15th Triennial World Congress, Barcelona, Spain

IMPROVING FAULT DIAGNOSIS USING PROXIMITY AND HOMOGENEITY MEASURE

Gábor Terstyánszky*, László Kovács,**

** Dept. of Software Engineering, University of Westminster, United Kingdom*
e-mail: terstyg@wmi.ac.uk
*** Dept. of Information Technology, University of Miskolc, Hungary*
e-mail: kovacs@iit,uni-miskolc.hu

Abstract: It is impossible to define all faults in the design phase. As a result, faults not priori known may appear in systems that must be managed by fault diagnosis. Faults priori unknown modify the distribution of the input patterns and the homogeneity of assignments of input patterns to the output space. The change in distribution of input patterns may modify clusters and cluster class assignment. To identify changes in distribution of input patterns a proximity measure and in cluster class assignment a homogeneity measure is used, respectively. After appearing faults not known priori the RBF neural network is trained on-line using a modified supervised-unsupervised learning algorithm taking into account the proximity and homogeneity values. *Copyright © 2002 IFAC*

Keyword: fault diagnosis, neural networks, learning algorithms, classification

1. INTRODUCTION

To detect and isolate priori known faults and faults priori unknown the neural network-based pattern recognition is applied. This approach divides features into the following two categories (Sorsa and Koivo, 1994): normal mode (normal class) and several fault modes (fault classes). In the design phase engineers analyse systems to define input patterns, features and classes required for pattern recognition, and their assignment to faults. As a result of this analysis, there will be a set of priori known faults to support off-line training (Chen and Orady, 1996). It is impossible to define all would-be faults in the design phase. On-line training of the neural network using unsupervised learning is required to detect faults not know priori. The modified supervised-unsupervised learning algorithms either re-arrange existing clusters or add a new cluster to the existing clusters and re-arrange clusters near the new cluster after appearing faults not known priori. The paper presents a modified unsupervised-supervised learning algorithm that is able to detect occurrence of faults not known priori.

2. PATTERN RECOGNITION-BASED FAULT DIAGNOSIS

The fault diagnosis, based on neural network and pattern recognition, is divided into two stages: feature extraction and classification. In the first stage the neural network performs feature extraction, feature generation and feature selection having a set of input patterns **x**. It maps the set of the input patterns **x** in the input space to a feature vector **y** in the feature space. In the second stage the feature vector **y** is transformed into classes of the decision space **z**. There are two phases of the neural network-based fault diagnosis: training phase and the generalisation phase. In the off-line training phase either supervised or unsupervised learning methods train neural networks to diagnose faults. In the supervised learning approach a set of input vectors along with the corresponding outputs is presented to a neural network to establish relation between input patterns and classes. The supervised learning algorithms provide fault diagnosis of a priori known faults. The unsupervised learning approach forms clusters of feature vectors with the same properties and assigns them to classes. The unsupervised learning algorithms support diagnosing faults not known priori. In the generalisation phase, which is an on-line process, the neural network performs feature extraction and classification. To diagnose faults the Radial Basis Function neural network was selected – RBF – because it incorporates both the supervised and unsupervised learning method.

In RBF neural network, according to Haykin, the input nodes (or inputs) pass the input value to the nodes of the hidden layer. Each hidden node receives the original input value because connections among input and hidden nodes are not weighted. The hidden nodes are the radial basis units whose transfer functions are non-monotonic functions, which are usually based on the Gaussian density function. The input of each radial basis function is the distance between the input vector **x** and its centre c_j. The output vector **z** of the RBF neural network is:

$$\mathbf{z} = \sum_{j=1}^{n} w_j g_j(\|\mathbf{x} - c_j\|) = \mathbf{w}^T \mathbf{y} \quad (1)$$

where g_j - radial basis function,
$\|\mathbf{x}\text{-}c_j\|$ - distance measure,
$\mathbf{w}$ - weight of the feature vector,
$\mathbf{y}$ - feature vector,
$i=1\ldots m$ - number of inputs,
$j=1\ldots n$ - number of RBFs,
$k=1\ldots p$ - number of outputs,

