1998

Power system security boundary visualization using intelligent techniques

Guozhong Zhou
Iowa State University

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Power system security boundary visualization using intelligent techniques

by

Guozhong Zhou

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Electrical Engineering (Electric Power)

Major Professor: James D. McCalley

Iowa State University

Ames, Iowa

1998

Graduate College
Iowa State University

This is to certify that the Doctoral dissertation of

Guozhong Zhou

has met the dissertation requirements of Iowa State University

Signature was redacted for privacy.

Major Professor

Signature was redacted for privacy.

For the Major Program

Signature was redacted for privacy.

For the Graduate College

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

1.1 Motivation for security boundary visualization

In the open access environment, utilities face a number of challenges to be more competitive. One of these challenges is that typical operating conditions tend to be much closer to security boundaries. This is because transmission utilization is increasing in sudden and unpredictable directions, and competition together with other regulatory requirements make new transmission facility construction more difficult. Consequently, security levels for the transmission network must be accurate and easily identified on-line by system operators.

Power system security assessment can be divided into two levels: classification and boundary determination. Classification involves determining whether the system is secure or insecure under all prespecified contingencies. The system is secure if the post-contingency performance is acceptable. Classification does not in itself indicate distance from the operating condition to the insecure conditions. Boundary determination, on the other hand, involves quantifying this distance. A boundary is represented by constraints imposed on parameters characterizing precontingency conditions. These *pre-contingency* parameters are called critical parameters. Once the boundary is identified, security assessment for any operating point can be given as the distance (in terms of critical parameters) between the current operating point and the boundary. Assessment in terms of precontingency operating parameters instead of the postcontingency performance measure is more meaningful to the operator.

1.2 Traditional security boundary visualization approach

In many North American utilities, the traditional boundary characterization approach is used to generate a two-dimensional graph called a nomogram [1] - [3], of which two axes correspond to two critical parameters. To develop a nomogram, two critical parameters are chosen and all other critical parameters are set to selected values within a typical operating range. The noncritical parameters, which may influence the postcontingency performance measure, are set to constant values biased to be conservative with respect to the influence on the performance measure. Points on the nomogram curve are determined by repeating computer simulations, varying one critical parameter while keeping the other constant. If the relationship between the performance measure and the critical parameters is fairly linear, interpolation and/or extrapolation helps to obtain boundary points. In this case, only three or four simulations may be needed. But highly nonlinear relationships may require more simulations. The above procedure is repeated for selected values of the first critical parameter provides enough boundary points to draw the nomogram curve. In the case where there is a third critical parameter value change, a different nomogram curve is drawn for each value of this third critical parameter so that the result is a family of nomogram curves, as illustrated in Figure 1.1. Inclusion of a fourth critical parameter value change requires a distinct family of nomogram curves (i.e., a new page) for each new value of the fourth critical parameter. Inclusion of fifth critical parameter value change requires a distinct family of pages for each new value of the fifth critical parameter, and so on. Obviously, the above manual approach requires intensive labor involvement. The other disadvantages include inaccurate boundary representation and little flexibility to integrate with the energy management system (EMS).

There are two main approximations made in nomogram development which can ultimately result in inaccurate boundary characterization when the nomogram is used by

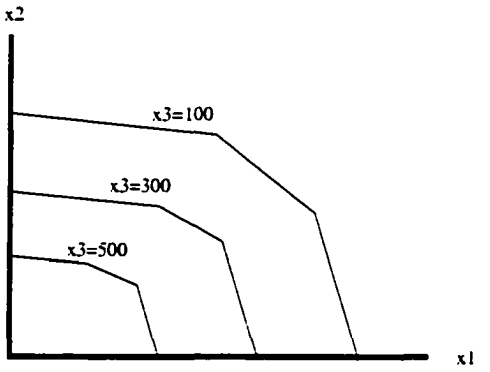


Figure 1.1 Nomogram curves for three critical parameters

the operator. One is linear interpolation between points. The other is insufficient information contained in critical parameters. Because of labor requirements, the simulation procedure described above is normally used to obtain only a very few points on the boundary; indeed, often only the "corner points" are obtained. The remaining portions of the boundary are obtained by drawing a straight line through the computed points. This approximation can be improved by automating the security assessment process so that the number of points generated is only limited by computer availability, and by using neural networks, an interpolation tool that models very well nonlinear portions of the boundary. As for the second approximation, nomogram development usually limits the number of critical parameters to five or less, even when there are other parameters known to be influential. There are two reasons for this. First, having more critical parameters requires performing more simulations. Second, it is difficult to compactly represent the information to the operator if there are more than five critical parameters. For example, if the fourth and fifth parameters have six different levels of interest, the operator would require 36 pages of three paramater nomogram curves. Limiting the number of critical paramaters may mean that the information content of the parameters that are used is insufficient for accurately predicting the performance measure. The approximation due to insufficient information contained in critical parameters can be improved by automatically performing a large number of simulations with little atten-

tion from the analyst, and by using neural networks that provide compact boundary characterization for any number of critical parameters. The difficulty in integrating the procedure into the EMS lies in the manual nature of the traditional approach. This can be solved by using an automatic approach.

Therefore, to overcome the disadvantages of the traditional approach, an automatic security boundary visualization methodology has been developed using intelligent techniques [4] - [8].

1.3 Automatic security boundary visualization methodology

As Figure 1.2 shows, the procedure developed in this dissertation to determine and illustrate security boundaries includes six steps.

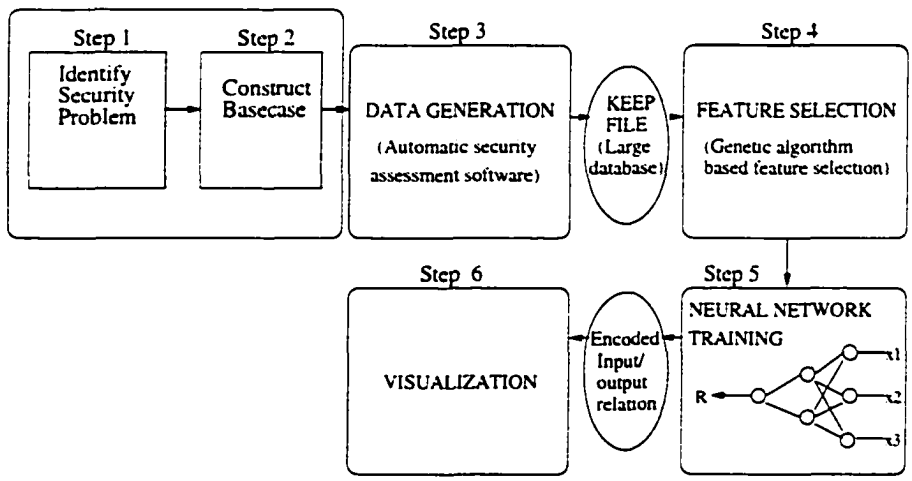


Figure 1.2 Procedure for security boundary visualization

1. *Security problem identification:* This step is no different from what is done when generating nomograms in the traditional, manual way. Therefore, it is a task of the engineer to identify the specific set of security constraints to be characterized and candidate operating parameters that may have influence on these constraints.

2. *Base case construction:* This is also much like what is done in the traditional way. A base case power flow solution needs to be constructed that appropriately models the system conditions of concern.

3. *Data generation:* This step involves an automated procedure that generates a large database, with each record consisting of precontingency operating parameters and the corresponding postcontingency performance measure.

4. *Feature selection:* This step selects the best subset of precontingency operating parameters for use in predicting the postcontingency performance measure.

5. *Neural network training:* Based on the selected parameters and the data base obtained from previous steps, this step perform neural network training for function approximation of the mapping from the precontingency operating parameters to the postcontingency performance measure.

6. *Visualization and presentation:* This step integrates the encoded neural network mapping into a visualization software that can be used on-line to generate the security boundary.

Although it is assumed in this dissertation that steps 1 to 5 are performed off-line, it is expected that our development of these steps will ultimately be applicable on-line.

1.4 Sample system

A sample system is shown in Figure 1.3. This is just a simplified illustration of a real system, a 2500-bus model of Northern California. The detailed system is used for all simulations throughout this research. This 2500-bus nodel will be used for testing all studies conducted in this dissertation. The load in the subarea, during high loading

conditions, is greater than the generation capacity in the subarea. Therefore, a significant amount of power must be imported into the subarea to meet the demand. There are several ties between the subarea and the remaining part of the system. Based on experience and knowledge of the system, we know that operation of generators A, B, and C under high subarea loadings is constrained by two different security problems as described below.

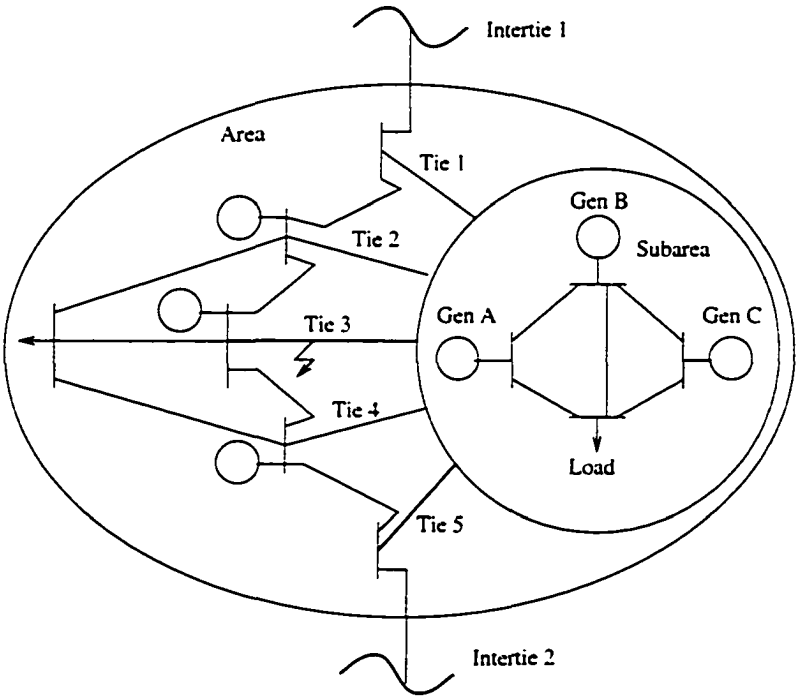


Figure 1.3 A sample system

1.4.1 Thermal overload

Thermal overload on a piece of equipment means that the equipment carries a current above its normal current carrying ability. Thermal overload can cause overheating on the equipment which results in equipment performence degradation. For example, thermal overload on an overhead line can cause sag increase which results in short-circuit faults and conductor strength decrease which reduces the life time of the line. Thermal overload occurs on tie line 5 when tie line 3 is outaged under certain conditions. The

postcontingency performance measure is the flow in ampere on tie line 5. The emergency rating of tie line 5 is $I_0 = 600$A. Therefore, the threshold value for this constraint is $I_0 = 600$A, and when the performance measure is normalized according to

$$R = \frac{I - I_0}{I_0}$$

we have $R_0 = 0$ on the boundary. The more negative the R value, the more secure the system.

1.4.2 Voltage instability

Voltage instability is the inability of a power system to maintain steady acceptable voltages at all buses in the system under normal operating conditions and small disturbances. The main factor causing instability is the inability of the power system to meet the demand for reactive power. A system is voltage unstable if at least one bus voltage magnitude decreases as the capacitive reactive power injection at the same bus is increased [9]. Various analytical methods on voltage stability can be found in [10] - [17]. For the sample system, the outage of tie line 3 can cause voltage instability problems in the subarea. One load bus in the subarea has been identified as the most reactive deficient bus. A secure operating point on the boundary is defined as one for which this bus has at least Q_0=200 MVAr of reactive margin following loss of tie line 3. The performance measure is MVAr margin Q at this load bus. When the performance measure for this problem is normalized according to

$$R = \frac{Q_0 - Q}{Q_0}$$

we have $R_0 = 0$ on the boundary.

1.5 Contributions of this dissertation

The objective of this dissertation is to develop the security assessment methodology of Figure 1.2 that results in highly accurate portrayal of operating boundaries for system operators in terms of easily monitorable operating parameters such as generation, load, flow, and voltage levels. In the context of this objective, this dissertation will make the following main contributions.

1.5.1 Development of a systematic data generation method

Data generation is always an important issue for neural network applications. A systematic data generation method has been developed to generate high quality data through simulation for any kind of security assessment problems[1]. The developed method uses boundary search and revised Latin hypercube sampling to ensure that the data set is centered around the boundary to be characterized and well resolved throughout the parameter operating ranges.

1.5.2 Development of an efficient feature selection tool for operating parameter selection

Another important issue in neural network applications is the chioce of parameters to be used as the neural network inputs. This is because good or bad features lead to good or bad neural network prediction accuracy. A genetic algorithm based feature selection tool has been developed to select the best operating parameters as the neural network inputs. A multilayer perceptron was initially used for fitness evaluation. However, it has been found that use of radial basis function networks instead of multilayer perceptrons, the solution speed could be greatly improved. With k-nearest neighbor method, the solution speed could be further improved.

[1]The initial development work was done by Dr. Shimo Wang.

1.5.3 Application of data analysis techniques for neural network training

Even if care is taken to ensure the quality of data during data generation, it is still possible that the data set contains outliers or is not well distributed. Statistical data analysis techniques have been used to analyze the data for outliers and for distribution. These techniques include principal component analysis, k-nearest neighbor estimation, multivariate normality test, scatter plots, and projection pursuit visualization.

1.5.4 Confidence interval calculation for neural network performance evaluation

A confidence interval calculation has been derived for multilayer perceptrons. It has been used to measure the reliability of neural network outputs.

1.5.5 Sensitivity analysis of neural network outputs with respect to inputs

The relative influential ability of the neural network input parameters on the output can provide preventive control guidance in manoeuvering the current operating point with respect to the boundary. Therefore, sensitivity analysis formulas of neural network outputs with respect to inputs have been derived for multilayer perceptrons and the corresponding calculations have been performed for the neural network that was used to characterize the boundary.

1.5.6 Development of a composite boundary visualization algorithm

Based on the trained neural network, a composite boundary visualization algorithm has been developed to draw a two-dimensional diagram for multiple security problems under the same contingency. The boundary can be on-line displayed to the user. The user can select to view any boundary for any pair of controllable operating parameters.

1.5.7 Available transfer capability calculation based on the boundary

Under the open access power marketing environment, the available transfer capability between control areas or along a transaction path is required to be posted regularly. A new method for the available transfer capability calculation has been proposed based on the boundary. This method is very straightforward once the boundary is obtained.

1.6 Organization of this dissertation

After introduction in Chapter 1, the remaining chapters of this dissertation are organized as follows: Chapter 2 reviews the literature on intelligent technique applications to security assessment. Chapter 3 presents a systematic method for data generation and some techniques for data analysis. Chapter 4 describes a genetic algorithm based feature selection tool for choosing operating parameters as neural network inputs. Chapter 5 discusses the neural network training and analysis. Chapter 6 describes boundary presentation techniques. Chapter 7 draws conclusions and suggests further work.

2 LITERATURE REVIEW

2.1 Introduction

Intelligent techniques mainly include neural networks, expert systems, fuzzy logic, evolutionary algorithms (genetic algorithms, evolution strategies, and evolutionary programming), and machine learning. These techniques have been applied with some success to various areas in power systems in the past few years. Security assessment is one area with many such applications. Security assessment can be divided into static security assessment and dynamic security assessment depending on whether transient behavior is considered. For example, thermal overload and voltage violations following the loss of a transmission line are static security problems, and the loss of synchronism among generators and voltage collapse following a disturbance are dynamic security problems. Since analytical techniques for security assessment are quite time-consuming, they are difficult to apply for on-line use although there has been progress in this area recently [18, 19]. Because neural networks can map nonlinear relationships among data and the solution speed for trained neural networks is fast, they are suitable for on-line use. A large number of publications on neural network applications to security assessment have appeared since Sobajic et al. [20] and Aggoune et al. [21] explored neural network capability of assessing transient stability and steady-state security. Some representative examples of them are discussed below.

2.2 Neural network applications to security assessment

Sobajic et al. [20] use multilayer perceptrons as classifiers with precontingency parameters (rotor angles of generators, accelerating power, and accelerating energy) as inputs and postcontingency security measure (critical clearing time) as output. Aggoune et al. [21] also use multilayer perceptrons as classifiers, but inputs are real and reactive injections, excitation gains of generators and other parameters, and the output is the security status which is determined by system eigenvalues. Zhou et al. [22] apply a neural network to the concept of system vulnerability based on the transient energy function method. Miranda et al. [23] propose an approach to transient stability enhancement based on neural network sensitivity analysis. El-Keib et al. [24] apply multilayer perceptrons for voltage stability assessment with the energy margin as the stability index and determine input parameters for neural network training by sensitivity analysis. Momoh et al. [25] also use multilayer perceptrons to determine reactive compensation to enhance voltage stability. La Scala et al. [26] apply multilayer perceptrons to voltage security monitoring based on a dynamic system model.

In the previous references, supervised learning algorithms are used for neural network training. Niebur et al. [27] use a Kohonen network to map similar input patterns into neighboring nodes in topology. Unsupervised learning algorithms are used for neural network training. After training, if one knows which cluster is associated with which state (secure or insecure), one can identify the current system as secure or insecure after mapping the operating state. A similar approach is studied by El-Sharkawi et al. [28] where different maps are constructed for different contingencies. Sobajic et al. [29] use a combination of supervised and unsupervised learning for dynamic security assessment. More literature on neural network applications to power systems can be found in [30, 31].

2.3 Problems related to neural network applications

There are many problems to be considered when using neural networks for security assessment. These problems include selection of neural network architecture and learning algorithm, checking data quality, selection of the most predictive features to reduce the input dimensionality, monitoring the training process to avoid underfitting or overfitting, and performance evaluation of the output. Some of them have been studied in the power system context. Zayan et al. [63] compare the minimum entropy method and the *Karhunen – Loève* expansion for feature extraction with application to security classification. Muknahallipatna et al. [64] use the linear correlation method and discriminant analysis for feature extraction. Some of these problems will be addressed in the subsequent chapters.

2.4 Other intelligent technique applications to security assessment

In addition to the previously described neural network applications, other intelligent methods have also found use in security assessment, either as main tools or as supporting techniques. Wehenkel et al. [32, 33, 34, 35, 36] has made significant contributions to security assessment by applying machine learning approach via decision trees. This work has resulted in an approach to fast and accurate classification of an operating point (see [36] for in-depth discussions). Similar investigations by others are also reported [37, 38, 39, 40]. Pecas Lopes et al. [41] report a fast dynamic security assessment method using genetic programming. Hatziargyriou et al. [42] conduct a comparative study on k-nearest neighbor, decision tree, and neural network approaches for a small isolated power system. Their results show that the main advantage of the neural network approach over the other two is its more accurate classification. When using neural networks, it

is required, at least for large systems, that the computer simulations be automated. Therefore, many of the previous references also mention some form of an expert system for this purpose. Marceau et al. [43] reports an advanced expert system of this nature. Huneault et al. [44] provide a literature and industry survey of expert system applications in power engineering.

As each intelligent technique has its own disadvantages, some studies on hybrid intelligent technique applications to security assessment have been reported. Wehenkel et al. [45, 46] propose two hybrid approaches. One is the combination of decision tree and neural network where a tree is first built and then translated into a layered neural network. This can improve the reliability of decision trees while maintaining their advantages of simplicity and computational efficiency. The other is the combination of decision trees and k-nearest neighbor where, in the nearest neighbor distance computation, the algorithm uses only attributes selected by a decision tree. This method is faster and more reliable compared with k-nearest neighbor method. Yan et al. [47] use a hybrid expert system/neural network for contingency classification. El-Sharkawi et al. [48, 49] use the genetic algorithm to set up the initial weights or train neural networks for dynamic security assessment.

2.5 Another approach - security assessment via boundary visualization

Most references mentioned above concern classification of the operating point as secure or insecure following a contingency with respect to a particular security problem such as thermal overload, dynamic security, and voltage instability. A few of them use postcontingency margins as security criteria. This research focuses on a methodology that can be used on-line for determining and illustrating the security boundary for any type of security problem. Another approach adopted in this dissertation is function

approximation instead of classification. In this approach, one will know not only whether the current system operating condition is secure but also how far it is from the boundary in terms of easily controllable parameters; there is therefore built-in to our approach corrective action guidance. A systematic method has been developed to generate data for neural network use. The resulting data generation software is a general, flexible package that can also be used for other problems involving neural network applications. Another important problem in neural network applications is what input features should be used. To solve this problem, a genetic algorithm based feature selection tool has been developed for neural networks. Finally, it is emphasized that the approach to obtain the boundary can be applied to all kinds of security problems and their combinations, rather than just one.

3 DATA GENERATION AND DATA ANALYSIS

3.1 Introduction

Data generation is a very important part of the procedure. Quality of the data is a key factor that affects the performance of neural network. hence the accuracy of the boundary. If the data does not reflect the actual behaviors of power system operations. one can not expect the resulting boundary to be correct. The developed software to generate the data is called an automatic security assessment software (ASAS). It is descibed in in Section 3.2. Data analysis is to analyze the data to ensure that both data distribution and data coverage are good for neural network training. Several data analysis techniques are discussed for use in analyzing the data following data generation in Section 3.3

3.2 Data generation

ASAS, illustrated in Figure 3.1, is used to generate a large database containing data characterizing precontingency operating conditions and corresponding system performances for one specific contingency. The simulation tool, as a module, interfaces with the other ASAS modules so that one can replace it with software appropriate for analysis of the problem under study. In the thermal overload example, the simulation tool is a power flow program called IPFLOW developed by EPRI. IPFLOW is also used to set up the initial operating conditions. Power flow study is a basic tool for system design.

planning, and operation. The power flow equations can be expressed as

$$P_i = \sum_{k=1}^{N} | Y_{ik} V_i V_k | \cos(\theta_{ik} + \theta_k - \theta_i) \quad (3.1)$$

$$Q_i = -\sum_{k=1}^{N} | Y_{ik} V_i V_k | \sin(\theta_{ik} + \theta_k - \theta_i) \quad (3.2)$$

where P_i and Q_i are the real and reactive power injections at bus i, Y_{ik} is the admittance between bus i and bus j, V_i is the complex voltage at bus i, θ_{ik} is the phase angle difference of voltage from bus i to bus k, and θ_i is the phase angle of voltage at bus i. There are typically $N - 1 P_i$ equations and $N - N_g - 1 Q_i$ equations to solve where N is the number of bueses and N_g is the number of buses with generators. Solution to the power flow program can provide a snapshot of the system operating conditions. The basic idea for data generation is to change the operating conditions and run the power flow program to simulate the specific contingency.

In the voltage stability example, the simulation tool is a voltage assessment program called VCMARGIN provided by a power company.

Section 3.2.1 provides an overview of the fundamental concepts on which development of ASAS is based. Section 3.2.2 describes the key criteria to be used in setting up the data for ASAS. Sections 3.2.3 and 3.2.4 provide details on the two techniques called boundary search and structured Monte Carlo sampling.

3.2.1 Overview

ASAS generates the data required for neural network training. This data consists of a large number of samples, with each sample corresponding to a simulation of the same contingency but for different operating conditions, and consisting of precontingency operating parameters (the critical parameter candidates) together with the postcontingency performance measure. A KEEP file is generated as illustrated by Table 3.1.

Here, x_i is the i-th parameter of N_c critical parameter candidates denoted by vector

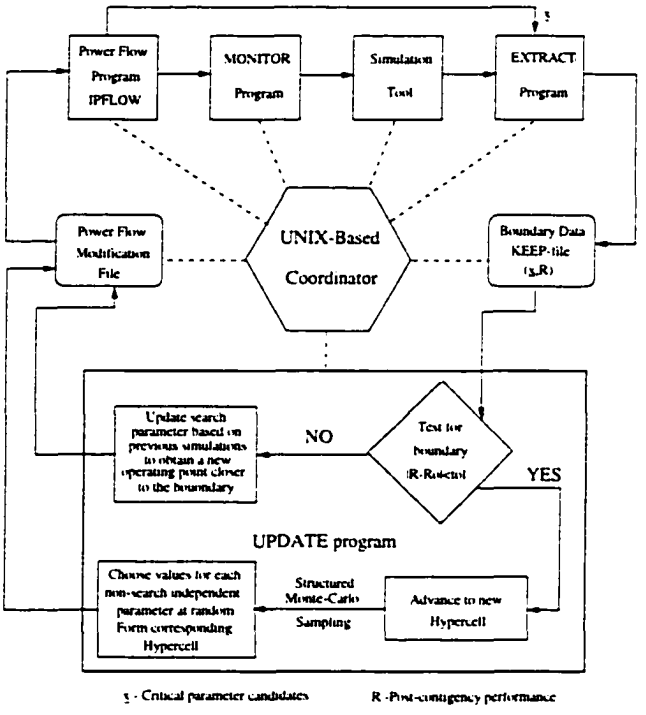


Figure 3.1 Automatic security assessment software package.

Table 3.1 The structure of a KEEP file.

Sample No.	x_1	x_2	$\cdots$	x_{N_c}	R
1					
2					
$\vdots$					
N_r					

$\mathbf{x}$, and R is the performance measure. There are N_r samples (rows) in the KEEP file. This is the output of ASAS that is used to train the neural network.

The objective of the data generation software is: *Given an operating range and an operating range resolution, create a data set of samples that best reflects the dependency of the postcontingency performance measure on variation in precontingency operating conditions.*

More formally, let us represent the vector of bus voltage angles and magnitudes by $\mathbf{w}$, the vector of real and reactive bus injections by $\mathbf{u}$, and the vector of circuit flows by $\mathbf{y}$. Then the power flow equations can be written as $\mathbf{h}(\mathbf{w}, \mathbf{u}) = \mathbf{0}$, and circuit flows can be computed by $\mathbf{y} = \mathbf{g}(\mathbf{w})$. Let us further represent the scalar postcontingency performance by R which quantifies the acceptability of the system response following a specific contingency. Then the value of R is uniquely determined by the precontingency operating condition, i.e., $R = F(\mathbf{w}, \mathbf{u})$. We assume, however, that R may be computed more compactly and with a good level of accuracy, using a small subset of parameters $\mathbf{x}^s$ taken from $\mathbf{w}$, $\mathbf{u}$, and $\mathbf{y}$, i.e., $\mathbf{x}^s \subset (\mathbf{w} \cup \mathbf{u} \cup \mathbf{y})$ resulting in R according to $R = f(\mathbf{x}^s)$. Given that we can obtain the function f, and given that we specify a threshold performance measure R_0 beyond which system performance is unacceptable, the boundary associated with the contingency can be expressed as the set of feasible operating conditions which solve $f(\mathbf{x}^s) = R_0$. We shift and normalize the function to force $R_0 = 0$ so that $R < 0$ denotes acceptable performance and $R > 0$ denotes unacceptable performance.

There are two special techniques employed by ASAS in accomplishing the stated objective. These are boundary search and structured Monte Carlo sampling. The first technique provides that the samples generated by ASAS center around the boundary to be characterized. The second technique provides that the sample points are well distributed and highly resolved throughout the operating range of interest. These two techniques are illustrated in Figure 3.1. The forward path at the top of Figure 3.1 per-

forms the setup of the operating conditions via a power flow solution and then simulates the contingency. The contingency simulation tool interfaces with other ASAS software in a modular way so that one can replace it with software appropriate for analysis of the problem under study. The decision of which operating points to model is made in the two feedback loops at the bottom of Figure 3.1. The outer loop uses Structured Monte Carlo sampling to lead ASAS on a predefined "trajectory" of *operating states* in terms of all but one operating parameter. The one remaining operating parameter, called the *search parameter*, is varied in the inner loop to generate several samples on both sides of the boundary for each operating state.

3.2.2 Data preparation

As neural networks can do well in interpolation but not in extrapolation, it is important to guide the data generation to capture the breadth of the operating range. Attempts to use the neural network outside this operating range will result in decreased accuracy or even misleading output.

3.2.2.1 Choice of Critical Parameter Candidates

Data preparation for ASAS requires that the engineer first classify certain precontingency parameters, as described below.

Critical parameter candidates (CPCs)

These are the precontingency operating parameters that are expected to be good predictors of the performance measure. This is equivalent to saying that they are expected to be influential with respect to the postcontingency system performance. They are chosen by the engineer using judgment, experience with the system, and physical understanding of the security problem. They can be any parameter that can be monitored

by the operator in the control center, including generation levels, load levels, voltage magnitudes, line flows, and unit commitment (i.e., unit status). The engineer should overselect CPCs; if there is doubt about whether a parameter should be a CPC, then it should be included as a CPC. The vector of all CPCs is denoted as $\mathbf{x} = [x_1, x_2, \cdots, x_{N_c}]^T$ where T indicates the vector transpose operation.

Independent critical parameter candidates (ICPCs)

These are critical parameter candidates that can be directly controlled by the analyst when running a power flow program, i.e., they are input data to a power flow calculation. Examples include MW generation levels, real or reactive load levels or load power factors, unit commitment (i.e., unit status), and PV bus voltage levels. Examples of parameters that could not be ICPCs are circuit flows, PQ bus voltage magnitudes, bus voltage angles, and reactive generation. There are two special kinds of ICPCs:

1. *Controllable ICPCs:* It is important that at least one of the ICPCs be controllable by the operator in order to provide for corrective action guidance, i.e., the boundary displayed to the operator should be characterized by at least one parameter that the operator can directly control, not to manoeuver the boundary, but to manoeuver the operating point with respect to the boundary. Controllable ICPCs include MW generation levels and PV bus voltage levels. Load levels are generally not controllable.

2. *Search parameter* z_1*:* The search parameter is varied in an intelligent fashion to generate operating points on both sides (secure side and insecure side) of the boundary to be characterized. This will be explained more fully later. The search parameter must be a continuous valued ICPC.

