1999 European Control Conference (ECC)
31 August – 3 September 1999, Karlsruhe, Germany

DIAGNOSISING FAULTS BY SUPERVISED AND UNSUPERVISED LEARNING

L. Kovács, G. Z. Terstyánszky

Dept. of Information Technology,
University of Miskolc, H-3515 Miskolc-Egyetemváros, Hungary
fax: (+36)-46-362-615, e-mail: kovacs@iit.uni-miskolc.hu

Keywords fault diagnosis, neural network, supervised and unsupervised learning,

Abstract

Neural networks provide a solution to overcome drawbacks of the quantitative fault diagnosis because first, they are capable to model off-line the behaviour of linear and non-linear systems. Secondly, they can also learn on-line the behaviour of a system requiring no priori knowledge about the system. The neural networks are particularly good for fault diagnosis of systems that have imperfect models and/or incomplete data. There are two basic learning methods of neural networks that are applied to fault diagnosis: supervised and unsupervised learning methods. To solve problem of priori unknown faults, unsupervised learning is used. The Counterpropagation network was selected to diagnose faults as result of analysis of supervised and unsupervised learning methods applied to neural networks.

1 Introduction

Quantitative model-based fault diagnosis generates information about the location and the time of the occurrence of a fault in a system by comparing behaviour of the system and its model. The fault diagnosis procedure is divided into two consequent phases:

- fault detection
- fault isolation.

In the first phase a residual is defined to detect a fault. The residual is examined in the second phase in order to locate the fault. This approach has two basic assumptions: first, a precise model of the system is given second, there are no unknown inputs. This is not the case in many physical systems because they have either imprecise model or unknown inputs. Neural networks provide a solution to overcome these drawbacks according to [brow] because first, they are capable to model off-line linear and non-linear physical systems using training data. Secondly, they can also learn on-line not requiring a priori knowledge about the particular physical system. Thirdly, neural networks can recognise spatial, temporal and other relationships in physical systems.

There are several problems in the fault diagnosis based on neural networks are as follows:

- basic task is to distinguish the error-free class from the classes belonging to one of the faults,
- there are different classes (fault types) and classes may overlap with each other,
- regions of classes are depending on each other.
- input vector space may have large number of dimensions,
- similar input vectors may belong to different classes.

There are two basic approaches in application of neural networks to fault diagnosis procedures: residual-based approach and pattern recognition approach. The first approach was selected to diagnose faults because it is suitable to signal the occurrence of faults comparing corresponding outputs of the physical system and its model.

2 Analysis of supervised and unsupervised learning methods in fault diagnosis

There are several approaches how to teach neural networks. They are as follows: supervised learning, unsupervised learning and their combination i.e. supervised-unsupervised learning.

The *supervised learning*, according to [poli], requires a priori knowledge of what the result should be. In this learning method, there is a trainer that corrects the network's response to a set of inputs. Pairs of inputs and outputs have to be presented to the network during the learning phase. The network takes an input and produces the corresponding output, which it then compares to the correct output. As a result, the network constructs an internal representation of inputs and outputs. The neural network is trained with a set of input-output pairs that describes a priori known states. The neural network learns a function that defines input-output relations. For a given new input, its corresponding output can be determined from the function that most closely replicates this pattern.

The supervised learning is closely related to classification. The supervised learning is able to handle a

ISBN 978-3-9524173-5-5

priori known normal and fault states. The classification can handle both normal states and a priori known fault states, but it is unable to handle a priori unknown fault states. The Multi-Layered Percepton network - MLP - and the Radial Basis Function network - RBF -, based on the supervised learning were studied in [dalm] how to apply them to fault diagnosis. Several learning algorithms of the MLP network, for example back-propagation algorithm, the momentum algorithm and the learning algorithm of the RBF network, were applied to an autonomous mobile vehicle - AMV - using the NeuralWare software package to detect and isolate a priori known faults.

