Proceedings of the 6th WSEAS Int. Conf. on NEURAL NETWORKS, Lisbon, Portugal, June 16-18, 2005 (pp45-49)

Initialized RHPNN for Fault Detection in MEMS

R.Asgary and K.Mohammadi
Department of Electrical Engineering
Iran University of Science & Technology

Abstract:-Micro Electro Mechanical Systems will soon usher in a new technological renaissance. Just as ICs brought the pocket calculator, PC, and video games, MEMS will provide a new set of products and markets. Learn about the state of the art, from inertial sensors to microfluidic devices[1].
Over the last few years, considerable effort has gone into the study of the failure mechanisms and reliability of MEMS. Although still very incomplete, our knowledge of the reliability issues relevant to MEMS is growing. One of the major problems in MEMS production is fault detection. After fault diagnosis, hardware or software methodscan be used to overcome it. Most of MEMS have nonlinear and complex models. So it is difficult or impossible to detect the faults by traditional methods, which are model-based. In this paper an initialized Robust Heteroscedastic Probabilistic Neural Network is used for fault detection in a RF MEMS.

Key-Words: - Fault detection, intelligent method, Neural Networks, MEMS.

1 Introduction

Reliability of Micro Electro Mechanical Systems (MEMS) is a very young and fast-changing field. Fabrication of a MEMS System involves many new tools and methods, including design, testing, packaging and reliability issues. Especially the latter is often only the very last step that is considered in the development of new MEMS. The early phases are dominated by considerations of design, functionality and feasibility; not reliability [2].
One important reason for missing reliability data is that in view of the use of new materials and process, the material data, the know-how on failure modes, the means and the procedures to perform reliability tests and consequent failure analysis are often not present and unknown.
The traditional approaches to fault detection and diagnosis involve the limit checking of some variables or the application of redundant sensors. More advanced methods rely on the spectral analysis of signals emanating from the machinery or on the comparison of the actual plant behavior to that expected on the basis of a mathematical model. The latter approaches include methods which are more deterministically framed and those formulated on more of a statistical basis, and parameter estimation.
In methods based on mathematical models, the models obtained must be linear. To work with non-linear systems, it is necessary to select a point and obtain a linearized model around it.
In MEMS most of the parts are strictly non linear and finding a proper model is difficult or sometimes impossible. So using mathematical model for fault detection in MEMS is not a good idea. The constraints of this kind of model have motivated the development of artificial intelligent approaches.
In this paper, we will use a probabilistic neural network for fault detection in MEMS.

2 Fault detection methods

The work on fault diagnosis in the AI community initially focused on the expert system or knowledge-based approach where heuristics are applied to explicitly associate symptoms with fault hypothesis. The short coming of a pure expert system approach led to the development of model-based approaches based on qualitative models in the form of qualitative differential equations, signed digraphs, qualitative functional and structural models. Other approaches assume the availability of process history based data which are then used to develop neural network approaches [3-6].
NNs mimic intelligence. The learning or training nodes of neural networks is different from that of traditional statistical methods.
The results of comparison between a model based fault detection, MBFD, method and a neural network fault detection and classification method is provided in Table1 [7].

As we can see, both of them have their strengths and weaknesses. In MEMS usually there is not a proper and accurate model. Additionally, our knowledge about faults, their sources and effects is not complete. So usually there are novel or undetermined faults which are not considered in model. As a result neural network is a proper tool for fault detection and classification in MEMS.

Criterion	MBFD	NN
Novel faults	Poor	Fair
Robustness to noise	Fair	Good
Resolution	Fair	Good
Adaptability	Good	Fair
Range of application	Good	Bad

Table1- Comparison of MBFD and NN

Generally speaking, there are four types of neural networks:

-Back propagation Neural Network (BPNN)

-Probabilistic Neural Network (PNN)

-Self-Organizing Mapping (SOM)

-Radial Basis Function Neural Network (RBF)

There are some drawbacks to BPNN and SOM. The BPNN requires a large number of training patterns to let network learn the underlying mapping function. The second problem is that the accuracy of the training patterns should not be a measure of whether a model is good or not. BPNN has a low reliability with novel data.