The ICPCs are denoted as $\mathbf{z} = [z_1, z_2, \cdots, z_{M_z}]^T$. Note that the vector $\mathbf{z}$ is a subset

of the vector $\mathbf{x}$, i.e., $\mathbf{z} \subset \mathbf{x}$.

Dependent critical parameter candidates (DCPCs)

All CPCs that are not ICPCs are dependent critical parameter candidates (DCPCs). These include PQ bus voltage magnitudes and angles, PV bus reactive injections and line flows. They are computed as the solution to a power flow computation.

3.2.2.2 Operating range, simulation time, and hypercell identification

Having identified the CPCs and the ICPCs, there are four steps to preparing the data:

1. Identify the number of simulations allowable to generate the data. This can be estimated by running one complete simulation using ASAS and measuring the simulation time. Divide the desired time to generate the data (e.g., one day) by the simulation time.

2. Define the operating range for the problem under study. This is done by identifying the operating range for each ICPC as the credible minimum $z_{i,min}$ and maximum $z_{i,max}$ values for that parameter.

3. Identify the search parameter and denote it as z_1.

4. Define the desired resolution for each ICPC. This is how many different values of the parameter will be chosen for each complete cycle through the range for that parameter. Parameters expected to be most influential with respect to the performance measure should have the highest resolution. The basic idea for the procedure is to assign the resolution for each parameter according to the influential capability of the parameter on the performance measure and the desired number of simulations. In other words, the higher resolution is assigned to the parameter

with higher influential capability and the estimated simulation time should be less than the desired simulation time.

3.2.3 Boundary search

In this section, the inner loop of Figure 3.1 is described. The action of this loop results in boundary centered data generation, i.e., the data to be used to train the neural network will be centered about the boundary. In other words, there will exist samples corresponding to operating points on both sides of the boundary. This is desirable because it ultimately causes the neural network mapping function to retain high accuracy for operating points close to the boundary but to slightly decrease in accuracy as operating points become more distant from the boundary. Loss of accuracy for points distant from the boundary is not of concern because only solutions on the boundary are revealed to the operator.

3.2.3.1 Search algorithm

The algorithm requires definition of a *state*. A state is a specified operating condition in terms of all ICPCs except the search parameter, i.e., it is a unique choice of values for $z_2, z_3, \ldots, z_{M_z}$. For each state, the search parameter is varied back and forth across the boundary until the performance measure is within a tolerance of the threshold level corresponding to the boundary. A secant root finding method [50] was initially used in the search algorithm, but we found it unattractive due to its uni-directional solution approach, i.e., it does not move back and forth across the boundary. Therefore, a bisection method is used instead.

3.2.3.2 Nonconverged precontingency operating conditions

It is possible that a precontingency operating condition, as defined by a state together with a specific value of the search parameter, may result in a nonconverged power flow

solution. For each state, the search algorithm begins by simulating the contingency under two operating conditions distinguished by the values of the search parameter: $z_{1,min}$ and $z_{1,max}$. In setting up the precontingency operating conditions, if nonconvergence is detected at both extremes, then the algorithm moves to a new state. If nonconvergence is detected at one extreme but the case solves at the other extreme, then the algorithm uses the bisection method to find a converged case that is on the opposite side of the boundary from the first solved case. Then, the algorithm continues with a second bisection method initiated from the two identified solved cases to search for the boundary.

3.2.3.3 Unacceptable precontingency conditions

Even when the precontingency power flow solution converges, it is possible that it may violate an operating limit such as circuit overload or a bus voltage minimum or maximum limit. This situation is detected via the program MONITOR. This program requires specification by the user of the precontingency constraints of concern, i.e., ASAS does not check all limits by default.

In the developed approach for handling this situation, it is taken into consideration that the data is generated only for characterizing the postcontingency performance for the specific contingency of interest. Therefore, the fact is simply identified that a precontingency violation occured, but the search continues with respect to the security boundary for the contingency of interest. The user must check to discern whether the precontingency violation is more constraining than the security boundary. If it is, the precontingency constraint needs to be modeled in the visualization software.

3.2.3.4 No solution in performance evaluation

It is possible that the operating condition is both convergent and acceptable, but the simulation fails to yield a performance measure. This may occur, for example, when studying a thermal overload problem and the postcontingency power flow fails to con-

verge, or when studying voltage instability using P-V or Q-V "nose" curve analysis and the program may fail to fully get around the nose. In this case, ASAS moves to a new search parameter value and tries again to obtain a performance value for this state.

3.2.4 Revised Latin hypercube sampling

The outer loop of Figure 3.1 represents the portion of ASAS designed to select the states, where a state is a specific selection of all independent critical parameter candidates (ICPC) $z_2, \cdots, z_{M_z}$ except for the search parameter z_1, as defined above. The objective is to obtain sample data points that cover the breadth of the operating range, but with the best possible resolution. A revised Latin hypercube technique is used to accomplish this. This technique is described below.

3.2.4.1 The uniform sampling

Obtaining sample data points for the entire operating range is straightforward. With the state operating range defined by $z_{i,min} \leq z_i \leq z_{i,max}$, $\forall\, i \neq 1$, it is divided into cells (two dimensions, z_2 and z_3), cubes (three dimensions, z_2, z_3, and z_4), or, most generally, hypercubes (four or more dimensions). There are n_i designated intervals for each z_i, with each interval spanning $(z_{i,max} - z_{i,min})/n_i$. The number of states is equal to the number of hypercubes, given by $N_s = \Pi_{i=2}^{M_z} n_i$. A step-by-step advancement is initiated in which a simulation is performed and a sample data point is obtained for each hypercube. This approach requires a decision regarding which point in each hypercube to sample. One simple approach here would be to sample the center of each hypercube. This approach guarantees a uniform distribution of sampled data points.

3.2.4.2 The revised Latin hypercube sampling

This part of the approach is motivated by the assumption that neural network accuracy, for a given number of data sample points, is best when each parameter is *maximally resolved*, i.e., when, for each parameter, the number of different values equals the number of different points. Figure 3.2 illustrates uniform sampling (i.e., without the Monte Carlo sampling), where each parameter is not maximally resolved, i.e., there are nine points but only three values per parameter: 100, 200, and 300 for z_2 and 200, 400, and 600 for z_3.

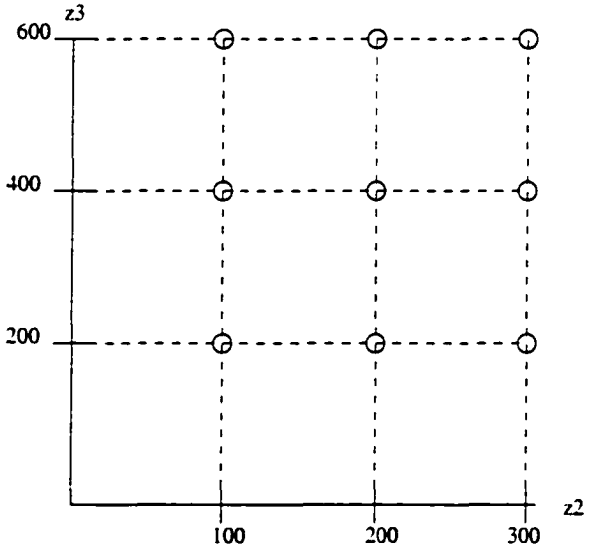


Figure 3.2 Uniform sampling in two dimensions.

The latin hypercube sampling [87, pp. 553–555] is a mixture of random and systematic sampling to ensure that each interval of a parameter is visited by exactly one Monte Carlo sampling. However, it is not guaranteed that each hypercube is also visited by exactly one Monte Carlo sampling. In consideration of this fact, a revised Latin hypercube sampling is proposed to achieve the maximum resolution for each parameter when deciding which point within each hypercube to sample. Figure 3.3 illustrates this approach for the two-dimensional case. One notes here that z_2 and z_3 each take on nine

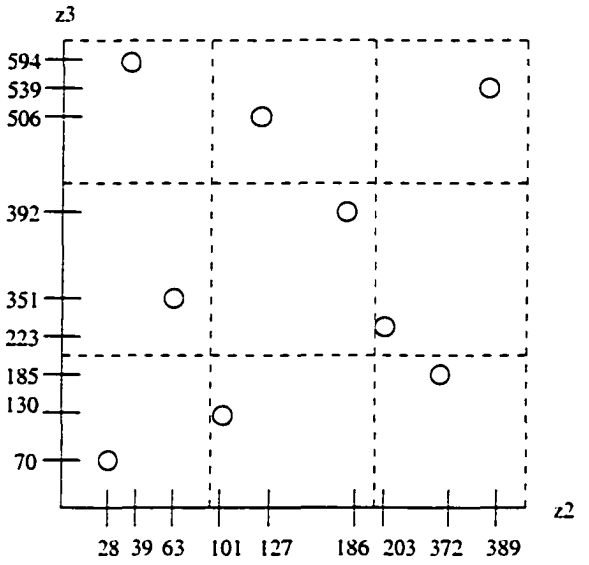


Figure 3.3 Revised Latin hypercube sampling in two dimensions.

different values instead of only three.

If the number of points is infinite, the Monte Carlo procedure is guaranteed to obtain a uniform distribution of values for each interval of each parameter. Although the number of points is typically large, this number must, of course, be finite. Therefore, there is always the risk that some regions of some parameters are never sampled while the other regions are highly sampled. This problem can be eliminated for a certain ICPC by choosing a large number of intervals n_i. However, this could result in an excessive number of simulations. In order to avoid excessive simulations while increasing the number of intervals for one ICPC, one or more other ICPCs are identified as secondary and the number of intervals n_i is chosen to be 1 for these ICPCs. Secondary ICPCs should be of less influence with respect to the system performance measure. For each state, then, the value of a secondary ICPC is chosen at random from the entire operating range of the parameter. Therefore, secondary ICPCs will vary from state to state, but their inclusion will not increase the number of simulations.

3.2.5 Data generation for the sample system

3.2.5.1 Thermal overload

For the thermal overload problem, 32 critical parameter candidates were selected. These included 4 independent critical parameter candidates (the subarea load and 3 generations) and 28 dependent critical parameter candidates. The subarea load, generation B, and generation C were divided into 8, 7, and 7 intervals, respectively. The search parameter was generation A. The four external generators were each divided into one interval, i.e., the full operating range. The purpose of varying external dispatch was not to capture the influence of these particular generators but to capture the influence of variation on the subarea tie lines, so the importance of the external generator resolutions was secondary. A total of 2568 power flow simulations were run and a data set of 1005 samples was obtained.

3.2.5.2 Voltage instability

For this problem, 5 more critical parameter candidates were added that were related to the available reactive supply close to the load bus that was being studied (Q_{max} values of 5 groups of generators). For example, Q_{max} of generator group 1 was the available reactive supply of this group of generators with real power generation A. There were three identical units in this group. Values of Q_{max} depended on how many units were commited. Therefore, Q_{max} was not a continuously-valued parameter. In this case, the search parameter could not be generation A because variation in real power generation would require the Q_{max} of group 1 also change. This would result in two parameter changes, which would make the boundary search a very complicated procedure. Therefore, the subarea load was chosen as the search parameter. Generations A, B, and C were each divided into 9, 7, and 8 intervals, respectively. The four external generators were also each divided into one interval, i.e., the full operating range. A total of 16000

power flow simulations were run and a data set of 8000 samples has been obtained by traversing the 504 ($9 \times 7 \times 8$) states four times. These samples were divided into three parts:

1. The first 2000 samples, which corresponded to one full traversal through the 504 states, were used as input to the feature selection software.

2. The first 6000 samples, which corresponded to three full traversals through the 504 states, were used for neural network training.

3. The last 2000 samples, which corresponded to one full traversal through the 504 states, were used for neural network test.

3.3 Data analysis

The purpose of data analysis is to analyze the data to ensure that data coverage of the problem space is appropriate for neural network training. In data generation, outliers are likely to occur if any module is imperfect. During on-line use, the new operating point might be an outlier with respect to the training data. Therefore, it is useful to detect outliers in either case.

3.3.1 Techniques for outlier detection

There is no formal, widely accepted, definition of what is meant by an outlier. An informal, intuitive definition is that an outlier is a data point that is inconsistent with the remainder of the data set [74, Chapter 10]. For power system security assessment, an outlier is a data point that is far away from other data points; it is an atypical operating point.

Outlier detection is useful for neural network training and prediction. During training, if an outlier appears in the training data set, it may influence the neural network

training and cause performance inaccuracy. During prediction, i.e., when a trained neural network is used for prediction, if a new data point is an outlier with respect to the training data set, the trained neural network may output a misleading result. This is because neural networks tend to perform well in data interpolation but not in extrapolation.

For a one-dimensional data set, it is relatively easy to detect outliers by direct observation of data or by simple calculation. In case of multi-dimensional data set, more advanced methods are needed because the data can not be readily plotted to pinpoint the outliers.

3.3.1.1 Principal component analysis

The central idea of principal component analysis (PCA) [74, 75] is to reduce the dimensionality of a data set in which there are a large number of interrelated variables, while retaining maximum information content. Specifically, PCA searches for a few uncorrelated linear combinations of the original variables that capture most of the variation in the original variables.

Suppose that $\mathbf{x}$ is a vector of p random variables. The first step is to find a linear transformation $\alpha_1^{\mathrm{T}}\mathbf{x}$ which has maximum variance from all values of vector $\mathbf{x}$, where α_1 is a vector of p constants, $\alpha_{11}, \alpha_{12}, \cdots, \alpha_{1p}$, so that

$$\alpha_1^{\mathrm{T}}\mathbf{x} = \sum_{j=1}^{p} \alpha_{1j} x_j$$

Next, find a linear transformation $\alpha_2^{\mathrm{T}}\mathbf{x}$, uncorrelated with $\alpha_1^{\mathrm{T}}\mathbf{x}$, which has maximum variance, and so on. The kth derived variable $\alpha_k^{\mathrm{T}}\mathbf{x}$, uncorrelated with $\alpha_1^{\mathrm{T}}\mathbf{x}$, $\alpha_2^{\mathrm{T}}\mathbf{x}$, $\cdots$, $\alpha_{k-1}^{\mathrm{T}}\mathbf{x}$, is the kth principal component. At most p PCs can be found, but it is hoped that most of the variation in $\mathbf{x}$ will be accounted for by m PCs, where $m \ll p$. It can be derived that the kth PC is given by $z_k = \alpha_k^{\mathrm{T}}\mathbf{x}$ where α_k is an eigenvector of the sample

covariance matrix given by

$$\mathbf{S} = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{x}_i - \overline{\mathbf{x}})^T \tag{3.3}$$

corresponding to its kth largest eigenvalue λ. Furthermore, if $\alpha_\mathbf{k}$ is chosen to have unit length ($\alpha_\mathbf{k}^\mathbf{T}\alpha_\mathbf{k} = \mathbf{1}$), then the variance of z_k is λ_k. The derivation of the above result can be found in reference [74, pp. 4–5].

The plot of the first two principal components may reveal some important features of the data set. This is because the plot is equivalent to a projection of the multidimensional data swarm onto the plane that shows the largest spread of the points. The resulting two-dimensional representation captures more of the overall configuration of the data than does a plot of any two of the original variables [75]. If the first two principal components account for a major portion of the total variation of the data set, then the 2-D representation of the p-dimensional data set can be examined for outliers. In general, the first two principal components tend to detect the outliers that inflate variances or covariances, where these outliers are more likely detectable by plotting the original variables. On the other hand, the last two principal components may provide additional information that is not available in plots of the original variables [74, Chapter 10].

The above method has been applied to the data sets for the thermal overload problem and the voltage instability problem. Figures 3.4 and 3.5 show the accumulative variation that is the sum of variations of individual principal components. For the thermal overload problem, it can be seen that the first two principal components account for 71.5% of the total variation. Then, the first two principal components are plotted as shown in Figure 3.6. The last two principal components are shown in Figure 3.7. From the plots, no outliers are observed. For the voltage instability problem, the same method can not be used for outlier detection because the first two PCs account for only 35.5% of the total variation.

To assess the effectiveness of the principal component analysis method, sample 10

in the data set for the thermal overload problem was altered. The Subarea Load, Generation A, Generation B, and Generation C were changed from 0.7670, 1.0000, 0.9764, 0.0822 to 1.0000, 0.0000, 0.0000, 0.0000 in per unit, respectively. The new sample 10 means that the Subarea Load was at its maximum, but no local generations were available in this subarea. This situation was not in accordance with the others so that it was an outlier. For the new data set, the first and last two principal components are shown in Figure 3.8 and Figure 3.9. It can be seen that sample 10 is not detectable by plotting the first two principal components, but it is standing out as an outlier in the plot for the last two principal components.

The principal component analysis method is appropriate for analyzing the training data set, but it is not appropriate for on-line determining whether a new sample is an outlier or not because of its computational time.

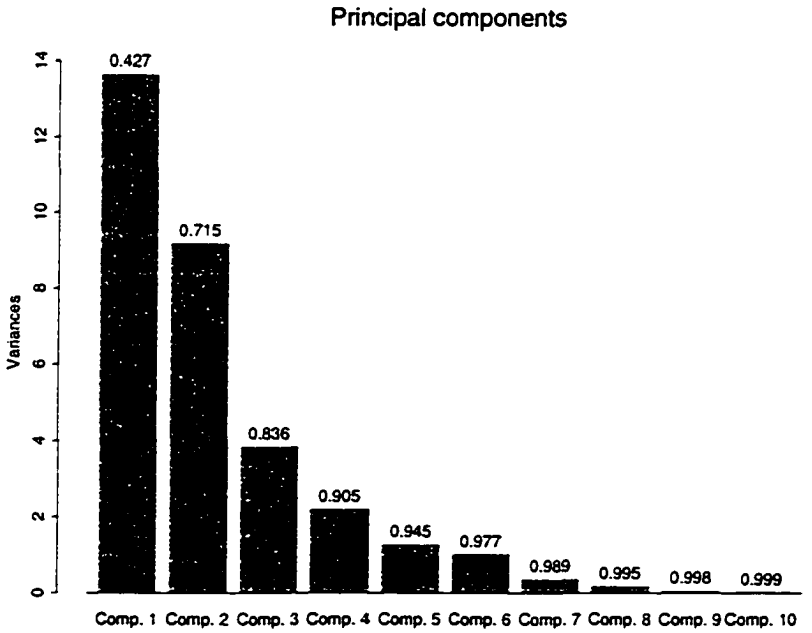


Figure 3.4 Accumulative variation of principal components for the thermal overload problem.

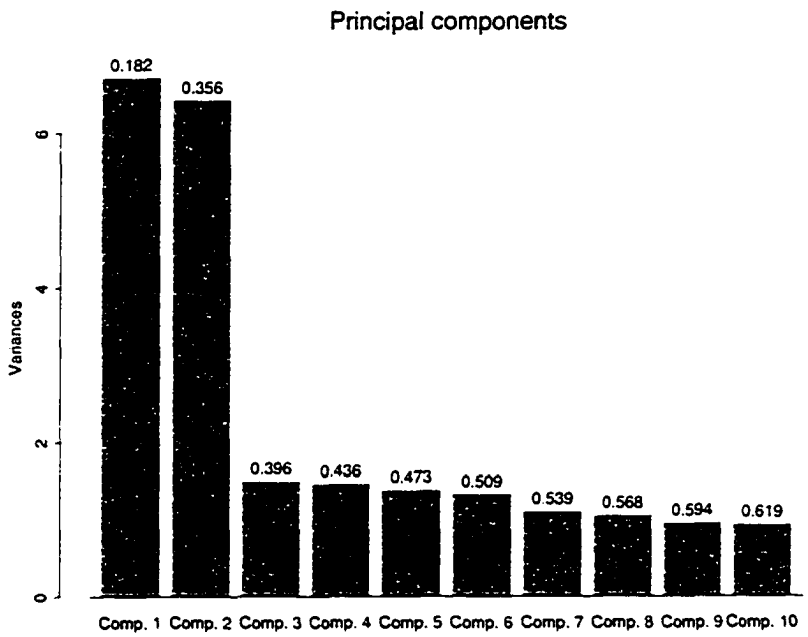


Figure 3.5 Accumulative variation of principal components for the voltage instability problem.

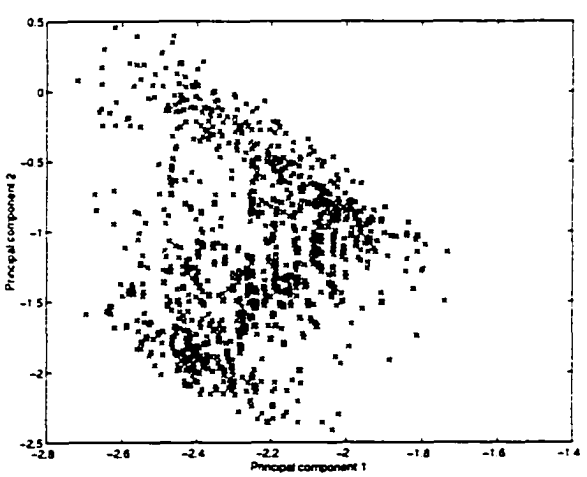


Figure 3.6 The first two principal components for the thermal overload problem with the original data set.

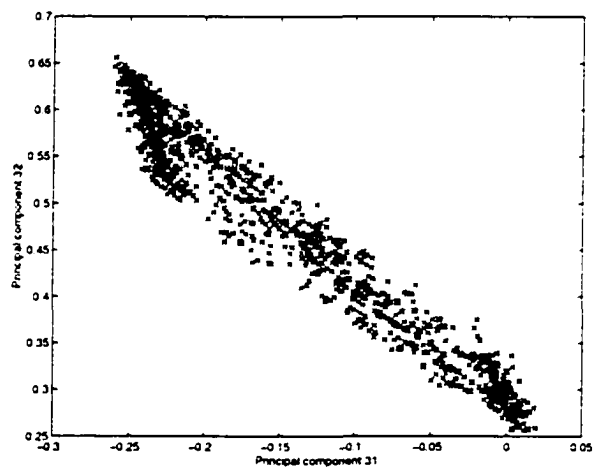


Figure 3.7 The last two principal components for the thermal overload problem with the original data set.

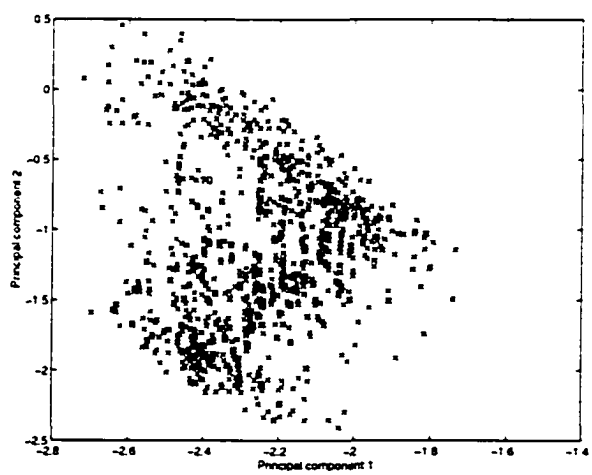


Figure 3.8 The first two principal components for the thermal overload problem with the altered data set.

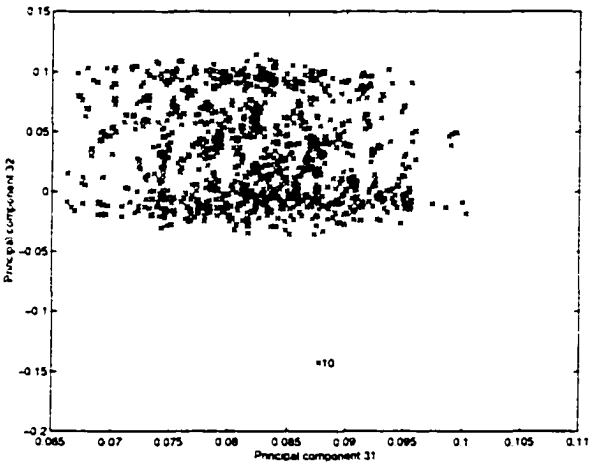


Figure 3.9 The last two principal components for the thermal overload problem with the altered data set.

3.3.1.2 Local training data density estimation

One can estimate the training data density around a given data point using the k-nearest neighbor method. If this local data density is below a given value, this data point is identified as one that lies outside the range of the training data set. This is particularly useful when the trained neural network is used to predict the result for a new data point, as would be the case when boundaries are drawn for on-line security assessment. If the new data point is an outlier, the neural network output is not reliable.

For a given point (e.g., the current operating point), the outlier detection algorithm identifies the N nearest neighbors within the training data, and then computes the average distance from the given point to these N nearest neighbors where N is determined by experience. The distance between the given point and one of the N nearest neighbors (a training point) is computed as the Euclidean distance. That is, if one denotes the

given point by $\mathbf{p} = [p_1 p_2 ... p_m]^T$ and the i-th nearest neighbor as $\mathbf{q}_i = [q_{i1} q_{i2} ... q_{im}]^T$ where the elements of each vector are the normalized values of the critical parameters used to characterize the security problem, then the distance between the given point $\mathbf{p}$ and the i-th nearest neighbor $\mathbf{q}_i$ is

$$d_i = ||\mathbf{p} - \mathbf{q}_i|| = \sqrt{(p_1 - q_{i1})^2 + (p_2 - q_{i2})^2 + ... + (p_m - q_{im})^2}$$

and the average distance is given by

$$D = \frac{\sum_{i=1}^{N} d_i}{N}$$

The local training data density is defined as

$$\rho = \frac{1}{D}$$

This method has been applied to the voltage instability problem. From the test data of 2,000 samples that were not used for the neural network training, a sample set of 219 data points was drawn and the neural network test error against the local training data density was plotted. A total of 19 points were generated outside the range of training data set so that they were considered to be outliers. A test on these 19 points was also performed. Figure 3.10 provides a plot of error against ρ for each of the 19 outliers and for each of the 219 test data points with $N = 50$. The circles represent the outliers and the crosses represent the 219 test data points.

In Figure 3.10, the crosses are concentrated in the lower right portion of the plot, indicating that, for these test data points, the local training data density is relatively high and the neural network error is relatively small. In contrast, the circles are concentrated on the upper left portion of the plot, indicating that, for the outliers, the density is relatively low and the neural network error is relatively large. These two results indicate that neural network error and the local training data density are related: the lower the density, the greater will be the error. From the plot, we see that as long as the density is

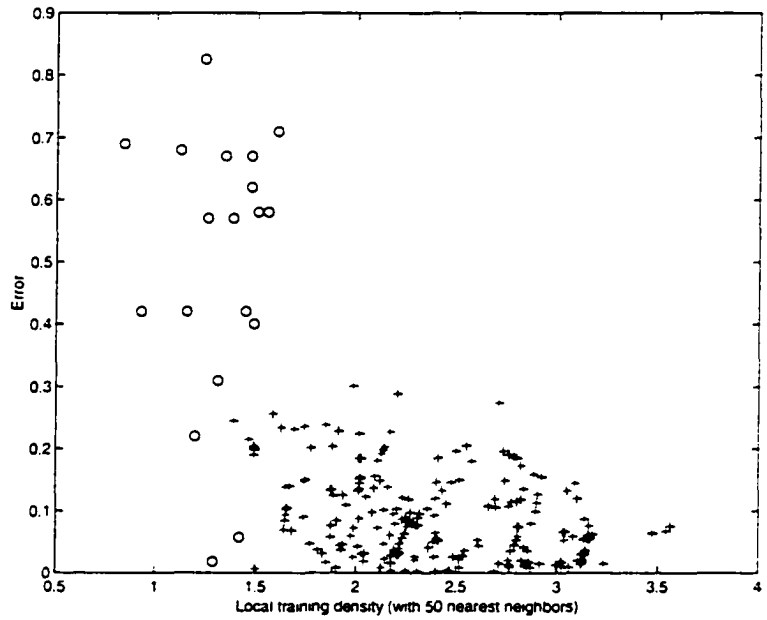


Figure 3.10 Error versus density in training data

greater than about 1.7 the error will be less than 20% (of 200 MVAr). This provides us with a rule for assessing the reliability of the neural network evaluation for a new data point; compute the density ρ and determine whether this density is greater than 1.7: if so, the resulting boundary should be identified as potentially inaccurate. The local training data density method is appropriate for on-line use because it is fast.

3.3.1.3 Multivariate normality test

A procedure based on the standardized distance from each sample point $\mathbf{x}_i$ to the sample mean point $\overline{\mathbf{x}}$

$$D_i^2 = (\mathbf{x}_i - \overline{\mathbf{x}})^T \mathbf{S}^{-1} (\mathbf{x}_i - \overline{\mathbf{x}})$$

can be used to detect which data point is a possible outlier, where $\mathbf{S}$ is the sample covariance matrix defined as

$$\mathbf{S} = \frac{1}{n-1} \sum_{i=1}^{n} (\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{x}_i - \overline{\mathbf{x}})^T$$

If the $\mathbf{x}_i$'s are multivariate normal, then

$$u_i = \frac{nD_i^2}{(n-1)^2}$$

has a β distribution where n is the number of samples. The u_i's are ranked ascendingly and then the quantiles of the β distribution

$$v_i = \frac{i-\alpha}{n-\alpha-\beta+1}$$

are calculated where

$$\alpha = \frac{p-2}{2p}$$

$$\beta = \frac{n-p-2}{2(n-p-1)}$$

Quantiles are similar to the more familiar percentiles, which are expressed in terms of percent: a test score at the 90th percentile, for example, is above 90% of the test scores and below 10% of them. Quantiles are expressed in terms of fractions or proportions. Thus the 90th percentile score becomes the 0.9 quantile score. Finally, the Q–Q (quantile–quantile) plot of (v_i, u_i) is obtained. The Q–Q plot is a graphical way to compare two distribution functions F_1 and F_2 with regard to their shape. It is simply a plot of points whose x-coordinate is the pth quantile of F_1 and whose y-coordinate is the pth quantile of F_2. The relationship between F_1 and F_2 can be revealed by the shape of the Q–Q plot. For example, the Q–Q plot will be a straight line through the origin with slope of 1 if F_1 and F_2 are the same; it will still be a straight line with different slope and intercept if F_1 and F_2 differ only in scale and location. For details about Q–Q plots, see reference [75, pp. 105–107]. From the Q–Q plot of (v_i, u_i), it can be observed that a departure from normality occurs if a nonlinear pattern exists in the plot. A formal significant test is also available for $D_{(n)}^2 = \max D_i^2$, but with only low dimensional data sets. With high dimensional data sets, we can first check if there is a significant change on D_i^2's; if yes, outliers possibly occur and then the Q–Q plot can be used to confirm.

