Scientia Iranica, Vol. 14, No. 2, pp 143–151

SCIENTIA IRANICA

Designing an Efficient Probabilistic Neural Network for Fault Diagnosis of Nonlinear Processes Operating at Multiple Operating Regions

R. Eslamloueyan[1], R.B. Boozarjomehry[1] and M. Shahrokhi[*]

Neural networks have been used for process fault diagnosis. In this work, the cluster analysis is used to design a structurally optimized Probabilistic Neural Network. This network is called the Clustered-Based Design Probabilistic Neural Network (CBDPNN). The CBDPNN is capable of diagnosing the faults of nonlinear processes operating over several regions. The performance and training status of the proposed CBDPNN is compared to a conventional Multi-Layer Perceptron (MLP) that is trained on the whole operating region. Simulation results indicate that both schemes have the same performance, but, the training of CBDPNN is much easier than the conventional MLP, although it has about 50% more neurons in its hidden layer. Both schemes can reasonably handle an increase in fault deteriorations. However, the training time for CBDPNN is much less than that of MLP. This issue gets severely important when the number of measured variables, along with process faults, increases. Since, for plant-wide fault diagnosis, the reduction in training time is crucial, the advantages of CBDPNN make it more appropriate for fault diagnosis compared to other alternatives for such a case.

INTRODUCTION

Fault detection and diagnosis are important tasks in process engineering. Plant faults may cause abnormal operation and, if not detected early, can cause emergency shutdown and even equipment damage or casualty. It should be noted that, even if emergency shut down or accident does not occur, the yield and product quality of a plant operating under abnormal situations (i.e. process variables deviate significantly from their nominal values) will not be maintained. Industrial statistics now estimate the economic impact due to emergency shut down and abnormal situations to be around $20 billion a year in the US petrochemical industries alone [1]. If the value of losses caused by abnormal situations of all process plants in the world were estimated, the importance of plant fault detection and diagnosis would be more clarified. Today's process plants are very complex, with many measured variables used for plant monitoring and, hence, Process Fault Diagnosis (PFD) in such plants is a very difficult task, even for an experienced operator. Therefore, designing an intelligent real time system for PFD has received considerable attention, both from industry and academia, because of the economic and safety impact involved [2].

Although there are various methods for PFD in open literature, based on the form of process knowledge used by these methods, they can be classified as process model-based and process history-based techniques. The model used in the former category can be a quantitative deep model [3] or a qualitative causal model, like a signed digraph [4]. The process history-based methods make use of the large amounts of process data obtained from recorded measured variables of the plant during abnormal and normal operations. This category consists of techniques like expert systems and statistical and neural network methods. Application of expert systems for fault diagnosis can be found in the open literature [5,6]. Multivariate statistical techniques, such as Principal Component Analysis (PCA) and Partial Least Squares (PLS), have been

1. *Department of Chemical and Petroleum Engineering, Sharif University of Technology, Tehran, I.R. Iran.*
*. *Corresponding Author, Department of Chemical and Petroleum Engineering, Sharif University of Technology, Tehran, I.R. Iran.*

employed for fault detection and diagnosis [7,8]. Using Artificial Neural Networks (ANN) in process fault diagnosis has received considerable attention over the last few years. Multilayer perceptron networks have been used in different frameworks for PFD [9-13]. Also, there are some other neuromorphic approaches used for PFD [14-18].

In general, the majority of neural network schemes used for PFD are based on the assumption that the process operating condition is not changed significantly. There are some cases in process industries where the plant has to be operated under several operating conditions (e.g., producing products with different compositions based on market demand). The change of operating condition of a nonlinear process may deteriorate the performance of a diagnostic scheme designed to detect process faults over a specific operating region. To overcome this problem, in this work, an approach is proposed that uses a single diagnostic scheme with an optimum structure and a simple training procedure for detecting faults over the whole operating region. The basic idea is to design a Probabilistic Neural Network (PNN) with a structure optimized, based on the cluster analysis of the training patterns. The potential of Radial Basis Function (RBF) neural networks for PFD has been demonstrated in the last decade. RBF networks have been used in the detection and isolation of process faults, either by off-line learning [19] or by on-line learning of the fault patterns through an adaptive network [20]. In the following sections, first, the proposed Cluster-Based Designed PNN (CBDPNN) is presented and the learning procedure is described briefly. Then, the simulated process used to evaluate the performance of the proposed diagnostic scheme is introduced. Finally, the performance of the proposed CBDPNN is evaluated and compared with PNN and MLP.

