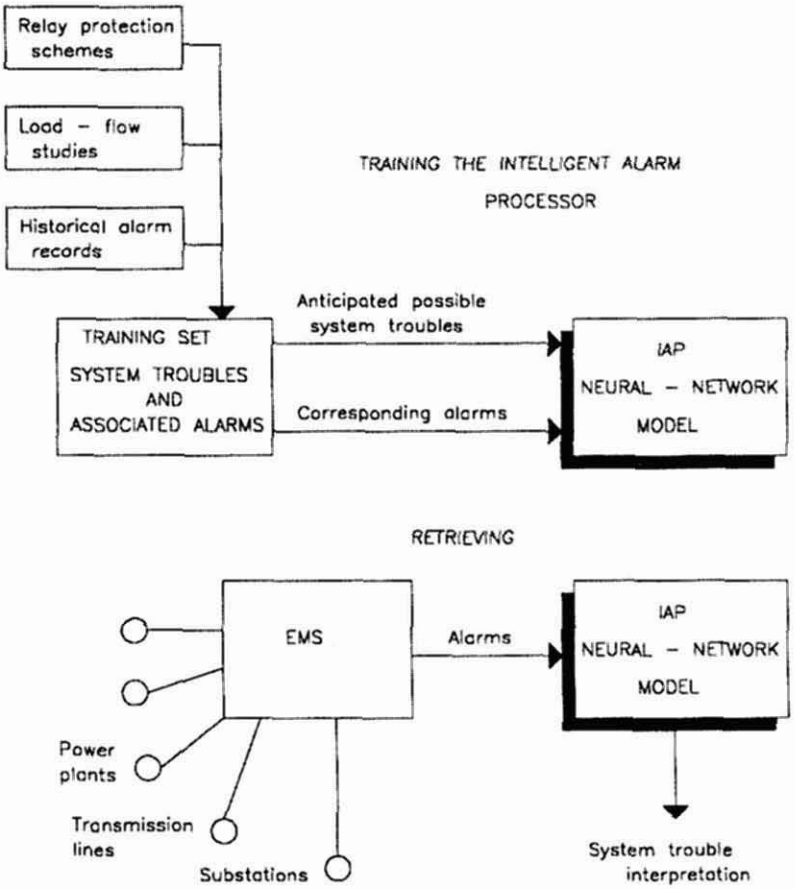


Figure 14. Concept of using NN for IAP

From [59] courtesy of IEEE, (C) IEEE,1989

each fault. These patterns are generated by the relay protection schemes and power flow analyses. These patterns are then used to train a multi-layer perceptron using back-propagation [59]. In the retrieving phase, incoming alarm patterns from the energy management system (EMS) are interpreted to predict the possible fault scenario.

The concept was tested on a 115kV/12kV substation for 65 different fault conditions with 99 bit alarm patterns [59]. It was also tested on the IEEE 30 bus system for 72 different bus and line fault conditions

was able to correctly classify all noiseless input patterns. NN was also able to correctly classify some of the noisy patterns. Noisy patterns were generated by randomly toggling certain bits of the original input pattern. It is also worth mentioning that when noisy patterns were incorrectly classified by the NN, the system operator, given the same noisy pattern, also reached the same wrong conclusion.

6.6.2 Comments

This is an area where NN seems to have great potential due to its intrinsic noise rejection and self learning capabilities. The reported study is preliminary in the sense that it does not take into account some of the characteristics of the alarms such as the order in which they are reported, the magnitude of the violations, and the behavior of alarms over a certain time period. A combination of several NNs' are proposed to capture the different system problem characteristics and the time-sequential significance of the alarm data in order to draw more definitive conclusions.

6.7 STATIC SECURITY ASSESSMENT

Static security assessment is defined as the ability of a power system to reach a state within the specified safety and supply quality following a contingency. The time period of consideration is such that the fast acting automatic control devices have restored the system load balance, but the slow acting controls and human decisions have not responded.

Static security assessment consists of three distinct stages. They are *contingency definition* (CD), *contingency selection* (CS), and *contingency evaluation* (CE). CD defines a contingency list to be

processed comprising of those cases [illegible]
is deemed sufficiently high. CS is the process that shortens the original long list of contingencies by removing the vast majority of cases having no violations. Two commonly used algorithms for CS are contingency screening contingency ranking. These methods were introduced in a previous section. There has also been an increasing effort towards applying expert systems to augment the analytical CS methods [51]. CE is the process where the selected contingencies are simulated on the power system in order to evaluate the post-contingency security variables. The resulting system attributes are checked for security violations. the calculations are performed on each of the list of ranked contingencies. The number of cases evaluated depends on the amount of time and computer resources available for the task.

6.7.1 Neural Network Approach

From a pattern recognition perspective, CE is a two class classification problem where the pre-contingency system attributes are used to predict post-contingency system security status. A multi-layer perceptron can be trained to perform this pattern classification [51]. But for a large power system, where a large number of attributes and operating conditions are needed to classify the system security, a single NN approach may be an enormous computational exercise. One way of reducing the dimensional complexity is to use a modular approach where the security problem is divide into smaller tasks or reduced topology. A modular NN can then be used to handle each task or topology.

Figure 15 shows a possible modular approach to large power system problem. A specific NN for predicting security status under a specific contingency is proposed. This is necessary due to the variations in which a contingency manifests itself based on the nature, location and clearing strategy. Furthermore, for a given contingency, the mechanisms leading to line and voltage violations are fundamentally

lifferent. Line violations are brought about by real power overflows, vhile voltage violations are brought about by an excess or a leficiency of reactive power. Therefore, separate NNs are trained for ıssessing line and voltage violations under the same contingency.

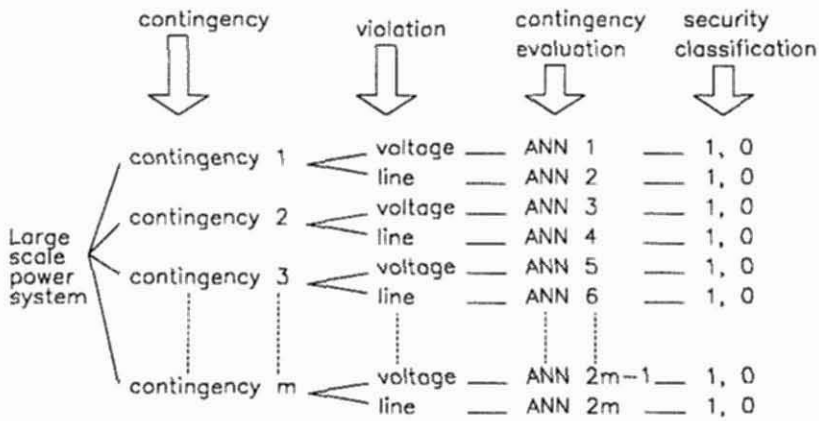


Figure 15. The proposed NN's approach to SSA

.7.2 Generating training Data

:ach training pattern for a particular contingency is selected to orrespond to a different power system loading condition. These atterns can be generated by perturbing each of the real and reactive oads with a uniformly distributed random variable within the pecified range. The perturbations are uncorrelated. The pre-ontingency system states $\mathbf{X}^0$, are given by the solution to the power ow equations,

$$\mathbf{f}^0 (\mathbf{X}^0, \mathbf{U}, \mathbf{L}) = \mathbf{0} \tag{49}$$

where,

L - Load demand
U - Control vector (such as s generator real power and voltage)
$(.)^0$ - Pre-contingency value of "."

U and **L** are inputs to equation (49). The control vector **U**, is selected to minimize an objective function $F(\mathbf{X}^0,\mathbf{U})$ which represent a pre-contingency optimal dispatch strategy,

$$F(\mathbf{X}^0, \mathbf{U}) = \sum_{i}^{N_g} (C_{2i} Pg_i^2 + C_{1i} Pg_i + C_{0i}) \tag{50}$$

where C_{2i}, C_{1i} and C_{0i} are coefficients. Pg_i is generation of machine i. The control vector **U** is given by

$$\mathbf{U} = [Pg] = [Pg_1, Pg_2, \ldots\ldots Pg_{Ng}]^T \tag{51}$$

To minimize the cost index of equation (50), a Lagrangian function is introduced,

$$L(\mathbf{X}^0, \mathbf{U}, \mathbf{L}, \lambda) = F(\mathbf{X}^0, \mathbf{U}) + \lambda^T f^0(\mathbf{X}^0, \mathbf{U}, \mathbf{L}) \tag{52}$$

where λ is the lagrange multiplier vector. The minimization process is iterative with respect to $\mathbf{X}^0$, **U**, and λ. A gradient based search technique is used for the process. The control vector **U** is bounded by the constraint,

$$\mathbf{U}_{min} \leq \mathbf{U} \leq \mathbf{U}_{max} \tag{53}$$

Based on generator ratings and system considerations. A solution to this constrained optimization problem should satisfy the Kuhn-Tucker corner conditions. This procedure is commonly known as an Optimal Power Flow (OPF).

after the system states $\mathbf{X}^k$ in the load flow equations is obtained,

$$\mathbf{f}^k (\mathbf{X}^k, \mathbf{U}^k, \mathbf{L}^k) = 0 \tag{54}$$

where,

$\mathbf{X}^k$ - post-contingency state vector
$\mathbf{U}^k$ - post-contingency control vector
$\mathbf{L}^k$ - post contingency demand

In this study, $\mathbf{L}^k$ is assumed to remain equal to its pre-contingency value. The post-contingency control vector $\mathbf{U}^k$ is calculated based on the type of fault: for a sizable disruption of real power, such as the loss of a tie-line or a generator, the outputs of the remaining generators are adjusted on the basis of their individual speed-droop characteristics; or else, only the swing bus absorbs the slack generation. The droop of each individual generator is assumed to be proportional to its maximum ratings. Therefore, if tripping of a tie line causes a surplus of real power Δp, the individual generator power settings are adjusted as,

$$\mathbf{U}^k = \mathbf{U} - \Delta\mathbf{U} \tag{55}$$

where,

$$\Delta U = \frac{\Delta p}{\sum Pg_{(max)\,i}} \; Pg_{(max)} \qquad (i = 1,...N_g)$$

The bus voltages and line currents are then checked against their safe operating limits specified by,

$$\mathbf{G} (\mathbf{X}^k, \mathbf{U}^k) \leq \mathbf{0} \tag{56}$$

is labeled secure if no violations are found, otherwise the power system is insecure.