Most of neural network-based fault diagnosis systems require a priori fault classes that are used to train the networks in order to recognise faults. It may be extremely difficult or dangerous to acquire fault data from physical systems. To solve this problem, models have been used to generate training sets. Thus, the efficiency of the supervised learning depends upon the quality of the data that it has been trained. To handle priori unknown faults, *unsupervised learning* is used. The neural network classifies the data and the network learns new faults and adapts them to similar faults that have already occurred. In unsupervised learning, there is no trainer. Instead, the network is simply exposed to a number of inputs. The self-organising capability may involve competition and co-operation or both. In competitive learning nodes in each cluster compete with one another for the right to recognise some feature in the input. The node with the highest activation will be selected i.e. winner-take-all. Next, weights of this node are updated in order to increase similarity. In co-operative learning the network reaches global organisation by way of local - non-linear, parallel and recursive - interactions. In this case nodes within each cluster work together to try to simulate each other. The output of each cluster is usually sent to some other part of the network. The part of the network receiving the output may than in turn to modify its external connections.

The unsupervised learning is based on the clustering approach. It looks for regularities in input-output pairs to define clusters using some form of utility measures. Codebook vectors represent clusters. New data are assigned to existing clusters using some form of a utility measure. The network organises itself in clusters in order to classify inputs. The different clusters represent different features of inputs. The input data are divided into disjoint clusters such that similarities between individuals in the same cluster larger than individuals in different clusters.

The Kohonen network - KHN - and the Counterpropagation network - CPN - was selected in order to examine whether they are suitable to diagnose priori unknown faults. The SOM, the LVQ and the CPN algorithm was examined in the AMV using the NeuralWare software package in [dalm].

Supervised and unsupervised learning algorithms can be combined to form a *supervised-unsupervised learning* algorithm that is suitable to diagnose both a priori known and unknown faults. Each system may be given by a set of input vectors representing a set of known system states corresponding to normal and/or fault states. The system is taught with this set of training data corresponding to known system states - normal and a priori known fault states -. Based on this the fault diagnosis procedure assigns input vectors to clusters or rejects them during the generalisation phase. There may be unknown states from which no training data are available. Evaluating input vectors, it is very important to recognise whether the current data corresponds to one of the known states or to an unknown state. In the second case the data is “rejected”. Having these data we can form a new class during a clustering process. An unsupervised learning procedure generates a new partition, possibly including new clusters, after a predefined number of input vectors has been rejected. This new partition is determined in such a way that each cluster is correctly classified using all other states. The set of classes is extended by this new class.

Taking into account advantages and disadvantages of RBF and CPN network, the CPN network was selected because it includes both the supervised and unsupervised learning. According to [dalm],[leon] the CPN trains faster than other usual classification neural networks.and it leads to better decision boundaries. The CPN is suggested to use for classification problems when training data are sparse and unrepresentative. The CPN network has the following advantages and disadvantages in application to fault diagnosis:

advantages:

- the CPN requires relatively small number of codebook vectors,
- dissimilar input vectors may belong to similar output classes,
- codebook vector of the CPN can be assigned to more than one output class,
- codebook vectors can move to the correct regions,
- it enables representing the same class in the different regions of the input space.

disadvantages:

- if an output class consists of disjoint regions, the competitive layer should contain several codebook vectors belonging to this class,
- the efficiency of the CPN depends on the initial value of the codebook vectors.

3. Supervised-unsupervised learning using counter propagation network

The CPN is a feedforward network. Its architecture is a combination of the self-organising map of Kohonen and the

outstar structure of Grossberg. Two types of layers are used. The hidden layer is a Kohonen layer with competitive nodes that do unsupervised learning. The output layer is a Grossberg layer, which is fully connected with the hidden layer and is not competitive. When presented with an input vector, the network classifies it using a learned reference vector. When trained, the network works as follows: after presentation of an input to the input layer, the nodes in the hidden layer sum their inputs and compete to respond to that input. The node with the highest input wins and its activation is set to 1 while all others are set to 0. After the competition, the output layer calculates a weighted-sum on the outputs of the hidden layer.