Figure 3.11 shows the plot of D_i^2's for the thermal overload problem with 1,005 samples. It can be seen that there is one data point (top right corner of Figure 3.11) which makes a significant change on D_i^2's. This data point will be referred to as sample 265. Therefore, sample 265 is a possible outlier. Figure 3.12 shows the corresponding Q-Q plot of (v_i, u_i). From the plot, it can be seen that sample 265 stands out clearly. Therefore, it is an outlier with respect to a normality distribution. In other words, one can say that the data distribution is not normal with respect to sample 265. Figure 3.13 shows the Q-Q plot of (v_i, u_i) for the same problem with sample 265 removed.

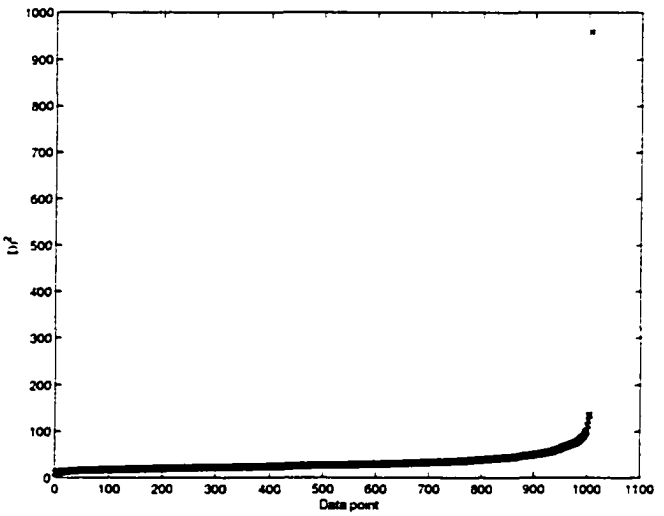


Figure 3.11 Plot of D_i^2's for the thermal overload data set with 1,005 samples

For the voltage instability problem, D_i^2's were also calculated and there was no significant change found. Thus, no outliers in the data set were identified.

The multivariate normality test can be used to determine whether or not the distribution of the training data set is normal.

3.3.1.4 Kohonen self-organizing map

Kohonen self-organizing map [77, Chapter 10] attempts to represent a data set by a smaller data set, by reducing either the number of input patterns or the number of

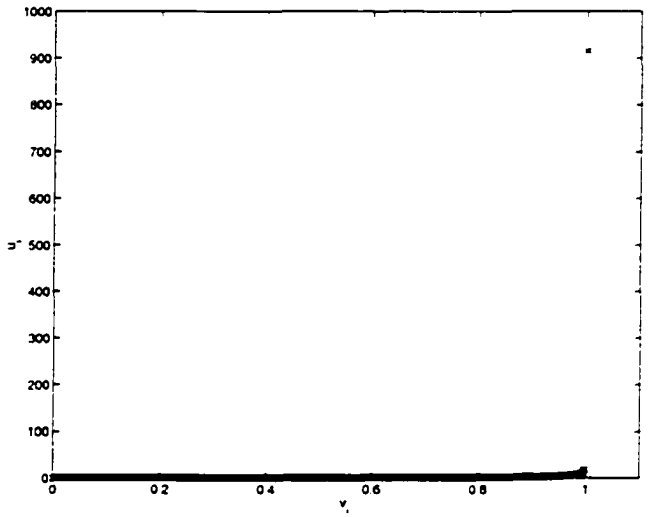


Figure 3.12 Q–Q plot of v_i and u_i for the thermal overload data set with 1,005 samples

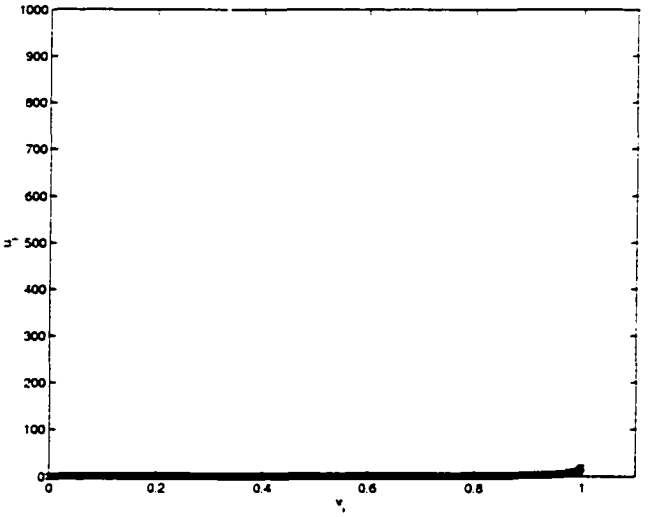


Figure 3.13 Q–Q plot of v_i and u_i for the thermal overload data set with 1,004 samples (sample 265 removed)

attributes while topologically preserving similarities present in the input vectors. It belongs to the class of unsupervised learning algorithms that are complementary to the classical statistical nonparametric data analysis techniques.

The learning algorithm of the Kohonen self-organizing map can be described as follows:

1. *Initialization:* Set random values for the initial weights $\mathbf{w}_j(0), j = 1, \cdots, N$ where N is the number of neurons in the map.

2. *Sampling:* Choose an input sample $\mathbf{x}$ randomly from the training data set.

3. *Similarity matching:* Find the best-matching (winning) neuron $i(\mathbf{x})$ at time n, using the minimum Euclidean distance criterion:

$$i(\mathbf{x}) = arg_j \min \|\mathbf{x}(n) - \mathbf{w}_j\|, \ j = 1, \cdots, N \tag{3.4}$$

4. *Updating:* Adjust the weight vectors of all neurons, using the update formula

$$\mathbf{w}_j(n+1) = \begin{cases} \mathbf{w}_j(n) + \eta(n)[\mathbf{x}(n) - \mathbf{w}_j(n)], & j \in \Lambda_{i(\mathbf{x})}(n) \\ \mathbf{w}_j(n), & \text{otherwise} \end{cases} \tag{3.5}$$

where $\eta(n)$ is the learning rate, and $\Lambda_{i(\mathbf{x})}(n)$ is the neighborhood function centered around the winning neuron $i(\mathbf{x})$; both of the parameters are varied dynamically during learing.

5. *Continuation:* Continue with Step 2 until no noticeable changes in the map are observed.

The procedure to use a Kohonen self-organizing map to detect outliers is as follows:

- Use the training data set to train a Kohonen self-organizing map so that a topology preserving 2-D grid is formed.

- Observe whether there are isolated points in the 2-D grid. If yes, these isolated points correspond to the outliers in the training data set.

A 10×10 grid is chosen to represent the Kohonen network. Any data point is mapped to one of the nodes in the network. Figures 3.14 and 3.15 are the maps for the thermal overload problem (1.005 samples) and voltage instability problem (2.000 samples). One could not find any isolated node in the maps, therefore no outlier could be identified using this method.

The main advantage of the Kohonen self-organizing map lies in its graphical interpretation capability. Also, the number of neurons is independent of the dimension of the input vector and of the size of the training data set. However, there is no theoretical guidance in choosing the appropriate number of neurons in the map.

3.3.2 Techniques for outlier detection and distribution visualization

While it is important to do outlier detection for the training data set, meanwhile, it must be ensured that the training data set provides an accurate representation of the entire problem space. The actual distribution of the parameters may be not uniform or well-resolved due to various practical limitations such as constraints among parameters, simulation tool limitations, or unrealistic situations leading to power flow nonconvergence. The following techniques can not only detect outliers but also visualize the data distribution.

3.3.2.1 Scatter plots

1-D scatter plots

These plots can provide an intuitive feeling about the range and distribution of an attribute. They also help to detect possible outliers. These plots have been obtained for the thermal overload problem and the voltage instability problem. In the plots,

each point corresponds to each sample. If there are multiple samples with the same value for the plotted parameter, then the multiple points are evenly plotted at that value. Figures 3.16 and fig:a19 are two examples. It can be observed that the overall distribution is good (i.e., the entire problem space is covered) although sample 996 is slightly far away from other sample points.

2-D scatter plots

These plots can provide a graphical interpretation on whether the data set is well distributed throughout the problem space. Two kinds of plots can be obtained. One is a plot of attributes versus the performance measure. The other is a plot of a pair of attributes. The plots have been obtained for the thermal overload problem and the voltage instability problem. Figures 3.18–3.21 are some of them.

3-D trajectory plots

These plots can provide a dynamic visualization on ASAS procedures, i.e., how the training dat set is generated. Figure 3.22 shows a trajectory inside a 3-D space representing the three precontingency critical parameters.

3.3.2.2 Projection pursuit visualization

The 1-D, 2-D, and 3-D plots have been obtained for the data distribution analysis over the problem space. However, these plots can only provide certain views on data distribution. An obvious remedy is to look at the high-dimensional point clouds, and base the description of the structure on these views. As human perception in more than three dimensions is difficult, the dimensionality has to be reduced, most simply, by projection. Since projection of the data generally implies loss of information, multivariate structure does not usually show up in all projections, and no single projection might contain all the information. Therefore, it is important to judiciously choose the set of projections

on which the structure shows up. The projection pursuit visualization [84, 85] uses the gradient of a projection pursuit function to visualize the high-dimensional data set from various interesting views in order to search for structures in high-dimensional space. The resulting plots are two-dimensional data clouds where the two coordinate axes represent two projection directions and each point corresponds to a sample.

Figures 3.23 - 3.25 are examples of views for the thermal overload with the original data set. Figures 3.26 - 3.28 are examples of views for the thermal overload with the altered data set. It can be observed that the data distribution for the thermal overload problem is good. But sample 996 is an outlier for the original and altered data sets. Sample 10, as expected, is an outlier for the altered data set. Figure 3.29 is an example of views for the voltage instability problem. It can also be seen that the data distribution for the voltage instability problem is good.

The projection pursuit visualization method is an effective tool for interactively analyzing the training data set.

3.4 Summary

Data generation and data analysis have been discussed in this chapter. In data generation, a systematic approch has been developed to automatically generate a database for any type of security problem by using the correponding simulation tool. The developed software uses boundary search and revised Latin hypercube sampling to ensure that the generated sample points are centered around the boundary and they are highly resolved and well distributed. Various data sets have been generated for the thermal overload and voltage instability problems.

Data analysis techniques are used to analyze the training data set to ensure that the data coverage of the problem space is appropriate for neural network training and prediction. Two aspects have been investigated: outlier detection and distribution anal-

ysis. The techniques for outlier detection are the principal component analysis, the local training data density estimation, the multivariate normality test, the Kohonen self-organizing map, the scatter plots, and the projection pursuit visualization. Of the above, the scatter plots and the projection pursuit visualization can also be used to analyze data distribution. The local training data density estimation is appropriate for determining whether a new sample is an outlier with respect to the training data set. All of the other methods are most appropriate for analyzing the training data set before neural network training, and they are complementary to each other. As a general procedure, start with the principal component analysis, then use the multivariate normality test, finally do the projection pursuit visualization. At the same time, the scatter plots can be used as a simple tool for data analysis. The Kohonen self-organizing map is not recommended unless one can choose a reasonable number of neurons on the map.

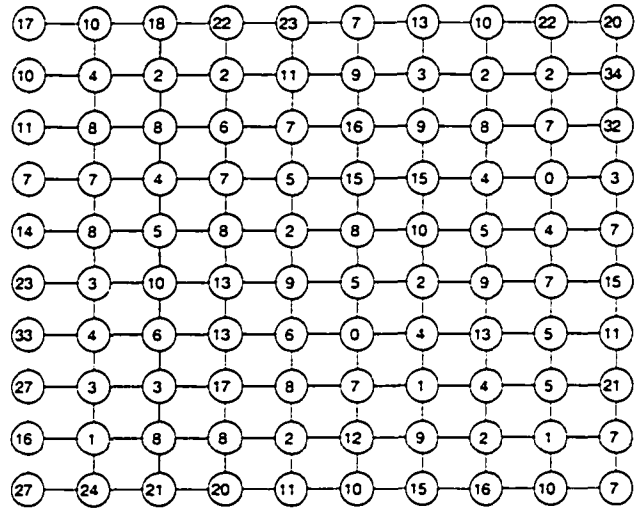


Figure 3.14 Organization of a Kohonen self-organizing map for the thermal overload problem

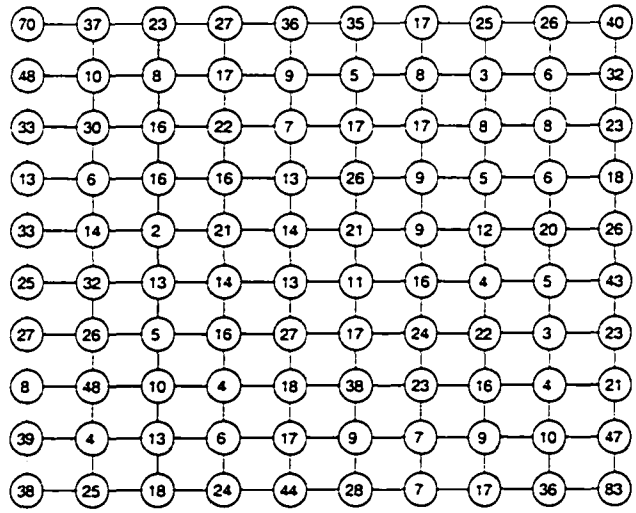


Figure 3.15 Organization of a Kohonen self-organizing map for the voltage instability problem

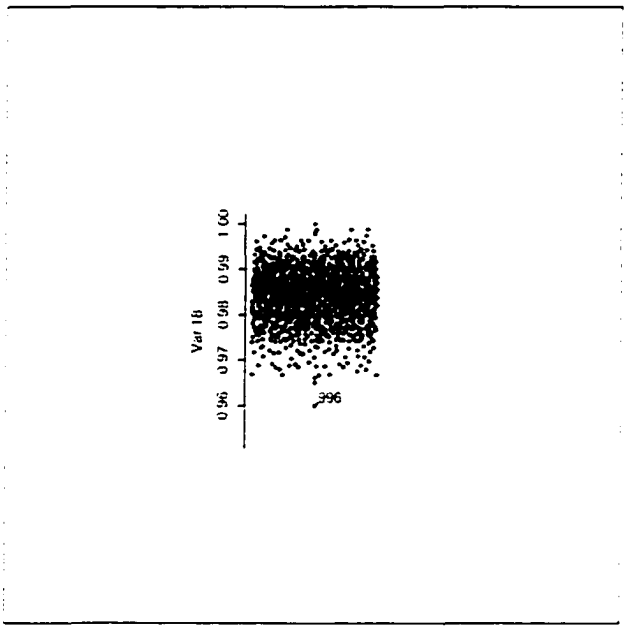


Figure 3.16 Attribute 18 (voltage magnitude) for the thermal overload problem.

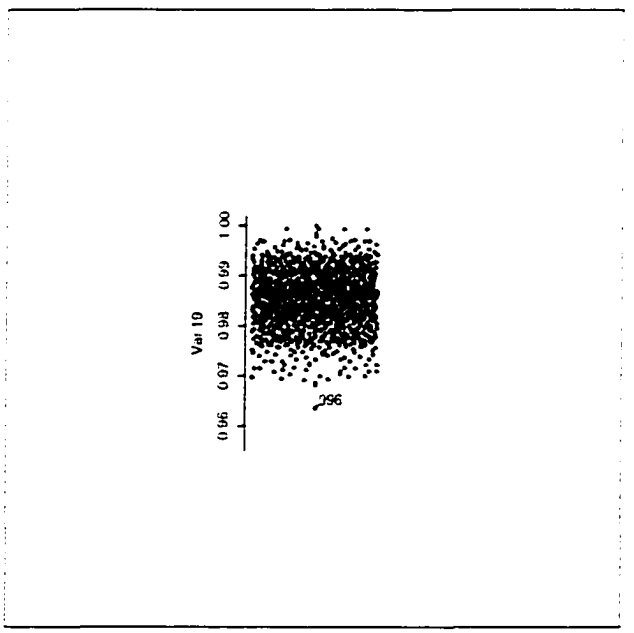


Figure 3.17 Attribute 19 (voltage magnitude) for the thermal overload problem.

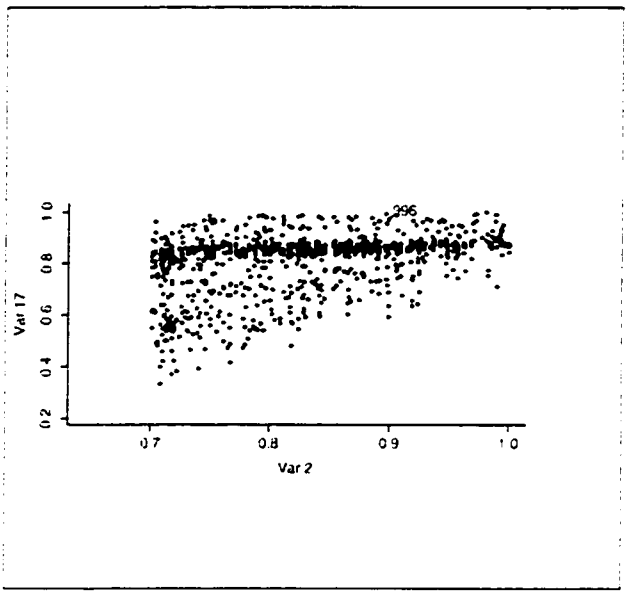


Figure 3.18 Attribute 17 (line flow) versus subarea load for the thermal overload problem.

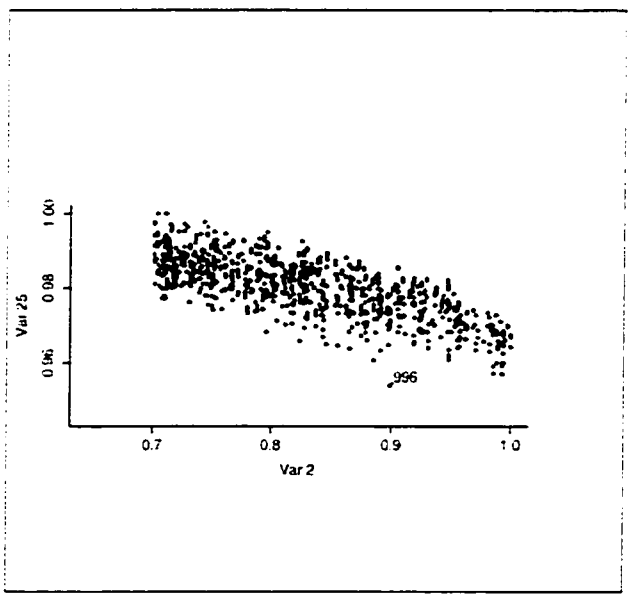


Figure 3.19 Voltage magnitude 7 versus subarea load for the thermal overload problem.

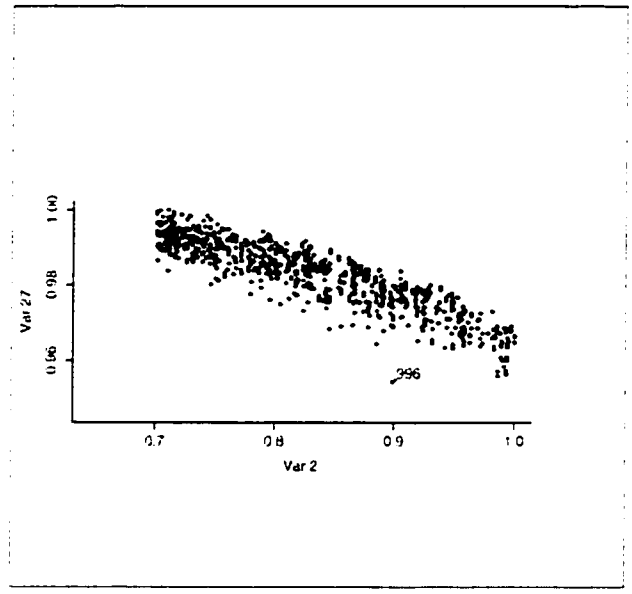


Figure 3.20 Voltage magnitude 9 versus subarea load for the thermal overload problem.

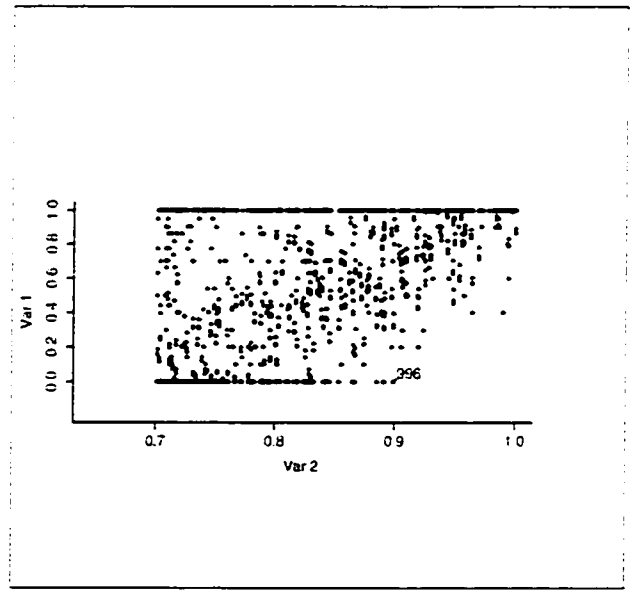


Figure 3.21 Generation A versus subarea load for the thermal overload problem.

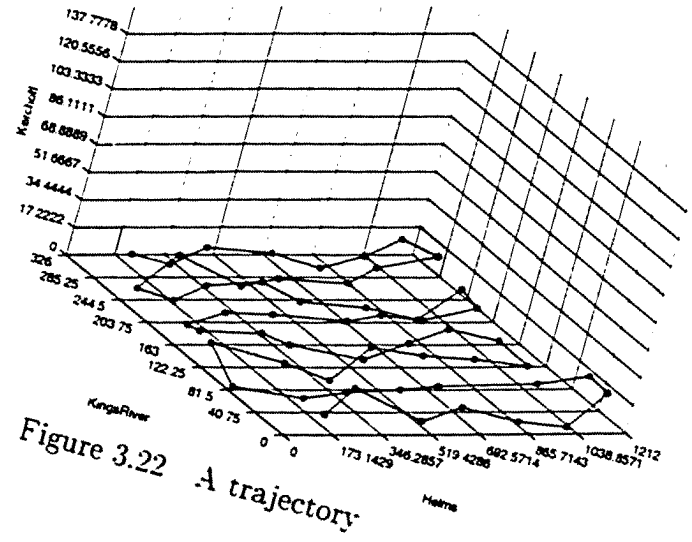


Figure 3.22 A trajectory

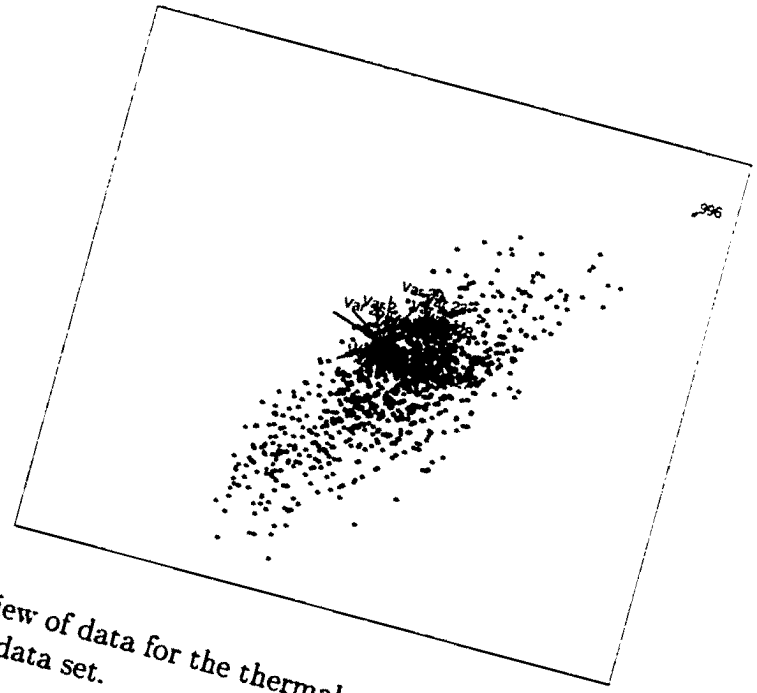


Figure 3.23 A view of data for the thermal overload problem with the original data set.

Figure 3.24 A view of data for the thermal overload problem with the original data set.

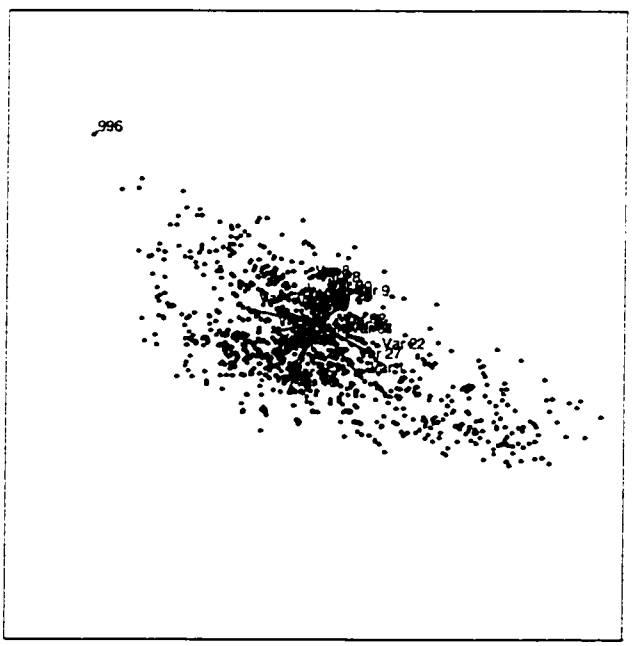


Figure 3.25 A view of data for the thermal overload problem with the original data set.

Figure 3.26 A view of data for the thermal overload problem with the altered data set.

Figure 3.27 A view of data for the thermal overload problem with the altered data set.

Figure 3.28 A view of data for the thermal overload problem with the altered data set.

Figure 3.29 A view of data for the voltage instability problem.

4 FEATURE SELECTION

4.1 Introduction

Security assessment studies have traditionally depended on engineering judgment to select the critical parameters, i.e., to select the parameters to be used in characterizing the boundary for each security problem. The goal of this chapter is to develop an automatic approach to select critical parameters from the critical parameter candidates. More specifically, the goal is to select the best critical parameters for the entire operating range, not necessarily for any one particular operating point. Once candidate critical parameters have been selected and a database has been generated, this becomes a standard feature selection problem. It is not intended that automatic feature selection will replace the engineering judgment, but rather it is intended that it will enrich the judgment by confirming and extending physical understanding.

Feature selection is a process to identify a subset of the initial attributes, which are also called features or parameters. The words attribute, feature, and parameter will be used interchangeably in this dissertation. It can be thought of as a search problem over the space of all combinations of features where a function is used to evaluate what is "best". Also, it may be possible to combine and mathematically transform the original attributes so that one can obtain new attributes that condense the maximum discriminatory information from the available data. This is so called feature extraction. In this dissertation, no distinction between these two cases, feature selection and feature extraction, is treated; both are called feature selection.

Following a brief literature overview in the next section, three classes of feature selection methods are discussed: cluster analysis, Kohonen self-organizing map, and genetic algorithm based methods.

4.2 Literature overview

As indicated in reference [61], researchers in the statistics and pattern recognition communities have investigated feature subset selection problems for decades [51, 52, 53, 54, 55, 56], but most work has concentrated on subset selection using linear regression, which is not very effective for highly non-linear problems. However, some statistical methods, such as principal component analysis, are still useful for analyzing some relationships among features. Zayan et al. [63] have compared the minimum entropy method and the *Karhunen – Loève* expansion method for dynamic security classification and concluded that the former is better. Muknahallipatna et al. [64] have compared the linear correlation method and discriminant analysis and found that the latter is better. In the machine learning literature, the most closely related work is MLC++ software[1] and Predict software [57]. MLC++ uses a statistical method to search solution space. The Predict software, in a way similar to [58], uses the genetic algorithm to search the solution space. Downey et al. [59] also use the genetic algorithm to search the solution space but use a statistical method to do the evaluation.

The selection of attributes is based only on the ability to predict the output. There is no efficient way to account for cardinality constraints on the feature set although there are several techniques that have attempted to find minimum feature subsets (see [62] for a discussion). Furthermore, there is no capability to preselect some features into the solution. Therefore, a genetic algorithm based neural network feature selection tool was developed in [61] and refined in this dissertation.

[1]This machine learning software was developed by Stanford University and is available in the public domain.

4.3 Critical parameter candidates for the sample system

In the sample as introduced in Chapter 1, there are 32 critical parameter candidates for the thermal overload problem and 37 critical parameter candidates for the voltage instability problem. They are listed in Table 4.1. The objective of feature selection is to select critical parameters from these candidates.