6.7.3 Feature selection

Each pattern vector should contain all possible variables affecting system security such as load powers, bus voltages, line flows etc. With feature extraction, the dominant variables are selected. By this method, the dimension of the pattern vectors can be substantially reduced. For example, assume a pattern with D dimensional normalized measurement vector,

$$Y = [y_1, y_2, \ldots, y_D]^T$$

Assume that the dominant number of variables is $d << D$. The security classification is then based on these d components. The heuristic notion of interclass distance is used to accomplish this task. Given a set of patterns with dimension D, it is reasonable to assume that the pattern vectors for each of the two classes (secure/insecure) occupy a distinct region in the observation space. The average pairwise distance between the patterns is a measure of class separability in the region with respect to the particular variable. The following function F provides a measure of the importance in each variable.

$$F_j = \frac{|m_{js} - m_{ji}|}{\sigma_{js}^2 + \sigma_{ji}^2} \qquad 0 < j \leq D \tag{57}$$

where,

$$m_{js} = \frac{1}{N_s} \sum_{l} y_{js} \qquad 0 < l \leq N_s$$

$$m_{ji} = \frac{1}{N_i} \sum_{n} y_{ji} \qquad 0 < n \leq N_i$$

$$\sigma_{js}^2 = \frac{1}{N_s} \sum_{i} (y_{js} - m_{js})^2 \qquad 0 < l \leq N_s$$

$$\sigma_{ji}^2 = \frac{1}{N_i} \sum_{i} (y_{ji} - m_{ji})^2 \qquad 0 < n \leq N_i$$

The subscript 's' stands for 'secure' while 'i' stands for 'insecure'. N_s and N_i indicate the number of secure and insecure patterns that form the training set. m_{js} and m_{ji} denote the corresponding in-class means of the j^{th} attribute. σ_{js} and σ_{ji} are the standard deviations. The variables are ranked according to the following steps.

1. Calculate Fj $\forall$ $0 < j \leq D$
2. Rank all Fj in a descending order
3. Go to the 1^{st} ranked variable.
4. Calculate correlation coefficients (CC) of all lower ranked variables with respect to the 1^{st} ranked variable. The CC is defined as,

$$CC_{ij} = \frac{E[(y_i - m_i)(y_j - m_j)]}{\sigma_i \sigma_j} \qquad 0 < j \leq D$$

5. Eliminate all lower ranked variables which have a $|CC| > 0.9$
6. Go to the next highest ranked variable and go to step 4.

The process is repeated until all the variables are ranked or discarded. The resulting ordered list of variables are considered to be key features in training the NN classifier.

6.7.4 Training the neural network

In order to evaluate the performance of the trained NN classifier, the following definitions are introduced.

False Alarm: When a true secure operating point as described by the oracle, is classified as insecure by the NN.

False Dismissal: When a true insecure operating point as described by the oracle, is classified as secure by the NN.

The following percentages are also introduced to obtain a quantitative measure of the classification performance. The percentage false alarms, false dismissals and false classifications are calculated using the following definitions:

$$\%\ \text{false alarms} = \frac{\#\ \text{of false alarms}}{\text{total true secure states}} \times 100$$

$$\%\ \text{false dismissals} = \frac{\#\ \text{of false dismissals}}{\text{total true insecure states}} \times 100$$

$$\%\ \text{false classifications} = \frac{\text{false alarms + false dismissals}}{\text{true secure + true insecure states}} \times 100$$

6.7.5 Tests results

The concept is tested on the study system of Figure 16. It includes 4 generators ($N_g = 4$), 8 loads ($N_b = 8$) and 16 transmission lines ($N_l = 16$). The influence of the external networks is modelled by a bi-directional power flow at boundary buses #9 and #10 respectively.

Table 4 shows the operating point and the allowed perturbation in the real and reactive loads at each bus. The tie line flow is considered to be either positive or negative depending on the direction of flow.

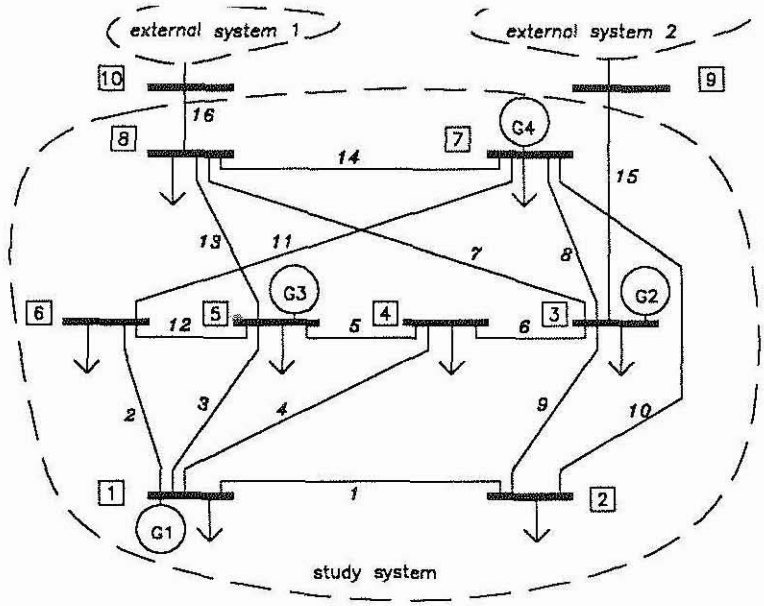


Figure 16. The test power system

Table 4. The range of load parameters

bus #	bus type	real load limits (puMW)	reactive load limits (puMVar)
1	slack	9.0 - 11.0	0.0 - 1.0
2	load	11.2 - 16.8	0.0 - 1.0
3	generation	13.5 - 16.5	0.0 - 1.0
4	load	14.0 - 16.0	0.0 - 1.0
5	generation	13.5 - 16.5	0.0 - 1.0
6	load	15.4 - 28.6	9.1 - 16.9
7	generation	9.0 - 11.0	0.0 - 1.0
8	load	0.0 - 2.0	5.0 - 15.0
9	boundary	-7.5 - 7.5	-7.5 - 7.5
10	boundary	-7.5 - 7.5	-7.5 - 7.5

In this test, the tripping of tie line #16 is investigated. A single pre-contingency pattern contains 54 different attributes including all the real and reactive generation (P_{gi},Q_{gi}), real and reactive loads (P_{bj},Q_{bj}), all the bus voltage magnitudes (V_{bj}) and all the line currents (I_{Lk}) in

the system. The key features (variables) for training [illegible]
selected as described earlier. Six features were used for NN training: Q_{b8}, V_{b8}, Q_{g2}, Q_{b10}, I_{L7}, I_{L14}. The training and testing statistics of the NN are given in Table 3.5.

Table 5 Training and testing statistics for the NN in Case I.

Network architecture & training information		Testing statistics	
inputs	6	testing data	500
outputs	1	true secure patterns	346
hidden layers	1	true insecure patterns	154
hidden neurons	6	false alarms	9
iteration step	0.05	false dismissals	4
momentum factor	0.01	% false alarms	2.601
training patterns	1550	% false dismissals	2.597
iteration cycles	2000	%false classifications	2.600

Table 6. Training and testing statistics for the NN in Case II.

Network architecture & training information		Testing statistics	
inputs	7	testing data	500
outputs	1	true secure patterns	160
hidden layers	1	true insecure patterns	340
hidden neurons	6	false alarms	3
iteration step	0.10	false dismissals	2
momentum factor	0.01	% false alarms	1.875
training patterns	1550	% false dismissals	0.588
iteration cycles	1000	%false classifications	1.000

In the second case, the contingency is the tripping of the transmission line between buses #5 and #6. The training data are generated similar to the previous case. The input attributes for the NN are selected by the feature selection algorithm described earlier. The features Q_{b6}, Q_{g1}, Q_{g3}, Q_{g4}, I_{L3}, I_{L11} and I_{L12} are selected. The training and testing statistics for the NN in case II are given in table 6.