SOM is known as a topological mapping algorithm, in which patterns with similar characteristics cluster together automatically. Output nodes will thus be ordered by competitive learning. The learning rate and neighbor size of SOM have to be optimally selected by experience, and a SOM net needs a large time to converge.

In this paper Robust Heteroscedastic Probabilistic Neural Network is used for fault detection and classification in MEMS. We use LVQ method as an extra stage to determine initial center values. Results show that centers are arranged well and performance increases.

3 Robust Heteroscedastic Probabilistic Neural Network

A PNN classifies data by estimating the class conditional probability density functions, because the parameter of a PNN cannot be determined analytically. To do this it requires a training phase, followed by a validation phase, before it can be used in a testing phase. A PNN consists of a set of Gaussian kernel functions. The original PNN uses all the training patterns as the centers of the Gaussian kernel functions and assumes a common variance or covariance, which is named homoscedastic PNN. To avoid using a validation data set and to determine analytically the optimal common variance, a maximum likelihood procedure was applied to PNN training. On the other hand, the Gaussian kernel functions of a heteroscedastic PNN are uncorrelated and separate variance parameters are assumed. This type of PNN is more difficult to train using the ML training algorithm because of numerical difficulties. A robust method has been proposed to solve this numerical problem by using the jackknife, a robust statistical method, hence the term ‘robust heteroscedastic probabilistic neural networks’ [8].

The RHPNN is a four layer feedforward neural network based on the Parzen window estimator that realizes the Bayes classifier given by

$$g_{Bayes}(x) = \arg\left(\max_{1 \le j \le k} \{\alpha_j f_j(x)\} \right) \quad (1)$$

Where X is a d-dimensional pattern, $g(x)$ is the class index of x, the a priori probability of class j ($1 \le j \le k$) is α_j and the conditional probability density function of class j is f_j. The object of the RHPNN is to estimate the values of f_j. This is done using a mixture of Gaussian kernel functions.

RHPNN has been shown in Fig.1. In this figure two classes are shown. First class is considered for fault free and the second class for faulty patterns. There is only one fault free kernel because with only one Gaussian function all fault free patterns can be shown. There are many different faults and the distances between them are unknown, so in second class, more than one kernel is considered. The optimum number of kernels in second layer is the minimum that each kernel has at least one faulty pattern. The first layer of the PNN is the input layer. The second layer is divided into K groups of nodes, one group for each class.

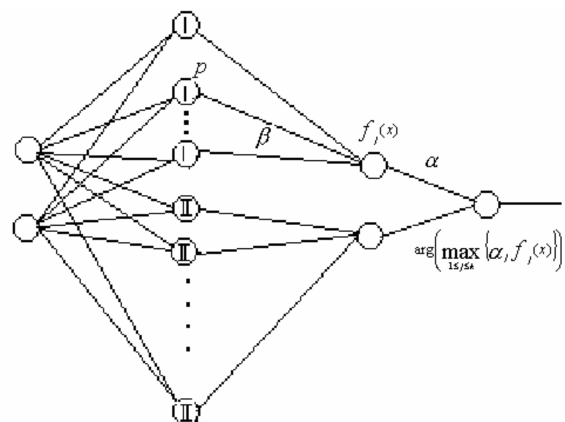

Fig.1- Four layer feed forward RHPNN

The ith kernel node in the jth group is described by a Gaussian function

$$p_{i,j}(x) = \frac{1}{(2\pi\sigma_{i,j}^2)^{d/2}} \exp\left(-\frac{\|x - c_{i,j}\|^2}{2\sigma_{i,j}^2}\right) \quad (2)$$