PROBABILISTIC NEURAL NETWORK AND CLUSTER-BASED DESIGN PNN

Probabilistic neural networks can be used for classification problems [21]. Figure 1 shows the architecture of this scheme. In this figure, Q is the number of training patterns consisting of normal and all fault patterns. Each target vector has K elements corresponding to normal and faulty conditions. One of these elements is one and the rest are zero. Thus, each input symptom vector is associated with one of the K classes. The first layer input weights, IW, are set to the transpose of the matrix formed by the Q training patterns. When an input symptom is presented to the input layer of the network, a vector is produced whose elements indicate how close the input symptom is to the centers of the RBF neurons. These elements are multiplied, element by element, by the bias "b" and sent to the radial basis transfer function. The transfer function for a radial basis neuron is given below:

$$\text{radbasis}\,(n) = e^{\ n^2}. \tag{1}$$

A plot of the radial basis transfer function is illustrated in Figure 2. The following equations show how the output vector of the RBF layer is calculated:

$$a_i = b_i \times e^{\ (\|IW_i \quad p\|)^2}, \tag{2}$$

$$\|IW_i \quad p\| = \sqrt{\sum_{j=1}^{R} (IW_{i,j} \quad p_j)^2}, \tag{3}$$

where:

R	the number of elements of the input vector "p",
a_i	ith element of the RBF output vector "a",
b_i	ith element of the bias vector "b",
IW_i	ith row of weight matrix of input layer (IW),
p_j	jth element of the input vector "p".

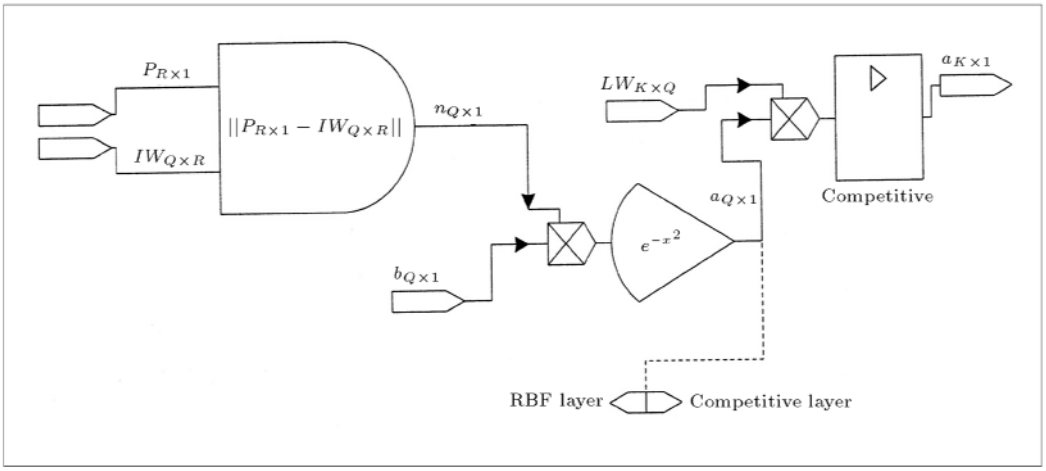

Figure 1. Schematic diagram of a PNN.

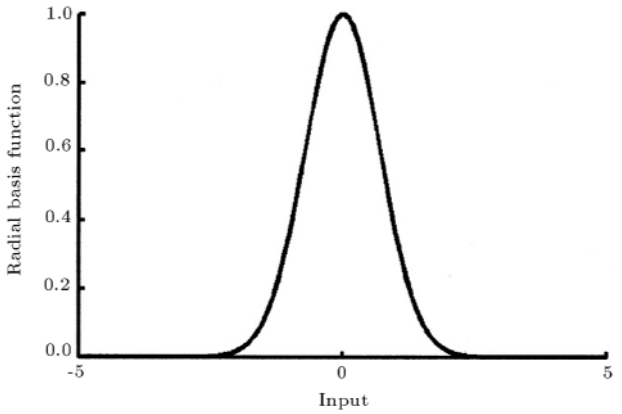

Figure 2. Radial basis function.

