TEXAS A&M FAULT DISTURBANCE CONFERENCE, MARCH 1994

AUTOMATED FAULT ANALYSIS USING NEURAL NETWORK

M. Kezunovic, I. Rikalo — Texas A&M University

D. J. Sobajic — Electric Power Research Institute

C. W. Fromen, D. R. Sevcik — Houston Lighting & Power Company

ABSTRACT

The topic of this paper is the design of a Neural Network (NN) for power system fault detection and classification. The basic task of the NN is to classify new "unseen" fault patterns based on the fault examples provided during the training session. The source of the training and the testing data was EMTP simulation. The power system model used was a three source model with line couplings. Possible application of the proposed Neural Network Classifier would be as a part of an automated fault analysis solution. It is to be used as an aid to the system operators for fast characterization of the Digital Fault Recorder event files. The NN-based solution has been compared to an existing solution obtained using digital signal processing and expert system. It has been demonstrated that the NN-based solution is at least as selective as the one based on the expert system. Some major advantages in using NN-based solution where observed when system condition changes require further "tuning" of the automated analysis.

INTRODUCTION

Power system fault detection and classification are inherent parts of a fault analysis. The fault analysis is an important monitoring function since it provides both operators and protection engineers with information needed to assess consequences of a fault

occurrence. Operators need to confirm faulted sections before a restoration is attempted. Protection engineers need to analyze operation of relays, circuit breakers and communication equipment in order to assess equipment performance.

Automated fault detection and classification for fault analysis purposes can be implemented in a number of different ways. Most common approaches in the past were to use signal processing techniques [1]. Some of the most recent approaches suggest the use of a solution combining both signal processing and expert system techniques [2].

This paper introduces a new approach to the automated fault detection and classification implementation using neural network. A similar neural network algorithm was used in different applications [3, 4, 5]. This approach provides some potential computational advantages over the existing approaches. The NN-based solution described in this paper is compared with a solution combining expert system and signal processing techniques introduced by the authors earlier [6, 7].

The first part of the paper is related to the NN algorithm description. Test results generated using Electromagnetic Transient Program (EMTP) are presented next. Comparison between solutions based on NN and the one based on a combination of signal processing and expert system techniques is given at the end.

TYPICAL NEURAL NETWORK APPLICATIONS

A general, yet rigorous, definition of an neural network can be stated as follows [8]:

> "A neural network is a parallel, distributed information processing structure consisting of processing elements (which can possess a local memory and carry out localized information processing operations) interconnected together with

unidirectional signal channels called connections. Each processing element has a single output connection which branches ("fans out") into as many collateral connections as desired (each carrying the same signal - the processing element output signal). The processing element output signal can be of any mathematical type desired. All of the processing that goes on within each processing element must be completely local; i.e., it must depend only upon the current values of the input signal arriving at the processing element via impinging connections and upon values stored in the processing element's local memory."

The key elements of most NN descriptions are distributed representation, the local operations, and nonlinear processing. Neural networks are primarily used in situations where only a few decisions are required from a massive amount of data and situations where a complex nonlinear mapping must be learned. Main applications of the present day neural network computing include:

- functional approximation
- clustering
- data compression
- optimization
- topological mapping

Functional Approximation

Feedforward neural network can be looked upon as a functional mapping between two spaces, i.e. representation space (NN inputs) and interpretation space (NN output).

Evaluation of the given mapping is done by propagating the activations from input layer toward the output layer. The class of functions that can be approximated by a network is determined by the network topology and activation functions. A large number of interesting application-oriented tasks in functional approximation theory belong to categories of classification and control.

Currently, many neural network applications can be characterized as classification tasks. Most feedforward network models, using supervised learning techniques, have been developed explicitly for classification as a goal application.

The control task can be defined as follows: For a given device, a controller is to be designed to stabilize it or to keep it on a certain trajectory. There exists several NN applications in this field, but still the focus of research is on looking for new, more efficient learning paradigms.

Clustering

The clustering task can be formulated in the following way: A given set of objects, characterized by a fixed length vector of features, is to be partitioned into a certain number of clusters such that the variability of objects within each cluster is low, while variability between clusters is high. In other words the task is to find such a distribution of objects that will provide acceptable trade-off between maximization of intercluster variances and minimization of intracluster variances. Just as a reminder, there are $\frac{1}{c!}\sum_{i=1}^{c}\binom{c}{i}(-1)^{c-i}i^n$ partitions of n objects into c non-empty clusters [9]. Clustering is realized through feedforward/recurrent networks using unsupervised learning techniques, or combination of supervised and unsupervised learning.

Data Compression

Data compression is typically an unsupervised learning task. Data compression is a general form of clustering, where for a given set of data patterns, compact representation is sought. Each pattern is compressed to a pattern whose dimension is much lower than that of the original pattern, or groups of patterns are represented by corresponding prototypes.

Optimization

As in the case of functional approximation, optimization is a very general task. Neural networks had been used in solving some difficult combinatorial optimization problems (such as the traveling salesman problem). Propagation procedure for feedback networks can be viewed as a minimization of a certain global function of activations of all neurons of the network, and hence can be used for solving minimization (or generally optimization) problems.

Topological Mapping

This is another interesting application where neural networks are able to find a mapping of a continuous input space to the quantized output space so that topological properties such as neighborhood in the input space are preserved in the output space.

NEURAL NETWORK ALGORITHM DESCRIPTION

The neural network application presented in this paper belongs to the group of the clustering tasks. The algorithm proposed in this paper implements a supervised "follow the leader" approach and controls the clustering process by a threshold called the vigilance parameter ρ, and by the Euclidean metric function. This algorithm is presented schematically in Fig. 1 and Fig. 2, and it consists of an unsupervised clustering part and a supervised extraction part. In the unsupervised clustering, all training patterns are clustered using the fixed value of vigilance parameter ρ. Cluster geometries depend on the adopted self-organization model.

In this paper, cluster is defined as a hypersphere and ρ is the radius. After stable cluster formation occurs, the supervised part is executed, and all homogenous clusters (i.e., clusters containing only patterns for a particular type of fault) are stored in the memory.

These are removed from the training set for further processing. After that, the vigilance parameter is decreased and the procedure is reiterated for that reduced ρ. The training is completed when there are no fault patterns left or the vigilance parameter has reached very small value ε. As a result of the training procedure, input training set is mapped into a smaller number of clusters with different ρ, which is schematically illustrated in Fig. 3. Every cluster is defined as a hypersphere with center b and radius ρ. Mathematical description of the unsupervised clustering part is given in Appendix A. In Fig. 2, N denotes the number of input features, P the number of patterns in the training set, K the number of clusters, $x_i^{(p)}$ the ith feature of the input pattern p (i=1,...,N; p=1,...,P), b_{ki} the center of the cluster k (i=1,...,N; k=1,...,K), ρ the vigilance parameter and n_k the number of patterns that belong to the cluster k.