Table 4.1 Critical parameter candidates for the thermal overload and voltage instability problems

Attribute number	Thermal overload	Voltage instability	Attribute number	Thermal overload	Voltage instability
1	GenA	GenA	20	Volt3	Volt3
2	Load	Load	21	Volt4	Volt4
3	Gen2	Gen2	22	Volt5	Volt5
4	Gen3	Gen3	23	Volt6	Volt6
5	Flow1	Flow1	24	Volt7	Volt7
6	Flow2	Flow2	25	Volt8	Volt8
7	Flow3	Flow3	26	Volt9	Volt9
8	Flow4	Flow4	27	Volt10	Volt10
9	Flow5	Flow5	28	Volt11	Volt11
10	Flow6	Flow6	29	Volt12	Volt12
11	Flow7	Flow7	30	Volt13	Volt13
12	Flow8	Flow8	31	Volt14	Volt14
13	Flow9	Flow9	32	Flow14	Flow14
14	Flow10	Flow10	33		Qmax1
15	Flow11	Flow11	34		Qmax2
16	Flow12	Flow12	35		Qmax3
17	Flow13	Flow13	36		Qmax4
18	Volt1	Volt1	37		Qmax5
19	Volt2	Volt2			

4.4 Statistical method – Cluster analysis

Cluster analysis [73] can be used to analyze similarities or dissimilarities among groups of objects, i.e., patterns (Q-analysis) or attributes (R-analysis). There are many cluster analysis methods in statistics. Hierarchical cluster analysis is one which results in a tree or dendrogram showing the hierarchy of similarities among all pairs of objects. According to [73, p.21], hierarchical cluster analysis is more widely used than nonhier-

archical cluster analysis such as the k-means clustering method because of its simplicity in mathematics and its effectiveness in practice. The general procedure for hierarchical cluster analysis is as follows:

- *Form a data matrix*: The columns represent attributes and the rows represent patterns.
- *Normalize the data matrix*: Normalize the data so that attributes contribute more equally to the similarity calculation. For example, if the range of values on the first attribute is much greater than the range of values of the second, the first attribute will carry more weight in determining the similarities among objects.
- *Compute similarities between pairs of objects*: Use a distance measure (e.g., Euclidean) to compute similarities between pairs.
- *Execute a clustering method to produce a tree or dendrogram*: Show the hierarchy of similarities between pairs of patterns or attributes using a tree or dendrogram.

This is especially applicable to attribute similarity analysis or when the number of patterns is not large so that one can clearly plot the clustering tree.

The hierarchical cluster analysis has been applied to the data set for the thermal overload and voltage instability problems. The clustering tree for the thermal overload problem is shown in Figure 4.1. One can see that the attributes form several clusters according to their similarities. The result is reasonable in terms of physical interpretations of these attributes. For example, attributes 9 (Flow5) and 17 (Flow13), 10 (Flow6) and 11 (Flow7) are flows of circuits in series, respectively, and attributes 16 (Flow12) and 32 (Flow14) are flows of a transformer and a line in series. As to attributes 13 (Flow9) and 14 (Flow10), they are effectively in parallel from inspection of the one-line diagram and observation of power flow variations. The plot shows that the two attributes in each

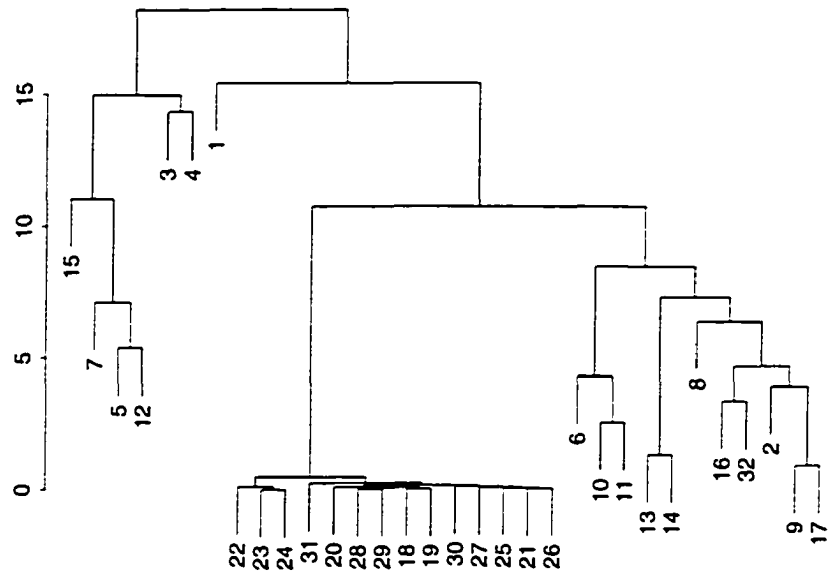


Figure 4.1 Cluster tree for the data set of the thermal overload problem

pair are located closely in the tree. Attributes 18 (Volt1) to 31 (Volt14) are all of the bus voltages located in the same region of the system.

Figure 4.2 shows the clustering tree for the voltage instability problem. One can also observe the similar phenomena. For example, attributes 9 (Flow5) and 17 (Flow13), 10 (Flow6) and 11 (Flow7) are flows of circuits in series, respectively.

For feature selection, a straightforward method based on cluster analysis is to pick up one attribute from each major cluster, which includes more attributes than other clusters. For the thermal overload problem with cardinality level 8, for example, attributes 5 (Flow1), 9 (Flow5), 10 (Flow6), and 18 (Volt1) can be selected attributes in addition to preselected attributes 1 to 4. However, the cardinality level should be specified in advance.

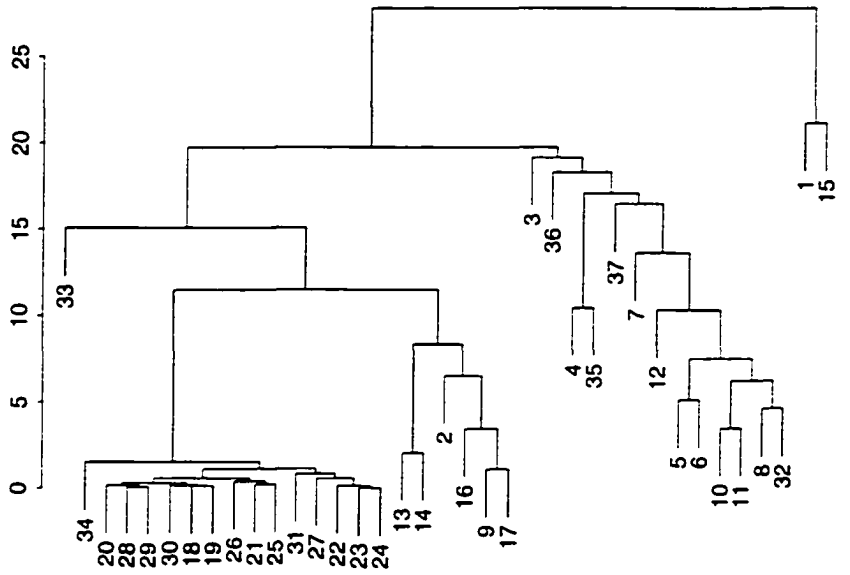


Figure 4.2 Cluster tree for the data set of the voltage instability problem

4.5 Neural network method – Kohonen self-organizing map

As discussed in Chapter 3, the Kohonen self-organizing map attempts to represent a data set by a smaller one, by reducing either the number of input patterns or the number of attributes while topologically preserving similarities present in the input vectors. When it is used to reduce the number of attributes, it can be viewed as a feature selection tool.

Figures 4.3 and 4.4 are organized maps for the thermal overload and voltage instability problems. The number in a circle represents the number of attributes clustered. Table 4.2 lists each node with the associated attributes. From the maps and the table, it can be seen that the results are quite similar to those from cluster analysis. For instance, attributes 18 through 31 for the thermal overload problem are clustered because they are all bus voltage magnitudes. Features 9 and 17 are clustered for both thermal overload and voltage instability problems.

For feature selection, a straightforward method based on Kohonen self-organizing

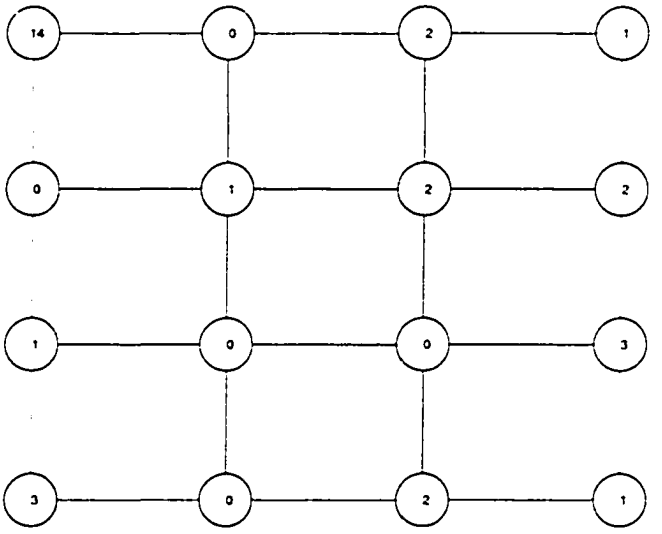


Figure 4.3 A 4×4 organized map of attributes for the thermal overload problem.

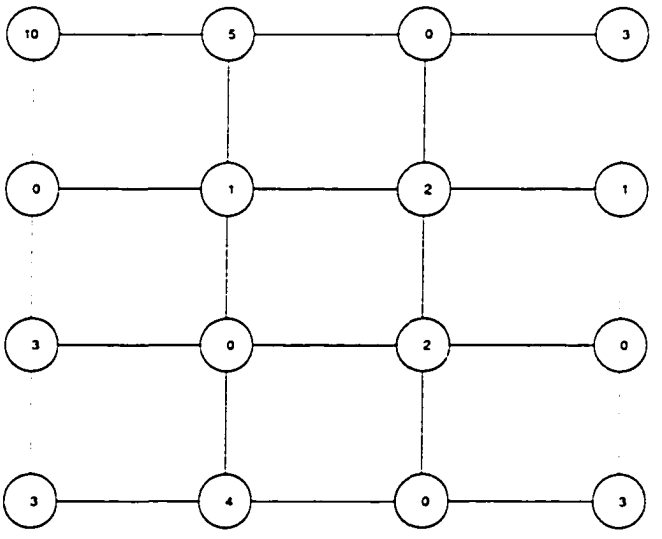


Figure 4.4 A 4×4 organized map of attributes for the voltage instability problem.

Table 4.2 Nodes on the map with their associated attributes for the thermal overload problem.

Node	Features (Thermal overload)	Features (Voltage instability)
1	18–31 (Volt1–Volt14)	18–21 (Volt1–Volt4), 25 (Volt8), 26 (Volt9), 28–30 (Volt11–Volt13), 34 (Qmax2)
2		22–24 (Volt5–Volt7), 27 (Volt10), 31 (Volt14)
3	9 (Flow5), 17 (Flow13)	
4	8 (Flow12)	13 (Flow9), 14 (Flow10), 33 (Qmax1)
5		
6	16 (Flow12)	2 (Load)
7	10 (Flow6), 32 (Flow14)	4 (Gen3), 35 (Qmax3)
8	6 (Flow2), 11 (Flow7)	1 (Gen1)
9	2 (Load)	9 (Flow5), 16 (Flow12), 17 (Flow13)
10		
11		36 (Qmax4), 37 (Qmax5)
12	5 (Flow1), 7 (Flow3), 12 (Flow8)	
13	1 (Gen1), 13 (Flow9), 14 (Flow10)	8 (Flow4), 10 (Flow6), 32 (Flow14)
14		5 (Flow1), 6 (Flow2), 11 (Flow7), 12 (Flow8)
15	3 (Gen2), 4 (Gen3)	
16	15 (Flow11)	3 (Gen2), 7 (Flow3), 15 (Flow11)

map is to pick up one attribute from each major node, which includes more attributes than other nodes. For the thermal overload problem with cardinality level 8, for example, attributes 5 (Flow1), 9 (Flow5), 10 (Flow6), and 18 (Volt1) can be selected attributes in addition to preselected attributes 1 to 4. However, the cardinality level should be specified in advance.

4.6 Genetic algorithm method

The genetic algorithm (GA) [65, 66] is used to search the feature space. The evaluation function is the fitness function which includes network accuracy, feature cardinality, and feature constraints such as set controllability. Figure 4.5 illustrates the basic functional structure of the developed genetic algorithm based neural network feature selection tool (GANN), which was initially designed and implemented by Mr. Qianglin Zhao [61].

4.6.1 Design of GANN

Figure 4.5 shows the functional structure of GANN. Since the two important components are the search approach and the evaluation function, they are discussed first, and then the procedure in Figure 4.5 is explained.

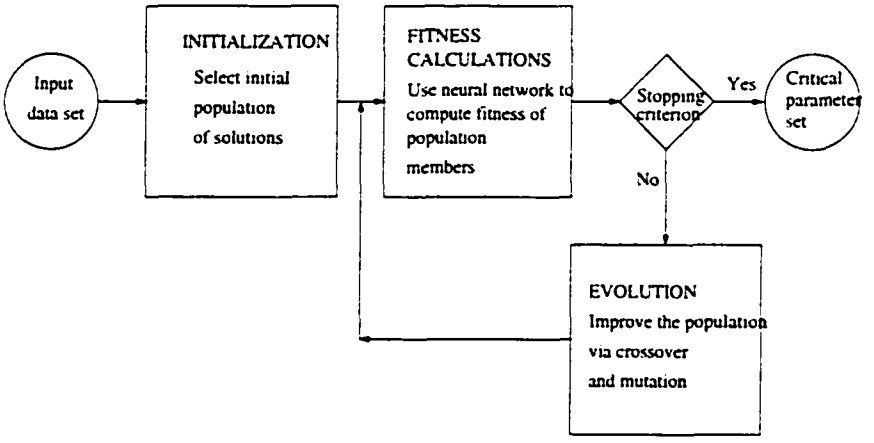


Figure 4.5 Basic functional structure of GANN

4.6.1.1 Search Approach

If there are n candidate features, the solution space will consist of 2^n possible solutions. Even if the number of candidate features can be reduced to 30 or 40 using engineering judgment, the solution space is still very large ($2^{30} = 1.1 \times 10^9$) so that exhaustive search is not practical. Genetic algorithms are very effective in solving combinatorial problems for which a performance measure, i.e., a fitness function, can be defined for each combination under consideration. Therefore, the GA is used for an efficient search for the best subset of features. In this design, the user specifies parameters that should be forced into the solution to satisfy the feature controllability constraint. The user also specifies a cardinality range. The GA then identifies the most accurate solution containing the forced parameters for each cardinality level in the spec-

ified range. Finally, the user can easily select the solution that offers the best tradeoff between accuracy and cardinality.

4.6.1.2 Evaluation function

The evaluation function is used in fitness calculations, as shown in Figure 4.5, to calculate the fitness for a possible solution. The larger the fitness, the better the feature subset. The fitness of the i-th individual in the population can be computed according to $F_i = w_1 F_{1i} + w_2 F_{2i}$ where F_{1i} and F_{2i} are the fitness components corresponding to the neural network accuracy and cardinality, respectively, and w_1 and w_2 are their corresponding weights. F_{1i} is taken as the reciprocal of the average absolute error computed by testing a neural network trained with inputs corresponding to the selected attributes of the individual. F_{2i} is equal to n_j, the cardinality level of individual j, if n_j is smaller than the desired cardinality level; otherwise, it is zero. Weights w_1 and w_2 are set to 1.0 and 3000.0, respectively. The cardinality component is weighted much higher than the accuracy component and therefore dominates the early generations during evolution so that members in later generations are those with the desired cardinality, and evolution proceeds to maximize accuracy.

4.6.1.3 Procedure of GANN

The procedure of GANN includes the following steps:

1. Population initialization – Randomly generate the initial population consisting of M members. Each member is a possible solution represented as a binary vector whose length is equal to the number of candidate features. A "1" means the feature is selected and a "0" means the feature is not selected.

2. Fitness calculation – Calculate the fitness value of each member in the current population.

3. Elitist strategy - Select the first N members with the largest fitness values to be directly placed into the next generation.

4. Crossover - Select $M - N$ members using roulette wheel selection to be placed into a mating pool. With probability proportional to fitness values, pick a pair of parents and perform crossover operation with a high probability to generate offspring.

5. Mutation - Perform mutation operation with a low probability on the offspring to obtain a new generation of population.

6. Stopping criterion check - Check the stopping criterion: if it is satisfied, stop; otherwise, go to step 2. A limit on the number of generations as the stopping criterion. Experience with the algorithm has resulted in a choice of 200 for this limit.

4.6.1.4 Neural network

The multilayer perceptrons (MLPs) with backpropagation (BP) learning algorithm is used to perform the evaluation for different feature sets. In order to limit computational time, we use a one hidden layer neural network with only one neuron in the hidden layer. After selection of the best features using a simple fixed neural network, we apply an optimization procedure to select and train the best neural network architecture, in terms of the number of hidden neurons, using a fixed feature set.

4.6.2 Application to the sample system

For the line thermal overload problem, we ran GANN for 11 different cardinality levels, from 5 to 10, with parameters 1 - 4 forced into the solution. The results are given in Table 4.3. The result of cardinality level 8 is chosen because it provides the best

Table 4.3 Feature selection results for the line thermal overload with cardinality levels 5–10 (using MLPs)

Parameter Name	Cardinality Level					
	5	6	7	8	9	10
Gen1	1	1	1	1	1	1
Load	1	1	1	1	1	1
Gen2	1	1	1	1	1	1
Gen3	1	1	1	1	1	1
Flow1	0	0	0	0	0	0
Flow2	0	0	0	0	0	0
Flow3	0	0	0	0	0	0
Flow4	0	0	0	0	0	0
Flow5	0	1	0	0	1	1
Flow6	0	0	0	0	1	0
Flow7	0	0	0	0	0	0
Flow8	0	0	0	0	0	0
Flow9	0	0	0	0	0	0
Flow10	0	0	0	0	0	0
Flow11	0	1	0	0	0	0
Flow12	0	0	0	0	0	0
Flow13	1	0	1	1	0	1
Volt1	0	0	0	0	0	1
Volt2	0	0	0	0	0	0
Volt3	0	0	0	0	0	0
Volt4	0	0	1	1	1	0
Volt5	0	0	0	0	0	0
Volt6	0	0	0	0	0	0
Volt7	0	0	0	0	0	0
Volt8	0	0	0	0	0	0
Volt9	0	0	0	0	0	1
Volt10	0	0	0	0	0	0
Volt11	0	0	0	0	1	0
Volt12	0	0	0	1	0	1
Volt13	0	0	0	0	0	0
Volt14	0	0	0	0	0	0
Flow14	0	0	1	1	1	1
Average Error	0.0315	0.0207	0.0090	0.0082	0.0084	0.0078

tradeoff between cardinality and error. The average error is given as a percentage of 600 A, which is the maximum flow on the overloaded line. The results generally agree with engineering judgment. For example, parameters Flow5, Flow13, and Flow14 are selected the most often since these parameters represent the flows on the overloaded circuit Flow13 (Flow5 and Flow13 are approximately in series) and the outaged circuit (Flow14). The other parameters selected for cardinality level 8 are bus voltage magnitudes near the overloaded or outaged circuit.

For the voltage instability problem, we ran GANN for 5 different cardinality levels, from 9 to 13, with parameters 1 - 4 forced into the solution. The results are given in Table 4.4. The results also generally agree with engineering judgment. For example, parameters 33 and 36 are always selected since they represent two synchronous condensers close to the load bus with deficient reactive power. The result of cardinality level 11 is chosen because it provides the best tradeoff between cardinality and error. Here, the average error is given as a percentage of 200 MVAr, which is the reactive power margin on the the load bus with the most deficient reactive supply.

For both problems, the critical parameters include load, generations, line flows, and bus voltage magnitudes. For the voltage instability problem, an additional type of critical parameter is available reactive power supply. There are three available reactive power supplies corresponding to three different groups of generators, $Qmax1, Qmax2$, and $Qmax3$, each of which is the sum of the reactive capacities for the committed generators in that group. These values change accordingly as the unit commitment status in each group changes.

It is necessary to point out that GANN is not guaranteed to give the best solution because of its stochastic nature and the fact that the number of evolutions can not be infinite. However, given a finite and sufficiently large number of evolutions, it has been found that GANN will always produce a good solution.

Table 4.4 Feature selection results for the voltage instability with cardinality levels 9–13 (using MLPs)

Parameter Name	Cardinality Level				
	9	10	11	12	13
Gen1	1	1	1	1	1
Load	1	1	1	1	1
Gen2	1	1	1	1	1
Gen3	1	1	1	1	1
Flow1	0	1	0	1	0
Flow2	0	0	0	1	0
Flow3	0	0	0	0	0
Flow4	0	0	0	0	0
Flow5	0	0	0	0	0
Flow6	0	0	0	0	0
Flow7	0	0	0	0	0
Flow8	0	0	0	0	0
Flow9	0	0	1	0	1
Flow10	0	0	0	0	0
Flow11	0	0	0	0	1
Flow12	0	0	0	0	0
Flow13	0	0	1	1	0
Volt1	0	1	0	0	0
Volt2	0	0	0	0	0
Volt3	0	0	0	0	0
Volt4	0	0	0	0	0
Volt5	0	0	0	0	1
Volt6	0	0	0	0	0
Volt7	1	0	1	0	1
Volt8	0	0	0	0	0
Volt9	0	0	0	0	0
Volt10	0	0	0	0	1
Volt11	0	0	0	0	0
Volt12	0	0	0	0	0
Volt13	0	0	0	0	1
Volt14	1	1	1	1	1
Qmax1	1	1	1	1	1
Qmax2	1	0	0	0	0
Qmax3	0	0	1	1	0
Qmax4	1	1	1	1	1
Qmax5	0	1	0	0	0
Flow14	0	0	0	1	0
Average Error	0.0824	0.0742	0.0665	0.0736	0.0783

4.6.3 Improvement on solution speed

4.6.3.1 GANN with radial basis function networks

There are some problems when the GA is used to do feature selection where a neural network architecture with standard backpropagation (BP) algorithm is adopted to compute fitness. One of them is that the standard BP algorithm is very slow. Since each potential solution requires neural network training, the training speed is the most significant influencing factor with respect to the overall solution time. This is a major disadvantage of using GA with the BP algorithm to do feature selection for neural networks. For instance, it took 88886 seconds of CPU time to obtain the solution for cardinality 8. To overcome this problem mentioned above, the BP algorithm in GANN is replaced by a radial basis function network with a fast training method.

If we view the neural network design problem as a curve fitting problem in a high-dimensional space, learning is equivalent to finding a surface in a multidimensional space that best fits the training data. Generalization is then equivalent to using this multi-dimensional surface to interpolate the test data. Such a viewpoint is indeed the motivation behind the method of radial basis functions (RBFs). Broomhead and Lowe [67] first explored the use of RBFs in neural networks. Moody and Darken [68], Renals and Rohwer [69], and Poggio and Girosi [70] among others made major contributions to the theory, design, and application of RBF networks.

The basic architecture of a RBF network is shown in Figure 4.6. It includes three entirely different layers. The first layer is an input layer for which each node corresponds to an attribute of an input pattern. The second layer is a hidden layer that is used to cover the whole input space with each node covering a different region. The third layer is an output layer responding to the input patterns. The transformation from the input layer to the hidden layer is nonlinear, whereas the transformation from the hidden layer to the output layer is linear. As illustrated in Figure 4.7, an activation function for a

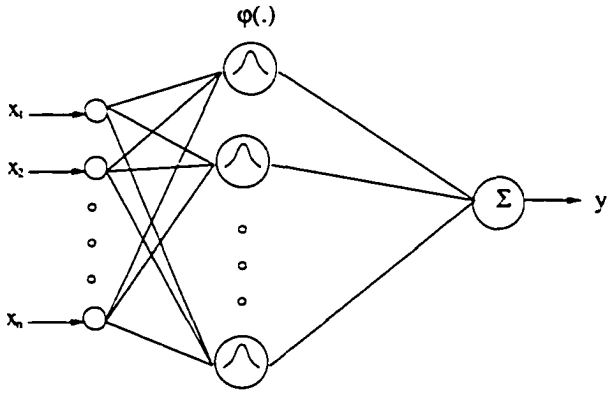


Figure 4.6 A radial basis function network

hidden layer node is a locally radially symmetric function (typically Gaussian function), where the output decays to zero as the distance (usually Euclidean distance) between the input vector and its center increases, see Figure 4.7.

There are two important parameters, center and width, associated with an activation function. A center is a vector with the same dimension as the input pattern, which represents a cluster center of the input space. A width is used to control the spread of the RBF so that its output decays more slowly or more rapidly as the distance between the input vector and the center increases. Each RBF $\varphi(\cdot)$ responds to a small convex region of the feature space. A large number of these functions cover the entire

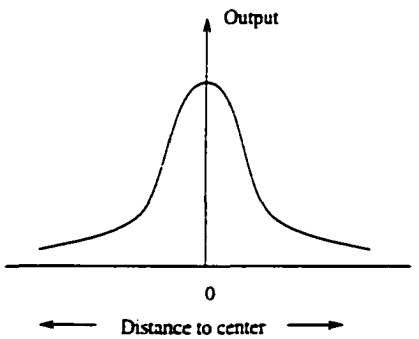


Figure 4.7 A radial basis function in one-dimensional space

feature space so that the output layer neurons can join them with different weights to accomplish the function approximation or classification. Figure 4.8 shows a portion of two-dimensional feature space covered by RBFs.

Without loss of generality, consider a RBF network with only one output. The output of such a network can be expressed as

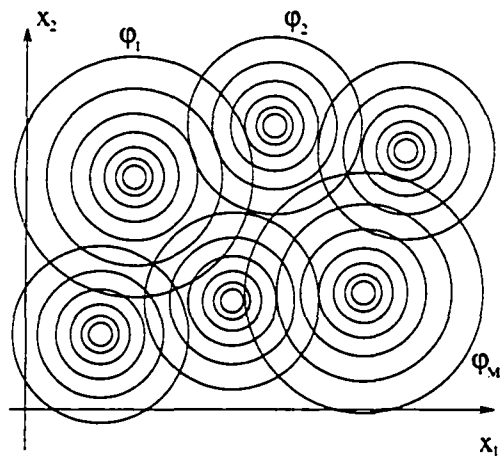


Figure 4.8 A portion of two-dimensional feature space covered by RBFs

$$f(\mathbf{x}) = \sum_{i=1}^{M} w_i \varphi_i(\|\mathbf{x} - \mathbf{c}_i\|) \tag{4.1}$$

where $\mathbf{x}$ is an input pattern, $\mathbf{c}_i$ is the center for hidden node i, w_i is the weight between hidden node i and the output node, and w_0 is a bias weight. M is the number of hidden nodes, $\varphi(\cdot)$ is the activation function for the hidden layer. The norm $\|\cdot\|$ can be Euclidian or Mahalanobis distance [75, p. 87]. A RBF approximation using Gaussian functions is considered below.

When $\varphi(\cdot)$ is a Gaussian function, (4.1) can be seen as approximating a probability density by a mixture of Gaussian functions. The Gaussian function can be expressed as

$$\varphi(u) = e^{-\frac{u}{2\sigma^2}} \tag{4.2}$$

The output can then be expressed as

$$f(\mathbf{x}) = \sum_{i=1}^{M} w_i e^{-\frac{\|\mathbf{x}-\mathbf{c}_i\|}{2\sigma^2}} \tag{4.3}$$

RBF networks provide an attractive approach for function approximation because of their flexible structure, fast training, powerful generalization capability, and conceptual elegance. Park and Sandberg [71] have shown that RBF networks with Gaussian basis functions are universal function approximators. Girosi and Poggio [72] have shown pointwise convergence property of a class of RBF networks.

There are different training strategies for RBF networks [77]. Since the linear weights associated with the output node tend to evolve more slowly than the nonlinear activation functions of the hidden nodes, it is reasonable to train the network layer by layer to get fast training speed. The training strategies can be divided into the following three classes depending on how the centers of the RBFs are specified.

Fixed centers selected randomly

The simplest and fastest approach is to randomly select centers from the training data set and keep them constant throughout the training. This is reasonable provided that the training data are well representative of the problem. The widths for all RBFs are also fixed and are the same. This width can be taken as the standard deviation of the Gaussian function, expressed as

$$\sigma = \frac{d}{\sqrt{2M}} \tag{4.4}$$

where d is the maximum distance between the selected centers. Such a choice for the standard deviation σ is to ensure that RBFs are neither too peaked to cover the whole input space nor too flat to distinguish between dissimilar input patterns.

Then, the only parameters that need to be trained are the weights between the hidden and output layer, which can be computed directly by solving linear equations. Usually the number of training patterns is much larger than the number of selected centers.

so the resulting linear equations are overdetermined. A straightforward procedure for solving such equations is to use the pseudoinverse method [67] to obtain a solution with the minimum least square error.

The linear weights can also be solved by iteration using a gradient-descent technique. This approach is much faster than the BP algorithm for MLPs because it adjusts only the weights between the hidden and output layer whereas MLPs adjust weights between all layers. The generalization capability and accuracy level depends on the number of centers and the representativeness of the training data set to the problem that is being studied.

Unsupervised selection of centers

In principle, any unsupervised or clustering algorithm can be used to specify the centers. For example, one may use k-means clustering algorithm [68], hierarchical cluster analysis, or self-organizing map. After determining the centers, one may obtain the linear weights either by directly solving the linear equations or by iteration.

Supervised selection of centers

This is the most flexible but most time-consuming training approach among the three strategies. The centers, widths, and linear weights are all adjusted through a supervised training process. A natural candidate for such a process is an error correction iteration algorithm using gradient-descent technique.

