

# Improving the performance of radial basis function classifiers in condition monitoring and fault diagnosis applications where ‘unknown’ faults may occur

Yuhua Li, Michael J. Pont<sup>1</sup> and N. Barrie Jones  
Control & Instrumentation Research Section,  
Department of Engineering, University of Leicester,  
Leicester, LE1 7RH, United Kingdom

**Abstract:** This paper presents a novel technique which may be used to determine an appropriate threshold for interpreting the outputs of a trained Radial Basis Function (RBF) classifier. Results from two experiments demonstrate that this method can be used to improve the performance of RBF classifiers in practical applications.

**Keywords:** Radial basis function network, condition monitoring, novelty detection, threshold determination

Pre-print of:

Li, Y., Pont, M.J. and Jones, N.B. (2002) “Improving the performance of radial basis function classifiers in condition monitoring and fault diagnosis applications where ‘unknown’ faults may occur”, *Pattern Recognition Letters*, **23**: 569-577.

---

<sup>1</sup> To whom correspondence should be addressed.

# 1. Introduction

Radial Basis Function (RBF) neural networks (Broomhead and Lowe, 1988) have been widely used in classification problems such as speech recognition, medical diagnosis, handwriting recognition, image processing, and fault diagnosis.

In such applications, RBF networks are frequently used with a ‘winner takes all’ (WTA) output rule (Scholkopf et al., 1997). This rule is simple to use, and is often an appropriate solution (Cordella et al., 1995). However, in this paper, we are primarily concerned with condition monitoring and fault diagnosis (CMFD) systems: for such systems, the WTA rule is not always appropriate. Instead, the outputs of the classifier may be interpreted by applying a threshold to the output vector: if the value of an output neuron exceeds the given threshold, then an example of the corresponding class is said to have occurred (e.g. Joshi et al., 1997; Cheon et al., 1993).

There are two reasons why threshold-based output interpretations are appropriate in CMFD systems. The first reason is that a CMFD classifier is often required not only to classify known ‘normal’ and ‘fault’ input vectors, but also to recognise that a particular input is neither ‘normal’, nor a member of one of the existing fault categories: this process is often called novelty detection (Bishop, 1994). By using a threshold-based classifier, outputs may be readily interpreted as an ‘unknown fault’ when none of the ‘normal’ or ‘fault’ output neurons exceeds the threshold. The second reason for using threshold-based output interpretations is that the WTA classifier is unsuitable for use in applications where an input vector may need to be assigned to more than one class: this is often the case in CMFD applications, where multiple faults may occur simultaneously (see, for example, Cheon et al., 1993).

Despite these potential advantages, use of a threshold classifier for CMFD applications can be problematic, because the overall performance of the system depends on the use of an appropriate threshold value: in most published work in this area, thresholds are simply empirically set, usually at values of ‘0.5’ (Watanabe et al., 1994), a process which does not necessarily result in optimal classifier performance (Joshi et al., 1997). To improve the performance of threshold classifiers in CMFD and other application areas, methods for identifying the optimal threshold values are required.

In this paper, we present a novel technique for determining an appropriate threshold for RBF classifiers: this technique is based on an analysis of the relationship between the behaviour of a well-trained RBF classifier and its response to the data set.

The paper is organised as follows: in Section 2, a method for determining a suitable threshold is derived, based on theoretical considerations. Following that, the results of two empirical tests are presented in Section 3. The results are discussed in Section 4.

## 2. Theoretical Considerations

A technique for reliable threshold selection in RBF classifiers will be derived in this section. The problem of threshold selection for neural network classifiers is first formulated in Section 2.1. In Section 2.2, the decision behaviour of RBF classifiers is mathematically and geometrically analysed. The reliable threshold selection method for RBF classifiers is then proposed in Section 2.3.

### 2.1 The general problem

A basic learning problem can be represented by six components (Smolensky et al., 1996):  $X$ ,  $Y$ ,  $A$ ,  $H$ ,  $P$ , and  $L$ . The first four components are the instance(input vector), outcome, decision, and decision rule spaces, respectively.  $X$  is an arbitrary set,  $Y, A \in \{0,1\}$ , and  $H$  is a family of functions from  $X$  into  $A$ . The fifth component,  $P$ , is a family of joint probability distributions of  $Z = X \times Y$ . These represent the possible ‘states of nature’ that might be governing the generation of examples. The last component, the loss function  $L$ , is a mapping from  $Y \times A$  into the real number set  $\Re$ .