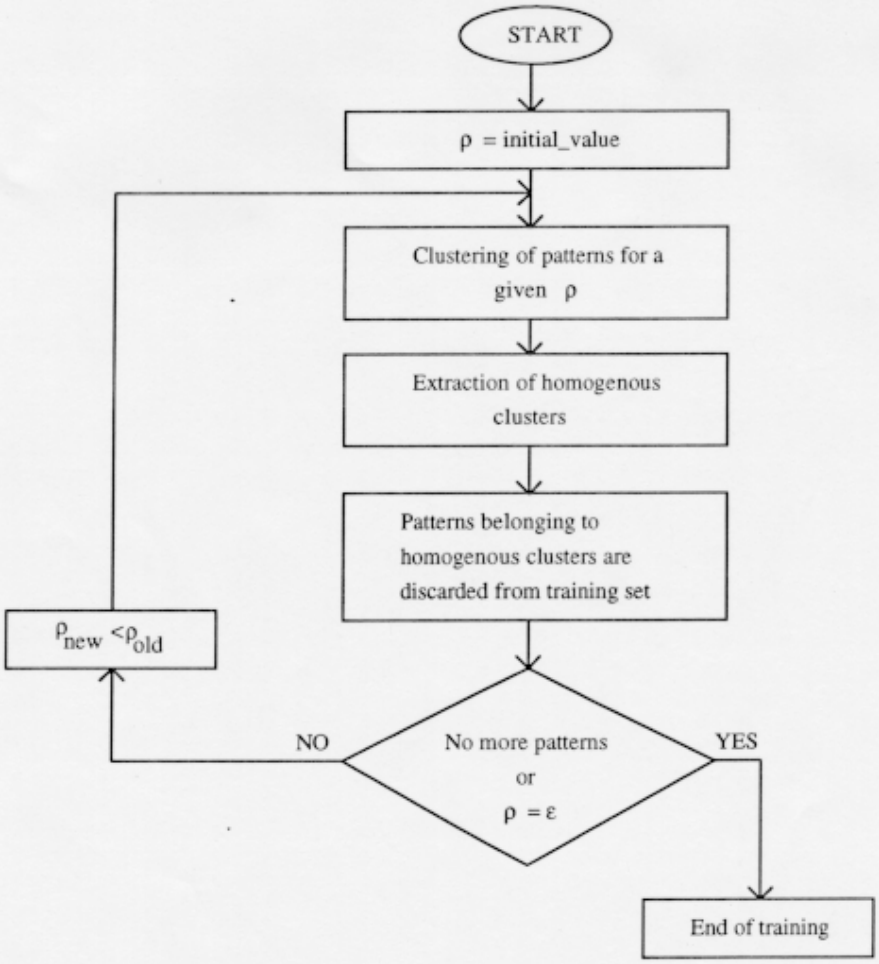

Figure 1. Block diagram of the supervised clustering algorithm

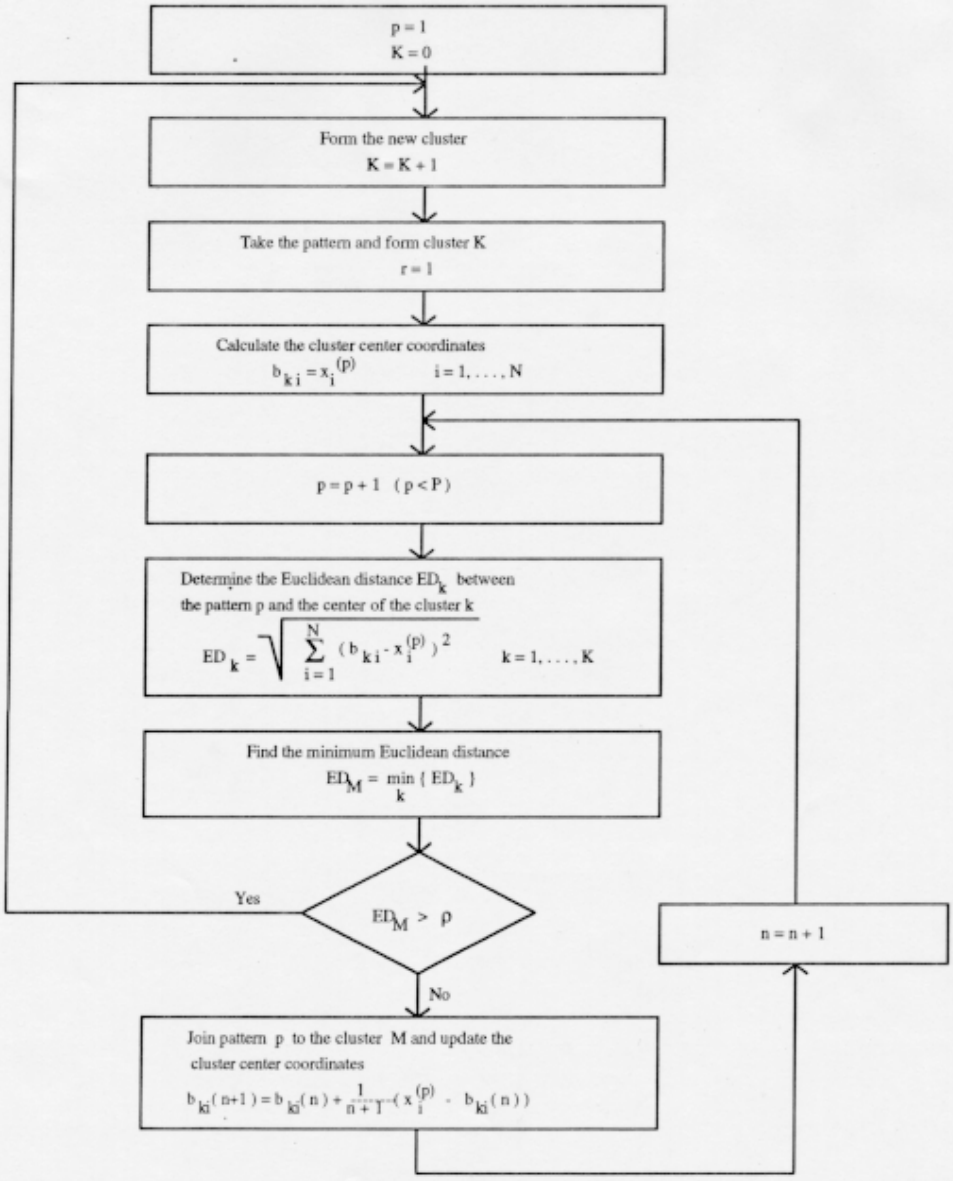

Figure 2. An illustration of a "follow the leader" unsupervised clustering algorithm

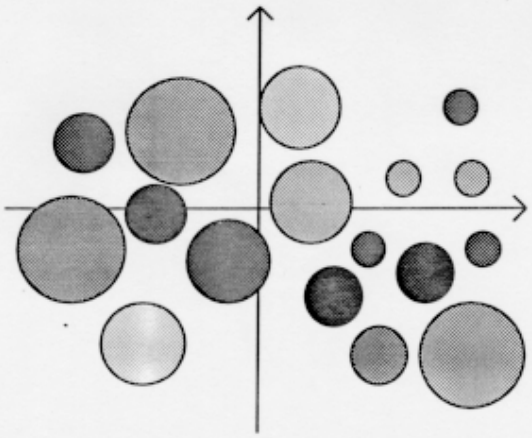

Figure 3. Schematic illustration of the outcome of the training process

EMTP SIMULATION

Fault patterns are generated using Electromagnetic Transient Program (EMTP) simulation [10]. A one line diagram of the modeled power system is given in Fig. 4. It represents part of an actual 161 kV power system with short transmission lines. The transmission line considered for simulation of fault events is the one between buses 2 and 3. It is fully transposed and 13.35 miles long. Data for the EMTP model of power system are given in the Appendix B.

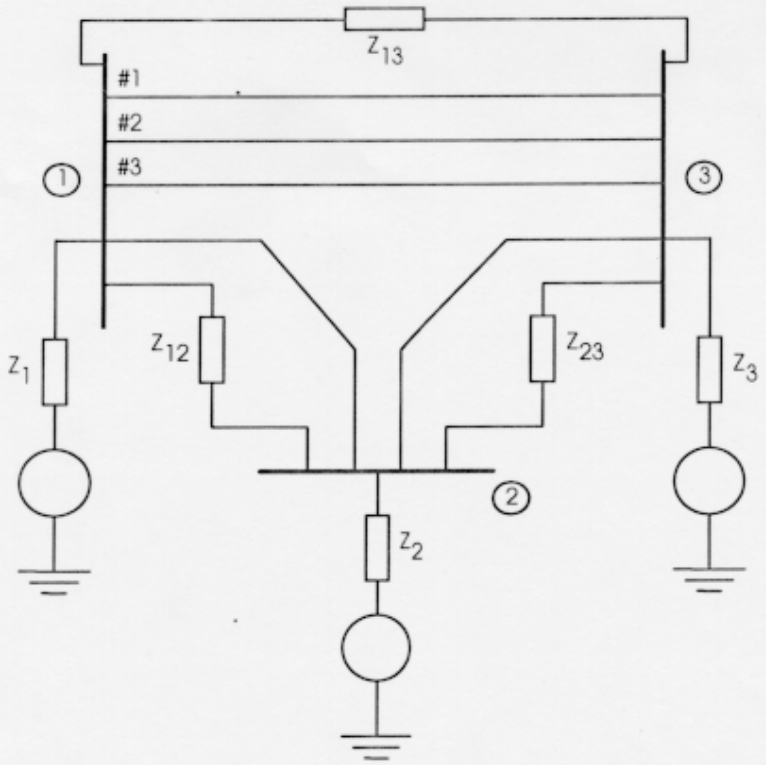

Figure 4. One line diagram of the model of the 161 kV power system

A number of EMTP simulations of various fault events were performed for generating fault patterns to be used for training and testing of the neural network. The maximum time of simulation was 50 ms, while the data sampling time step used was 0.5 ms (sampling frequency f_S= 2 kHz).