The squared error for the output can be written as

$$E = \frac{1}{2}\sum_{k=1}^{N}(d_k - y_k)^2 \tag{4.5}$$

The update equations for the linear weights, centers, and widths are given as follows [77, Chapter 7]

$$w_i(t+1) = w_i(t) + \eta_1 \sum_{k=1}^{N} (d_k - y_k) e^{-\frac{\|\mathbf{x}_k - \mathbf{c}_i\|^2}{2\sigma_i^2}} \tag{4.6}$$

$$c_{ij}(t+1) = c_{ij}(t) + \frac{\eta_2}{\sigma_i^2} \sum_{k=1}^{N} (d_k - y_k) w_i(t) e^{-\frac{\|\mathbf{x}_j - \mathbf{c}_i\|^2}{2\sigma_i^2}} \cdot (x_{kj} - c_{ij}(t)) \tag{4.7}$$

$$\sigma_i^2(t+1) = \sigma_i^2(t) + \eta_3 \sum_{k=1}^{N} (d_k - y_k) w_i z_{ik} \cdot \frac{\|\mathbf{x}_k - \mathbf{c}_i\|^2}{2\sigma_i^4(t)} \tag{4.8}$$

For the thermal overload problem, GANN was run with RBFNs using the learning algorithm of fixed centers, for 5 different cardinality levels, from 5 to 10, with parameters 1-4 forced (preselected) into the solution, meaning that all solutions provided by GANN will have GenA, AreaLoad, GenB, and GenC, in addition to some other parameters.

The results of GANN runs are given in Table 4.5 [82]. The level 8 cardinality is chosen because it represents the best tradeoff between cardinality and accuracy.

The results generally agree with engineering judgment. For example, Flow5 and Volt1 are chosen most often; Flow5 represents the flow on the overloaded circuit and Volt1 is a bus voltage close to the outaged circuit. The results show that GANN with RBFNs using the learning algorithm of fixed centers runs faster than with MLPs using the BP learning algorithm. For example, at cardinality level of 8, it took almost 10 times faster for the former than the latter (8903 seconds with RBFNs versus 88886 seconds with MLPs).

For evaluation of the feature selection effect, the test results with 8 selected attributes and with all 32 attributes are shown in Figure 4.9. It can be seen that the test error is much smaller using the selected attributes. Furthermore, training with fewer selected attributes takes less time than with all 32 attributes although the CPU time is increasing with the number of centers in both cases, as shown in Figure 4.10. It is noted that the

Table 4.5 Feature selection results for the thermal overload with cardinality levels 5–10 (using RBFNs)

Parameter Name	Cardinality Level					
	5	6	7	8	9	10
GenA	1	1	1	1	1	1
Load	1	1	1	1	1	1
GenB	1	1	1	1	1	1
GenC	1	1	1	1	1	1
Flow1	0	0	0	0	0	0
Flow2	0	0	0	0	0	0
Flow3	0	0	0	0	0	0
Flow4	0	0	0	0	0	0
Flow5	0	1	0	1	1	0
Flow6	0	0	0	0	0	0
Flow7	0	0	0	0	0	0
Flow8	0	0	0	0	0	0
Flow9	0	0	1	0	1	0
Flow10	0	0	0	0	0	0
Flow11	0	0	0	0	0	0
Flow12	0	0	0	0	0	1
Flow13	1	0	1	0	0	1
Volt1	0	1	1	1	0	1
Volt2	0	0	0	0	1	0
Volt3	0	0	0	1	0	0
Volt4	0	0	0	0	0	0
Volt5	0	0	0	0	0	0
Volt6	0	0	0	0	0	1
Volt7	0	0	0	0	1	0
Volt8	0	0	0	0	0	1
Volt9	0	0	0	0	0	0
Volt10	0	0	0	0	0	1
Volt11	0	0	0	0	0	0
Volt12	0	0	0	0	0	0
Volt13	0	0	0	0	0	0
Volt14	0	0	0	0	1	0
Flow14	0	0	0	1	0	0
Average Error	.0222	.0161	.0095	.0087	.0083	.0061

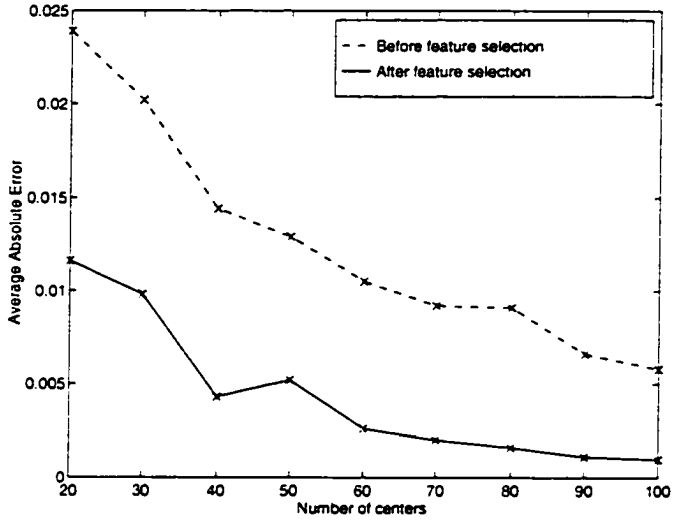


Figure 4.9 Test errors versus number of centers (random selection of centers)

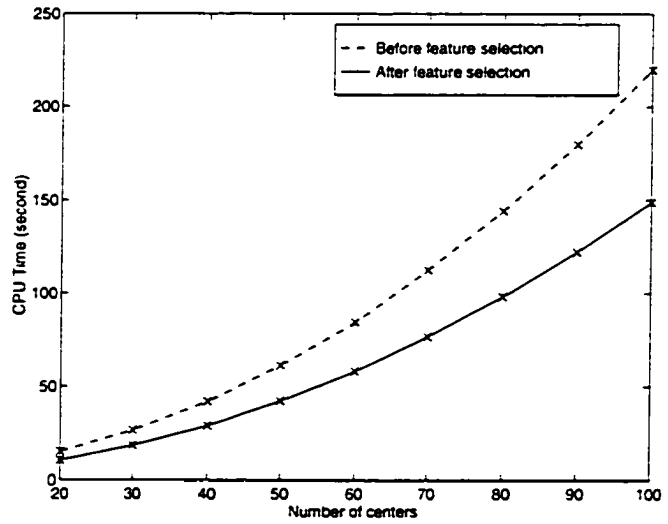


Figure 4.10 CPU time versus number of centers (random selection of centers)

accuracy level is improved with the increase of the number of centers, but this is achieved at the price of solution speed decrease. Therefore, the user must make a tradeoff between accuracy and speed by choosing the number of centers. Like choosing the hidden nember of hidden nodes for MLPs, unfortunately, there is no theoretical guidance for choosing the number of centers for RBFNs.

4.6.3.2 Genetic algorithm with k-nearest neighbor method

To further improve the solution speed, the k-nearest neighbor method instead of neural networks is used to evaluate feature sets. Table 4.6 shows the feature selection results using RBFNs and KNNs for the thermal overload problem. From the table, we can see that the solution speed with KNNs is much faster. However, the error will be much larger if we use KNNs to predict the postcontingency performance measure based on the selected features. Note that MLPs are still to be used in postcontingency performance measure prediction.

Table 4.6 Comparison of feature selection results using RBFNs and KNNs for the thermal overload problem

	Radial basis function networks	k-nearest neighbors
Features selected	1.2.3.4.9.17.18.32	1.2.3.4.9.17.19.32
Running time	8903	4709
Average error	0.0012	0.0270

4.7 Comparisons of different feature selection methods

The feature selection methods discussed above include the cluster analysis, the Kohonen self-organizing map, and the genetic algorithm based methods. The genetic algorithm based methods include those with MLPs, RBFNs, and KNNs. Two comparisons will be made as follows.

4.7.1 Solution speed

According to the decreasing order of solution speed, the first is the cluster analysis followed by the Kohonen self-organizing map, and the genetic algorithm based methods are the last. Of the variations of the genetic algorithm based methods, the method with KNNs is the fastest, that with RBFNs is the second fasted, and that with MLPs is the slowest. Table 4.7 lists the solution time for the genetic algorithm based methods with MLPs, RBFNs, and KNNs with cardinality level 8 for the thermal overload problem.

Table 4.7 Comparison of solution time (s)

MLPs	RBFNs	KNNs
88886	8903	4709

4.7.2 Prediction accuracy

The MLPs are used to predict the postcontingency performance measure using selected features because they are the most accurate. To compare the accuracy levels for features selected by different methods, a MLP with one hidden neuron is trained and tested using these features. Table 4.8 lists the selected features and the average errors for the genetic algorithm based methods with MLPs, RBFNs, and KNNs with cardinality level 8 for the thermal overload problem. From this table, we can find the following facts. First, the accuracy levels for cluster analysis and Kohonen self-organizing map vary depending on what features are selected from the clusters or nodes. Second, the accuracy level for GANN with MLPs is highest. This is reasonable because the features selected by GANN with MLPs were used to train a MLP itself. The accuracy level for GANN with RBFNs is lower than that for GANN with MLPs but higher than that for GANN with KNNs. For more thorough comparison, it is better to consider the variances as well.

Table 4.8 Comparison of prediction accuracy

Method	Features Selected	Average Error
GANN-MLP	1,2,3,4,17,21,29,32	0.0029
GANN-RBFN	1,2,3,4,9,17,18,32	0.0034
GANN-KNN	1,2,3,4,9,17,19,32	0.0036
Cluster Analysis	1,2,3,4,5,9,10,18	0.0051
	1,2,3,4,11,12,17,18	0.0045
	1,2,3,4,9,12,18,32	0.0035
	1,2,3,4,9,12,15,18	0.0032
	1,2,3,4,10,13,16,17	0.0045
Kohonen Map	1,2,3,4,5,9,10,18	0.0051
	1,2,3,4,5,6,9,18	0.0041
	1,2,3,4,7,17,18,32	0.0032
	1,2,3,4,7,17,20,32	0.0037
	1,2,3,4,12,17,18,32	0.0032

4.8 Summary

Three classes of feature selection methods have been discussed: cluster analysis, Kohonen self-organizing map, and genetic algorithm based methods. Cluster analysis and Kohonen self-organizing map divide the features into various clusters according to their similarities. They are fast. Then, for feature selection, only one feature needs to be picked up from each cluster to represent this cluster. Therefore, there are different feature selection results depending on which feature is selected from each cluster. Some results may give higher accuracy, some results may give lower accuracy. There is no guarantee that a good accuracy is achieved.

In consideration of this fact, genetic algorithm based methods are preferred. Among GANN with MLPs, GANN with RBFNs, and GANN with KNNs, GANN with KNNs has fastest solution speed but lowest accuracy, and GANN with MLPs highest accuracy but slowest solution speed, which should be used? By making a trade-off between solution speed and accuracy, GANN with RBFNs is recommended because it is much faster than GANN with MLPs without much loss of accuracy.

5 TRAINING AND USING NEURAL NETWORKS FOR BOUNDARY APPROXIMATION

5.1 Introduction

After generating data and selecting features, the next step is to train neural networks using the data and features to predict postcontingency performance given precontingency information, for a single contingency. Issues in training and using neural networks are discussed in this chapter. Overfitting, underfitting, and accuracy improvement near the boundary during training are addressed first to achieve the expected accuracy. Then, confidence interval calculation formulas for neural network outputs are derived and implemented for MLPs. Finally, sensitivities of neural network outputs with respect to inputs are derived and implemented for MLPs.

5.2 Training neural networks

5.2.1 Overfitting and underfitting

Overfitting and underfitting are very important problems for neural network training. If the number of degrees of freedom of the neural network is too high (low) with respect to the training set, the neural network is very likely overfitted (underfitted). If the neural network is underfitted, it is difficult to reach an acceptable convergent training error. If the neural network is overfitted, the training error may be low but the test error can be much higher. A neural network is said to memorize rather than learn the training data

if it responds to a sample in the training data set in the same manner as it was trained, but it responds to other samples with very poor results. In the case of overfitting, a neural network just memorizes the training data, hence the generalization capability is very poor. To alleviate this problem, the following approaches may be used:

Underfitting detection by a trial and error approach

After the training process has converged, if the training error is higher than the value one would expect, it means that an underfitting occurs. Therefore, the network complexity, i.e., the number of degrees of freedom of the neural network, should be increased and a new network trained. In the case of MLPs with one hidden layer, the number of hidden neurons should be increased. If an underfitting still occurs after training the new network, another trial is needed with respect to a more complex network structure.

Overfitting detection by monitoring and comparing training error and test error

Initially, both training error and test error decrease with training iterations, then flatten out until some point where the training error still decreases but the test error begins to increase. This point is where overfitting occurs. Figure 5.1 illustrates this situation. However, it is not always apparent to see this point because some oscillations of test error may occur.

5.2.2 Exploration of improvement on accuracy near the boundary

Because the ultimate goal of this research is to produce boundaries, which are contours of constant system performance, the accuracy of the function approximation

$$R = f(\mathbf{x})$$

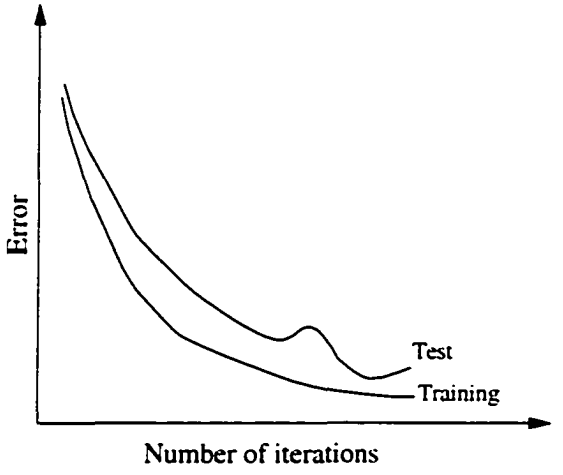


Figure 5.1 Variation of training and test errors with the number of iterations.

at the boundary $R = R_0$ is a major concern. However, because no other values of R are ultimately revealed to the operator, it is much less concerned with the accuracy of

$$R = f(\mathbf{x})$$

at points not on the boundary. Therefore, the accuracy off the boundary can be sacrificed if the accuracy on the boundary can be simultaneously improved. One method has been identified to do this. This method is called weighted output errors.

During training, let the output error near the boundary have larger weight than that far away from the boundary in the total error of all the outputs. The following function

$$w = 1 - d^{\frac{1}{x}} \tag{5.1}$$

is used to obtain a weight w for the desired output d where $x \geq 1$ (the smaller x, the larger the weights for points close to the boundary) and $d = 0$ occurs for a point on the boundary. Figure 5.2 shows a plot of function (5.1) for $x = 8$. It can be seen that the farther the desired output is from the boundary, the smaller the weight is. Figure 5.3 shows the distribution of 411 test samples. Figure 5.4 shows the test error distribution for test samples using the weighted errors training and that using regular training. However,

the result with weighted errors is not better than that without weighted errors for points close to the boundary as it was expected. The reason may be that the weight update approach is still the gradient descent technique, which is used for updating weights. To find the exact explanation, it requires further investigation, which is beyond the scope of this dissertation.

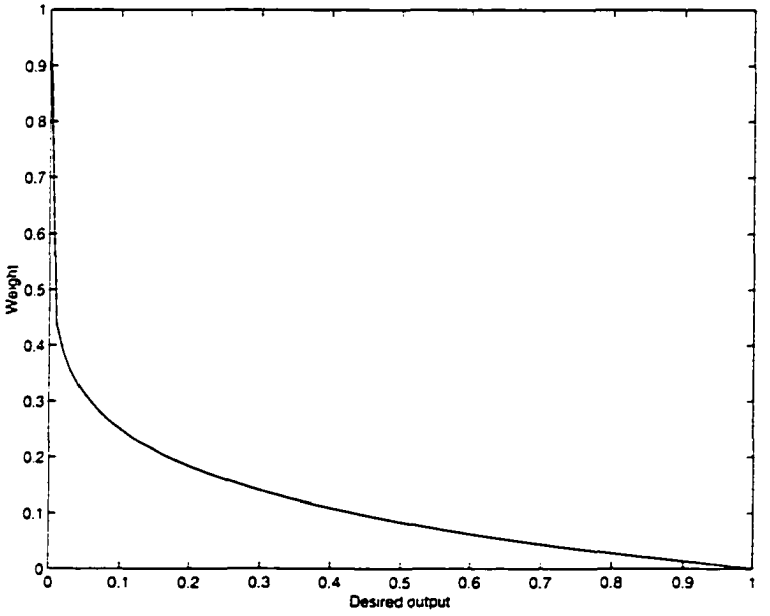


Figure 5.2 Plot of function $w = 1 - d^{\frac{1}{5}}$.

5.3 Confidence intervals for neural network outputs

When a trained neural network is used for prediction, it is desirable to evaluate the reliability of the performance. One way to do this is to give a confidence measure for the neural network output.

From the view of statistics, function approximation by neural networks can be thought of as a nonparametric nonlinear regression problem. Therefore, confidence interval estimation methods for nonparametric nonlinear regression can also be used for evaluating confidence intervals for neural network outputs. In what follows, the con-

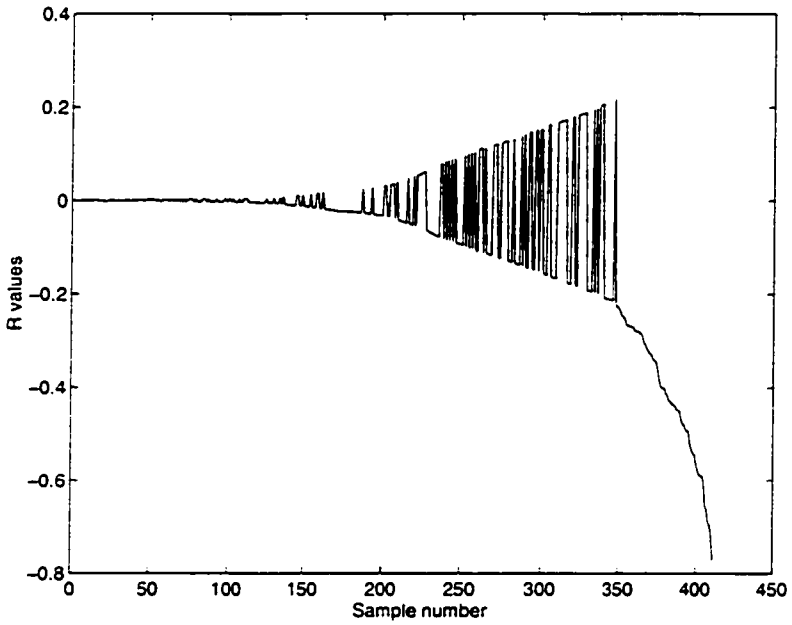


Figure 5.3 R values for the test samples.

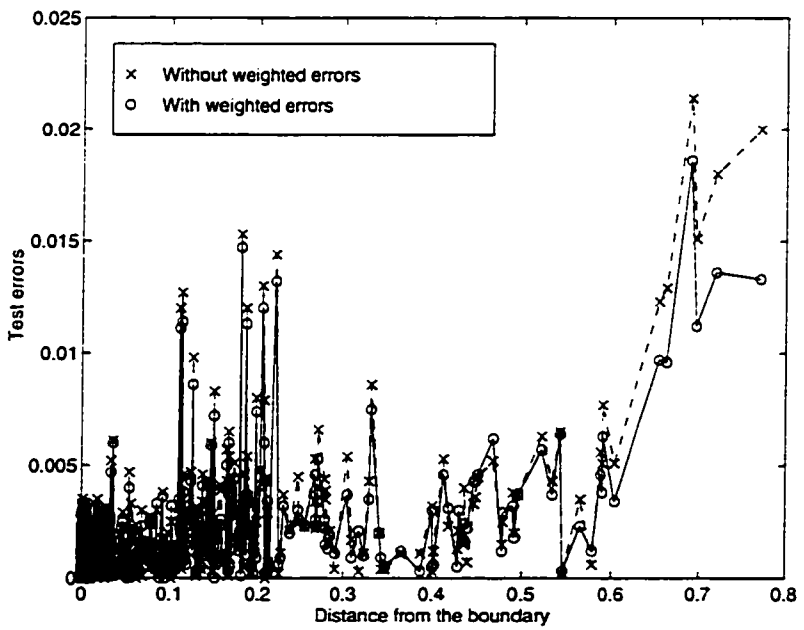


Figure 5.4 Test error distributions.

fidence interval expressions for MLPs will be derived based on a general formula for nonparametric nonlinear regression.

Consider a general nonlinear regression model [76] with p parameters to be estimated and n observations

$$y_i = f(\mathbf{x}_i, \theta) + \varepsilon_i, \ i = 1, \cdots, n \tag{5.2}$$

where y is the actual output, $\mathbf{x}$ is the input vector, θ is a vector of the parameters to be estimated, and ε_i is the error associated with the function f. We assume that ε_i's are normal and independent with zero mean and common variance σ^2. Let $\hat{\theta}$ be the least squares estimator of θ, then

$$S(\theta) = \sum_{i=1}^{n} [y_i - f(\mathbf{x}_i, \hat{\theta})]^2 \tag{5.3}$$

is minimized. The nonlinear function in (5.2) can be expanded in a Taylor series around the estimated vector $\hat{\theta}$ retaining only the first order terms. Thus

$$f(\mathbf{x}_i, \theta) \approx f(\mathbf{x}_i, \hat{\theta}) + \mathbf{g}_0^T(\theta - \hat{\theta}), \ i = 1, \cdots, n \tag{5.4}$$

where

$$\mathbf{g}_0^T = \left[\frac{\partial f(\mathbf{x}_i, \theta)}{\partial \theta_1}, \frac{\partial f(\mathbf{x}_i, \theta)}{\partial \theta_2}, \cdots, \frac{\partial f(\mathbf{x}_i, \theta)}{\partial \theta_p} \right]_{\theta = \hat{\theta}} \tag{5.5}$$

Equation (5.4) is a linear approximation of the nonlinear function in (5.2) in the neighborhood of $\hat{\theta}$, and it can be written as

$$y_i - f(\mathbf{x}_i, \hat{\theta}) = \mathbf{g}_0^T(\theta - \hat{\theta}) + \varepsilon_i, \ i = 1, \cdots, n \tag{5.6}$$

Because of the statistical independence between $\hat{\theta}$ and ε_i's, given $\mathbf{x}_0$, the variance of the output y_0 can be expressed as

$$\text{var}(y_0 - \hat{y}_0) = \text{var}(\varepsilon_0) + \text{var}(\mathbf{g}_0^T(\theta - \hat{\theta})) \tag{5.7}$$

For an error ε_0 with a normal distribution with a mean of zero and a variance of $\sigma^2(\mathcal{N}(0, \sigma^2\mathbf{I}_n))$, the distribution of $\theta - \hat{\theta}$ can be approximated to have the distribution $\mathcal{N}_p(0, \sigma^2[\mathbf{G}^T\mathbf{G}]^{-1})$, where $\mathbf{G}$ is the Jacobian matrix defined as

$$\mathbf{G} = \begin{bmatrix} \frac{\partial f(\mathbf{x}_1,\theta)}{\partial\theta_1} & \frac{\partial f(\mathbf{x}_1,\theta)}{\partial\theta_2} & \cdots & \frac{\partial f(\mathbf{x}_1,\theta)}{\partial\theta_p} \\ \frac{\partial f(\mathbf{x}_2,\theta)}{\partial\theta_1} & \frac{\partial f(\mathbf{x}_2,\theta)}{\partial\theta_2} & \cdots & \frac{\partial f(\mathbf{x}_2,\theta)}{\partial\theta_p} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f(\mathbf{x}_n,\theta)}{\partial\theta_1} & \frac{\partial f(\mathbf{x}_n,\theta)}{\partial\theta_2} & \cdots & \frac{\partial f(\mathbf{x}_n,\theta)}{\partial\theta_p} \end{bmatrix}_{\theta=\hat{\theta}} \tag{5.8}$$

Thus, we can obtain

$$\text{var}[y_0 - \hat{y}_0] = \sigma^2 + \sigma^2\mathbf{g}_0^T(\mathbf{g}^T\mathbf{g})^{-1}\mathbf{g}_0 \tag{5.9}$$

σ^2 can be approximated by

$$s^2 = \sum_{i=1}^{n} \frac{[y_i - f(\mathbf{x}_i, \hat{\theta})]^2}{n - p} \tag{5.10}$$

As a result, the standard error of a predicted value $f(\mathbf{x}_0, \hat{\theta})$ is given by

$$s_{f(\mathbf{x}_0,\hat{\theta})} = s\sqrt{\mathbf{g}_0^T(\mathbf{G}^T\mathbf{G})^{-1}\mathbf{g}_0} \tag{5.11}$$

The Student t-distribution is

$$t_{n-p} \sim \frac{y_0 - \hat{y}_0}{\sqrt{\text{var}[y_0 - \hat{y}_0]}}$$

$$\approx \frac{y_0 - \hat{y}_0}{s\sqrt{1 + \mathbf{g}_0^T(\mathbf{G}^T\mathbf{G})^{-1}\mathbf{g}_0}} \tag{5.12}$$

(5.13)

The approximate $100(1-\alpha)\%$ confidence interval on the prediction for a new observation $\mathbf{x}_0$ is then given by

$$f(\mathbf{x}_0, \hat{\theta}) \pm t_{\alpha/2, n-p} s \sqrt{1 + \mathbf{g}_0^T (\mathbf{G}^T \mathbf{G})^{-1} \mathbf{g}_0} \tag{5.14}$$

This is a general expression of the confidence interval for any regression problem. The plus and minus signs in (5.14) are used to give the upper bound and lower bound of the confidence interval. For all of the observations for which the confidence intervals are computed, the envelopes can be obtained by connecting all of the lower bound points and all of the upper bound points, respectively.

For the case of neural networks, weights correspond to θ and input patterns correspond to $\mathbf{x}_0$. Therefore, we can easily derive the confidence interval expressions for different types of neural networks using (5.14). The key is the calculation of derivatives of the output with respect to weights [81], which are used in formulation of $\mathbf{G}$.

For MLPs with one hidden layer as shown in Figure 5.5, the derivatives of output y_j with respect to weight w_{ji} between the hidden and output layer can be derived as

$$\begin{aligned} \frac{\partial y_j}{\partial w_{ji}} &= \frac{\partial y_j}{\partial v_j} \cdot \frac{\partial v_j}{\partial w_{ji}} \\ &= \varphi_j'(v_j) z_i \end{aligned} \tag{5.15}$$

The derivatives of output y_j with respect to weight w_{ik} between the input and hidden

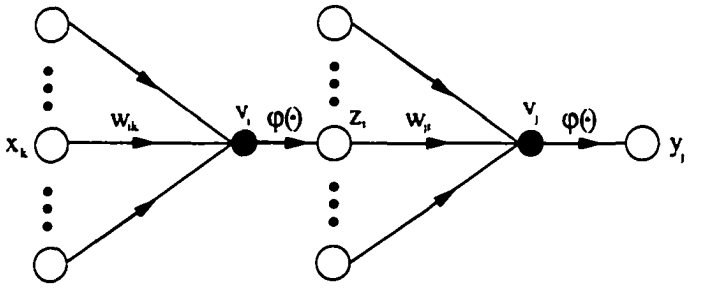


Figure 5.5 A detailed structure of a multilayer perceptron

layer can be derived as

$$\begin{aligned}\frac{\partial y_j}{\partial w_{ik}} &= \frac{\partial y_j}{\partial v_j}\cdot\frac{\partial v_j}{\partial z_i}\cdot\frac{\partial z_i}{\partial v_i}\cdot\frac{\partial v_i}{\partial w_{ik}} \\ &= \varphi_j'(v_j)w_{ji}\varphi_i'(v_i)x_k \end{aligned} \tag{5.16}$$

where z_i is the output of hidden node i. v_j is the dot product of the outputs from the hidden nodes and weights between the hidden nodes and output node j. $\varphi(\cdot)$ is the activation function of a node.

When a hyperbolic tangent function

$$\begin{aligned}\varphi(x) &= a\tanh(bx) \\ &= \frac{e^{bx}-e^{-bx}}{e^{bx}+e^{-bx}} \\ &= \frac{2a}{1+e^{-2bx}} - a \end{aligned} \tag{5.17}$$

is used as an activation function, its derivative is

$$\varphi'(x) = ab(1-\varphi^2(x)) \tag{5.18}$$

where a and b are constants specifying the range and the slope of the function, respectively. Using this activation function for all of the nodes, (5.15) and (5.16) become

$$\frac{\partial y_j}{\partial w_{ji}} = ab(1-\varphi^2(v_j))z_i \tag{5.19}$$

and

$$\frac{\partial y_j}{\partial w_{ik}} = a^2b^2(1-\varphi^2(v_j))w_{ji}(1-\varphi^2(v_i))x_k \tag{5.20}$$

respectively.

The confidence interval calculation method has been applied to the thermal overload problem for a demonstration. Figure 5.6 shows the 90% confidence intervals for a portion of test data for the thermal overload problem. It can be seen that the desired output falls between the envelopes of the confidence interval with probability about 90%.

