Fault Detection and Classification in Analogue Integrated Circuits using Robust Heteroscedastic Probabilistic Neural Networks

Z. R Yang, M. Zwolinski and C. D Chalk
Department of Electronics and Computer Science
University of Southampton
Southampton SO17 1BJ, UK
zry@ecs.soton.ac.uk

Abstract

A new neural network method has been used to distinguish faulty from fault-free circuit responses. This technique is significantly more accurate than other classification methods. A set of responses can be classified in the order of 1 second.

1. Introduction

There has been much interest in recent years in the testing of analogue integrated circuits and in the development of appropriate testing techniques. Such work has concentrated on the detection of catastrophic faults, such as opens and shorts. Detection of faults has depended on the responses of faulty circuits being *sufficiently* different from the fault-free response. The definition of sufficiency might be arbitrary thresholds of, say, 10% or a difference of at least 6σ, although the assumption of a Gaussian distribution is questionable [Spinks, 1997].

It has been suggested therefore that simple thresholds are insufficient for fault detection. The use of neural networks has been proposed elsewhere. Yu *et al* applied a back propagation neural network to fault diagnosis in a CMOS opamp circuit with gate oxide short faults [Yu, 1994]. A back propagation neural network generally requires a large number of training patterns to let the network learn the underlying mapping function (mapping a data space which contains faulty and fault-free responses to a desired diagnosis space). The diagnosis accuracy of the training patterns reported by Yu *et al* are 67% and 83.3% for ramp and sinusoid test stimulus respectively. Yu's work was further extended for multiple fault diagnosis by [Maidon, 1997]. [Somayajuk, 1996] applied a Kohonen neural network to cluster circuit faults. However, the learning rate and the neighbour size of a Kohonen neural network have to be optimally selected by experience and a Kohonen neural network needs a long time to converge [Bishop, 1996]. Furthermore, it is difficult to determine the boundary on the Kohonen mapping space for diagnosis in practice and a Kohonen neural network is unable to give a quantitative analysis [Yang, 1998].

In this paper, we show that the use of a novel type of neural network can give very accurate fault detection and classification.

In the next section, we describe the structure of the neural network used here. We follow this with an analogue circuit example that compares the use of this neural network technique with probabilistic thresholds and with simpler neural network methods.

2. Robust Heteroscedastic Probabilistic Neural Networks

A probabilistic neural network (PNN) classifies data by estimating the probability density functions (pdfs) of its different classes [Specht 1988, 1990]. Because the variance of the pdfs cannot be determined analytically, a validation phase is required before the testing phase. A PNN consists of a

set of Gaussian distribution functions. A PNN uses all the training patterns as the centres of the Gaussian distribution functions and assumes a common variance or covariance (this is known as a homoscedastic PNN). To avoid using a validation data set and to determine analytically the optimal common variance, a maximum likelihood (ML) procedure was used in PNN training [Streit, 1994]. However, Streit's PNN is still homoscedastic. On the other hand, the Gaussian distribution 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 procedure, because of numerical difficulties [Yang, 1998]. A robust method has been proposed to solve this numerical problem, hence the term "Robust Heteroscedastic Probabilistic Neural Network" (RHPNN).

2.1 The Heteroscedastic PNN

The PNN is a four layer feedforward neural network based on Parzen window estimator [Parzen, 1962] that realises the Bayes classifier given by (1).

$$g_{Bayes}(\mathbf{x}) = \arg\left(\max_{1 \le j \le K} \{\alpha_j f_j(\mathbf{x})\} \right) \tag{1}$$

where $\mathbf{x} \in \Re^d$ is a d-dimensional pattern, $g(\mathbf{x})$ is the class index of $\mathbf{x}$, 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 PNN is to estimate the values of f_j. This is done using a mixture of Gaussian kernel functions.

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. The ith kernel node in the jth group is described by a Gaussian function (2).

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

where $\mathbf{c}_{i,j} \in \Re^d$ 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, from (3).

$$f_j(\mathbf{x}) = \sum_{i=1}^{M_j} \beta_{i,j} p_{i,j}(\mathbf{x}), 1 \le j \le K, \tag{3}$$

where M_j is the number of nodes in the jth group in the second layer; and $\beta_{i,j}$ satisfies (4).

$$\sum_{i=1}^{M_j} \beta_{i,j} = 1, 1 \le j \le K. \tag{4}$$

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

$$\alpha_j = \frac{1}{K}, 1 \le j \le K. \tag{5}$$

2.2 The Robust ML Training Algorithm

The EM algorithm [Dempster, 1977] has been used to train homoscedastic PNNs [Streit, 1994]. Each iteration of the algorithm consists of an expectation process (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 "Jack-Knife" which is a robust statistical method [Miller, 1971].