3 DIAGNOSING A PRIORI KNOWN FAULTS BY RBF NEURAL NETWORK

During the off-line training phase unknown parameters of the RBF neural network are defined in order to support diagnosis of a priori known faults. The positions and the widths of the centres of the radial basis functions, and the weights associated with the output layer are the unknown parameters. To teach the hidden layer the unsupervised learning approach was selected. It is based on clustering that assigns input vectors $\mathbf{x}$ into clusters c_j containing points with the same features to obtain clusters as homogenous as possible. The clusters are described by radial basic functions and implemented by the nodes of the hidden layer. The parameters of the hidden layer are taught in two consecutive steps. First, locations of cluster centres are defined by c-means clustering algorithm. The cluster centre c_j is defined according to the equation:

$$\begin{aligned} &c_j(n+1) = c_j(n) + h[\mathbf{x}(n) + c_j(n)] \\ &\quad \textit{if } j = \textit{best matching node} \\ &c_j(n+1) = c_j(n) \\ &\quad \textit{otherwise} \end{aligned} \quad (2)$$

where h - learning rate,

Secondly, the "r" nearest-neighbour algorithm calculates widths of clusters. It defines the width of a radial basis function as the average Euclidean distance of its "r" nearest neighbouring centres. The width of the radial basis functions is:

$$s_j = \frac{1}{r} \sum_{l=1}^{r} (\| c_l - c_j \|)^2)^{\frac{1}{2}} \quad \forall i \neq j \quad (3)$$

where r - number of different centres,
c_l - nearest neighbour of centre c_j.

Parameters of the output layer are taught by the supervised learning method. The weights of the output nodes are determined by gradient descent method minimising the output error E.

$$E = \frac{1}{2} \sum_{j=1}^{n} \sum_{k=1}^{p} (z_{k,j} - \hat{z}_{k,j})^2 \quad (4)$$

where $z_{k,j}$ - required output of the k-th node,
$z_{k,j}$ - actual output of the k-th node.

Weights of the output nodes are updated according to the following equation until $E > \varepsilon$,:

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_j} = -\eta \sum_{l=1}^{p} g_j (z_{k,j} - \hat{z}_{k,j}) \quad (5)$$

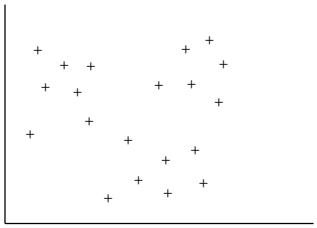

Fig. 1. Input (pattern) vectors to the RBF neural network

Having input vectors $\mathbf{x}_1$, $\mathbf{x}_2$, …$\mathbf{x}_n$, describing faults shown on Figure 1, the off-line training method is applied to the RBF neural network using the supervised and unsupervised learning algorithm. As a result of the off-line training an RBF network with 2 inputs, 3 hidden nodes and 2 output nodes was defined. It creates three clusters c_1, c_2, c_3 and two classes z_1 , z_2, as shown on Figure 2.

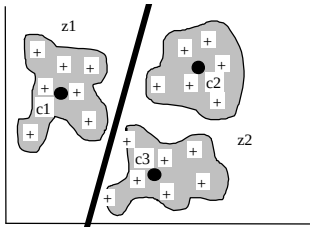

Fig. 2. Clusters and classes defined by the supervised-unsupervised learning algorithm

4. DIAGNOSING FAULTS NOT KNOWN PRIORI BY MODIFIED SUPERVISED-UNSUPERVISED LEARNING ALGORITHM

Fault analysis cannot define all faults in systems during the design phase. As a result, faults not known priori may appear in systems. They may modify the distribution of the input pattern vectors and the homogeneity of assignments of input patterns to the output space. To manage faults not known priori a modified supervised-unsupervised learning algorithm was elaborated. The change in distribution may affect clusters, i.e. it may either re-arrange existing clusters or add a new cluster to existing clusters and re-arrange clusters around the new cluster. The change in distribution may also influence the cluster -> class assignment. It may either modify the existing assignment or may add a new class and change cluster -> class assignment. As a result it is required to check first, whether an input pattern vector $\mathbf{x}$ belongs to a particular cluster secondly, whether it modifies the cluster -> class assignment.