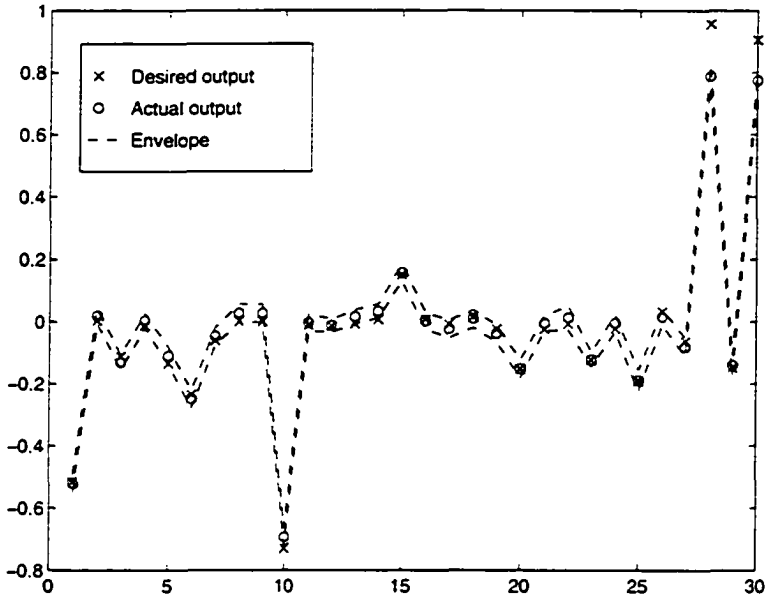


Figure 5.6 90% confidence interval for the test data of the thermal overload problem.

To apply the confidence interval calculation method to the boundary, simply calculate the confidence interval and obtain the lower and upper bounds for each boundary point, then connect the lower bound points and the upper bound points, respectively.

5.4 Sensitivity analysis of the neural network output with respect to input parameters

Analyzing the sensitivity of the neural network output with respect to input parameters will help operators understand the relative importance of these parameters in terms of their contributions to available security margin.

For MLPs with one hidden layer of m nodes (see Figure 5.5), the sensitivity of output j with respect to input k can be derived as follows

$$\frac{\partial y_j}{\partial x_k} = \frac{\partial y_j}{\partial v_j} \sum_{i=1}^{m} \frac{\partial v_j}{\partial z_i} \cdot \frac{\partial z_i}{\partial v_i} \cdot \frac{\partial v_i}{\partial x_k}$$

$$= \varphi_j'(v_j) \sum_{i=1}^{m} w_{ji} \varphi_i'(v_i) w_{ik} \tag{5.21}$$

When a hyperbolic tangent function (5.17) is used for activation functions for all of the nodes, (5.21) becomes

$$\frac{\partial y_j}{\partial x_k} = a^2 b^2 (1 - y_j^2) \sum_{i=1}^{m} w_{ji} (1 - z_i^2) w_{ik} \tag{5.22}$$

Figure 5.7 shows the sensitivity of the MLP output with respect to the 8 input parameters for the thermal overload problem. Figure 5.8 shows the sensitivity of the MLP output with respect to the 11 input parameters for the voltage instability problem. These sensitivity plots are with respect to a specific operating condition. For the thermal overload problem, the most sensitive parameter is input parameter 5, which is Flow13. Because the corresponding sensitivity is positive, increasing Flow13 will make the system more insecure. For the voltage instability problem, the most sensitive parameters are input parameters 7 and 8, which are the two bus voltage magnitudes, Volt7 and Volt14. Because the corresponding sensitivities are negative, an operating condition change that

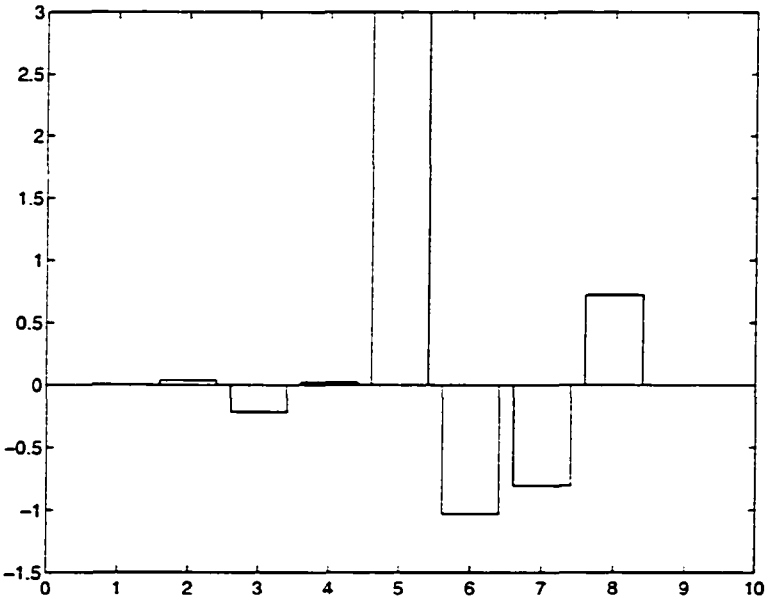


Figure 5.7 Sensitivity of the MLP output with respect to the 8 input parameters for the thermal overload problem.

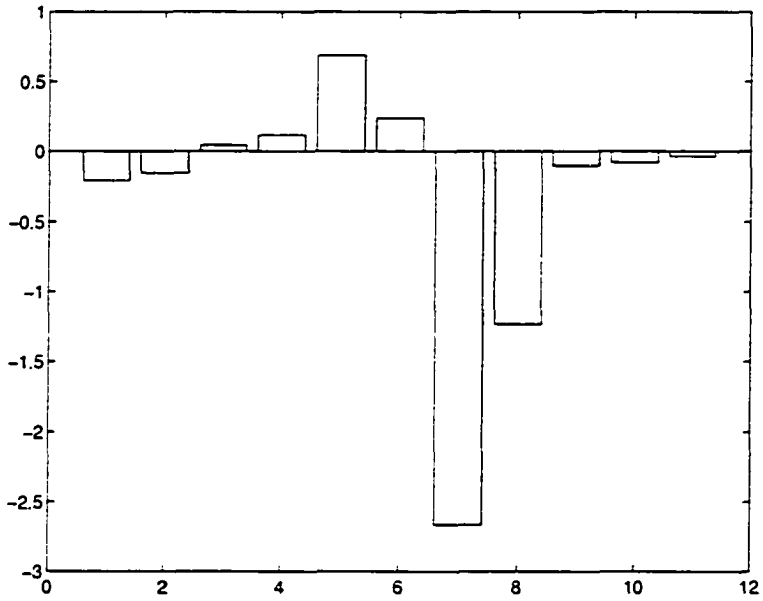


Figure 5.8 Sensitivity of the MLP output with respect to the 11 input parameters for the voltage instability problem.

results in an increase in the two voltage magnitudes can lead the system to more secure conditions. Therefore, this fact can be a guideline for the operator to adjust the most influential parameters to make the system more secure.

5.5 Summary

In this chapter, two issues related to neural network training have been discussed, then the confidence interval calculation has been derived and illustrated, finally the sensitivities of the neural network output with respect to the inputs has been derived and analyzed.

One issue in neural network training is to make sure that neither overtraining nor undertraining occurs. Overtraining can be detected by monitoring and comparing the training error and the test error. Undertraining can be detected by a trial and error approach. Another issue in neural network training especially for the boundary visualization problems is to make points closer to the boundary more accurate. A weighted

output error approach has been explored but is not effective.

Confidence intervals can be a measure of the neural network output reliability. The calculation formula have been derived for the MLPs based on nonparametric nonlinear regression. An example has been given to illustrate how to calculate confidence interval conculation for the thermal overload problem.

Sensitivities of the neural network output with respect to the inputs can provide a control guideline for the operator to effectively adjust the system to more secure conditions. The sensitivity calculation has been derived and demonstrated for the thermal overload and voltage instability problems.

6 PRESENTATION TECHNIQUES

6.1 Introduction

The ultimate goal of this research is to enable highly accurate nomogram presentation to the operator. This presentation should be fast and automatic so that it can interface with other software and communication equipment in the control center that provides values of the critical parameters corresponding to the current operating conditions.

This chapter addresses how to present the boundary to the operator, i.e., boundary visualization. First, the problem of boundary visualization is formulated in Section 6.2. The individual boundary visualization approach is described in Section 6.3. An algorithm to visualize composite boundaries is developed in Section 6.4. An approach to available transfer capability calculation is proposed in Section 6.5. Obtaining the update functions is described in Section 6.6.

6.2 Problem formulation

After the neural network is trained, the mapping between the critical parameters and the postcontingency performance measure is formed. This mapping function is used to draw a security boundary, i.e., all points on the boundary must satisfy this function. The other constraint is that all points on the boundary must satisfy the power flow equations. Therfore, in drawing a security boundary, the following constraints must be

satisfied simultaneously

$$f(\mathbf{x}) - R_0 = 0 \tag{6.1}$$

$$\mathbf{h}(\mathbf{u}) = 0 \tag{6.2}$$

where (6.1) represents the mapping function, (6.2) represents the power flow equations, $\mathbf{x}$ is the critical parameter vector, and $\mathbf{u}$ is the input parameter vector to the power flow program. If $\mathbf{x}$ does not include any dependent parameters, then $\mathbf{x} \subset \mathbf{u}$.

For convenience, the two critical parameters that characterize the space for which the boundary is drawn are referred to as the *presented* critical parameters and others are called *covered* critical parameters. In drawing a boundary in the space of the two presented parameters (independent critical parameters), (6.1) will be used to search the values of one presented critical parameter on the boundary points for given values of the other presented critical parameter. However, the problem is how to ensure that the boundary points satisfy the power flow equations. In fact, when one solves (6.1) in drawing the boundary, one will always obtain the boundary points satisfying (6.2) if each critical parameter vector $\mathbf{x}$ that is provided to the mapping function during the solution procedure satisfies the power flow equations. This is because the mapping function is obtained by training neural networks using data from simulations that satisfy the power flow equations. Thus, we need to make sure that the critical parameter vector provided to the mapping function satisfies the power flow equations whenever we solve (6.1). There are two different solutions to this problem, depending on whether any of the critical parameters are dependent.

If there are no dependent critical parameters, i.e., all critical parameters are independent critical parameters that would be part of input parameters to the power flow program, then the boundary is drawn in the space of the presented parameters with all covered parameters fixed. In this case, the power flow equations are automatically satisfied because variations of the two presented critical parameters do not influence the

covered parameters. If a covered parameter value changes, the boundary needs to be redrawn to account for this change.

If one or more critical parameters are dependent, their values can not be fixed in drawing the boundary. This is because their values are dependent on the independent critical parameters. Variations of the two presented critical parameters in drawing the boundary will influence the dependent parameters according to the power flow equation constraints. Therefore, this influence must be considered in order to satisfy the power flow equations. This can be done by introducing a full power flow solution into the visualization software; however, the increase in the computational cost would outweigh the improvement in accuracy. Instead, this dependency can be modeled using appropriate update functions as described in Section 6.6.

The following is the description of the algorithm developed to draw the boundary.

6.3 Individual boundary visualization

6.3.1 General description

After the neural network is trained, the obtained mapping function (6.1) is used to visualize the boundary. Let $\mathbf{x} = [\mathbf{z}^T, \mathbf{y}^T]^T$, $\mathbf{z} = [z_1, \cdots, z_n]^T$, $\mathbf{y} = [y_1, \cdots, y_l]^T$, then the mapping function can identify a two-dimensional boundary with axes of z_1 and z_2 where $z_{1,min} \leq z_1 \leq z_{1,max}$, $z_{2,min} \leq z_2 \leq z_{2,max}$ and $z_i (i = 3, \cdots, n)$ is constant. The algorithm starts from the current operating point $\mathbf{x}^{(0)} = (\mathbf{z}^{(0)}, \mathbf{y}^{(0)})$. If dependent parameters are used in characterizing the boundary, then variations in z_1 and z_2 that are made when drawing the boundary must be used to update $\mathbf{y}$. It is assumed that simple, approximate expressions may be developed to perform this update (see Section 6.6). These expressions are denoted as $\mathbf{g}_y(\mathbf{y}^{(0)}, \Delta z_1, \Delta z_2)$, where Δz_i is the change in z_i from the value $z_i^{(0)}$ from which the algorithm was initialized, i.e.

$$\Delta z_i = z_i^{(k)} - z_i^{(0)}$$

so that

$$\mathbf{y}^{(k)} = \mathbf{g}_y(\mathbf{y}^{(0)}, \Delta z_1, \Delta z_2) \tag{6.3}$$

Substitution of equation (6.3) into equation (6.1) allows the neural network mapping function to be expressed as a function of $\mathbf{z}$ only, i.e.,

$$f(\mathbf{z}^{(k)}, \mathbf{g}_y(\mathbf{y}^{(0)}, \Delta z_1, \Delta z_2)) - R_0 = 0 \tag{6.4}$$

A search algorithm has been developed to obtain the boundary from the above equation. The basic procedure is to repeatedly solve for z_2 as the given value of z_1 increases from the minimum to the maximum.

6.3.2 Application to the thermal overload problem

6.3.2.1 Boundary plotting

Eight initial points, labeled as 1:0, 2:0, etc. in Table 6.1, were used to initialize the boundary plots. These initial points represent eight different operating conditions and characterize the power system at, for example, different times of the day. A power flow case was constructed for each of these operating conditions, and from each of these cases, the values for the 4 dependent parameters were obtained. These values, together with the values of the 4 independent parameters, constitute the initial conditions required to draw the boundary corresponding to the thermal overload problem. These boundaries were drawn using the visualization software and are illustrated in Figures 6.1 – 6.4. One might note that Figure 6.3 indicates that the operating conditions corresponding to points 5 and 6 are identical in the presented parameters; however they are different in terms of the covered parameters as indicated in Table 6.1. The fact that the boundaries for the two operating conditions are slightly different reflects the ability of the procedure to respond to changes in covered critical parameters.

Table 6.1 Results of thermal overload boundary accuracy assessment

Point	Subarea load	Generation A		Generation B		Generation C		Qmax4	Qmax5		Malin	Vincent	R
		gen level	react cpblty	gen level	react cpblty	gen level	react cpblty	react cpblty	gen level	react cpblty			
1:0	2430	800	585	0	0	155	78	86.6	0	51.3	3221	-1500	.05222
1a	2190	512	585	0	0	155	78	86.6	0	51.3	3203	-1500	-.01557
1b	2460	978	585	0	0	155	78	86.6	0	51.3	3212	-1500	.01680
2:0	2540	800	585	0	175.3	155	78	86.6	0	51.3	3348	-1500	.11872
2a	2190	493	585	0	175.3	155	78	86.6	0	51.3	3354	-1500	-.01236
2b	2460	960	585	0	175.3	155	78	86.6	0	51.3	3370	-1500	.00718
3:0	2815	800	585	200	175.3	155	78	86.6	0	51.3	3563	-1500	.11758
3a	2460	464	585	200	175.3	155	78	86.6	0	51.3	3527	-1500	-.00845
3b	2685	853	585	200	175.3	155	78	86.6	0	51.3	3591	-1500	.00527
4:0	2900	800	585	400	175.3	155	78	86.6	0	51.3	3314	-1500	-.01255
4a	2550	169	585	400	175.3	155	78	86.6	0	51.3	3300	-1500	-.01115
4b	2820	636	585	400	175.3	155	78	86.6	0	51.3	3314	-1500	-.00963
5:0	2430	800	585	0	0	155	78	86.6	0	51.3	3221	-1100	.09767
5a	2190	627	585	0	0	155	78	86.6	0	51.3	3204	-1100	-.00817
5b	2460	1093	585	0	0	155	78	86.6	0	51.3	3221	-1100	.02832
6:0	2430	800	390	0	0	155	78	0	0	0	2793	-1100	.13992
6a	2190	669	390	0	0	155	78	0	0	0	2830	-1100	-.00352
6b	2370	979	390	0	0	155	78	0	0	0	2789	-1100	.04361
7:0	2430	800	585	0	0	0	0	86.6	0	51.3	3221	-1500	.21988
7a	2190	890	585	0	0	0	0	86.6	0	51.3	3203	-1500	.00401
7b	2325	1124	585	0	0	0	0	86.6	0	51.3	3212	-1500	.04194
8:0	2900	800	585	400	175.3	0	0	86.6	0	51.3	3314	-1500	.14173
8a	2460	388	585	400	175.3	0	0	86.6	0	51.3	3300	-1500	-.02435
8b	2685	777	585	400	175.3	0	0	86.6	0	51.3	3314	-1500	-.01101

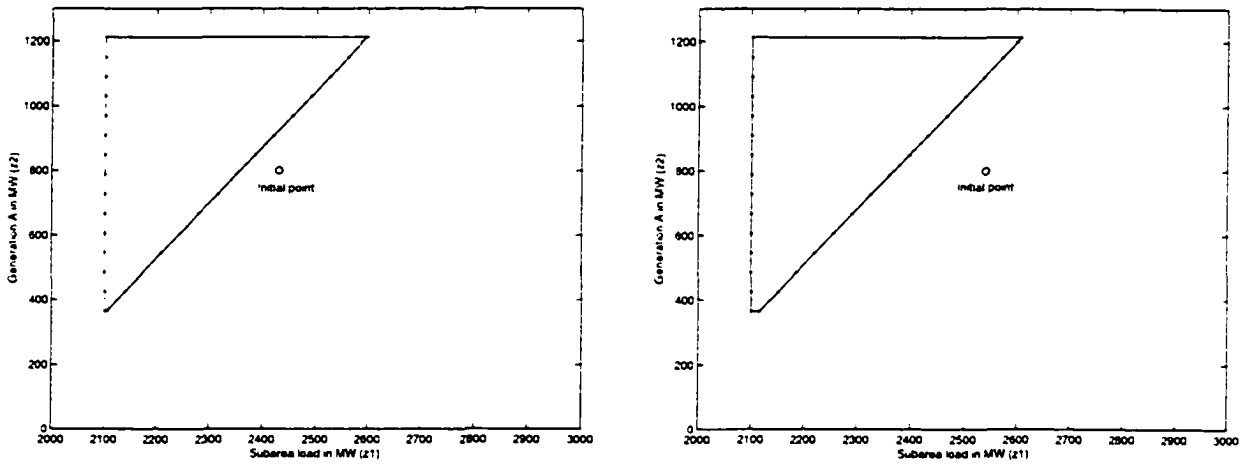


Figure 6.1 Thermal overload boundary for initial points 1:0 and 2:0

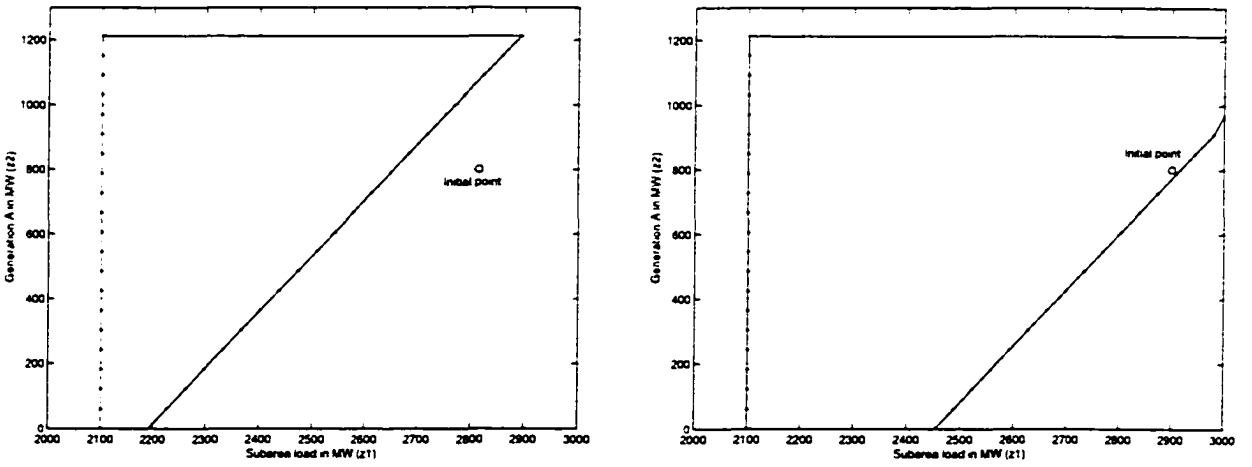


Figure 6.2 Thermal overload boundary for initial points 3:0 and 4:0

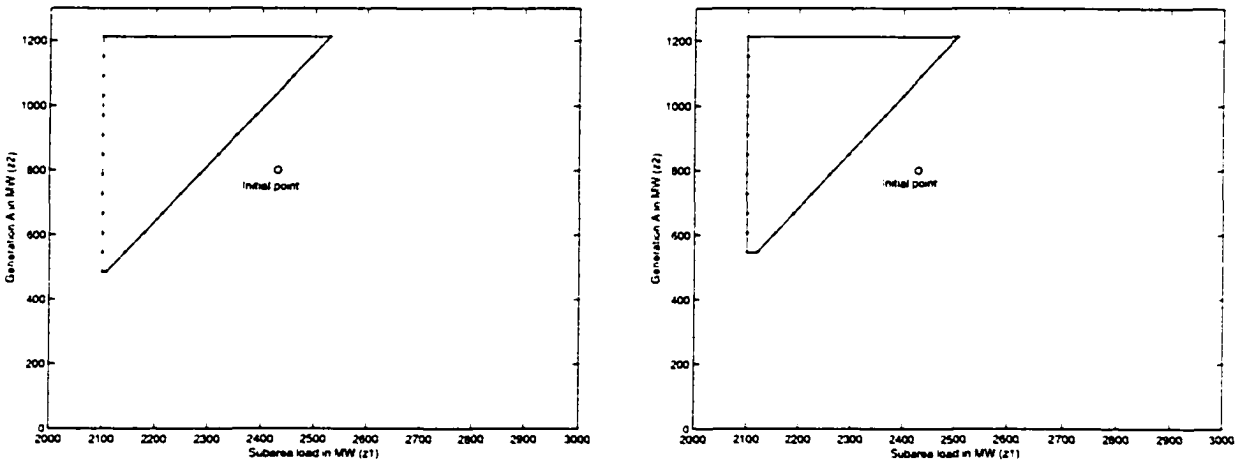


Figure 6.3 Thermal overload boundary for initial points 5:0 and 6:0

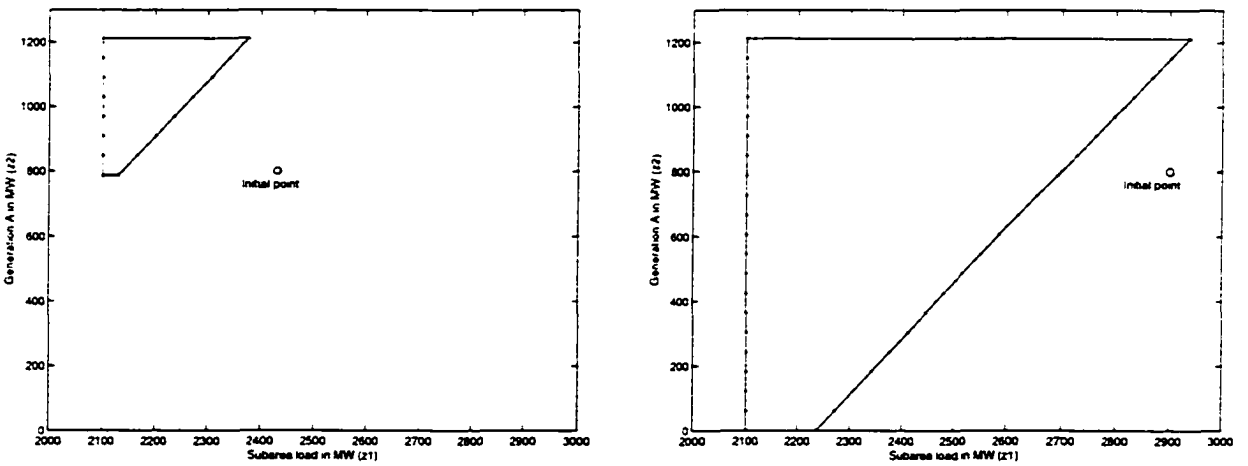


Figure 6.4 Thermal overload boundary for initial points 7:0 and 8:0

6.3.2.2 Accuracy assessment

For each boundary, two boundary points, labeled as 1a, 1b, etc. in Table 6.1, were identified to test for accuracy. The operating conditions corresponding to these points are designated using the same notation as in Table 6.1. For each boundary point, it should be the case that the current flow on the overloaded circuit is I_0=600 A. The last column of Table 6.1 provides the performance measure $R = \frac{I-I_0}{600}$. All points on the boundary (e.g., 1a,1b, and 2a,2b, etc.) should have $R = 0$ if the boundary is correct. Therefore, nonzero R for boundary points provide a measure of the difference between the actual flow, as determined by power flow simulation, and the flow predicted by the boundary, normalized by the threshold voltage of 600 A. This is an appropriate indicator of error, given as a percentage of the threshold flow.

The table indicates that maximum error on the boundary is 0.04361 (4.4%), which occurs for case 6. The errors for boundaries are expected since these boundaries incorporate two approximations: one associated with the update function and the other associated with the neural network prediction function.

6.3.3 Application to the voltage instability problem

6.3.3.1 Boundary plotting

ASAS was used to randomly generate eight cases for use in boundary accuracy assessment. These cases, labeled 9:0-16:0, are summarized in Table 6.2. The boundaries were drawn using the visualization software and are illustrated in Figures 6.5-6.8.

Table 6.2 Test results for ASAS generated operating conditions

Point	Gen A MW	Subarea Load MW	Gen B MW	Gen C MW	Flow9 MW	Folw13 MW	Volt7 kV	Volt14 kV	Qmax1 MVAr	Qmax3 MVAr	Qmax4 MVAr	R (Neural) MVAr	R (Actual) MVAr	Error MVAr
9:0	242.4	2460	155	260.8	240.6	132.6	.9673	1.0005	585	144	86.6	397	330	67
9a	242.4	2828	155	260.8	291.8	163.0	.9382	.9586	585	144	86.6	200	129	71
9a-c	242.4	2828	155	260.8	295.6	165.5	.9310	.9445	585	144	86.6	178	129	49
9b	909	2819	155	260.8	387.3	134.3	.9514	.9727	585	144	86.6	200	169	31
9b-c	909	2819	155	260.8	381.4	138.1	.9273	.9567	585	144	86.6	188	169	19
10:0	704.1	3000	87.8	205.6	407.8	188.9	.9041	.9028	585	144	86.6	67	41	26
10a	303	2803	87.8	205.6	324.3	179.9	.9594	.9802	585	144	86.6	200	144	56
10a-c	303	2803	87.8	205.6	323.7	185.8	.9320	.9455	585	144	86.6	159	144	15
10b	707.2	3803	87.8	205.6	377.3	164.7	.9439	.9761	585	144	86.6	200	188	12
10b-c	727.2	2803	87.8	205.6	379.6	168.0	.9333	.9557	585	144	86.6	179	188	-9
11:0	405.1	2564	81.9	156.3	331.0	154.3	.9475	.9503	585	73	0	183	188	-5
11a	424.2	2539	81.9	156.3	329.7	151.1	.9510	.9503	585	73	0	200	200	0
11a-c	424.2	2539	81.9	156.3	329.7	151.	.9506	.9503	585	73	0	200	205	-5
11b	909	2520	81.9	156.3	392.8	129.3	.9672	.9503	585	73	0	200	222	-22
11b-c	909	2520	81.9	156.3	389.9	132.0	.9396	.9503	585	73	0	184	222	-38
12:0	689.2	2437	111.6	224.5	342.8	134.3	.9493	.9642	390	110	43.3	242	227	15
12a	242.4	2511	111.6	224.5	295.5	155.6	.9636	.9599	390	110	43.3	200	171	29
12a-c	242.4	2511	111.6	224.5	294.9	158.6	.9423	.9440	390	110	43.3	176	171	5
12b	787.8	2500	111.6	224.5	364.5	136.5	.9385	.9517	390	110	43.3	200	160	40
12b-c	787.8	2500	111.6	224.5	364.2	136.5	.9386	.9519	390	110	43.3	200	160	40

Table 6.2 (continued)

13:0	405.1	2606	81.9	156.3	346.6	166.2	.9356	.9238	585	73	0	279	271	8
13a	303	2491	81.9	156.3	304.3	155.5	.9619	.9633	585	73	0	200	208	-8
13a-c	303	2491	81.9	156.3	312.8	157.4	.9401	.9399	585	73	0	211	208	3
13b	909	2487	81.9	156.3	359.0	140.2	.9416	.9569	585	73	0	200	194	6
13b-c	909	2487	81.9	156.3	359.5	142.2	.9399	.9490	585	73	0	223	194	29
14:0	689.1	2100	111.6	224.5	297.7	166.2	.9747	1.0065	390	110	43.3	-10	20	30
14a	303	2527	111.6	224.5	304.3	155.5	.9619	.9633	390	110	43.3	200	245	-45
14a-c	303	2527	111.6	224.5	312.8	157.4	.9401	.9399	390	110	43.3	242	245	-3
14b	727.2	2510	111.6	224.5	359.0	140.2	.9416	.9569	390	110	43.3	200	234	-34
14b-c	727.2	2510	111.6	224.5	359.5	142.2	.9399	.9490	390	110	43.3	207	234	-27
15:0	417.6	2325	85.8	217.2	307.1	138.6	.9647	.9845	390	108	43.3	290	299	-9
15a	303	2464	85.8	217.2	311.2	155.1	.9528	.9630	390	108	43.3	200	176	-24
15a-c	303	2464	85.8	217.2	312.4	155.7	.9497	.9583	390	108	43.3	207	176	-31
15b	727.2	2472	85.8	217.2	371.2	139.3	.9498	.9609	390	108	43.3	200	189	11
15b-c	727.2	2472	85.8	217.2	368.9	139.8	.9528	.9628	390	108	43.3	198	189	9
16:0	719.3	2691	43.8	210.0	406.5	162.3	.9309	.9328	585	148	43.3	262	251	11
16a	303	2574	43.8	210.0	344.0	164.4	.9629	.9669	585	148	43.3	200	204	-4
16a-c	303	2574	43.8	210.0	334.1	168.3	.9430	.9475	585	148	43.3	210	204	6
16b	909	2568	43.8	210.0	407.7	143.2	.9400	.9615	585	148	43.3	200	183	17
16b-c	909	2568	43.8	210.0	412.1	144.6	.9347	.9515	585	148	43.3	213	183	30