This paper mainly concerns the behaviour of  $A$  and  $L$  with respect to the threshold. Assume that the classifier has been well trained with samples,  $\vec{z}(z_1, \dots, z_m)$ , where  $z_i = (x_i, y_i) \in Z$ , drawn independently at random according to some probability distribution  $P \in P$ . After training, the classifier is applied to a set of samples whose class is known. Suppose the outcome of output neurons is  $\hat{y}$ , a threshold is applied to  $\hat{y}$ , then we have the hypothesis  $h \in H$  that specifies the appropriate action  $a \in A$  as:

$$h: X \rightarrow A, \quad a(x, \tau) = \theta(\hat{y} - \tau) \quad (1)$$

where:  $\theta(\delta) = \begin{cases} 0 & \text{if } \delta \leq 0 \\ 1 & \text{if } \delta > 0 \end{cases}$

this gives (1) the following values:

$$a(x, \tau) = \theta(\hat{y} - \tau) = \begin{cases} 0 & \text{if } \hat{y} \leq \tau \\ 1 & \text{if } \hat{y} > \tau \end{cases} \quad (2)$$

$a$  equals one for indicating the class occurred, zero otherwise. We consider the following loss function:

$$L(z, \tau) = L(y, a(x, \tau)) = \begin{cases} 0 & \text{if } a = y \\ 1 & \text{if } a \neq y \end{cases} \quad (3)$$

and the risk function (Scholkopf et al., 1997):

$$I(\tau) = \int L(z, \tau) dP(z) \quad (4)$$

since the probability distribution function  $P(z)$  is unknown but a random, independent sample of pairs

$z_i = (x_i, y_i)$  is given, then we have instead the empirical risk function from (4)

$$I(\tau) = \frac{1}{m} \sum_{i=1}^m L(y_i, a_i) \quad (5)$$

Thus the expected risk of the decision rule (or hypothesis) is just the probability that it predicts incorrectly. Our goal is then to make the risk function approach minimum, by determining an optimum value of the threshold,  $t$  (see Section 2.3).

## 2.2 Behaviour of the RBF classifier

A radial basis function (RBF) network with  $k$  hidden neurons has the form (Bishop, 1995):

$$y(x) = w^T \phi(x) + b \quad (6)$$

where  $\phi$  are basis functions: these can take several forms (see below). The weight coefficients  $w$  combine the basis functions into an output value.  $b$  is a bias term<sup>2</sup>.

The mostly commonly used RBF is the Gaussian basis function:

$$\begin{aligned} y(x) &= \sum_{i=1}^k w_i \phi_i(x) + b \\ &= \sum_{i=1}^k w_i \exp\left(-\frac{\|x - \mu_i\|^2}{\sigma_i^2}\right) + b \end{aligned} \quad (7)$$

---

<sup>2</sup> The significance of this bias is that the desired output values of the classifier have non-zero mean.

Thus in (7)  $\phi_i$  is  $i$ th Gaussian basis function with centre  $\mu_i$  and variance  $\sigma_i^2$ .

Since the receptive fields of radial basis functions in the hidden layer have a localised response behaviour in the input space, its response approaches zero at large radii, i.e.,  $\phi_i \xrightarrow{\|x-\mu_i\| \rightarrow \infty} 0$ .

Considering the most common form of coding scheme for classification using neural networks (Tarassenko and Roberts, 1994), the output value  $y$  is 1 if the input pattern  $x$  belongs to the class and 0 otherwise, that is,  $y \in [0,1]$  on the training data<sup>3</sup>. As an RBF classifier obtains its parameters through training, this gives the following statement about the interval of bias in the output layer:

*For classification using RBFN,  $y = w^T \phi(x) + b$ , if the output is set as  $y \in [0,1]$  on training data, then a well trained RBF classifier satisfy:  $b \in [0,1]$ .*

The validity of the above statement can be proved as follows:

Since  $y \in [0,1]$ , and  $\phi \in (0,1)$ ,  $w$  are finite numbers,

if an input vector is far from the centers of the radial basis functions, we have

$$\forall i, i = 1, \dots, k, \quad \|x - \mu_i\| \rightarrow \infty \Rightarrow \phi_i \rightarrow 0, \quad \text{also } \forall i$$

then from (7) we have  $\lim_{\phi \rightarrow 0} y = b$

assume  $b \notin [0,1]$ , say  $b > 1$ , then  $y = w^T \phi(x) + b \rightarrow b > 1$ , which contradicts the condition of  $y \in [0,1]$ . Thus  $b \leq 1$ . Similarly we can prove that  $b \geq 0$ .