The proximity measure was used to define how dissimilar or similar a pattern vector and a cluster where clusters contain subset of the training set. Each cluster has a particular distribution and set of pattern vectors. Appearance of faults not known priori changes the distribution of patter vectors and may modify the existing clusters. A pattern vector $\mathbf{x}$ is

assigned to a cluster c_j during the training phase taking into account proximity between **x** and c_l. Each cluster c_l has a representative, which is the centre of the cluster and described by the radial basis function. The proximity between **x** and c_l is measured as the proximity between **x** and the representative of the cluster c_l. To define the proximity the mean centre m_c point representative was selected. The mean centre m_c is defined as

$$\sum_{y \in c_j} d(m_c, y) \le \sum_{y \in c_j} d(x_f, y) \qquad (6)$$

where

d - dissimilarity between two points,
y - feature vector,
$\mathbf{x}_f$ - fault vector,
m_c ? c_l.

If the proximity measure of the fault vector is greater than the proximity measure of each cluster then the clusters must be modified using equation (1)-(2).

Two additional input vectors $\mathbf{x}_{f1}$, $\mathbf{x}_{f2}$, were added to the input space as shown on Figure 3. They represent faults not known priori, which were not included earlier in the training set. The supervised learning algorithm re-arranged the cluster c_1, c_2 and c_3 as shown on Figure 4. The number of hidden nodes (i.e. clusters) has not been changed.

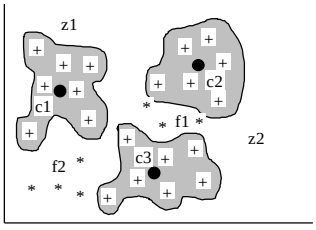

Fig. 3. Appearance of fault vectors x_{f1} and x_{f2}

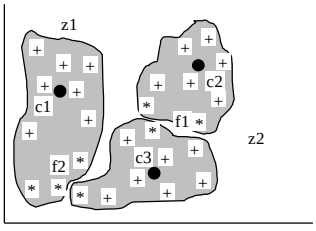

Fig. 4. Cluster after re-arrangement

If there are a priori known faults in systems then the assignment of feature vectors is homogenous, i.e. every input vector, belonging to the same cluster assigned to the same class. After appearing faults not known priori the assignment of feature vectors to fault classes may change. To handle change in the cluster -> class assignment a homogeneity value $h(\mathbf{x})$ was introduced for every radial basis function:

$$h(\mathbf{x}) = \text{argmax } a_k(\mathbf{x}) \qquad (7)$$

where $a_k(\mathbf{x}) = \Sigma\, r_k(\mathbf{y}) / D$

$r_k(\mathbf{y}) = w_{jk}(\mathbf{y}) / \Sigma\, \mathbf{w}(\mathbf{y})$

$\mathbf{w}_{jk}(\mathbf{y})$ - the weight value from the RBF function g_j to the output k representing the class z_k in the output layer,

$\Sigma\, \mathbf{w}(\mathbf{y})$ - the sum of weight values outgoing from the RBF g_j

D - the number of radial basis function in the region of **x**

The $r_k(\mathbf{y})$ approximates the posterior probability $q_k(\mathbf{x})$ of class k after presenting the input pattern vector **x** to the RBF neural network. The region of **x** includes those radial basis functions g_j which are closer to **x** than a given limit value e:

$$d(\mathbf{x}, \mathbf{z}) < ? \qquad (8)$$

The sum $a_k(\mathbf{x})$ is calculated in the region of **x**. As a result, for every incoming **x** input pattern vector the $h(\mathbf{x})$ value, which denotes the homogeneity of the region around the **x** vector, can be calculated. To take into consideration the homogeneity, the learning factor μ of the output nodes should depend not only on time but on the homogeneity $h(\mathbf{x})$, i.e.