The bias, b, allows the sensitivity of the neuron with radial basis function to be adjusted. For example, as Figure 1 illustrates, if a neuron had a bias of 0.1, its output would be 0.5 for an input vector, p, whose distance from the vector of weight factors (IW) is 8.326 ($0.8326/b$). An equivalent variable called "spread" is defined, which is related to the bias through the relationship bias = 0.833/spread. An input symptom vector close to a fault pattern vector will be represented by a number close to unity in the output vector "a". If an input is close to several training vectors of a single class, it will be represented by several elements of "a" that are close to unity. The second layer weights, LW, are set to the matrix T of target vectors. All elements of each target are zero, except the one corresponding to that particular class of training pattern vector. The multiplication, "$LW \times a$", sums the elements of "a" corresponding to each of the K classes. The competitive transfer function of the second layer forces the largest element of the resulting vector to converge to unity and the other elements to zero. The second layer output is determined, according to the following recurrence relation:

$$\overline{n}_{K\times 1} = LW_{K\times Q} \times a_{Q\times 1}, \tag{4}$$

$$\overline{n}_{(t+1)} = \text{poslin}[W \times \overline{n}_{(t)}], \tag{5}$$

$$w^2_{i,j} = \left\{ \begin{array}{ll} 1, & i = j \\ \varepsilon, & i \neq j \end{array} \right\}, \tag{6}$$

$$0 < \varepsilon < \frac{1}{K \quad 1}, \tag{7}$$

where "poslin" is the positive linear transfer function, defined below:

$$\text{poslin}(x) = \begin{Bmatrix} x, & x \geq 0 \\ 0, & x < 0 \end{Bmatrix}, \tag{8}$$

and $\overline{n}_{(t)}$ is the input vector to the competitive layer at iteration "t".

The recurrence relation eventually converges to the output vector, $\overline{a}$. In this way, the network classifies the input symptom vector into a specific one of K normal/fault classes, because that class has the maximum probability of being correct.

The spread should be so large that the active input regions of the RBF neurons have enough overlaps so that several RBF neurons have fairly large outputs for all situations. This makes the network functioning smoother and results in a better generalization for new symptom vectors occurring between training patterns. On the other hand, if the spread is so large that each RBF neuron responds in the same large area of the input space, then, the diagnostic ability of the PNN will fail entirely. The value of "spread" is determined as the minimum Euclidian distance among all patterns.

The fault patterns used in this paper are obtained by considering each possible fault occurring at several different operating conditions of the process. Hence, the number of patterns for each fault is equal to the number of operating conditions of the process. However, it should be noted that all of these patterns might not be required for the training of PNN. Only those patterns of a specified fault that are sufficiently dissimilar or inconsistent, should be used in the training of the PNN. The cluster analysis is applied to determine the minimum number of inconsistent patterns required for designing a PNN that is able to detect process faults appropriately.

Cluster analysis, also called segmentation analysis, is a way to partition a set of objects into groups or clusters, in such a way that the profiles of objects in the same cluster are very similar and the profiles of objects in different clusters are quite distinct [22]. The following procedure is applied to perform cluster analysis on a dataset:

1. The similarity or dissimilarity between every pair of objects in the dataset is determined. In this step, the Euclidian distances between objects are calculated;
2. The objects are grouped into a binary, hierarchical cluster tree. Indeed, the pairs of objects that are in close proximity are linked together using the distance information generated in step 1. Objects are paired into binary clusters and the newly formed clusters are grouped into larger clusters until a hierarchical tree is formed;
3. The objects in the hierarchical tree are divided into clusters using the inconsistency coefficient of the links in the cluster tree. This coefficient compares the length of a link in a cluster hierarchy with the average length of the neighboring links two levels below it in the cluster hierarchy.

Therefore, by specifying a value for the inconsis-

tency coefficient (between 0 and 1), the clusters of the objects can be determined. The hybrid structure of a Cluster-Based Designed PNN (CBDPNN) is illustrated in Figure 3.

As Figure 3 shows, the training procedure of a CBDPNN is initiated with guessing an initial value for the inconsistency coefficient (cutoff threshold). Then, all patterns of faults are introduced into the cluster analysis algorithm and, based on the specified inconsistency coefficient, fault patterns are grouped into various clusters. The mean average of the patterns of a specified fault that are fallen in a cluster is employed as the representative vector of that fault for designing a PNN. This reduces the number of learning patterns and, consequently, the number of required RBF neurons of the PNN. The bias used in the CBDPNN is the same as that obtained in the design of the PNN described previously. The performance of the CBDPNN is evaluated through applying all available test patterns to the CBDPNN. The output and the target matrices are used to calculate the following performance index of the network.

$$\mathrm{PI} = \sum_{i=1}^{S} \|a_i \quad T_i\| \,. \tag{9}$$

In the above equation, a_i and T_i are ith columns of the matrices "a" and T, respectively. Also, $\|V\|$ denotes the Euclidean norm of the vector, V.

This performance index is minimized through adjusting the inconsistency coefficient via a minimization algorithm. There are many minimization techniques that may be used for finding the optimum value of the inconsistency coefficient. The Golden Section algorithm, which is a simple and well-known minimization technique of a single variable optimization, is employed for this purpose [23].