Where $C_{i,j}$ is the mean vector and $\sigma_{i,j}^2$ is the variance. The third layer has k nodes; each node estimates f_j, using a mixture of Gaussian kernels,

$$f_j(x) = \sum_{i=1}^{Mj} \beta_{i,j} p_{i,j}(x), 1 \le j \le k$$

Where M_j is the number of nodes in the jth group in the second layer; and $\beta_{i,j}$ satisfies

$$\sum_{i=1}^{Mj} \beta_{i,j} = 1, 1 \le j \le k \quad (3)$$

The fourth layer of the PNN makes the decision from (1). The PNN is heteroscedastic when each Gaussian kernel has its own variance. The centers, $C_{i,j}$, the variance, $\sigma_{i,j}^2$ and the mixing coefficients, $\beta_{i,j}$ have to be estimated from the training data. One assumption is

$$\alpha_j = \frac{1}{k}, 1 \le j \le k \quad (4)$$

The EM algorithm has been used to train homoscedastic PNN's. Each iteration of the algorithm consists of an expectation (E) followed by a maximization process (M). This algorithm converges to the ML estimate. For the heteroscedastic PNN, the EM algorithm frequently fails because of numerical difficulties. These problems have been overcome by using a jackknife, which is a robust statistical method.

Suppose the training data is partitioned into k subsets, $\{x_n\}_{n=1}^N = \{\{x_{n,j}\}_{n=1}^{N_j}\}_{j=1}^K$, where $\sum_{j=1}^K N_j$ is the total number of samples and N_j is the number of training samples for class j. The training algorithm is now expressed as follows, where $\tilde{\sigma}_{m,i}^2\Big|^{(k)}$ and $\tilde{C}_{m,i}\Big|^{(k)}$ are the jackknife estimates of the previous values of $\sigma_{m,i}^2$ and $C_{m,i}$, respectively.

Step 1: Compute weights for $1 \le m \le M_i$, $1 \le n \le N_i$ and $1 \le i \le K$.

$$w_{m,i}^{(k)} = \frac{\beta_{m,i} p_{m,i}^{(k)}(x_{n,i})}{\sum_{l=1}^{Mi} \beta_{l,i} p_{l,i}^{(k)}(x_{n,i})} \quad (5)$$

Where

$$p_{l,i}^{(k)}(x_{n,i}) = \frac{1}{(2\pi\tilde{\sigma}_{l,i}^2\big|^{(k)})^{d/2}} \exp\left(-\frac{\left\|x_{n,i} - \tilde{c}_{l,i}\big|^{(k)}\right\|^2}{2\tilde{\sigma}_{l,i}^2\big|^{(k)}}\right) \quad (6)$$

Step 2: Update the parameters for $1 \le m \le M_i$ and $1 \le i \le K$

$$\tilde{c}_{m,i}\Big|^{k+1} = N_i\, c_{m,i}\Big|^{k+1} - \frac{N_i - 1}{N_i} \sum_{j=1}^{Ni} c_{m,i}\Big|_{-j}^{k+1} \quad (7)$$

Where

$$c_{m,i}\Big|^{k+1} = \frac{\sum_{n=1}^{N_i} w_{m,i}^k(x_{n,i})\, x_{n,i}}{\sum_{n=1}^{N_i} w_{m,i}^k(x_{n,i})} \quad (8)$$

$$c_{m,i}\Big|_{-j}^{k+1} = \frac{\sum_{n=1,n\neq j}^{N_i} w_{m,i}^k(x_{n,i})\, x_{n,i}}{\sum_{n=1,n\neq j}^{N_i} w_{m,i}^k(x_{n,i})}, 1 \le j \le N_i \quad (9)$$

Similarly:

$$\tilde{\sigma}_{m,i}^2\Big|^{k+1} = N_i\, \sigma_{m,i}^2\Big|^{k+1} - \frac{N_i - 1}{N_i} \sum_{j=1}^{Ni} \sigma_{m,i}^2\Big|_{-j}^{k+1} \quad (10)$$