All 11 possible faults are simulated (a-g, b-g, c-g, a-b, a-c, b-c, a-b-g, a-c-g, b-c-g, a-b-c and a-b-c-g faults) and the following three parameters are varied within each simulation:

- fault location (0.0, 0.14, 0.80 and 1.0, where 1.0 corresponds to the whole length of the transmission line)

- fault resistance (0 Ω, 3 Ω, 6Ω and 50 Ω)
- incidence angle of the fault occurrence (0°, 45°, and 90°)

A total number of 619 fault patterns were generated in this way. Also 1 pattern labeled as normal state was generated using EMTP to represent the steady state (no fault state) of the power system.

FAULT CLASSIFICATION USING NEURAL NETWORK

Several different input data sets into the NN were considered during design and testing. These input data sets are summarized in Table I. Input data sets 1, 2 and 3 consist of both prefault and postfault samples, while input data sets 4, 5 and 6 contain only postfault samples. Since the proposed neural network algorithm contains no hidden layers (flat net) the network structure depends on the type of the input data set. The number of neurons in the input layer is determined with the length of the input vector (e.g., for the input data set 1, the number of neurons is 600, while for the input data set 5 that number is 99).

Table I. Neural network input data sets

Input data set 1	Input data set 2	Input data set 3
Length of fault patterns is 3 cycles (50 ms)	Length of fault patterns is 3 cycles (50 ms)	Length of fault patterns is 3 cycles (50 ms)
Fault pattern contains both voltage and current samples	Fault pattern contains only current samples	Fault pattern contains only voltage samples
Number of samples in every fault pattern is 600	Number of samples in every fault pattern is 300	Number of samples in every fault pattern is 300

Table I. Neural network input data sets (cont'd)

Input data set 4	Input data set 5	Input data set 6
Length of fault patterns is 1 cycle (16.67 ms)	Length of fault patterns is 1 cycle (16.67 ms)	Length of fault patterns is 1 cycle (16.67 ms)
Fault pattern contains both voltage and current samples	Fault pattern contains only current samples	Fault pattern contains only voltage samples
Number of samples in every fault pattern is 198	Number of samples in every fault pattern is 99	Number of samples in every fault pattern is 99

Examples of the wave forms used as an input data set for the neural network are given in Fig. 5 and Fig. 6.