6.7.6 Comments

The feature selection criteria is based on the heuristic notion of inter-class distance. Selection of features based on their individual merit does not always ensure accurate selection of the discriminatory information.

Selection of loads based on a random number generator is not realistic. Load variations in an actual power system consists of a superposition of correlated and uncorrelated components. In this study, no provision to handle any topological variations brought about due regular powers system operating characteristics.

Security assessment by the above mentioned method would require a NN for each possible contingency. To cover all possible contingencies, a large number of NN may be needed. The implementation of such a scheme is practical only when NN hardware becomes available.

6.8 DYNAMIC SECURITY ASSESSMENT

In dynamic security, or small signal stability analysis, the power system model is linearized around a selected operating point and the corresponding system eigen values evaluated to predict system stability. For a power system to be evaluated at all possible operating conditions, the linearization and eigen value analysis has to be repeated for all the cases. This is a time consuming process that poses a challenge to performing dynamic security assessment (DSA) on-line. Thus NN may provide a potential avenue toward achieving this objective.

6.8.1 Problem Description

In dynamic security assessment, the power system stability is evaluated via frequency domain analysis. The power system is divided into a

study system and an external system. The external system can be replaced by a dynamic equivalent models while the study system is modelled in detail. The model of the entire power system is developed using the small signal analysis. The eigen values of the system are then computed and assessed at various operating conditions [16]. The linearized state space model of the power system can be considered as an oracle for NN training. The linearized model is derived by combining the set of state and algebraic equations listed in section 6.3 for all generators in the study area of the power system. The composite linearized state spaces equation take the form,

$$\frac{d\,\Delta \mathbf{X}}{dt} = \mathbf{A}(\mathbf{X}_0,\mathbf{U}_0)\,\Delta \mathbf{X} + \mathbf{B}(\mathbf{X}_0,\mathbf{U}_0)\,\Delta \mathbf{U}$$

where $\mathbf{X} = \mathbf{X}_0 + \Delta \mathbf{X}$ and $\mathbf{U} = \mathbf{U}_0 + \Delta \mathbf{U}$ are the state and input vectors for the system. The stability of the system is determined by calculating the eigen values of the system matrix $\mathbf{A}(\mathbf{X}_0,\mathbf{U}_0)$. Any eigen value with a non-negative real component is unstable mode of operation.

The stability of the power system as described above is heavily dependent on the operating condition and topology of the power system. The computation of the eigen values of a large system is a time consuming process that inhibits the on-line applications.

6.8.2 Neural Network Approach

Training data for dynamic security assessment can be generated off-line by using an oracle. Training data can also include measurements of previous assessments. A multi-layer perceptron is trained using back-propagation to learn the dynamic security status with respect to a selected set of variables **U** within a defined operating space [16]. A test example of 9 bus, 3 generators was used to validate the method. For simplicity, 3 independent input variables were selected as inputs

generator and complex power output ($S = \sqrt{P^2 + Q^2}$) of another generator. All other parameters were assumed to be constant. In the retrieving (testing) phase, 2-dimensional dynamic security contours of P,Q are obtained by fixing S at arbitrary values. The NN generated contour compared well with the actual contour obtained using the oracle [16].

6.8.3 Comments

The dimensionality of the security contours is a function of the size of the system under investigation. In a high dimensional operational space where a combination of correlated and uncorrelated variables forms the input space, the development of a NN based system for assessing dynamic security is a challenging problem.

6.9 CAPACITOR CONTROL

6.9.1 Introduction

Compensating the reactive power flow in utility systems is an area of continuous development. Reactive power has limiting effect on the operation of the power system due to the line losses and unnecessary equipment load. The reactive power compensation can be viewed as an optimization problem where several optimum sizes of capacitors can be placed at optimum locations to minimize a cost index such as line (or system) losses. This is a complex nonlinear optimization problem. Many techniques have previously been used such as gradient methods, linear, nonlinear and dynamic programming and expert system methods.

Consider a uniformly loaded feeder of 'h' length as shown in figure 17(a).

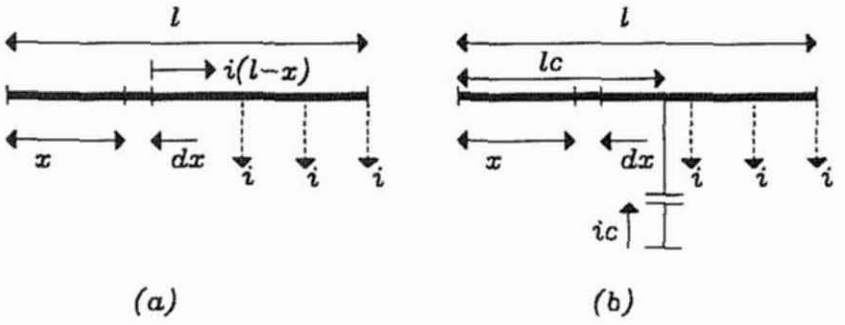


Figure 17. (a) Uniformly loaded cable (b) capacitor compensated cable

The 3Φ power loss in an elemental length dx due to the resistance of the cable is given by

$$d L_{3\Phi} = 3 r i^2 (h - x)^2 dx \qquad (58)$$

where

i - current per unit length
r - resistance per unit length
h - length of the cable

The total 3Φ power loss (w) along the feeder is given by

$$L_{3\Phi} = 3 r i^2 \int_0^h (h - x)^2 dx = r i^2 h^3 = R_T I_T^2 \qquad (59)$$

where

= r h - the total resistance of the cable

i h - the total load current drawn in to the cable

ming that the load is cyclic with a period of T hours, the total gy loss (wh) can be calculated as,

$$_{3\Phi} = \int_0^T L_{3\Phi}\, dt = R_T \int_0^T I_T^2\, dt = R_T I_{T\,MAX}^2 L_s T \qquad (60)$$

consider the installation of a capacitor bank at location h_c as n in figure 17(b). The 3Φ power loss (w) can now be modified as,

$$L_{3\Phi} = 3r \int_0^{h_c} (i(h-x) - i_c)^2\, dx + \int_{h_c}^{h} i^2 (h-x)^2\, dx$$

$$L_{3\Phi} = 3r [h^3 i^2/3 + (h_c - 2h_c h) i_2 h_c + i_2^2 h_c] \qquad (61)$$

e,

reactive component of current i

capacitive current provided by the bank

modified energy loss can be similarly calculated. The cost saving to installing capacitors to decrease energy and power losses is ı by,

$$C = K_1 \Delta E_{3\Phi} + K_2 \Delta L_{3\Phi} \qquad (62)$$

e K_1 and K_2 are two cost factors. The optimum size and location e capacitor bank can be explicitly calculated by setting the partial atives $(\partial \Delta C/\partial i_c)$ and $(\partial \Delta C/\partial l_c)$ to zero.

However, in a real power system, the situation is not that straight forward. The distribution system can have multiple capacitors with discrete tap settings. The load current may not be uniformly distributed and the load variations at different parts of the distribution network may be uncorrelated. Hence, no common load cycle can be identified. Also, other economic considerations such as depreciation, return on investment etc. may have to be included in the optimization model. In order to deal with these constraints, linear and nonlinear programming techniques can be employed. Expert systems also have been looked at as a possible alternative. However, solution accuracy and computational time are a major concern in most of these techniques.

6.9.3 Neural Network Approach

The NN assisted approach to the solution of capacitor control problem is expected to drastically reduce the calculation times and enable on-line adjustments. A specific example in the control of capacitors on a radial distribution system is addressed in [62]. The test power system is given in figure 18(a). The location of the capacitors are assumed pre-determined. The entire power system is divided into six subsystems, each with uniformly distributed loads marked by dotted lines. There are 6 measurement locations marked by M_1 through M_6. P, Q flow and the voltage magnitude $|V|$ are monitored at the capacitor locations. The aggregated load in each subsystem is assumed to take one of 4 feasible levels at 50%, 70%, 85% and 100% of the peak load with proportional variations in reactive power. The current tap setting of each capacitor is also known. The objective is to use 3 measurement quantities ($P, Q, |V|$) at locations M1 through M6 and the current tap settings of the capacitors C1 through C5 in order to calculate the optimum tap settings for the 5 capacitors.