The teaching of the CPN is performed in two consequent phases. The first phase adjusts the competitive layer, the second phase adjusts the output layer. The competitive layer can learn how to give the best response to a particular input vector. In the first phase an input vector is selected for training. This input vector should be normalised. Next, the winning node is determined by calculating the input value for each node and selecting the node whose weight vector is closest to the input vector. Thirdly, change in weights of the winning node is defined and weight vector is updated:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \alpha[\mathbf{x}(k)-\mathbf{w}(k)] \quad (1)$$

where $\mathbf{w}(k)$ - weight of the winning node,
$\mathbf{x}(k)$ -input vector to node,
α - learning rate,
k - discrete time.

These teaching steps mentioned above are repeated until all inputs vectors have been classified properly. In the second phase clusters and output vectors are mapped. There are two cases. First, input vectors in a cluster map to the same output vector. In this case mapping is not necessary. Secondly, input vectors in a cluster map to different output vectors. In this case a hidden node which wins for a particular class is determined. The weight vector is assigned to the weight vector of the appropriate connections to the output layer will be equal to:

$$\mathbf{w}_{ki}(k) = \mathbf{y}_i(k) \quad (2)$$

where $\mathbf{w}_{ki}(k)$ - weight on the connection from the i-th hidden node to the k-th output node,
$\mathbf{y}_i(k)$ - required output vector of a class.

In the second case a normalised input vector and its corresponding output vector is applied to the CPN network in order to define the winning competitive node. Next, weights on the connections from the winning competitive node to the output nodes are defined:

$$\mathbf{w}_i(k+1) = \mathbf{w}_i(k) + \beta\,[\mathbf{y}_{ki} - \mathbf{w}_i(k)] \quad (3)$$

where β - learning rate

The codebook vector of the CPN can be assigned to more than one output class. It requires a small number of codebook vectors. It enables representing the same class in the different regions of the input space. In CPN dissimilar input vectors may belong to similar output classes and codebook vectors can move to the correct regions.

If the faults and the density of the input vectors are a priori known, the CPN can be adjusted to the given situation, and the network can work very efficiently and reliable. But many system cannot be trained off-line as of result of their dynamic nature. Thus, some modifications should be introduced to suit the CPN to handle a priori unknown faults. Firstly, as the number of codebook vectors in the competitive layer is limited, the positions of these codebook vectors should be optimised. The general aim is to minimise the number of codebook vectors in a given region if the faults are homogenous there, i.e. every input vectors of this region are belonging to the same fault class. But, if the fault classes are heterogeneous, i.e. there are different faults within the region, a finer grid of codebook vectors is preferable. Secondly, in order to deal with new faults, the distance of the incoming input vector and of the winning codebook vector should be checked. If this distance is larger than a given threshold or its trend shows that it will cross this threshold, the system returns an unknown state, giving fault class that is nearest to them.

The unsupervised learning is based on clustering. The clustering can be treated as a reduction function **u**, where the cardinality of the function domain is less then the cardinality of the image set. In the case of neural networks, the **u** function depends on a set of weight patterns $\{\boldsymbol{w}_i\}$ that come from the connection weights between the neurons in the net:

$$\mathbf{u}(\{\mathbf{w}_i\}) : \mathbf{I} \rightarrow \mathbf{I}' \quad (4)$$

where $|\mathbf{I}| > |\mathbf{I}'|$

During the training phase, the values of $\{\mathbf{w}_i\}$ weights are altered. We call the vectors of **I** as input vectors and **I′** as codebook vectors. Modification of the $\{\mathbf{w}_i\}$ values is based on the fact, that each unsupervised neural network can be modelled by the cost function **E** that should be minimised [QbU]. If the general gradient method is applied, the learning method can be defined as

$$\{\mathbf{w}_i\}(k+1) = \{\mathbf{w}_i\}(k) + \alpha(k) \times (-\partial E/\partial\{w_i\}) \quad (5)$$

In fault diagnosis, we can use the clustering to associate an arbitrary input vector to an existing or a new codebook vector. But the fault diagnosis requires the ability to distinguish between the fault-free and the different fault states. During the test of the CPN for fault detection and isolation, we found some problems affecting the efficiency of the neural network. Some of the problems are the following:

- it is very hard to determine how many codebook vectors are necessary,
- the efficiency depends on the initial value of the codebook vectors,

- untrained input vectors and fault classes may appear during the operation.