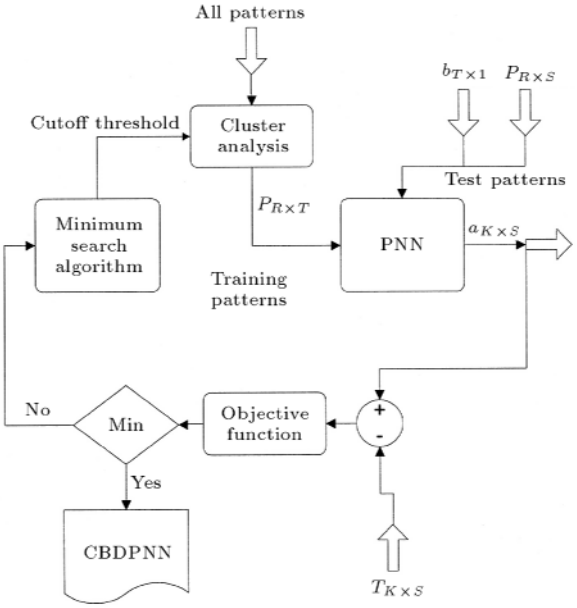

Figure 3. Flow chart for designing CBDPNN.

SIMULATED PROCESS

The nonlinear process considered for simulation is the same process studied by Henson et al. [24]. The system consists of two constant volume reactors cooled by a single coolant stream. An irreversible, exothermic reaction, $A \rightarrow B$, occurs in both reactors. The effluent concentration from the second tank is controlled by manipulating the coolant flow rate, q_c. Figure 4 shows, schematically, the simulated process. The mathematical modeling equations are presented in the Appendix. The parameters of the model can be obtained from [24].

Assuming the controller is always in a non-faulty operating mode, the following five possible causes of fault are considered for the plant:

$F1$	15% decrease in the feed concentration, C_{Af},
$F2$	10% increase in the feed temperature, T_f,
$F3$	10% decrease in the feed flow rate, q,
$F4$	10% increase in the coolant temperature, T_{cf},
$F5$	15% deactivation of catalyst, k_d.

The following five variables are measured for the process fault diagnosis:

T_1	outlet temperature of the first reactor,
T_2	outlet temperature of the second reactor,
q_c	coolant flow rate,
T_f	feed temperature,
T_{cf}	coolant temperature.

The above-mentioned measured variables are selected based on the fault observability and ease of measuring.

The plant produces the following product grades, based on market demand:

1. Pure grade: 99.5% M,
2. Chemical grade: 98% M,
3. Commercial grade: 95% M,
4. Raw grade: 90% M.

When the process shifts from one product grade to another, the operating region of the process varies

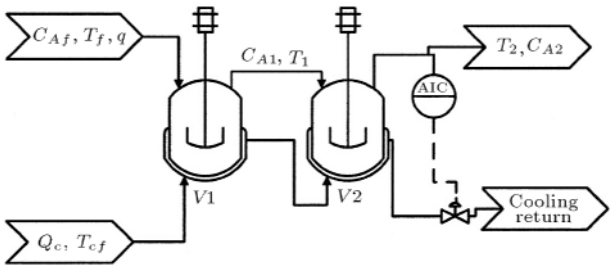

Figure 4. Flow diagram of the simulated process.

accordingly. Indeed, there are four operating regions associated with the four product grades. The steady state conditions of the plant at four different operating regions are given in Table A1. As noted before, the diagnostic scheme that has been trained, based on the fault patterns corresponding to one of these operating regions, may not work properly in other operating regions. This subject will be discussed in the next section.

TRAINING RESULTS

As mentioned before, the process operates at four different operating regions and since the process is highly nonlinear, a fault diagnostic scheme, trained in one of these operating regions, may fail at other operating conditions. To show this fact, a multilayer perceptron network is designed for each operating region. The required training fault patterns related to each operating region are obtained by simulation of the plant. Fault patterns used in this study are presented in Table 1. Normal distribution random noises with standard deviations of 0.2°C and 1 L/min for temperature and flow measurements, respectively, were added to the training patterns to generate the required data for training the MLP networks. The training results of five MLP networks, called hereafter Single Artificial Neural Networks (SANN), are given in Table 2. The transfer functions of the neurons in the hidden and output layers of these SANN's are tan-sigmoid and log-sigmoid, respectively. The number of neurons in input, hidden and output layers of each SANN is presented in Table 2. There are six training patterns for each operating region that consists of one normal pattern and five single fault patterns. By adding noise to these patterns, 2000 training data were generated for each pattern. The SANNs have been trained in a batch-wise manner, through the method of a scaled conjugate gradient [25].