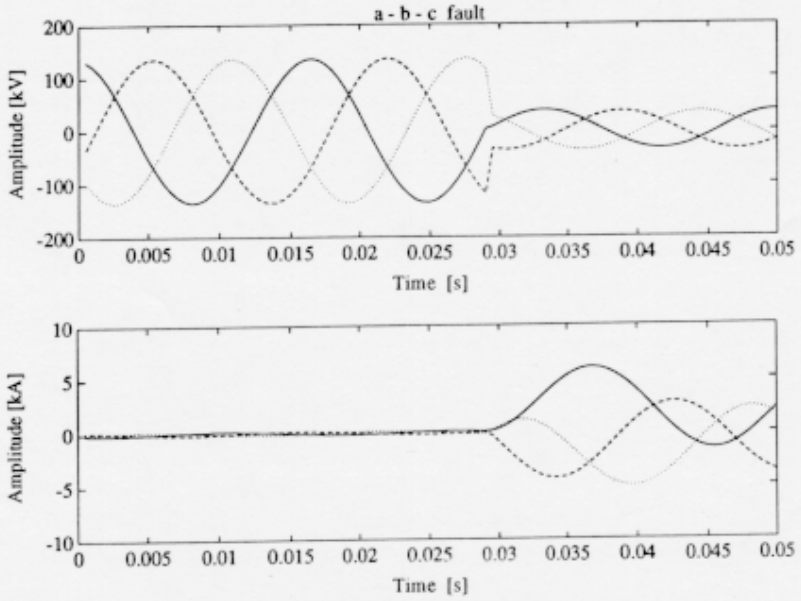

Figure 5a. Input data set 1

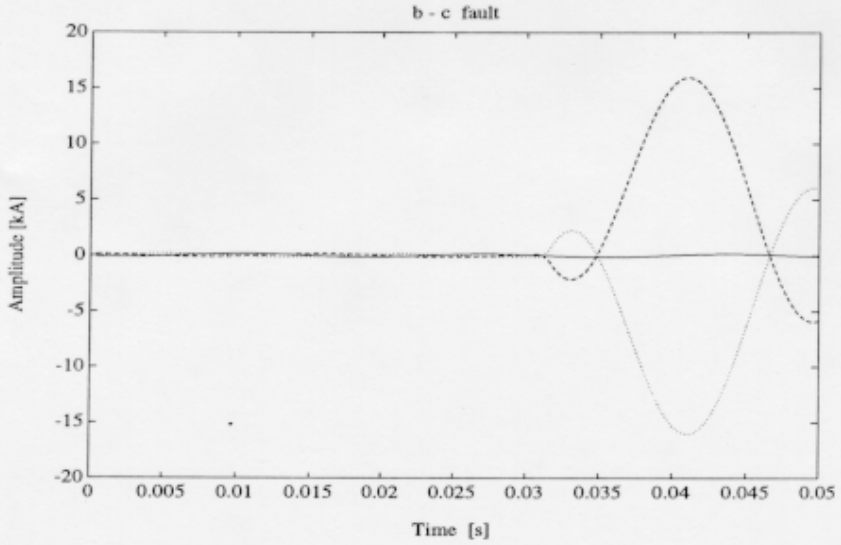

Figure 5b. Input data set 2

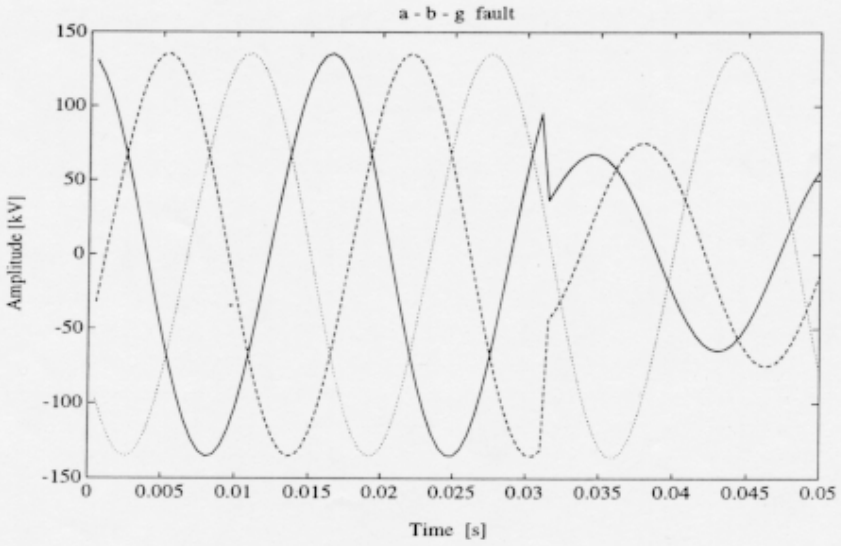

Figure 5c. Input data set 3

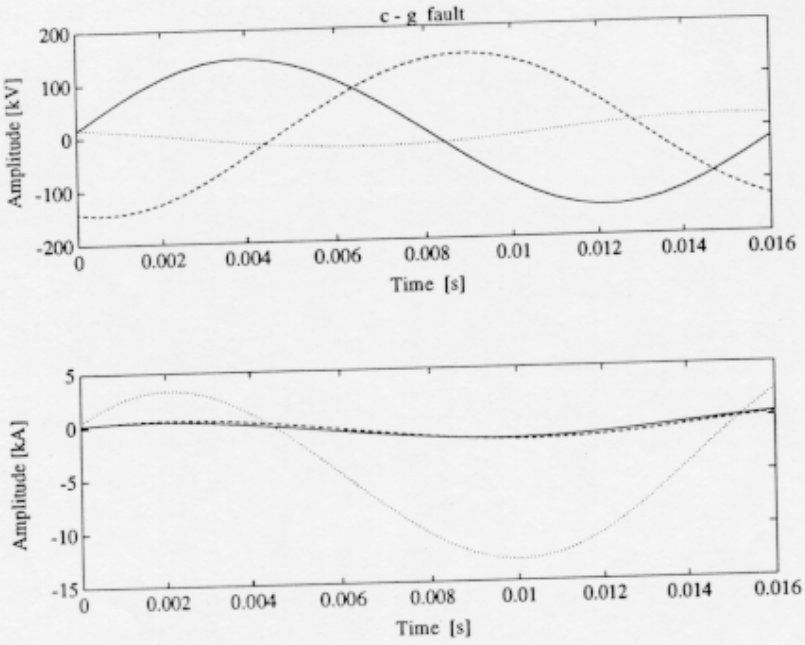

Figure 6a. Input data set 4

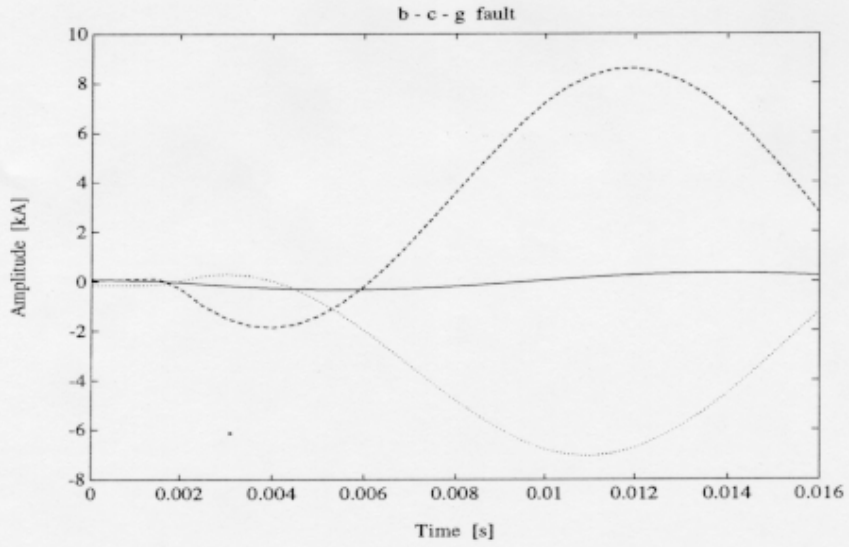

Figure 6b. Input data set 5

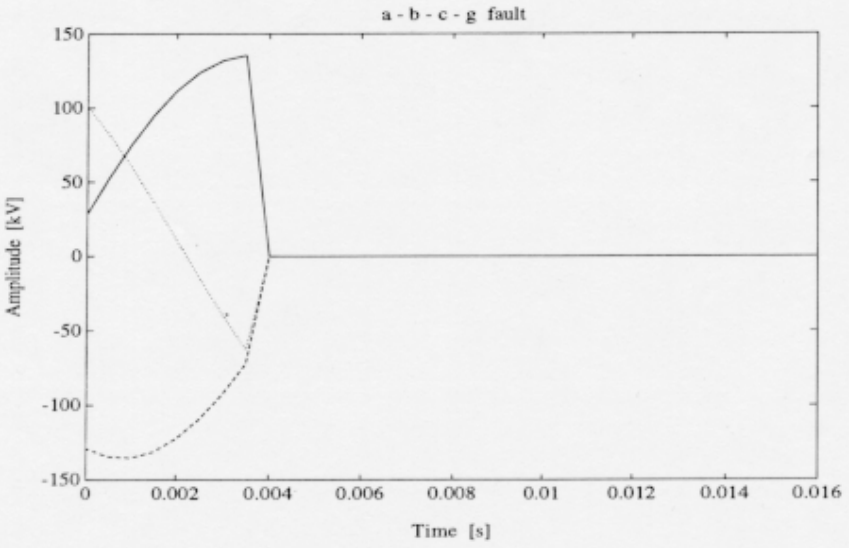

Figure 6c. Input data set 6

the total number of fault patterns used for training was 324. In addition to that, one steady state pattern was provided during the training. Distribution of the patterns according to the fault type is given below in Table II.

Table II. Distribution of Training Fault Patterns

Fault Type	Fault Location		Fault resistance		No. of patterns
	Local	Remote	Low	High	
a - g	6	18	16	8	24
b - g	12	24	24	12	36
c - g	12	24	24	12	36
a - b	12	18	20	10	30
a - c	12	21	22	11	33
b - c	12	21	22	11	33
a - b - g	10	20	24	6	30
a - c - g	10	20	24	6	30
b - c - g	10	20	24	6	30
a - b - c	8	14	14	8	22
a - b - c - g	8	12	18	2	20
Total	**112**	**212**	**232**	**92**	**324**

Results of training (clustering) for every input data set are given in Table III. The neural network algorithm is implemented in C programming language on the IBM RISC 6000 platform. Total time needed for neural network training in the most complex case (for input data set 1) was approximately 15 min. A little over 8 min. was needed to train neural network for input data set 5 or 6.

Class membership of each cluster is determined during the supervised part of training. A cluster is considered homogenous if it contains fault patterns of a particular type (a - g, b -

c - g, a - b - c, etc.). The following additional labels are given according to the cluster contents:

- if the cluster contains fault patterns only for fault locations 0.0 or 0.14 (where 1.0 denotes full length of the transmission line), the cluster is labeled with the word *local* (meaning that cluster represents faults that happened near the local bus)

- if the cluster contains patterns for other fault locations (i.e., 0.80 or 1.0), the cluster is labeled with the word *remote*.

- if the cluster contains only patterns representing faults with fault resistance equal 0 Ω, 3 Ω or 6 Ω, they are labeled with the word *low* (meaning low fault resistance).

- if the cluster contains only patterns representing faults with fault resistance of 50 Ω, they are labeled with the word *high* (meaning high fault resistance).

Hence, two valid cluster labels are:

1. **a - b** meaning the cluster contains fault patterns of the type a - b fault with mixed high and low fault resistance patterns as well as a - b fault patterns with different fault locations.
2. **b - c - g - remote - low** meaning the cluster contains fault patterns of the fault type b - c - g; fault resistance for this class is low resistance (0 Ω, 3 Ω or 6 Ω) and fault location is remote.

Table III. Training Results

Input data set	# of input patterns	Number of clusters
1	325	131
2	325	132
3	325	105
4	325	128
5	325	128
6	325	118

During the testing, NN was presented with 295 new fault patterns. Neural network never "saw" these patterns and the task was to classify new patterns based solely on the previous experience (i.e., using the information "learned" during the training).