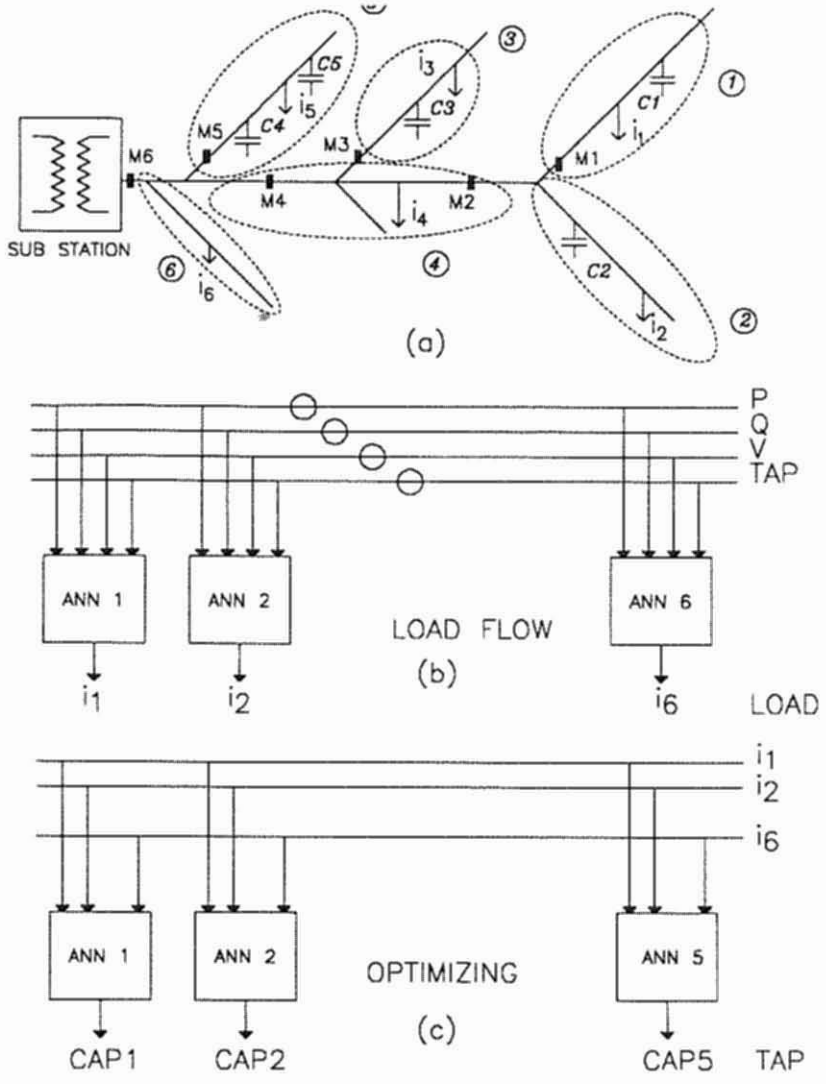


Figure 18. The capacitor control through NNs

From [62] courtesy of IEEE, (C) IEEE,1989

The problem is solved in two stages. Both stages use multi-layer perceptrons trained by back-propagation. In stage I, 6 NNs, shown in figure 18(b), are trained to perform a power flow calculation. The training data for the this stage are the P, Q, |V| measurements for all feasible combinations of load levels and capacitor settings. The

output of the NNs are uniform load currents i_1 through i_6. In the figure, the circles placed on the lines indicate multiple measurements.

In stage II, the outputs of the NNs of stage I (i_1 through i_6) are used to train 5 NNs as shown in figure 18(c). In this stage, the NNs are trained to select the optimum tap setting of all 5 capacitors. Training data for stage II are generated by the optimizing algorithm. Different combinations of aggregated loads on the 6 subsystems are assumed. In the retrieving phase, the NN estimated the optimum tap settings.

6.9.4 Comments

Perhaps one of the most significant contribution of this work is the partitioning of the overall problem into in to smaller subproblems. Then individual NN's are used where each id dedicated to solve a specific subproblem. This modular approach facilitates faster and simpler training of the NN's. Also it simplifies the maintenance (updating) of the NN's.

6.10 TOPOLOGICAL OBSERVABILITY

Topological observability is a method for selecting certain locations in the power system where measurements can be collected in order to observe the entire power network. Once the topological observability is concluded, a *State estimation* technique is used to filter any errors in the measurements and to estimate the states at locations where measurements can not be obtained.

6.10.1 Problem Description

Consider the nonlinear measurement model for the power system consisting of (n) states and (m) measurements, and m > n

$$\mathbf{z} = \mathbf{h}(\mathbf{x}) + \mathbf{v}_z \tag{63}$$

where

$\mathbf{z}$ measurement vector (m x 1)
$\mathbf{x}$ state vector (n x 1)
$\mathbf{v}_z$ measurement noise vector (m x 1)

The maximum likelihood estimate of $\mathbf{x}$ is obtained by minimizing the cost function,

$$J(\mathbf{x}) = (\mathbf{z} - \mathbf{h}(\mathbf{x}))^T \mathbf{R}^{-1} (\mathbf{z} - \mathbf{h}(\mathbf{x})) \tag{64}$$

where $\mathbf{R} = E[\mathbf{v}_z \mathbf{v}_z^T]$, and E[.] is the expectation of "."

To obtain the optimal solution of $\mathbf{x}$, the first derivative of the cost function is set to zero

$$\frac{\partial J(\mathbf{x})}{\partial \mathbf{x}} = (\mathbf{H}_{\mathbf{x}^\wedge})^T \mathbf{R}^{-1} (\mathbf{z} - \mathbf{h}(\mathbf{x}^\wedge)) = 0 \tag{65}$$

where $\mathbf{H}_{\mathbf{x}^\wedge} = \frac{\partial \mathbf{h}}{\partial \mathbf{x}^\wedge}$

Where $\mathbf{x}^\wedge$ is the estimated value of $\mathbf{x}$.
For a linearized de-coupled measurement model, the measurement equation takes the form

$$\mathbf{z} = \mathbf{H}\mathbf{x} + \mathbf{v} \tag{66}$$

where $\mathbf{H}$ is the linearized measurement matrix. In this case, the optimal state estimate $\hat{\mathbf{x}}$ can be proved to be as follows

$$\hat{\mathbf{x}} = (\mathbf{H}^T \mathbf{R}^{-1} \mathbf{H})^{-1} \mathbf{H}^T \mathbf{R}^{-1} \mathbf{z} \tag{67}$$

The power system is said to be topologically observable if $\mathbf{H}$ has a rank of (n). The question to be answered is whether or not a system is observable through a given types and locations of measurements. If

the system is not observable, then the question is what other measurements are required to make it observable.

Among the commonly used techniques for topological observability are: heuristic methods, and graph theory methods. These techniques are associated with different degrees of accuracy and computational effort. In an effort to find the most efficient way to handle the combinatorial complexity of this problem, a NN approach has been looked at as a possible alternative.

6.10.2 Neural network approach

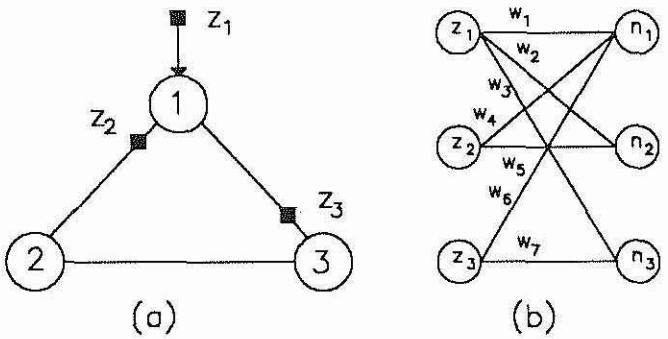


Figure 19. (a) 3 node test system (b) corresponding measurement allocation graph

From [63] courtesy of IEEE, (C) IEEE,1989

The proposed method [64], starts from a graph theoretic definition of topological observability and converts it to an integer programming problem. It is then solved by using a Hopfield neural network model. Based on the converged solution of the Hopfield model, it can be determined whether the system is observable under the assumed measurement placement.

The topologically observable measurement allocation graph is defined as having a structure where, *a node has at least one measurement and a measurement is assigned to at least one node.*

ire 19 shows a simple three node example. Let G be a matrix
se elements represent the relationship between the meters and
nodes in the allocation graph. Elements of G can be defined as

$$G_{ij} = \begin{cases} a_k: \text{ meter } z_i \text{ covers node j: } 1 \geq a_k \geq 0: \ a_k \text{ - integer} \\ 0: \text{ meter } z_i \text{ does not cover node j} \end{cases} \quad (68)$$

s important to note that the value of a_k can take only values {0,1}
e the hopfield network iterates to a solution. But for the
ulation of G, the values of a_k are assumed to be bounded within
interval {0,1}.) For the three node power system shown in figure
ı), a graph G can be written as

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix}$$

far as graph G is concerned, to meet the topological observability
dition, the following conditions must be met

Measurement z_i is assigned to only one node. Hence, at least one value of a_k in each row should be equal to one, i.e

$$\sum_k a_k \geq 1 \quad \text{where } k \in \text{row(i)} \quad (69)$$

Node n_i has at least one measurement. Hence, at least one value of a_k in each column should be equal to one, i.e

$$\sum_j a_j \geq 1 \text{ where } j \in \text{colm(i)} \quad (70)$$

eliminated by introducing slack variables. Once the slack variables are eliminated from the basis, the following combined model results.

$$\mathbf{D\,a} = 1 \qquad (71)$$

The vector **a** contains elements a_k while elements of matrix D denoted by d_{ij} take values {0,1} based on the constraint equations. Solving equations (71) with the constraints ($1 \geq a_k \geq 0$: a_k - integer) is equivalent to minimizing the cost function

$$E = \frac{1}{2}[p_1 G_1 + p_2 G_2] \qquad (72)$$

where,

$$G_1 = \sum_{i=1} (1 - \sum_{j} d_{ij} a_j)^2 \qquad (i = 1,...m+n \quad j = 1,...n_w)$$

$$G_2 = \sum_{i} a_i (1 - a_i)$$

p_1 and p_2 are to weighting factors whose choice is arbitrary. The cost index G_1 goes to zero when the matrix equation in (72) is satisfied while index G_2 forces the values of a_k to be either 0 or 1. Therefore both G_1 and G_2 should ideally converge to zero.