$$h(n, h(\mathbf{x})) = h(1 - h(\mathbf{x}))\, f(n) \qquad (9)$$

In regions, where the cluster -> class assignment is non-homogenous, like boundary regions between classes, the input pattern vectors have a larger attraction power than in homogenous regions. As a result, weights of the output nodes should be modified according to the following equation:

$$w_{jk}(n+1) = w_{jk}(n) + h(n, h(\mathbf{x}))[\mathbf{x}_j(n) - w_{jk}(n)] \qquad (10)$$

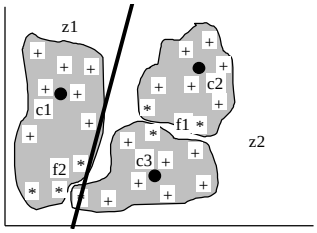

Fig. 5. Classes after re-arrangement

Class z_1, and z_2, has a homogenous assignment on Figure 1. It has been sligthly changed after appearing input vectors $\mathbf{x}_{f1}$, $\mathbf{x}_{f2}$, because c_1 and c_2 are assigned to both classes.

5. APPLICATION OF RBF NETWORK FOR DIAGNOSING FAULTS IN AMV

The modified supervised-unsupervised learning algorithm is applied to a model of an autonomous mobile vehicle (AMV) to diagnose faults. The AMV, shown in Figure 6, is a four-wheel driven vehicle with a separate DC motor drive for each wheel. The chassis of the vehicle is divided into two parts, tractor and a semi-trailer that are connected by a rotary joint to attenuate the slide-slip of wheels. The vehicle is capable to move straight backward and forward, to corner, to follow a trace taking into account its environment and to perform docking. The control system of the AMV is divided into three hierarchical layers. The first layer contains pilot

subsystems P_1... P_4 that control DC motors of wheels. The second layer includes navigator subsystems N_1...N_2 that enables the AMV to follow a given trace. The third layer consists of the global planner GP that supervises the pilot and navigator subsystems. A virus subsystem, V was also added to the AMV to generate faults. The virus subsystem is capable to generate faults in actuators, in sensors, in the vehicle and its environment.

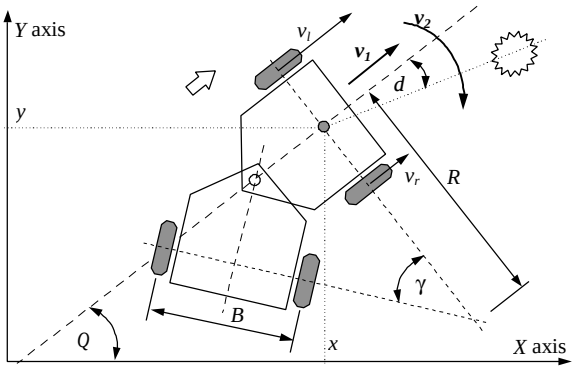

Fig. 6. The autonomous mobile vehicle

where

x,y - Cartesian position,

Q - heading angle to the X axis,

δ - heading error, B -wheel base,

R - curve radius, γ - rotary joint angle,

v_1 v_2 - forward and angular velocity,

v_l v_r - absolute velocity of the left & right wheel,

To evaluate how efficient is the RBF network-based fault diagnosis one of the pilot subsystems, shown in Figure 6**,** was investigated. The on-line unsupervised learning reveals how the input patterns are similar or different even if the classes of the training patterns are not available using the clustering approach.