Classification was based on the following logic:

- if new pattern "belongs" to the nearest cluster (i.e., Euclidean distance between pattern and cluster centroid was smaller than radius of that cluster), pattern inherited class membership of that cluster
- if new pattern "does not belong" to the nearest cluster (i.e., Euclidean distance between pattern and cluster centroid was greater than radius of that cluster), then pattern was assigned label according to the class membership of the nearest three clusters (3 nearest neighbors).

The results are summarized according to the NN input data sets in Table IV. Considering pairs of input data sets 1 and 2, and 4 and 5 it can be noticed that NN performance is not

deteriorating if only samples of phase currents are taken as pattern features. On the other hand, computational effort and time are greatly reduced if the voltage samples are disregarded.

Table IV. Neural Network Classification Results

Input Data Set	Correct Classification			Incorrect Classification			Correct [%]
	inside the cluster	outside the cluster	total	inside the cluster	outside the cluster	total	
1	214	68	282	7	6	13	95.59
2	218	65	283	7	5	12	95.93
3	195	86	281	1	13	14	95.25
4	209	67	276	11	8	19	93.56
5	207	71	278	11	8	19	93.56
6	199	80	279	1	15	16	94.58

All incorrectly classified cases are listed in tables in the Appendix C.

EVALUATION OF THE NEURAL NETWORK BASED APPROACH VS. EXPERT SYSTEM BASED APPROACH

Expert systems and neural systems are two complementary technologies in the field of Artificial Intelligence (AI). An expert system is constructed by the contribution of one or more human experts. Human experts supply their own tested methods and knowledge to give the computer the basis for appropriate answers for a specific problem. When new

expertise stops coming , the domain knowledge and methods for handling this knowledge stop growing. On the other hand, a NN system learns directly by interacting with the application. Given enough time and experience or training, the NN system will ideally learn everything about the application. It will be able to learn what is presently not known by any of the experts.

Below are presented some of the beneficial characteristics of NN systems compared to the conventional Expert Systems [11]:

- Generalization
- Graceful degradation
- Adaptivity and learning
- Parallelism

Generalization

Inferences of any logic based system are performed on syntactically completely defined strings of symbols, and there is no concept of similarity between the symbols. That means, an arbitrary small change in input leads to completely different inferences. Since no two real-world situations are exactly the same, some type of continuous representation and inference system is required, with similarity a metrics such that for similar inputs or situations, outputs or inferences are similar, too. This is an inherent property of neural networks.

Graceful Degradation

In the case of inaccurate or incomplete data, a logic based inference system, because it lacks similarity concept, may lead to results that differ drastically from those corresponding to exact data. This is typical behavior of the rule-based expert systems. Their encoded knowledge in the form of the rules is specified during the design period. If

some of the inputs are inaccurate or missing, the expert system reaction may be completely incorrect because the inference chain cannot be used in the same way as with exact data.

This is not the case for neural networks. If a small proportion of the input data is missing or distorted, performance deteriorates only slightly. Performance deterioration is proportional to the extent of data inaccuracy or incompleteness.

Adaptivity and Learning

The main property of logic based methods is their rigidness. The theory of logic is concerned with the task of drawing sound inferences from a given set of logical statements rather than with a question of how this set is generated. Acquisition of general statements from individual cases in real-world context is an unsolved problem. The maintenance of expert systems is yet another problem. If the environment changes slightly, there are no easy methods of incremental adaptation to such a modified environment. This is an important handicap since few tasks are invariant in time.

On the contrary, in the neural network systems, learning by example is the only way of encoding knowledge into them. Also, adaptation is straightforward, because there is no strict difference between learning as an initial knowledge acquisition, and adaptation as a process of maintenance of acquired knowledge in changing environment conditions.

Parallelism

In the existing logic based systems, inference procedures are mostly sequential. Sequential processing of features separately can lead to lengthy procedures or to erroneous dead ends from which recovery can not be readily achieved. This contrasts with the inherent parallelism of virtually all neural networks algorithms. In most cases, all NN units can be updated simultaneously.

NEURAL NETWORK VS. EXPERT SYSTEM COMPARISON

Results of fault classification using neural network are given in the previous paragraph, and they show the behavior of the NN in absolute terms. The existing rule-based Expert System (ES) [2, 6, 7] has been chosen to make relative comparison between two different approaches for solving the same problem (i.e. detection and classification of faults in power systems). A functional outline of the ES is given in Fig. 7.

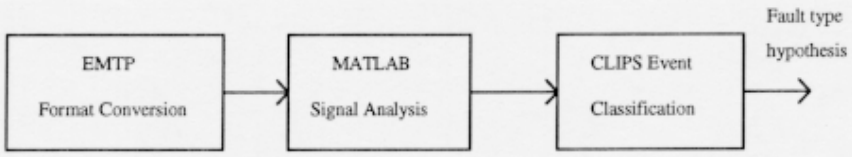

Figure 7. Expert System block diagram

Input fault events were the same as those used for testing of the NN. Signal processing techniques (using MATLAB built-in functions [12]) are used to process voltage and current wave forms in order to determine signal parameters. These parameters are then used to classify a given event. An expert system is implemented using CLIPS for this purpose [13]. Table V shows the parameters used for determining the type of the fault. This table consists of voltage and current patterns that describe fault type. In the case that given relations are not satisfied, the event is not a fault.

Table V. Behavioral Patterns of the Basic Parameters

Event Type	0 Seq. Current	Faulted Current	Unfaulted Current	0 Seq. Voltage	Faulted Voltage	Unfaulted Voltage	Line Voltage
a - g	$I_0 > \frac{I_a}{5}$	$I_a > 1.4 I_p$	$I_{b,c} < \frac{I_a}{3}$	$V_0 > \frac{V_n}{25}$	$V_a < \frac{9V_n}{10}$	$V_{b,c} > \frac{96V_n}{100}$	$V_{ab} = V_{ca}$
a - b	$I_0 < \frac{I_a}{100}$	$I_a > 1.4 I_p$ $I_b > 10 I_p$	$I_c < \frac{I_a}{10}$	$V_0 < \frac{V_n}{100}$	$V_a < \frac{8V_n}{10}$ $V_b < \frac{7V_n}{10}$	$V_c > \frac{99V_n}{100}$	$V_{ab} < \frac{8V_{ln}}{10}$
a - b - g	$I_0 > \frac{I_a}{10}$	$I_a > 1.4 I_p$ $I_b > 10 I_p$	$I_c < \frac{I_a}{10}$	$V_0 > \frac{V_n}{20}$	$V_a < \frac{8V_n}{10}$ $V_b < \frac{8V_n}{10}$	$V_c > \frac{98V_n}{100}$	$V_{ab} < \frac{8V_{ln}}{10}$
a - b - c	$I_0 < \frac{3I_b}{100}$	$I_f > 10 I_p$		$V_0 < \frac{V_n}{100}$	$V_f < \frac{8V_n}{10}$		$V_{ll} < \frac{8V_{ln}}{10}$
a-b-c-g	$I_0 < \frac{3I_b}{100}$	$I_f > 10 I_p$		$V_0 < \frac{V_n}{100}$	$V_f < \frac{8V_n}{10}$		$V_{ll} < \frac{8V_{ln}}{10}$

The task of the Expert System was to classify a set of 295 fault events. The fault events were generated using EMTP simulation and are the same as those used for NN input in data set 1. The expert system requires both prefault and postfault samples in order to calculate network parameters, so input data sets 4, 5 and 6 which can be used by the neural net classifier are in the expert system case inappropriate.

The result of the classification was:

- for 288 fault types classification was correct
- for 7 fault types classification was incorrect,

In other words the expert system was correct in 97.64% cases. Fault patterns that are classified incorrectly are listed in Table VI.