By equating the coefficients of the energy function in equation (72) and the generic hopfield energy function given by,

$$E(l) = -\frac{1}{2} \sum_{i} \sum_{j} w_{ij}\, a_i(l)\, a_j(l) - \sum_{i} \theta_i\, a_i(l) \qquad (i,j = 1,.N)$$

(73)

can be proved that

$$w_{ij} = -p_1 \sum_k d_{ki} d_{kj} + p_2 d_{ij} \tag{74}$$

$$\theta_i = p_1 \sum_k d_{ki} - \frac{p_2}{2} \tag{75}$$

'here k = 1,....,m+n

'he solution procedure is as follows. First, the weights **w** and the ıresholds θ of the hopfield network are set according to equations 74) and (75) respectively. Starting from an arbitrary set of a_i within ıe range [0,1], the hopfield network is allowed to converge to a stable)lution where the energy function is sufficiently minimized. The alues of a_i are then taken from the converged hopfield network and ıbstituted in order to find a redundancy factor R given by

$$R = \sum_k a_k - n \tag{76}$$

'here k = 1,....,n_w (n_w - number of variables a_i's), and n is the umber of nodes for a set of measurement placements, the system is ıid to be topologically observable if the value of R given by equation 79) is a non-negative quantity.

.10.3 COMMENTS

)ue to the poor attractor dynamics of the hopfield network, the)lution was seen to converge to a local minimum thereby giving an ıcorrect observability picture. The convergence was also largely ependent on the choice of the parameter p_1 and p_2. Subsequently, ıe same formulation was solved using a Boltzman machine to obtain

better results. Improved convergence properties were observed in this formulation.

6.11 IDENTIFICATION AND CONTROL OF A DC MOTOR

6.11.1 Introduction

An electric drive system is considered "high performance" when the rotor position or shaft speed can be made to follow a pre-selected track at all time [66,67]. A track (or trajectory) is a desired time history of the particular controlled variable. This type of high performance drive system is essential in applications such as robotics, actuation and guided manipulation where precise movements are required [66,67].

A fast controller is an essential feature of such a drive system [66,67]. The objective of a speed controller is to manipulate the terminal voltage in such a manner as to make the rotor speed follow a specified trajectory with minimum deviation. The resulting control signal should be reasonably well behaved in order to be implemented using a general purpose converter [67].

One of the main difficulties with conventional tracking controllers for electric drives is their inability to capture the unknown load characteristics over a widely ranging operating point. This makes the tuning of the respective controller parameters difficult [66-69]. There are many techniques that can overcome this problem. In adaptive control for instance, this problem is overcome by identifying the overall behavior of the motor using a linear parametric (ARMAX) model at prespecified time intervals [66,67,69]. But load torque is usually a nonlinear function of a combination of variables such as speed and position of the rotor. Hence identifying the overall nonlinear system through a linearized model around a widely varying

which can lead to instability or inaccurate performance [69].

The ability of NNs to learn large classes of nonlinear functions is well known [33,70]. It can be trained to emulate the unknown nonlinear plant dynamics by presenting a suitable set of input/output patterns generated by the plant [70-74]. Once system dynamics have been identified using an NN, many conventional control techniques can be applied to achieve the desired objective. Among these techniques is indirect model reference adaptive control (MRAC) [69,70] which is specifically useful in trajectory control applications. An attempt has been made to merge the accuracy of MRAC systems and the calculation speed of NNs to come up with a trajectory controller for a dc motor.

This section introduces a NN based identification and control system [65]. It is formulated as a MRAC system for trajectory control of a dc motor. No prior knowledge of the load dynamics is assumed. The main purpose of the controller is to achieve trajectory control of speed when the load parameters are unknown.

6.11.2 Problem Description

The dc motor is the obvious proving ground for advanced control algorithms in electric drives due to the stable and straight forward characteristics associated with it. It is also ideally suited for trajectory control applications as shown in references [66-68]. From a control systems point of view, the dc motor can be considered as a SISO plant, thereby eliminating the complications associated with a multi-input drive systems.

The dc motor dynamics are given by the following two equations

$$K\, i_a(t) = J\,[dw_p(t)/dt] + D\, w_p(t) + T_L(t) \qquad (78)$$

where,

$\omega_p(t)$ - rotor speed rad/s
$V_t(t)$ - terminal voltage v
$i_a(t)$ - armature current A
$T_L(t)$ - load torque Nm
J - rotor inertia Nm^2
K - torque & back emf constant NmA^{-1}
D - damping constant Nm s
R_a - armature resistance Ω
L_a - armature inductance H

The load torque $T_L(t)$, can be expressed as

$$T_L(t) = \Psi(\omega_p) \qquad (79)$$

where the function $\Psi(\omega_p)$ depends on the nature of the load. The exact functional expression of $\Psi(\omega_p)$ is assumed to be unknown.

In order to derive training data for the NN and to apply the control algorithms, a discrete-time dc motor model is required. Let's assume the load torque $T_L(t)$ of equation (79) to be

$$T_L(t) = \mu\, \omega_p^2(t)\, [\text{sign}(\omega_p(t))] \qquad (80)$$

where μ is a constant. The function is set up so that the direction of load torque always opposes the direction of motion. The motivation for choosing this particular function is that it is a common characteristic for most propeller driven or fan type loads. However, it

is important to note that the choice of load torque is completely arbitrary and does not influence the proposed control algorithm.

The discrete-time model is derived by first combining equations (77), (78) and (80) and then replacing all continuous differentials with finite differences. The resulting state space equation is

$$\begin{aligned} \omega_p(k+1) &= \alpha\,\omega_p(k) + \beta\,\omega_p(k-1) \\ &+ \gamma\,[\mathrm{sign}(\omega_p(k))]\omega_p^{\,2}(k) \\ &+ \delta\,[\mathrm{sign}(\omega_p(k))]\omega_p^{\,2}(k-1) + \xi\,V_t(k) \end{aligned} \tag{81}$$

where α, β, and ξ are constant values based on the motor parameters J, K, D, R_a, L_a, and the sampling period T, while γ and δ in addition to being functions of the above parameters are also functions of μ. The value k denotes the k^{th} time step.

A separately excited dc motor with name plate ratings of 1 hp, 220 v, 550 rpm is used in all simulations. Following parameter values are associated with it.

J = 0.068 Kg m^2
K = 3.475 Nm A^{-1}
R_a = 7.56 Ω
L_a = 0.055 H
D = 0.03475 Nm s
μ = 0.0039 Nm s^2
T = 40 ms

6.11.3 Identification and Control using NN

Figure 19 shows the basic concept of identification and control of the dc motor using an NN. The scheme is very similar to indirect model

reference adaptive control [69,70] where the motor is first identified between a combination of its input and output variables using an NN. The weights from the trained NN identifier are then used in the controller to calculate the terminal voltage which will asymptotically drive the motor shaft speed $\omega_p(k)$ towards the reference model output $\omega_m(k)$.

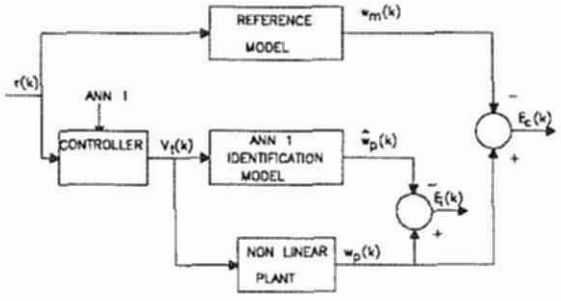


Figure 20. NN based identification and control system

From [65] courtesy IEEE (C) IEEE, 1991

In the case presented in figure 20, the quantities $\varepsilon_i(k)$ and $\varepsilon_c(k)$ are defined as the identification and tracking error respectively. The objective in identification is to minimize

$$[\varepsilon_i(k)]^2 \ \forall \ kT \in [0, t_f]$$

while the control strategy is to calculate a suitable terminal voltage $V_t(k)$ which minimizes

$$[\varepsilon_c(k)]^2 \ \forall \ kT \in [0, t_f]$$

where $[0, t_f]$ is the time window under review. The desired behavior of the motor is specified through a stable reference model. For a desired speed trajectory $\{\omega_m(k)\}$, a bounded control sequence $\{r(k)\}$ could be derived by using the reference model. This forms the activation signal for the control system.

determined by the consequent design of the controller. This is because, as seen from figure 20, the controller uses information from the identified model to predict the controlled input.

The dc motor characteristics are identified by presenting a set of input/output patterns to the NN and by adjusting its weights accordingly by using back error propagation. The extent of training depends on the degree of complexity of the dynamics to be learned. One of the first tasks in training an NN is to define a region of operation with respect to its input/output variables. In conforming with the mechanical and electrical hardware limitations of the motor, and with a hypothetical operating scenario in mind, the following constrained operating space is defined.

$$-30.0 < \omega_p(k) < 30.0 \quad \text{rad/s}$$

$$| \omega_p(k-1) - \omega_p(k) | < 1.0 \quad \text{rad/s}$$

$$| V_t(k) | < 100 \text{ v}$$

Two different identification topologies are introduced. They are both oriented towards achieving the same control objective. Depending on the circumstances, one or the other may be used. The NN identification model performance is assessed by comparing the estimated output and the actual motor output for a common arbitrary excitation signal.