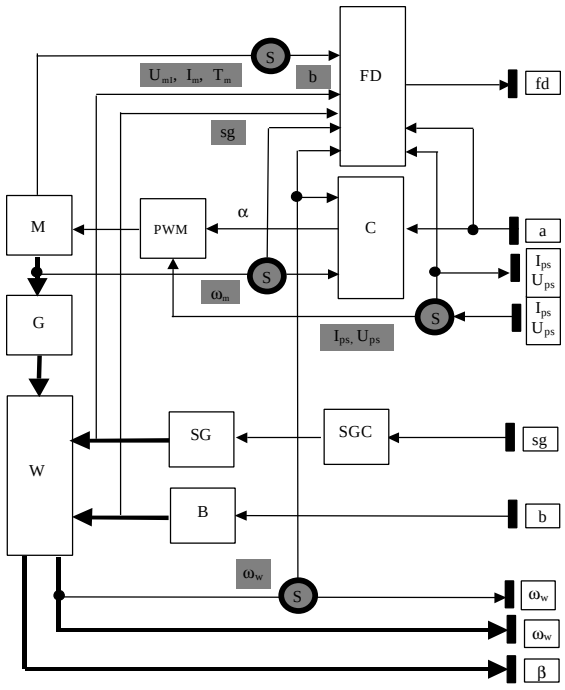

Fig. 7. Pilot subsystem

where

B - brake C - controller FD - fault diagnoser
G - gear M - DC motor PWM –supply control
S - sensor SG - steering gear
SGC – steering gear control W - wheel

The pilot subsystem has nine input parameters (faults) in the input pattern space and output five parameters (fault classes) in the decision space:

input parameters:

b - B control signal, sg – SG control signal,
I_m - M current, U_m – M voltage,
T_m - M temperature, ω_m – M angle velocity,
I_{ps} - PS current, U_{ps} – PS voltage,
ω_w - W angle velocity

output parameters:

f_B - fault in brake f_{SG} – fault in steering gear
f_M - fault in DC motor, f_W – fault in wheel,
f_{PS} - fault in power supply

The block diagram of the system under investigation is shown in Figure 7. A large data set, describing the normal mode and fault modes, should be obtained from the AMV for off-line training of the RBF neural network. Samples of input and output parameters were extracted from the AMV using the MatLab-SimuLink tool to form the training and testing data for the RBF network.

In the first phase of the simulation the RBF neural network is trained off-line to be able to diagnose a priori known fault using the supervised and unsupervised learning method given by equations (1)-(5). As a result of the training the RBF network contained of 9 inputs, 5 hidden layers and 3 outputs.

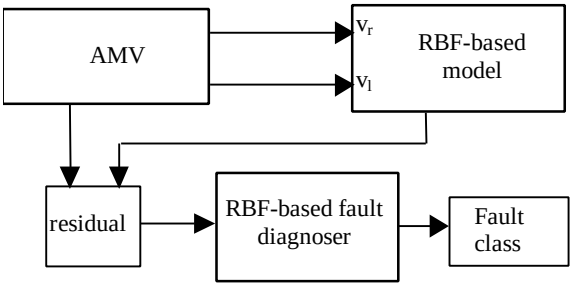

Fig. 7. Simulation system

In the second phase of the simulation one known fault in the DC motor, **M** is generated. After that a fault in the wheel, **W,** which was not known priori is simulated. The RBF-based fault diagnoser is able to detect both faults using the combined supervised-unsupervised learning algorithm.

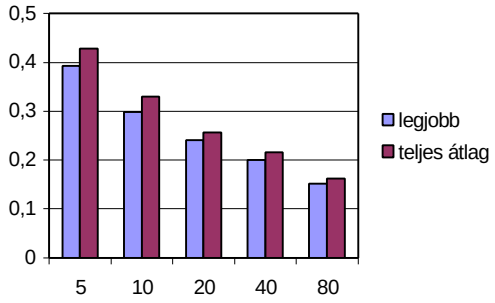

Fig. 8. Average quadratic error versus number of neurons

The simulation results are shown on Figure 8, Figure 9 Figure 10 and in Table 1.