Thus  $b \in [0,1]$   $\delta$

To understand the physical meaning of outcomes and the bias in the output layer, consider a geometric interpretation.

Suppose there is a two-class problem with inputs distributing in two dimensions as in Figure 1 (a). An RBF

---

<sup>3</sup> The condition  $y \in [0,1]$  applies for the **training data only**, using the coding scheme discussed above. Even when an ‘optimally’ trained network is used for classification, the network output may still fall outside the interval  $[0,1]$ . In this case, the classifier still can perform well, when an appropriate threshold is applied to the network output.

classifier with two input and two output neurons is used for this classification problem. After training, the values of output neurons with respect to the input variable  $x_1$  are as in Figure 1 (b). The corresponding output neuron has a high value if the input is within the region of the class, while the other output neuron has a low value. Outcomes of all output neurons will be asymptotic to their bias if the input is out of their class region.

It is notable that the bias value of a trained RBF classifier may lie out the interval  $[0, 1]$  in some implementations. However, such classifiers will perform very poorly, as demonstrated in Sections 3 and 4. In these circumstances the RBF classifier will generally need to be re-trained by adjusting the training parameters (that is, the number and the spread constant of the RBFs).

### 2.3 *Reliable threshold selection for RBF classifiers*

As discussed in Section 2.1, for the problem of classification, our goal is to determine a threshold that tends to minimise the probability of erroneous classification in a given class. To derive the optimal threshold for a radial basis function classifier, the general model (Smolensky et al., 1996) is used. The goal of designing a network is to find that network which provides the most likely explanation of the observed data set. To do this it is necessary to try to maximise the probability:

$$P(N|D) = \frac{P(D|N)P(N)}{P(D)} \quad (8)$$

where  $N$  represents the network (with all of the weights and biases specified),  $D$  represents the observed data set, and  $P(D|N)$  is the probability that the network  $N$  would have produced the observed data  $D$ . Applying the monotonic logarithm transformation (Bishop, 1995) to (8), we have:

$$\ln P(N|D) = \ln P(D|N) + \ln P(N) - \ln P(D) \quad (9)$$

Thus, maximising (9) is equivalent to maximising (8). Since the probability distribution of the data is not dependent on the network,  $\ln P(D)$  will have no contribution to the maximising solution of  $\ln P(N|D)$  and is dropped from (9). The second term of (9) is a representation of the probability of the network itself; that is, it is the a priori probability or a priori constraint on the network: since our method assumes that the classifier has been well trained - and our purpose is to determine an optimal threshold for the trained RBF classifier - we will also drop this term.

The first term represents the probability of the data given the network: that is, it is a measure of how well the

classifier accounts for the data. Therefore the threshold selection problem is equivalent to a requirement to maximise  $\ln P(D|N)$ . Further, if the data are broken into two parts: the output  $y$  and the input  $x$ , then:

$$\begin{aligned}\ln P(D|N) &= \ln P((x, y)|N) \\ &= \ln P(y|x \wedge N) + \ln P(x)\end{aligned}\quad (10)$$

Finally, suppose that the input  $x$  does not depend on the network, the last term of (10) has no effect in maximising  $\ln P(D|N)$ . Therefore, we need only maximise the first term  $\ln P(y|x \wedge N)$ .