As indicated in Table 2, the SANN covering the whole operating region of the process, i.e. SANN1234, is more complex and requires much more CPU time for training in comparison to other SANNs. Furthermore, as the number of faults increase, training of this scheme becomes more and more difficult. It should be noted that in the training of each SANN, one half of the data was used for training, one quarter for validating and the rest for testing. Figure 5 illustrates the plot of training error versus epochs during the training of SAN1234.

To compare a PNN trained on the whole operating region with the SANNs described before, a PNN is

Table 1. Single fault patterns at different operating regions of the process.

Measurement		T_1 (K)	T_2 (K)	q_c (L/min)	T_f (K)	T_{cf} (K)
Operating Region	**Fault**					
1: Pure Grade	F_1	-2.097	-1.380	-27.65	0	0
99.5% M	F_2	0.913	-0.917	37.23	35	0
	F_3	-1.994	-2.176	-5.918	0	0
	F_4	1.278	-1.284	58.63	0	35
	F_5	3.089	3.426	-6.595	0	0
2: Chem. Grade	F_1	-0.214	-3.673	-20.26	0	0
98.0% M	F_2	70.06	-35.75	106.2	35	0
	F_3	0.744	-2.504	-2.966	0	0
	F_4	73.77	-35.86	153.1	0	35
	F_5	-0.9	3.86	-4.86	0	0
3: Comm. Grade	F_1	-0.145	-3.491	-23.77	0	0
95.0% M	F_2	55.63	-21.29	92.54	35	0
	F_3	0.504	-2.273	-4.754	0	0
	F_4	61.26	-25.84	194.2	0	0
	F_5	-0.644	3.512	-6.144	0	0
4: Raw Grade	F_1	-0.098	-3.475	-27.27	0	0
90.0% M	F_2	26.91	-4.441	54.06	35	0
	F_3	0.398	-2.121	-6.002	0	0
	F_4	52	-18.42	238.8	0	35
	F_5	-0.522	3.282	-7.465	0	0

Table 2. Single artificial neural networks trained at different operating regions.

Name	Number of Neurons: Input/Hidden/Output	Training Data	Error Goal	CPU[1] Time (sec.)
SANN1	5/4/6	Region 1	0.001	116.11
SANN2	5/5/6	Region 2	0.001	164.44
SANN3	5/5/6	Region 3	0.001	175.16
SANN4	5/5/6	Region 4	0.001	262.93
SANN1234	5/10/6	All regions	0.001	1517.7

1- PC: Pentium III, CPU: Intel 733 MHz.

Table 3. Fault patterns at different operating regions classified through cluster analysis.

Cluster No.	$F1$ at Region:	$F2$ at Region:	$F3$ at Region:	$F4$ at Region:	$F5$ at Region:
1	One	-	-	-	-
2	Two	-	-	-	-
3	Three, four	-	-	-	-
4	-	One, four	-	-	-
5	-	Two, three	-	-	-
6	-	-	One	-	-
7	-	-	Two	-	-
8	-	-	Three, four	-	-
9	-	-	-	One	-
10	-	-	-	Two	-
11	-	-	-	Three, four	-
12	-	-	-	-	One
13	-	-	-	-	Two
14	-	-	-	-	Three, four

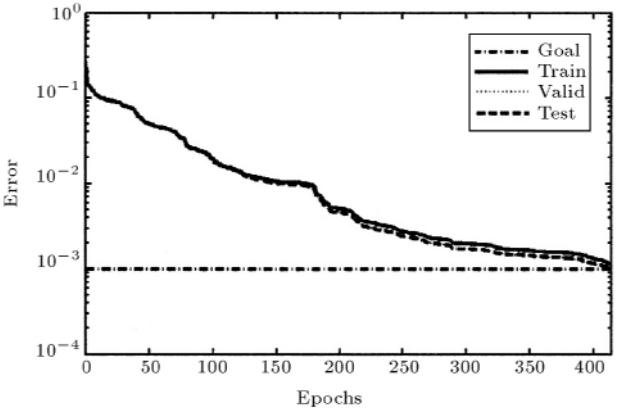

Figure 5. Performance error versus epochs during training of SAN1234.