Equation (81) which describes the motor dynamics, can be partitioned as

$$\omega_p(k+1) = f[\omega_p(k), \omega_p(k-1)] + \xi V_t(k) \tag{82}$$

where the function f [.] is given by,

$$f[\omega_p(k), \omega_p(k-1)] = \alpha\,\omega_p(k) + \beta\,\omega_p(k-1)$$

$$+ \gamma\,[\text{sign}(\omega_p(k))]\,\omega_p^2(k) \qquad (83)$$

$$+ \delta\,[\text{sign}(\omega_p(k))]\,\omega_p^2(k-1)$$

and is assumed to be unknown. A NN is trained to emulate this unknown function f [.]. The values $\omega_p(k)$ and $\omega_p(k-1)$ which are the independent variables of f [.], are selected as the inputs to the NN. The corresponding NN output target $f[\omega_p(k), \omega_p(k-1)]$ is given by equation (83). The target is also equivalent to the value of $\omega_p(k+1)$ in equation (81) with the terminal voltage $V_t(k)$ set to zero. The latter method is useful when deriving training data from actual hardware.

Table 7 Training and Testing statistics of the NN

number of inputs	3
number of outputs	1
number of hidden layers	1
number of hidden neurons	5
number of training patterns P	600
number of training sweeps	1000
learning step η	0.1
momentum gain υ	0.1
E_{total} threshold ε	0.04

The NN is trained off-line using randomly generated input patterns of $[\omega_p(k), \omega_p(k-1)]$ and the corresponding target $f[\omega_p(k), \omega_p(k-1)]$. The choice of $\omega_p(k)$ and $\omega_p(k-1)$ have to satisfy the constraints previously specified.

The motor speed is estimated by the trained NN predictor as

$$\hat{\omega}_p(k+1) = N[\omega_p(k), \omega_p(k-1)] + \xi\,V_t(k) \qquad (84)$$

where N[.] denotes the NN output for a given set of "." inputs. A " ^ " indicates an estimated value of the quantity directly below it. The NN topology and the training effort are briefly described by the statistics in table 7.

As mentioned earlier, except for the number of inputs/outputs of the NN, all other design and learning parameters are selected by trial and error.

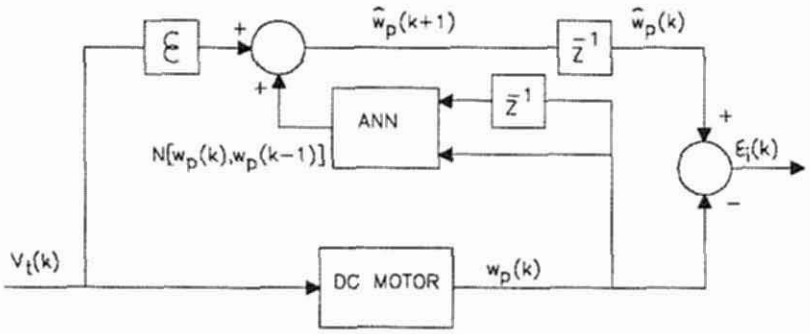


Figure 21. Structure of the NN for identification of the dc motor

From [65] courtesy IEEE (C) IEEE, 1991

The trained NN is applied as a series-parallel type identifier as described in reference [70], to estimate the value of the function f [.]. The structure of the identifier is shown in figure 21. z^{-1} in any figure indicates a unit time delay. The performance of the trained NN identifier is evaluated by comparing the actual and estimated speeds as calculated from equations (82) and (84) respectively for the following arbitrarily selected terminal voltage sequence

$$V_t(k) = 50 \sin(2\pi kT/7) + 45 \sin(2\pi kT/3) \quad \forall \; kT \in [0, 20]$$

The results are given in figure 22. It is seen that the two tracks are barely distinguishable from each other. The maximum identification error is 0.36 rad/s.

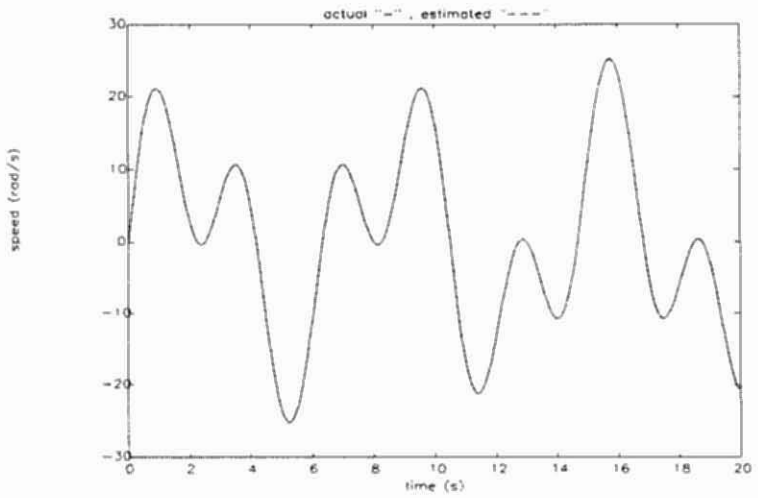


Figure 22. Actual and estimated rotor speeds

From [65] courtesy IEEE (C) IEEE, 1991

It is important to note that this algorithm assumes the availability of the value ξ for its operation. It can be proved that ξ is a function of J, K, R_a, L_a, D and the sampling period T and that it can be explicitly evaluated if these parameters are known. Since all the parameters are motor specific, it is fair to assume their availability. However, when none of the motor parameters are available, topology II is proposed for dc motor identification.

6.11.4 Trajectory Control of DC Motor using NN

The objective of the control system is to drive the motor so that its speed, $\omega_p(k)$, follows a prespecified trajectory, $\omega_m(k)$. This is done by letting the dc motor follow the output of a selected reference model throughout the trajectory [70]. The following second order reference model is selected.

$$\omega_m(k+1) = 0.6\,\omega_m(k) + 0.2\,\omega_m(k-1) + r(k) \tag{85}$$

r(k) is the bounded input to the reference model. The coefficients are selected to ensure that its poles are within the unit circle and has the type of response that can be achieved by the dc motor. For a given desired sequence $\{\omega_m(k)\}$ (trajectory), the corresponding control sequence $\{r(k)\}$ can be calculated using equation (85).

The controller uses the previously trained NN to estimate the motor terminal voltage $V_t(k)$ which enables accurate trajectory control of the shaft speed $\omega_p(k)$. Performance of the two controllers are simulated for arbitrarily selected speed tracks $\{\omega_m(k)\}$. A graphical comparison of the specified and actual speed trajectories are presented.

Let's for a moment assume that the tracking error $\varepsilon_c(k)$ is zero, and that the nonlinear function f[.] in equation (82) is known. The control input $V_t(k)$ to the motor at the k^{th} time step can be calculated as

$$\begin{aligned} V_t(k) &= [\, -f\,[\omega_p(k), \omega_p(k-1)] \\ &\quad + 0.6\,\omega_p(k) + 0.2\,\omega_p(k-1) + r(k)] \,/\, \xi \end{aligned} \tag{86}$$

Substituting this result in equation (82) and combining with equation (85) gives the tracking error difference equation

$$\varepsilon_c(k+1) = 0.6\,\varepsilon_c(k) + 0.2\,\varepsilon_c(k-1) \tag{87}$$

Since the reference model is asymptotically stable, it follows that $\lim \varepsilon_c(k+1) = 0$ for arbitrary initial conditions. However, since f [.] is not known, its value is estimated using the trained NN. The estimated terminal voltage is given by,

$$\begin{aligned} \hat{V}t(k) &= [\, -N\,[\omega_p(k), \omega_p(k-1)] \\ &\quad + 0.6\,\omega_p(k) + 0.2\,\omega_p(k-1) + r(k)] \,/\, \xi \end{aligned} \tag{88}$$

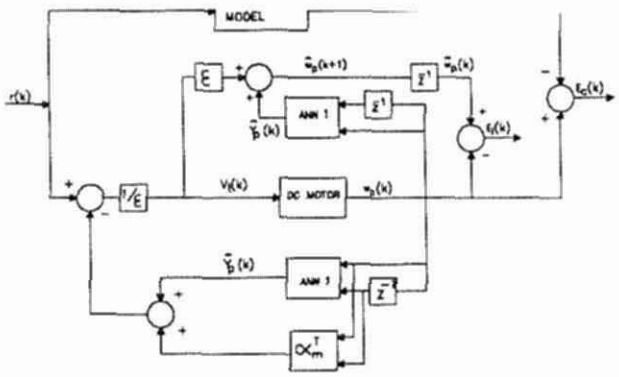


Figure 23. The overall structure of the controller

From [65] courtesy IEEE (C) IEEE, 1991

The overall structure of the identification and control system is displayed in Figure 23. The tracking control capability of the model was investigated for different arbitrarily specified trajectories. Only a specific result is shown for brevity. In this case, the specified speed trajectory is defined by

$$\omega_m(k) = 10\sin(2.0\pi kT/4) + 16\sin(2.0\pi kT/7) \quad \forall \; kT \in [0, 20]$$

For the above trajectory, the corresponding {r(k)} is derived by using equation (85). This is applied to the model shown in figure 23. The matrix α_m corresponds to the reference model coefficients [0.6 0.2].