For the classification problem, the output vectors,  $a$ , defined in (2), consist of a sequence of 0's and 1's. In this case, we imagine that each element of classifier output,  $a$ , represents the probability that the corresponding element of the desired output  $y$  takes on the value 1. Then the probability of the data given the network, for  $c$  classes problem, is represented by the binomial distribution (Fleming and Nellis, 1994):

$$P(y|x \wedge N) = \prod_{i=1}^c a_i^{y_i} (1 - a_i)^{1-y_i} \quad (11)$$

Thus we need to maximise the log of (11) given by:

$$F = \sum_{i=1}^c (y_i \ln a_i + (1 - y_i) \ln(1 - a_i)) \quad (12)$$

By differentiating (12) with respect to decision action  $a_i$ , we then obtain:

$$\frac{\partial F}{\partial a_i} = \frac{y_i - a_i}{a_i(1 - a_i)} \quad (13)$$

To obtain the stationary points of  $F$ , we set (13) to zero. We then have:

$$y_i - a_i = 0 \quad (14)$$

Since to the left of the above stationary point,  $\frac{\partial F}{\partial a_i}$  is positive and to the right  $\frac{\partial F}{\partial a_i}$  is negative, the stationary point from (14) is the maximum point of  $F$ .

Since for an RBF network:

$$\hat{y}_i(x) = w^T \phi(x) + b_i \quad (15)$$

As discussed in Section 2.2,  $b_i \in [0,1]$ , and the classifier output as in (15) satisfies  $\hat{y}_i \rightarrow b_i$  for an input pattern located far from the training patterns. To satisfy (14), we wish  $a_i$  to coincide with the actual  $y_i$  when  $\mathbf{t}_i$

(assuming, here, that each output node of the RBF network is allowed to have a different threshold) is applied to  $\hat{y}_i$ , as in (2). In other word,  $a_i$  should be 1 if the input pattern is within the training class, and should be 0 if it is out of the training class. Therefore, we must select:

$$\tau_i = b_i \quad (16)$$

The threshold with the value given by (16) decides whether samples are classified into that class. If there is noise or disturbance in the data, we expect that the turning point will be less sensitive to the data set, and we therefore increase (16) by a small amount,  $\varepsilon$ :

$$\tau_i = b_i + \varepsilon \quad (17)$$

The introduction of  $\varepsilon$  is to make the classifier robust to noise and disturbance while having little increase on the misclassification rate.

Finally, if we use a single-threshold RBF classifier, then we obtain:

$$\tau = \max(b) + \varepsilon \quad (18)$$

Here,  $\varepsilon$  is a very small positive constant to make  $\tau$  slightly greater than  $\max(b)$ : of course,  $\tau$  must not exceed 1.

### 3. Experimental Tests

In this section, the threshold determination technique derived in Section 2 is assessed in two experimental studies. The chosen data sets were obtained, firstly, from a mathematical model simulating static fault diagnosis and, secondly, from a non-linear model of a diesel engine cooling system. In both cases, the input variables were normalised to  $[0, 1]$ , and the classifiers were trained using the orthogonal least square algorithm (Chen et al., 1991) in the Matlab Neural Network Toolbox.

Please note that, in addition to determining the threshold value, implementing an effective RBF classifier for a given task involves determining two further important parameters:

- a) the maximum number ( $me$ ) of radial basis functions to use in the hidden layer;
- b) the spread constant ( $sc$ ) of the radial basis function.

For each of the following two experiments, the classifier was trained using a range of possible values for  $me$  and  $sc$ . The trained classifier was then tested both on the (seen) training set and on (unseen) test sets. The classification error rate on each data set is estimated using (5).

The key purpose of each study was to explore the impact of the threshold values. To this end a ‘traditional’ threshold value ( $\tau_0$ ) of 0.5 was used. This was compared with a threshold value ( $\tau_1$ ) determined according to (18). More explicitly,  $\varepsilon$  was selected as:  $\varepsilon = 0.05 \times \max(b)$ , to provide a threshold slightly greater than  $\max(b)$ .

### 3.1 Mathematical Model Data Set

The following model represents a large class of static diagnosis problems and is adapted from that introduced in (Leonard and Kramer, 1990), and described as follows:

$$\mathbf{Y} = \mathbf{Y}_0 + \mathbf{a}\mathbf{p} + \mathbf{v} \quad (19)$$

Here  $\mathbf{Y}$  represents the measurement vector of the plant,  $\mathbf{Y}_0$  represents the plant nominal steady state.