designed by using 21 training patterns consisting of one normal pattern and all single fault patterns presented in Table 1. This PNN has 5 neurons in the input layer, 21 RBF neurons in the hidden layer and 6 neurons in the output layer. The maximum spread value, used in the PNN design, is equal to 0.07. Although the PNN structure is more complex than the SANN1234, its training time is almost negligible (about 0.4 second), compared to the time required for training the SANNs. On the other hand, as the number of fault patterns increases the number of hidden RBF neurons of the PNN increases proportionally. However, the number of RBF neurons in the hidden layer of the PNN can be optimized by the method explained before through designing a Cluster-Based Designed PNN (CBDPNN). Applying the procedure for designing a CBDPNN led to a PNN with 15 RBF neurons in the hidden layer. The training time for CBDPNN is about 8.5 seconds, which is still much less than the training time of the SANNs. The final clusters, determined through training CBDPNN, are presented in Table 3. Although the number of RBF neurons of CBDPNN is 1.5 times greater than the hidden neurons of the SANN1234, its training time is almost 200 times less than the training time of SANN1234.

Table 4. Faults detected by the diagnostic schemes at different operating regions.

Diagnostic Scheme	Operating Region			
	One	**Two**	**Three**	**Four**
SANN1	All	N^*, $F4$	N, $F1$, $F4$	N, $F1$, $F4$
SANN2	N, $F1$, $F3$, $F5$	All	N, $F1$, $F2$, $F3$, $F5$	N, $F1$, $F3$, $F5$
SANN3	N, $F1$, $F3$, $F5$	N, $F1$, $F3$, $F4$, $F5$	All	N, $F1$, $F2$, $F3$, $F5$
SANN4	N, $F1$, $F2$, $F3$, $F5$	N, $F3$, $F4$, $F5$	N, $F1$, $F3$, $F4$, $F5$	All
SANN1234	All	All	All	All
PNN	All	All	All	All
CBDPNN	All	All	All	All

*: N denotes normal condition.

DIAGNOSIS RESULTS

After adding noise to the measured variables of fault patterns at all operating regions, they were introduced into different diagnostic schemes explained before. The results are given in Table 4. It should be mentioned that the neurons of the output layer of the MLP network are log-sigmoid, which produce a real number between 0 and 1. The criterion for detecting a fault is that the value of the relevant output neuron be greater than 0.5. As can be seen, none of the SANNs trained, based on the data obtained at a specific operating region, can diagnose all faults correctly at other operating regions. This is due to the nonlinear behavior of the process over the whole operating region. On the other hand, the SANN1234, which used the fault patterns of all the operating regions, could diagnose all of the faults and the normal case over the whole operating region, at the expense of more difficult training and much higher CPU time. Although, in this case, one could train SANN1234 over the whole operating region, notice that the increasing difficulty of network training may cause the complete failure of the training algorithm.

As can be observed from Table 4, PNN with 21 and CBDPNN with 15 RBF hidden neurons have detected all faults correctly, at the expense of a greater number of hidden layer neurons, in comparison to SANN1234. However, in spite of more hidden neurons, the training of the network is much easier and the required CPU time of PNN is much less than that of SANNs. On the other hand, by using a CBDPNN, the number of hidden neurons can be made as small as possible. The advantage of CBDPNN for fault diagnosis becomes more evident when the degree of nonlinearity and complexity of the plant increase and the operating condition varies over a wide range. The flexibility of the CBDPNN and SANN1234 in diagnosing faults with a higher degree of deterioration has been evaluated by simulation. The results of this evaluation reveal that both diagnostic schemes can tolerate 30% magnification in $F1$, 50% magnification in $F2$ and $F4$, and 100% magnification in $F3$ and $F5$. Therefore, with respect to the generalization ability, CBDPNN is as good as SANN1234.

CONCLUSIONS

In this paper, the performance of a Probabilistic Neural Network (PNN) in diagnosing the faults of a nonlinear process plant operating at different operating conditions is evaluated and compared to the multi-layer perceptron networks. Also, to optimize the structure of the PNN, a cluster analysis is employed to find the minimum required RBF hidden neurons. The developed Cluster-Based Designed PNN (CBDPNN), with a smaller number of hidden neurons compared to PNN, can diagnose all of the faults over the whole operating region correctly. Although the number of hidden neurons of the CBDPNN is greater than the SANN trained over the whole operating region (SANN1234), its training procedure is much easier and the required CPU time for training is almost negligible. As the number of faults, measured variables and operating regions of a plant were increased, the training of a SANN capable of diagnosing faults over all operating regions became prohibitively difficult. Also, in such a case, the number of data points required for training a SANN and, consequently, the cost of the data gathering, is considerably much more than that of a CBDPNN. Evaluation of CBDPNN and SANN1234, for diagnosing faults with a higher degree of deterioration, reveals that both schemes can tolerate fault deterioration in a reasonably wide range.