Where

$$\sigma_{m,i}^2\Big|^{k+1} = \frac{\sum_{n=1}^{N_i} w_{m,i}^k(x_{n,i}) \left\|x_{n,i} - \tilde{c}_{m,i}\big|^k\right\|^2}{d\sum_{n=1}^{N_i} w_{m,i}^k(x_{n,i})} \quad (11)$$

$$\sigma_{m,i}^2\Big|_{-j}^{k+1} = \frac{\sum_{n=1,n\neq j}^{N_i} w_{m,i}^k(x_{n,i}) \left\|x_{n,i} - \tilde{c}_{m,i}\big|^k\right\|^2}{d\sum_{n=1,n\neq j}^{N_i} w_{m,i}^k(x_{n,i})}, 1 \le j \le N_j \quad (12)$$

and

$$\beta_{m,i}\Big|^{k+1} = \frac{1}{N_i} \sum_{n=1}^{N_i} w_{m,i}^k(x_{n,i}) \quad (13)$$

4 LVQ Algorithm

Most partitional clustering algorithms focus on pursuing optimum partition of input space by iteratively updating the cluster center location, e.g. the popular fuzzy C-means and the K-means algorithms. One major drawback of the partitional clustering method is its sensitivity to the initial prototypes. Often two different choices of initial prototypes may result in quite different clustering results.

Another problem exhibited by many clustering algorithms is how the number of clusters k for a given input data set is determined. In some applications, the k value is known a priori. In other cases it may be reasonable to expect cluster substructures at more than one k value. In this situation, it is necessary to identify the k value that gives the most plausible number of clusters in the data for the analysis at hand.

Training steps for determining voronoi vectors are as follows:
Step1: Each node is associated with two resource centers, α_j(t) and β_j(t). Initially both counters are set to zero. Each input presentation (14) is used to determine the winning node j and update the weight vector thereof.

$$w_j(t+1) = w_j(t) + \varepsilon_j(t)[x(t) - w_j(t)]y_j(t) \qquad (14)$$

$$y_j = \begin{cases} 1 \ , if \quad d_j \le d_i \quad , i \ne j \quad and \quad i = 1,2,...,M. \\ 0 \ , Otherwise \end{cases} \qquad (15)$$

where d_j = distance measure . Clearly only the winning node can update its weight vector. The counters α_j(t) and β_j(t) of the winner are increased by 1 and d_j^2, respectively.
Step2: After λ(t) input representations, only the node with the maximum product resource m_j is allowed to generate a new node.

$$m_j = \alpha_j(t) \times \beta_j(t) \qquad (16)$$

Using (14) and (16), neural network in essence adopts a parallel competition principle. That is, (14) governs the input competition for determining the winning node, while (16) governs the competition for determining the Mother node. After a node generation, the initial resource counters of the son node share half the counters of its Mother node. The initial weight vector of the son node is given by:

$$w_k(t+1) = w_j(t) \times (1 \pm \frac{1}{Q}) \qquad (17)$$

Where Q is a large perturbation constant.
A large Q constant (e.g. 2000) serves to make the weight vector of the new node slightly different from its mother node. The resource counters of all nodes then decreased by a factor of $M(t)/M_f$, where $M(t)$ and M_f denote the current number of nodes and the maximum allowable number of nodes, respectively. Extra formulas can be found in Ref[9].

5 Simulation Results

EM3DS is a MEMS simulator software, which has been used for fault simulation in RF MEMS. 20 faults and one fault free pattern have been simulated in a RF low pass filter MEMS. These 20 faults consist of both digital and analog faults. Changing substrate resistance, magnetic and electric properties, shorts and opens, disconnections, connection between separate parts and some other faults have been simulated by software. The S parameters are calculated and used for training and testing RHPNN neural network. We have used a 2 dimension data as input to neural network. The real and imaginary parts of S_{11} are 2-dimensional input data. The structure of the RF MEMS has been shown in Fig.2.

At first all the patterns in the pool are used to build up a model, which is able to group all the fault free and faulty circuits into n groups. The strategy for selecting the value of n is to ensure each kernel has at least one pattern of a fault free or faulty circuit, falling in it. The optimal n for RF low pass filter is 7. One kernel is belonged to fault free class and the others are belonged to faulty classes. With the RHPNN it is not necessary to define a class label for each faulty pattern, which is a vector containing real and imaginary parts of S_{11} parameter.