In order to improve the efficiency of the CPN, we introduced some modifications of the basic CPN algorithm. These modifications are related to the optimisation of the training process and detection of a priori unknown faults.

Regarding the accuracy of the network, the larger is the number of the codebook vectors, the better is the approximation. On the other side, the number of the codebook vectors is limited by computing constraints, for example execution time and memory space. Thus, our task is to find the optimal set of codebook vectors, containing "n" vectors, that describes all fault classes defined during training. First step is to measure the goodness of the codebook vector set. During the training phase, this goodness can be evaluated by the difference of the output values defined by using input vectors and of output values approximated by using codebook vectors:

$$\sum |o_i - u_i| \qquad (6)$$

where o_i - output value related to the input vector, and
u_i - output value approximated by using codebook vectors.

This function can be minimised if the density of codebook vectors is low i.e. when the class distribution is homogenous. In contrast the density of codebook vectors is high if the class distribution is heterogeneous.

Thus, our task is to find the optimal set of codebook vectors containing no more than "n" vectors. The first step is to define a quantity to measure the goodness of the codebook vector set. It was assumed that there is a finite number of fault classes. Every "I" input vector is assigned to only one of the fault classes. On the other hand it was also assumed that the association layer of the CPN is based on the basic SOM algorithm. The data of the input vectors and the corresponding fault classes are stored in a file, thus we can access it several times.

The inhomogenity of the codebook vector set is measured in this approach by

$$h = \Sigma_i h_i = \Sigma_i \Sigma_j \alpha_{ij} cnt_{ij} \qquad (7)$$

where α_{ij} - weight of cnt_{ij}
cnt_{ij} - number of input vector belonging to the the i-th codebook vector and to the j-th fault class.
i - index of a codebook vector, and
j - index of a fault class.

The cnt_{ij} is equal to the number of input vector belonging to the i-th codebook vector and to the j-th fault class. The α_{ij} is the weight of cnt_{ij} where $\alpha_{i'j}$ is equal to k if and only if the $cnt_{i'j}$ value is the (k+1)-th element in the ordered list of cnt_{ij} values. Thus, if every input vectors belonging to the i-th codebook vector have the same fault class, the h value is minimal, i.e. zero. The h has the maximal value, if every fault class has the same number of input vector members.

A heuristic algorithm was developed to improve the "h" value of a given initial codebook vector set. This algorithm includes the following three phases:

Initial phase: The algorithm starts with an empty set of codebook vectors. The first *n* codebook vectors are chosen randomly from the set of input vectors. This gives a better distribution compared with the method in which we choose the first *n* elements from the set of the input vectors. After then, codebook vectors are taught by the basic SOM algorithm. The result of this step corresponds to the normal final state in CPN.

Selection phase: The *h(C)* value of the *C* codebook vector set is evaluated. In order to improve the *h* value we perform a heating process. In this phase, we select a codebook vector to be temporally removed from *C*. We choose the $c \in C$ vector, so that

$$H(C \backslash c) \Rightarrow \text{minimal}$$

Iteration phase: The previously removed *c* vector will be inserted into *C* again. The starting position of *c* is position of the *c'* having the maximal *h(c')* value. The position of *c* is altered in an iterative way. The optimum position is searched by the hill-climbing gradient method.

4 Application of supervised and unsupervised learning methods

The supervised-unsupervised learning method is being applied to a model of an autonomous mobile vehicle (AMV) in order to diagnose fault in actuators, sensors and the system. The AMV, shown in Fig. 1, is a four-wheel driven vehicle. Separate DC motor drives each wheel. The chassis of the vehicle is divided into two parts that are connected by a joint in order to attenuate the slide slip of wheels. Each part of the vehicle has a pair of wheels. A rechargeable battery provides the power supply of the AMV. 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.