The training data is partitioned into K subsets.

$$\{\mathbf{x}_n\}_{n=1}^{N} = \left\{\{\mathbf{x}_{n,j}\}_{n=1}^{N_j}\right\}_{j=1}^{K}, \tag{6}$$

where

$$\sum_{j=1}^{K} N_j = N \tag{7}$$

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 $\tilde{\mathbf{c}}_{m,i}\Big|^{(k)}$ are the (Jack-Knife) estimates of the previous values of $\sigma_{m,i}^2$ and $\mathbf{c}_{m,i}$, respectively.

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

$$w_{m,i}^{(k)}(\mathbf{x}_{n,i}) = \frac{\beta_{m,i}\, p_{m,i}^{(k)}(\mathbf{x}_{n,i})}{\sum_{l=1}^{M_i} \beta_{l,i}\, p_{l,i}^{(k)}(\mathbf{x}_{n,i})}, \tag{8}$$

where

$$p_{l,i}^{(k)}(\mathbf{x}_{n,i}) = \frac{1}{\left(2\pi\tilde{\sigma}_{l,i}^2\Big|^{(k)}\right)^{d/2}} \exp\left(-\frac{\left\|\mathbf{x}_{n,i} - \tilde{\mathbf{c}}_{l,i}\Big|^{(k)}\right\|^2}{2\tilde{\sigma}_{l,i}^2\Big|^{(k)}}\right). \tag{9}$$

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

$$\tilde{\mathbf{c}}_{m,i}\Big|^{(k+1)} = N_i\, \mathbf{c}_{m,i}\Big|^{(k+1)} - \frac{N_i - 1}{N_i} \sum_{j=1}^{N_i} \mathbf{c}_{m,i}\Big|_{-j}^{(k+1)} \tag{10}$$

where

$$\mathbf{c}_{m,i}\Big|^{(k+1)} = \frac{\sum_{n=1}^{N_i} w_{m,i}^{(k)}(\mathbf{x}_{n,i})\mathbf{x}_{n,i}}{\sum_{n=1}^{N_i} w_{m,i}^{(k)}(\mathbf{x}_{n,i})} \tag{11}$$

$$\mathbf{c}_{m,i}\Big|_{-j}^{(k+1)} = \frac{\sum_{n=1,n\ne j}^{N_i} w_{m,i}^{(k)}(\mathbf{x}_{n,i})\mathbf{x}_{n,i}}{\sum_{n=1,n\ne j}^{N_i} w_{m,i}^{(k)}(\mathbf{x}_{n,i})}, 1 \le j \le N_i \tag{12}$$

$$\tilde{\sigma}_{m,i}^2\Big|^{(k+1)} = N_i\, \sigma_{m,i}^2\Big|^{(k+1)} - \frac{N_i - 1}{N_i} \sum_{j=1}^{N_i} \sigma_{m,i}^2\Big|_{-j}^{(k+1)} \tag{13}$$

where

$$\sigma_{m,i}^2\Big|^{(k+1)} = \frac{\sum_{n=1}^{N_i} w_{m,i}^{(k)}(\mathbf{x}_{n,i})\left\|\mathbf{x}_{n,i} - \tilde{\mathbf{c}}_{m,i}\Big|^{(k)}\right\|^2}{d\sum_{n=1}^{N_i} w_{m,i}^{(k)}(\mathbf{x}_{n,i})} \tag{14}$$

$$\sigma_{m,i}^2\Big|_{-j}^{(k+1)} = \frac{\sum_{n=1,n\ne j}^{N_i} w_{m,i}^{(k)}(\mathbf{x}_{n,i})\left\|\mathbf{x}_{n,i} - \tilde{\mathbf{c}}_{m,i}\Big|^{(k)}\right\|^2}{d\sum_{n=1,n\ne j}^{N_i} w_{m,i}^{(k)}(\mathbf{x}_{n,i})}, 1 \le j \le N_i \tag{15}$$

and

$$\beta_{m,i}\Big|^{(k+1)} = \frac{1}{N_i} \sum_{n=1}^{N_i} w_{m,i}^{(k)}(\mathbf{x}_{n,i}) \tag{16}$$

3. Experimental results

A folded cascode operational amplifier circuit was used to evaluate the RHPNN. 92 short and open faults were modelled using 1TΩ open circuit (drain open, source open) and 1Ω short circuit (gate-source, gate-drain) transistor fault models. The stimulus was a 0.5V amplitude sinusoid at 300kHz with a DC offset of –3V. The fault-free and each faulty circuit were simulated using nominal parameter values and 30 Monte Carlo simulations were performed on each version of the circuit to model parametric variations due to process changes. MITEL process parameters were used, varying Vt, Tox, mobility, and lateral diffusion. From each simulation, four parameters were derived: the DC voltage at the output; the DC supply current; the RMS value of the AC component of the output voltage and the RMS value of the AC component of the supply current.