NOMENCLATURE

a	output vector from RBF layer
a_i	ith element of the RBF output vector "a"
$\overline{a}$	output vector from PNN

A_i heat transfer area for ith tank
b bias vector
b_i ith element of the bias vector "b"
C_{Af} concentration of component "A" in the feed, gmol/L
$CA1$ concentration of component "A" at outlet stream from first reactor, gmol/L
$CA2$ concentration of component "A" at outlet stream from second reactor, gmol/L
C_p heat capacity
$e^{\ x}$ exponential function
E activation energy
F fault
h heat transfer coefficient
IW input weight matrix
IW_i ith row of IW
k_0 preexponential factor
K number of classes
LW layer weight matrix
n input vector to RBF neurons
$\overline{n}_{(t)}$ the input vector to the competitive layer at iteration "t"
N normal condition
OF objective function
P input pattern vector to the network
p_j jth element of the input vector "p"
q volumetric flow rate into the first reactor, L/min
q_c volumetric flow rate of coolant, L/min
Q number of all training patterns
R number of measured symptom variables
S number of testing patterns
T number of training patterns after clustering
T_{cf} coolant feed temperature, K
T_f feed temperature to the reactor, K
$T1$ outlet temperature from the first reactor, K
$T2$ outlet temperature from second reactor, K
$V1$ liquid volume in the first reactor, L
$V2$ liquid volume in the second reactor, L

Greek Letters

ΔH heat of reaction
ε inhibitory weight of the competitive weight matrix
ρ fluid density
ρ_c coolant density

REFERENCES

1. Venkatasubramanian, V. "Challenges in the industrial applications of fault diagnostic systems", *Comput. Chem. Engng.*, **40**, pp 785-791 (2000).
2. Nimo, I. "Adequately address abnormal situation operations", *Chem. Eng. Progress*, **91**, pp 36-45 (1995).
3. Frank, P.N. "Fault diagnosis in faulty systems using analytical and knowledge based redundancy- a survey and some new results", *Automatica*, **26** pp 459-474 (1990).
4. Vedam, H. and Venkatasubramanian, V. "Signed digraph based multiple fault diagnosis", *Comput. Chem. Engng.*, **21**, pp S5655-S5660 (1997).
5. Leung, D. and Romagnoli, J. "Dynamic probabilistic model-based expert system for fault diagnosis", *Comput. Chem. Engng.*, **24**, pp 2473-2492 (2000).
6. Young, E.S., Chang, T.S., Shin, D. and Yoon, E.S. "Cooperative problem solving in diagnostic agents for chemical processes", *Comput. Chem. Engng.*, **24**, pp 729-734 (2000).
7. Lin, W., Qian, U. and Li, X. "Nonlinear dynamic principal component analysis for on-line process monitoring and diagnosis", *Comput. Chem. Engng.*, **24**, pp 423-429 (2000).
8. McGregor, J.F., Jaeckle, C., Kiparissides, C. and Koutoudi, M. "Process monitoring and diagnosis by multiblock PLS methods", *AIChE Journal*, **40**, pp 826-835 (1994).
9. Venkatasubramanian, V. and Chan, K. "A neural network methodology for process fault diagnosis", *AIChE Journal*, **35**, pp 1993-2002 (1989).
10. Watanabe, K., Matsuura, I., Abe, M., Kubota, M. and Himmelblau, D.M. "Incipient fault diagnosis of chemical processes via artificial neural networks", *AIChE Journal*, **35**, pp 1803-1812 (1989).
11. Fan, J.Y., Nikolaou, M. and White, R.E. "An approach to fault diagnosis of chemical processes via neural networks", *AIChE Journal*, **93**, pp 82-88 (1993).
12. Watanabe, K., Hirota, S. and Hou, L. "Diagnosis of multiple simultaneous fault via hierarchical artificial neural networks", *AIChE Journal*, **40**, pp 839-848 (1994).
13. Eslamloueyan, R., Shahrokhi, M. and Bozorgmehri, R. "Multiple simultaneous fault diagnosis via hierarchical and single artificial networks", *Scientia Iranica, Transactions on Chemical and Environmental Engng.*, **10**(3), pp 300-310 (2003).
14. Tsai, C.S. and Chang, C.T. "Dynamic process diagnosis via integrated neural networks", *Comput. Chem. Engng.*, **19**, pp S747-S752 (1995).

15. Vedam, H. and Venkatasubramanian, V. "A wavelet theory based trend analysis system for process monitoring and diagnosis", *Proceeding of the American Control Conference*, New Mexico, pp 309-313 (June, 2000).