Measurement  $\mathbf{Y}$  is a function of plant physical parameters  $\mathbf{p}$ , and suffers from measurement noise  $\mathbf{v}$ .  $\mathbf{a}$  is a distribution matrix of parameter effects on the measurement vector. Plant faults are caused by deviation of parameters. Here we assume that  $\mathbf{Y}$  has two measurements  $y_1$  and  $y_2$ ,  $\mathbf{p}$  has two parameters  $p_1$  and  $p_2$ , and:

$$\mathbf{a} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

This means that  $p_1$  fault causes  $y_1$  and  $y_2$  to deviate in the same direction, and  $p_2$  fault causes  $y_1$  and  $y_2$  to deviate in opposite directions. The classes were defined as:

$$\begin{aligned} \text{Normal (C}_0\text{): } & |p_1| < 0.05, |p_2| < 0.05 \\ \text{Fault 1 (C}_1\text{): } & |p_1| > 0.05, |p_2| < 0.05 \\ \text{Fault 2 (C}_2\text{): } & |p_2| > 0.05, |p_1| < 0.05 \\ & v_1, v_2 \sim N(0, 0.015) \end{aligned}$$

One set of training data was generated with values of  $p_1$  and  $p_2$  sampled from the normal distribution  $N(0, 0.25)$ .

In total, 600 input/output pairs were generated from Equation (19) and were used for training all networks.

Two additional sets of test data each with 300 input/output pairs were also generated, designated ‘Test Set 1’ and ‘Test Set 2’. These data sets were intended to explore how this approach performs, both in terms of interpolation (Test Set 1) and extrapolation (Test Set 2). Test Set 1 had the same distribution as the training set. Test Set 2 had values distributed over the whole parameter space. For this set, samples within the region of training data were assigned to one of the known classes, all other samples were assumed to belong to unknown faults.

Using the training data set, RBF classifiers were trained by changing the number and the spread constant of

radial basis functions in the hidden layer. Table 1 lists the misclassification rate of trained RBF classifiers. In the table, training data, Test Set 1 and Test Set 2 are represented by  $D_0$ ,  $D_1$  and  $D_2$ , respectively. Row 2 is the maximum number of hidden neurons (me) and the spread constant (sc). Elements in row  $b$  are biases of corresponding output neurons. The following rows are the misclassification rate for the data sets ( $D_0$ ,  $D_1$  and  $D_2$ ) using  $\tau_0=0.5$  and  $\tau_1=\max(b)+\epsilon$ , respectively. As noted above,  $\epsilon$  was set to  $\epsilon = 0.05 \times \max(b)$ .

From the table, it is clear that when using either  $\tau_0$  or  $\tau_1$ , the classifiers provide a very similar misclassification rate on the training set and test set 1. However, for samples out of the training region the classifier using  $\tau_1$  produces a lower misclassification rate.

These results may be readily understood. They arise because the appropriately trained classifier is intended to produce a high output value (close to 1) for samples in the class while a low output value (close to 0) for samples in other classes. Thus there is a large interval for threshold selection. Theoretically one could use any threshold between 0 and 1 for a perfectly-trained classifier which is required only to classify samples within the ‘training’ range.

To further explore the effects of the value of the threshold on the misclassification rate, different thresholds were used for the classifier 5 in Table 1. Figure 2 shows the misclassification rate versus the threshold.

These empirical results confirm the findings in (18). Specifically, as is apparent in Figure 2, the classifier with a threshold slightly larger than  $\max(b)$ , say  $\tau_1$ , produces a minimum misclassification rate for Test 2 while a near minimum misclassification rate for the training set and Test Set 1, and  $\tau_1$  is the turning point of misclassification rate on Test Set 2.

It is frequently a requirement (in CMFD applications) that the classifier will provide us not only with a high performance for known conditions but also good performance in the presence of unknown faults. For this purpose, these empirical results support the use of a classifier with threshold  $\tau_1$ .

### ***3.2 Diesel engine cooling system diagnosis data set***

The second data set used in this study was generated from a non-linear model of the cooling system for a diesel

engine, developed elsewhere in our research group (Twiddle, 1999). The model can simulate various faults, including those considered in this study: ‘fan fault’ (that is, the radiator fan is permanently off), ‘thermostat fault’ (that is, the thermostat is stuck open) and ‘pump fault’ (that is, the coolant pump has 50 percent of the capacity of the normal condition). To detect these faults, this experiment used six measurements: the ambient temperature; the engine block temperature, the coolant temperature at engine block inlet; the coolant temperature at the engine block outlet; the coolant temperature at the radiator outlet; and the engine speed.