Table VI. Patterns Incorrectly Classified by Expert System

Simulated Fault Type	*ES Classification*
a - b fault, fault location 1.0, $R_f = 50\ \Omega$	none
a - c fault, fault location 1.0, $R_f = 50\ \Omega$	none
a - c fault, fault location 1.0, $R_f = 50\ \Omega$	none
b - c fault, fault location 1.0, $R_f = 50\ \Omega$	none
a - g fault, fault location 0.8, $R_f = 50\ \Omega$	none
b - g fault, fault location 1.0, $R_f = 50\ \Omega$	none
c - g fault, fault location 1.0, $R_f = 50\ \Omega$	none

The main reason for expert system erroneous classification of these events was unexpected current wave forms for remote high fault resistance fault patterns. Expert system rules are based on a general assumption that in the case of a fault, faulted phase currents are larger than respective prefault currents, which was not the case for fault patterns below. Fig. 8 gives one of those events as an example. This is a phase **a** to phase **b** fault, where phase **b** current dropped after the fault.

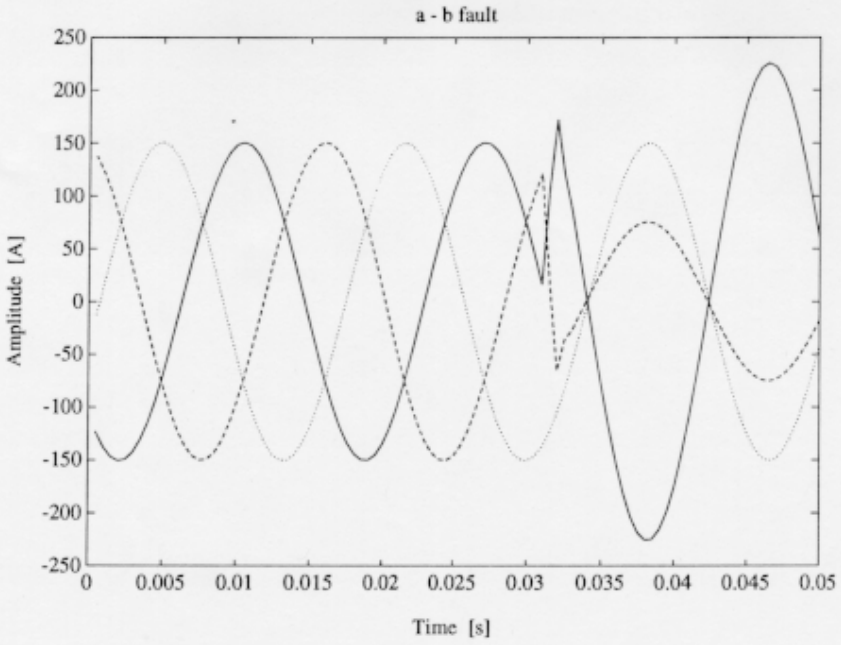

Figure 8. Phase a to phase b high impedance remote fault

CONCLUSIONS

This paper shows that the proposed neural network is successful in power system fault detection and classification. Compared to the existing expert system based solution for the same task, neural network proved to be as selective and at the same time computationally less demanding. The obtained results indicate that supervised clustering technology can be used for accurate pattern classification. The proposed NN algorithm is easy to implement and contains no hidden layers (flat network). One advantage of using flat network is that

both supervised and unsupervised learning can be carried out with the same network structure.

REFERENCES

[1] A. Wisniewski, "*Digital High-Speed Calculation of the Distorted Signal Fundamental Component*", IEE Proceeding, Vol. 137, Part C, No. 1, January 1990.

[2] M. Kezunovic, et al., "*Digital Fault Recorder File Classification and Analysis Using Expert System and Signal Processing Techniques*", Texas A&M Relay Conference, College Station, Texas, April 1993.

[3] D. J. Sobajic, Y. H. Pao, "*On-Line Transient Stability Evaluation by Associative Dichotomous Classification*", 11th PSCC, Avignon, France, September 1993.

[4] M. Djukanovic, D. J. Sobajic, Y. H. Pao, "*Artificial Neural Network Based Identification of Dynamic Equivalents*", Electric Power Systems Research, 24 (1992) p.39-48

[5] T. Dalstein, et. al., "*Neural Network Approach to Fault Direction Identification in Electric Power Systems*", North American Power Symposium 1993, October, Washington, D.C.

[6] M. Kezunovic, et al., "*An Expert System for Transmission Substation Event Analysis*", IEEE Transactions on Power Delivery, Vol. 8, No. 4, pp 1942-1949, October 1993.

[7] M. Kezunovic, et al., "*An Expert System for DFR File Classification and Analysis*", 4th Symposium on Expert System Applications to Power Systems, Melbourne, Australia, February 1993.

[8] R. Hecht-Nielsen, "*Applications of counterpropagation networks*", Neural Networks, Vol. 1, pg. 131-140, 1988.

[9] R. O. Duda, P. E. Hart, "*Pattern Classification and Scene Analysis*", John Wiley & Sons, Inc., New York, 1973.

[10] *Electromagnetic Transient Program - Workbook*, Electric Power Research Institute, Palo Alto, California, September 1986.

[11] T. Hrycej, "*Modular Learning in Neural Networks*", John Wiley & Sons, Inc., New York, 1992

[12] PC - MATLAB Users Guide, The MathWorks, Inc., South Natick, Massachusetts, May 1989.

[13] CLIPS Reference Manual, Artificial Intelligence Section, Johnson Space Center, Houston, Texas, September 1987.

Appendix A.

PROBLEM:

Given a set of P (p=1, 2, ..., P) patterns $\underline{x}^{(p)}$

$$\underline{x}^{(p)} = \left[x_1^{(p)}, x_2^{(p)}, \cdots, x_N^{(p)}\right]^T$$

Find a family of (hyper)spherical clusters C_k (k = 1, 2, ..., K) so that

$$C_k = Set\left\{\underline{x}^{(p)} \middle| \left(\underline{x}^{(p)} - \underline{b}_k\right)^T \left(\underline{x}^{(p)} - \underline{b}_k\right) \le \rho^2\right\}$$

Sphere radius ρ is given, and is assumed to be the same for all clusters.

Each cluster is defined with its center $\underline{b}_k$ and radius ρ

$$C_k = C_k(\underline{b}_k, \rho)$$

so what remains to be found is a set of vectors (centroids) $\underline{b}_k$ for k = 1, 2, ..., K and K.

Following algorithm implements unsupervised learning technique which is a neural-net implementation of the ISODATA clustering algorithm.

INITIALIZATION RUN

Reorder patterns in the data set.

STEP 0.

1. Find a center $\underline{b}$ of the entire data set

$$\underline{b} = \frac{1}{P}\sum_{p=1}^{P} \underline{x}^{(p)}$$

2. Find distances between each pattern $\underline{x}^{(p)}$ and the center $\underline{b}$ and rank them in an increasing order.

STEP 1.

Form cluster 1

$$\underline{b}_1(1) = \underline{x}^{(1)}$$

Meaning cluster C_1 with centroid $\underline{b}_1$ contains 1 pattern.

STEP 2.

If

$$\left(\underline{x}^{(2)} - \underline{b}_1\right)^T \left(\underline{x}^{(2)} - \underline{b}_1\right) \leq \rho^2$$

adapt $\underline{b}_1$ as

$$\underline{b}_1(2) = \underline{b}_1(1) + \frac{1}{2}\left(\underline{x}^{(2)} - \underline{b}_1(1)\right)$$

If

$$\left(\underline{x}^{(2)} - \underline{b}_1\right)^T \left(\underline{x}^{(2)} - \underline{b}_1\right) > \rho^2$$

form cluster 2 as

$$\underline{b}_2(1) = \underline{x}^{(2)}$$

In doing so after presenting q < P patterns the situation is as follows:

m - clusters exists, their centroids $\underline{b}_m$ are known and we know how many patterns belong to each cluster n_m.

$$\begin{array}{ll} C_1 & \underline{b}_1(n_1) \\ C_2 & \underline{b}_2(n_2) \\ \vdots & \vdots \\ C_m & \underline{b}_m(n_m) \end{array}$$

Clearly,

$$\sum_{j=1}^{m} n_j = q$$

So, when we present next pattern q+1 we first allocate the closest cluster τ, by

$$\min_j \left\{ \left(\underline{x}^{(q+1)} - \underline{b}_j \right)^T \left(\underline{x}^{(q+1)} - \underline{b}_j \right) \right\} = r_\tau^2$$

and then compare r_τ^2 *and* ρ^2

If $r_\tau^2 \le \rho^2$

then adapt cluster as

$$\underline{b}_\tau(n_\tau + 1) = \underline{b}_\tau(n_\tau) + \frac{1}{n_\tau + 1}\left(\underline{x}^{(q+1)} - \underline{b}_\tau(n_\tau) \right)$$

If $r_\tau^2 > \rho^2$

then form new cluster as

$$\underline{b}_{m+1}(1) = \underline{x}^{(q+1)}$$

This procedure is repeated until the entire set of patterns is processed once.