16. Chen, B.H., Wang, X.Z., Yang, S.H. and McGreavy, C. "Application of wavelets and neural networks to diagnostic system development- 1. Feature extraction", *Comput. Chem. Engng.*, **23**, pp 899-906 (1999).

17. Ruiz, D., Nougues, J.M., Calderon, Z., Espuna, A. and Puigjaner, L."Neural network based framework for fault diagnosis in batch chemical plants", *Comput. Chem. Engng.*, **24**, pp 777-784 (2000).

18. Rengaswamy, R., and Venkatasubramanian, V. "A fast training neural network and its updation for incipient fault detection and diagnosis", *Comput. Chem. Engng.*, **24**, pp 431-437 (2000).

19. Leonard, J.A. and Kramer, M.A. "Radial basis function networks for classifying process faults", *IEEE Control System Mag.*, **11**, pp 31-38 (1991).

20. Yu, D.L., Gomm, J.B. and Wiliams, D. "Sensor fault diagnosis in a chemical process via RBF neural networks", *Control Engineering Practice*, **7**, pp 49-55 (1999).

21. Wasserman, P.D., *Advanced Methods in Neural Computing*, Van Nostrand Reinhold, New York, pp 35-55 (1993).

22. Everitt, B., Landau, S. and Leese, M., *Cluster Analysis*, London-New York, 4th Ed. (2001).

23. Edgar, T.F. and Himmelblau, D.M., *Optimization of Chemical Processes*, McGraw Hill Book Company, 1st Ed. (1988).

24. Henson, M.A. and Seborg, D.A. "Input-output linearization of general nonlinear processes", *AIChE Journal*, **36**, pp 1753-1757 (1990).

25. Moller, M.F. "A scaled conjugate gradient algorithm for fast supervised learning", *Neural Networks*, **6**, pp 525-533 (1993).

APPENDIX

Mass and energy balances:

$$\frac{dC_{A1}}{dt} = \frac{q}{V_1}(C_{Af} \quad C_{A1}) \quad k_0 C_{A1} \exp\left(\frac{E}{RT_1}\right),$$

$$\frac{dT_1}{dt} = \frac{q}{V_1}(T_f \quad T_1) + \frac{(\quad \Delta H) k_0 C_{A1}}{\rho C_p} \exp\left(\frac{E}{RT_1}\right) + \frac{\rho_c C_{pc}}{\rho C_p V_1} q_c \left[1 \quad \exp\left(\frac{hA_1}{\rho_c C_{pc} q_c}\right)\right](T_{cf} \quad T_1),$$

Table A1. Plant steady state conditions at different operation regions.

Operating Region	Normal Steady State Condition
One	$C_{A1} = 0.0853$ kmol/m^3, $C_{A2} = 0.005$ kmol/m^3 $T_1 = 441.94$ K, $T_2 = 449.97$ K, $q_c = 1.651 \times 10^{\ 3}$ m^3/s
Two	$C_{A1} = 0.9594$ kmol/m^3, $C_{A2} = 0.02$ kmol/m^3 $T_1 = 355.09$ K, $T_2 = 472.8$ K, $q_c = 0.9934 \times 10^{\ 3}$ m^3/s
Three	$C_{A1} = 0.9695$ kmol/m^3, $C_{A2} = 0.05$ kmol/m^3 $T_1 = 354.05$ K, $T_2 = 452.57$ K, $q_c = 1.421 \times 10^{\ 3}$ m^3/s
Four	$C_{A1} = 0.964$ kmol/m^3, $C_{A2} = 0.1$ kmol/m^3 $T_1 = 353.51$ K, $T_2 = 437.76$ K, $q_c = 1.754 \times 10^{\ 3}$ m^3/s

$$\frac{dC_{A2}}{dt} = \frac{q}{V_2}(C_{A1} \quad C_{A2}) \quad k_0 C_{A2} \exp\left(\frac{E}{RT_2}\right),$$

$$\frac{dT_2}{dt} = \frac{q}{V_2}(T_1 \quad T_2) + \frac{(\quad \Delta H) k_0 C_{A2}}{\rho C_p} \exp\left(\frac{E}{RT_2}\right) + \frac{\rho_c C_{pc}}{\rho C_p V_2} q_c \left[1 \quad \exp\left(\frac{hA_2}{\rho_c C_{pc} q_c}\right)\right] \times \left[T_1 \quad T_2 + \exp\left(\frac{hA_1}{\rho_c C_{pc} q_c}\right)(T_{cf} \quad T_1)\right].$$