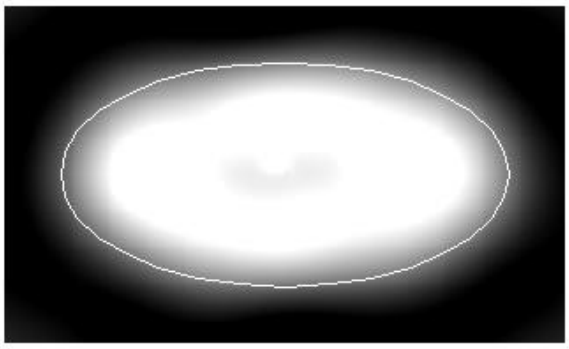

Fig. 9. Simulation results

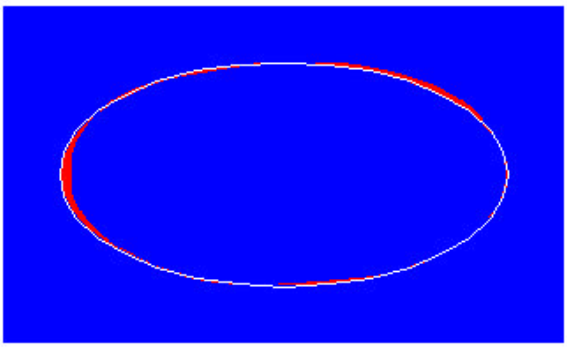

Fig. 10. Simulation results

8. CONCLUSIONS

To detect and isolate a priori known and unknown faults the application of the RBF neural network to fault diagnosis was investigated. A priori known faults are diagnosed by the neural network, which are trained off-line by a supervised-unsupervised learning algorithm. Faults not known priori modify the distribution of the input patterns and the homogeneity of assignments of input patterns to the output space. To identify changes in distribution of input patterns and in assignment of feature vectors and classes the proximity and homogeneity measure was used, respectively. After appearing faults not known priori the RBF neural network was upgraded using a modified supervised-unsupervised learning algorithm taking into account the proximity and homogeneity values. The modified supervised-unsupervised learning algorithms first, either re-arrange existing clusters or add a new cluster to the existing clusters and re-arrange clusters near the new cluster after appearing faults not known priori. Secondly, it either re-assigns clusters to existing classes or adds a new class and re-assign classes and clusters.

REFERENCES

Brown, M, Harris C. J (1994). *Neuro-fuzzy Adaptive Modelling and Control*. Prentice-Hall.

Chen Y, Orady E (1996). Integrated Diagnosis Using Information Gain Weighted Radial Basis Function Neural Networks. *Computer and Industrial Engineering*, **Vol. 30**, No. 2, pp. 243-255

Dalmi I., Kovács L. & Terstyánszky G (1999). Diagnosing Faults by Supervised and Unsupervised Learning. European Control Conference ECC'99, Karlsruhe, Germany, 31 August – 03 September 1999, CDROM F1046-3.

Haykin S (1999) *Neural Network. A Comprehensive Foundation*. Prentice Hall.

Karayiannis N. B, Mi G. W (1997). Growing Radial Basis Neural Networks: Merging Supervised and Unsupervised Learning with Network Growth Techniques. *IEEE Trans. On Neural Network*, **Vol. 8**, No. 6, pp. 1492-1506

Sorsa T., Koivo H. N (1994). Application of Artificial Neural Network in Process Fault Diagnosis. *Automatica*, **Vol. 29**, pp. 843-849

Theodoridis S, Koutroumbas K (1999). *Pattern Recognition*. Academic Press.

Table 1 Simulation results

number of neurons	smallest average quadratic error	average of the quadratic error	percent of the correct answers
5	0,3928	0,4286	94,22
10	0,2975	0,3302	94,81
20	0,2413	0,2560	94,67
40	0,2007	0,2154	96,74
80	0,1519	0,1627	98,43