STABILIZATION RUN

We present every pattern, $\underline{x}^{(p)}$, again. Let say presently pattern p belongs to cluster C_k. First, we find the shortest distance between $\underline{x}^{(p)}$ and all existing centroids $\underline{b}_j$

$$\min_j \left\{ \left(\underline{x}^{(p)} - \underline{b}_j \right)^T \left(\underline{x}^{(p)} - \underline{b}_j \right) \right\} = r_\tau^2$$

Second

1. If $\tau = k$ *and* $r_\tau^2 \le \rho^2$

   then no learning occurs; check next pattern p + 1.

2. If $\tau \ne k$ *and* $r_\tau^2 \le \rho^2$

   then adapt $\underline{b}_\tau$ and $\underline{b}_k$ as

   $$\underline{b}_\tau(n_\tau + 1) = \underline{b}_\tau(n_\tau) + \frac{1}{n_\tau + 1}\left(\underline{x}^{(p)} - \underline{b}_\tau(n_\tau) \right)$$

$$\underline{b}_k(n_k - 1) = \underline{b}_k(n_k) - \frac{1}{n_k - 1}\left(\underline{x}^{(p)} - \underline{b}_k(n_k)\right) \quad for \quad n_k > 1$$

$$Cluster \quad \underline{b}_k(n_k - 1) \quad is\ discarded \qquad for \quad n_k = 1$$

3. If $r_\tau^2 > \rho^2$

   form new cluster C_m

   $\underline{b}_m(1) = \underline{x}^{(p)}$

   and adapt "previous" centroid $\underline{b}_k$ as

$$\underline{b}_k(n_k - 1) = \underline{b}_k(n_k) - \frac{1}{n_k - 1}\left(\underline{x}^{(p)} - \underline{b}_k(n_k)\right) \quad for \quad n_k > 1$$

$$Cluster \quad \underline{b}_k(n_k - 1) \quad is\ discarded \qquad for \quad n_k = 1$$

NOTES:

1. Initialization run goes only once.

2. Stabilization run can repeat several times. The end is when no patterns change their clusters in two successive stabilization runs.

3. To run this algorithm one has to keep accounts of:
   - where each pattern $\underline{x}^{(p)}$ belongs.
   - how many patterns are in each cluster
   - centroid of each cluster

4. The main difficulty is to have a good estimate of radius ρ. This is unsupervised learning so "we don't know" what the patterns say: In other words we don't know their class membership before hand. Once learning is done we have to attach some meaning to each cluster. This may require running the learning process several times for different ρ until we are satisfied with clustering outcome.

5. In consulting (testing) we simply take new pattern $\underline{x}$ and find

$$\min_j \left\{ (\underline{x} - \underline{b}_j)^T (\underline{x} - \underline{b}_j) \right\} = r_s^2$$

if $r_s^2 \leq \rho^2$

we will attach the class membership of the cluster C_s to the pattern $\underline{x}$

if $r_s^2 > \rho^2$

we may still do the same thing but with greater risk or reject to classify $\underline{x}$.

6. Hints:

- Normalize the data set so that each feature of $\underline{x}$, namely x_i is scaled between -1 and +1, or
- Normalize every pattern so that $\left\| \underline{x}^{(p)} \right\| = 1$

Appendix B.

Table B - I. Source Impedances

Bus Name	Per-Unit Value		Actual Value [Ω]
1	Z_0	0.58 + j 6.32	1.503 + j 16.382
	Z_1	0.58 + j 11.41	1.503 + j 29.576
2	Z_0	0.07 + j 1.07	0.181 + j 2.774
	Z_1	0.04 + j 0.73	1.1044 + j 1.892
3	Z_0	0.75 + j 407	1.944 + j 10.550
	Z_1	0.31 + j 3.04	0.804 + j 7.880

Table B - II. System Equivalents

Bus Name	Per-Unit Value		Actual Value [Ω]
1 - 3	Z_0	119.69 + j 188.93	310.248 + j 489.725
	Z_1	1.80 + j 11.44	4.666 + j 29.654
2 - 3	Z_0	∞	∞
	Z_1	12.58 + j 74.00	32.609 + j 191.815
1 - 2	Z_0	39.79 + j 100.63	103.140 + j 260.843
	Z_1	2.75 + j 18.32	7.128 + j 47.487

Table B - III. Self Impedances of Transmission Lines

Bus Name	Per-Unit Value		Actual Value [Ω]
1 - 3	Z_0	8.94 + j 28.34	23.1734 + j 73.4001
	Z_1	1.52 + j 9.00	3.9400 + j 23.4844
1 - 3	Z_0	8.52 + j 29.23	22.0847 + j 75.7071
	Z_1	1.38 + j 8.80	3.5771 + j 76.1300
1 - 3	Z_0	8.40 + j 29.37	21.7730 + j 76.1300
	Z_1	1.34 + j 8.73	3.4734 + j 22.6200
1 - 2	Z_0	8.42 + j 26.74	21.8255 + j 69.3128
	Z_1	1.50 + j 8.47	3.8882 + j 21.9551
2 - 3	Z_0	3.67 + j 12.38	9.5130 + j 32.0902
	Z_1	0.67 + j 3.92	1.7367 + j 10.1610

Appendix C.

Table 1. Incorrectly classified cases for input data set 1.

No.	Actual simulated fault	Neural net classification
1.	a - b - c fault, fault location 1.0, $R_f = 0$	a - c - g fault, remote, low impedance
2.	a -b -c -g fault,fault location 0.14,$R_f = 0$	a - b - c fault, local, low impedance
3.	a - b - g fault, fault location 0.0, $R_f = 50$	a - b fault, local, low impedance
4.	a - b - g fault, fault location 1.0, $R_f = 3$	a - b fault, remote, low impedance
5.	a - b - g fault, fault location 0.79, $R_f = 3$	a - b fault, remote, low impedance
6.	a - b - g fault,fault location 0.79,$R_f = 50$	a - b fault, remote, low impedance
7.	a - b - g fault, fault location 0.81, $R_f = 3$	a - b fault, remote, low impedance
8.	a - b - g fault,fault location 0.81,$R_f = 50$	a - b fault, remote, low impedance
9.	b - c fault, fault location 1.0, $R_f = 0$	b - c - g fault, remote, low impedance
10.	b - c - g fault, fault location 0.0, $R_f = 50$	b - c fault, local, low impedance
11.	a - b - g fault, fault location 0.79, $R_f = 6$	a - b fault, remote, low impedance
12.	a - b - g fault, fault location 0.81, $R_f = 6$	a - b fault, remote, low impedance
13.	b - g fault, fault location 1.0, $R_f = 6$	a - b fault, remote, low impedance

Table 2. Incorrectly classified cases for input data set 2.