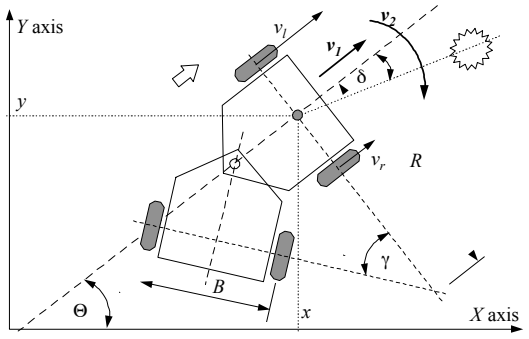

Fig. 1 The autonomous mobile vehicle

where x,y - Cartesian position,
Θ - heading angle to the X axis,
δ - heading error,
B -wheel base,
R - curve radius,
γ - rotary joint angle,
v_1 - forward velocity,
v_2 - angular velocity,
v_l - absolute velocity of the left wheel,
v_r - absolute velocity of the right wheel.

The control system of the AMV is divided into three hierarchical layers. The first layer contains pilot subsystems, P_1... P_4 that control the motion of the AMV by controlling DC motors of wheels. The second layer includes navigator subsystems, N_1...N_2 that provide following a given trace. The third layer consists of the global planner, GP that supervises the pilot and navigator subsystem. In order to detect and isolate faults they should be simulated in the AMV. A virus subsystem, V is added to the AMV to generate faults.

The modified CPN learning algorithm was applied to the AMV. The modifications resulted in a significant improvements compared to the basic CPN method. The Table 1. shows some typical h values, where h_{init} denotes the h value after the initial phase (i.e. the result of the basic SOM), h_{sele} denotes the result after the selection phase and h_{fina} the final result of the proposed algorithm

In order to detect a priori unknown faults after the training phase it should be detected whether an input vector belongs to a cluster not recognised in the training phase. Based on the clustering method, we can use the following techniques to give alarm about the possibility of unknown faults:

- the distance of the input vector from the closest codebook vector is greater then a threshold,

- the distance of the input vector from the closest codebook vector shows a tendency towards a fault region.

h_{init}	214	257	201	118	148	328	121	208
h_{sele}	111	141	168	38	133	176	47	173
h_{fina}	96	36	102	36	81	37	47	105

Table 1. Test results

5 Conclusions

In the CPN network random initialisation can cause problems, particularly in higher dimensional space because the weight vectors get stuck in subspaces where the probability of an input data vector is near to zero. One way to prevent this is the convex combination. Weight vectors of the network are equal at the start. The network is fed by fractional amounts of patterns. The inputs are then slowly built up to the full input patterns. The CPN network's performance is the best on tightly clustered inputs in distinct groups.

6 Acknowledgements

Authors would like to thank EU COPERNICUS programme for the support within the framework of the research project "Integration of Quantitative & Qualitative Fault Diagnosis" No. PL-96-4383.

7 References

[brow] Brown, M, Harris C. J: Neurofuzzy Adaptive Modelling and Control, Prentice-Hall, 1994

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

[dalm] Dalmi I, Kovacs L, Lorant I, Terstyanszky G: Adaptive Learning and Neural Networks in Fault Diagnosis
International conference CONTROL'98, 01-04 Sep. 1998, Swansea, United Kingdom,
Proceedings **Vol. 1**. pp. 284-289

[leon] Leonard, Kramer, Ungar: Using radial Basis Functions to Approximate a Function and its Error Bounds
IEEE Transaction on Neural Networks, 1992, pp 624-627

[poli] Policarpou M, Helmicki A: Automated Fault Detection and Accommodation. A Learning System Approach, IEEE Trans. on Systems, Man and Cybernetics, 1995, **Vol. 25**, No. 11, pp. 1447-1458

[sors] Sorsa T., Koivo H. N: Application of Artificial Neural Network in Process Fault Diagnosis., Automatica, **Vol. 29**, pp. 843-849