Using the model, a training data set ( $D_0$ ) with three states (normal, fan fault, and thermostat fault) was generated. Two different test data sets were also generated. One test set (Test Set 1,  $D_1$ ) has the same three classes as the training data. Another test data set (Test Set 2,  $D_2$ ) has the same three classes plus ‘pump fault’, here pump fault was considered as an unknown fault.

Each data set has 300 samples. In each case, the data sets consisted of equal numbers of samples for each class.

Table 2 lists the misclassification rate of classifiers with different training parameters.

To explore, again, the effects of the value of the threshold on the misclassification rate, different thresholds were used for the classifiers 3 and 5 in Table 2. Figure 3 shows the misclassification rate versus the threshold. Table 3 lists the misclassification rates versus thresholds around  $\max(b)$ . Here  $\epsilon$  in Equation (18) was set to be proportional to  $\max(b)$ , i.e.,  $\epsilon = \lambda \cdot \max(b)$ , where  $\lambda$  is a small coefficient in  $\{-0.1, -0.05, -0.01, 0.0, 0.01, 0.05, 0.1\}$ .

The results show that the classifiers with  $\tau_0$  produce a lower misclassification rate for training data and Test Set 1 than that with  $\tau_1$ , while classifiers with  $\tau_1$  can give a better misclassification rate for Test Set 2.

As demonstrated in Experiment 1, the classifier with  $\tau_1$  gives a near minimum misclassification rate for samples within the range of training data, so  $\tau_1$  can still be used in such applications. However, if we are concerned with the classifier performance in applications with possible unknown classes,  $\tau_1$  is more suitable, at the cost of the risk of some misclassification of known classes.

## 4. Discussion and Conclusions

In this paper, a technique for determining a reliable threshold for RBF classifiers has been derived, and assessed in two empirical studies. The approach is easy to use and is seen to be particularly effective in classification problems where there may be possible new classes or ‘unknown faults’.

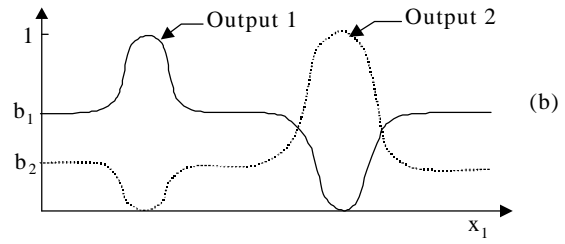
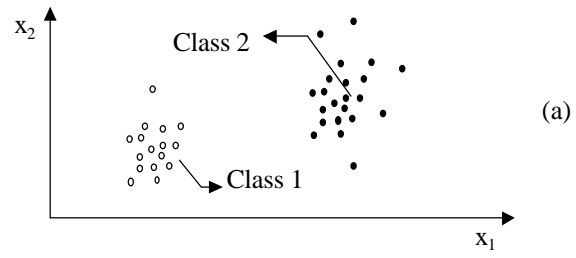
Overall, these results suggest a two-phase approach to RBF classifier use in situations where unknown faults may occur. In the first phase, we deal with the possibility of unknown faults. Since the threshold,  $\tau_1$  (determined by the method described in this paper), results in a measurable improvement of classifier performance for cases with unknown faults,  $\tau_1$  is used in these circumstances. However, as the ability to detect unknown faults **may** be at the expense of a decrease in the classification performance for known classes, we also perform a second phase. In this second phase, the classifier threshold will be modified in order to translate the ‘unknown faults’ into ‘known faults’. In other words, when we have collected sufficient data about unknown faults, the classifier will be re-trained using all available data. A traditional threshold value of  $\tau_0$  may then be applied to the re-trained classifier, since ‘unknown faults’ are less likely to occur.