Figure 24 compares the actual and specified speed trajectories for the above sinusoidal reference track. Close model following is observed. The maximum tracking error is 0.55 rad/s.

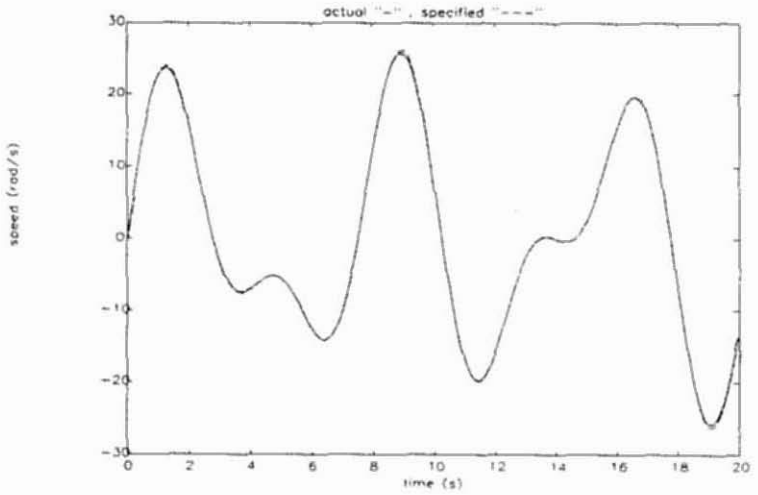


Figure 24. Tracking performance for a sinusoidal reference track

From [65] courtesy IEEE (C) IEEE, 1991

.11.5 Comments

. dc motor has been successfully controlled using an NN. The ıknown, time invariant, nonlinear operating characteristics of the dc ıotor and its load have been successfully captured by an NN. The ɔncepts of model reference adaptive control have been used in ɔnjunction with the trained NN to achieve trajectory control of the ıtor speed. Two different controller topologies have been presented. oth display good tracking performance. Simulations were also ɛrformed under noisy operating conditions to study the degree of ıbustness of the controller, which is an important consideration in ıy practical application.

entification of a time varying drive system using an NN is also of ınsiderable interest and needs to be studied. The overall system ıbility was never investigated from a conventional control theoretic ɔint of view and is worth studying. Implementation of the control hemes on actual hardware is currently under progress.

References

[1] D.H. Ackley, G.E. Hinton and T.J. Sejnowski, "A learning algorithm for Boltzmann machines," *Cognitive Science*, vol.9, pp.147-169, 1985.

[2] Y.S. Abu-Mostafa and J.M. St. Jaques, "Information capacity of the Hopfield model," *IEEE Trans. on Information Theory*, vol.IT-31, p.461 (1985).

[3] M. Aggoune, M. El-Sharkawi, D. Park, M. Damborg, and R. Marks II, "Preliminary results on using artificial neural networks for security assessment," Proc. of 1989 Power Ind. Comp. Appl. Conf., Seattle, WA, May, 1989 (to appear in IEEE Trans. Power Sys.)

[4] M.E. Aggoune, M.A. El-Sharkawi, D.C. Park, M.J. Damborg and R.J. Marks II, "Preliminary results on using artificial neural networks for security assessment," *IEEE Transactions on Power Engineering* (in press).

[5] L.E. Atlas, D. Cohn, R. Ladner, M.A. El-Sharkawi, R.J. Marks II, M.E. Aggoune, D.C. Park, "Training connectionist networks with queries and selective sampling," *Advances in Neural Network Information Processing Systems 2*, Morgan Kaufman Publishers, Inc., San Mateo, CA., 1990, pp.566-573.

[6] J.A. Anderson and E. Rosenfeld, Eds., **Neurocomputing: Foundations of Research**, MIT Press, Cambridge, MA, 1988.

[7] L.E. Atlas, R. Cole, Y. Muthusamy, A. Lippman, G. Connor, D.C. Park, M. El-Sharkawi & R.J. Marks II, "A performance comparison of trained multi-layer perceptrons and classification trees," *Proceedings of the IEEE*, vol.78, pp.1614-1619 (1990).

II, A. Lippman, R. Cole and Y. Muthusamy, "A performance comparison of trained multi-layer perceptrons and trained classification trees," *Proc. 1989 IEEE International Conference on Systems, Man and Cybernetics*, (Hyatt Regancy, Cambridge, Massachusetts, 14-17 Nov. 1989), pp.915-920.

[9] L.E. Atlas and Y. Suzuki, "Digital systems for artificial neural networks," *IEEE Circuits & Devices Magazine*, vol. 5, pp.20-24 (1989).

10] E. Baum and D. Haussler, "What size net gives valid generalization," in **Neural Information Processing Systems**, Morgan Kaufmann, 1989.

11] K. F. Cheung, R. J. Marks II and L. E. Atlas, "Synchronous vs. Asynchronous Behavior of Hopfield's CAM Neural Net," *Applied Optics*, vol.26, no.22, pp.4808-4813, 1987.

12] K.F. Cheung, S. Oh, R.J. Marks II and L.E. Atlas "Bernoulli clamping in alternating projection neural networks," Proceedings of the 1989 International Symposium on Computer Architecture and Digital Signal Processing (Hong Kong Convension and Exhibition Centre, 11-14 October, 1989).

13] J. Dayhaff, **Neural Network Architectures, An Introduction**, Van Nostrand Reinhold, 1990.

14] M.J. Damborg, M.A. El-Sharkawi & R.J. Marks II, "Potential applications of artificial neural networks to power system operation," *Proc. 1990 IEEE International Symposium on Circuits and Systems* 1-3 May, 1989, New Orleans, Louisiana- invited paper.

15] R.C. Eberhart and R.W. Dobbins, **Neural Network PC Tools: a Practical Guide**, Academic Press, 1990.

Damborg and L.E. Atlas, "Dynamic security assessment of power systems using back error propagation artificial neural networks," *Proceedings of the 2nd Annual Symposium on Expert Systems Applications to Power Systems*, pp.366-370, 17-20 July 89, Seattle.

[17] N.H. Farhat, "Optoelectronic neural networks and learning machines," *IEEE Circuits & Devices Magazine*, vol.5, pp.32-41 (September, 1989).

[18] S. Geman and D. Geman, "Stochastic Relaxation, Gibb's Distribution, and the Baysian Restoration of Images," *IEEE Tran. Pattern Recognition and Machine Intelligence*, vol PAMI-6, pp.721-741, 1984.

[19] H.P. Graf & L.D. Jackal, "Analog electronic neural network circuits," *IEEE Circuits & Devices Magazine*, vol.5, pp.44-49 (July, 1989).

[20] S.A. Grossberg, **Neural Networks and Natural Intelligence**, MIT Press, Cambridge, MA, 1988.

[21] D.O. Hebb, **The Organization of Behaviour**, John Wiley, New York, 1949.

[22] J. J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," *Proceedings of the National Academy of Science*, USA, vol.79, pp.2554-2558, 1982.

[23] J.J. Hopfield and D.W.Tank, "Neural computation of decisions of decisions in optimazation problems," *Biol. Cyber.*, vol.52, pp.141-152 (1985).

[24] J.N. Hwang, S. Oh, J.J. Choi & R.J. Marks II, "Classification boundaries and gradients of trained multilayer perceptrons," *Proc. 1990 IEEE International Symposium on Circuits and Systems*, 1-3 May, 1990, New Orleans, Louisiana.

[25] J.N. Hwang, J.J. Choi, S. Oh and R.J. Marks II, "Query based learning applied to partially trained multilayer perceptrons," *IEEE Transactions on Neural Networks*, vol. 2, pp. 131-136, 1991.

[26] C.C. Klimasauskas, **The 1989 Neuro-Computing Bibliography**, MIT Press, Cambridge, 1989.

[27] T. Kohonen, **Self-Organization and Associative Memory**, 2nd Edition, **Springer-Verlag**, Berlin, 1988.

[28] D. Lapedes and R. Farber, "Nonlinear signal processing using neural networks; prediction and system modeling," Los Alamos National Laboratory, Los Alamos, New Mexico, TR LA-UR-87-2662, 1987

[29] A. Linden and J. Kindermann, "Inversion of multilayer nets," *Proc. Int'l Joint Conf. on Neural Networks*, II 425-430, Washington D.C., June 1989.

[30] R.P. Lippmann, "An introduction to computing with neural networks," *IEEE ASSP Magazine*, pp.4-22 (April 1987).

[31] R.J. Marks II, L.E. Atlas, D.C. Park and S. Oh, "The effect of stochastic interconnects in artificial neural network classification," *Proceedings of the IEEE International Conference on Neural Networks*, San Diego, July 24-27, 1988, vol.II, pp.437-442.

[32] R. J. Marks II, S. Oh and L. E. Atlas, "Alternating Projection Neural Networks," *IEEE Circuits and Systems*, vol.36, no.6, pp.846-857. 1989.