3.1 Training the RHPNN

The training procedure was as follows. The results of 15 of the Monte Carlo simulations of the fault-free circuit and 2 of each of the Monte Carlo simulations of each faulty circuit were pooled at random for training. At first, all the patterns in the pool were used to build up a model, able to group all the fault-free and faulty circuits into *n* groups. The strategy for selecting *n* is to ensure each kernel has at least one pattern (of a fault-free or faulty circuit) in it. The optimal *n* was found to be 11.

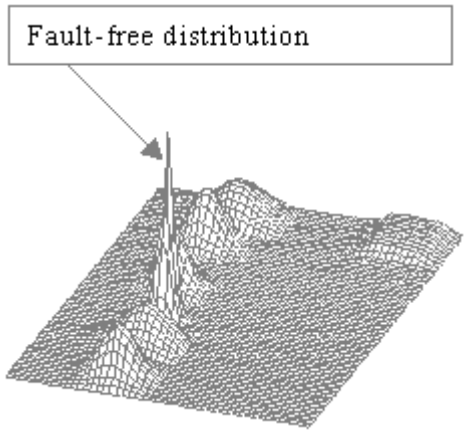

Figure 1 Probability distribution for AC and DC voltages

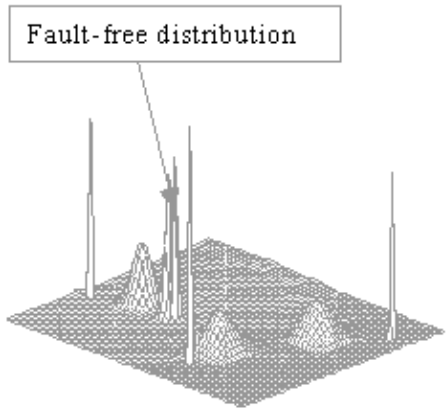

Figure 2 Probability distribution for 4 dimensional case

It should be noted that the RHPNN works in a different way to other neural networks, such as the back propagation neural network [Rumelhart, 1988] or the Kohonen self-organising neural network [Kohonen, 1989]. With the RHPNN, it is not necessary to define a class label for each faulty pattern, which is a vector containing voltages, currents or both corresponding to a faulty circuit. All the faulty patterns are labelled with the same number when training a RHPNN model. During training, the RHPNN is able to cluster the patterns automatically. We know that some faulty patterns are very close to fault-free patterns and some faulty patterns have a large deviation from other faulty patterns. This phenomenon affects the training of a neural network. After the 1st training is completed, it is, therefore worthwhile considering a further training phase to refine the model. The strategy is to re-train the RHPNN for those groups containing more than one faulty pattern. The method is very simple in that the pool for re-training is composed of 15 randomly selected fault-free patterns plus the faulty patterns appearing in the group on which we are focusing.

Figure 1 shows the probability distributions for the various groups, when the AC and DC voltages are considered. Note that the variance of the fault-free group is much smaller than that of the other

groups. Similarly Figure 2 shows the four dimensional probability distribution. Here, it can be seen that the probability distributions of the various groups are very distinct.

3.2 Fault Detection

Table 1 shows the percentage of correct classifications between the faulty and fault-free circuits using the 3σ threshold test, a standard Bayes statistical discrimination method [Chou, 1990], a standard PNN and a RHPNN for each of the parameters measured, individually. Note that the Bayes method misclassifies all the fault responses in the DC voltage test. It can be seen that the RHPNN improves the fault detection significantly compared with other methods. For example, the RHPNN improves the fault detection compared with the 3σ thresholding method to a significance level of 0.1% except for the DC current case.

Table 1 Correct classification of faulty circuits using Threshold, Bayes, PNN and RHPNN methods.

Method	**AC(voltage)**	**AC(current)**	**DC(voltage)**	**DC(current)**
Threshold	49%	61%	55%	58%
BAYES	69%	34%	-	2%
PNN	82%	70%	57%	2%
RHPNN	79%	77%	71%	60%

Table 2 shows how the accuracy of classification may be improved by pairing parameters for the threshold test, a (standard) PNN [Specht, 1988] and a RHPNN. The Bayes method is not applicable here. Similarly, Table 3 shows the fault detection rate when all four parameters are used.