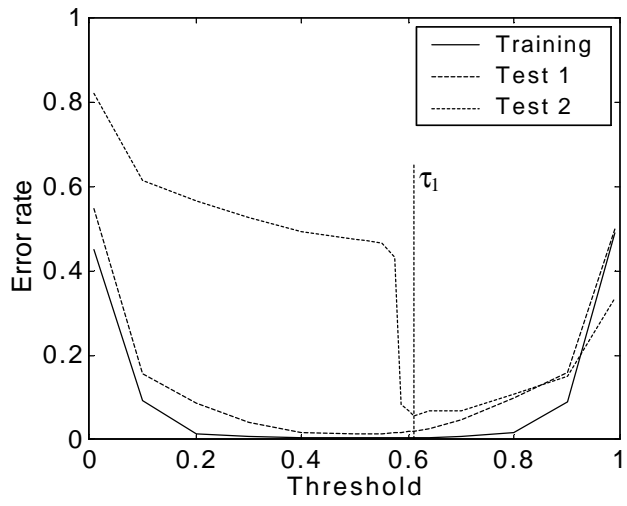
It should be noted that the results in this paper have also demonstrated that the bias,  $b$ , at the output layer of a well trained RBF classifier should satisfy the condition  $b \in [0,1]$ . This result may be used to check whether a particular classifier has trained successfully, and the classifier should be discarded if  $\max(b) > 1$  or  $\min(b) < 0$ . Thus, on this basis, it would be sensible to abandon classifiers numbered 8 and 9 in Table 1.

Finally, it should also be noted that care must be exercised when accepting the training if  $\max(b)$  is close to 1, because the classifier will leave little margin for response to previously unseen samples. An ‘ideal’ RBF classifier will therefore have a low misclassification rate, and a small value for  $\max(b)$ .

## References

- Bishop, C.M., 1994. Novelty detection and neural-network validation, IEE Proceedings-Vision Image and Signal Processing 141, 217-222
- Bishop, C.M., 1995. Neural Networks for Pattern Recognition. Clarendon Press, Oxford.
- Broomhead, D.S., D. Lowe. 1988. Multivariable function interpolation and adaptive networks. Complex Systems 2, 321-335.
- Chen, S., F.N. Cowan, P.M. Grant. 1991. Orthogonal least squares learning algorithm for radial basis function networks. IEEE Transactions on Neural Networks 2, 302-309
- Cheon, S.W., S.H. Chang, H.Y. Chung, Z.N. Bien. 1993. Application of neural networks to multiple alarm processing and diagnosis in nuclear-power-plants. IEEE Transactions on Nuclear Science 40, 11-20
- Cordella, L.P., C. Destefano, F. Tortorella, M. Vento. 1995. A method for improving classification reliability of multilayer perceptrons. IEEE Transactions on Neural Networks 6, 1140-1147
- Fleming, M.C., J.G. Nellis. 1994. Principles of Applied Statistics. Routledge.
- Joshi, A., N. Ramakrishnan, E.N. Houstis, J.R. Rice. 1997. On neurobiological, neuro-fuzzy, machine learning, and statistical pattern recognition techniques. IEEE Transactions on Neural Networks 8, 18-31.
- Leonard, J.A., M.A. Kramer. 1990. Classifying process behaviour with neural networks: strategies for improved training and generalization. Proceedings of the American Control Conference, San Diego, CA, USA, 2478-2483.
- Scholkopf, B., K.K. Sung, C.J.C. Burges, F. Girosi, P. Niyogi, T. Poggio, V. Vapnik. 1997. Comparing support vector machines with Gaussian kernels to radial basis function classifiers. IEEE Transactions on Signal Processing 45, 2758-2765.
- Smolensky, P., M. C. Mozer, D. E. Rumelhart. 1996. Mathematical Perspectives on Neural Networks. Lawrence Erlbaum Associates, Publishers, Mahwah, New Jersey.
- Tarassenko, L., S. Roberts 1994. Supervised and unsupervised learning in radial basis function classifiers. IEE Proceedings-Vision Image and Signal Processing 141, 210-216.
- Twiddle, J.A., 1999. Fuzzy model based fault diagnosis of a diesel engine cooling system. Department of Engineering, University of Leicester, Report 99-1.
- Watanabe, K., S. Hirota, L. Hou, D.M. Himmelblau. 1994. Diagnosis of multiple simultaneous fault via hierarchical artificial neural networks. AIChE Journal 40, 839-848.