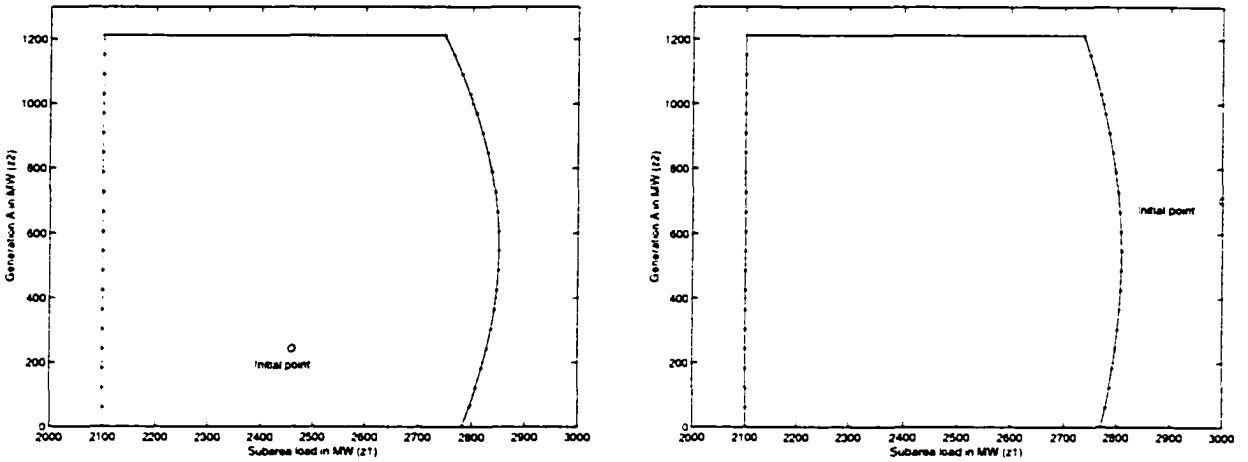


Figure 6.5 Voltage instability boundary for initial points 9:0 and 10:0

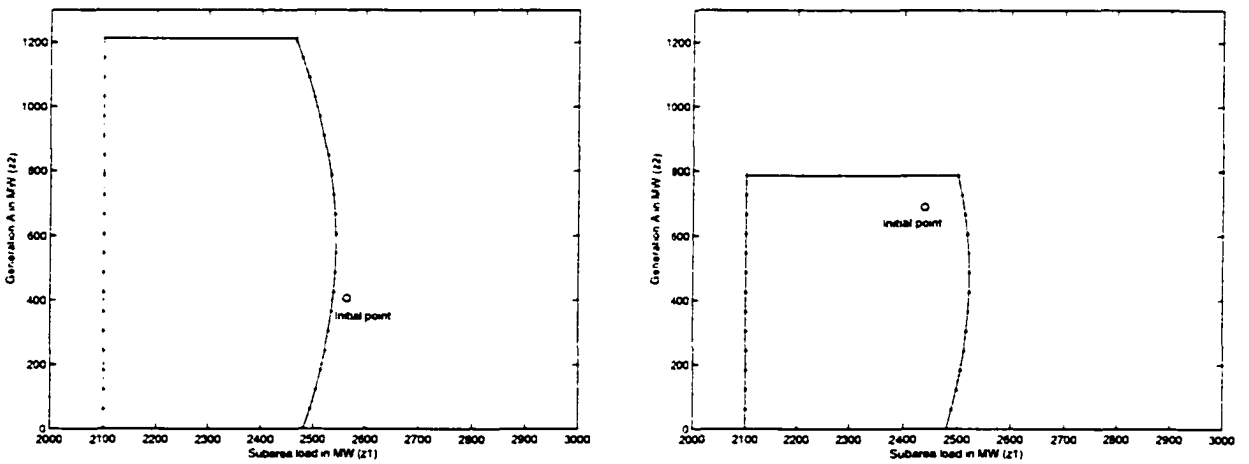


Figure 6.6 Voltage instability boundary for initial points 11:0 and 12:0

6.3.3.2 Accuracy assessment

For each boundary, two points were identified to test for accuracy. These points are labeled on Table 6.2 as 9a,9b, and 10a,10b, etc. The operating conditions corresponding to these points are designated using the same notation as in Table 6.2, where, for these entries in the table (9a,9b, and 10a,10b, etc.), the values of the dependent parmaters were

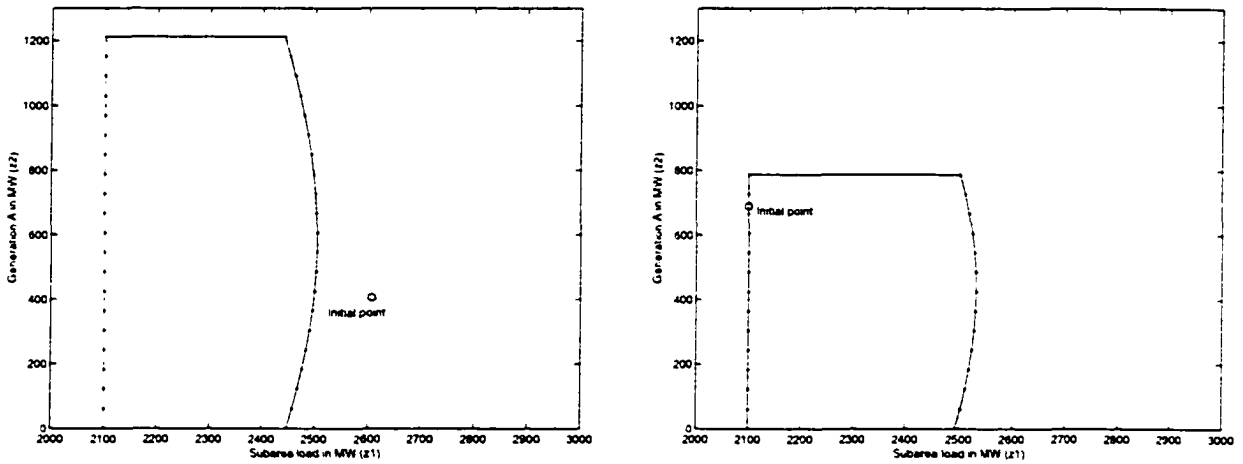


Figure 6.7 Voltage instability boundary for initial points 13:0 and 14:0

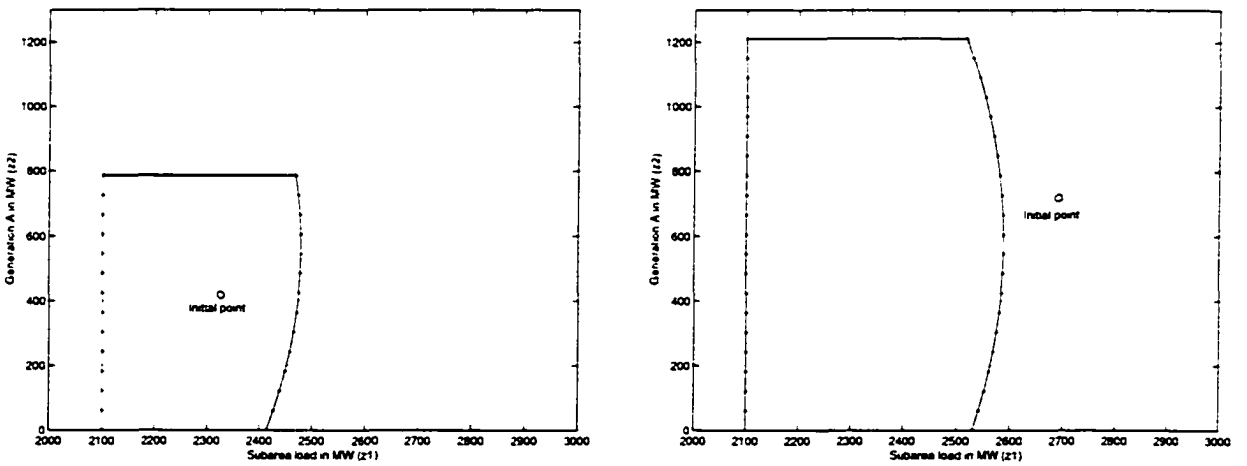


Figure 6.8 Voltage instability boundary for initial points 15:0 and 16:0

computed from *the initial point and the update functions* via the visualization program.

Also shown in Table 6.2, for each boundary point, is an additional point labeled with a "-c," e.g., 9a-c,9b-c and 10a-c,10b-c, etc. Here, the values of the dependent paramaters were computed from *a full power flow solution* (as opposed to using the initial point and the update functions); the dependent parameter values of entries 9a,9b, and 10a,10b, etc., have therefore been "corrected." These entries allow one to identify the error in the dependent value calculations caused by the update functions.

Table 6.2 also provides two R-values for each entry and an error value. One R-value is from the neural network and the other is "actual" as computed by manual solution of the power flow case and application of the VCMARGIN program. The difference between the two R-values provides the error. All R-values have been converted from per-unit (as actually computed by the neural network) to MVARS and represent the additional *purely* reactive load at the load bus necessary to cause voltage instability.

It is important to interpret the error results of Table 6.2 in a physically meaningful way. To this end, a sensitivity calculation was performed on several different operating conditions whereby the change in R for a 100 MW shift in the subarea load was computed. For all operating conditions, it was found that this 100 MW shift is equivalent to about 50 MVAr of variation in R, which is a 2:1 ratio of the subarea load MW to MVAr margin. This ratio enables us to make approximate interpretation of the MVAr margin error in terms of subarea MW load. For example, point 9a–c in Table 6.2 indicates a 49 MVAr margin error. This is equivalent to about $2 \times 49 - 98$ MW of subarea load, which is about 5% of the minimum typical load of 2100 MW.

The error for the initial point (e.g., 9:0, 10:0, etc.) is provided but is generally not indicative of the boundary error since it could be far away from the boundary where the neural network is most accurate. It can be seen that the maximum error for initial points is 67 MVAr.

The error for the boundary points influenced by the update functions (e.g., 9a,9b and

10a,10b, etc.) indicate the total error associated with the boundary at the evaluated point, where the total error is a composite of the error from the neural network and the error from the update functions. It can be seen that the maximum error for these boundary points is 71 MVAr, equivalent to 142 MW of the subarea load. This error should be assessed in light of the following comments:

- *Error and margin:* The fact that the error never exceeds the applied 200 MVAR margin implies that all points within the indicated boundary are in fact secure.

- *Neural Network Error:* The error for the points with "corrected" dependent parameter values (e.g., 9a-c,9b-c and 10a-c,10b-c, etc.) removes the error associated with the update functions and indicates only the error associated with the neural network. The largest neural network error is 49 MVAr, equivalent to about 98 MW of the subarea load.

- *Error and distance from boundary:* The largest error for cases with "uncorrected" dependent parameter values (e.g., 9a,9b, and 10a,10b, etc.) occurs for cases 9a, 10a, and 14a. In contrast, the error for other cases with "uncorrected" dependent parameter values is smaller. The reason for the larger error in cases 9, 10, and 14 relative to the other cases is attributed to the influence of the update functions. Cases 9, 10, and 14 are initialized by points that are more distant from the boundary than the points which initialize the other cases, as indicated by the larger difference between 200 (the boundary value) and MVAr margin at the initial point itself in entries 9:0, 10:0, and 14:0 relative to entries 11:0, 12:0, 13:0, 15:0, and 16:0. This can also be seen by inspecting Figures 6.5–6.8. Thus, for case 9, 10, and 14, the update functions have more influence.

6.3.4 Choice of presented parameters by users

So far, only the Subarea Load and Generation A are specified as presented parameters for boundary visualization. However, if the user is interested in boundaries using other critical parameters as presented parameters, these boundaries can also be easily obtained. Therefore, this gives more flexibility to the user on how he or she wants to view the system security. For example, Figures 6.9 - 6.12 show some boundaries using various critical parameters as the presented parameters for the thermal overload problem.

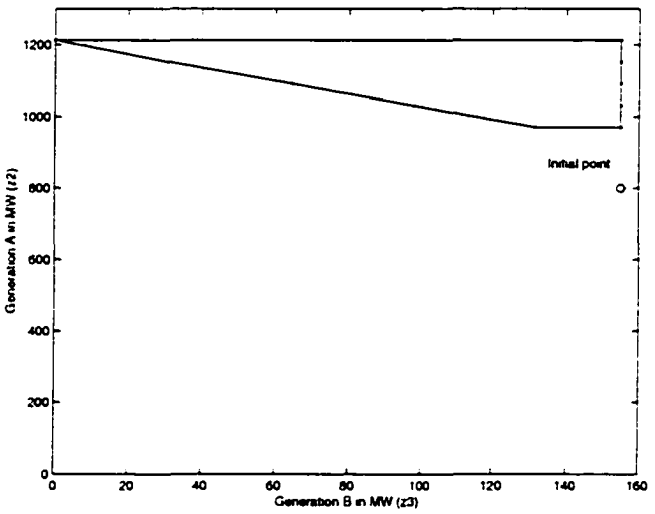


Figure 6.9 A boundary with presented parameters Generation A and Generation B.

6.4 Composite boundary visualization

6.4.1 Conceptual Formulation

The goal is to draw a single continuous boundary, i.e., find a closed secure region, given several individual security boundary characterizations so that each portion of the composite boundary corresponds to the most restrictive security problem under the given contingency. Suppose we have M individual boundary characterizations that are

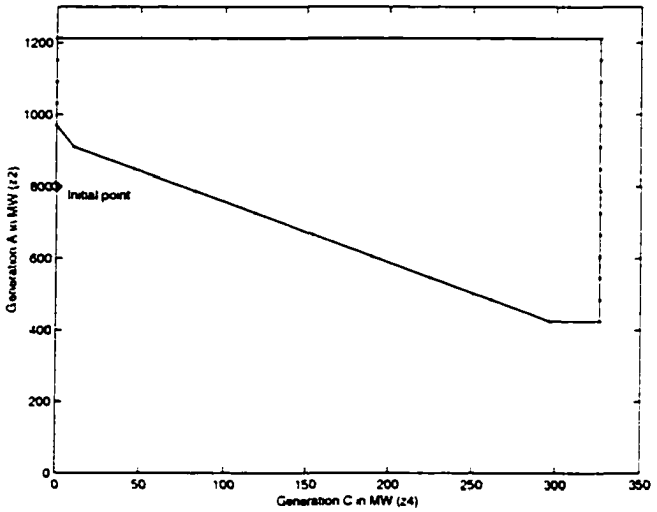


Figure 6.10 A boundary with presented parameters Generation A and Generation C.

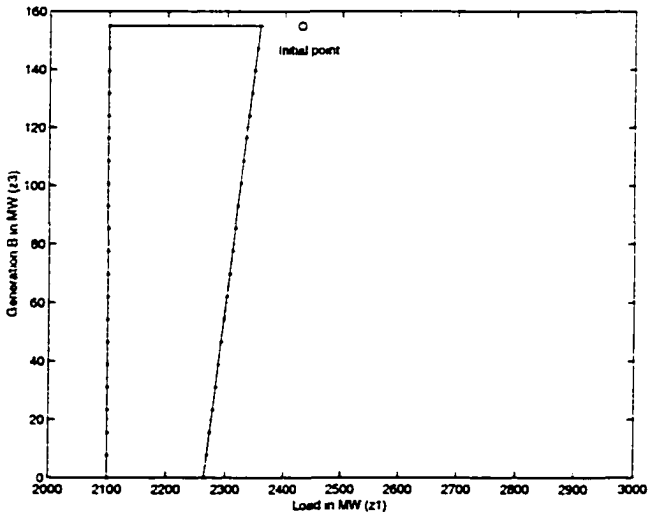


Figure 6.11 A boundary with presented parameters Generation B and Subarea Load.

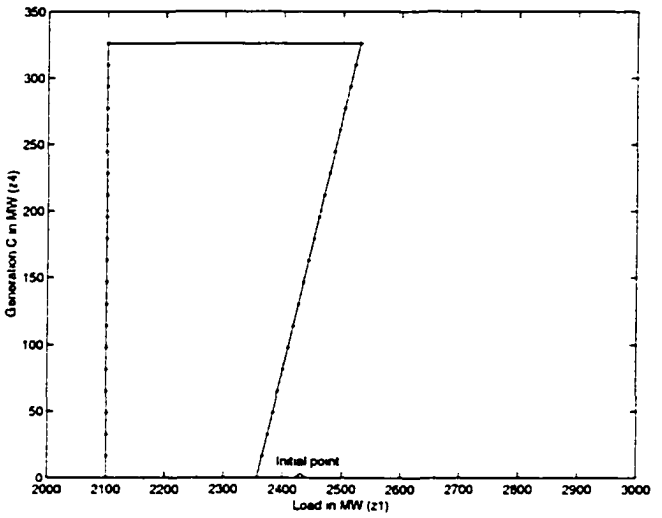


Figure 6.12 A boundary with presented parameters Generation C and Sub-area Load.

represented as

$$f_i(\mathbf{x}_i) = R_i, \ i = 1, \cdots, M \tag{6.5}$$

A neural network is trained for each of these problems and a C-code is extracted that characterizes the functional relationship $f_i(\mathbf{x}_i) = R_i$ between critical parameter $\mathbf{x}_i$ and the performance measure R_i specific to problem i. In general, $\mathbf{x}_i \neq \mathbf{x}_j$ if $i \neq j$. However, they have at least one common critical parameter, otherwise there is no reason to use them together to draw a composite boundary. For the purpose of illustration of the developed automatic security boundary characterization approach, it is assumed that there are two common critical parameters denoted by z_1 and z_2.

In order to obtain a continuous composite boundary (or equivalently, a closed secure region), we add following four boundaries:

$$z_1 - z_{1,min} = R_{M+1}$$

$$z_1 - z_{1,max} = R_{M+2}$$

$$z_2 - z_{2,min} = R_{M+3}$$

$$z_2 - z_{2,max} = R_{M+4}$$

These boundaries form a rectangle enclosing the region of interest in the study and correspond to the bounds on the presented parameters. Figure 6.13 shows a composite boundary formed by various individual boundaries. Clearly, the problem is to identify and draw only the portions of the boundaries enclosing the shaded region. Suppose $R_i < 0$ corresponds to the secure operating conditions and $R_i > 0$ to insecure conditions. The problem is then equivalent to finding the region S in which for every point $s(z_1, z_2) \in S$

$$f_i(\mathbf{x}_i) < 0. \ \forall \ i = 1 \ldots\ldots M \tag{6.6}$$

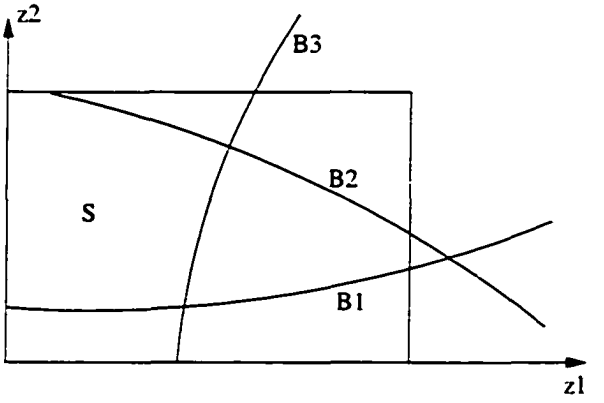


Figure 6.13 A composite boundary formed by individual boundaries.

6.4.2 Algorithm Description

As shown in Figure 6.13, some points on some individual boundaries do not contribute to forming the composite boundary. To reduce computational cost, it is desired to avoid drawing these portions of the boundaries. Therefore, starting from the minimum value of z_1, for each interval, as in Figure 6.14, two or more individual boundary functions

that are binding need to be identified. To do this, the functions are ranked in descending order of z_2, which is obtained by solving

$$f_j(z_1, z_2, \ldots) = 0, \; j = 1, \ldots, M \tag{6.7}$$

using a bisection search technique. Neighboring boundary functions are paired according to which one follows the other in this ranking. For each pair of neighboring functions, an interior check is performed for security, i.e., an arbitrarily selected point (marked with crosses) between two neighboring boundaries is checked to see if it satisfies (6.6). If so, this point is inside the secure region, and the corresponding neighboring individual boundary functions must be the binding functions for the composite boundary for this interval. The composite boundary is therefore identified as the pair of individual boundaries, and all other individual boundaries are ignored in this interval.

If, in the next interval, there are no other individual boundary functions between the two binding functions identified in the previous interval, then these functions are also binding for the new interval. In this case, it is not necessary to perform the interior check for this interval. This is why there are no crosses for some intervals in Figure 6.14. Once it is no longer possible to find any secure point, i.e., there are no neighboring individual boundary functions for which (6.6) is satisfied, then z_1 has exceeded its feasible or secure range, and the algorithm stops. At this point, a secure region is formed and a continuous boundary is obtained, as in Figure 6.14. This algorithm is applicable to both linear and nonlinear individual boundaries as long as they form one single continuous region. Cases where individual boundaries form several disjoint regions are not considered because they are rare in practice.

6.4.3 Applications to the sample system

The critical parameters chosen by the feature selection software are indicated in Tables 6.3 and 6.4 together with their values corresponding to the initial operating

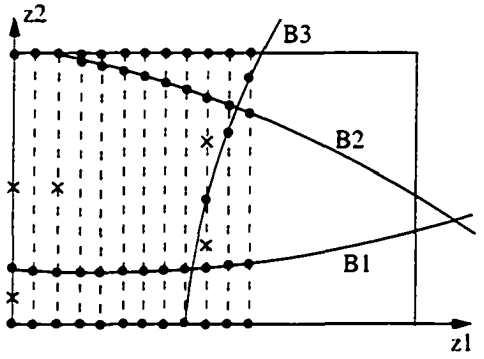


Figure 6.14 Illustration of the algorithm to determine the composite boundary.

point.

Table 6.3 Initial operating condition for line thermal overload and transformer overload

Op. Pt. No.	Independent Parameters (MW)				Dependent Parameters (flow:MW,voltage:p.u.)			
	Load z_1	GenA z_2	GenB z_3	GenC z_4	Flow13 y_1	Flow14 y_2	Volt4 y_3	Volt12 y_4
1	2430.0	800.0	260.8	155.0	140.17	220.57	1.0063	1.0712

Table 6.4 Initial operating condition for voltage instability

Op. Pt. No.	Independent Parameters (MW)				Dependent Parameters (flow:MW,voltage:p.u.)				Independent Parameters (MVAr)		
	Load z_1	GenA z_2	GenB z_3	GenC z_4	Flow9 y_1	Flow13 y_2	Volt7 y_3	Volt14 y_4	Qmax1 z_5	Qmax3 z_6	Qmax4 z_7
1	2430.0	800.0	260.8	155.0	240.6	140.17	0.9673	1.0005	585.0	144.0	86.6

Figures 6.15 – 6.17 show the boundaries for the line thermal overload, transformer

overload. and voltage instability problems. respectively. The corresponding composite boundary for all three problems is shown in Figure 6.18. The region enclosed by the composite boundary indicates the secure operating region. Note that only a portion of each individual boundary is used to obtain the composite boundary. The initial operating conditions for these examples are shown in Tables 6.3 and 6.4.

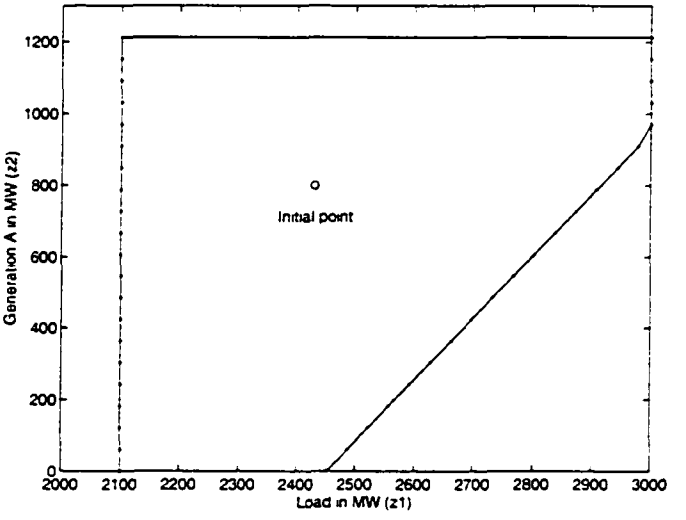


Figure 6.15 Line thermal overload boundary.

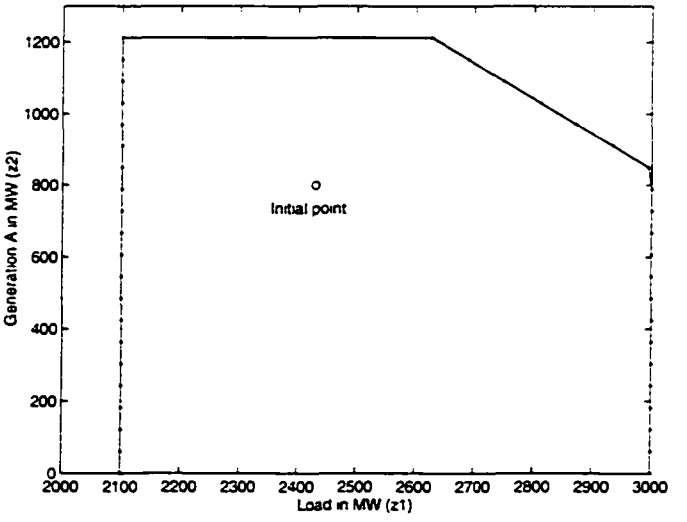


Figure 6.16 Transformer overload boundary.

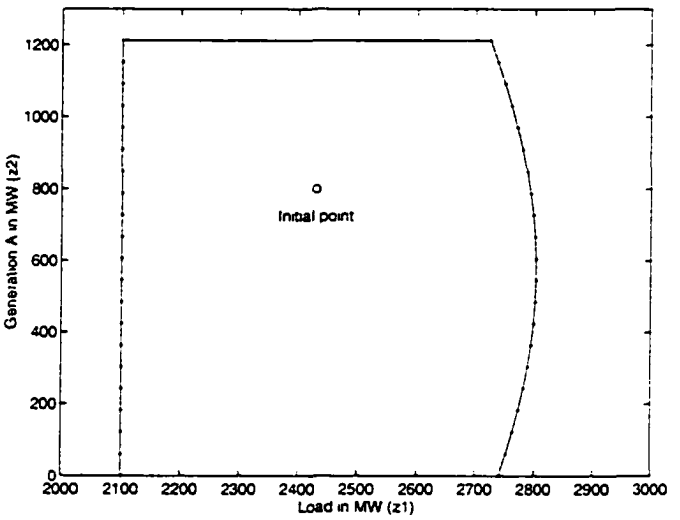


Figure 6.17 Voltage instability boundary.

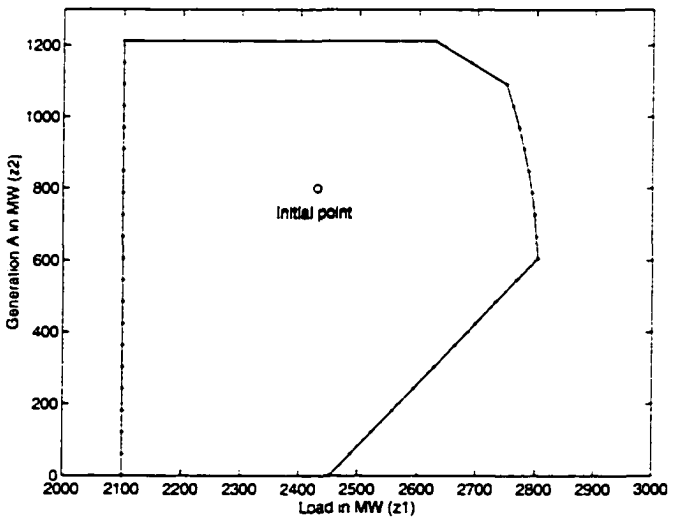


Figure 6.18 Composite boundary for line thermal overload, transformer overload, and voltage instability problems.

6.5 Available transfer capability calculation

According to North American Electric Reliability Council (NERC) [83], available transfer capability (ATC) is defined as a measure of the transfer capability remaining in the physical transmission network for further commercial activity over and above already committed uses. The ATC between two areas provides an indication of the maximum amount of electric power above base case conditions that can be transferred reliably from one area to another over all transmission facilities without violating a specific set of precontingency and postcontingency criteria for the facilities in the control area under study.

After we obtain the security boundary, it is straightforward to calculate the ATC for the interface between the area and the subarea once a specific direction of changes in the critical parameter values is given.

Figure 6.19 shows the ATC calculation result for a composite boundary. The ATC for Generation A is 132.4585 MW and it is 76.4749 for the subarea load, given that Generation A and Subarea load increase at the same step from the current operating point.

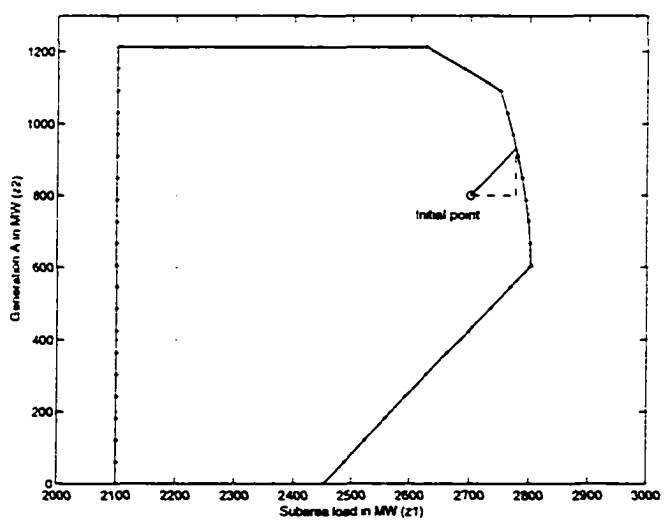


Figure 6.19 The ATC calculation illustration.