Table 2 Correct classification of faulty circuits in two dimensions using Threshold, PNN and RHPNN

Method	**AC**	**DC**	**Voltage**	**Current**
Threshold	63%	66%	62%	72%
PNN	71%	42%	69%	5%
RHPNN	96%	80%	96%	78%

AC: the data space formed by AC voltages and AC currents
DC: the data space formed by DC voltages and DC currents
Voltage: the data space formed by AC voltages and DC voltages
Current: the data space formed by AC currents and DC currents

Table 3 Correct classification in four dimensions using Threshold, PNN and RHPNN

Method	
Threshold	73%
PNN	72%
RHPNN	98%

3.3 Run Time Behaviour of RHPNN

Table 4 shows the time taken to train the RHPNN and the time taken to apply a test using the RHPNN to the test data. The HSPICE simulation time is not included in these figures. These results were obtained on an UltraSPARC 30 running at 300 MHz. It can be seen that pass/fail decisions on all the circuits can be made in the order of 1 s. Note also that the training time for a new circuit is typically of the order of 1 minute.

Table 4 Training and Testing Time of RHPNN

	Converging steps	Training time (s)	Testing time ($s.10^{-4}$/pattern)
AC voltage	5	7.55	1.69
DC voltage	14	20.59	1.54
AC current	44	62.57	1.96
DC current	5	7.53	1.89
AC	17	35.23	2.08
DC	41	75.49	2.16
Current	23	46.15	2.23
Voltage	26	51.31	2.08
All	20	59.21	6.16

4. Conclusions

The Robust Heteroscedastic Probabilistic Neural Network is an extremely reliable technique for distinguishing good analogue circuits from faulty. By examining the AC and DC voltage and current responses of an opamp stimulated with a single frequency sinusoidal input, correct classification was obtained with an accuracy of 98%. Using other techniques, the best result obtained was around 73%. Further the training and testing times for this technique were extremely small, suggesting that this approach may be suitable for production testing.

Acknowledgments

This work has been supported by EPSRC grant GR/L35829.

References

Bishop, C. M., Svensen, M. and Williams, C. K. I. (1996). GTM: a principled alternative to the self-organizing map, *Working paper at Aston University*, http://www.ncrg.aston.ac.uk.

Chou, Y. (1970). *Statistical Analysis*, Holt Rinehart and Winston Inc.

Dempster, A. P., Laird, N. M. and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm, *Journal of Royal Statistic Society* (B), 1-38.

Kohonen, T. (1989). *Self-organisation and Associative Memory*, 3rd edn. Berlin: Springer-Verlag.

Maidon, Y., Jervis, B. W. and Lesage, S. (1997). "Diagnosis of multifaults in analogue circuits using multilayer perceptrons", IEE Proc. Circuits Devices Systems, **144**, 149-154.

Miller, R. G. (1974). The jackknife - a review, *Biometrika*, **61**, 1-15.

Pazen, E. (1962). On estimation of a probability density function and mode, *Annals of Mathematical Statistics*, **3**, 1065-1076.

Rumelhart, D. E. and McClelland, J. L. (1986). *Parallel distributed processing: exploration in the cognition*, MIT press, Cambridge, MA.

Somayajula, S. S., Sanchez-Sinencio, E. and de Gyvez, J. P. (1996). "Analog fault diagnosis based on ramping power supply current signature clusters", IEEE Trans. On Circuits and Systems-II: Analog and Digital Signal Processing, **43**, 703-712.

Specht, D. (1988). Probabilistic neural networks for classification, mapping, or associative memory, *The International Conference on Neural Networks*, **1**, 525-530.

Specht, D. (1990). Probabilistic neural networks, *Neural Networks*, **3**, 109-118.

Spinks S.J., Chalk C.D., Zwolinski M., and Bell I.M. (1997), "Generation and Verification of Tests for Analogue Circuits Subject to Process Parameter Deviations", Proc. 1997 IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, DFT'97, October 1997, October 20-22, Paris, pp.100-108.

Streit, R. L. and Luginbuhl, T. E. (1994). Maximum likelihood training of probabilistic neural networks, *IEEE Trans. on Neural Networks*, **5**, 764-783.

Yang, Z. R. (1998). *UK Construction Company Failure Prediction: A Robust Heteroscedastic Parzen Window Classifier*, PhD Thesis, University of Portsmouth.

Yu, S., Jervis, B. W., Eckersall, K. R., Bell. I. M., Hall, A. G. and Taylor, G. E. (1994). "Neural network approach to fault diagnosis in CMOS opamps with gate oxide short faults", Electronics Letters, **30**, 695-696.