[33] J.L. McClelland and D.E. Rumelhart, **Parallel Distributed Processing, vols 1, 2 & 3**, MIT Press, Cambridge, MA, 1986, 1988.

[34] R. J. McEliece, E. C. Posner, E. R. Rodemichand S. S. Venkatesh, "The Capacity of the Hopfield Associative Memory," *IEEE Trans. Inf. Theory*, vol IT-33, no.4, pp.461-482, 1987.

[35] W.C. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol.5, pp.115-133 (1943).

[36] J.G. McDonnell, R.J. Marks II and L.E. Atlas, "Neural networks for solving combinatorial search problems: a tutorial," *Northcon/88 Conference Record, vol.II*, pp.868-876, (Western Periodicals Co., North Hollywood, CA), Seattle WA, October 1988.

[37] C. Mead, **Analog VLSI and Neural Systems**, Addison Wesley, Reading, MA (1989).

[38] M.L. Minsky and S.A. Papert, **Perceptrons**, MIT Press (1969); Expanded Edition, (1988).

[39] S. Oh, D. C. Park, R. J. Marks II and L. E. Atlas, "Nondispersive Propagation Skew in Iterative Neural Networks and Optical Feedback Processors," *Optical Engineering*, vol.28, pp.526-532, 1989.

[40] S. Oh and R.J. Marks II, "Dispersive propagation skew effects in iterative neural networks," *IEEE Transactions on Neural Networks*, vol. 2, pp. 160-162, 1991.

[41] Y.H. Pao, **Adaptive Pattern Recognition and Neural Networks**, Addison Wesley, 1989.

[42] D.C. Park, M.A. El-Sharkawi, R.J. Marks II, L.E. Atlas & M.J. Damborg, "Electric load forecasting using an artificial neural network," *IEEE Power Engineering Systems 1990 Summer Meeting*, Minneapolis, Minnesota, 15-19 July 1990.

tively Trained Neural Network," *IEEE Transactions on Neural Networks* (in press).

[44] C. Peterson, "Parallel distributed approaches to combinatorial optimazation: benchmark studies on traveling salesman problem," *Neural Computation*, vol.2, pp.261-269 (1990).

[45] F. Rosenblatt, "The perceptron: a probalistic model for information storage and organization in the brain," *Psychological Review*, col.65, pp.386-408 (1958).

[46] D.W. Tank and J.J. Hopfield, "Simple neural optimization networks," *IEEE Transactions on Circuits and Systems*, vol.CAS-33, p.533 (1986).

[47] M.Takeda and J.W. Goodman, "Neural networks for computation," *Applied Optics*, vol.25, pp.3033-3046, 1986.

[48] L. Valient, "A theory of the learnable," *Communications of the AMC*, vol.27, pp.1134-1142 (1984).

[49] P.D. Wasserman, **Neural Computation: Theory and Practice**, Van Nostrand Reinhold, 1989.

[50] P.D. Wasserman, **Neuralsource: the Bibliographic Guide to Artificial Neural Networks**, Van Nostrand Reinhold, 1989.

[51] S. Weerasooriya, M.A. El-Sharkawi, M. Damborg and R.J. Marks II, "Towards static security assessment of a large scale power system using neural networks," *IEEE Transactions on Power Engineering* (under review)

[52] B. Widrow and M.E. Hoff, "Adaptive switching circuits," *1960 IRE WESCON Convention Record*, New York, IRE, pp.96-104 (1960).

rity Assessment for Electric Power Systems," *IEEE Transactions on PWRS*, vol. 4, February, pp 220-228, 1989.

[54] T.M. Peng, N.F. Hubele and G.G. Karady, "Conceptual approach to the application of neural network for short-term load forecasting," *Proc of the 1990 ISCAS* vol. 3, pp 2942-2945, New Orleans, LA, May, 1990.

[55] M. Chow and R.J. Thomas, "Neural networks synchronous machine modeling," *Proc of the 1989 ISCAS* vol. 1, pp 495-498, Portland, OR, May, 1989.

[56] R. Fischl, M. Kam, J.C. Chow, and S. Ricciardi, "Screening power system contingencies using a back-propagation trained multiperceptron," *Proc of the 1989 ISCAS* vol. 1, pp 486-489, Portland, OR, May, 1989.

[57] H. Mori, H. Uematsu, S. Tsuzuki, T. Sakurai, Y. Kojima and K.Suzuki, "Identification of Harmonic Loads in Power Systems using an Artificial Neural Network," 2nd Symposium on Expert System Applications to Power Systems, Seattle, July, 1989.

[58] H. Mori and S. Tsuzuki, "Comparison between backpropagation and revised GMDH techniques for predicting voltage harmonics," *Proc of the 1990 ISCAS* vol. 2, pp 1102-1105, New Orleans, LA, May, 1990.

[59] P. Chan, "Application of neural-network computing in intelligent alarm processing," *PICA conference proceedings*, Seattle, WA, May, 1989.

[60] H. Tanaka, S. Matsuda, H. Ogi, Y. Izui, H. Taoka, and T. Sakaguchi, "Design and evaluation of neural network for fault diagnosis," 2nd Symposium on Expert System Applications to Power Systems, Seattle, WA, July, 1989.

[61] M.E. Aggoune, L.E. Atlas, D.A. Cohn, M.J. Damborg, M.A. El-Sharkawi and R.J. Marks II, "Artificial Neural Networks for Power Systems Static Security Assessment," International Symposium on Circuits and Systems, Portland, OR, 1989.

[62] N.I. Santoso O.T. Tan, "Neural Net based Real-Time Control of Capacitors Installed on Distribution Systems," 89 SM 768-3 PWRD, IEEE Power Engineering Society, Summer Meeting, Long Beach, California, 1989.

[63] H. Mori and S. Tsuzuki, "Power System Topological Observability Analysis using a Neural Network Model," 2nd Symposium on Expert System Applications to Power Systems, Seattle, July, 1989.

[64] H. Mori, and S. Tsuzuki, "Determination of power system topological observability using the boltzman machine," *Proc of the 1990 ISCAS* vol. 3, pp 2938-2941, New Orleans, LA, May, 1990.

[65] Siri Weerasooriya and M. A. El-Sharkawi, "Identification and Control of a DC Motor using a Back-propagation Neural Networks," Paper No 91 WM 288-1 EC, IEEE-PES Winter Meeting, New York, February, 1991.

[66] S. Weerasooriya and M. A. El-Sharkawi, "Adaptive tracking control for high performance dc drives," *IEEE Transactions on Energy Conversion*, vol. 5, pp 122-128, September, 1989.

[67] M. A. El-Sharkawi and S. Weerasooriya, "Development implementation of self-tuning tracking controller for dc motors," *IEEE Transactions on Energy Conversion*, vol. 5, pp 122-128, March 1990.

[68] B. C. Kuo, **Digital control systems**, HRW Inc., 1980.

[69] K. J. Astrom and B. Wittenmark, **Adaptive Control**, Addison Wesley, 1989.

[70] K. S. Narendra and K. Parthasarathy "Identification control of dynamical systems using neural networks," *IEEE Transactions on Neural Networks*, vol. 1, pp 4-27, March 1990.

[71] P. J. Antsaklis "Neural networks in control systems," *IEEE Control systems magazine*, vol. 10, no 3, pp 3-5, April, 1990.

[72] D. H. Nguyen and B. Widrow "Neural networks for self-learning control systems," *IEEE Control systems magazine*, vol. 10, no 3, pp 18-23, April, 1990.

[73] S. R. Chu, R. Shoureshi and M Tenorio "Neural networks for system identification," *IEEE Control systems magazine*, vol. 10, no 3, pp 31-35, April, 1990.

[74] Eu-Chuang Chen "Back-propagation neural networks for nonlinear self-tuning adaptive control," *IEEE Control systems magazine*, vol. 10, no 3, pp 44-48, April, 1990.

[75] D.J. Sobajic Y.H. Pao, and J. Dolce, "On-line monitoring and diagnosis of power system operating conditions using artificial neural networks," *Proc of the 1989 ISCAS* vol. 3, pp 2243-2246, Portland, OR, May, 1989.

[76] R.J. Thomas, E. Sakk, K.Hashemi, B.Y. Ku, and H.D. Chiang, "On-line security screening using an artificial neural network," *Proc of the 1990 ISCAS* vol. 3, pp 2921-2924, New Orleans, LA, May, 1990.

[77] D.J. Sobajic Y.H. Pao, W. Njo, and J. Dolce, "Real-time security monitoring of electric power systems," *Proc of the 1990 ISCAS* vol. 3, pp 2929-2932, New Orleans, LA, May, 1990.

[78] C. Maa, C. Chiu, and M.A. Shanblatt, "A constrained optimization neural net technique for economic power dispatch," *Proc of the 1990 ISCAS* vol. 3, pp 2946-2950, New Orleans, LA, May, 1990.

[9] R. Fischl, M. Kam, J.C. Chow, and S. Ricciardi, "An improved hopfield model for power system contingency classification," *Proc of the 1990 ISCAS* vol. 3, pp 2925-2928, New Orleans, LA, May, 1990.