6.6 Obtaining update functions

As mentioned earlier, if there are covered dependent critical parameters, the dependency of these covered parameters on the presented critical parameters must be modeled since variations on the presented critical parameters in drawing the boundary will influence the dependent parameters according to the power flow equation constraints. This dependency can be modeled using appropriate update functions.

When the presented parameters are real power injections, as is often the case, an allocation rule must be defined on how to compensate for changes in real power generation or demand. Suppose there are M_g load following generators. The allocation coefficients A_m's are defined as the percentage of the power imbalance caused by Δz_1 or Δz_2 that is redispatched to load following generator m. These factors may be obtained from economic dispatch base points and participation factors [86, pp. 44-46], or they may be simply estimated, as in this research.

The update function is

$$\begin{aligned} y_i^{(k)} &= g_{y_i}(y_i^{(0)}, \Delta z_1, \Delta z_2) \\ &= y_i^{(0)} + \textstyle\sum_{m=1}^{M_g}[A_m K_{i,m1}\Delta z_1 + A_m K_{i,m2}\Delta z_2] \\ &= y_i^{(0)} + \textstyle\sum_{m=1}^{M_g}[b_{i,m1}\Delta z_1 + b_{i,m2}\Delta z_2] \end{aligned} \tag{6.8}$$

where the distribution coefficients $K_{i,mj}$ may be computed using DC load flow approximations. They can also be obtained via sensitivity analysis using an AC power flow program where, for example, presentation parameter 1 is increased by Δz_1, and this increase is compensated by load following generator m according to the allocation coefficients, and the change in flow Δy_i on circuit i is obtained (an increase is positive and a decrease is negative), so that $K_{i,m1} = \frac{\Delta y_i}{\Delta z_1}$. $b_{i,mj} = A_m K_{i,mj} (j = 1, 2)$ indicates the change in circuit flow or PQ bus voltage magnitude y_i due to the portion of change in z_j, which is compensated by load following generator m. $K_{i,mj}$ is the increase in circuit i flow or bus i voltage magnitude when generator m compensates for a 1 MW

increase in z_j. The terms Δz_1 and Δz_2 are both positive when the parameter *increases*, independent of whether the parameter represents load or generation.

6.7 Summary

The presentation techniques have been developed in this chapter for on-line visualization of boundaries. For one security problem under a specific contingency, a search algorithm has been developed to obtain the boundary points in the two-dimensional space based on the neural network mapping function. In drawing the boundary, the influence of variations of the two presented critical parameters on the dependent critical parameters is modeled using update functions from linearization of power flow equations. For multiple security problems under the same contingency, a composite boundary visualization algorithm has been proposed to efficiently account for the multiple security problems. The algorithm is applicable to any linear or nonlinear boundaries. Based on the security boundary, an efficient approach has been proposed to calculate the available transfer capability.

7 CONCLUSIONS AND FURTHER WORK

7.1 Conclusions

Power system security is always an important problem. In today's open access power market, it is even more important since typical system operating conditions are much closer to the system limits. It is not only necessary to know whether the system is secure or not, but also necessary to be aware of how secure it is. The traditional security boundary visualization approach results in a two-dimensional graph called a nomogram. However, an intensive labor involvement, inaccurate boundary representation, and little flexibility for integration with EMS greatly restrict use of nomograms under competitive utility environment. Motivated by the new operation environment and based on the traditional nomogram development procedure, an automatic security boundary visualization methodology has been developed to measure the system security level in terms of controllable operating parameters. This methodology provides a new security assessment tool for power system operations. It can be easily used on-line in the control center and hence the sytem security level can be promptly captured by the operator through viewing the visualized boundary. Therefore, the work in this dissertation has produced an approach that provides operators with a way to securely operate the system very close to the boundary. This means that the existing transmission equipment can be used more fully, and as a consequence, the electric energy marketplace is less constrained and able to operate closer to its economic optimum.

The main steps involved in the methodology include data generation, feature selec-

tion, neural network training, and visualization. All these steps have been investigated to improve the accuracy or the solution speed of security boundaries.

In data generation, a systematic approach to data generation has been developed to generate high quality data. The resulting software, ASAS, is modular and cen be easily modified for different security problems.

In data analysis, several data analysis techniques have been used to analyze the data generated by ASAS before neural network training. The techniques for outlier detection are the principal component analysis, the local training data density estimation, the multivariate normality test, the Kohonen self-organizing map, the scatter plots, and the projection pursuit visualization. Of the above, the scatter plots and the projection pursuit visualization can also be used to analyze data distributions. Because of its speed, the local training data density estimation is appropriate for determining whether a new sample is an outlier with respect to the training data set. This is useful in the control room to determine whether an operating condition is within the range of neural network capability. All of the other methods are most appropriate for analyzing the training data set before neural network training, and they are supplementary to each other.

In feature selection, three classes of feature selection methods have been discussed: cluster analysis, Kohonen self-organizing map, and genetic algorithm based methods. The objective is to select the best features as neural network input parameters. The direct results from the cluster analysis and the Kohonen self-organizing map are clusters of similar features. For feature selection, one feature needs to be picked up from each cluster to represent this cluster. Therefore, there are different feature selection results depending on which feature is selected from each cluster. Some results may give higher accuracy, some results may give lower accuracy. There is no guarantee that acceptable accuracy is achieved although these methods are fast.

In consideration of this fact, genetic algorithm based methods have been preferred to perform feature selection. In this approach, various techniques have been tried for use

as the GA fitness function. These include multilayer perceptrons, radial basis function networks, and k-nearest neighbor. Of these, use of k-nearest neighbor has the fastest solution speed but the lowest accuracy. On the other hand, use of multilayer perceptrons results in the slowest solution speed but highest accuracy. By making a trade-off between solution speed and accuracy, use of radial basis function networks is recommended because it is much faster than use of multilayer perceptrons without much loss of accuracy.

In neural network analysis, a confidence interval calculation method has been derived and implemented for MLPs. The confidence interval provides a measure for reliability of the neural network output. This is important because it guides users in interpretation and use of the neural network outputs. In addition, sensitivity analysis of the neural network output with respect to input parameters has also been derived. With sensitivity analysis, one can easily assess which input parameter is more important in contribution to the neural network output. This information can provide guidance on preventive control regarding the security boundary.

In boundary presentation, a composite security boundary visualization algorithm has been proposed to present accurate boundaries in two dimensional diagrams to operators. The algorithm can be applied to any type of security problem, therefore it provides a general way to integrate boundaries corresponding to problems of different types. Based on the boundary, the available transfer capability can be obtained.

7.2 Further work

Further work can be done to improve the accuracy of boundaries, the overall solution speed, and the usefulness of the boundaries. In data generation, modifications are needed for ASAS so that the procedure can handle the influence of topological changes. In addition, it would be more efficient to integrate the data analysis techniques into

ASAS rather than use these techniques after ASAS runs. In feature selection and neural networks, parallel computing is worth investigation so that the solution speed can be further improved. Also, implementation of on-line training is of interest because the neural network needs to be retrained if the operating conditions or system topology changes significantly. In boundary presentation, it would be useful to investigate methods of increasing computational speed for the visualization software because the current software may be too slow for a large number of security problems. So far, the discussions are restricted to deterministic boundaries, i. e., constant system performance. It would be of interest to develop contours of constant risk that can be used to balance economics and security levels.

BIBLIOGRAPHY

[1] P. Shanahan and S. Naumann, "Evaluation of simultaneous transfer capabilities," *Proceedings of the 1995 American Power Conference*, Chicago, IL, April 1995, pp. 1463-1468.

[2] J. Kanetkar and S. Ranade, "Compact representation of power system security - a review," *Proceedings of the North American Power Symposium*, Reno, Nevada, October 1992, pp. 312-321.

[3] R. Farmer, "Present day power system stability analysis methods in the Western United States," *Proceedings of the International Symposium on Power System Stability*, Ames, IA, May 13-15, 1985, pp. 39-44.

[4] J. D. McCalley, S. Wang, Q. Zhao, G. Zhou, R. T. Treinen, and A. Papalexopoulos, "Security boundary visualization for systems operation," *IEEE Transactions on Power Systems*, vol. 12, no. 2, May 1997, pp. 940-947.

[5] G. Zhou and J. D. McCalley, "Composite security boundary visualization," Accepted to be published in *IEEE Transactions on Power Systems*.

[6] G. Zhou, V. Van Acker, S. Wang, M. A. Mitchell, and J. D. McCalley, "On-line visualization of security boundaries using neural networks with feature selection," Submitted to *International Journal of Engineering Intelligent Systems*, December 1997 (in review).

[7] J. D. McCalley, G. Zhou, and V. Van Acker, "Power system security boundary visualization using neural networks," Submitted to *International Journal of Neuro-*

computing, A Special Issue on Applications of Artificial Neural Networks in Power Systems, December 1997 (in review).

[8] J. D. McCalley, G. Zhou, V. Van Acker, M. A. Mitchell, V. Vittal, S. Wang, and J. A. Peças Lopes, "Power system security boundary visualization using neural networks," *Proceedings of Bulk Power System Dynamics and Control IV - Restructuring*, Santorini, Greece, August 24-28, 1998.

[9] P. Kundur, *Power System Stability and Control*, New York: McGraw-Hill, 1994.

[10] C. Taylor, *Power System Voltage Stability*, New York: McGraw-Hill, 1993.

[11] V. Ajjarapu and C. Christy, "The continuation power flow: a tool for steady state voltage stability analysis," *IEEE Transactions on Power Systems*, vol. 7, no. 1, February 1992, pp. 417-423.

[12] B. Lee and V. Ajjarapu, "Invariant subspace parametric sensitivity (isps) of structure preserving power system models," *IEEE Transactions on Power Systems*, vol. 11, no. 2, May 1996, pp. 845-850.

[13] M. M. Begovic and A. G. Phadke, "Control of voltage stability using sensitivity analysis," *IEEE Transactions on Power Systems*, vol. 7, no. 1, February 1992, pp. 115-123.

[14] P-A Lof, T. Smed, G. Andersson, and D. J. Hill, "Fast calculation of a voltage stability index," *IEEE Transactions on Power Systems*, vol. 7, no. 1, February 1992, pp. 55-64.

[15] C. A. Canizares and F. L. Alvarado, "Point of collaps and continuation methods for large ac/dc systems," *IEEE Transactions on Power Systems*, vol. 8, no. 1, February 1993, pp. 1-8.

[16] T. V. Cutsem, "An approach to corrective control of voltage instability using simulation and sensitivity," *IEEE Transactions on Power Systems*, vol. 10, no. 2, May 1995, pp. 616-622.

[17] I. Dobson and L. Lu, "New methods for computing a closest saddle node bifurcation and worst case load power margin for voltage collapse," *IEEE Transactions on Power Systems*, vol. 8, no. 3, August 1993, pp. 905-913.

[18] K. Demaree, T. Athay, K. W. Cheung, Y. Mansour, E. Vaahedi, A. Y. Chang, B. R. Corns, and B. W. Garrett, "An on-line dynamic security analysis system implementation," *IEEE Transactions on Power Systems*, vol. 9, no. 4, November 1994, pp. 1716-1722.

[19] G. C. Ejebe, C. Jing, J. G. Waight, V. Vittal, G. Pieper, F. Jamshidian, D. Sobajic, and P. Hirsch, "On-line dynamic security assessment: transient energy based screening and monitoring for stability limits," *1997 IEEE PES Summer Meeting*, Berlin, Germany, July 20-24, 1997.

[20] D. Sobajic and Y. Pao, "Artificial neural-net based dynamic security assessment for electric power systems," *IEEE Transactions on Power Systems*, vol. 4, Feb., 1989, pp 220-228.

[21] M. Aggoune, M. El-Sharkawi, D. Park, M. Damborg, and R. Marks, "Preliminary results on using artificial neural networks for security assessment," *IEEE Transactions on Power Systems*, vol. 6, no. 2, May 1991, pp. 890-896.

[22] Q. Zhou, J. Davidson, and A. Fouad, "Application of artificial neural networks in power system security and vulnerability assessment," *IEEE Transactions on Power Systems*, vol. 9, no. 1, Feb., 1994, pp. 525-532.

[23] V. Miranda, J. Fidalgo, J. Peças Lopes, and L. Almeida, "Real time preventive actions for transient stability enhancement with a hybrid neural network - opti-

mization approach," *IEEE Transactions on Power Systems* vol. 10, no. 2, May, 1995, pp 1029-1035.

[24] A. A. El-Keib and X. Ma, "Application of artificial neural networks in voltage stability assessment," *IEEE Transactions on Power Systems*, vol. 10, no. 4, November 1995, pp. 1890-1896.

[25] J. A. Momoh, L. G. Dias, and R. Adapa, "Voltage stability assessment and enhancement using artificial neural networks and reactive compensation," *Proceedings of the International Conference on Intelligent Systems Applications to Power Systems*, Orlando, FL, January 28-February 2, 1996, pp. 410-415.

[26] M. La Scala, M. Trovato, and F. Torelli, "A neural network-based method for voltage security monitoring," *IEEE Transactions on Power Systems*, vol. 11, no. 3, August 1996, pp. 1332-1341.

[27] D. Niebur and A. J. Germond, "Power system static security assessment using the Kohonen neural network classifier," *IEEE Transactions on Power Systems*, vol. 7, no. 2, May 1992, pp. 865-872.

[28] M. A. El-Sharkawi and R. Atteri, "Static security assessment of power system using Kohonen neural network," *Proceedings of the Second International Forum on Applications of Neural Networks to Power Systems*, Yokohama, Japan, April 19-22, 1993, pp. 373-377.

[29] D. J. Sobajic and Y-H. Pao, "Combined use of unsupervised and supervised learning for dynamic security assessment," *IEEE Transactions on Power Systems*, vol. 7, no. 2, May 1992, pp. 878-884.

[30] T. S. Dillon and D. Niebur, eds., *Neural Networks Applications in Power Systems*, London, UK: CRL Publishing Ltd., 1996.

[31] M. El-Sharkawi and D. Niebur, eds., *Artificial Neural Networks with Applications to Power Systems*. IEEE PES special publication 96 TP 112-0, 1996.

[32] L. Wehenkel, T. Van Cutsem, and M. Pavella, "An artificial intelligence framework for on-line transient stability assessment of power systems," *IEEE Transactions on Power Systems*, vol. 4., no. 2, May 1989, 789-800.

[33] L. Wehenkel, M. Pavella, E. Euxibie, B. Heilbronn, "Decision tree based transient stability method: a case study," *IEEE Transactions on Power Systems*, vol. 9., no. 1, Feb. 1994, pp. 459-469.

[34] L. Wehenkel and M. Pavella, "Decision tree approach to power systems security assessment," *Electrical Power and Energy Systems*, vol. 15, no. 1, Feb., 1993, pp. 13-36.

[35] T. Van Cutsem, L. Wehenkel, M. Pavella, B. Heilbronn, and M. Goubin, "Decision tree approaches to voltage security assessment," *IEE Proceedings-C*, vol. 140, no.3, May 1993, pp. 189-198.

[36] L. Wehenkel, *Machine Learning Algorithms to Power System Security Assessment*, Aggregate Dissertation, University of Liege, Belgium, May 1994.

[37] S. Rovnyak, S. Kretsinger, J. Thorp, and D. Brown, "Decision trees for real-time transient stability prediction," *IEEE Transactions on Power Systems*, vol. 9., no. 3, Aug. 1994, pp 1417-1426.

[38] N. Hatziargyriou, G. Contaxis, and N. Sideris, "A decision tree method for on-line steady-state security assessment," *IEEE Transactions on Power Systems*, vol. 9, no. 2, May 1994, pp 1052-1061.

[39] N. Hatziargyriou, S. Papathanassiou, and M. Papadopoulos, "Decision trees for fast security assessment of autonomous power systems with a large penetration

from renewables," the 1994 PES Summer Meeting, Paper 94 SM 368-1 EC, San Francisco, 1994.

[40] C. Yang and Y. Hsu, "Estimation of line flows and bus voltages using decision trees," *IEEE Transactions on Power Systems*, vol. 9, no. 3, August 1994, pp 1569-1574.

[41] J. A. Peças Lopes, J. V. Ranito, J. Neto, and N. Hatziargyriou, "Derivation of classification structures for fast evaluation of dynamic security assessment in power systems using genetic algorithms," *Proceedings of Stockholm Power Technology Conference*, Stockholm, Sweden, 1995.

[42] N. D. Hatziargyriou, S. A. Papathanassiou, J A. Peças Lopes, J. N. Fidalgo, and V. Van Acker, "Fast dynamic security assessment of autonomous power systems with a large penetration from wind - The Lemnos study case," *European Wind Energy Association Conference and Exhibition*, Thessaloniki, Greece, October 1994.

[43] R. Marceau, R. Mailhot, and F. Galiana, "A generalized shell for dynamic security analysis in operations planning," *IEEE Transactions on Power Systems*, vol. 8, no. 3, Aug., 1993, pp 1098-1832.

[44] M. Huneault, C. Rosu, R. Manoliu, and F. Galiana, "A study of knowledge engineering tools in power engineering applications," *IEEE Transactions on Power Systems*, vol. 9, no. 4, Nov., 1994, pp 1825-1832.

[45] L. Wehenkel and V. Akella, "A hybrid decision tree-neural network approach for power system dynamic security assessment," *Proceedings of the 4th International Symposium on Expert Systems application to Power Systems*, Melbourne, Australia, January 1993, pp. 285-291.

[46] L. Wehenkel, T. Van Cutsem, M. Pavella, Y. Jacquemart, B. Heilbronn, and P. Pruvot, "Machine learning, neural networks, and statistical patterns recognition for voltage security: a comparative study," *Engineering Intelligent Systems*, vol. 2, no. 4, December 1994.

[47] H. H. Yan, J-C. Chow, M. Kam, R. Fischl, and C. R. Sepich, "Hybrid expert system/neural network hierarchical architecture for classifying power system contingencies," *Proceedings of the First International Forum on Applications of Neural Networks to Power Systems*, Seattle, WA, July 23-26, 1991, pp. 76-82.

[48] M. A. El-Sharkawi and S. J. Huang, "Application of genetic-based neural networks to power system static security," *Proceedings of International Conference on Intelligent Systems Applications to Power Systems*, Montpellier, France, September 5-9, 1994, pp. 423-430.

[49] M. A. El-Sharkawi and S. J. Huang, "Development of genetic algorithm embedded Kohonen neural network for dynamic security assessment," *Proceedings of International Conference on Intelligent Systems Applications to Power Systems*, Orlando, FL, January 28-February 2, 1996, pp. 44-49.

[50] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling, "Numerical recipes: the art of scientific computing," New York: Cambridge University Press, 1986.

[51] N. R. Draper and H. Smith, *Applied Regression Analysis*, 2nd edition, New York: John Wiley & Sons, 1981.

[52] A. J. Miller, *Subset Selection in Regression*, New York: Chapman and Hall, 1990.

[53] J. Neter, W. Wasserman, and M. H. Kutner, *Applied Linear Statistical Models*, 3rd edition, Homewood, IL: Irwin, 1990.

[54] P. A. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach*, Englewood Cliffs, NJ: Prentice-Hall International, 1982.

[55] A. N. Mucciardi, and E. E. Gose, " A comparison of seven techniques for choosing subsets of pattern recognition properties," *IEEE Transactions on Computers*, vol. 20, no. 9, 1971, pp. 1023-1031.

[56] G. H. John, R. Kohavi, and K. Pfleger, "Irrelevant features and the subset selection problem," *Machine Learning: Proceedings of the Eleventh International Conference*. San Francisco, CA: Morgan Kaufmann, 1994.

[57] _____, *NeuralWorks Predict Manual*. NeuralWare, Inc., Pittsburgh, PA, 1995.

[58] H. Vafaie, and K. De Jong, "Genetic algorithms as a tool for feature selection in machine learning," *Fourth International Conference on Tools with Artificial Intelligence*. Arlington, VA, November 1992, pp. 200-203.

[59] T. J. Downey, and D. J. Meyer, "A genetic algorithm for feature selection." *Proceedings of the Artificial Neural Networks in Engineering Conference*. St. Louis, MO, 1994, pp. 363-368.

[60] J. M. Steppe, K. W. Bauer, and S. K. Rogers, "Integrated feature and architecture selection," *IEEE Transactions on Neural Networks*, vol. 7, no. 4, July 1996, pp. 1007-1014.

[61] Q. Zhao, *Genetic Algorithm/Neural Network Feature Extraction and Application to Power System Security Assessment*, M.S. Thesis, Iowa State University, Ames, IA, 1996.

[62] J. Yang and V. Honavar, "Feature subset selection using a genetic algorithm," *Proceedings of International Conference on Genetic Programming*, Stanford, CA, July 13-16, 1997, pp. 380-385.

[63] M. B. Zayan, M. A. El-Sharkawi, and N. R. Prasad, "Comparative study of feature extraction techniques for neural network classifier," *Proceedings of International Conference on Intelligent Systems Applications to Power Systems*, Orlando, FL, January 28-February 2, 1996, pp. 400-404.

[64] S. Muknahallipatna and B. H. Chowdhury, "Input dimension reduction in neural network training - case study in transient stability assessment of large sys-

tems." *Proceedings of International Conference on Intelligent Systems Applications to Power Systems*, Orlando, FL, January 28-February 2, 1996, pp. 50-54.

[65] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Reading, MA: Addison-Wesley, 1989.

[66] L. Chambers, ed., *Practical Handbook of Genetic Algorithms: Applications, Volume I*, Boca Raton, Florida: CRC Press, 1995.

[67] D. S. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex Systems*, 2, 1988, pp. 321-355.

[68] J. Moody and C. J. Darken, "Fast learning in networks of locally tuned processing units," *Neural Computation*, vol. 1, 1989, pp. 281-294.

[69] S. Renals, "Radial basis function network for speech pattern classification," *Electronics Letters*, vol. 25, 1989, pp. 437-439.

[70] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proceedings of the IEEE*, vol. 78, 1990, pp. 1481-1497.

[71] J. Park and I. W. Sandberg, "Universal approximation using radial basis function networks," *Neural Computation*, vol. 3, 1991, pp. 246-257.

[72] F. Girosi and T. Poggio, "Networks and the best approximation property," *Biological Cybernetics*, vol. 63, 1990, pp. 169-176.

[73] H. C. Romesburg, *Cluster Analysis for Researchers*, Belmont, California:Lifetime Learning Publications, 1984.

[74] I. T. Jolliffe, *Principal Component Analysis*, New York: Springer-Verlag, 1986.

[75] R. H. Rencher, *Methods of Multivariate Analysis*, New York: John Wiley & Sons, 1995.

[76] R. H. Myers, *Classical and Modern Regression with Application, 2nd edition*, Belmont, California: Duxbury Press, 1990.

[77] S. Haykin, *Neural Networks: A Comprehensive Foundation*, New York, New York: Macmillan College Publishing, 1994.

[78] D. Rumelhart and J. McClelland, eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1, Cambridge, MA: MIT Press, 1986.

[79] K. Swingler, *Applying Neural Networks: A Practical Guide*, San Diego, CA: Academic Press, 1996.

[80] D. M. Skapura, *Building Neural Networks*, New York, New York: ACM Press, 1996.

[81] G. Chryssolouris, M. Lee, and A. Ramsey, "Confidence interval prediction for neural network models," *IEEE Transactions on Neural Networks*, vol. 7, no. 1, January 1996, pp. 229-232.

[82] G. Zhou, J. D. McCalley, and V. Honavar, "Power system security margin prediction using radial basis function networks," *Proceedings of the 29th North American Power Symposium*, Laramie, Wyoming, October 13-14, 1997, PP. 192-198.

[83] _____, *Available Transfer Capability Definitions and Determination*, North American Electric Reliability Council, June 1996.

[84] D. Cook, A. Buja, and J. Cabrera, "Direction and motion control in the grand tour," *Proceedings of the 23rd Symposium on the Interface*, Springer-Verlag, 1991.

[85] D. F. Swayne, D. Cook, and A. Buja, *User's Manual for XGobi: A Dynamic Graphics Program for Data Analysis Implemented in the X Window System (Release 2)*, Bellcore, 1991.

[86] A. Wood and B. Wollenberg, *Power Generation, Operation, and Control*, New York: John Wiley & Sons, 1984.

[87] H. Kumamoto and E. J. Henley, *Probabilistic Risk Assessment and Management for Engineers and Scientists*, New York: IEEE Press, 1996.

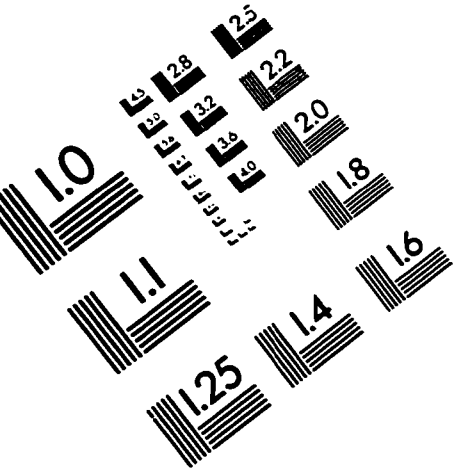
ACKNOWLEDGMENTS

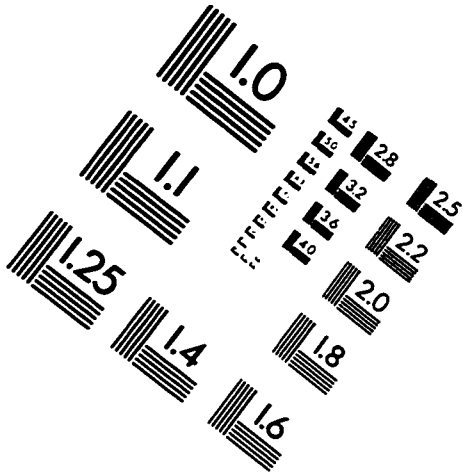
I would like to express my sincere gratitude to my major professor Dr. James D. McCalley for his guidance and inspiration. I would also like to thank my committee members, Dr. Gerald B. Sheblé, Dr. Venkataramana Ajjarapu, Dr. Lalita Udpa, and Dr. Vasant Honavar, for their valuable comments.

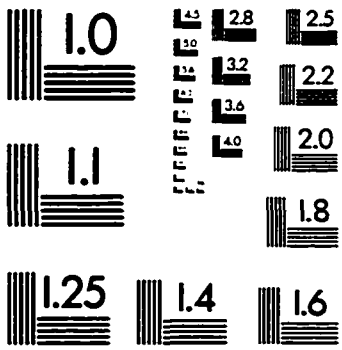
Thanks also go to my colleagues and fellow students, Dr. Shimo Wang, Mr. Qianglin Zhao, Mr. Chin Khor, Mr. Vincent Van Acker, Mr. Matthew A. Mitchell, and Mr. Bradley M. Nickell, with whom I worked harmoniously and discussed many useful ideas regarding the research, to the faculty and staff in power group for their supportive and cooperative attitude, to Pacific Gas and Electric Company for their financial support for the project upon which this dissertation is based, and to Electric Power Research Center at Iowa State University for their financial support for the research work in this dissertation.

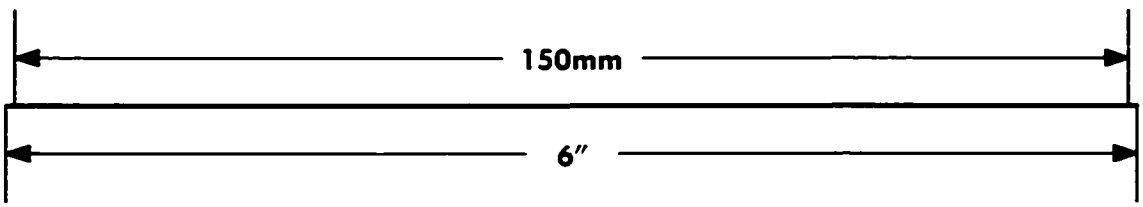
Last, but by no means least, I wish to thank my parents, Zhiying Zhou and Guilan Tang, for their love and support throughout my life, and my wife, Lixin Wang, for her love and understanding during my study.

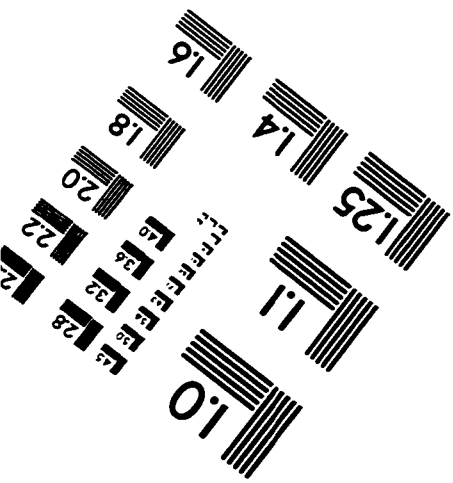
IMAGE EVALUATION
TEST TARGET (QA-3)

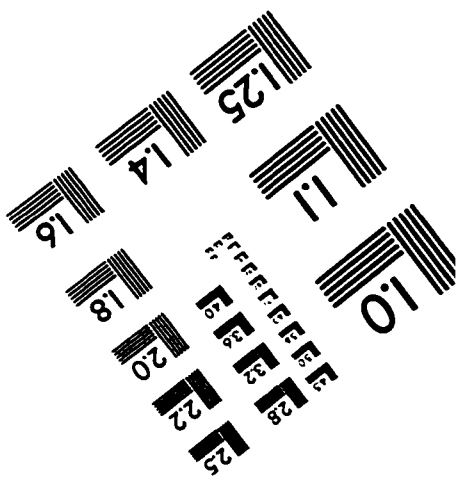












APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989