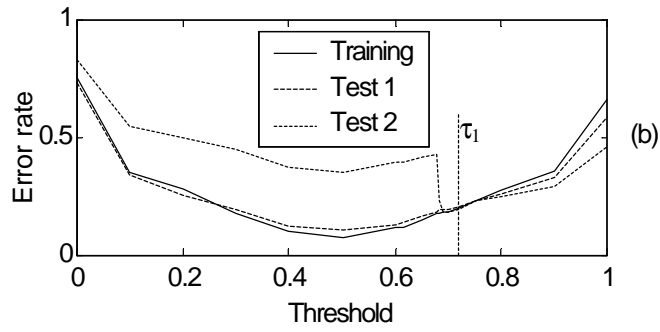
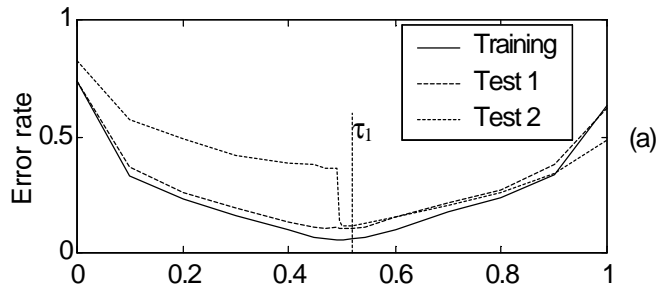


Figure 1 Outcomes of output neurons

Figure 2 Misclassification rate vs. Threshold (classifier 5)

Figure 3 Misclassification rate vs. threshold

(a) for classifier 3 and (b) for classifier 5

Table 1 Classification error rate (%) for the mathematical model data

Classifier Number	1	2	3	4	5	6	7	8	9
me/sc	50/0.025	50/0.03	50/0.1	100/0.025	100/0.04	100/0.06	100/0.075	100/0.1	100/0.2
	0.0224	0.1230	0.0065	0.0629	0.0154	0.0283	0.0100	-0.0058	-0.4157
b	0.5684	0.4196	0.3969	0.5021	0.5801	0.5832	0.3828	0.3922	0.6168
	0.4092	0.4574	0.5966	0.4350	0.4045	0.3885	0.6072	0.6137	0.7990
$D_0$ with 0.5	4.67	5.17	0.5	0.5	0.33	0.33	0.33	0.33	0.5
$D_0$ with $\tau_1$	11.83	4.17	1.17	0.67	0.5	0.33	0.5	0.5	6.5
$D_1$ with 0.5	9.67	9.0	1.67	2.67	1.33	1.0	1.33	2.0	3.67
$D_1$ with $\tau_1$	23.67	8.0	2.67	4.67	2.0	1.33	2.67	3.0	15.67
$D_2$ with 0.5	53	9.33	45.33	48.33	47.67	47.0	47.67	47.33	52.0
$D_2$ with $\tau_1$	14.67	8.67	22.33	7.67	5.67	9.67	15.33	27.33	54.33

Table 2 Classification error rate (%) for cooling system diagnosis

Classifier Number	1	2	3	4	5	6
me/sc	100/0.15	100/0.2	100/0.25	100/0.26	100/0.27	100/0.3
	0.3080	0.1825	0.0898	0.1583	0.1341	0.0527
b	0.3451	0.4873	0.4941	0.6440	0.6847	0.7573
	0.3469	0.3302	0.4161	0.1978	0.1812	0.1900
$D_0$ with 0.5	15.67	8.33	5.33	5.0	7.67	5.33
$D_0$ with $\tau_1$	18.67	8.67	6.00	12.33	20.0	22.33
$D_1$ with 0.5	21.67	13.0	10.33	11.33	11.0	10.33
$D_1$ with $\tau_1$	22.33	13.67	10.33	18.0	20.67	21.67
$D_2$ with 0.5	20.0	12.67	12.0	35.67	35.33	35.33
$D_2$ with $\tau_1$	18.33	12.33	11.67	17.0	19.67	20.67

Table 3 Classification error rate (%) using threshold around  $\max(b)$   
 $\tau = \max(b) + \varepsilon = \max(b) + \lambda \cdot \max(b)$

$\lambda$	Classifier 3			Classifier 5		
	$D_0$	$D_1$	$D_2$	$D_0$	$D_1$	$D_2$
-0.1	7.0	11.33	37.67	11.67	13.67	39.67
-0.05	6.33	10.33	36.33	15.0	16.33	42.0
-0.01	5.33	11.0	36.33	18.0	18.67	43.33
0	5.33	10.67	13.67	18.33	19.67	23.33
0.01	5.33	10.33	12.0	18.67	19.67	19.0
0.05	6.33	10.33	11.67	20.0	20.67	19.67
0.1	7.0	11.33	12.33	23.33	23.33	23.67