For each fault 5 patterns and for fault free, 10 patterns have been simulated, so total number of learning patterns is (20*5+10=110).

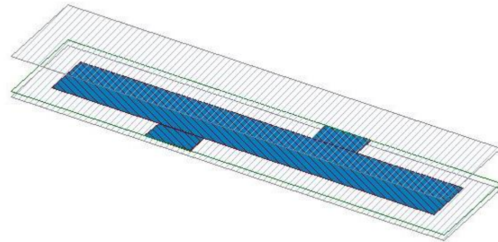

Fig.2. RF low pass filter

All the faulty patterns are labeled with the same number when training a RHPNN model. During training the RHPNN is able to cluster the patterns automatically. This is an advantage compared with some other neural networks.
After training RHPNN, 50 faulty and fault free patterns have been applied to it. Table1 shows the details of misclassification when applying the RHPNN to this example.
Another RHPNN is used for fault detection in the other MEMS. After training, 50 faulty and fault free patterns have been applied to it. Table 3 shows the details.

	Detected as Fault	Detected as Fault free	Correct fault detection percent
40 Faulty Pattern	37	3	%92.5
10 Fault free Pattern	1	9	%90
Total 50			%92

Table2-RF low pass filter fault detection by RHPNN

	Detected as Fault	Detected as Fault free	Correct fault detection percent
40 Faulty Pattern	38	2	%95
10 Fault free Pattern	1	9	%80
Total 50			%94

Table3– RF low pass filter fault detection by initialized RHPNN

5 Conclusion

MEMS have a nonlinear and complex model. Most of the times, there are novel and unknown faults in MEMS too. So, finding a proper model is difficult and model-based fault detection methods can't find the faults properly.

In this paper, we proposed a probabilistic neural network for fault detection in MEMS. LVQ method has been used to convert input data to proper partitions. Then each voronoi vector is used as a kernel center in RHPNN. The results show that using LVQ method for center initialization, prevents local minima in training. In this method RHPNN trains faster and better.

Extra work is needed to define how the initial centers can be used for determining optimum center values and kernel numbers.

References:

[1]Bruno Murari, "Integrated Nanelectronic Components into Electronic Microsystems", *IEEE Trans. on Reliability*, vol.52, No.1, 2003, pp.36-44.

[2] R. Muller, U. Wagner, W. Bernhard, "Reliability of MEMS-a methodical approach", *Proc.11th European symposium on reliability of electron devices, failure physics and analysis*, ,2001, pp.1657-62.

[3]D. Micusik, V. Stopjakova, "Application of Feed Forward Artificial Neural Network to the Identification of Defective Analog Integrated Circuits", *Neural Compute & Application Journal, Springer-verlag*, 2002, pp.71-9.

[4] P. wang, G. Vachtseranos, "Fault Prognosis Using Dynamic wavelet Neural Networks", *Proc. of 39th IEEE Conference on Decision and Control*, Vol.3, 2001, pp.857-870.

[5] S.H. Yang, B.H. Chen, "Neural Network Based Fault Diagnosis Using Unmeasurable Inputs", *Journal of Artificial Intelligence-PERGAMON*, Vol.13, 2000, pp.345-356.

[6] M.A. El-Gamal, "Genetically Evolved Neural Networks for Fault Classification in Analog Circuits", *Journal of Neural Computing & applications, Springer*, Vol.11, 2002, pp.112-121.

[7] R. Rengaswamy, K.E. Arzen, "A Comparison of Model-Based and Neural Network-Based Diagnostic Methods", *Journal of Artificial Intelligence-PERGAMON*, Vol.14, pp.805-818, 2001.

[8] Z. Yang, Zwolinski, "Applying A Robust Heteroscedastic Probabilistic Neural Network to Analog Fault Detection and Classification", *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol.19, No.1, 2000, pp.142-151.

[9] Jung-Hua wang, Jen-Da Rau and Wen-Jeng Liu, "Two Stage Clustering via Neural Network", *IEEE Trans. Neural networks*, Vol.14, No.3, 2003, pp.606-615.