No.	Actual simulated fault	Neural net classification
1.	a -b -c -g fault,fault location 0.14,R_f =0	a - b - c fault, local, low impedance
2.	a - b - g fault, fault location 0.0, R_f =50	a - b fault, local, low impedance
3.	a - b - g fault, fault location 1.0, R_f = 3	a - b fault, remote, low impedance
4.	a - b - g fault, fault location 0.79, R_f =3	a - b fault, remote, low impedance
5.	a - b - g fault,fault location 0.79,R_f =50	a - b fault, remote, low impedance
6.	a - b - g fault, fault location 0.81, R_f =3	a - b fault, remote, low impedance
7.	a - b - g fault,fault location 0.81,R_f =50	a - b fault, remote, low impedance
8.	b - c fault, fault location 1.0, R_f = 0	b - c - g fault, remote, low impedance
9.	b - c - g fault, fault location 0.0, R_f =50	b - c fault, local, low impedance
10.	a - b - g fault, fault location 0.79, R_f =6	a - b fault, remote, low impedance
11.	a - b - g fault, fault location 0.81, R_f =6	a - b fault, remote, low impedance
12.	b - g fault, fault location 1.0, R_f =6	a - b fault, remote, low impedance

Table 3. Incorrectly classified cases for input data set 3.

No.	Actual simulated fault	Neural net classification
1.	a - b fault, fault location 1.0, $R_f = 50$	b - g fault, remote, high impedance
2.	a - b - g fault, fault location 0.0, $R_f = 50$	a - b fault, local, low impedance
3.	a - b - g fault,fault location 0.79,$R_f = 50$	a - b fault, remote, low impedance
4.	a - b - g fault,fault location 0.81,$R_f = 50$	a - b fault, remote, low impedance
5.	a - c - g fault, fault location 0.0, $R_f = 50$	a - c fault, local, low impedance
6.	a - c - g fault,fault location 0.79,$R_f = 50$	a - c fault, remote, low impedance
7.	a - c - g fault,fault location 0.81,$R_f = 50$	a - c fault, remote, low impedance
8.	b - c - g fault, fault location 0.0, $R_f = 50$	b - c fault, local, low impedance
9.	b - c - g fault, fault location 1.0, $R_f = 3$	b - c fault, remote, low impedance
10.	b - c - g fault,fault location 0.79,$R_f = 50$	b - c fault, remote, low impedance
11.	b - c - g fault, fault location 0.81, $R_f = 3$	b - c fault, remote, low impedance
12.	b - g fault, fault location 1.0, $R_f = 50$	normal - state (no fault state)
13.	c - g fault, fault location 1.0, $R_f = 50$	normal - state (no fault state)
14.	a - c - g fault, fault location 0.0, $R_f = 6$	a - c fault, local, low impedance

Table 4. Incorrectly classified cases for input data set 4.

No.	Actual simulated fault	Neural net classification
1.	a - b - g fault, fault location 0.0, R_f =3	a - b fault, local, low impedance
2.	a - b - g fault, fault location 0.0, R_f =50	a - b fault, local, low impedance
3.	a - b - g fault, fault location 1.0, R_f = 3	a - b fault, remote, low impedance
4.	a - b - g fault, fault location 0.14, R_f =3	a - b fault, local, low impedance
5.	a - b - g fault,fault location 0.79,R_f = 3	a - b fault, remote, low impedance
6.	a - b - g fault,fault location 0.79,R_f =50	a - b fault, remote, low impedance
7.	a - b - g fault,fault location 0.81,R_f = 3	a - b fault, remote, low impedance
8.	a - b - g fault,fault location 0.81,R_f =50	a - b fault, remote, low impedance
9.	b - c fault, fault location 1.0, R_f = 0	b - c - g fault, remote, low impedance
10.	b - c - g fault, fault location 0.0, R_f =50	b - c fault, local, low impedance
11.	b - c - g fault, fault location 0.14, R_f =3	b - c fault, local, low impedance
12.	a - b fault, fault location 0.14, R_f = 6	a - b - g fault, remote, high impedance
13.	a - b - g fault,fault location 0.0, R_f = 6	a - b fault, local, low impedance
14.	a - b - g fault,fault location 0.79,R_f =6	a - b fault, remote, low impedance
15.	a - b - g fault,fault location 0.81,R_f = 6	a - b fault, remote, low impedance
16.	a - c - g fault, fault location 0.0, R_f = 6	a - c fault, local, low impedance
17.	b - c fault, fault location 0.14, R_f = 6	b - c - g fault, remote, high impedance
18.	b - c fault, fault location 0.14, R_f = 6	b - c - g fault, remote, high impedance
19.	b - c - g fault, fault location 0.0, R_f = 6	b - c fault, local, low impedance

Table 5. Incorrectly classified cases for input data set 5.

No.	Actual simulated fault	Neural net classification
1.	a - b - g fault, fault location 0.0, $R_f = 3$	a - b fault, local, low impedance
2.	a - b - g fault, fault location 0.0, $R_f = 50$	a - b fault, local, low impedance
3.	a - b - g fault, fault location 1.0, $R_f = 3$	a - b fault, remote, low impedance
4.	a - b - g fault, fault location 0.14, $R_f = 3$	a - b fault, local, low impedance
5.	a - b - g fault, fault location 0.79, $R_f = 3$	a - b fault, remote, low impedance
6.	a - b - g fault, fault location 0.79, $R_f = 50$	a - b fault, remote, low impedance
7.	a - b - g fault, fault location 0.81, $R_f = 3$	a - b fault, remote, low impedance
8.	a - b - g fault, fault location 0.81, $R_f = 50$	a - b fault, remote, low impedance
9.	b - c fault, fault location 1.0, $R_f = 0$	b - c - g fault, remote, low impedance
10.	b - c - g fault, fault location 0.0, $R_f = 50$	b - c fault, local, low impedance
11.	b - c - g fault, fault location 0.14, $R_f = 3$	b - c fault, local, low impedance
12.	a - b fault, fault location 0.14, $R_f = 6$	a - b - g fault, remote, high impedance
13.	a - b - g fault, fault location 0.0, $R_f = 6$	a - b fault, local, low impedance
14.	a - b - g fault, fault location 0.79, $R_f = 6$	a - b fault, remote, low impedance
15.	a - b - g fault, fault location 0.81, $R_f = 6$	a - b fault, remote, low impedance
16.	a - c - g fault, fault location 0.0, $R_f = 6$	a - c fault, local, low impedance
17.	a - c - g fault, fault location 0.0, $R_f = 6$	a - c fault, local, low impedance
18.	b - c fault, fault location 0.14, $R_f = 6$	b - c - g fault, remote, high impedance
19.	b - c - g fault, fault location 0.0, $R_f = 6$	b - c fault, local, low impedance

Table 6. Incorrectly classified cases for input data set 6.

No.	Actual simulated fault	Neural net classification
1.	a - b fault, fault location 1.0, $R_f = 50$	a - g fault, remote, high impedance
2.	a - b fault, fault location 0.805, $R_f = 50$	a - g fault, remote, high impedance
3.	a - b - g fault, fault location 0.0, $R_f = 50$	a - b fault, local, low impedance
4.	a - b - g fault, fault location 1.0, $R_f = 3$	a - b fault, remote, low impedance
5.	a - b - g fault,fault location 0.79,$R_f = 50$	a - b fault, remote, low impedance
6.	a - b - g fault,fault location 0.81,$R_f = 50$	a - b fault, remote, low impedance
7.	a - c - g fault, fault location 0.0, $R_f = 50$	a - c fault, local, low impedance
8.	a - c - g fault,fault location 0.79,$R_f = 50$	a - c fault, remote, low impedance
9.	a - c - g fault,fault location 0.81,$R_f = 50$	a - c fault, remote, low impedance
10.	b - c - g fault, fault location 0.0, $R_f = 50$	b - c fault, local, low impedance
11.	b - c - g fault, fault location 1.0, $R_f = 3$	b - c fault, remote, low impedance
12.	b - c - g fault,fault location 0.79,$R_f = 50$	b - c fault, remote, low impedance
13.	b - c - g fault,fault location 0.81,$R_f = 50$	b - c fault, remote, low impedance
14.	b - g fault, fault location 1.0, $R_f = 50$	normal - state (no fault state)
15.	c - g fault, fault location 1.0, $R_f = 50$	normal - state (no fault state)
16.	a - c - g fault, fault location 0.0, $R_f = 6$	a - c fault, local, low impedance